The PHP Company

User Guide
Zend Studio 9.x

By Zend Technologies, Inc.

Zend Studio
The Professional PHP
Development Environment

A Complete Solution for PHP Application Development

Disclaimer

The information in this document is subject to change without notice and does not represent a commitment on the part of Zend Technologies Ltd. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the written permission of Zend Technologies Ltd.

All trademarks mentioned in this document, belong to their respective owners.

Zend Studio 9.0 User Guide issued November 2011.

Product Version: 9.x

DN: ZS-UG-141111-09-01

Table of Contents

What's New in Zend Studio 9.x

The following new features are available in Zend Studio 9.0:

- [New Customization Wizard](#)
- [New Project Wizards](#)
- [Amazon AWS Toolkit for Zend Studio](#)
- [Application Deployment Support](#)

Note:

To see What's New in previous versions of Zend Studio go to the [Zend Studio What's New](#) section of the Zend website.

Tutorials

This Tutorials section contains short tutorials on popular tasks that can be performed with Zend Studio. The aim of these tutorials is to help you start developing as fast as possible by getting acquainted with the new features in Zend Studio. Each tutorial covers workflow issues from A-Z describing the processes and workflow that should be followed in order to complete the tasks.

Get up and running with one of these tutorials:

[Customizing and Registering Zend Studio](#)
[Running Applications on a Local Zend Server](#)
[Deploying Applications to Zend Server](#)
[A Guided Tour to Zend Application Support](#)
[Building Applications on the Zend Developer Cloud](#)
[Tutorial Code](#)

Deploying Applications to Zend Server

Zend Studio allows you to create a new PHP project from an existing SVN project. In this tutorial you will be provided with an existing SVN project.

In this Tutorial you will Learn

- To create an API Key. The API Key allows Zend Studio to access your server.
- To create a new PHP project from SVN in Zend Studio. You will be provided with an existing SVN account and project.
- To deploy your project on a remote Zend Server.
- To launch your application. After creating your application, Zend Studio enables you to run your application from within the IDE and in an internal browser.

Prerequisites

- Download Zend Server 5.5 or above from the [Zend Server Downloads page](#) . For more information see the [Zend Server Installation Guide](#).
  For this tutorial your server can be an:
  - Accessible remote Zend server.
  - Local Zend Server (In this tutorial, if you are using a local server it will be accessed as a remote server).
- Download Zend Studio 9.0 or above from the [Zend Studio Downloads page](#). For more information see the [Zend Studio Installation Guide](#).
- For Windows 7 users- Full write permissions for the 'C:\Program Files(x86)\Zend\Apache2\htdocs' directory.

Step 1: Creating an API Key

This procedure describes how to add an API key, which you will need to be able to use Zend Server's API.

To add an API key:

1. In Zend Server, go to **Administration | API Keys**.
2. Go to the Add API Key area, enter a name for your API key and select a full access level from the Access Level dropdown list.

3. Click **Add**.

   Your API key now appears in the list of API keys.
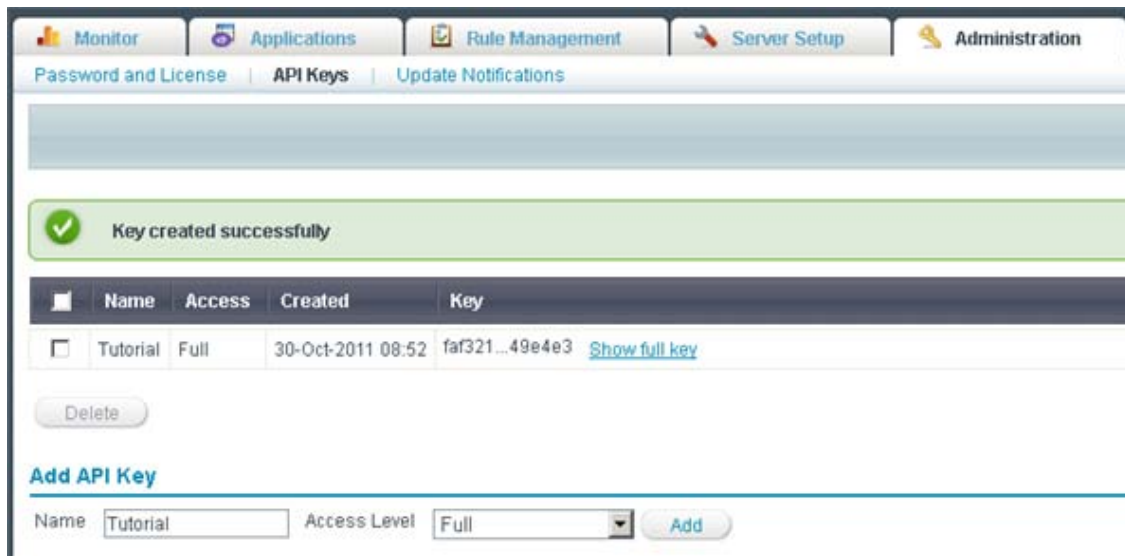
For information on deleting an API key see [Deleting an API Key](#).

Step 2: Creating a New Project from SVN

Creating a new project from SVN in Zend Studio allows you to import an existing SVN project into your IDE.

For this tutorial you will use a provided SVN project.

To create a new project from SVN:

1. Open Zend Studio.

2. Go to **File | New** | **PHP Project from SVN**.

   If the SVN plugin is not installed, Zend Studio will automatically begin the process for installing the plugin that will include restarting the program. Wait until this installation is complete before continuing. See Customizing and Registering Zend Studio for more information.
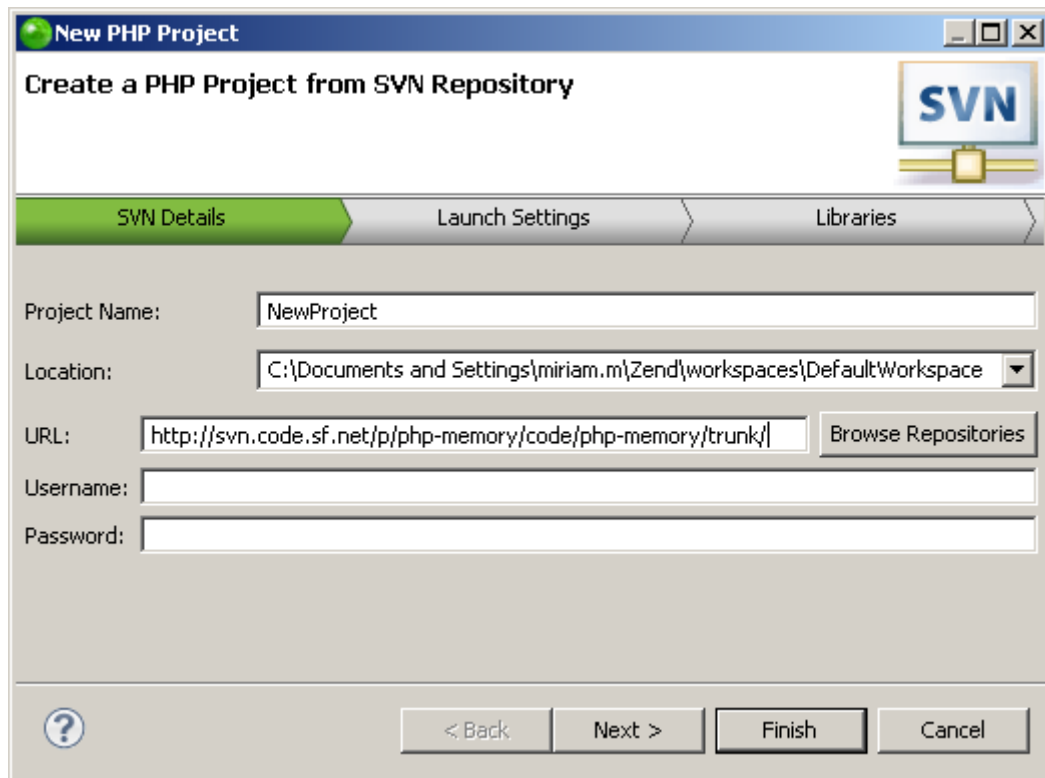
3. Enter the following information:

- Project Name- Name your project "NewProject"

- Location- The project location for this tutorial is 'C:\Program Files(x86)\Zend\Apache2\htdocs'

- URL- Enter the following URL: http://svn.code.sf.net/p/php-memory/code/php-memory/trunk/

- User Name- In this tutorial, leave this field empty.

- Password- In this tutorial, leave this field empty.

4. Click **Next** to move to the Launch Settings dialog.
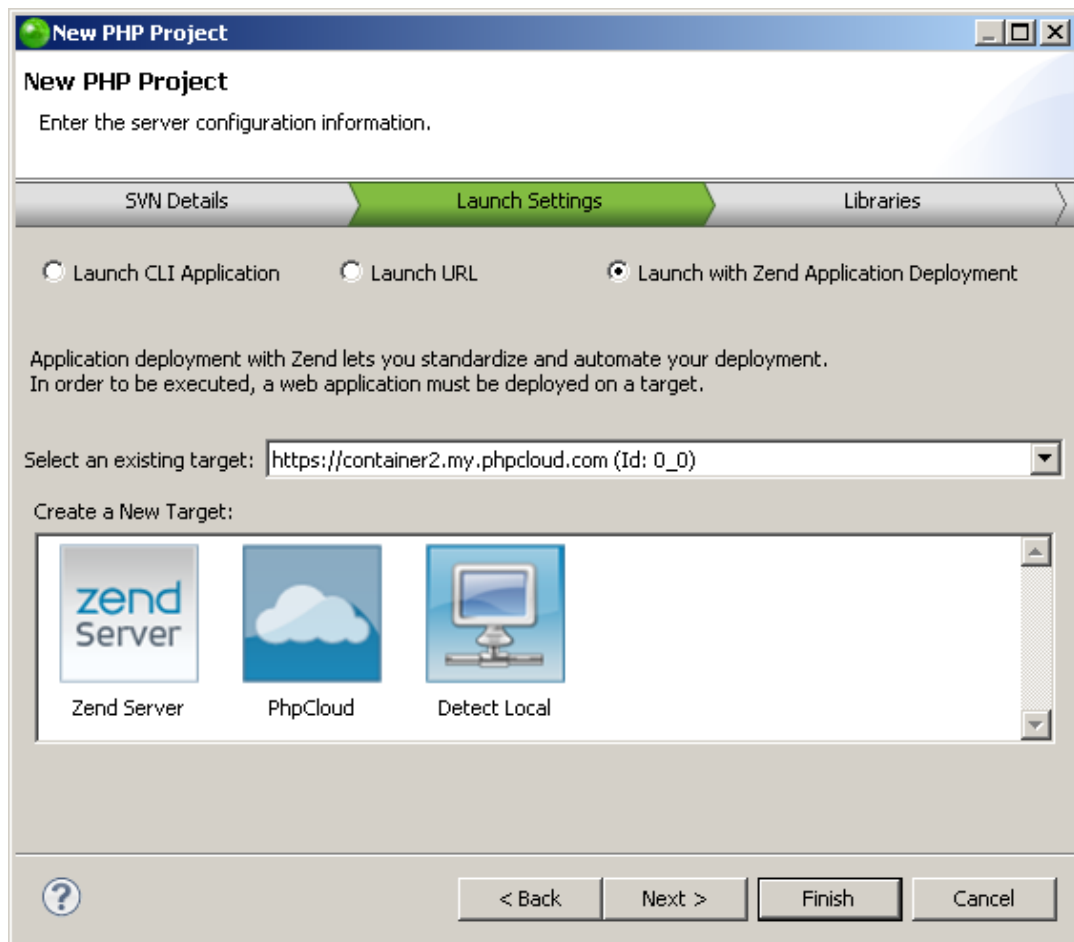
5. Select **Launch with Zend Application Deployment** and click the Zend Server icon

The Target Details dialog is displayed.

6. Enter the server Host Name (address), Key Name and Key Secret and click **Test Connection**.

7. If no problems arise, click **Finish**.

The Launch Settings dialog is displayed again with the new target defined.
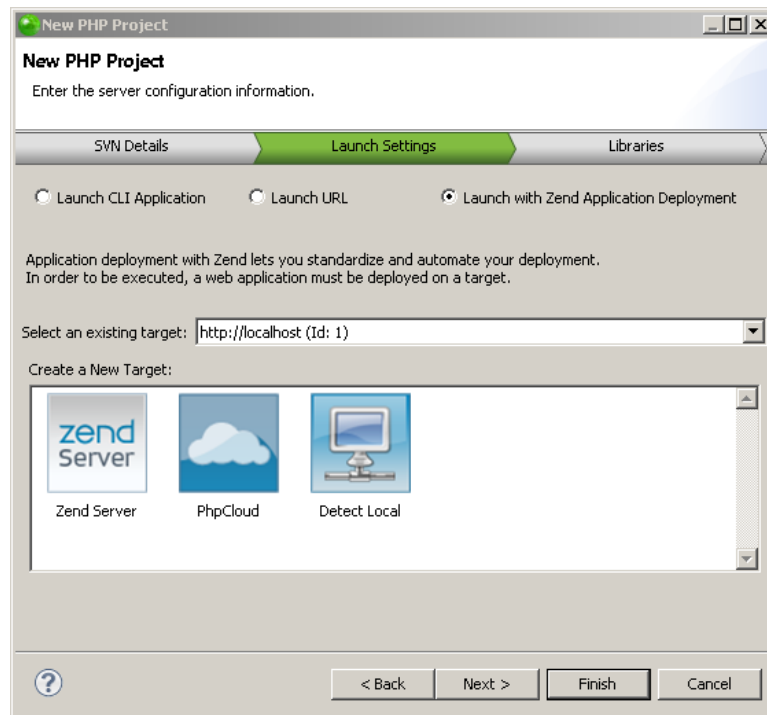
8. Click **Finish.**

Your application is created and displayed in the main view and PHP Explorer. By default, the deployment.xml file is opened and displayed.
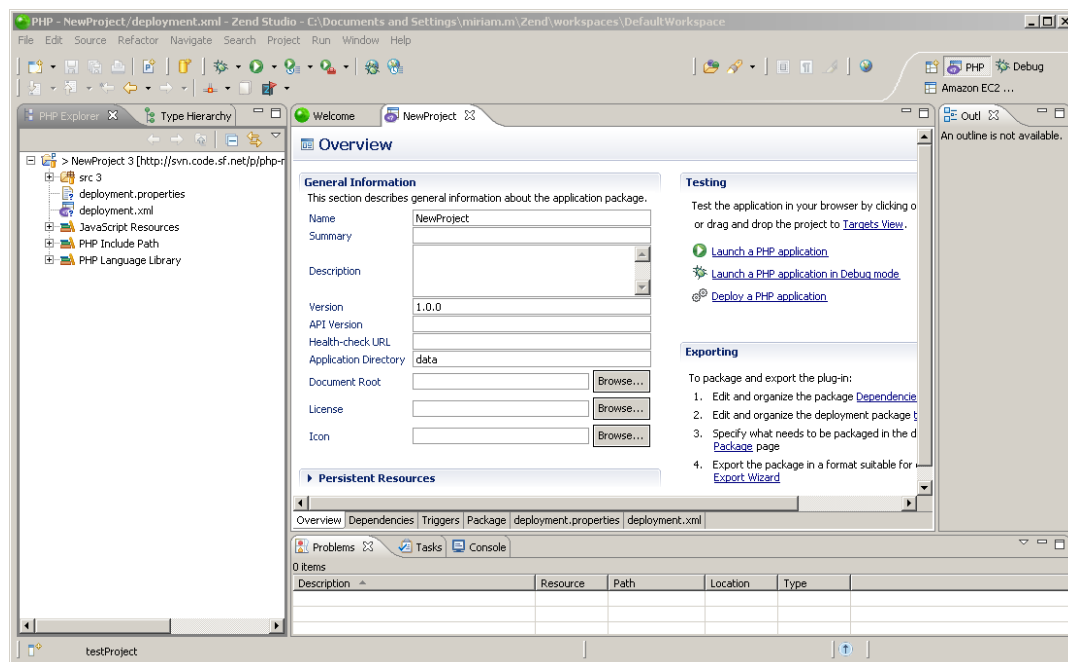
Step 3: Launching your Application

The final stage of this tutorial will explain how to finalize the testing stages for your application by launching it in Zend Studio's internal browser.

To launch your application:

1. In the deployment.xml file's overview tab, click **Browse** next to the Document Root field, and define the root as **data | src | pub**.

2. In the Testing Area, click **Launch a PHP Application**.

    Your application will appear in the internal Web browser.
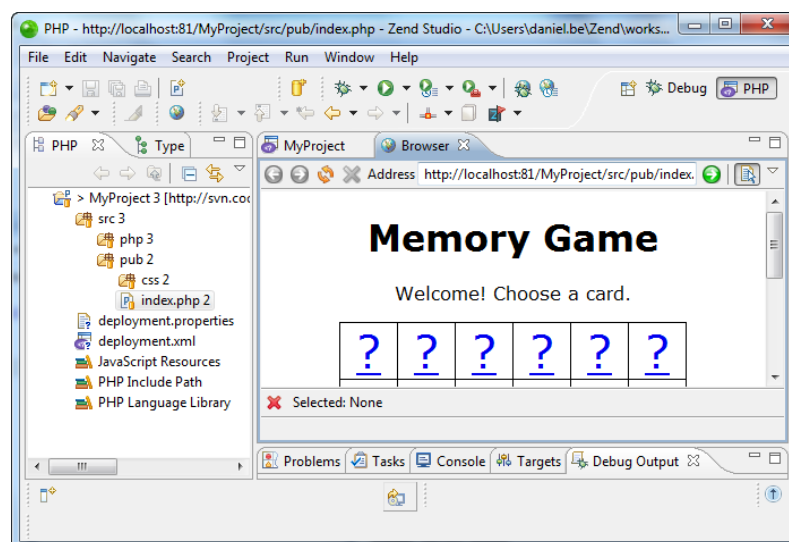
Building Applications on the Zend Developer Cloud

Zend Studio and the Zend Developer Cloud support an integration platform with GitHub projects. This allows you to use the features of GitHub, Zend Studio, and the Zend Developer Cloud for the same application. You can now use the cloud to deploy, debug and modify your GitHub project from Zend Studio.

In this Tutorial you will Learn

- To create GitHub and Zend Developer Cloud accounts. You will need these accounts to maintain a project in GitHub and deploy on Zend's cloud platform.
- To create a new container in the Zend Developer Cloud. The container holds your application in the cloud.
- To create a new project from GitHub. You will be using an existing GitHub project to create a new project in Zend Studio.
- To launch the application on the Zend Developer Cloud and debugging it in Zend Studio.

Prerequisites

- An installed Zend Studio 9.0 or above. To install Zend Studio go to http://www.zend.com/en/products/studio/downloads.

Step 1: Creating an Account for Github and the Zend Developer Cloud

In order to use GitHub and the Zend Developer Cloud, you must first set up valid accounts for both. If you already have these accounts, skip to Step 2.

Creating a GitHub Account

To create a GitHub account:

1. Open the GitHub website at https://github.com/.
2. Click **Plans, Pricing and Signup** in the center of the welcome page.
   The Plans and Pricing page appears.
3. Click **Create a free account**.
   The Signup for GitHub page appears.
4. Enter your account details in the requested fields, and click **Create an account**.
   Your new account page appears.
5. Follow the instructions on the GitHub homepage and create a project and repository.

For more information on creating a GitHub project go to the http://github.help.com

Creating a Zend Developer Cloud Account

To create a Zend Developer Cloud account:

1. Open the Zend Developer Cloud website at https://my.phpcloud.com/.

   Currently, the Zend Developer Cloud platform is open for early access by invitation only. If you have been invited to test-drive the Zend Developer Cloud, log in with your zend.com account credentials and skip the following steps.

2. Click **I do not have an account**.

   The Create a new account page appears.

3. Enter your account details in the requested fields, and click **Register**.

   The Thank you for creating a Zend account page appears.

4. Please answer the questions that appear on the page and click **Finish**.

   The Log In page appears.

5. Enter your account details, and click **Log In**.

   The Developer Cloud Registration page appears.

6. Type your invitation key, and click **I accept the terms, sign me up!** If you do not ahbe an invitation key, click **I don't have an invite key, give me one!**

   The Zend Developer Cloud welcome page appears.

7. Click **Start now** to begin working!

Save your Zend Developer Cloud credentials as you will need them later in this workflow.

Step 2: Developing a GitHub project with Zend Studio and Zend Developer Cloud

Once you have an existing GitHub and Zend Developer Cloud account, you can use the integration features embedded in Zend Studio and Zend Developer Cloud. You can develop, test and deploy a GitHub application on the cloud through Zend Studio.

Before creating the new project in Zend Studio, you will need to create a new container to hold the application in the cloud.

To create a new container in the Zend Developer Cloud:

1. Go to the Zend Developer Cloud homepage , open your account and click **Start Now** on the welcome page.

   The Create Container page appears.

2. Enter the container name and password in the relevant fields, and click **Create Container.**

   The SSH Keypair created window appears.

3. Click **Download private key in PEM format**.

Save the key on your machine as you will need to browse to it when connecting Zend Studio to your server. Your container has been created and will appear under **My Containers** on the homepage.

Step 3: Creating a New Project from GitHub in Zend Studio

You are now ready to integrate Zend Studio with GitHub and the Zend Developer Cloud. You will begin with creating a new project from GitHub in Zend Studio.

To create a new project from GitHub in Zend Studio:

1. Go to **File | New | PHP Project from GitHub.**

The Create Project from GitHub dialog appears.

If the GitHub plugin is not installed, Zend Studio will automatically begin the process for installing the plugin that will include restarting the program. Wait until this installation is complete before continuing. For more information see <u>Customizing and Registering Zend Studio</u>.

2. Enter the following details:

- Project Name: *helloworld*
- User: *gadigold*

3. Click **Refresh.**

   Your application URL will appear in the Location URL field.

4. Click **Next** to move to the Launch Settings dialog.
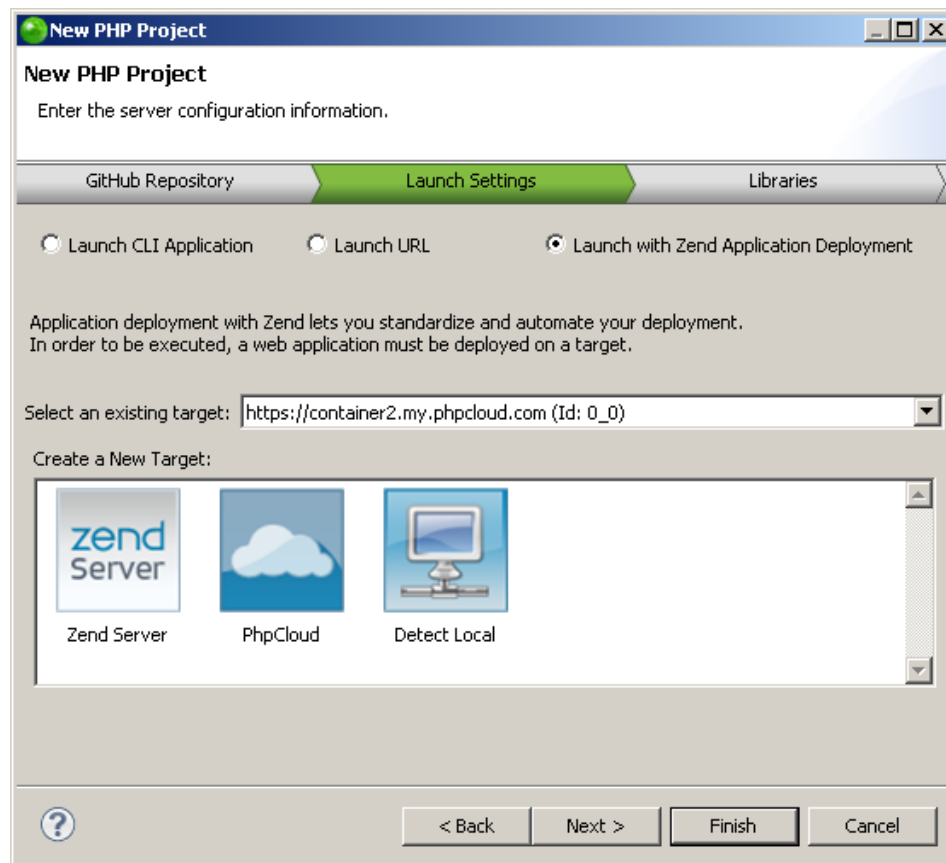
5. Select **Zend Application Deployment** and create a new target by clicking on the

    PHPCloud icon

    The Add Target dialog box appears.

6. Enter your Zend account credentials and browse to your PEM key in the relevant fields, and click **Test Connection**.

7. If no errors appear, click **Finish.**

    The Launch Settings section reappears with the target you added in the **Select an existing target** field.

8. Select the new target from the dropdown list, and click **Finish.**

Your application opens in the Zend Studio workspace.
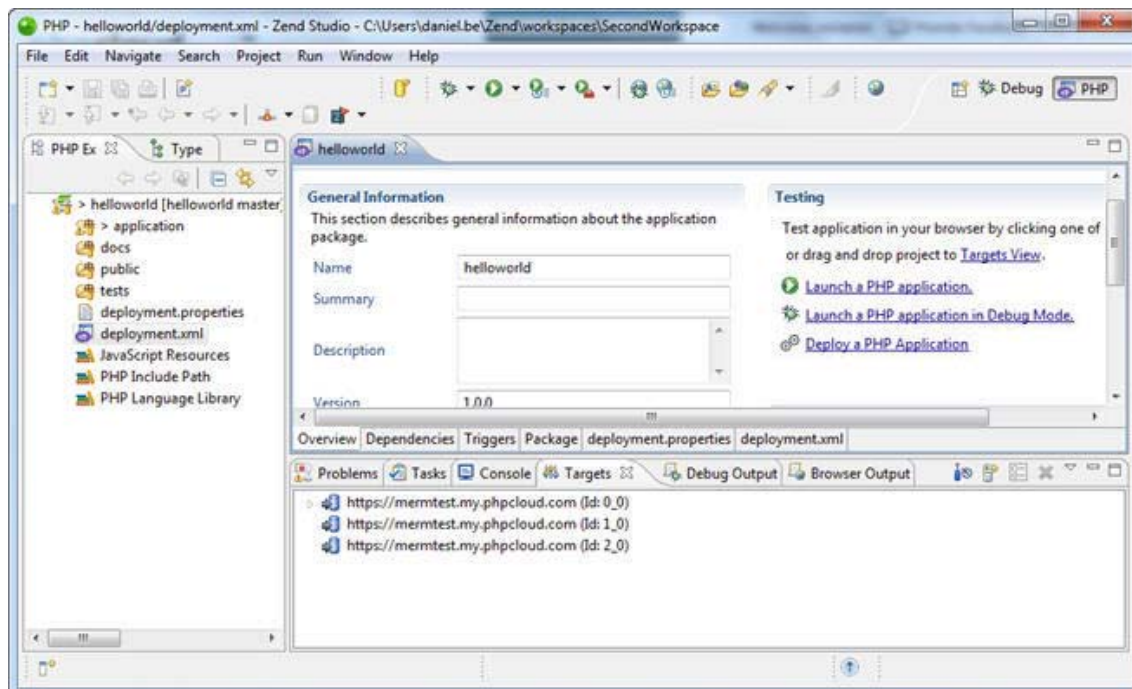
Step 4: Launching Your Application on the Zend Developer Cloud

The next step in this tutorial will explain how to launch the application on the Zend Developer Cloud.

To launch your application on the Zend Developer Cloud:

1. After creating your project, the default file, deployment.xml, opens in the Overview tab.
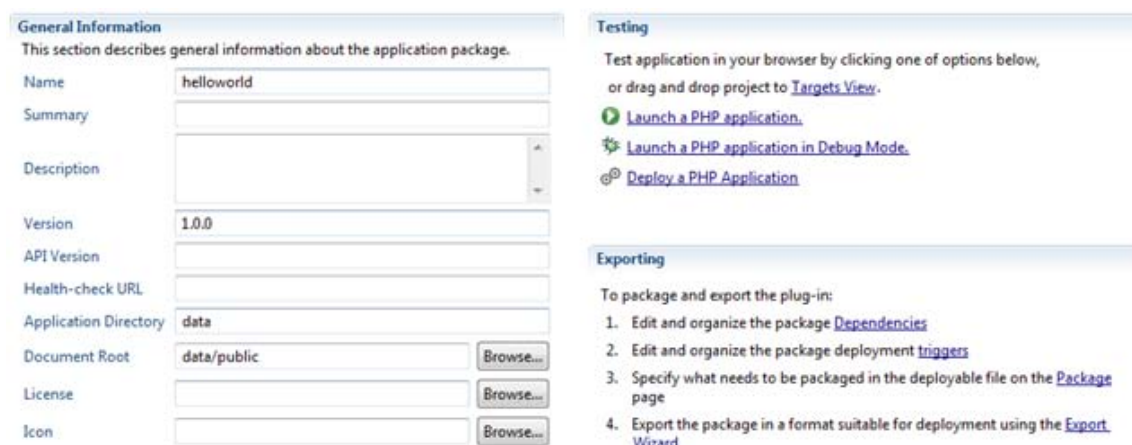
2. In the **Testing** area, click **Launch a PHP Application.**

The Launch PHP Application dialog opens with the default container you defined during the project creation selected, and your application's URL.
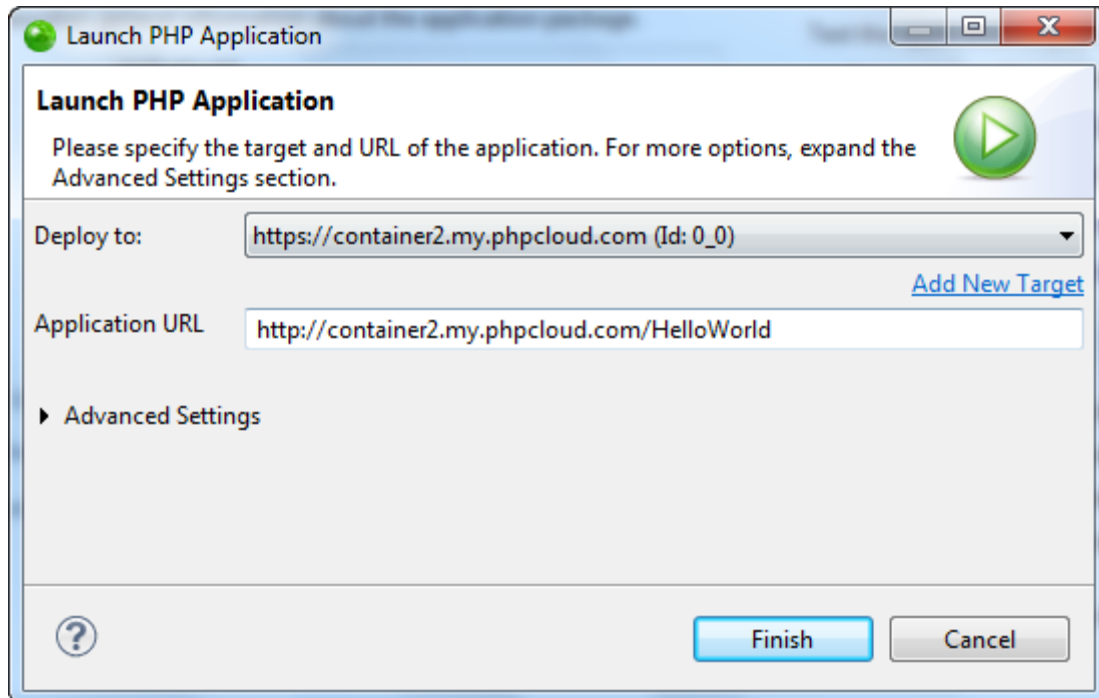
3. Click **Finish** and **Yes** when prompted about your container's authenticity.
4. Your application appears in your container's page in the Zend Developer Cloud website, and opens in Zend Studio's internal browser.

Note:

After deployment, Zend Studio connects to Zend Developer Cloud via the .PEM file using SFTP. Any changes that you make in your application will from now on be automatically updated on the cloud.

Step 5: Debugging Your Application in Zend Studio

You will now learn how to debug the application.

To debug your application in Zend Studio:

1. In the Testing area, click **Launch a PHP Application in Debug Mode**.

   Your application now appears in the Debug perspective and stops at the first breakpoint.
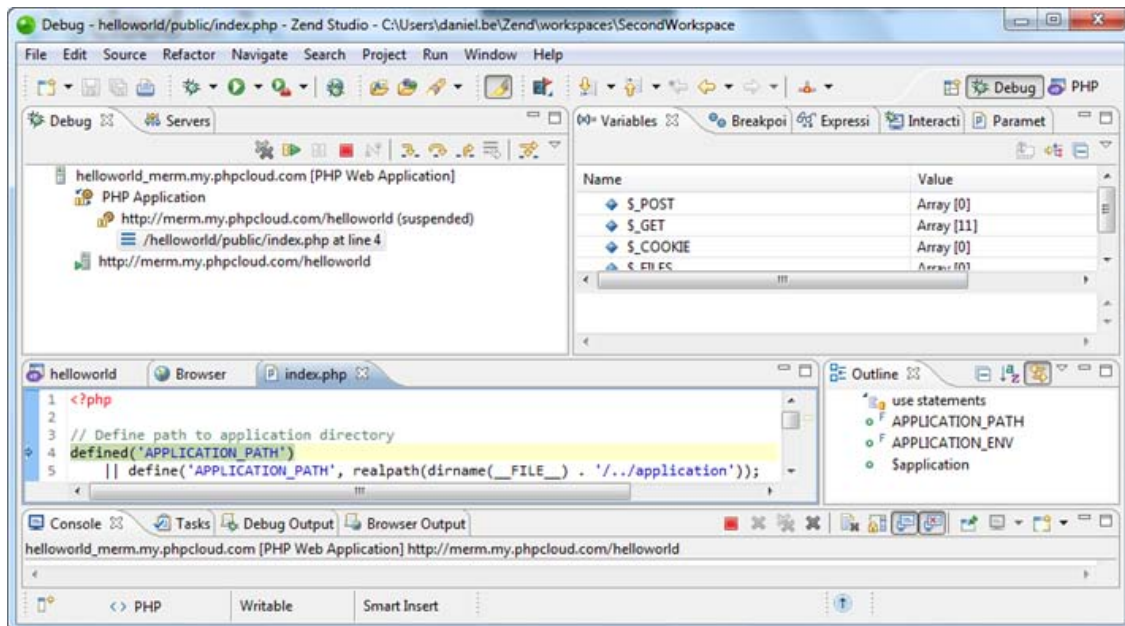
2. To continue the debug click .

Running Applications on a Local Zend Server

This tutorial will explain how to locally create and launch a new project on your machine.

You will not be using deployment as part of the workflow.

If you wish to work with deployment, see Deploying Applications to Zend Server and Building Applications on the Zend Developer Cloud.

In this Tutorial you will Learn

- To create a new Zend Framework project under the local server document root. For more information on Zend Framework go to http://framework.zend.com/
- To debug your application and modify code.
- To launch your application.

Prerequisites

- Download Zend Server 5.5 or above from the Zend Server Downloads page . For more information see the Zend Server Installation Guide.
- Download Zend Studio 9.0 or above from the Zend Studio Downloads page. For more information see the Zend Studio Installation Guide.
- For Windows 7 users, full write permissions for the *'C:\Program Files(x86)\Zend\Apache2\htdocs'* directory.

Step 1: Creating a New Local Zend Framework Project

You will now create a new Zend Framework local project to work with in this tutorial.

To create a new local Zend Framework project:

1. Open Zend Studio.
2. Go to **File | New** | **Local PHP Project**.
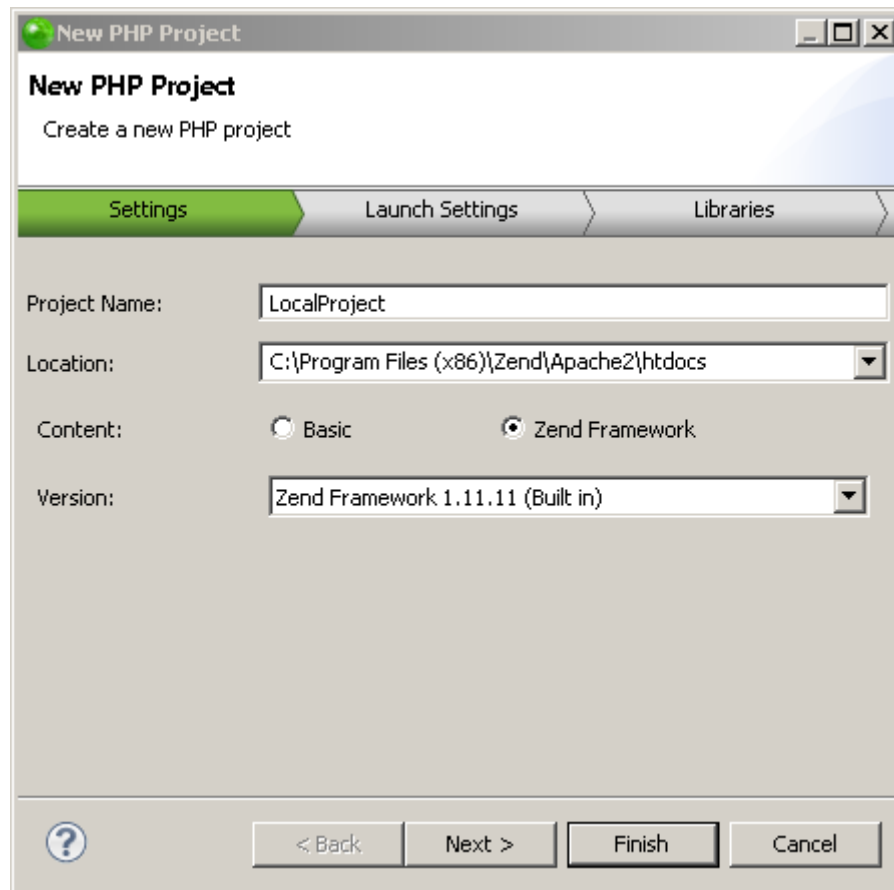
3. Enter the following information:
- Project Name- Name your project "LocalProject"
- Location- The project location for this tutorial is *'C:\Program Files(x86)\Zend\Apache2\htdocs'*
- Content - Select Zend Framework
- Version – Leave as the default

4.   Click **Next** to move to the Launch Settings dialog.
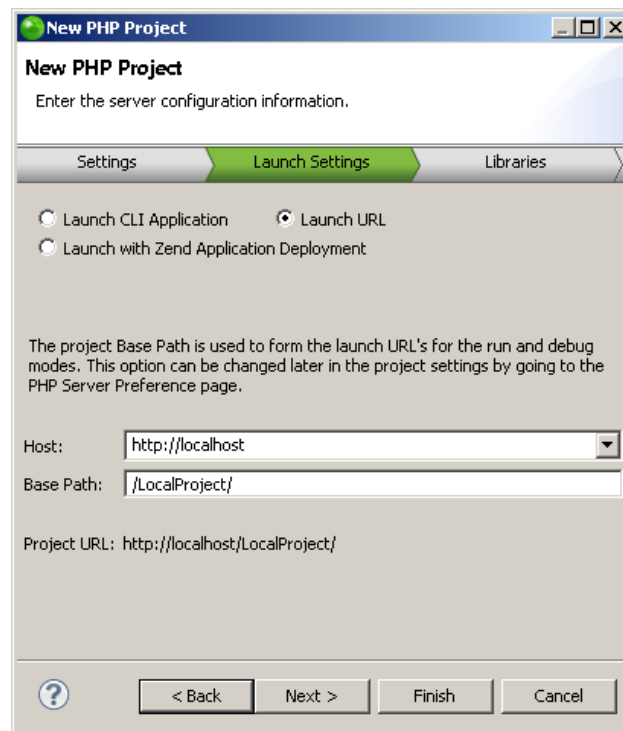
5.   Select **Launch URL**, and click **Finish**.

Your new project is created and appears in the PHP Explorer view.
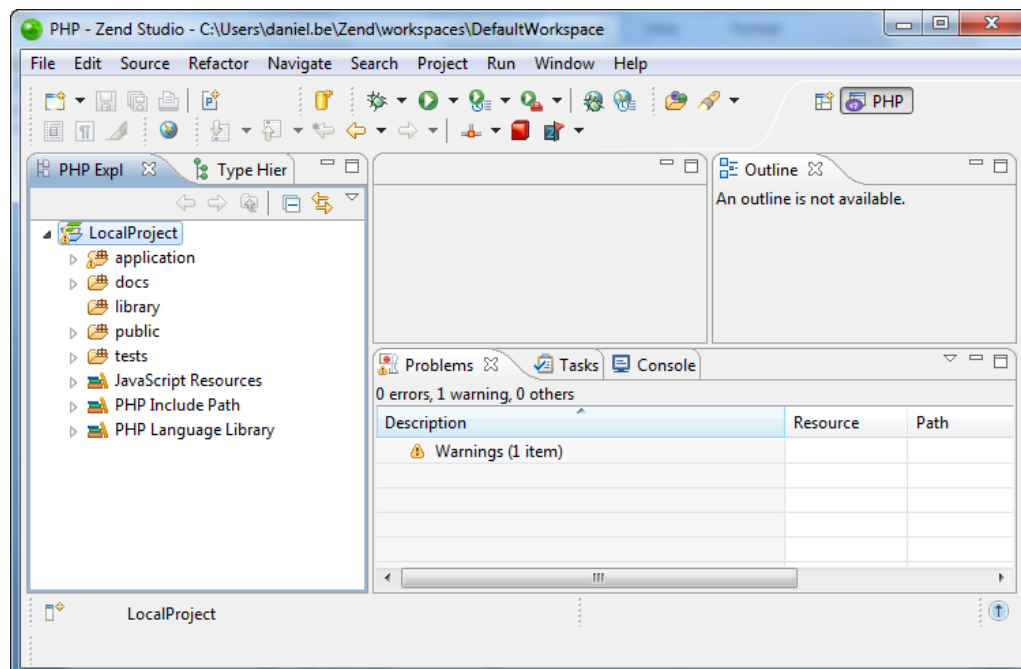
Step 2: Debugging Your Application

The Zend Studio debugging function allows you to test your files and applications and detect errors in your code. The debugger allows you to control the execution of your program using a variety of options including setting breakpoints, stepping through your code, and inspecting your variables and parameters.

To debug your application:

1. In the PHP Explorer, right-click the project and select **Debug As | PHP Web Application**. A dialog is displayed confirming the launch URL.
2. Click **OK**.
3. When prompted to open the Debug perspective, click **Yes**. Your application appears in the Debug perspective and stops at the first breakpoint.
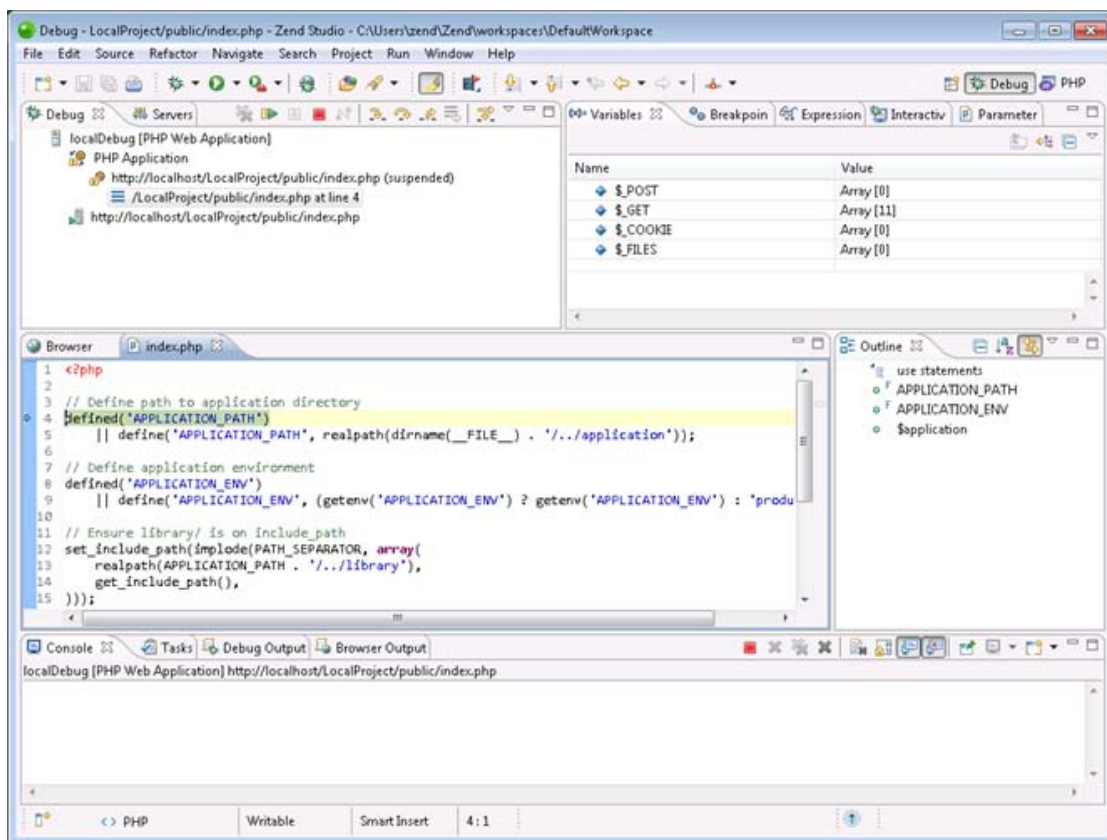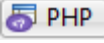
4. To continue the debug click .

Step 3: Modifying your Code

To be able to view any changes made to the code in your application, this tutorial will first explain how to modify the code.

To modify the code of your application:

1. Return to PHP Perspective by clicking on the PHP Perspective icon 🖥 PHP .
2. In the PHP Explorer view, expand your project and go to **Application| View | Scripts | Index | index.phtml**
3. Make a small textual change and save the project.

Step 4: Running Your Application

You will now run the application to see it with the modified code.

To run the application:

1. In the PHP Explorer view, expand your project and go to **Public | index.php .**
2. Right-click the index.php file and select **Run As | PHP Web Application**.

   Your application appears in the internal Web browser with the changes.
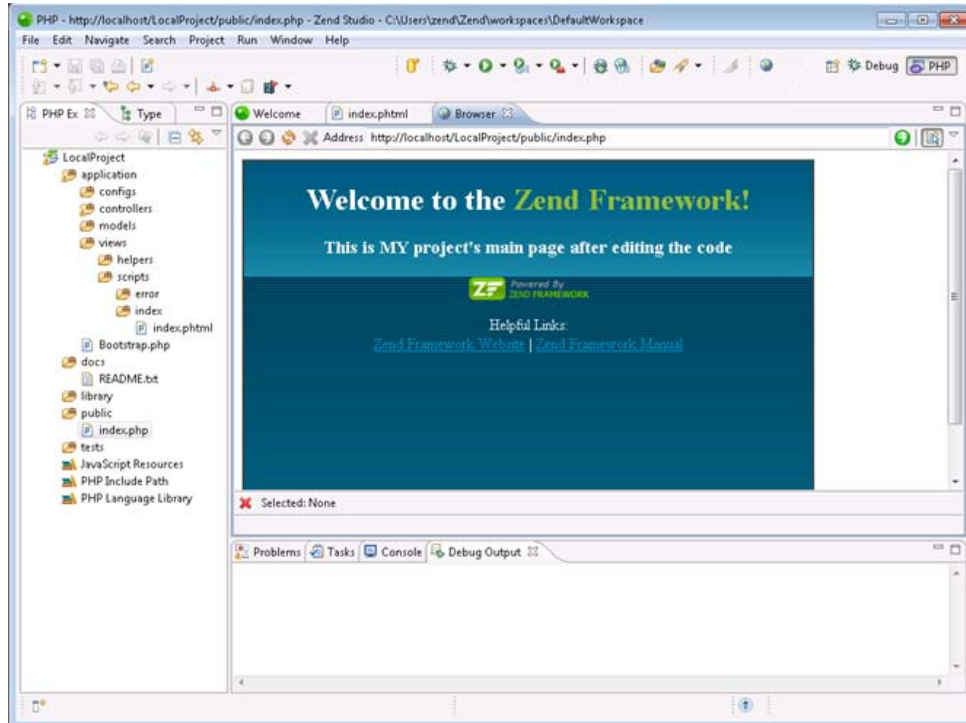
Customizing and Registering Zend Studio

Zend Studio now enables you to customize your working environment according to the specific requirements that your project entails. Plugins can be added and removed according to your preferences, ensuring a much lighter and faster operation of Zend Studio.

The Zend Studio Plugin List includes widely used plugins such as CVS, SVN, RSS, etc.

In This Tutorial You Will Learn

- To customize Zend Studio by adding and removing the plugins you want to work with. The Welcome page includes a list of available plugins, which you can add or remove from your Zend Studio.
- To register Zend Studio. Once Zend Studio is installed, all its features will be available for a 30 day trial. To enable full features after this period of time, you will need to register the product.
- To unregister Zend Studio. Unregistering Zend Studio will enable you to change the license number.

Prerequisites

- An installed Zend Studio 9.o or above. To install Zend Studio go to http://www.zend.com/en/downloads.
- A valid Zend Studio license. For more information on purchasing a Zend Studio license go to http://shop.zend.com/en/zend-studio-for-eclipse.html.

Step 1: Customizing Zend Studio

After installing Zend Studio, your project may require plugins that are not included in the product's download. All plugins are available from within the product and can be removed if you no longer need them.

Installing Plugins

In this tutorial you will be adding the Eclipse Color Theme and Mylyn plugins.

To add the Eclipse and Mylyn plugins:

1. Go to **Help | Welcome** to open the Zend Studio Welcome screen.
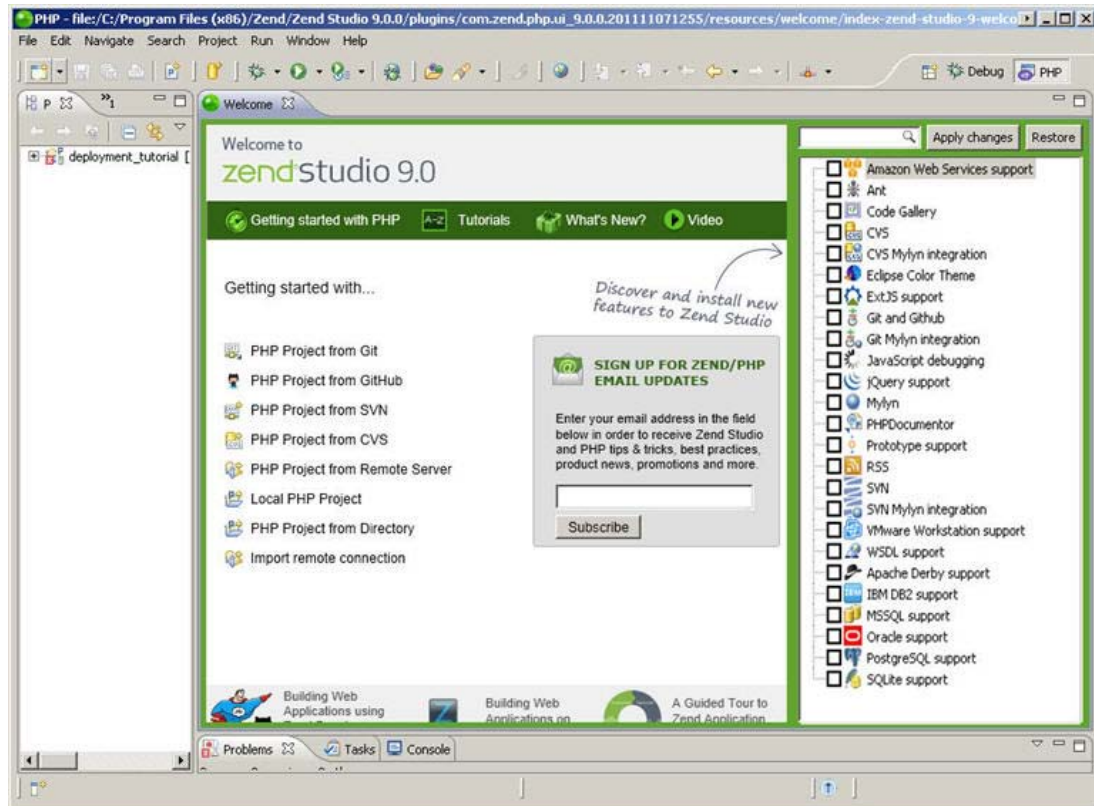
2. From the plugin list on the right, mark the **Eclipse Color Theme** and **Mylyn** check boxes**,** and click **Apply changes**.
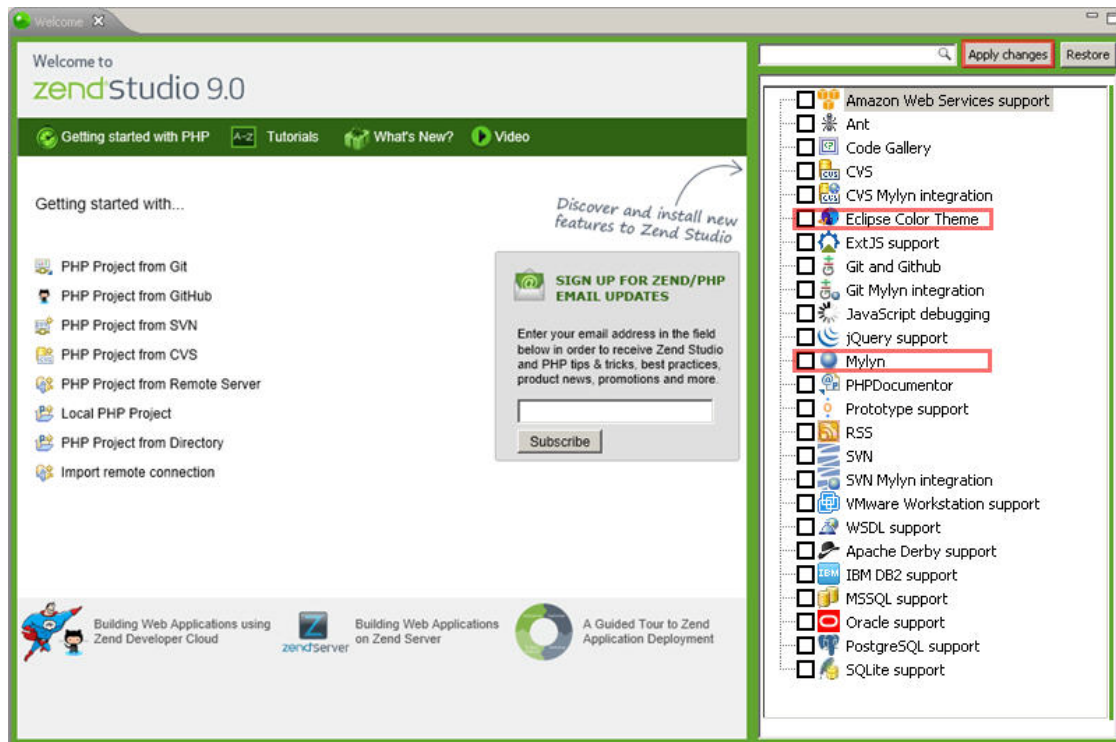
3. When prompted, click **Restart Now**.

Zend Studio now includes the Eclipse Color Theme and Mylyn plugins.

Note:

If you do not restart Zend Studio, the changes will not be applied.

Uninstalling Plugins

To remove the Eclipse Color Theme and Mylyn plugins:

1. Go to **Help | Welcome** to open the Zend Studio Welcome screen.
2. Unmark the Eclipse Color Theme and Mylyn plugins and click **Apply changes**.
3. When prompted, click **Restart Now**.

Zend Studio restarts with the plugins uninstalled.

Note:

At any stage prior to applying changes and restarting Zend Studio, clicking on **Restore** will revert to current settings.

Step 2: Registering Zend Studio

Once Zend Studio is installed, all its features will be available for a 30 day trial. At the end of the 30 day trial, the professional features will be disabled. These include refactoring, getter and setter generation, new PHP element creation wizards, Zend Framework integration, profiling, PHPUnit testing, Zend Server integration, debugging through tunneling, advanced code formatting, code galleries and more.

To enjoy the full range of Zend Studio functionality, you should register a valid license. You can purchase a Zend Studio license from the Zend Store.

To register your Zend Studio:

1. If your Zend Studio is already running, close and reopen it. If it is not already running, open Zend Studio.

   A **Zend Studio Trial** screen appears.
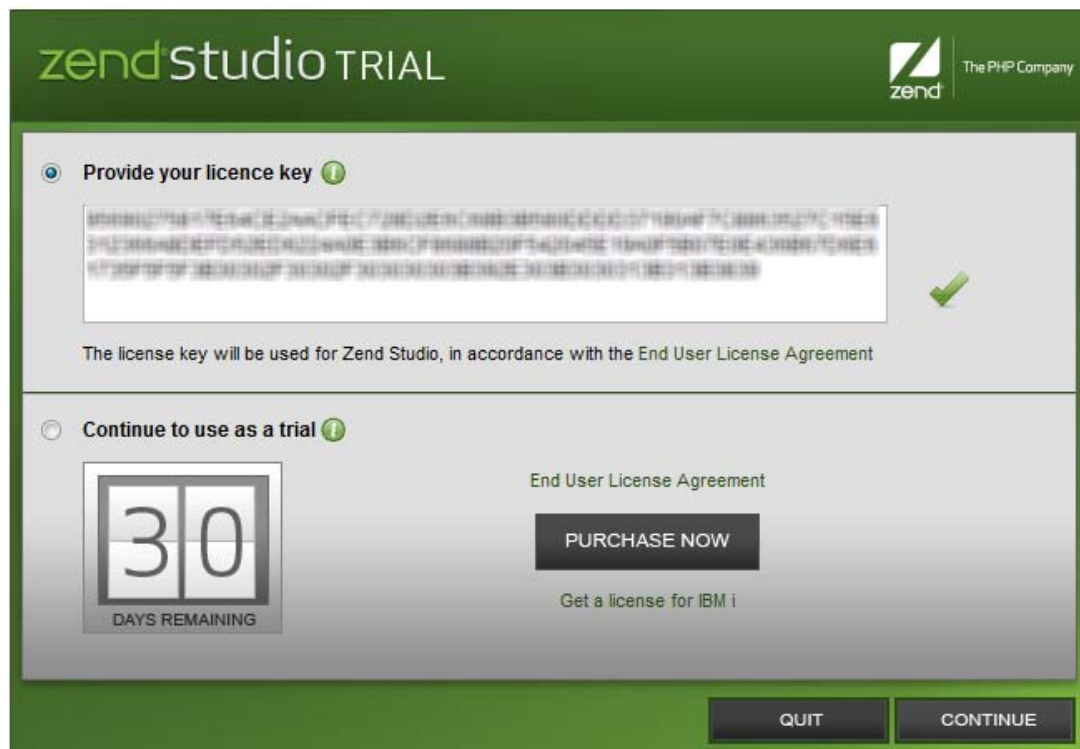
2. Select **Provide your license key**, and enter the key in the field below.

   If you do not have a license, go to the Zend Studio product page and click **Purchase Now**.

3. Click **Continue**.

   Your registered Zend Studio opens.

You can now begin working with your Zend Studio or see additional Tutorials.

Unregistering Zend Studio

If you wish to change the license key for Zend Studio you will first need to unregister. Once you unregister Zend Studio, all professional features will become disabled after 30 days.

To unregister Zend Studio :

1. Go to **Help | Unregister**.

2. When asked to confirm the unregistration process, click **OK.**

    Zend Studio restarts unregistered.

A Guided Tour of Zend Application Deployment Support

This tutorial will help you understand how to deploy your application using the deployment.XML file. You will be working with an existing GitHub project and its deployment.xml file to customize your application's deployment and to create a ZPK package.

In this Tutorial You will Learn

- To create a new project from GitHub in Zend Studio: You will be provided with an existing GitHub account and project. For more information, see Creating a GitHub Account.
- To work with your project's deployment.xml file: Launching a project in Zend Studio with deployment support creates a deployment.xml file. This file enables you to customize your application before the packaging process.
- To create a ZPK package: Packaging your application enables you to export it to other members of the working process for testing and evaluation.

Provided Items

For this tutorial we will provide you with the following:

- The GitHub project you will be working with.
- The finished GitHub project after the tutorial. The finished GitHub project will include all the changes that are described in this tutorials and can be downloaded from files.zend.com/help/zend-studio/zend-studio-creating_deployment_package_complete.zip
- Code for inserting into the project's files. To access the code, see Tutorial Code or download the zip file from files.zend.com/help/zend-studio/zend-studio-creating_deployment_package_complete.zip

Prerequisites

- Zend Studio 9.0 or above which can be downloaded from the Zend Studio Downloads page. For information on installation see the Zend Studio Installation Guide.
- Zend Server 5.5 or above which can be downloaded from the Zend Server Downloads page . For more information see the Zend Server Installation Guide.
  The Zend Server Installer includes optional components that can be downloaded as a part of the installation process. For this tutorial you will need to install the MySQL component.

Step 1: Creating a GitHub Project in Zend Studio

For this tutorial you will first need to create a GitHub project in Zend Studio. You will be importing an existing GitHub project that we have provided for the purposes of this tutorial.

For information on creating a GitHub account and project, see http://help.github.com

To create a new project from GitHub in Zend Studio:

1. Go to **File | New | PHP Project from GitHub.**

   The GitHub Repository dialog appears.

   If the GitHub plugin is not installed, Zend Studio will automatically begin the process for installing the plugin that will include restarting the program. Wait until this installation is complete before continuing. For more information see Customizing and Registering Zend Studio.
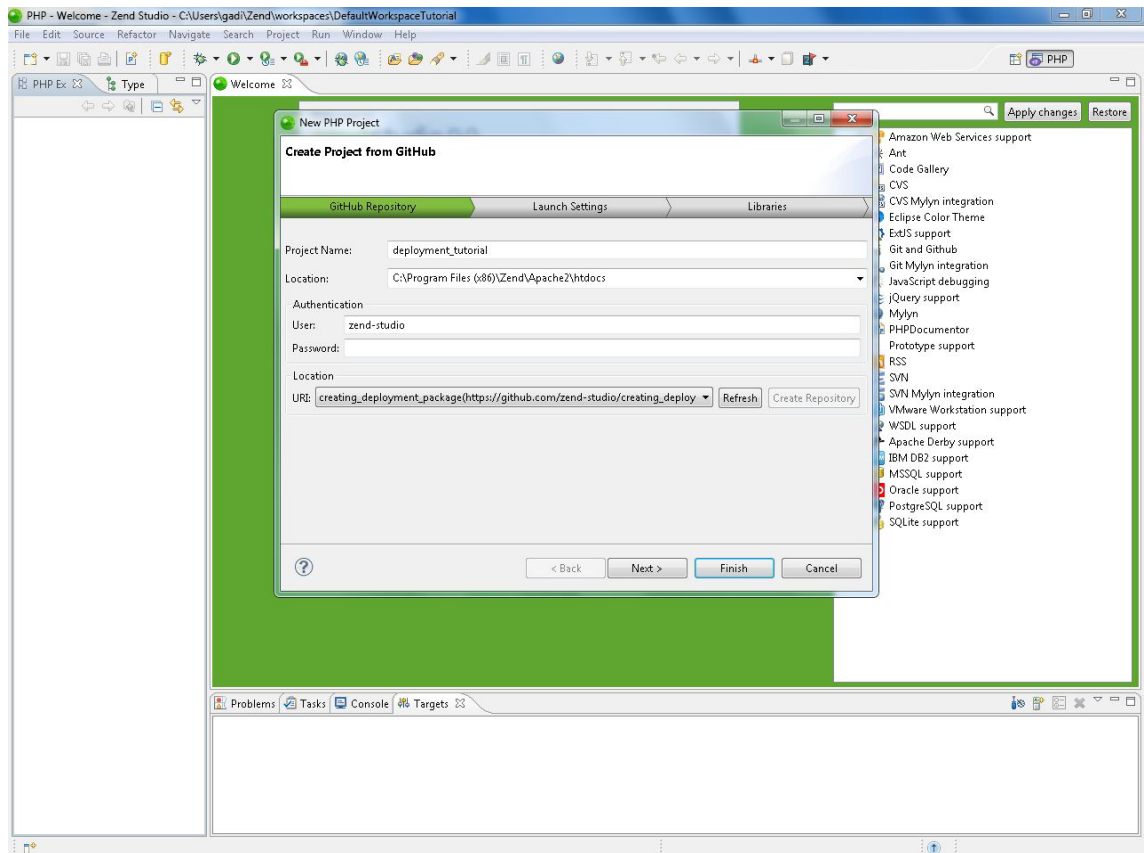
2. Enter the following information:

- Project Name: Name your project *deployment_tutorial*

- User: *zend-studio*

3. Click **Refresh.**

   Your application location will appear in the Location URL field.

4. Click **Next** to move to the Launch Settings dialog.

5. Select **Launch URL**, enter the address of the local host server and click **Finish**.

6. Run your application by right-clicking the project in the PHP Explorer and selecting **Run As PHP Application**.

Your application opens in the Zend Studio workspace and is running locally.
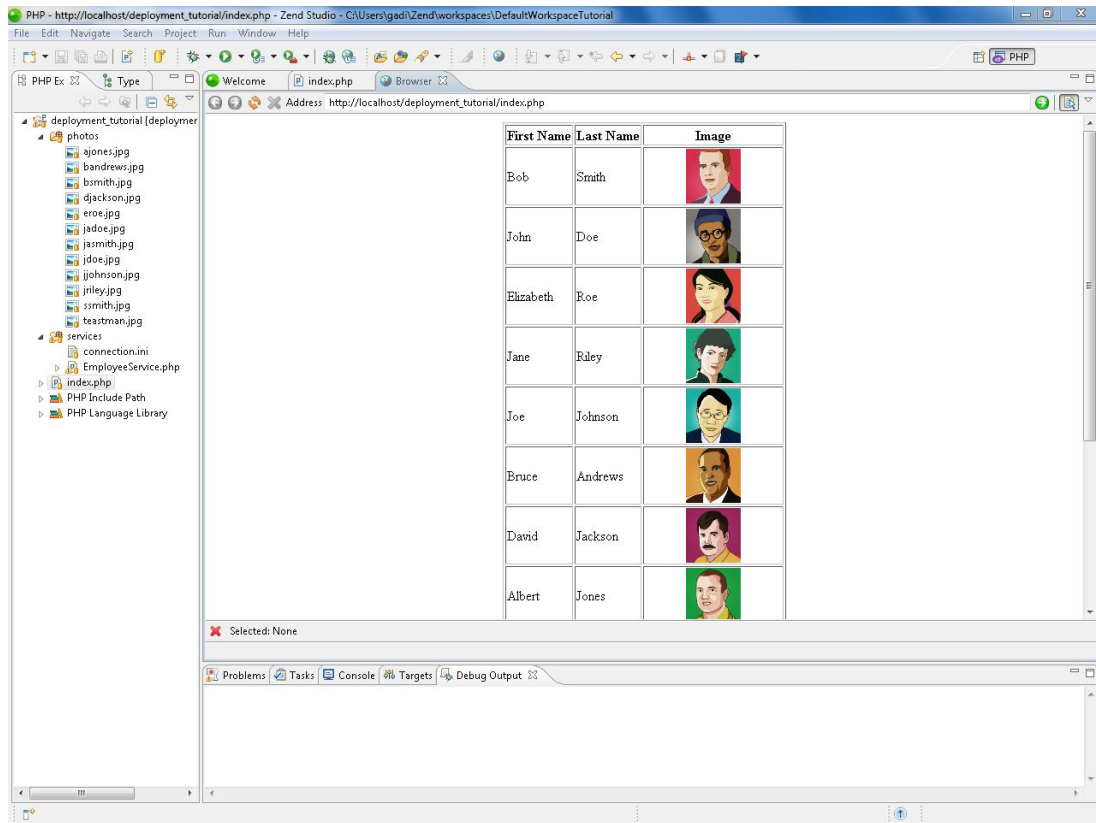
Note:

It is recommended that you save a copy of your project on your machine before continuing this procedure.

Step 2: Enabling Deployment for your Project

The project we have created was not defined to include Application Deployment Support. In order to work with the deployment.xml file and create the ZPK application package, Application Deployment Support must be enabled.

To enable deployment:

Add deployment support to your project by right-clicking your project in the PHP Explorer view and select **Configure | Add Application Deployment Support.**

Your project will appear in the PHP Explorer view with a deployment.xml file.
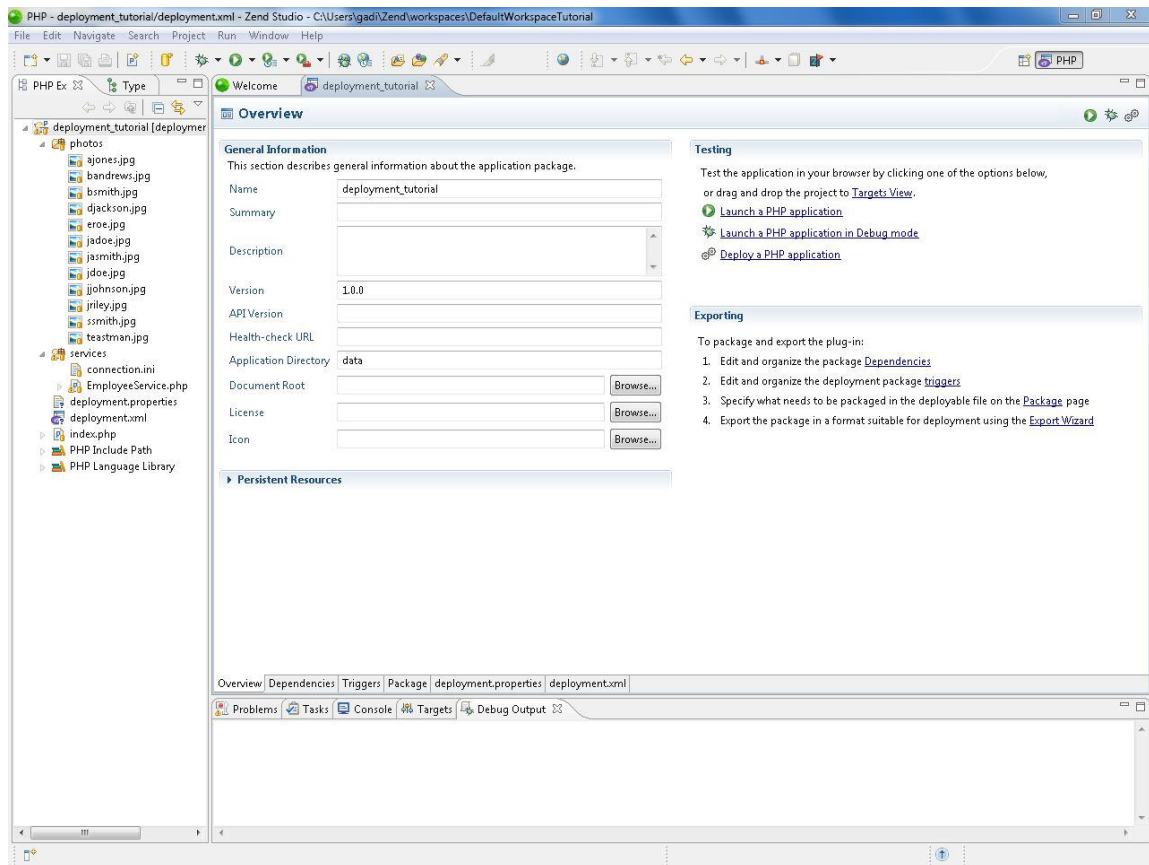
Step 3: Working with the deployment.xml File

You will now learn how to edit the deployment.xml file before creating the ZPK package. The deployment.xml file allows you to customize the deployment settings for your application.

After enabling Application Deployment Support, the deployment.xml file appears. The following procedure is according to the tabs in the file.

Overview Tab

In the Overview tab, enter general information about the project:

- Name: deployment_tutorial

- Version: 1.0
- Application Directory: data

Dependencies Tab

Before deployment, Zend Server checks that the target's settings match the application defined dependencies.

1. Click **Add**.

   The Add Dependency dialog is displayed.
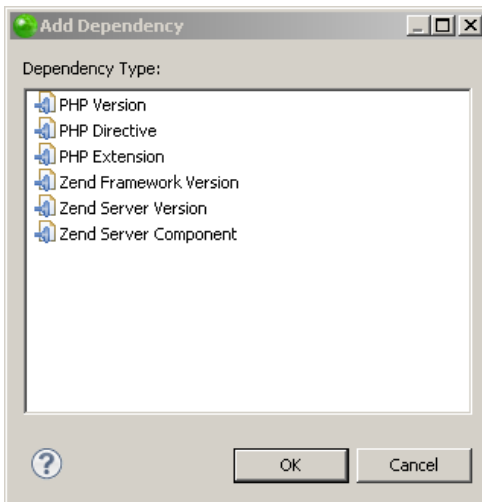
2. Click **PHP Version**.

   The Version Dependency dialog appears.
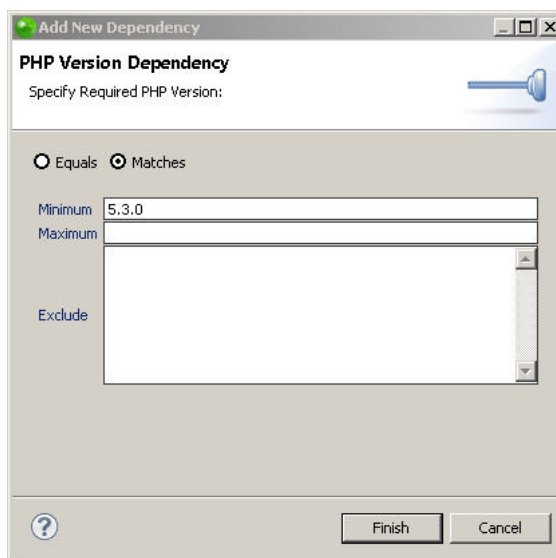
3. Select **Matches** and enter "5.3.0" in the Minimum field.

4. Click **Finish**.

The dependency is displayed in the Dependencies list.
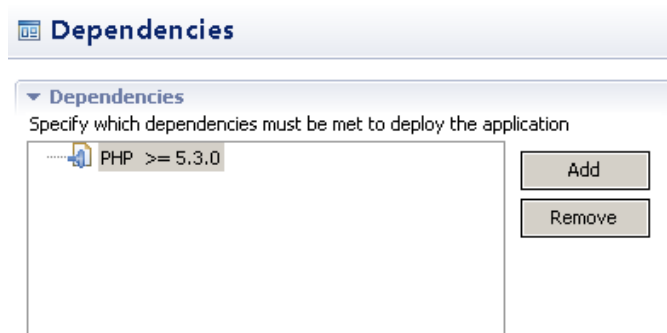
Triggers Tab

The Triggers tab includes the Deployment Scripts area and the Variables and Parameters area.
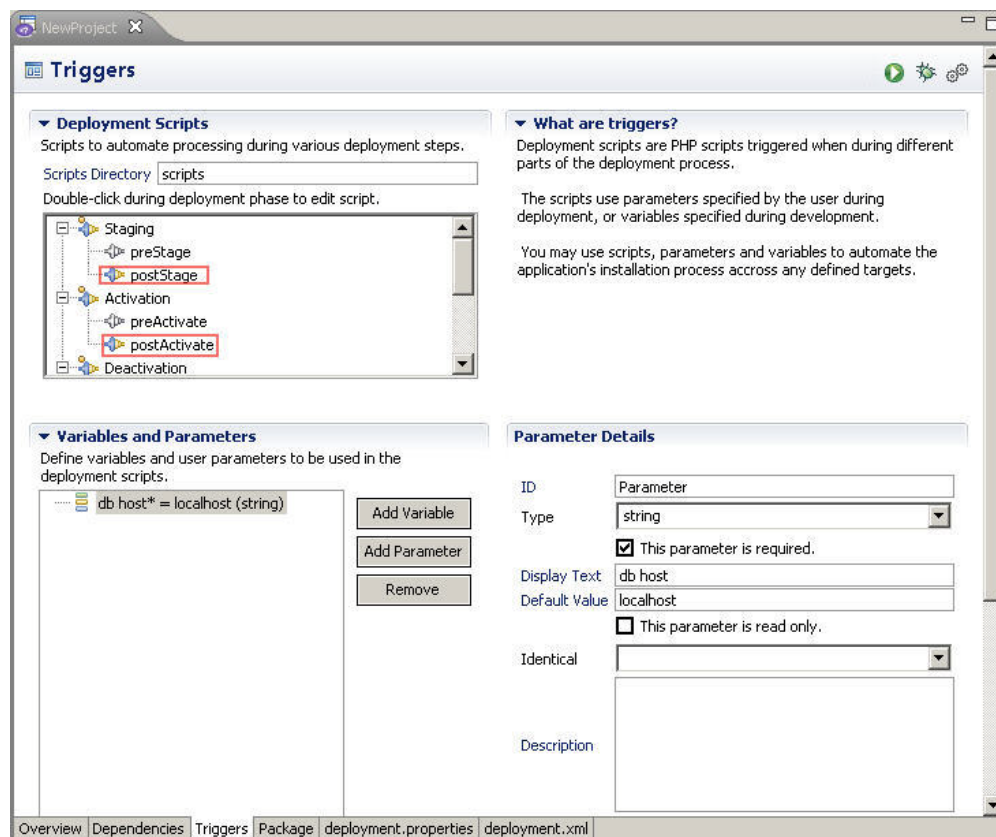
Deployment Scripts

You will now create new files that include the deployment scripts for this tutorial project.

1. Double-click the postStage and postActivate triggers.
2. When prompted, click **Yes** to confirm the creation of the trigger scripts.

    The new PHP trigger files are displayed and also appear in the Scripts file in the PHP
    Explorer view.

3. Copy and paste code into the newly created trigger files:

- postStage – This hook script changes the connection configurations according to the user
    parameters. The connections.ini file is created.

    You can copy the code from <u>Tutorial Code</u> or download the attached zip file at
    <u>files.zend.com/help/zend-studio/zend-studio-creating_deployment_package_complete.zip</u>

- postActivate - This hook script creates the database table using the scripts to be created
    in the next section..

    You can copy the code from <u>Tutorial Code</u> or download the attached zip file at
    <u>files.zend.com/help/zend-studio/zend-studio-creating_deployment_package_complete.zip</u>

Parameter Script

This file is used for converting user-entered parameters and built-in environment parameters to local php parameters that can be used by all the script files.

1. In PHP Explorer, expand the project, right-click the Scripts file, and select **New | PHP File**.

   The New PHP File dialog is displayed.
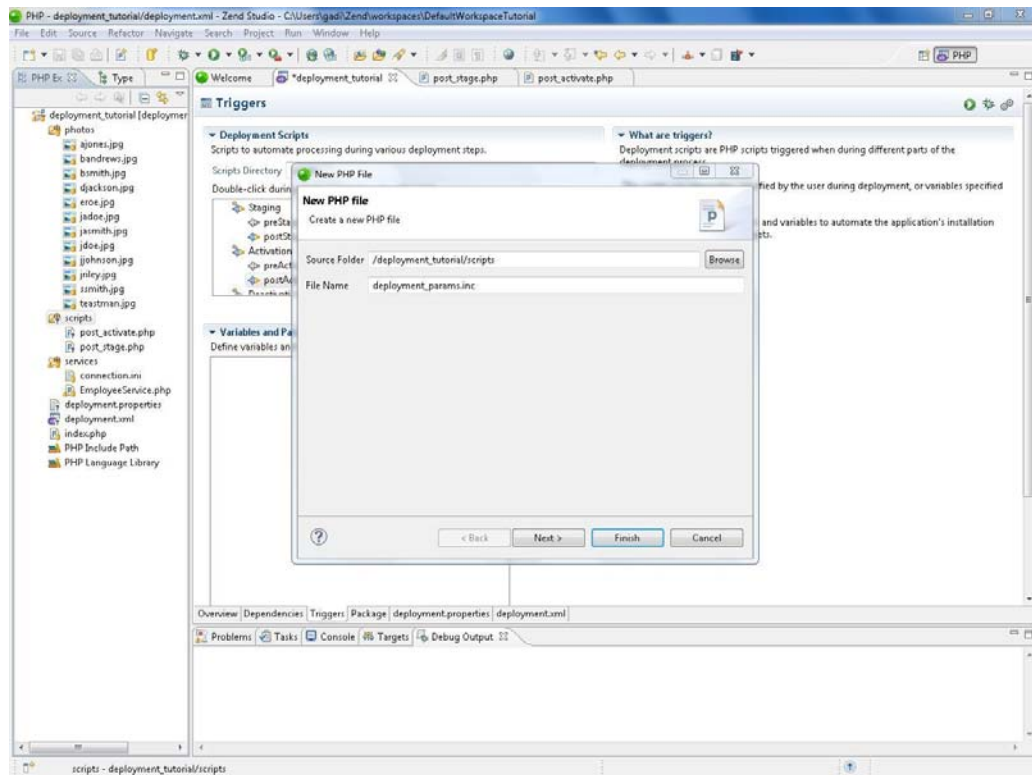
2. In the File Name field, enter "deployment_params.inc", and click **Finish**.

   The new file is displayed and appears in the Scripts file in the PHP Explorer.

3. Copy the code for the relevant file from [Tutorial Example Codes](#) and paste into the new file.

4. Save the file.

Additional Scripts

1. Create two more PHP files named as follows:

   - deployment_demo_table.sql- This file is the SQL database dump file.

Note:

The "TABLE_PLACEHOLDER" string will be used in the subsequent create_table.php file.

   - create_table.php- This file creates the table using the SQL file.

2. Repeat steps 3-4 in the previous section for each file.

   Your project now includes five script files that are displayed in the deployment.xml main view and in the PHP Explorer.
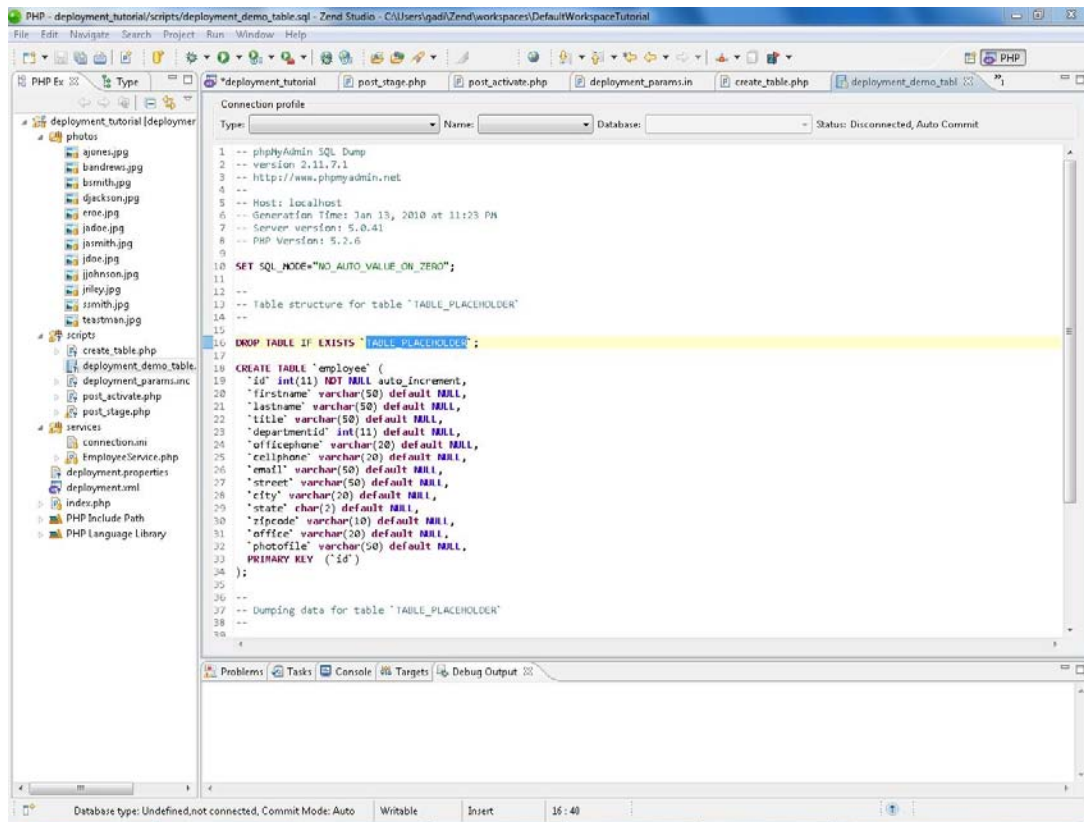
Variables and Parameters

Add parameters that are used in the hook scripts. You will enter the value of the parameters during the deployment process.

1. Return to the Triggers tab.

2. Click **Add Parameter**.

   The new parameter will appear in the Variables and Parameters list.

3. In the Parameter Details area, which can be opened by selecting a parameter, enter the following definitions:

▪ ID:DB_Host

▪ Type: string

▪ Mark This Parameter is Required.

- Display Text: db host.
- Default Value: localhost.

4. Repeat steps 2-3 with the following definitions:

Parameter ID	Type	This Parameter is required	Display Text	Default Value
DB_Name	string	Y	db name	test
DB_Table	string	N	table name	employee
DB_USERNAME	string	Y	username	
DB_PASSWORD	password	Y	db password	

5. The new parameters are displayed in the Variables and Parameters list:

Package Tab

In the Scripts Directory Content, we define which files to include in the package.

1. Click the **Package** tab in the deployment.xml file.

2. In the Scripts Directory Content, mark the all the scripts in the scripts file.
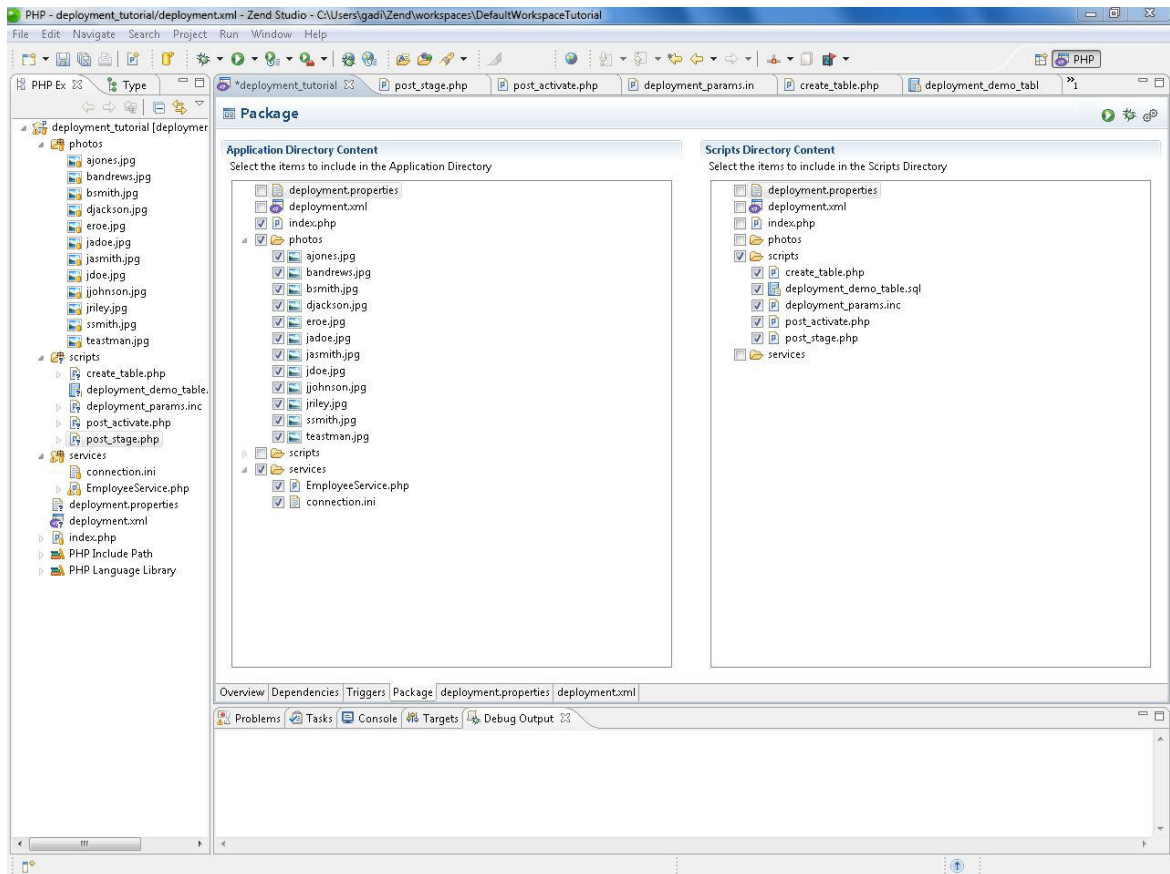
3. Save the project.

Step 4: Launching the Application

You will now test the application by launching it on your development server.

To launch the application:

1. Click the **Overview** tab in the deployment.xml file.

2. In the Testing Area, click **Launch a PHP Application**.

   The Launch PHP Application dialog is displayed, warning that no target has yet been defined for the deployment.
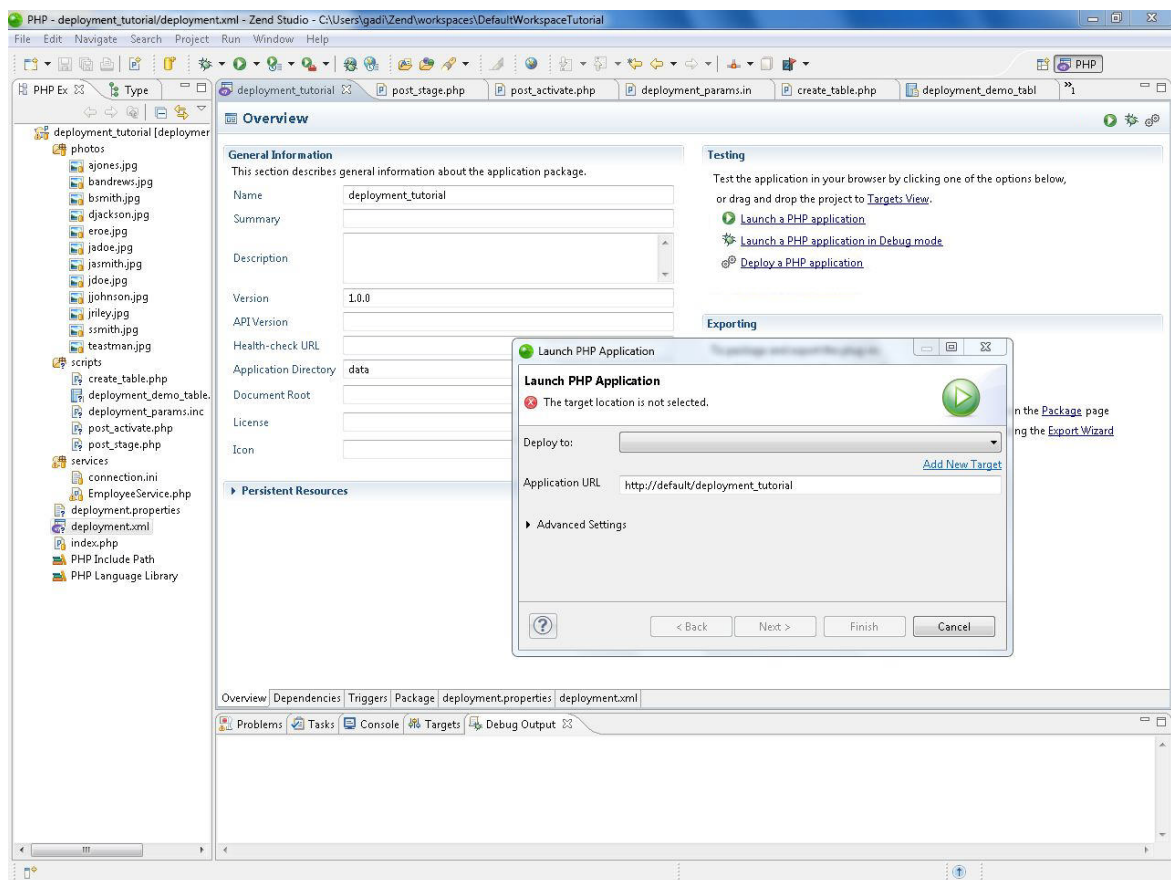
3. Select **Add New Target**.

   The Add Target dialog is displayed.
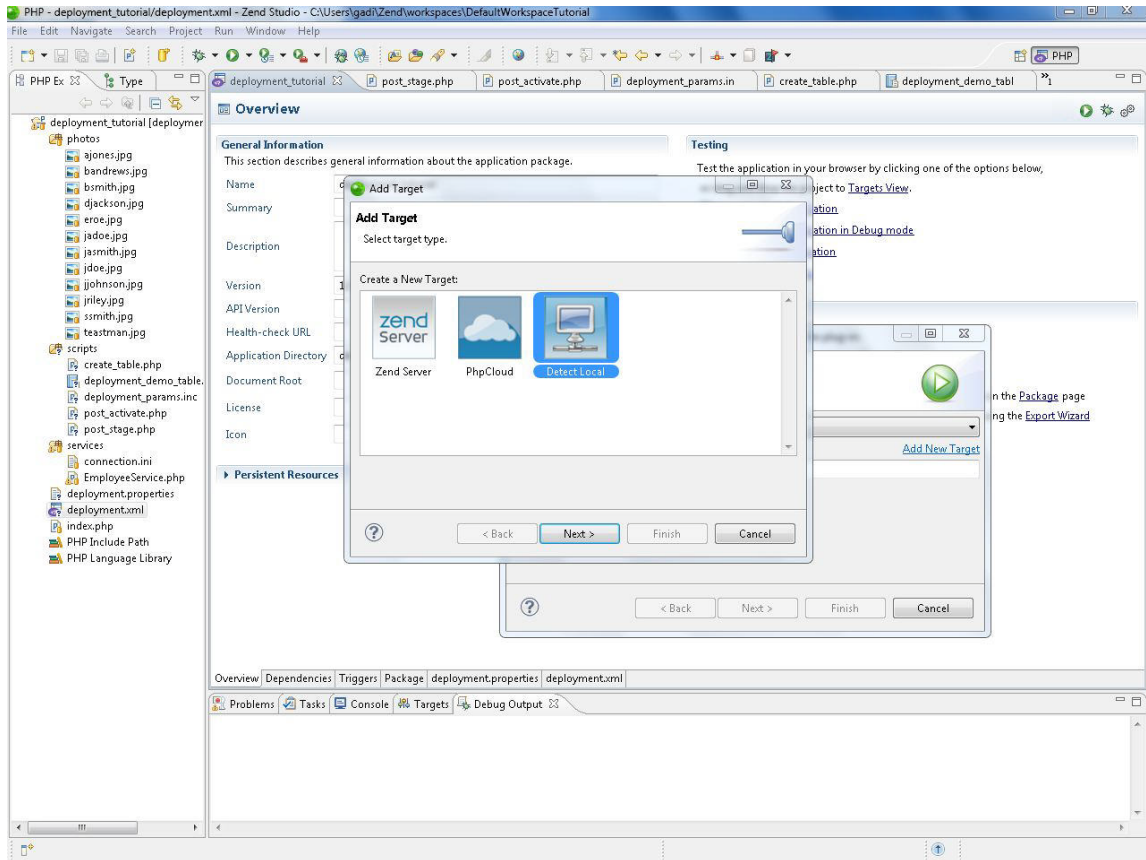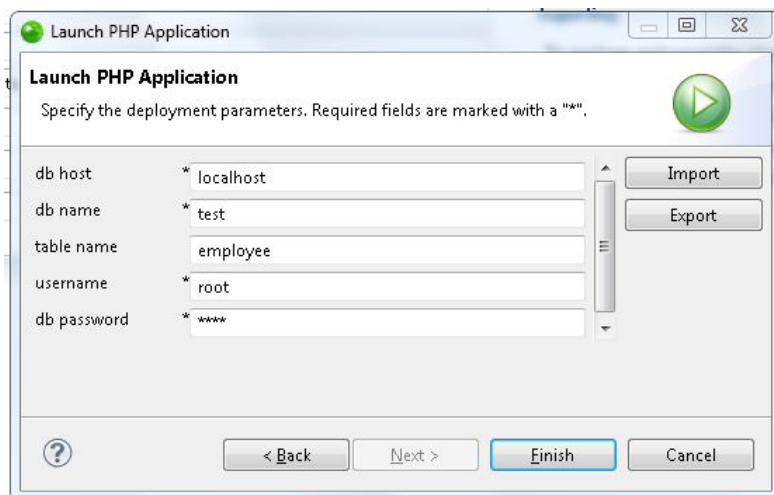
4. Select the Detect Local icon and click **Test Connection**.

5. If no problems arise, click **Finish**.

   The Launch PHP Application dialog is displayed again. Change the URL to tutorial_deploy to avoid two applications with the same name.
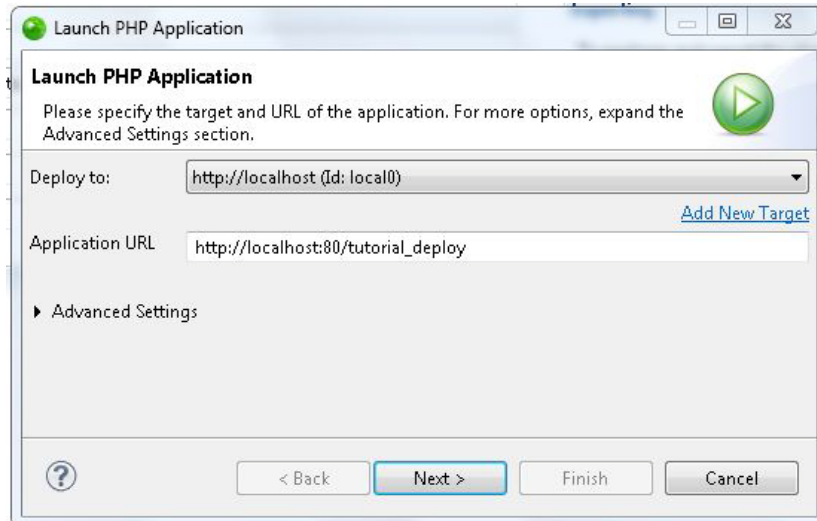
6. Click **Finish**.

The Parameters dialog is displayed.
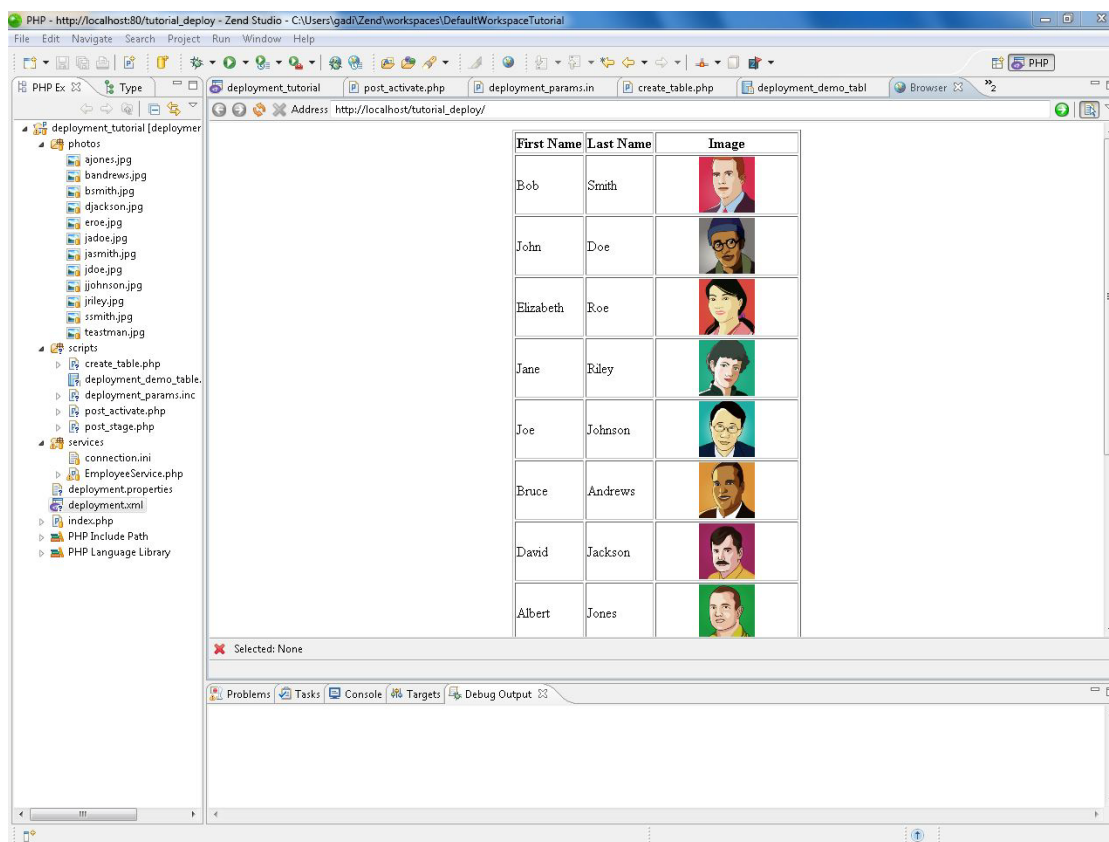
7. Enter the values for the parameters as you defined them.

The application will open in the internal browser.

Step 5: Exporting the Application

You will now package the application into a ZPK file. This makes your application portable and accessible by creating a file that can easily be sent to other members of the work process.

To package your application:

1. Click the **Overview** tab in the deployment.xml file.

2. In the Exporting area, click **Export Wizard**.

   The Export Deployment Package dialog is displayed.
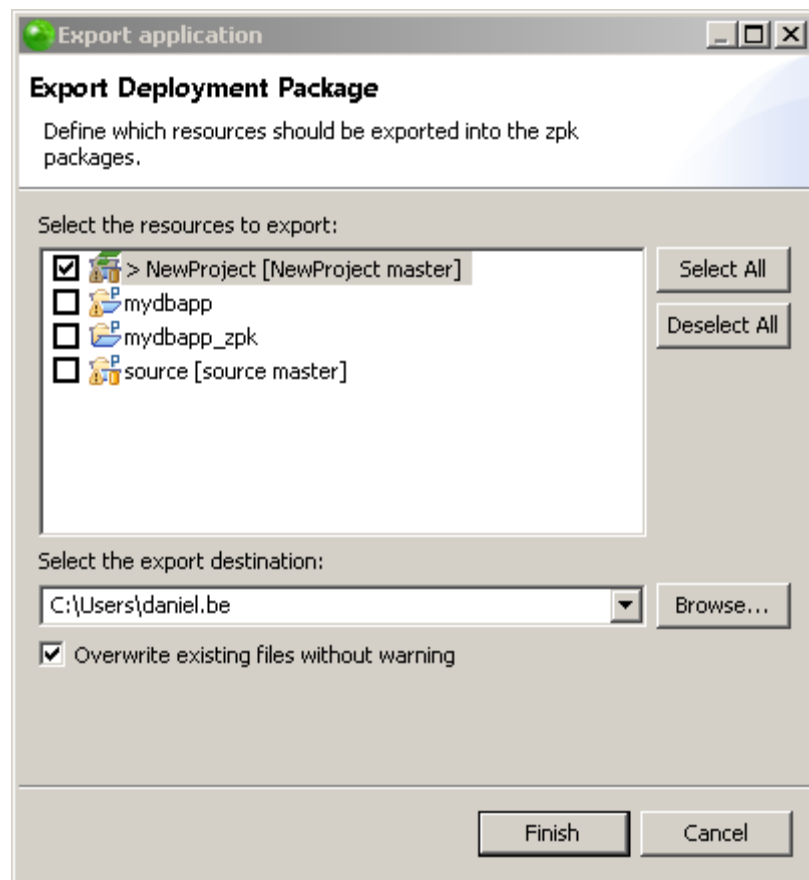
4. Select the project to be packaged.

5. Select the export destination by clicking **Browse**.

6. Click **Finish**.

   Zend Studio creates the ZPK package and exports it to the designated destination folder.

Tutorial Code

Below is the code used for the [A Guided Tour to Zend Application Deployment Support](#) tutorial.

1. Copy and paste the code into the created files.
2. Save before continuing to the next steps.

The code can also be downloaded together with all the project components from

[files.zend.com/help/zend-studio/zend-studio-creating_deployment_package_complete.zip](#)

post_activate.php File

```php
<?php
ini_set("max_execution_time", 1000);
if (getenv("ZS_RUN_ONCE_NODE") == 1) {
require_once(dirname(__FILE__) . "/create_table.php");
}
echo "Post Activate Succesful";
exit(0);
?>
```

post_ stage.php File

```php
<?php
require_once ("deployment.inc");
//replace values in EmployeeService File
$ini_path=$appLocation . "/services/connection.ini";
$open_file=fopen ($ini_path,"w");
fwrite($open_file,"[details]\n");
fwrite($open_file,"host = " . $dbHost . "\n");
fwrite($open_file,"user = " . $dbUsername . "\n");
fwrite($open_file,"pass = " . $dbPassword . "\n");
fwrite($open_file,"db = " . $dbName . "\n");
fwrite($open_file,"table = " . $dbTable . "\n");
```

deployment_params.inc File

```php
<?php
/*
 * This file is used for Converting User entered Params and some built in environmanet
 * paramaters to Local php parameters, that can be used by all script files
 */
ini_set ( "display_errros", 1 );
ini_set ( "error_reporting", E_ALL );
set_include_path ( get_include_path () . PATH_SEPARATOR . dirname ( __FILE__ ) );
// get the env vars
$baseUrl = getenv ( "ZS_BASE_URL" );
if (! $baseUrl) {
echo ("ZS_BASE_URL env var undefined");
exit ( 1 );
}
$appLocation = getenv ( "ZS_APPLICATION_BASE_DIR" );
if (! $appLocation) {
echo ("ZS_APPLICATION_BASE_DIR env var undefined");
exit ( 1 );
}
$dbHost = getenv ( "ZS_DB_HOST" );
if (! $dbHost) {
echo ("ZS_DB_HOST env var undefined");
exit ( 1 );
}
$dbUsername = getenv ( "ZS_DB_USERNAME" );
if (! $dbUsername) {
echo ("ZS_DB_USERNAME env var undefined");
exit ( 1 );
}
$dbPassword = getenv ( "ZS_DB_PASSWORD" );
if (! $dbPassword) {
echo ("ZS_DB_PASSWORD env var undefined");
exit ( 1 );
}
$dbName = getenv ( "ZS_DB_NAME" );
```

```php
if (! $dbName) {
echo ("ZS_DB_NAME env var undefined");
exit ( 1 );
}
$dbTable = getenv ( "ZS_DB_TABLE" );
if (! $dbName) {
echo ("ZS_DB_TABLE env var undefined");
exit ( 1 );
}
```

create_table.php File

```php
<?php
require_once ("deployment_params.inc");
//replace values in sqlFile File
$filestr = file_get_contents(dirname ( __FILE__ ) . "/deployment_demo_table.sql");
$filestr = str_replace("TABLE_PLACEHOLDER", $dbTable, $filestr);
file_put_contents(dirname ( __FILE__ ) . "/deployment_demo_table.sql", $filestr);
$link = mysql_connect ( $dbHost, $dbUsername, $dbPassword );
mysql_select_db ( $dbName, $link );
$queries = explode ( ";\n", file_get_contents ( dirname ( __FILE__ ) .
"/deployment_demo_table.sql" ) );
foreach ( $queries as $id => $query ) {

if ($query != '') {

$result = mysql_query ( $query, $link );
if (! $result) {
echo ( "Invalid query [$query]: " . mysql_error () );
die (1 );
}
;

}
}
mysql_close ( $link );
```

echo "Created sample Table $dbTable at $dbHost " . PHP_EOL;

deployment_demo_table.sql

```
-- phpMyAdmin SQL Dump
-- version 2.11.7.1
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Generation Time: Jan 13, 2010 at 11:23 PM
-- Server version: 5.0.41
-- PHP Version: 5.2.6
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
--
-- Table structure for table `TABLE_PLACEHOLDER`
--
DROP TABLE IF EXISTS `TABLE_PLACEHOLDER`;
CREATE TABLE `TABLE_PLACEHOLDER` (
  `id` int(11) NOT NULL auto_increment,
  `firstname` varchar(50) default NULL,
  `lastname` varchar(50) default NULL,
  `title` varchar(50) default NULL,
  `departmentid` int(11) default NULL,
  `officephone` varchar(20) default NULL,
  `cellphone` varchar(20) default NULL,
  `email` varchar(50) default NULL,
  `street` varchar(50) default NULL,
  `city` varchar(20) default NULL,
  `state` char(2) default NULL,
  `zipcode` varchar(10) default NULL,
  `office` varchar(20) default NULL,
  `photofile` varchar(50) default NULL,
  PRIMARY KEY  (`id`)
);
--
-- Dumping data for table `employee`
```

--
INSERT INTO `TABLE_PLACEHOLDER` VALUES(1, 'Bob', 'Smith', 'User Research Specialist', 1, '(415)555-1111', '(415)444-2222', 'bsmith@adobe.com', '601 Townsend St', 'San Francisco', 'CA', '94103', 'Cube 1234', 'bsmith.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(2, 'John', 'Doe', 'Senior Computer Scientist', 2, '(408)555-2222', '(408)444-1111', 'jdoe@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'E12-123', 'jdoe.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(3, 'Elizabeth', 'Roe', 'Senior Experience Designer', 1, '(408)324-6432', '(408)234-3453', 'eroe@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'E11-123', 'eroe.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(4, 'Jane', 'Riley', 'Quality Engineer', 2, '(415)123-1234', '(415)123-4567', 'jriley@adobe.com', '601 Townsend St', 'San Francisco', 'CA', '94103', 'Cube 4567', 'jriley.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(5, 'Joe', 'Johnson', 'Computer Scientist', 3, 'None', '(415)324-9870', 'jjohnson@adobe.com', 'Unseated', 'San Francisco', 'CA', '94103', 'Unseated', 'jjohnson.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(6, 'Bruce', 'Andrews', 'Chief Executive Officer', 4, '(408)234-1253', '(408)324-3464', 'bandrews@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'W18-502', 'bandrews.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(7, 'David', 'Jackson', 'Chief Technology Officer', 5, '(408)321-3332', '(408)893-9808', 'djackson@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'W6-211', 'djackson.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(8, 'Albert', 'Jones', 'Principal Scientist', 5, '(408)234-5234', '(408)342-5345', 'ajones@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'W6-321', 'ajones.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(9, 'Terry', 'Eastman', 'Food Services Manager', 6, 'None', 'None', 'teastman@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'Unseated', 'teastman.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(10, 'Jane', 'Smith', 'Principal Scientist', 2, '(408)125-6534', '(408)758-4512', 'jasmith@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'W3-459', 'jasmith.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(11, 'Samantha', 'Smith', 'Product Marketing Manager', 7, '(408)155-5893', '(408)452-5555', 'ssmith@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'E12-234', 'ssmith.jpg');
INSERT INTO `TABLE_PLACEHOLDER` VALUES(12, 'Jane', 'Doe', 'Human Resources Manager', 7, '(408)485-9856', '(408)451-4774', 'jadoe@adobe.com', '345 Park Ave', 'San Jose', 'CA', '95110', 'E5-321', 'jadoe.jpg');

Concepts

Update Manager	PHP Include Path
Customizing	PHP Build Path
PHP Version Support	Path Mapping
New Project Wizards	Zend Browser Toolbar
Content Assist	Tunneling
Syntax Coloring	Zend Server
Automatic Completion	Code Tracing
Matching Brackets	PHPUnit Testing
Mark Occurrences	Refactoring
Code Folding	JavaScript Support
Code Commenting	JavaScript Debugger
Hover Support	JavaScript Libraries
Override Indicators	PHPDocs
PHP Working Sets	Code Galleries
Type Hierarchy	Zend Guard Integration
PHP Manual Integration	RSS Feeds
Real Time Error Detection	WSDL - Web Services Description Language
Local History	Zend Studio for IBM i Extras
CVS	Remote Server Support
SVN	Mylyn Integration
Zend Framework Development	Phar Integration
Database Connectivity	Ajax Tools
Running	VMware Workstation Integration
Debugging	Amazon AWS Toolkit for Zend Studio
Profiling	Application Deployment
Breakpoints	

Update Manager

Zend Studio's Update Manager allows for the easy installation of extra plug-ins, the updating of existing features and the easy updating of Zend Studio with the latest offerings from Zend.

See the Workbench User Guide for more on installing and updating features with the Update Manager.

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Customizing

Zend Studio enables you to customize your working environment according to the specific requirements that your project entails. Plugins can be added and removed according to your preferences, ensuring a much lighter and faster operation of Zend Studio.

For more information on installing or uninstalling plugins in Zend Studio, see Customizing Zend Studio Using the Welcome Page.

Welcome Page

Zend Studio's Welcome page is the first page that you will encounter after installing or opening the product. You can close the tab that contains the page and access the page at a later point by going to **Help | Welcome**.

On the right of the Welcome page is a list of the available plugins that can be added to or removed from your IDE.
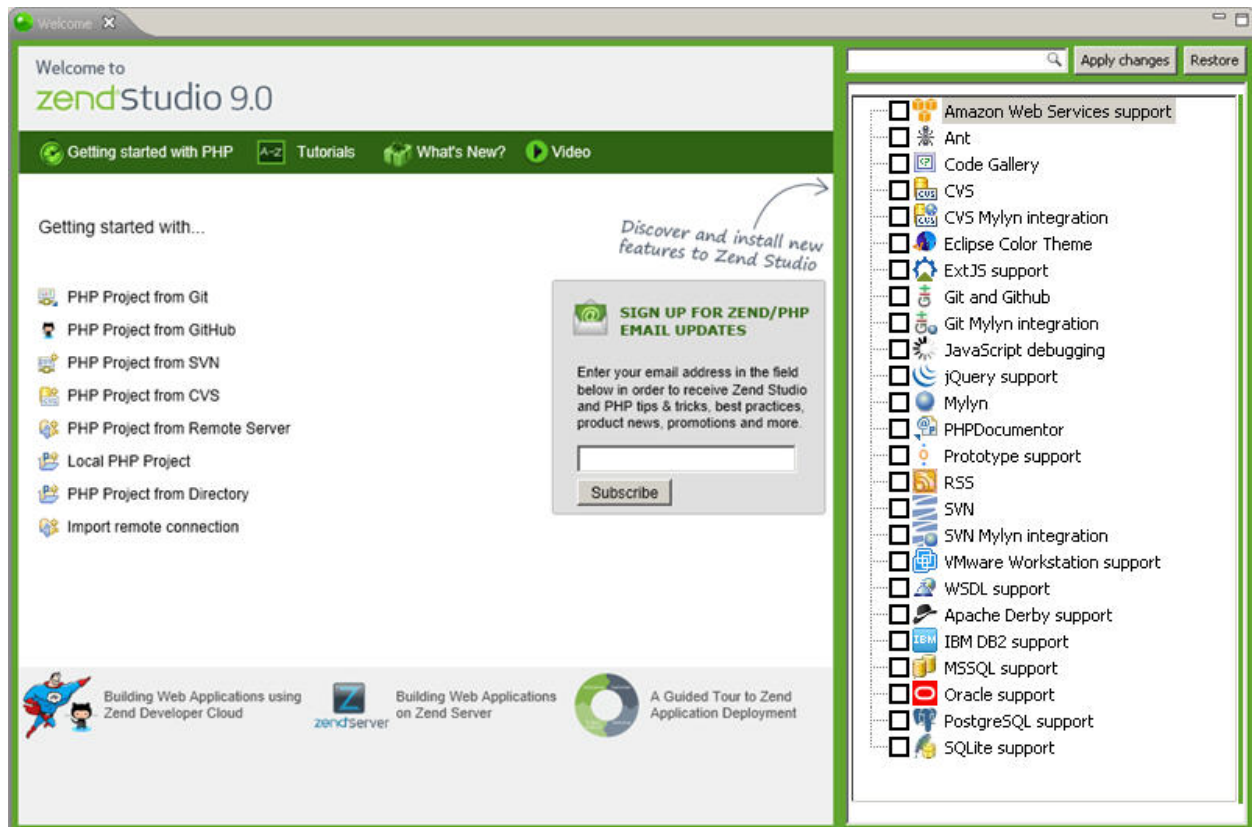
Available Plugins

To learn more about the available plugins, hover over the relevant plugin and click **Learn more**.

- Amazon Web Services support - See Amazon AWS Toolkit for Zend Studio.

- Ant

- Code Gallery

- CVS- See CVS.

- CVS Mylyn Integration

- Eclipse Color Theme

- ExtJS Support - See JavaScript Libraries.

- Git and GitHub

- Git Mylyn integration

- JavaScript debugging - See JavaScript Debugger.

- jQuery support - See JavaScript Libraries.

- Mylyn- See Mylyn Integration.

- PHPDocumentor - See PHPDocs.

- Prototype support - See JavaScript Libraries.

- RSS - See RSS Feeds.

- SVN - See SVN.

- SVN Mylyn integration

- VMware Workstation support - See VMware Workstation Integration.

- WSDL support - See WSDL - Web Services Description Language.

- Apache Derby enablement

- IBM DB2 enablement

- MSSQL enablement

- Oracle enablement

- PostgreSQL enablement

- SQLite enablement

PHP Version Support

Zend Studio supports PHP versions 4, 5.1/5.2 and 5.3.

PHP version settings affect:

- The elements displayed in the PHP Functions view.
- The options available in Content Assist.
- Debugging and Profiling functionality.

PHP version settings can be configured from the following places:

- PHP Executables can be added and edited from the PHP Executables Preferences page.
- Compatible Interpreters for selected PHP versions can be managed in the Execution Environments Preferences page.
- The default PHP executable used for new projects can be set in the PHP Interpreter Preferences page. Through this page you can also set the PHP version for specific projects.
  In addition, you can select which PHP Version should be used when creating a new PHP project by marking the Enable Project Settings checkbox in the New PHP Project dialog.
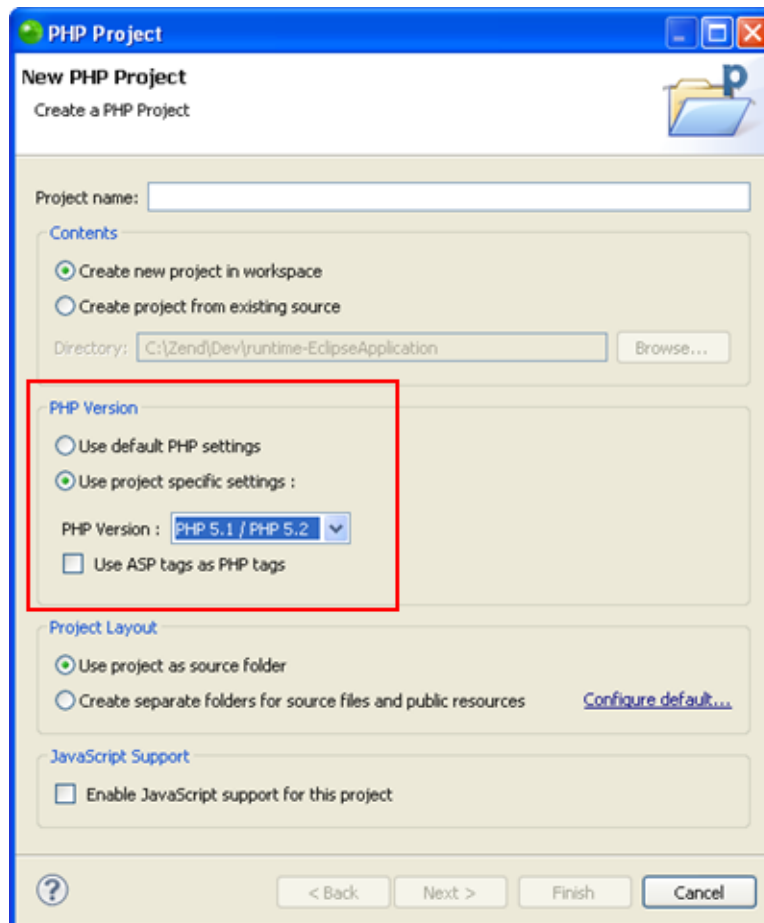
- The default PHP executable used with the debugger can be set in the <u>Debugging Preferences</u> page, accessed from **Window | Preferences | PHP | Debug**. Through this page you can also set the PHP executable used to debug specific projects.

  In addition, you can also configure the PHP executable used for each Debug and Profile   session in the Debug / Profile configuration screens.

Note:

In order to perform local debugging for PHP 4.x projects, you must manually add the relevant PHP executable to the <u>PHP Executables Preferences</u> page.
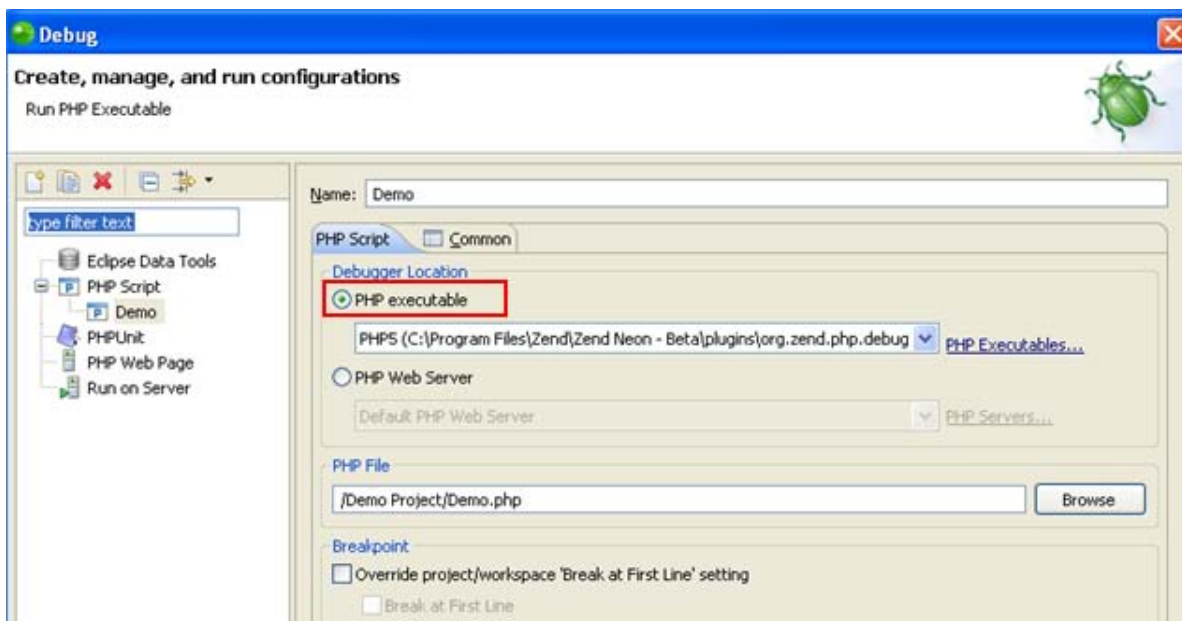
Note:

In order to minimize errors, the PHP Executable used for debugging /profiling should match the PHP version used for the project. This can be defined in the <u>Execution Environments Preferences</u> page.

New Project Wizards

Zend Studio's new project wizards make creating new projects an easy and simple task, enabling you to develop faster.

The wizards are tailored according to the type of project created but have the same UI based workflow.

New Project Workflow

The New Project workflows include the following stages:

1. Settings dialog: Here you will define the name of the project, it's location, and source information.
2. Launch Settings dialog: Here you will define the method of launching and, when opting to include deployment, also the target for deployment.

   For more information on these methods, see Launching an Application and Managing Targets.
3. Libraries dialog: Here you will select a library to add to your project.

To create a new project in Zend Studio, go to **File | New** and select which kind of project you wish to create.

New Project Wizards in Zend Studio

The following projects can be created in Zend Studio:

- New Project from CVS
- New Project from SVN
- New Project from Git
- New Project from GitHub
- New Local PHP Project
- New PHP Project from Existing Directory
- New PHP Project from Remote Server
- New PHP Project

Content Assist

The Content Assist feature enables the selection and insertion of existing code elements to complete partially entered code.

A list of possible code elements is displayed in relevant locations within your files according to the context of your cursor, which you can then select to be automatically entered into your code.

Each type of code element will have a unique icon. See PHP Icons for a description of each of the icons.

▲	Reserved PHP Words
●	Functions
▤	Templates
Ⓖ	Classes
❶	Interfaces
o^c	Constants
●	Variables (public)
Ⓟ	PHP File Include Call
Ⓝ	Namespaces (PHP 5.3)

Content Assist works with the following elements: PHP Classes, Functions, Variables, Constants, Keywords, Interfaces, attributes, values, nested functions, names, syntax and include calls, as well as all user defined Classes, Functions and Constants.

Note:
Content Assist will also be available for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information.

Content Assist in PHPDoc Block

Content Assist provides proposals for the PHPDoc tags:

- @return - A type description.

- @see - Documents an association to another element.

- @throws - Documents an exception thrown by a method.

Example:

```
19⊖ /**
20    * @return |
21    * @param i
22    * @desc Re
23    */
24⊖ function ro
25  {
26        $bgcolo
27        $bgcolo
28
```


Directory - standard.php
DirectoryIterator - SPL.php
DomainException - SPL.php
DOMAttr - dom.php
DOMCdataSection - dom.php
DOMCharacterData - dom.php
DOMComment - dom.php

Templated Content Assist

Applying Content Assist proposals to the editor creates a templated expression that provides argument completion according to the declaration of the method. Templated Content Assist helps you edit your method parameters quicker.

Example:

```
5
6        array_combine($keys, $values)|
7
```

$keys, $values

Using elements within the same scope

Elements within the same active project, file or function will be available for use with Content Assist.

Examples:
- Variables within a function will be added to the Content Assist list when the cursor is within that function.
- Elements defined within one file will be available in the Content Assist menu in other files within the same project.

Function Parameter Hints

When entering a function call, a Function Parameter Hint box will be displayed detailing the types of parameters which should be entered within the parentheses of the function call.

Example:

```php
<?php    string $name, mixed $value, bool[optional] $case_insensitive = null
define ()
?>
```

If the Function Parameter Hint box is not displayed automatically, place your cursor between the parentheses and press **Ctrl+Shift+Space**.

Content Assist for Include Statements

Content Assist can be activated for require and include calls to call files contained within the same project.

Inserting quotation marks between the parentheses of an include/require call will cause the Content Assist window to display the files available for the function. Selecting a file will complete the include/require call.

Example:

```php
include ("")
                  P Debug.php          Location:
                  P PHPDocument1.php    /Testing/Debug.php
?>                P test/Debug.php
```

```php
16   include ("Debug.php")
17   }
```

Class Type Hints

By using a comment you can assign a variable its exact class value. This assignment will affect the content assist of this variable accordingly.

Example:

```php
<?php

function getClass() {

    return new Test ( );

}
class Test {
    function printValues( $a,  $b) {
        echo "Values: $a, $b";
    }
}
/* @var $myVar Test */
$myVar = getClass ();
$myVar->
          ● printValues($a, $b) - Test    Location
                                               PHPFiveThree\abc.php
                                          Class
                                               Test
```

By using a comment you can assign a variable its exact class value. This assignment will affect the content assist of this variable accordingly.

To assign a variable its class value using a comment:

1. Create your function and assign variables to it.

2. Enter a comment in the form:

```
/* @var $"new variable" "Class name" */
```

Content Assist for Magic Members

Zend Studio supports Content Assist options for 'magic members'. These are properties and methods which were declared using the @property or @method tags within PHP DocBlock comments rather than in the PHP script itself.

See

http://manual.phpdoc.org/HTMLSmartyConverter/PHP/phpDocumentor/tutorial_tags.property.pkg.html for more information on magic members.

Example:

```php
<?php
/**
 * @property-read int $foo the foo prop
 * @property-write Magician $bar the bar prop
 * @method int borp() borp(int $int1, int $int2) multiply two integers
 */
class Magician {
    private $_thingy;
    private $_bar;
}

$blah=new Magician();

$blah->
?>
```

borp()	Location
bar	MagicMembers.php
foo	**Class**
	Magician

59

Camel Case Matches

Content Assist supports camel case matches when entering code, which displays Content Assist options based on an element's initials.

Example:

```
/**
 * @param unknown_type $personCount
 */
public static function setPersonCount($personCount) {
    $this->pC
```

Note:

Camel Case matching is case sensitive.

Namespaces

When the PHP executable used for the project is set to version 5.3, content assist is available for namespaces.

Namespace elements are represented by the Namespace icon ⓝ .

Example:

Configuring Content Assist

To configure PHP Content Assist options, go to the [Content Assist Preferences](#) page, accessible from **Window | Preferences | PHP | Editor | Content Assist**.

To configure JavaScript Content Assist options, go to the JavaScript Content Assist Preferences page, accessible from **Window | Preferences | Web | JavaScript | Editor | Content Assist**.

Syntax Coloring

Zend Studio can automatically apply different colors and font attributes to different PHP syntax elements in order for your scripts to be more easily navigable and to help you find relevant sections quickly and easily. With the Syntax Coloring Rules you can set the foreground color, background color and font type for a range of different elements in your code through the Syntax Coloring Preferences page. This allows for a quick assessment of your code with the different elements individually visible, making errors visually distinct and easier to detect.

Completing the relevant element will cause the required color and font settings to be applied to it.
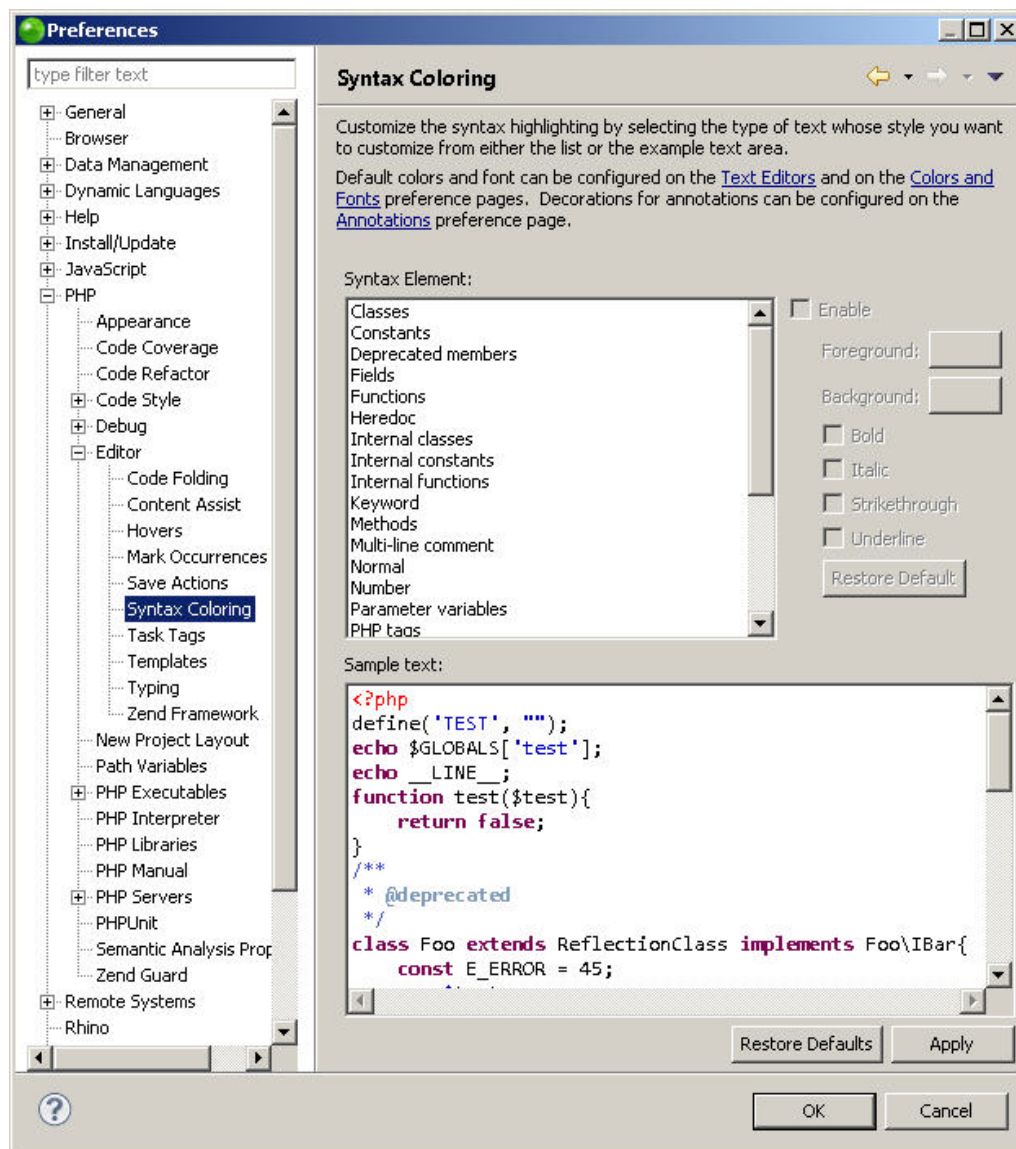
Note:

All elements have a default Background of white.

The color and font settings can be configured from the [Syntax Coloring preferences](#) page, accessed from **Window | Preferences | PHP | Editor | Syntax Coloring**.

Note:

Syntax Coloring will also be available for JavaScript elements if JavaScript support was enabled for the project. See [Enabling JavaScript Support in PHP Projects](#) for more information.

To configure JavaScript Syntax Coloring preferences, go to **Window | Preferences | JavaScript | Editor | Syntax Coloring**.

Automatic Completion

Zend Studio can be set to automatically complete certain types of patterns. To use the auto-complete function, type the opening character of the pattern in the editor and press **Enter**. The pattern will be automatically completed by the relevant characters being inserted.

The following types of patterns can be auto-completed:

- "Strings" - Automatically inserts a pair of double quotes.
- (Parentheses) and [Square] brackets - Automatically inserts a pair of brackets.
- {Braces} - Automatically inserts a pair of curly brackets
- PHPDoc and comment regions - Automatically inserts the "/** **\ " PHPDoc characters.

The types of patterns that can be auto-completed can be configured from the Typing Preferences page, accessible by going to **Window | Preferences | PHP | Editor | Typing**.

Drag and Drop

The Drag and Drop functionality allows you to click on a selected chunk of code and drop it anywhere in the editor. This not only helps you work more efficiently, but also helps minimize the errors that are created when editing or cutting/pasting your code.

Drag and Drop is available in PHP and JavaScript editors.

To Drag and Drop a chunk of code in an editor:

1. Highlight the chunk of code you want to move with your cursor.
2. Click and hold down your mouse within the highlighted code to grab it, and drag the chunk to the selected line in the editor.
3. Release the mouse to place the chunk of code in its new location.

```
<?php
$releasetag = "M7";
$version = array("Windows 32-bit" => "win32.zip", "Linux x8
$sizes    = array("Windows 32-bit" => "141M", "Linux x86/GTI

print '<a href="http://www.eclipse.org/pdt/downloads/?show
```
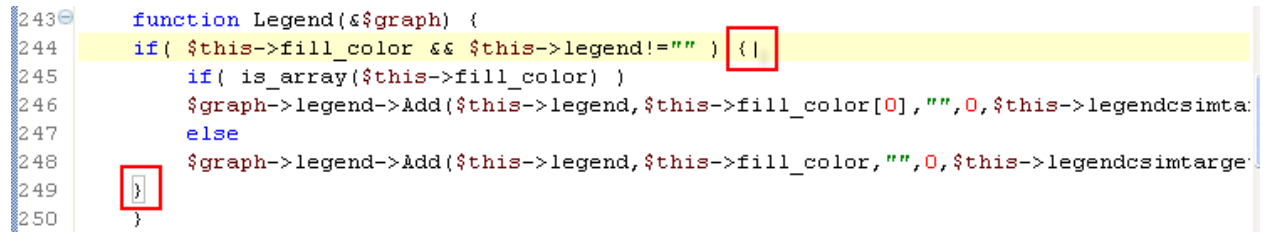
For more information on editors see the Editors topic in the Workbench User Guide.

Matching Brackets

Zend Studio can help you to easily navigate through your script by finding brackets' matching pairs.

To see a bracket's pair, click to the right of the bracket. Its matching pair will be highlighted.

To jump to the matching bracket, press **Ctrl+Shift+P**.

```
243    function Legend(&$graph) {
244    if( $this->fill_color && $this->legend!="" ) {|
245        if( is_array($this->fill_color) )
246        $graph->legend->Add($this->legend,$this->fill_color[0],"",0,$this->legendcsimta
247        else
248        $graph->legend->Add($this->legend,$this->fill_color,"",0,$this->legendcsimtarge
249    }
250    }
```

Mark Occurrences

The Mark Occurrences feature enables you to see where an element is referenced by simply clicking on the element in the editor.

When the Mark Occurrences feature is enabled, all occurrences of the element within the active file will be highlighted, and indicators will be displayed in the annotations bar (to the right of the editor).

In addition, hovering over the element will cause a tooltip to be displayed with the location of all references of the element, including occurrences of the element in other files.
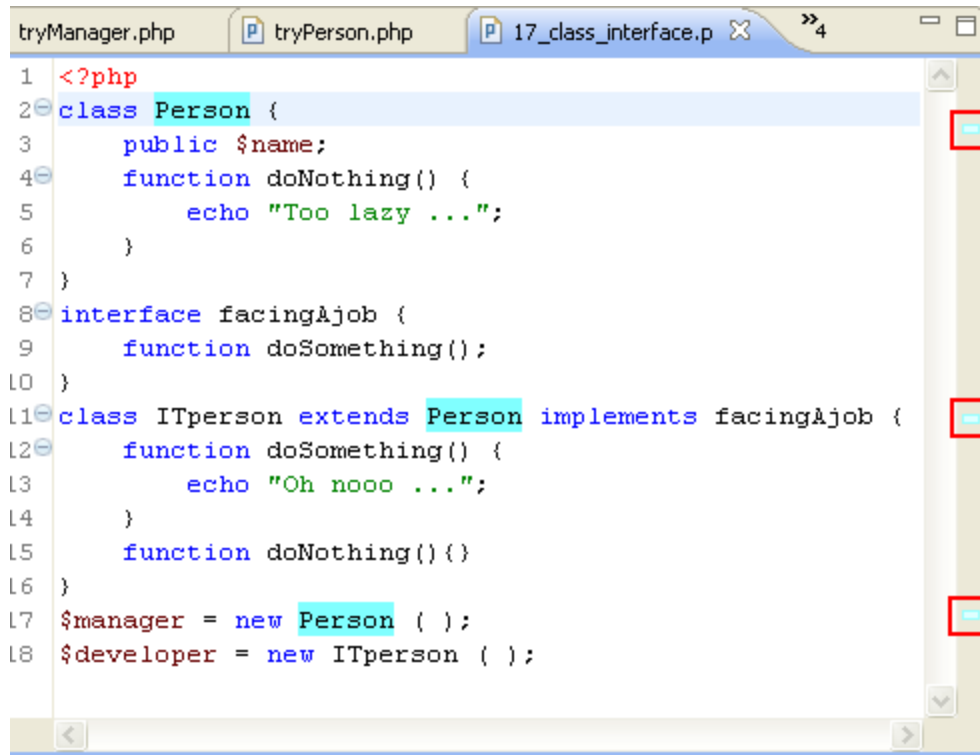
Occurrences can be of 2 types - "Write Occurrence" (for occurrences that are in a write mode) or "Read Occurrence" (for occurrences that are in a read mode). These will be highlighted in different colors according to the setting configured for PHP elements 'read' and 'write' occurrences in the Annotations preferences page (**Window | Preferences | General | Editors | Text Editors | Preferences**). Here you can also configure the indicators used in the vertical ruler/marker bar (to the left of the editor) and the annnotations bar (to the right of the editor).

The types of elements whose occurrences will be highlighted can be configured in the Mark Occurrences preferences page (**Window | Preferences | PHP | Editor | Mark Occurrences**).
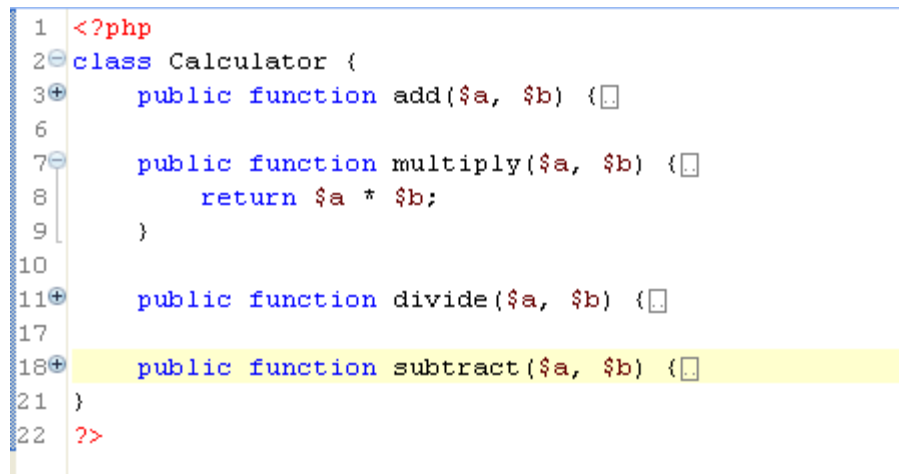
By default, the following types of elements are marked:

- Method Exit - Exit points (throws / return / end of flow) of a method will be marked
- Break / Continue - Scope (for, foreach, while, do-while or switch structure) of a break / continue statement will be marked
- Local variables
- Global variables
- Parameters
- Functions
- Methods
- Fields
- Class Constants
- Constants (defined)
- Class/Interface
- HTML Tags

Code Folding

Code Folding collapses or "folds" the display of a block of code. The editor will then display only the first line of code instead of the entire block. This reduces visual clutter by enabling users to selectively hide and display complicated text while still viewing those subsections of the text that are relevant.

Code Folding is applied by default for functions and PHPDocBlocks. You can configure which of these are folded by default through the Folding preferences dialog.

Folded code has a plus sign [+] on the vertical marker bar to the left of the Editor. Clicking this sign will expand the block of code.

Opened, unfolded code has a minus sign [-] on the vertical marker bar to the left of the Editor. Clicking this sign will fold the block of code.

See the Using Code Folding topic for more information.

Code Commenting

Commenting your code involves adding characters (normally slashes and stars) which mark certain areas of code as 'comments'.

Comments are used for reference information only and will not be run as part of your code. It is good programming practice to comment all functions, classes and methods in your code. This helps both the developer and others who might look at the code to understand its purpose.

Comments can be either single line or multi-lined:

- Single lined comments will have the following format:

```
// This is a single line comment.
```

- Multi-lined comments will have the following format:

```
/* This is
a comment
on multiple
lines
*/
```

Note:

Code Commenting will also be available for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information.

phpDoc Block Comments

Zend Studio offers a preset means for adding phpDoc comments to files by providing an input line when including statements, classes, class variables, and constants to the code. Developers are prompted to immediately add a description ensuring that the added elements are documented in their context and in real-time.

phpDoc blocks are descriptive comments that are part of the application code. They are used to describe the PHP element in the exact location in the code where the element appears. The block consists of a short description, long description, and phpDoc tags.

Example:

When creating a phpDoc Block commen for the following function:

```
function add ($a, $b) {
    return $a + $b;
}
```

the following comment will be created:

```
/**
 * Enter description here...
 *
 * @param unknown_type $a
 * @param unknown_type $b
 * @return unknown
 */
```

The comments should now be edited with the relevant description and parameters.

Descriptions that are added for a code element are also automatically added to the Content Assist bank so that the next time the code element is used it is readily available from the Content Assist list. The element's descriptions will also appear in the Outline view.

Note:

Zend Studio offers Content Assist support for magic members declared in code comments. See the Content Assist concept for more information.
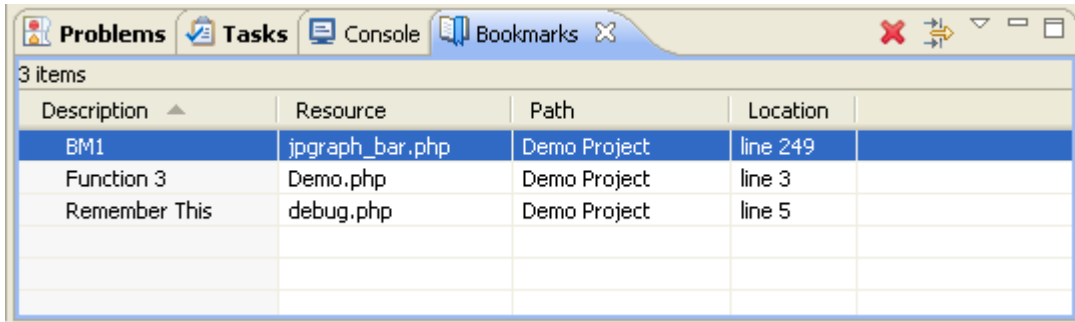
phpDoc blocks also serve as the input for creating a PHPDoc.

Bookmarks

Bookmarks can be used as placeholders within your scripts to allow easy navigation to pre-defined places within your scripts and resources.

Bookmarks are indicated by a bookmark icon in the vertical marker bar to the left of the editor.

Bookmarks are displayed in the Bookmarks view, which can be opened by going to **Window | Show View | Bookmarks**.

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Hover Support

Hovering over an element will cause a tooltip to appear with information about that element, containing the location of its declaration and any additional information (description, parameters, etc.) contained in the element's relevant PHPdoc comment:
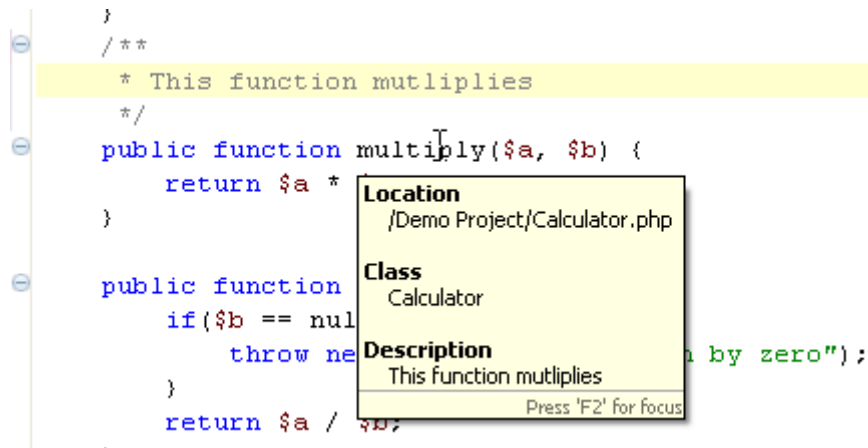
Hover information also includes other information such as overrides and returns for methods, types, and other elements:
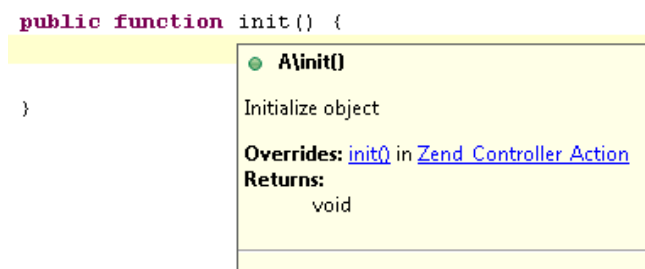
Holding down **Ctrl** while hovering over an element will also show you everything contained within that element:
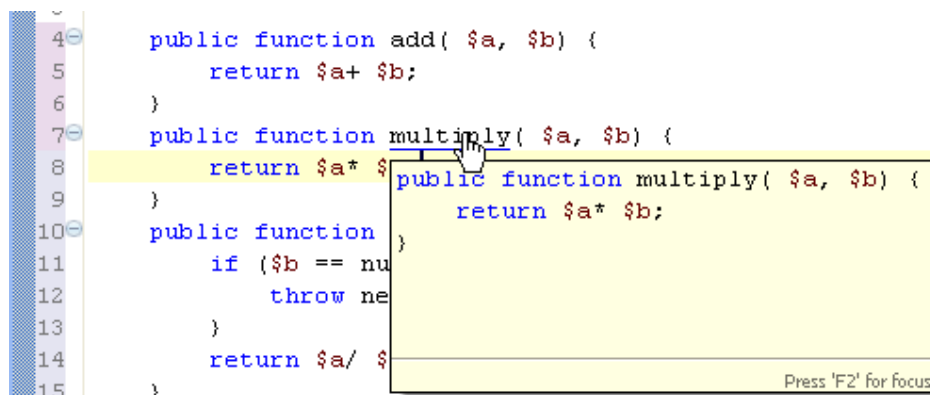
When hovering, press **F2** for the Hover tooltip to come into focus. This ensures that it is displayed even when not hovering and enables you to select the text from within it.

Clicking on an element while hovering will take you to that element's declaration. See Using Smart Goto Source for more information.

Configuring Hover Preferences

To configure your hover settings, go to the Hover Preferences page, accessible by going to **Window | Preferences | PHP | Editor | Hovers**.

Note:
Hovering will also be enabled for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information. JavaScript hover settings can be configured from the JavaScript Hovers preferences page, accessible by going to **Window	Preferences	Web	JavaScript	Editor	Hovers**.

Override Indicators

'Override Indicators' are markers that display special decoration icons to indicate methods that override or implement other methods from a super-type's class/interface.

The override indicators will be displayed as triangles in the marker bar (the vertical ruler to the left of the editor) next to the method's definition.
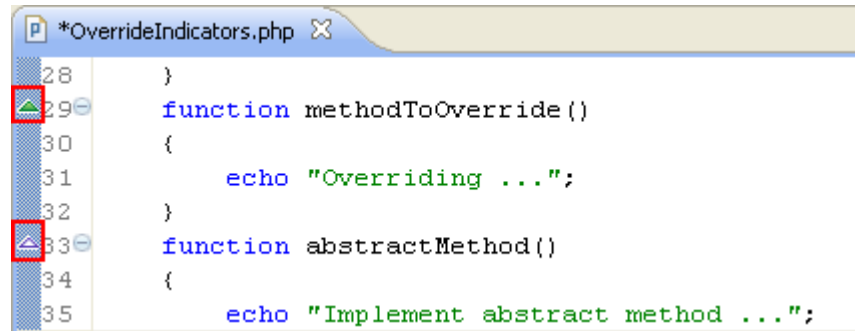
Green triangles indicate an overridden method, while a white triangle indicates an implemented method.

Hovering over a triangle displays a tool-tip that reveals the overridden or implemented method's super-type. Clicking the triangle will open the type in an editor.
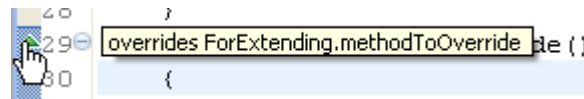
Note:

Markers will not be displayed if the methods do not comply with the overriding rules.

These are as follows:

- You cannot override a non-static method with a static method.
- The overriding method must not be more restrictive than the overridden method
- 'private' methods are not inherited and cannot be overridden.

Override Indicators Preferences

The display preferences for the override indicators can be configured from the Annotations preferences page (**Window | Preferences | General | Editors | Text Editors | Annotations**).

Select the Override Indicators (org.eclipse.php.ui.overrideIndicator) option to configure the Override Indicators for PHP.

PHP Working Sets

PHP Working Sets are groups of PHP resources which allow you to view or apply actions to a group of pre-defined elements. Workings Sets group elements under one title so that they can be easily selected when viewing resources or carrying out actions. Elements included in a PHP Working Set can include any PHP project, folder or file.

Example for using Working Sets:

Selecting the 'Select Working Set' option in the PHP Explorer view's context menu allows you to choose specific Working Sets (i.e. group(s) of projects/files) to be displayed in the view.
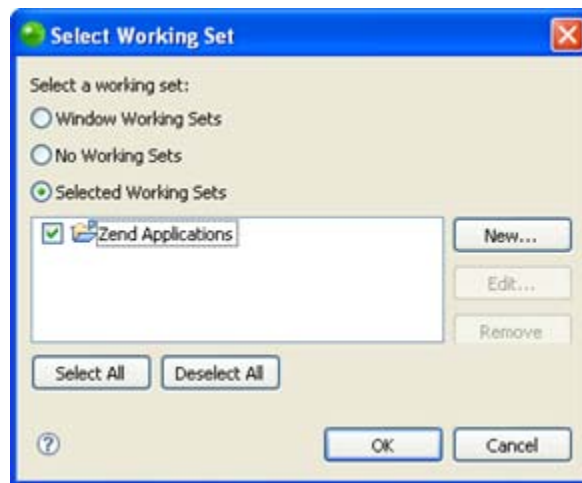
Projects/files not in the selected Working Set will not be displayed in the view.

If you have not created PHP Working Sets, see Creating PHP Working Sets for more information.

Type Hierarchy

The Type Hierarchy views display the hierarchy for a given type (a class name, interface name or class methods, constants and fields.). This allows you to view an element's supertypes (types higher in the hierarchy) or subtypes (lower in the hierarchy) within a tree structure, providing you with an overview of your element's structure.

A type hierarchy can be displayed in two ways:

1. **Quick Type Hierarchy view -** Displays a simple hierarchy view within the editor.
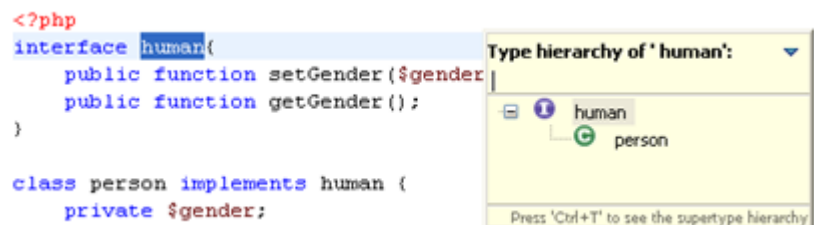
2. **Type Hierarchy view -** A standalone view.
   - The Type Hierarchy Tree - Displays the type's supertypes and/or subtypes.
   - Member list pane - Displays the type's members.
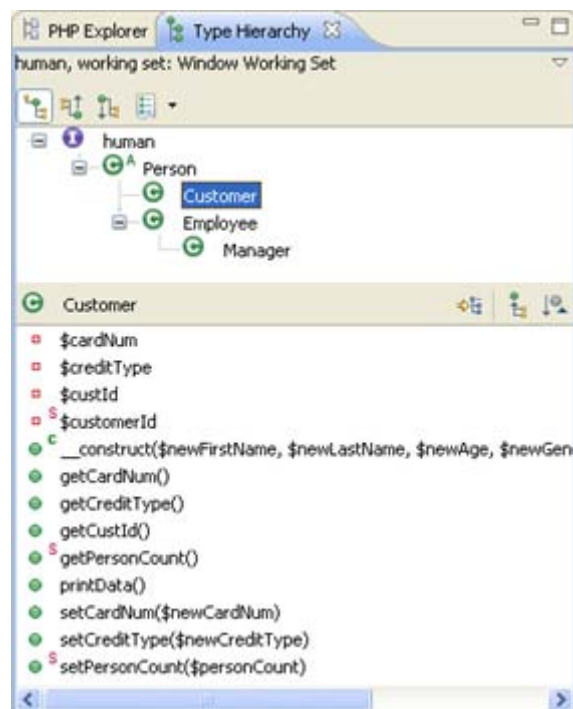
See Type Hierarchy View for more information.

PHP Manual Integration

Zend Studio can integrate with PHP Manual sites in order to get the most up-to-date PHP information. The PHP Functions view lists all PHP functions contained within the PHP manuals, and can be used in order to easily add these functions into your scripts. To add a function to your code, simply place the cursor in the required position in the Editor and double-click the required element from the list.
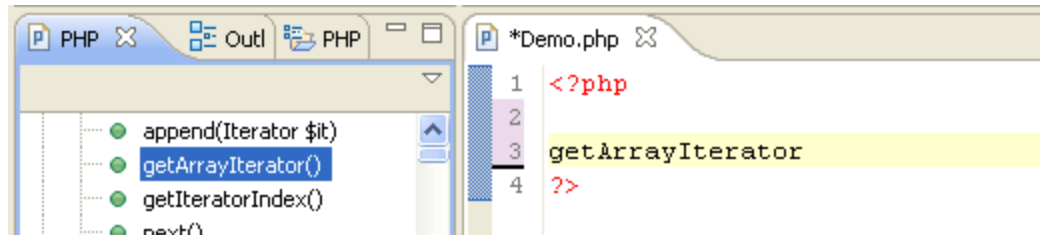
You can open a PHP manual site with an explanation about most of the functions on the list by right-clicking a function in PHP Functions view and selecting **Open Manual**.

A new browser window will open with an explanation of the function from the PHP Manual site.

Note:

Some functions will not have a description assigned to them in the PHP Manual site. In this case, a browser will open with a 'Cannot find server' error message.

PHP Manual sites can be added and edited from the PHP Manual Preferences page.

In addition, The PHP Functions view can be used in order to easily add functions into your scripts. To add a function to your code, simply place the cursor in the required position in the Editor and double-click the required element from the list.

To open a PHP Manual, right-click in the Editor and click **Open PHP Manual** -or- press **Shift + F2**.

Real Time Error Detection
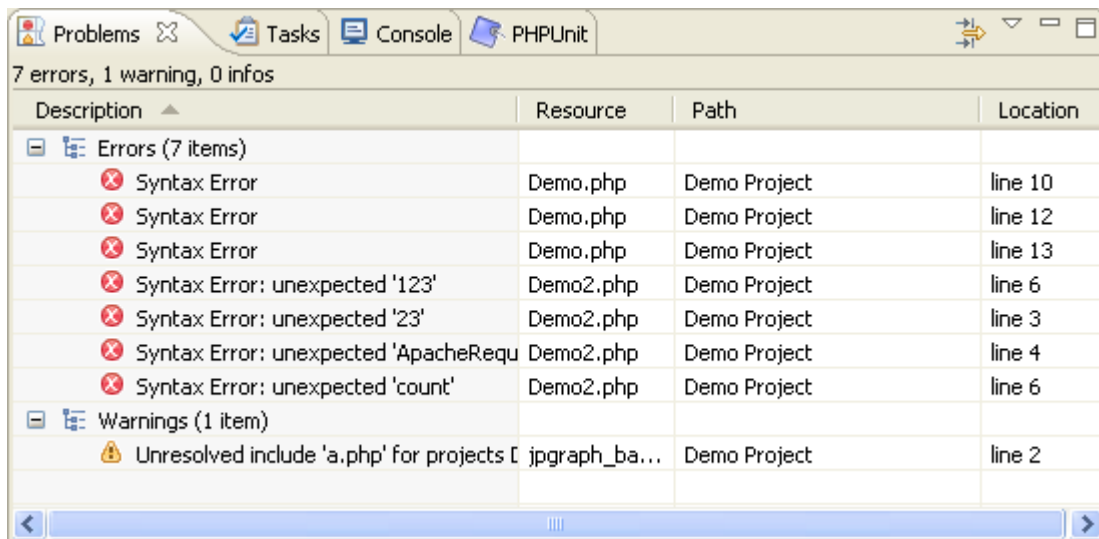
Zend Studio automatically highlights errors and problems in your PHP script.

Errors and warnings will be displayed in the Problems view, accessed from **Window | Show View | Other | General | Problems**.

In addition, error icons and warning icons will be displayed in the vertical marker bar to the left of the editor window, as well as next to the relevant project in PHP Explorer view (these will be indicated in the PHP project/file icons - e.g.).

All warnings, errors and problems in open projects will be logged in the Problems view, which displays the following information:

- Description - A detailed description of the error.
- Resource - The name of the resource containing the problem.
- Path - The name of the Project containing the resource.
- Location - The line number(s) of the error within the file.
- Type - The Type of error occurring.

The Problems view groups problems according to Errors, Warnings or Info.

Double-clicking on an error in the Problems view will take you to the relevant location in the Editor.

If the Problems view is not displayed, go to **Window | Show View | Problems**.

Semantic Analysis

Zend Studio's Semantic Analysis mechanisms can detect problems beyond the regular parsing warnings and errors. This helps developers to analyze static source code to enforce good coding practices and scan PHP code. The Semantic Analysis feature achieves this functionality by attempting to reconcile problematic code and locating unreachable code (code that has been defined but is not used or with empty variables).

Semantic Analysis warning messages help to ensure that your code is written and formatted in a way that will result in optimal performance for your script. In addition, it supplies you with practical suggestions for improving the code.

Semantic Analysis problems will be displayed in the Problems view and be indicated in the code with warning and error icons (see Real Time Error Detection for more information).

Quick Fix

The Semantic Analysis Quick Fix option enables you to easily change your problematic code according to suggestions supplied by the Semantic Analysis mechanism. This includes suggested actions for element creation or implementation.

Example:
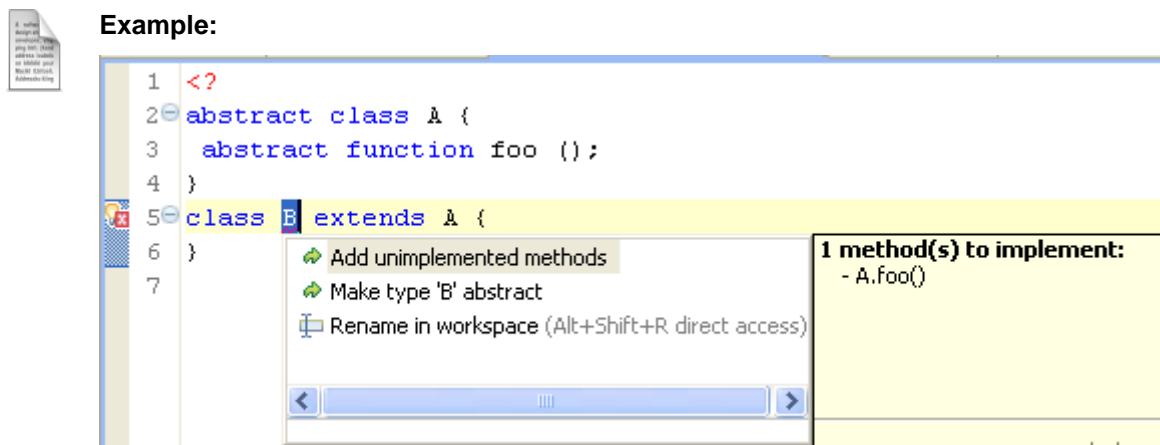
See Applying Quick Fixes for more information.

Preferences

To enable/disable Semantic Analysis and to configure which occurrences will trigger warning or error messages, go to the Semantic Analysis Preferences page, accessible by going to **Window | Preferences | PHP | Semantic Analysis**.

Local History

A Local History of a file is maintained when you create or modify and save a file. Each time you edit and save a new version of a file, a copy of it is saved in local history.

Each state in the local history is identified by the date and time the file was saved.

This allows you to compare your current file state to a previous state, replace the file with a previous state, or restore a file that was deleted.

Note:
There is no Local History associated with Projects or with Folders.

See the Using Local History topic for more information.

CVS

A Concurrent Versions System (CVS) repository is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to revert file and project states back to previous versions.

CVS functionality can be accessed from the CVS Repository Exploring Perspective, accessed by going to **Window | Open Perspective | Other | CVS Repository Exploring**.

See [Working in a Team Environment with CVS](#) in the Workbench User Guide for more information.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help	Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

SVN

SVN, or Subversion, is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions.

SVN functionality can be accessed from the SVN Repository Exploring Perspective, accessed from **Window | Open Perspective | Other | SVN Repository Exploring**.

See the [Subversive User Guide](#) for more information on SVN.

Zend Framework Development

Zend Framework is Zend's open source, object-oriented web application framework implemented in PHP 5.

For information on Zend Framework, visit the Zend Framework site at: http://framework.zend.com or http://framework.zend.com/manual/en for the Zend Framework Reference manual.

Zend Framework Project and Element Creation

Zend Studio's Zend Framework integration functions allow you to create a new Zend Framework Project that is organized into Framework's Controller-Model-View system. The project will have Zend Framework's libraries added to its include path, allowing access to all Zend Framework's elements.

The new Zend Framework project that is created in Zend Studio will contain basic files for a simple "Hello, World!" program.

When a Zend Framework Project is created, you will be prompted to open the Zend Framework Perspective, containing the MVC Outline view.
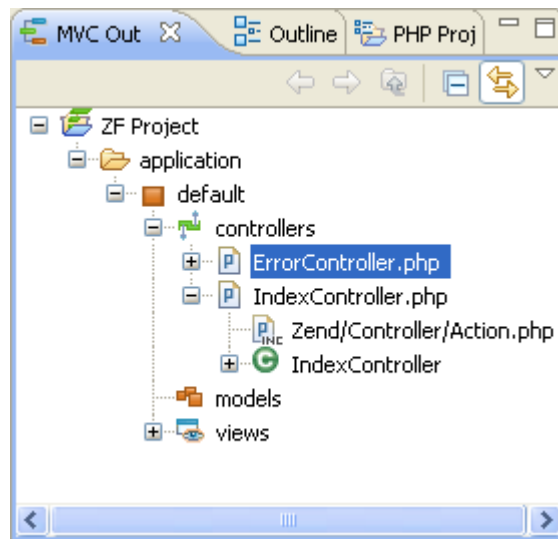
The MVC Outline view provides an outline of all controller, model and view classes, files, variables and related functions.

Note:

To manually open the Zend Framework Perspective, go to Window | Open Perspective | Zend Framework.

Once you have created a Zend Framework project, you can use Zend Studio's New Zend Table, Zend View, Zend Controller, Zend Module, Zend View Helper or Zend Action Helper Wizards to create new Zend Framework elements.

Zend Framework Content Assist

Once Zend Framework's libraries are included in a project's include path, its classes, functions, iterators and variables will be available for use from the Content Assist window. Simply press **Z** followed by **Ctrl+Space** in the Editor to view the list of available Zend Framework elements:
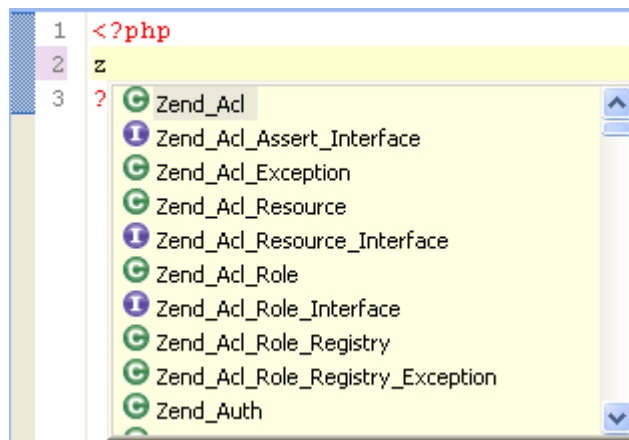
For more on Zend Framework, visit the Zend Framework site at http://framework.zend.com or the Zend Framework Reference Manual at http://framework.zend.com/manual/en. For more external resources, see Useful Links.

Database Connectivity

Zend Studio's Data Tools Platform plugin allows you to connect to, view, edit and run queries on databases. The Data Tools Platform provides connectivity with a number of databases.
Data Tools Platform functionality is accessed through the Database Development Perspective.

For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

For instructions on connecting to and accessing your database, see Connecting to Databases.

Running

Zend Studio allows you to run the applications you are working on from the workbench. This allows you to run and test your applications during development.

Zend Studio includes several different methods of running your files and applications:

PHP Script Local Running

Allows you to run files situated in your workspace. This enables you to locally validate freshly developed code before deploying to a Web server. Using this option, you can run applications which do not require user input or responses from the server.

Note:
When running internal files the Zend Studio internal debugger uses its own PHP executable that was installed together with Zend Studio.

See [Running PHP Scripts Locally](#) for more information.

PHP Web Page Running

Allows you to run applications situated on a server, including any required interactive user input.

The PHP Web Page Run dialog has an option to give the files you are working on first priority when running, using the "Local Copy" option. This means that, when possible, file content is taken from the files situated on your Workspace. This prevents you from having to upload the latest revisions.

Note:
It's recommended that your local project structure reflect the project structure on your server.

See [Running PHP Web Pages](#) for more information.

Note:
Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.
The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

Debugging

The Zend Studio debugging function allows you to test your files and applications and detect errors in your code.

The debugger allows you to control the execution of your program using a variety of options including setting breakpoints, stepping through your code, and inspecting your variables and parameters.

See above for a list of the different debugging methods Zend Studio includes.

PHP Script Local Debugging

Allows you to debug files on your workspace using Zend Studio's internal debugger.

The Internal Debugger enables developers to locally validate freshly developed code before deploying to a web server. The internal option means that files located on your workspace can be debugged. When debugging internal files the Zend Studio Internal Debugger uses its own PHP executable that was installed together with Zend Studio.

See Locally Debugging a PHP Script for more information.

PHP Web Page Debugging

Allows you to debug applications situated on a server. It allows you to debug whole applications, including any required interactive user input.

The PHP Web Page Debug has an option to give the files you are working on first priority when debugging, using the "Local Copy" option. This means that, when possible, file content is taken from the files situated on your Workspace. This prevents you from having to upload the latest revisions.

Note:

It's recommended that your local project structure reflect the project structure on your server.

See Debugging a PHP Web Page for more information.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

URL Debugging

Allows you to enter a URL to debug an application on a server. Only server files will be debugged, so the files do not need to exist locally in your Workspace.

See Debugging a URL for more information.

Note:

Your server must be running the Zend Debugger in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads .

Toolbar Debugging

Allows you to Debug files and applications directly from your browser using the Zend Debugger Toolbar, available for download from the Zend Studio Resources site (http://www.zend.com/en/products/studio/downloads).

See Debugging Using the Zend Debugger Toolbar for more information.

Profiling

The Zend Profiler displays a breakdown of the executed PHP code in order to detect bottlenecks in scripts by locating problematic sections of code. These are scripts that consume excessive loading-time. The Profiler provides you with detailed reports that are essential to optimizing the overall performance of your application.

See above for a list of the five different profiling methods Zend Studio includes.

PHP Script Local Profiling

Allows you to profile files on your workspace using Zend Studio's internal debugger.

The Internal Debugger enables developers to locally validate freshly developed code before deploying to a web server. The internal option means that only files located in local directories can be profiled. When profiling internal files the Zend Studio Internal Debugger uses its own PHP executable that was installed together with Zend Studio.

See [Locally Profiling a PHP Script](#) for more information.

PHP Web Page Profiling

Allows you to profile applications situated on a server. It allows you to profile whole applications and projects.

The PHP Web Page Profile setting has an option to give the files you are working on first priority when profiling, using the "Local Copy" option. This means that, when possible, file content is taken from the files situated on your Workspace. This prevents you from having to upload the latest revisions.

Note:

It's recommended that your local project structure reflect the project structure on your server.

See [Profiling a PHP Web Page](#) for more information.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

URL Profiling

Allows you to enter a URL to profile an application on a server. Only server files will be profiled, so the files do not need to exist locally in your Workspace.

See [Profiling a URL](#) for more information.

Once a Profile session has been executed (see Using the Profiler for more information on how to execute a profiling session), various views in the Profiling Perspective provide information on the performance of your script. See PHP Profile Perspective for more information on the various views.

Toolbar Profiling

Profile files and applications directly from your browser.

See Profiling Using the Zend Debugger Toolbar for more information.

Profiler Views

Profiler Information view

Provides general information on the profiling duration and date, number of files constructing the requested URL and more. In addition, it displays a Time Division Pie Chart for the files in the URL. The right side displays time division in a pie chart and the left side provides the following information:
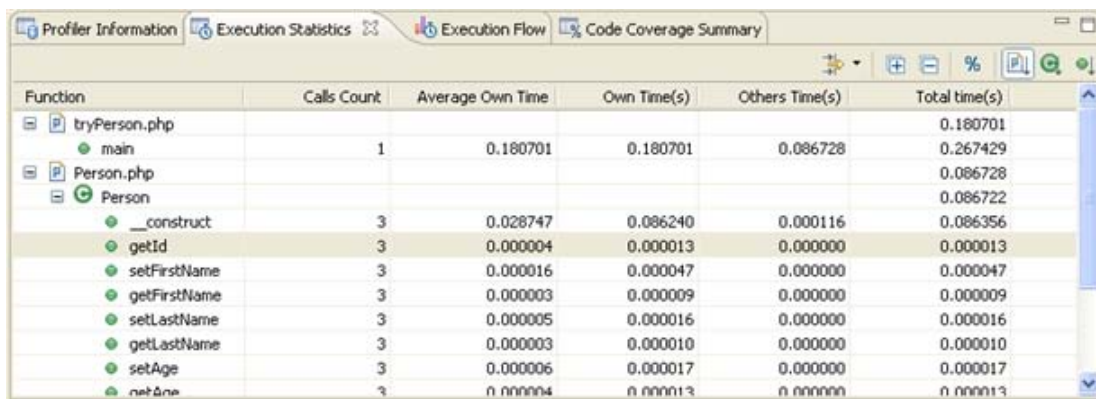
- URL - The URL analyzed (if applicable).
- Query - The specific query parameters.
- Path - The location of the first file called.
- Total Request Time - Total processing time for the entire page.
- Number of Files - Number of files processed.
- Date - Date and time that the profiling took place.
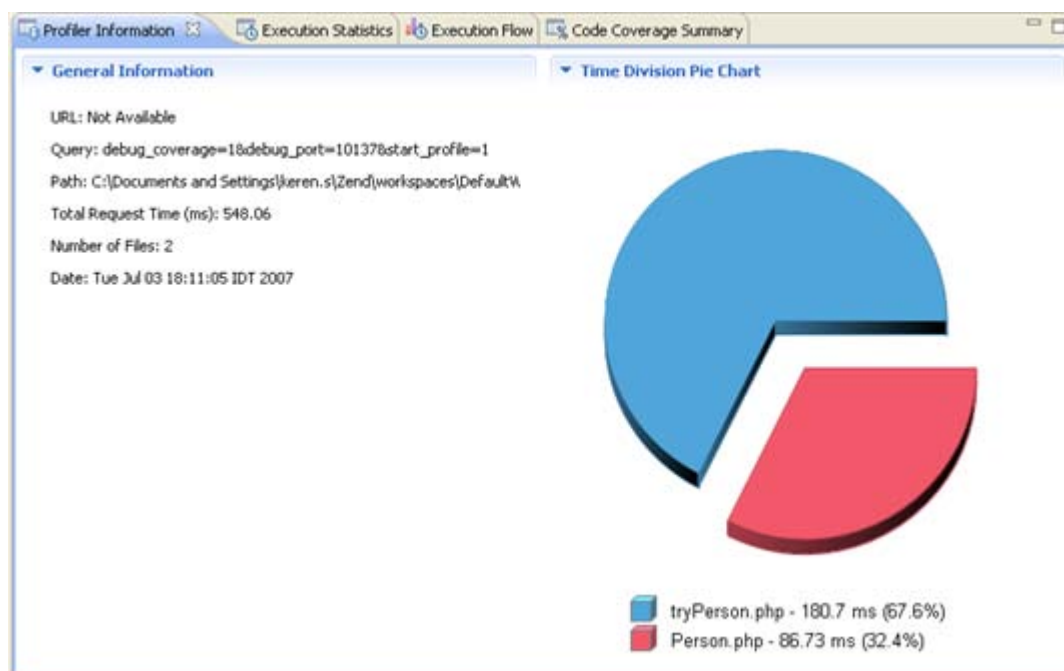
Execution Statistics view

Displays the list of files that were called during the profiling process and detailed information on processing times for elements within the files. The window contains statistics relevant to each element as follows:

- Function - The name and location of the function.
- Calls Count - The number of times that the function was called.
- Average Own Time - The average duration without internal calls.
- Own Time(s) - The net process duration without internal calls.
- Others Time(s) - Time spent on calling other files.
- Total Time(s) - The total time taken to process.

Function	Calls Count	Average Own Time	Own Time(s)	Others Time(s)	Total time(s)
tryPerson.php					0.180701
main	1	0.180701	0.180701	0.086728	0.267429
Person.php					0.086728
Person					0.086722
__construct	3	0.028747	0.086240	0.000116	0.086356
getId	3	0.000004	0.000013	0.000000	0.000013
setFirstName	3	0.000016	0.000047	0.000000	0.000047
getFirstName	3	0.000003	0.000009	0.000000	0.000009
setLastName	3	0.000005	0.000016	0.000000	0.000016
getLastName	3	0.000003	0.000010	0.000000	0.000010
setAge	3	0.000006	0.000017	0.000000	0.000017
getAge	3	0.000004	0.000013	0.000000	0.000013

Note:
Click the 'Show as percentage' button on the toolbar to see the statistics as percentages rather than times.
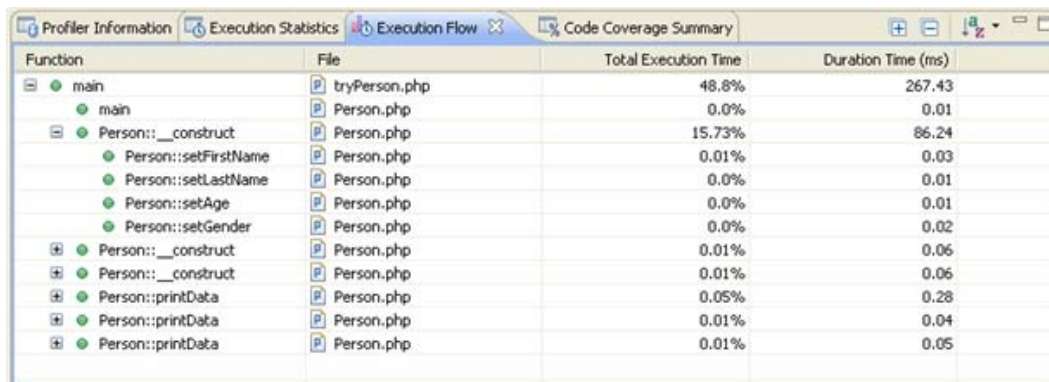
Right- clicking a function in the list gives you the option to 'Open Function Invocation statistics'. This will open a view with statistics about the selected function, the functions it was invoked by and functions that it invoked.

Execution Flow view

Shows the flow of the execution process and summarizes percentages and times spent on each function.

- Function - Function name.
- File - The file in which the function is located.
- Total Execution Time - Percent of time taken per function.
- Duration Time - Time taken per function in milliseconds.

Right-clicking a function in the list gives you the option to:

- View Function Call - Will open the selected function call in the editor.
- View Function Declaration - Will open the selected function declaration in the editor.
- Open Function Invocation statistics - Will open a view with statistics about the selected function, the functions it was invoked by and functions it invoked.

Code Coverage Summary
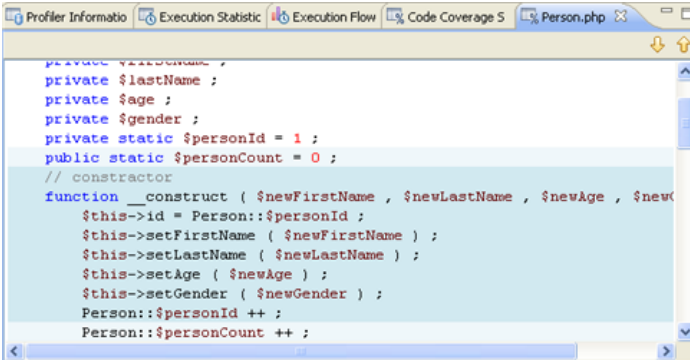
Summary of how many lines of code were covered during the Profiling process.

- Element - The file / project that was called.
- Covered Lines (Visited / Significant / Total) - Percentage of lines covered within each file. (Visited = Number of lines covered / Significant = number of lines that were significant to the function call / Total = Total number of lines in the file.)

Clicking on the 'Covered lines' percentages will open an editor containing the file, with the covered lines highlighted:

Breakpoints

Breakpoints allow you to set places in the code at which the debugging process will pause. Setting a breakpoint in your script will cause the debugger to stop at the specified line, allowing you to inspect it.

Breakpoints are represented by a blue ball in the vertical ruler to the left of the editor.
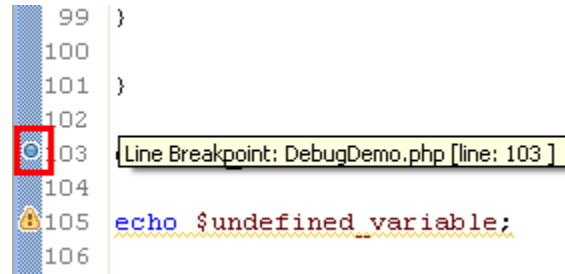
During the debugging process, breakpoints can be monitored in the Breakpoints View.

Conditional Breakpoints

Conditions can be added to breakpoints so that the debugger will only stop at them under certain conditions. These can be any valid boolean expressions.

Conditions can be added to breakpoints through the PHP Conditional Breakpoints Properties dialog.

Conditional Breakpoints are represented by a blue ball with a question mark.
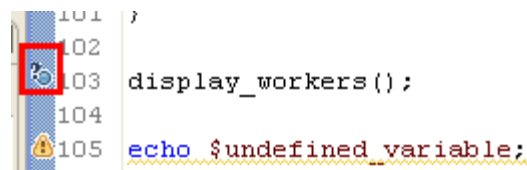
Example:

If a breakpoint has a condition of "$var > 5", the debugging process will stop at the breakpoint only if the value of $var is greater than 5.
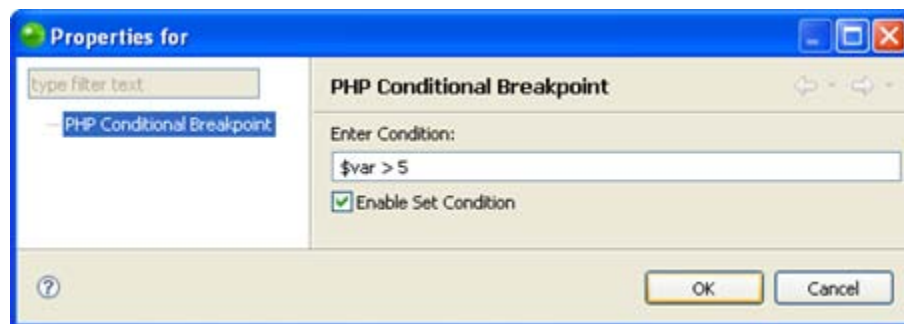
PHP Include Path

The PHP Include Path is a set of locations that is used for finding resources referenced by include/require statements.

Elements added to a project's Include Path affect the following:

- Running/Debugging / Profiling - Files that are called with a 'relative path' will be searched for during runtime in the resources and order specified in the include path.
- Go to source / content assist for include statements - Files that are called with a relative path" will be searched for according to the resources and order specified in the project's include path.
- Content Assist - Adding projects/libraries to a project's Include Path will make elements defined within the included projects/libraries available as Content Assist options to the project.

In 'include'/'require' calls, file locations can be defined in three ways:

a. **Absolute Path**- The exact file location is specified (e.g. C:\Documents and Settings\MyProject\myfolder\a.php). During Remote PHP Script (PHP Server) or PHP Web Page Running/Debugging /Profiling when the 'Local Copy' option is selected under the 'Source Location' category in the Advanced tab, the Path Mapping mechanism will be activated.

b. **Relative to the Current Working Directory** - File names preceded with a "./" or a "../" These will only be searched for relative to the PHP 'Current Working Directory'. You can find out the location of your Current Working Directory by running the command "echo getcwd()".

c. **Relative Path** - Only the file name or partial path is specified (e.g. /myfolder/a.php). In this case, Zend Studio will search for the file's path according to the resources and order configured in the project's Include Path.

   If the path of the file being searched for exists in more than one location, the file that is called will be the first one Zend Studio encounters during the search process.

   By default, the order in which Zend Studio searches for the file's path is as follows:

   i. Projects in the "debug target" (the first file to be debugged) project's Include Path, according to the order in which they are listed. If a project specified in the Include Path refers to other projects/libraries in its own Include Path, the file path will be searched for there before the search process continues.

   ii. The "debug target" file's project.

See http://il2.php.net/manual/en/function.include.php for more on PHP's search mechanism.

PHP Build Path

The PHP build process scans all resources that are on the project's PHP Build Path so that elements defined within them can be made available for Content Assist options and Refactoring operations . This is done in order to get notification about changes in the file system (e.g. files added/removed from the project, code changes etc.) and in order to maintain the code database (user classes, functions, variables etc.).

Configuring the project's Build Path allows you to select PHP resources to include/exclude from this process. Rather than automatically scanning all resources within the project, configuring the Build Path allows you to select which resources will be scanned (letting you, for example, exclude folders containing images, JavaScript files or other types of files not containing PHP code). This can significantly speed up the build process.

Path Mapping

Zend Studio enables you to map server paths to local paths while Debugging and Profiling on a server. Once a Path Map has been defined, if a file is called from the defined location on the server during debugging/profiling, its content will be taken from the set corresponding location on the file system/workspace.

Note:

Path Mapping is only activated during Remote PHP Script (PHP Server) Debugging/Profiling, or PHP Web Page Debugging /Profiling when the 'Local Copy' option is selected under the 'Source Location' category in the Advanced tab .

Example:
The server path 'C:\Documents and Settings\MyProject' has been mapped to '/MyProject' on the Workspace:
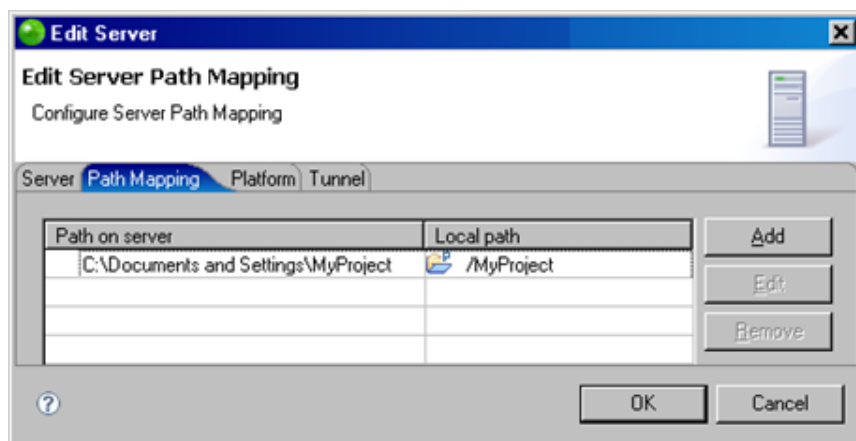
During Remote PHP Script Debugging , a file is called from location 'C:\Documents and Settings\MyProject\a.php':

```php
<?php
echo "a.php";
include ('C:\Documents and Settings\MyProject\a.php');
?>
```

The file content for a.php will be taken from the a.php file located in the 'MyProject' project, situated on the Workspace.

Note:

Server Path Maps can be viewed and defined in the Path Mapping tab of the PHP Servers Preferences page.

Defining Path Maps

Path Maps can be defined in three ways:

1. Manually, through the PHP Servers Preferences page or the Importing a Zend Server Event File wizard. See Managing Path Maps for more information.

2. Automatically whenever a file is debugged /profiled - A Path Map is automatically set between the path to the debug target's parent project (the parent project of the file from which the debugging process has been launched - e.g. C:\Workspace\MyProject) and the debug target's project in the Workspace (e.g. MyProject).

3. Through the Path Mapping dialog. This is launched during debugging /profiling whenever a file defined with an absolute path (See 'Include Paths' for more on absolute file locations) is called . In this scenario, a Path Mapping dialog will appear with a list of 'similar files' to the one being called.

   'Similar' files are files with the same name as the called file that are situated in the following locations:

   - Files in the project from which the file was called.
   - Files in projects that are in the Include Paths of the project from which the file was called.
   - Files that are open in a Zend Studio editor.

> **Note:**
> If the debug/profile session was triggered from the Zend Debugger Toolbar, all files in the Workspace with the same name will be listed.
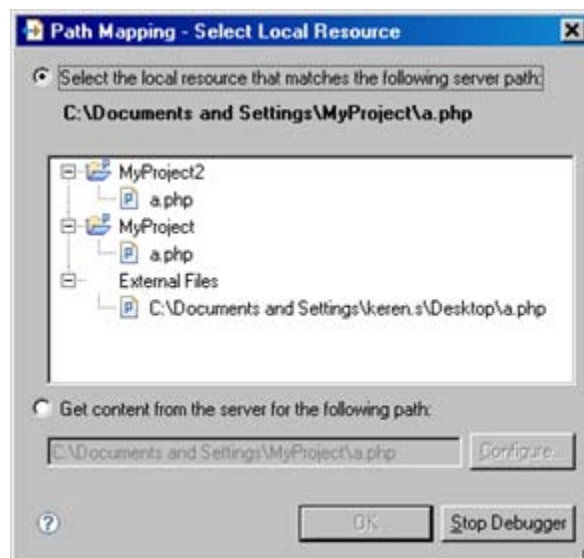
> **Note:**
> The dialog will not appear if a Path Mapping to the called location has already been defined.

Selecting a file from the list results in a Path Map being created between the called remote file's parent folder and the parent folder of the 'similar' file selected from the list. This means that every time a file is called from the same parent folder, its content will be taken from the file situated in the selected Workspace/local folder .

If none of the options in the Matching items list represent your desired file location, you may select the 'Get content from the server for the following path' option. This means that whenever this path is called during the debugging/profiling process, it will only be searched for on the server. (This is done using PHP's search mechanism - see http://il2.php.net/manual/en/function.include.php for more information.) You can click Configure to modify the path to include any parent or child directories.

Note:

Selecting the 'Get content from the server for the following path' option will only affect the current Debug / Profile session. No Path Mapping will be defined.

Zend Browser Toolbar

The Zend Browser Toolbar is an external add-on to Internet Explorer and Firefox browsers which allows you to initiate debugging or profiling sessions directly from your browser. Debugging / profiling through the Zend Browser Toolbar will launch a remote debug / profile session in Zend Studio.
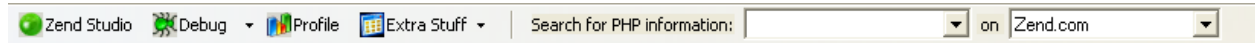
The Zend Browser Toolbar contains the following options:

- **Zend Studio** - Opens Zend Studio. Ensure the Zend Studio .exe file is configured in the Zend Toolbar Settings.
- **Debug** - Launches a Debugging session in Zend Studio.
- **Profile** - Launches a Profiling session in Zend Studio.
- **Extra Stuff** - Provides access to Zend Browser Toolbar settings, and links to useful Zend and PHP Information.
- **Search for PHP Information** - Allows you to quickly and easily search the web for PHP information.

See Installing and Configuring the Zend Browser Toolbar for information on how to get started with the Zend Browser Toolbar.

Tunneling

Tunneling provides a means of persistent connection between Zend Studio and a remote server situated behind a Firewall or NAT. After creating a Tunnel, all communication between Zend Studio and the server can go through that tunnel, instead of assigning more communication ports for the Debug/Profile Sessions made on the remote server.

Note:

A communication tunnel cannot currently be created to a Windows server.

[This persistent connection operates even when separated by a firewall. The advantage of this method is that it is possible to use the Zend Studio Integration on several nodes at once. For example, debugging an entire cluster of machines behind a load-balancer over a single debugger connection to Zend Server's Studio Server component.

The technology is based on two functional elements:

- The Zend Studio that includes an internal Web server that listens on the local host on a specific Auto Detection port.
- Zend Server automatically evaluates Zend Studio's Auto Detection port, by evaluating Zend Studio's settings. These are the Tunnel Settings that are defined in Zend Server.

The tunnel communication port should be used in the following circumstances:

1. When debugging or profiling files on a remote server which is behind a firewall or other security device.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

2. Establishing communication between Zend Studio and Zend Server when Zend Server is running on a remote server which is behind a firewall or other security device. The communication between Zend Studio and Zend Server facilitates the integration that combines Zend Server's event reporting capabilities with Zend Studio's editing, debugging and profiling features. This enables the viewing and debugging/profiling of Zend Server events in Zend Studio .

   See Zend Server Integration for more information.

Note:

To find out whether your server is situated behind a firewall, contact your System Administrator.

To set up a tunneling connection, several configuration settings need to be defined both in Zend Studio and on your server's debugger (this can be done through Zend Server or your php.ini file). See Setting Up Tunneling for more information.

Zend Server

Zend Server is a complete, enterprise-ready Web Application Server for running and managing PHP applications that require a high level of reliability, performance and security. It includes the most reliable and up-to-date version of PHP, tested PHP extensions, database drivers and other enhancements. Zend Server comes bundled with Zend Framework (the leading open-source PHP framework), Apache and MySQL.

Zend Server provides an optimal environment for developing and deploying your PHP applications. See Setting Up Remote Debugging for information on how to set up your Zend Server for debugging with Zend Studio.

See the Zend Server site (http://www.zend.com/products/server/) to learn more about and download Zend Server.

Zend Server Integration

Integrating Zend Studio with Zend Server allows you to benefit both from Zend Studio's debugging and profiling functionality and from Zend Server's Event Monitoring capabilities. Zend Server monitors and constantly tests your PHP environment and programs in order to allow you to gain maximum efficiency. Instances of problematic scripts and slow execution are captured by Zend Server as 'events.' Zend Server's integration with Zend Studio means that the problems identified by Zend Server can then be viewed, tested, debugged and profiled in Zend Studio. This can be done through the Zend Server user interface or by importing Zend Server Events into Zend Studio.

Viewing Events from Inside Zend Studio

Zend Server 's events can be viewed and debugged directly from within Zend Studio using the internal browser, which displays the Zend Server User Interface.

See Debugging / Profiling Events from Zend Server for more information.
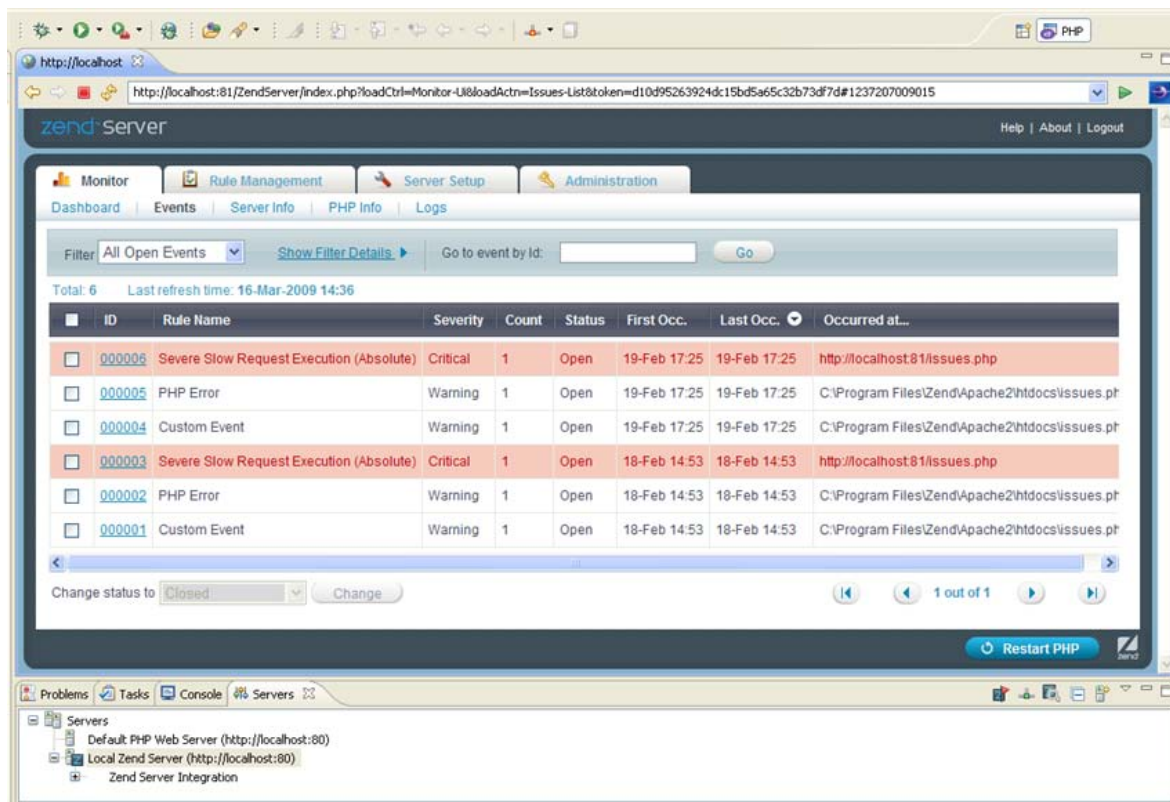
Importing Events into Zend Studio

Zend Server events can be exported from Zend Server as a .xml file, which can then be imported into Zend Studio and debugged. This is useful when the developer using Zend Studio does not have access to the Zend Server on which the event occurred.
See Importing a Zend Server Event File for more information.

Zend Server events can be imported as .amf files using the Code Tracing feature in Zend Studio. Integrating Code Tracing into Zend Studio allows you to open the source of the execution data inside of your environment. Code Tracing uses the Code Tracing Perspective to allow you to view the trace data within your environment.
This feature is useful in resolving time performance issues, memory performance issues, and workflow errors.

Code Tracing

Zend Server Code Tracing captures full execution data (trace data) of PHP applications in real time. The execution data includes function call trees, arguments and return values, function execution duration, memory usage and indication for an executed files name and line of code. This enables you to capture problems when they occur, which eliminates the need to set up environments and reproduce the steps that led up to the failure. Integrating Code Tracing into Zend Studio allows you to view the trace data and open code trace files inside of your environment. Code Tracing uses the Code Tracing perspective to allow you to view the trace data within your environment.

This feature is useful in resolving performance issues, memory usage issues, and functional errors that occur in a production environment.

In order to use the Code Tracing feature, you must first export a Zend Server Event File from Zend Server and then import the Zend Server Event File into Zend Studio. Once the Zend Server Event File has been imported, you can open the code trace and the source of trace data.

Important Note:

For more information about additional features and usability of Code Tracing see Code Tracing in the Zend Server Online Documentation.

The Code Tracing feature in Zend Studio allows you to do the following:

- Export Trace Information (Located in the Zend Server Online Documentation)
- Import a Zend Server Event File
- Open the Source of Trace Data

For more information see Working with Code Tracing.

Note:

The source of the execution data can only be opened in an environment in which the project already exists locally.

PHPUnit Testing

Unit testing is a procedure to test your code to ensure that individual units of source code are working properly and that the right output is being generated. Tests can be run on all or some functions within files, meaning that tests can be conducted before the file has been fully developed. Each test case should be independent of others to ensure that test results can pinpoint the location of the error.

Running unit tests can ensure that your code is stable and functioning correctly, and can help you to diagnose errors.

PHPUnit Test Cases

PHPUnit Test Cases can be created for each class within a file. Running a PHPUnit Test allows you to see which functions within the test are working correctly.

See Creating a PHPUnit Test Case and Running a PHPUnit Test Case for more information.

PHPUnit Test Suites

PHPUnit Test Suites can be created to run several PHPUnit test cases at once. See Creating a PHPUnit Test Suite and Running a PHPUnit Test Suite for more information.

Running PHPUnit Test Cases/Suites

Running a PHP Unit Test Case/Suite will result in the PHPUnit view being displayed, showing the results of the tests that were run.

Debugging PHPUnit Test Cases/Suites

Debugging a PHPUnit Test Case/Suite will result in the PHPUnit view being displayed, showing the results of the tests that were run, in addition to the normal debug functionality. This will also allow you to debug and analyze PHPUnit libraries.

Profiling PHPUnit Test Cases/Suites

Profiling a PHPUnit Test Case/Suite will result in the PHPUnit view being displayed, showing the results of the tests that were run, in addition to the normal profiling functionality. This will also allow you to profile and analyze PHPUnit libraries.

PHPUnit Reporting

Once a PHPUnit Test has been run, Zend Studio can generate a range of reports to easily view and analyze your tests.

The types of reports available are:

- XML - Generates an XML output of your test results. This can be used to create your own reports.
- 'plain.xsl' - Creates a report based on a predefined 'plain.xsl' format. The report shows It shows percentages, elapsed time, total tests, errors and failures. In addition it shows individual report status, message, stack trace and warnings with their stack traces.
- 'packages.xsl' - Creates a report based on a predefined 'packages.xsl' format. This is an extension of the plain report which divides the unit tests into 'packages'. Packages are defined by adding an "@package" annotation to the PHPDoc of test classes. Test classes without an @package annotation will be categorized in a 'default package'.

Custom XSL reports can also be generated by selecting the 'Generate with XSL...' option and selecting a previously created XSL report. These will then be added to the list of available reports.

See Reporting on PHPUnit Test Results for more information.

Refactoring

The Refactoring feature in Zend Studio allows you to:

- Rename and move files and elements within those files, while maintaining the links between the items. Once an element or file has been renamed or moved, all instances of that item within the project will be automatically updated to reflect its new name / location.

- Extract variables - Creates new variables for expressions.
  Creates a new variable assigned to the expression currently selected and replaces the selection with a reference to the new variable.

Example:

Implementing an extract method refactoring on the variable $my_form in the following code:

```php
<?php
if ($this->getRequest()->isPost()) {

            if ($my_form->isValid($request->getPost())) {

                    $model = new Default_Model_Guestbook($my_form->getValues());
                    $model->save();
                    return $this->_helper->redirector('index');
            }
        }
        if (!$my_form->isValid($request->getPost())) {

            $model = new Default_Model_Guestbook($my_form->getValues());
            $model->load();
        }
```

Will result in the following code being created (the changes have been highlighted):

```php
<?php
$isValid = $my_form->isValid($request->getPost());
if ($this->getRequest()->isPost()) {

            if ($isValid) {

                    $model = new Default_Model_Guestbook($my_form->getValues());
                    $model->save();
                    return $this->_helper->redirector('index');
            }
        }
        if (!$isValid) {

            $model = new Default_Model_Guestbook($my_form->getValues());
            $model->load();
        }
```

- Extract Methods - Creates a new method to replace all occurrences of a selected fragment of code.

109

Martin Fowler, the creator of the Refactoring concept, defines it as the following:

"Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Its heart is a series of small behavior preserving transformations. Each transformation (called a 'refactoring') does little, but a sequence of transformations can produce a significant restructuring. Since each refactoring is small, it's less likely to go wrong. The system is also kept fully working after each small refactoring, reducing the chances that a system can get seriously broken during the restructuring."

JavaScript Support

JavaScript is a scripting language designed to add interactivity to HTML pages.

Zend Studio provides support for JavaScript features in standalone JavaScript files as well as in PHP files and projects.

Some of the JavaScript features available for PHP projects and files are:

- JavaScript Element Outline - JavaScript elements will be displayed in the Outline view.
- JavaScript libraries will be added to the Include Paths of PHP projects for which JavaScript support has been enabled.
- JavaScript Build Paths - Setting the JavaScript Build Path for a project will allow you to determine which elements will be included in the project's build process. This allows access to content assist and refactoring options from resources in locations other than in the project itself.
- Advanced editing functionality in JavaScript editors. For example:
  - Drag and Drop behavior.
  - JavaScript Content Assist
  - JavaScript Syntax Coloring
  - JavaScript Code Commenting
  - JavaScript Hover Support
  - JavaScript Mark Occurrences
  - JavaScript Smart Goto Source
  - JavaScript Libraries
  - Ajax Tools

JavaScript support can be added to new or existing PHP projects so that all of Zend Studio's JavaScript features are available to the project.

For a full list of features enabled for JavaScript development, see Features - Javascript Developtment Toolkit (JSDT) in the JavaScript Development Toolkit User Guide.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help	Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

JavaScript Debugger

The Zend Studio JavaScript debugging function allows you to test your files and applications and detect errors in your code. With this function you can debug JavaScript code live from your workspace using an internal JavaScript Debugger.

To find out how to use the JavaScript Debugger see [Debugging JavaScript](#).

JavaScript Libraries

Enabling JavaScript Libraries in your project allows libraries to be referenced by the project and makes the elements within these resources available for operations such as Content Assist and Refactoring.

Zend Studio allows you to define a list of libraries which have to be loaded before analyzing JavaScript code in your project. This is the equivalent to HTML users writing directly in the script what JavaScript should be loaded by the browser.

Each library can be different; It can be a plain list of JavaScript files, a zip file with JavaScript files, a database of JavaScript function signatures, or a running JavaScript engine. Therefore, each library has an ID associated with it. For example: "com.zend.jsdt.support.jquery" or "org.eclipse.jsdt.system". The ID's uniquely identify the kind of JavaScript library being referred to so that Zend Studio can internally locate the right mechanism to load the library's contents.

JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects.

 JavaScript Libraries allows you to do the following:
- Quickly Add a Predetermined JavaScript Library
- Add a JavaScript Library
- Add a Library Folder to JavaScript Libraries
- Edit JavaScript Libraries
- Remove JavaScript Libraries

The supported JavaScript libraries are:

- [Dojo Library](#)
- ECMA 3 Browser Support Library - A standard JavaScript library.
- Internet Explorer Library - A JavaSript library specialized for Internet Explorer users.
- [jQuery Library](#)
- [ExtJS Library](#)
- Mozilla Firefox Library - A JavaSript library specialized for Mozilla Firefox users.
- [Prototype Library](#)
- User Library - Allows you to create or import a JavaScript library into your project.
- ECMAScript Built-In Library - A standard JavaScript library.
- Web Project support for JSDT - A standard JavaScript library.

PHPDocs

PHPDocs provides structured, easy-to-read documentation of all your php elements.

PhpDocumentor can automatically create PHPDocs from your scripts, using a templating system to change your source code comments into readable formats.

The PHPDoc Generator Wizard is Zend Studio's interface with phpDocumentor.

PHPDocs list all classes, functions, files and other elements in an easily-browsable format so that scripts can be easily navigated and understood.

PHPDocs also incorporate [PHPDoc Block comments](#) to provide descriptions and parameters for your code elements:
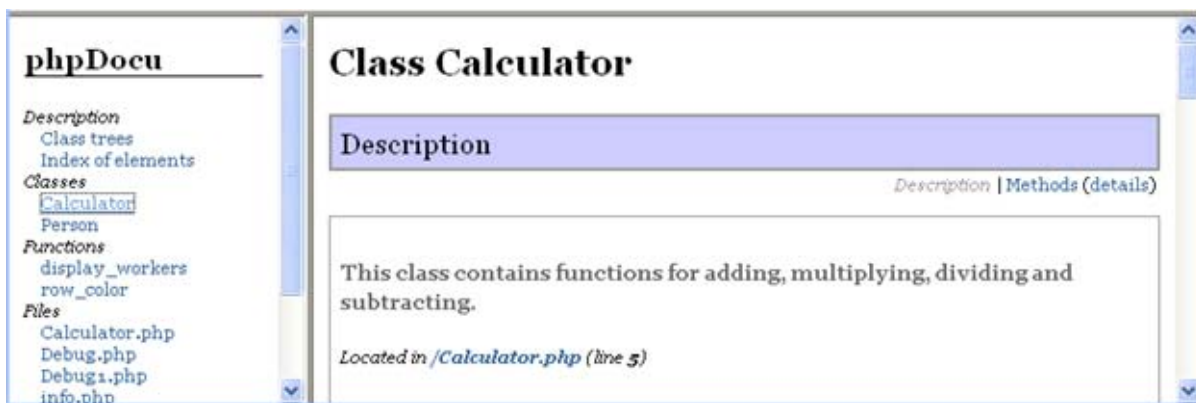
Code Galleries

Code snippets can be used to easily insert pre-defined sections of code into your script.
Code Galleries are pre-defined code snippets sites. Connecting to a Code Gallery will give you access to a selection of code snippets.

The Code Gallery view allows access to the Code Gallery sites and code snippets contained within them so that the code snippets can be easily inserted into your script.
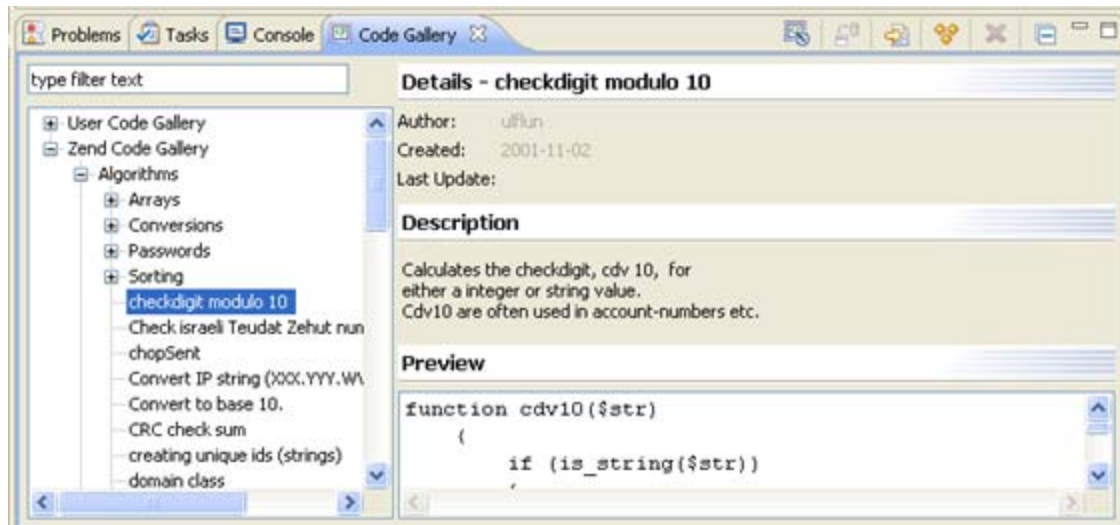
The Code Gallery view comes by default with a User Code Gallery, to which your own code snippets can be added, and the Zend Code Gallery. Access to the Zend Code Gallery will require registration to the Zend Network.

Selecting a code snippet will result in its details, description and preview being displayed in the left of the view.

Zend Guard Integration

Zend Guard is the first Electronic Licensing solution for the PHP marketplace. It includes the Encoding solution that pioneered PHP intellectual property protection. Unprotected intellectual property, in the form of plain text PHP scripts and software without license restrictions, can be copied, modified, and retained by someone else. It is available to your competitor, to hackers and even to developers at customer sites. Zend Guard provides tools that significantly lessen risk to your intellectual property. It is designed to prevent your property from being viewed or modified.

See the Zend Guard product site or the Zend Guard Online Documentation for more information on Zend Guard.

Zend Studio's integration with Zend Guard allows you to apply Zend Guard's encoding functionality to projects and applications created and stored in Zend Studio by allowing you to open them in Zend Guard. Conversely, files stored in Zend Guard can be opened and edited in Zend Studio.
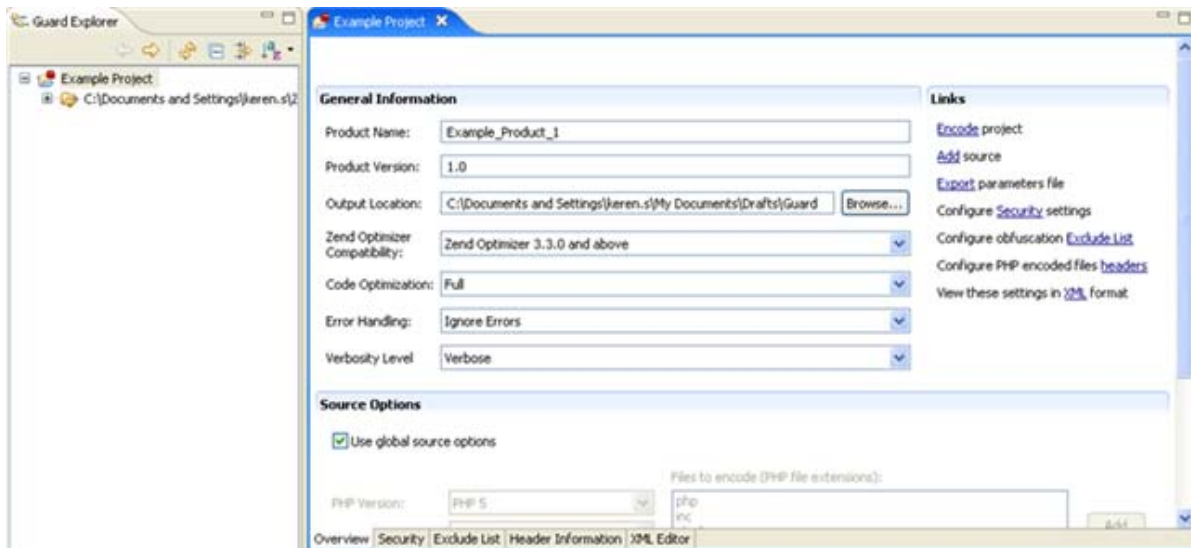
RSS Feeds

RSS is a format for syndicating news feeds from news sites and weblogs.

Zend Studio has a built-in RSS reader allowing you to view news and updates from the Zend Developer Zone, to view the latest mentions of Zend in the press or to view any other RSS feed channel.

RSS feeds are displayed as a list in the RSS view and can be displayed in Zend Studio's internal browser.

Each new item will be listed according to the RSS channel or time.
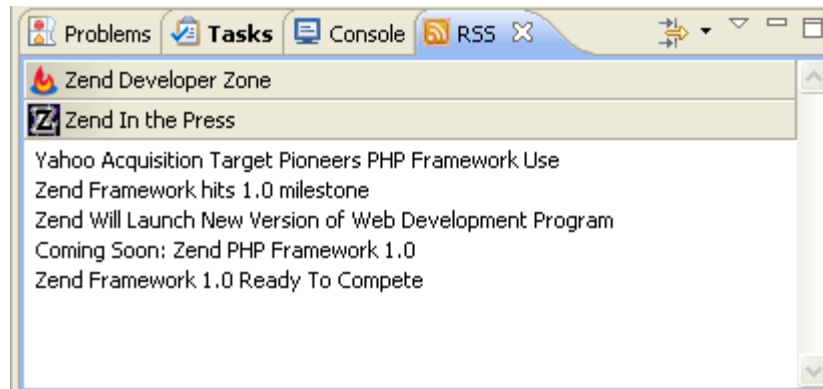
WSDL - Web Services Description Language

WSDL (Web Services Description Language) is an XML-formatted language used to describe Web service capabilities. Web services are a standardized way of allowing applications to interface and share data across the network. Web service messages are written in XML, thus allowing for different applications in different programming languages to interface with each other.

WSDL files define how the Web services work and the operations they perform. Zend Studio provides an integrated means for incorporating and inspecting WSDL files and a wizard for generating your own WSDL files.

Zend Studio for IBM i Extras

Zend Studio for IBM i contains additional extras to help you connect to and use IBM i functionality.

Free Registration

System i users who have downloaded Zend Studio are entitled to a free license.

To receive and activate your free license:

1. To go to the Zend registration website go to the Menu Bar | Help | Get a License.
2. In the site, enter your System i server serial number and optionally check the box to have the license emailed to you.
3. Click the Generate License button.
4. When prompted, enter you Zend User ID and Password.
   The license key will be generated and you will see your User Name and License Key displayed.
5. In Zend Studio , from the Menu bar go to Help | Register and enter your User Name and License Key.
6. Click OK.

Your Zend Studio for IBM i will be registered and all functionality will be available.

IBM i PHP API Toolkit functions Templates

For information on Templates and how to use them, see the 'Using Templates' topic.

Zend Studio contains templates for the following IBM i PHP API Toolkit functions:

IBM i Template	Explanation
i5ActiveJobs	Enables retrieving the system's active jobs: 1. Connects to i5 server 2. Opens active job list 3. Gets array for an active job entry 4. Closes handle received from i5_job_list function 5. Closes connection to i5 server
i5Connect	Enables connecting to the i5 server: 1. Connects to i5 server 2. Closes connection to i5 server

i5DataAreaCreate	Creates the data area:
	1. Connects to i5 server
	2. Creates data area of given size
	3. Closes connection to i5 server

i5DataAreaDelete   Enables deleting the data area:

1. Connects to i5 server
2. Deletes data area
3. Closes connection to i5 server

i5DataAreaRead   Enables reading from a data area:

1. Connects to i5 server
2. Reads from data area
3. Closes connection to i5 server

i5DataAreaWrite   Enables reading from a data area:

1. Connects to i5 server
2. Reads from the data area
3. Closes connection to i5 server

i5DtaqReceive   Enables reading data from the data queue without key:

1. Connects to i5 server
2. Reads data from the data queue without key
3. Closes connection to i5 server

i5DtaqReceiveKey   Enables reading data from the data queue with key:

1. Connects to i5 server
2. Reads data from the data queue with key
3. Closes connection to i5 server

i5DtaqSend   Enables putting data to the data queue without key:

1. Connects to i5 server
2. Puts data to the data queue without key
3. Closes connection to i5 server

i5DtaqSendKey   Enables putting data into the data queue without a key, it

1. Connects to i5 server
2. Puts data to the data queue without key
3. Closes connection to i5 server

i5JobLogs	Enables retrieving job log entries:

1. Connects to i5 server
2. Opens job log
3. Gets array for a job log entry
4. Closes handle received from i5_jobLog_list function
5. Closes connection to i5 server

i5ObjectListing	Enables getting an array with the message element for an object list entry:

1. Connects to i5 server
2. Opens object list
3. Gets for a object list entry
4. Closes handle received from i5_objects_list function
5. Closes connection to i5 server

i5Program	Enables calling a program and accept results from it:

1. Connects to i5 server
2. Opens a program or service procedure and prepares it to be run
3. Calls the program and optionally accepts results
4. Free program resource handle
5. Closes connection to i5 server

i5ProgramService	Creates Web Services class enabling invoking an RPG program:

1. Connects to i5 server
2. Opens a program or service procedure and prepares it to be run
3. Calls the program and optionally accepts results
4. Free program resource handle
5. Closes connection to i5 server

i5Spool	Enables getting spool file data from the queue and getting the data from the spool file:

1. Connects to i5 server
2. Creates an pool file lists, of certain output queue or for all queues
3. Gets spool file data from the queue
4. Get the data from the spool file
5. Free spool list resource
6. Closes connection to i5 server

i5UserSpaceCreate	Creates a new user space object:

1. Connects to i5 server
2. Creates new user space object
3. Closes connection to i5 server

i5UserSpaceDelete	Enables deleting a user space object:

1. Connects to i5 server
2. Deletes user space object
3. Closes connection to i5 server

i5UserSpaceGet	Retrieves user space data:

1. Connects to i5 server
2. Opens a user space and prepares it to be run
3. Retrieves user space data
4. Closes connection to i5 server

i5UserSpacePut	Enables to add user space data:

1. Connects to i5 server
2. Opens a user space and prepares it to be run
3. Adds user space data
4. Closes connection to i5 server

Content Assist

Zend Studio, contains Content Assist for commonly used IBM i functionality. Content Assist is available for the IBM i PHP Toolkit functions (listed above), as well as for connectivity to the Zend 5250 Bridge. For more information on Content Assist and how to use it, see the Content Assist topic.

Remote Server Support

Remote Server Support allows you to transparently access your remote server and remote resources. This provides an easy way to upload and download files from your remote server, as well as allowing you to develop your code in one environment, while in parallel executing it in a different environment. In addition, you can create and manage connections to your FTP and SSH remote systems through Remote Server Support. This will allow you to work on projects locally while keeping them updated on your remote server.
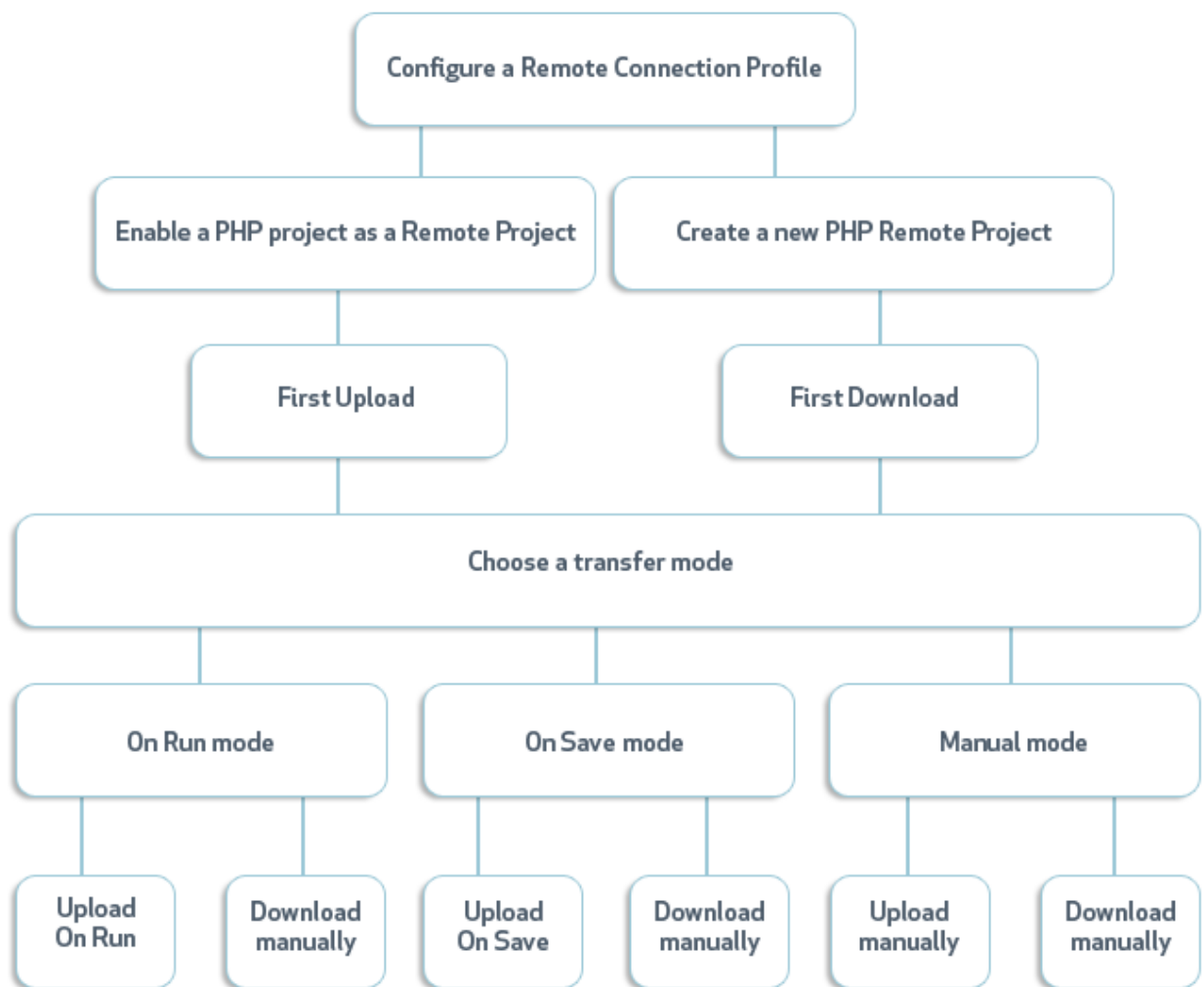
After setting the transfer mode, you can develop and work with your project as you normally do. Your project will be uploaded or downloaded to/from the remote server according to the properties you have set in the Remote Server Support Properties page. While developing, your changes can be uploaded to the remote server by the chosen method to facilitate running your code on a different environment for development and testing purposes.

To learn how to use Remote Server Support see Working with Remote Server Support.

Remote Server Support can be used with the existing Eclipse version control (CVS or SVN), or another version control you are using. For more information on built-in Eclipse version control options see the Subversive User Guide, or the Team CVS tutorial topic in the Workbench User Guide.

Mylyn Integration

Mylyn is a task-focused interface that reduces the overload of information and makes multi-tasking easy. It does this by making tasks a top priority, and integrating rich and offline editing for repositories such as Bugzilla, Trac, and JIRA. Once your tasks are integrated, Mylyn monitors your work activity to identify relevant information, and uses this task context to focus the user interface on the task-at-hand. This puts the information you need at your fingertips and improves productivity by reducing searching, scrolling, and navigation. By making task context explicit Mylyn also facilitates multitasking, planning, reusing past efforts, and sharing expertise.

The Task List allows you to view and manage your tasks. The Task List contains both "Local Tasks" and shared "Repository Tasks" that are stored in a task repository such as Bugzilla or Jira. Local tasks are typically contained in categories. Repository tasks are contained in special categories that represent queries.

Mylyn features the following:

- Task List Presentation
- Icon Legend and Color Coding
- Creating New Tasks
- Using Local Tasks
- Repository Tasks
- Task Focused Interface
- Task Focused Ordering
- Task Hyperlinking

For more information see the Mylyn User Guide.

Phar Integration

Phar Integration will enable you to integrate with phar archives. Phar is an archive system that enables you to group numerous files into a single file for easy distribution and installation. A phar archive provides a way to distribute a complete PHP application in a single file and run it from that file without needing to even use a disk. Using a Phar archive library is identical to using any other PHP library.

Phar Integration contains the following features:

- Export to phar.
- Import to phar.
- Add phar to your project.

Please click here for more information on Phar (external link).

Phar integration settings are defined per project and are part of the project library settings.

Ajax Tools

Ajax Tools is a set of features based on the Web browser incorporated into Zend Studio. This provides the advantage of having a fully functioning web browser in your environment as well as the ability to edit, debug, and monitor your projects live, thus improving and simplifying the process for you.
The additional functionality provided by Ajax Tools can be applied to HTML, CSS, JavaScript, and XML.
To use Ajax Tools functionalities go to the Web Browser Tools Perspective which can be manually accessed by going to **Window | Open Perspective | Other | Web Browser Tools Perspective**.

Ajax Tools provides the following Views:

- DOM Inspector View - The *DOM* Inspector view provides a pre-defined hierarchal tree of HEAD and BODY elements. The attributes and values of the selected node appear in the view as well.
- Browser Console View - The Browser Console view is an aggregative list of the execution errors, warnings, and information messages that occurred in the time the page open in the Internal Web Browser was running.
- Request Monitor View - The Request Monitor view allows you to analyze the requests that occur in the browser open in the Internal Web Browser. The request is separated into different components (request, waiting, and response), and allows you to see the exact time each component is active, in seconds. This view should be used when profiling your application.
- DOM Source View - The DOM Source view shows the content and structure, including the attributes and values, of the highlighted node (and its child nodes) in HTML format.
- CSS View - CSS style rules determine the formatting of an element. The CSS view provides four different tabs, each with a different approach to the CSS style rules in the browser, both active and inactive.
- JavaScript View - The JavaScript view allows you to evaluate JavaScript expressions. This is useful to test, check, and debug your JavaScript code.
- DOM Watcher View - The DOM Watcher view is a way to record events occurring in the node selected in the DOM Inspector view. This allows you to see what exact events are occurring live.
- DOM Compare View - The DOM Compare view compares DOM attributes, child nodes, and CSS properties of a node. This view will not appear automatically when selecting the Web Browser Tools Perspective. To open it go to **Window | View | DOM Compare**.

VMware Workstation Integration

Zend Studio allows you to integrate with VMware Workstation so that you can easily execute your project on a virtual machine. Working with a virtual machine allows you to develop your code on one operating system and execute it on a different one, all while working on one machine. This allows you to deploy and debug you PHP applications on a virtual machine as the server, all from your Zend Studio interface.

Prerequisites

The following components must be installed prior to the integration:

- Zend Studio 9.0 or above

- VMware Workstation 7.x or above, available for download at
  http://www.vmware.com/products/workstation/

- A configured virtual machine.
  The Zend Server image can be downloaded at
  http://www.zend.com/en/products/studio/downloads.
  See Importing the ZendServer.zip Image File into VMware Workstation to learn how to import the provided image, or see Creating a Custom Virtual Machine to learn how to create your own.

Once you have met all the required prerequisites you can begin to work with your VMware virtual machine.

Amazon AWS Toolkit for Zend Studio

The Amazon AWS Toolkit for Zend Studio simplifies the development cycle when using remote Amazon EC2 Instances by allowing PHP applications to be developed and debugged while working with Amazon and Zend Server DevPay Instances. The Zend Server Amazon Machine Image (AMI) is a complete virtual machine image that includes the latest version of Zend Server so that you can execute your PHP application on a pre-configured PHP runtime environment.

Zend DevPay allows you to purchase and manage your Zend Server AMI from your Amazon AWS Account. This makes working with a Zend Server AMI and the virtualization it provides in Zend Studio easy and efficient, letting you concentrate on developing your PHP application.

The Amazon AWS Toolkit for Zend Studio is made up of the following:

- Installing/Uninstalling the Amazon AWS Toolkit Plugin
- Prerequisites - All the prerequisites you must meet before setting up the Amazon AWS Toolkit for Zend Studio.
- Setting Up Your Zend Server Amazon EC2 Instance - Set up your Zend Studio and Zend Server Instance to be able to communicate with one another.
- Working with the Amazon AWS Toolkit for Zend Studio - Work with the Amazon AWS Toolkit for Zend Studioand its added functionalities in Zend Studio.
- AWS Toolkit Preferences
- Amazon AWS Toolkit Views - The views that are available with the Amazon AWS Toolkit for Zend Studioto help you manage your AMIs and Instances in your Zend Studio.

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

Application Deployment

Zend Studio offers users an integrated deployment platform allowing performance of all development processes within the same IDE. Using Zend Studio, you can create, launch, debug and deploy your application, all in the same workspace.

What is Deployment?

Deploying an application is the process of copying, configuring and enabling a specific application to a specific base URL on a target of your choice.

Once the deployment process has finished, the application becomes publicly accessible on the base URL.

The target carries out this two step process by first staging the application, and then activating it after successful staging.

The following chart illustrates the deployment process and the integration between Zend Studio and Zend Server:
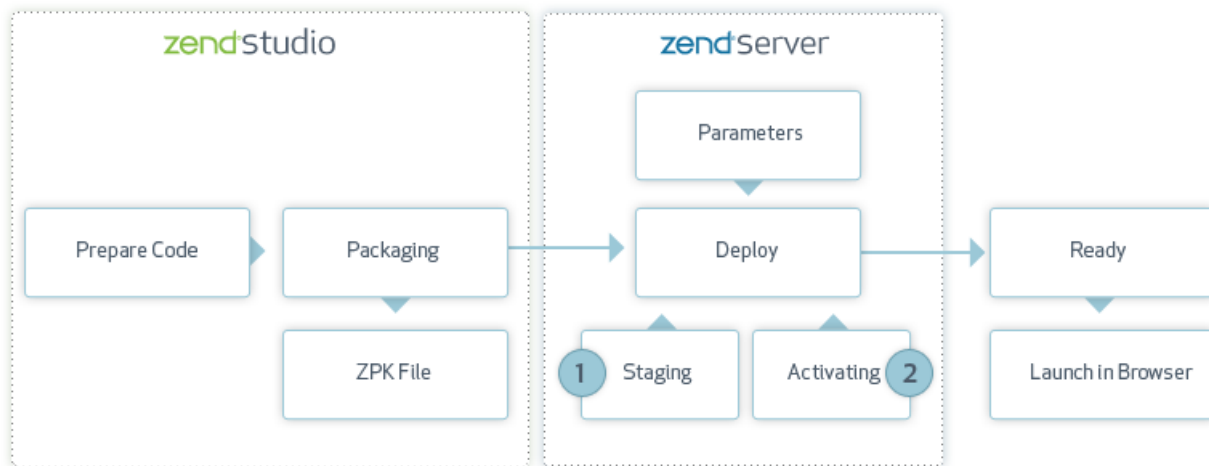
Why Use Deployment?

Deployment ensures the end-users have access to a properly functioning version of the Web application you have created.

Working with deployment in Zend Studio standardizes and automates the development process by establishing a consistent form for application production.

Deployment allows you to define the parameters necessary for your application in XML format as part of the application package. The application package includes the application data (source code), and any additional hook scripts you would like to use.

Once deployed, you can modify code and debug the application in update mode, enabling your application to run with the latest changes.

Deployment Options

Zend Studio supports several deployment options that can be applied at any stage of development- during the process of creating a new project or at any point thereafter:

- Deploying on a local Zend Server
- Deploying on a remote Zend Server
- Deploying on the Zend Developer Cloud

For more information on working with deployment in Zend Studio, see Working with Deployment.

For more information on working with deployment in Zend Server, see the Zend Server User Guide.

Tasks

Customizing Zend Studio Using the Welcome Page	Running Files and Applications
Creating New Projects	Debugging Files and Applications
Creating PHP Files	Profiling Files and Applications
Creating PHP Elements	Managing PHP Libraries
Zend Server Integration	Configuring a Project's PHP Include Path
Working with Code Tracing	Configuring a Project's PHP Build Path
Debugging and Profiling Zend Server Events	Managing Path Maps
Using Content Assist	Debugging and Profiling Zend Server Events
Using Templates	Using PHPUnit Testing
Using Drag and Drop	Using Refactoring
Formatting Code	Generating Getters and Setters
Using Code Folding	Overriding / Implementing Methods
Searching for PHP Elements	Creating a PHPDoc
Opening PHP Elements	Creating and Opening HTML Files
Opening Types/Methods	Using Code Galleries
Using Smart Goto Source	Integrating with Zend Guard
Viewing Type Hierarchies	Working with WSDL
Creating PHP Working Sets	Incorporating WSDL Files
Using Mark Occurrences	Viewing RSS Feeds and Adding RSS Channels
Finding and Replacing	Working with Remote Server Support
Applying Quick Fixes	Working with Mylyn Integration
Adding Comments	Developing with JavaScript
Adding PHP DocBlock Comments	Managing JavaScript Libraries
Using Local History	Working with Ajax Tools
Using CVS	Integrating with VMWare Workstation
Using SVN	Using the Amazon AWS Toolkit
Developing with Zend Framework	Working with Deployment
Connecting to Databases	

Customizing Zend Studio Using the Welcome Page

Zend Studio enables you to add or remove plugins from your product according to your preferences, allowing you to develop in a lighter and faster environment.

Installing Plugins

To add plugins in Zend Studio:

1. In the Menu bar, go to **Help | Welcome** to open the Zend Studio Welcome page.
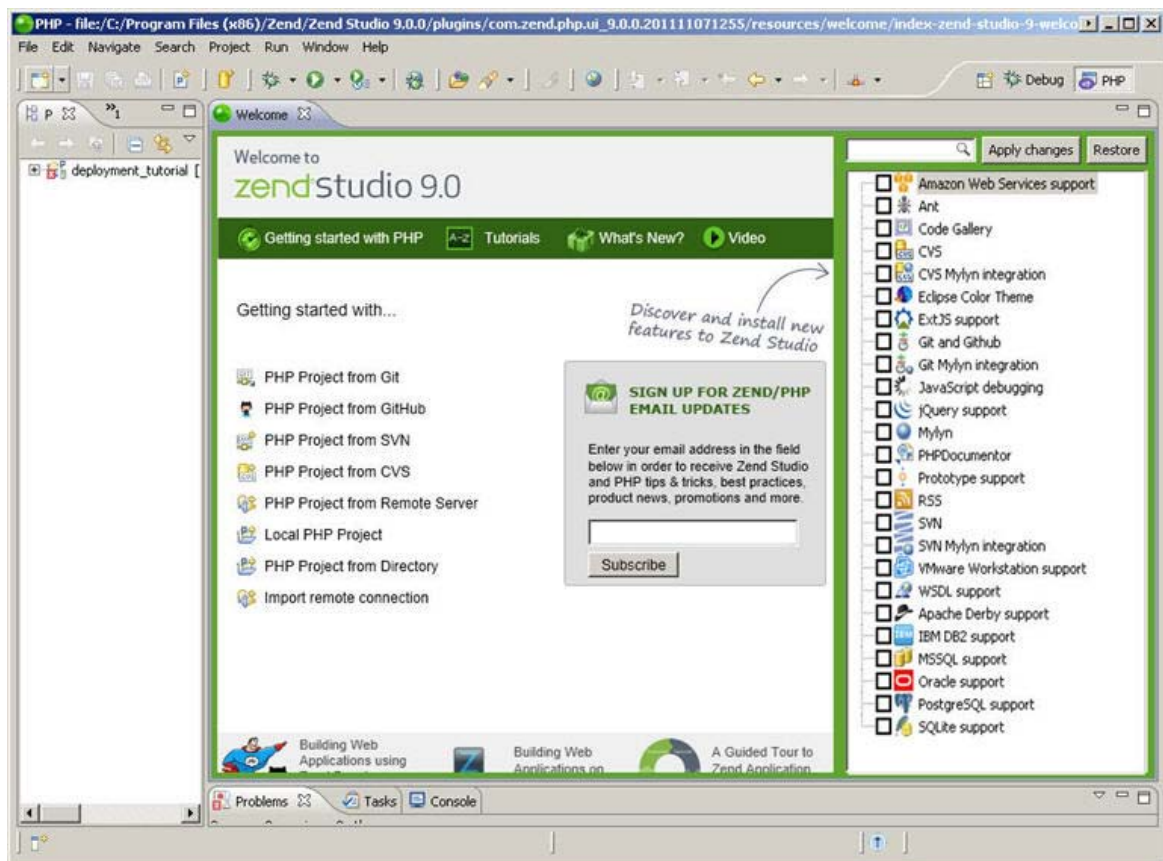
2. From the plugin list on the right, select the plugin you wish to add by marking the adjacent check-box, and click **Apply changes**.
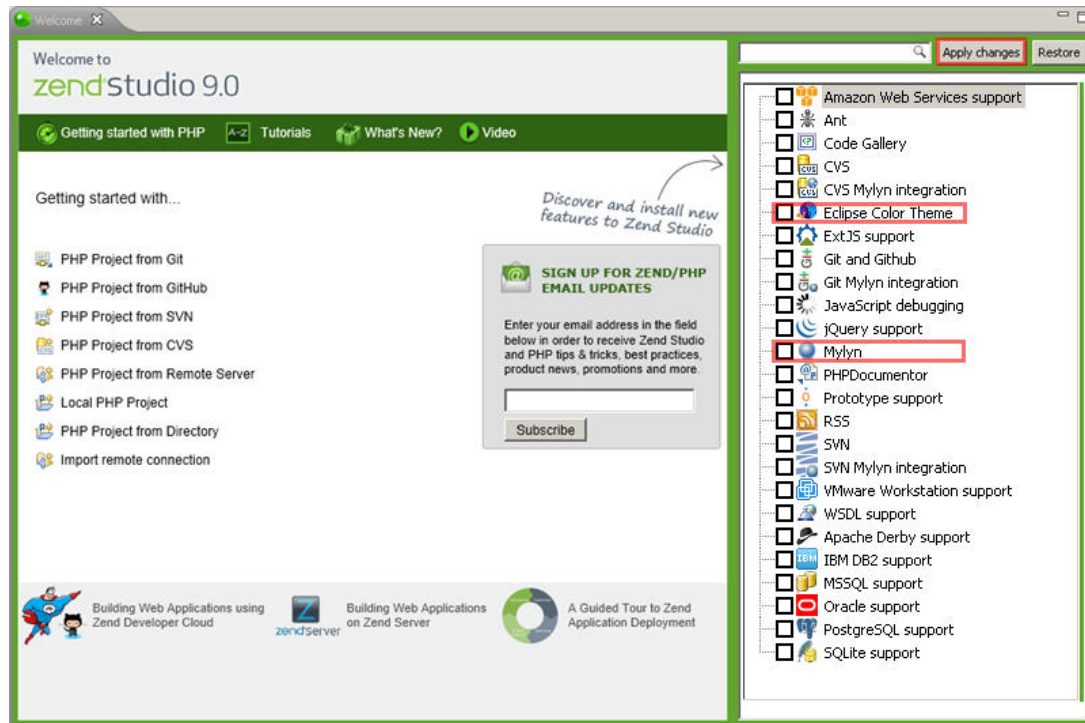
3. When prompted, click **Restart Now**.

    Zend Studio restarts with the plugins you selected installed.

Important!
If you do not restart Zend Studio, the changes will not be applied.

Creating a new project that necessitates the installation of a plugin, without this plugin having been installed beforehand, will automatically prompt you to begin the installation process of that specific plugin.

Uninstalling Plugins

To remove plugins from Zend Studio:

1. In the Menu bar, go to **Help | Welcome** to open the Zend Studio Welcome page.
2. Unmark the plugin you wish to remove, and click **Apply changes**.
3. When prompted, click **Restart Now**.

    Zend Studio restarts with the plugin uninstalled.

At any stage prior to applying changes and restarting Zend Studio, clicking on **Restore** will revert to current settings.

Creating New Projects

In product-name, the New Project Wizards make creating new projects a simple task.

There are several options to create new projects in Zend Studio:

- In the Welcome Page, click one of the links to the New Project Wizards.
- From the Menu-bar, select **File | New.**

  -OR-

  Click on the **New** icon

- In the PHP Explorer, right-click and select **New**.

For more information on the new project workflow and available New Project Wizards, see New Project Wizards.

Creating a New Project from CVS

A Concurrent Versions System (CVS) repository is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to revert file and project states back to previous versions.

Once projects are placed on the CVS repository, they can be checked out (imported) by anyone with access to that repository. CVS repository connections allow you to import projects from your repository to your workspace, which you can make and upload changes to.

This procedure describes how to import projects from a CVS repository into Zend Studio.

To create a new project from CVS:

1. Install the CVS plugin.

   For further information on installing plugins in Zend Studio, see <u>Customizing Zend Studio Using the Welcome Page</u>.

2. Go to **File | New | PHP Project from CVS**.

   -OR-

   In the PHP Explorer View, right-click and select **New | PHP Project from CVS**.

   The New PHP Project Wizard CVS Page dialog is displayed.

3. Enter the following information:

- Project Name: Enter the name of your project.
- Location: Enter the location folder for your project.
- Repository: Select an existing CVS repository from the dropdown menu.

  To define a new one, click **New Repository**.

  The Add a new CVS Repository dialog is displayed.

If you have not created a repository yet, enter the information required to identify and connect to the repository location:

- Host - The host address (e.g. mycomputer.com).
- Repository path - The path to the repository on the host (e.g /usr/local/cvsroot)
- User - The user name with which you connect to the repository.
- Password - The password for the user name.
- Connection Type - The authentication protocol for the CVS server.

- If the host uses a custom port, enable Use Port and enter the port number.
  - Module Name: Enter name of module from the defined repository select by clicking **Browse**.

    The Select Module dialog is displayed.
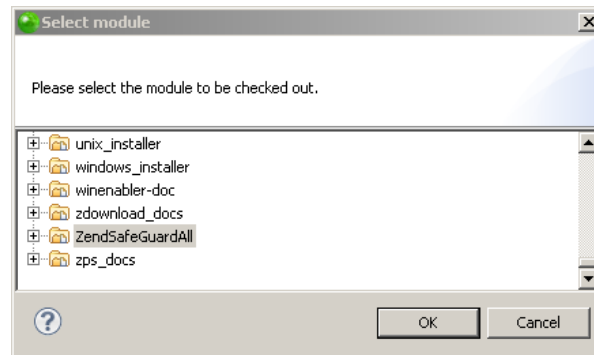
4. Select the requested module and click **OK**.

   The module is selected and appears in the Module Name path.

5. Click **Next**.

   The Launch Settings dialog is displayed.

   By default, the dialog displays with the Launch URL option selected.

   This option connects your application to a Local Web server with no Zend Deployment Support options. The name of your Host server and the application Base Path will automatically be displayed in the relevant fields. You may select a different server by clicking on the Host dropdown menu and clicking on **New Server**.

6. Two additional options are available for launching:

▪ Launch CLI Application: Select this option if you do not wish to enable deployment, and to launch your application as a Local CLI.

- Launch with Zend Application Deployment : Select this option to deploy your project on Zend Server, the Zend Developer Cloud or a Local server with deployment options.

  For more information on Zend's Application Deployment Support, see [Working with Deployment](#).

7. Click **Next**.

    The Libraries dialog is displayed.

8. Select the libraries you wish to add to the project, and click **Finish**.

Note:
If not already installed, selecting to install jQuery Library Support, ExtJS and Prototype will prompt a restart of Zend Studio.

The new project will be created in your workspace and displayed in the PHP Explorer View.
Once you have imported a project from CVS into your workspace, you can add files, edit existing files and commit your changes to the CVS repository.

Creating a New Project from Git

Git is a distributed revision control system with an emphasis on speed. Every Git working directory is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server.

Zend Studio allows you to create a new PHP project by importing an existing GitHub project.

To create a new Git project in Zend Studio:

1. Install the Git plugin.

   For further information on installing plugins in Zend Studio, see Customizing Zend Studio Using the Welcome Page.

2. Go to **File | New | PHP Project from Git.**.

   -OR-

   In the PHP Explorer View, right-click and select **New | PHP Project from Git.**.

   The New PHP Project Wizard GitHub Repository dialog is displayed.

3. Enter the following information:

▪ Project Name: Enter the name of your project.

▪ Location: Enter the location folder for your project.

▪ URL: Enter your Git repository URL.

▪ Authentication-

• User: Enter your Git username.

• Password: Enter your Git password.

4. Click **Next**.

The Launch Settings dialog is displayed.

By default, the dialog displays with the Launch URL option selected.

This option connects your application to a Local Web server with no Zend Deployment Support options. The name of your Host server and the application Base Path will automatically be displayed in the relevant fields. You may select a different server by clicking on the Host dropdown menu and clicking on **New Server**.

5. Two additional options are available for launching:

▪ Launch CLI Application: Select this option if you do not wish to enable deployment, and to launch your application as a Local CLI.

▪ Launch with Zend Application Deployment: Select this option to deploy your project on Zend Server, the Zend Developer Cloud or a Local server with deployment options. For more information on Zend's Application Deployment Support, see [Working with Deployment](#).

6. Click **Next**.

   The Libraries dialog is displayed.

7. Select the libraries you wish to add to the project, and click **Finish**.

Note:
If not already installed, selecting to install jQuery Library Support, ExtJS and Prototype will prompt a restart of Zend Studio.

The new project will be created in your workspace and displayed in the PHP Explorer View.

Creating a New Project From GitHub

GitHub is a web-based hosting service for software development projects that use the Git revision control system.

Zend Studio allows you to create a new PHP project by importing an existing GitHub project.

To create a new GitHub project in Zend Studio:

1. Install the GitHub plugin.

   For further information on installing plugins in Zend Studio, see [Customizing Zend Studio Using the Welcome Page](#).

2. Go to **File | New | PHP Project from GitHub**.

   -OR-

   In the PHP Explorer View, right-click and select **New | PHP Project from GitHub**.

   The New PHP Project Wizard GitHub Repository dialog is displayed.

3. Enter the following information:

- Project Name: Enter the name of your project.

- Location: Enter the location folder for your project.

- Authentication-

  - User: Enter your GitHub username.

  - Password: Enter your GitHub password.

4. Click **Refresh**.

Your application location will appear in the URL box or in the adjacent dropdown menu.
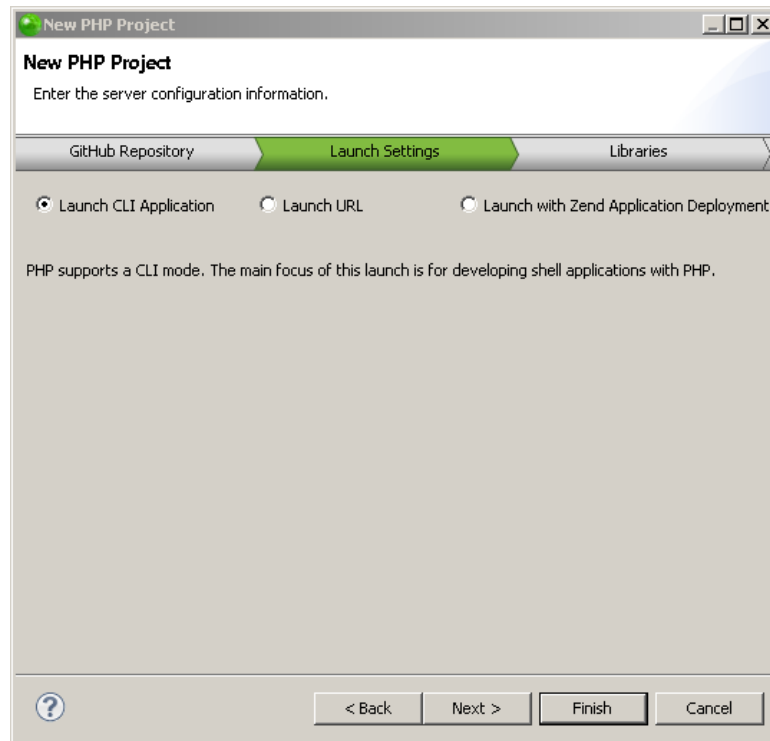
5. Click **Next**.

The Launch Settings dialog is displayed.

By default, the dialog displays with the Launch URL option selected.

This option connects your application to a Local Web server with no Zend Deployment Support options. The name of your Host server and the application Base Path will automatically be displayed in the relevant fields. You may select a different server by clicking on the Host dropdown menu and clicking on **New Server**.
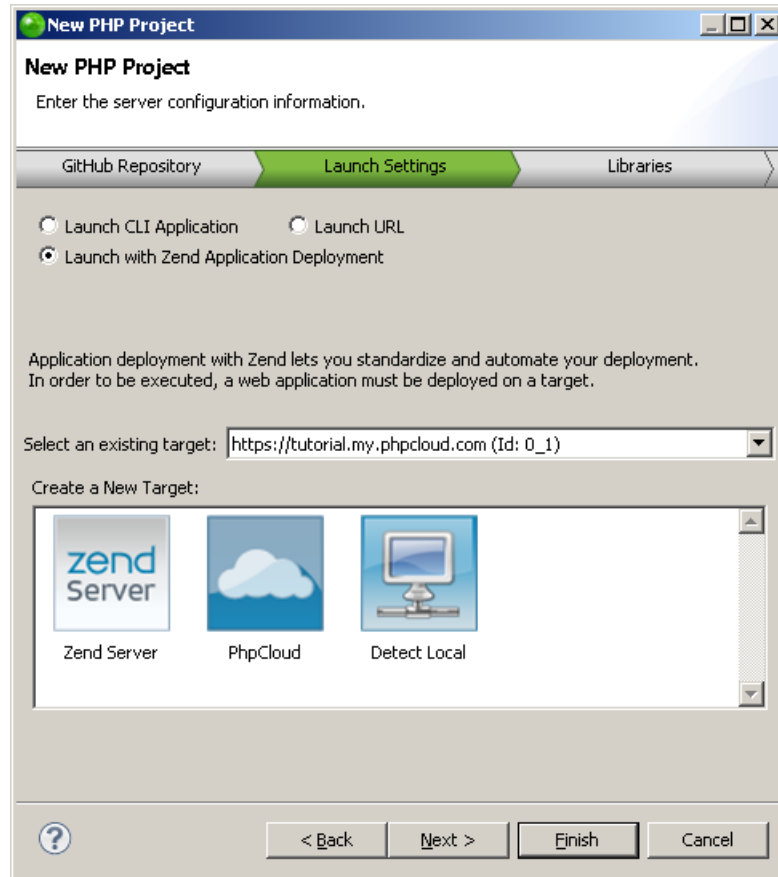
6. Two additional options are available for launching:

- Launch CLI Application: Select this option if you do not wish to enable deployment, and to launch your application as a Local CLI.
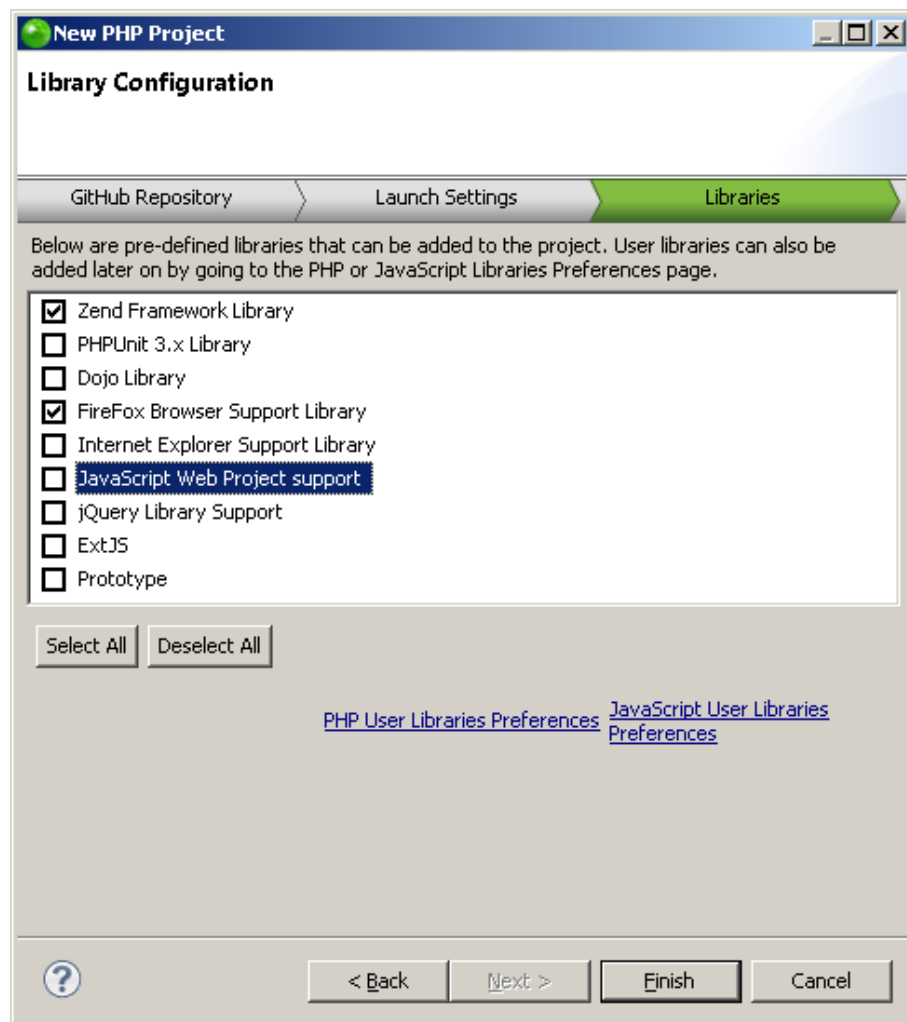
- Launch with Zend Application Deployment : Select this option to deploy your project on Zend Server, the Zend Developer Cloud or a Local server with deployment options. For more information on Zend's Application Deployment Support, see Working with Deployment.

7. Click **Next**.

The Libraries dialog is displayed.

8. Select the libraries you wish to add to the project, and click **Finish**.

Note:
If not already installed, selecting to install jQuery Library Support, ExtJS and Prototype will prompt a restart of Zend Studio.

The new project will be created in your workspace and displayed in the PHP Explorer View.

Creating a New Project from SVN

SVN, or Subversion, is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions.

Once projects are placed on the SVN repository, they can be checked out (imported) by anyone with access to that repository. SVN repository connections allow you to import projects from your repository to your workspace, which you can make and upload changes to.

This procedure describes how to import projects from a SVN repository into Zend Studio.

To create a new project from SVN in Zend Studio:

1. Install the SVN plugin. For further information on installing plugins in Zend Studio, see Customizing Zend Studio Using the Welcome page.

2. Go to **File | New | PHP Project from SVN**.

   OR-

   In the PHP Explorer View, right-click and select **New | PHP Project from SVN.**

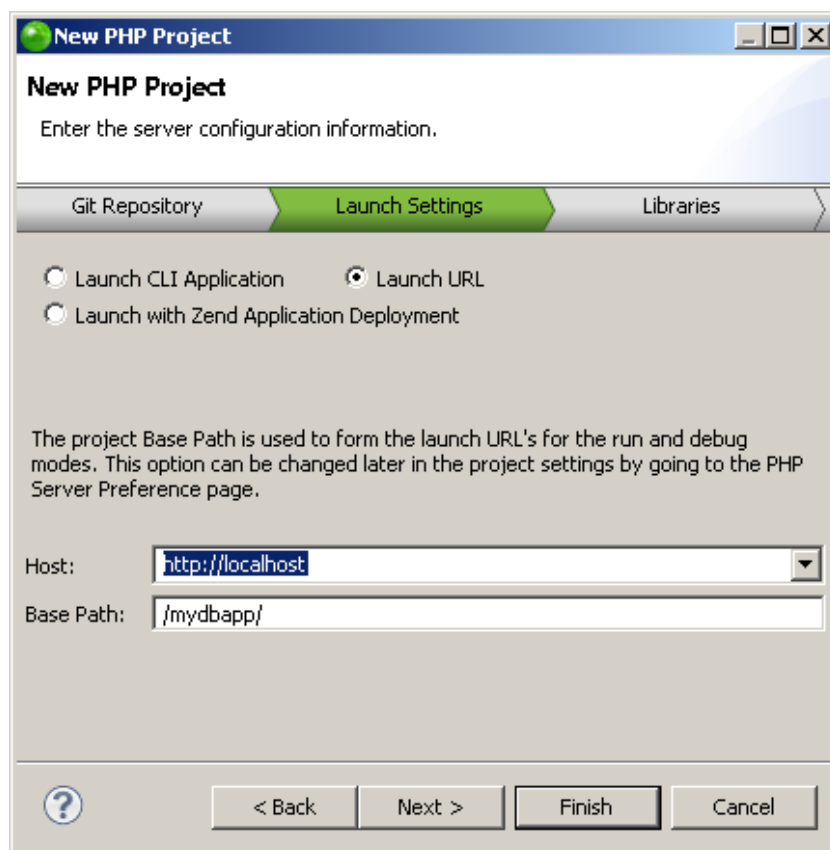   The New PHP Project SVN Details dialog is displayed.
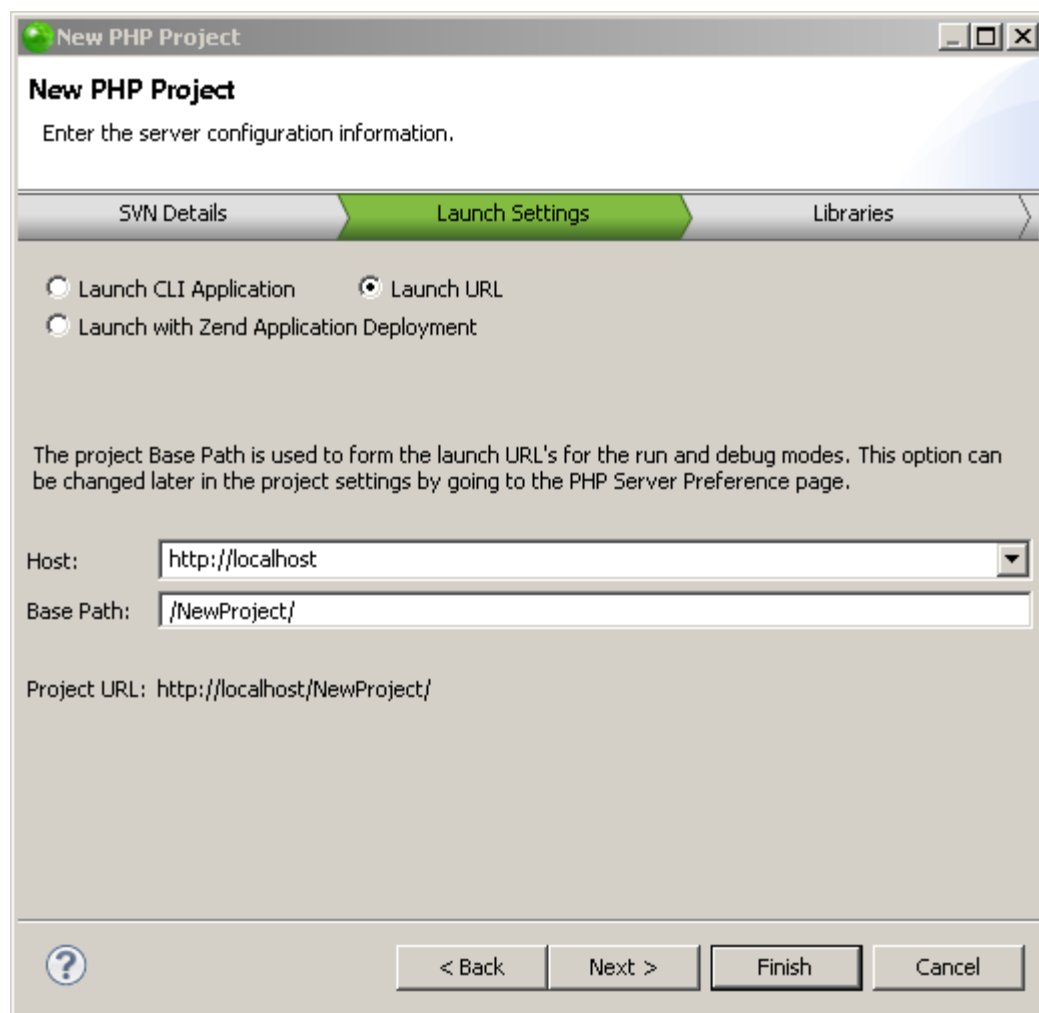
3. Enter the following information:
  - Project Name: Enter the name of your project.
  - Location: Enter the location folder for your project.
  - URL: Enter the URL address of your project.
    You can click on **Browse Repositories** to search for existing repository locations or create new repositories.
  - Username:Enter your SVN username.
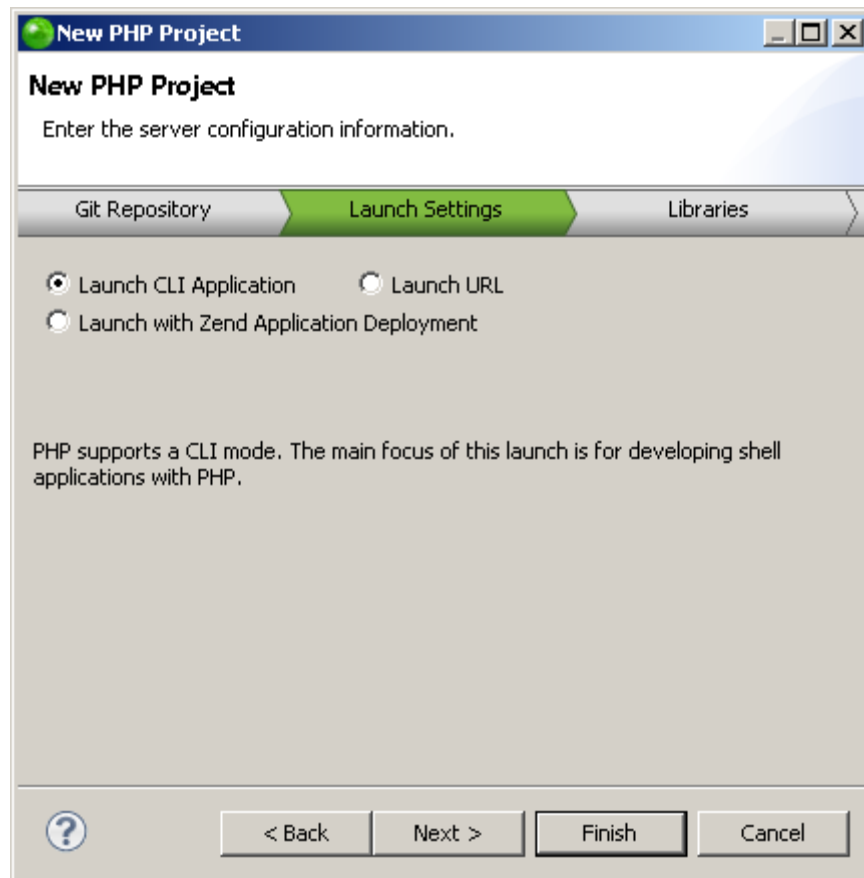 - Password: Enter your SVN password.
4. Click **Next**.
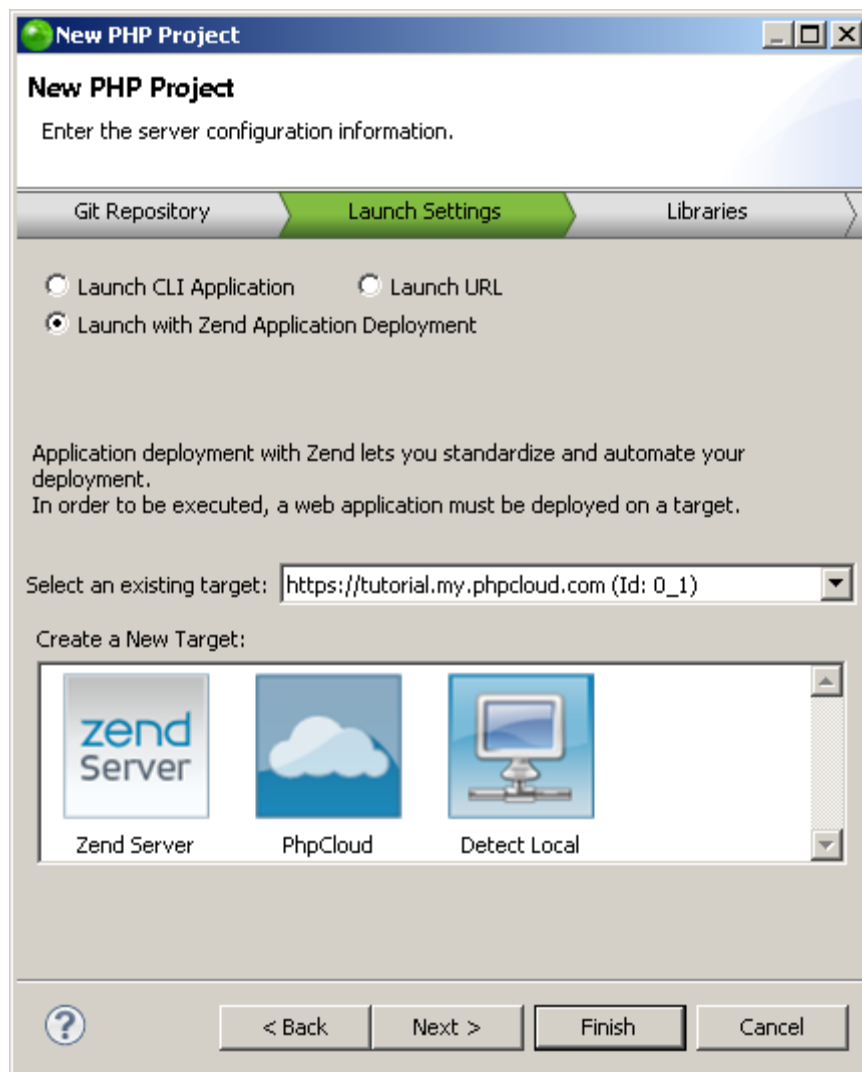 The Launch Settings dialog is displayed. By default, the dialog displays with the Launch URL option selected. This option connects your application to a Local Web server with no Zend Deployment Support options. The name of your Host server and the application Base Path will automatically be displayed in the relevant fields. You may select a different server by clicking on the Host dropdown menu and clicking on **New Server**.

5. Two additional options are available for deployment:

▪ Launch CLI Application: Select this option if you do not wish to enable deployment, and to launch your application as a Local CLI.
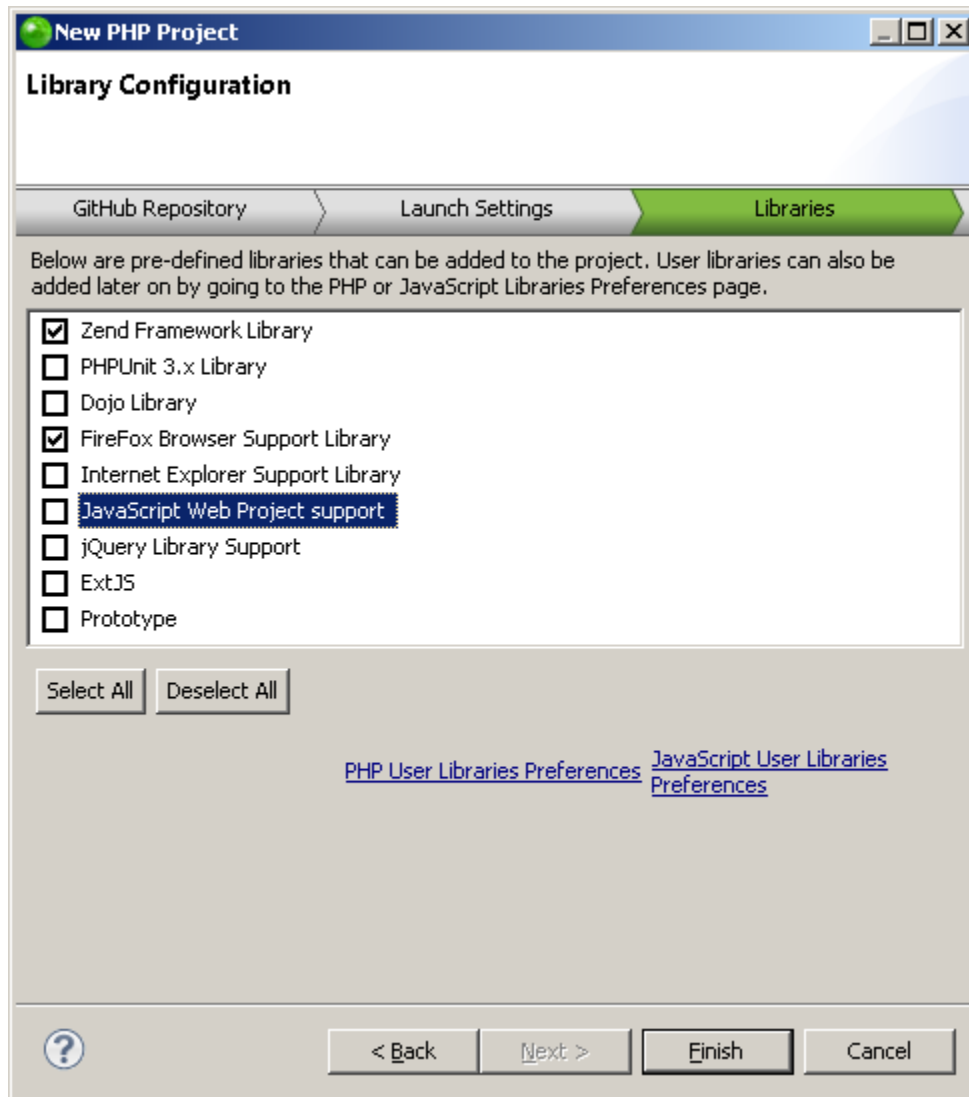
- Launch with Zend Application Deployment : Select this option to deploy your project on Zend Server, the Zend Developer Cloud or a Local server with deployment options. For more information on Zend's Application Deployment Support, see Working with Deployment.

6. Click **Next**.

   The Libraries dialog is displayed.

7. Select the libraries you wish to add to the project, and click **Finish**.

Note:

If not already installed, selecting to install jQuery Library Support, ExtJS and Prototype will prompt a restart of Zend Studio..

The new project will be created in your workspace and displayed in the PHP Explorer View.

Creating a New Local Project

This procedure describes how to create a new local PHP project.

To create a new local PHP project:

1. Go to **File | New | Local PHP Project**.

   -OR-

   In the PHP Explorer View, right-click and select **New | Local PHP Project**.

   The New PHP Project Settings dialog is displayed.

2. Enter the following information:

- Project Name: Enter the name of your project.

- Location: Enter the location folder for your project.

- Content: By default, **Zend Framework** will be selected as your content option. You can select **Basic**

  for using the basic content package.

- Version: Zend Studio has Zend Framework 1.11.11as a built-in content feature. If you wish to use the **Basic** content, you can select between php4, php5 or php5.3.

3. Click **Next**.

The Launch Settings dialog is displayed.

By default, the dialog displays with the Launch URL option selected.

This option connects your application to a Local Web server with no Zend Deployment Support options. The name of your Host server and the application Base Path will automatically be displayed in the relevant fields. You may select a different server by clicking on the Host dropdown menu and clicking on New Server.

4. Two additional options are available for launching:
- Launch CLI Application: Select this option if you do not wish to enable deployment, and to launch your application as a Local CLI.

- Launch with Zend Application Deployment : Select this option to deploy your project on Zend Server, the Zend Developer Cloud or a Local server with deployment options. For more information on Zend's Application Deployment Support, see [Working with Deployment](#).

5. Click **Next**.

    The Libraries dialog is displayed.

6. Select the libraries you wish to add to the project, and click **Finish**.

Note:
If not already installed, selecting to install jQuery Library Support, ExtJS and Prototype will prompt a restart of Zend Studio.

Creating a New Project from an Existing Directory

This procedure describes how to create a new PHP project from an existing directory on your machine.

You can import any existing files you want and develop them in Zend Studio.

To create a new PHP project from an existing directory:

1. Go to **File | New | PHP Project from Existing Directory**.

   -OR-

   In the PHP Explorer View, right-click and select **New | PHP Project from Existing Directory**.

   The New PHP Project Settings dialog is displayed.
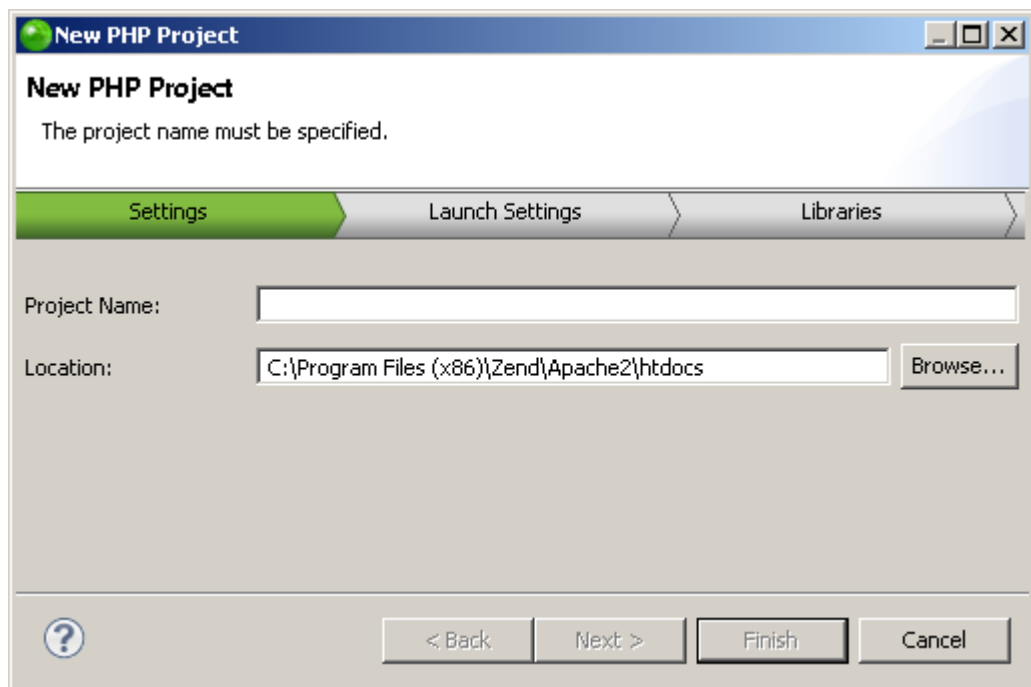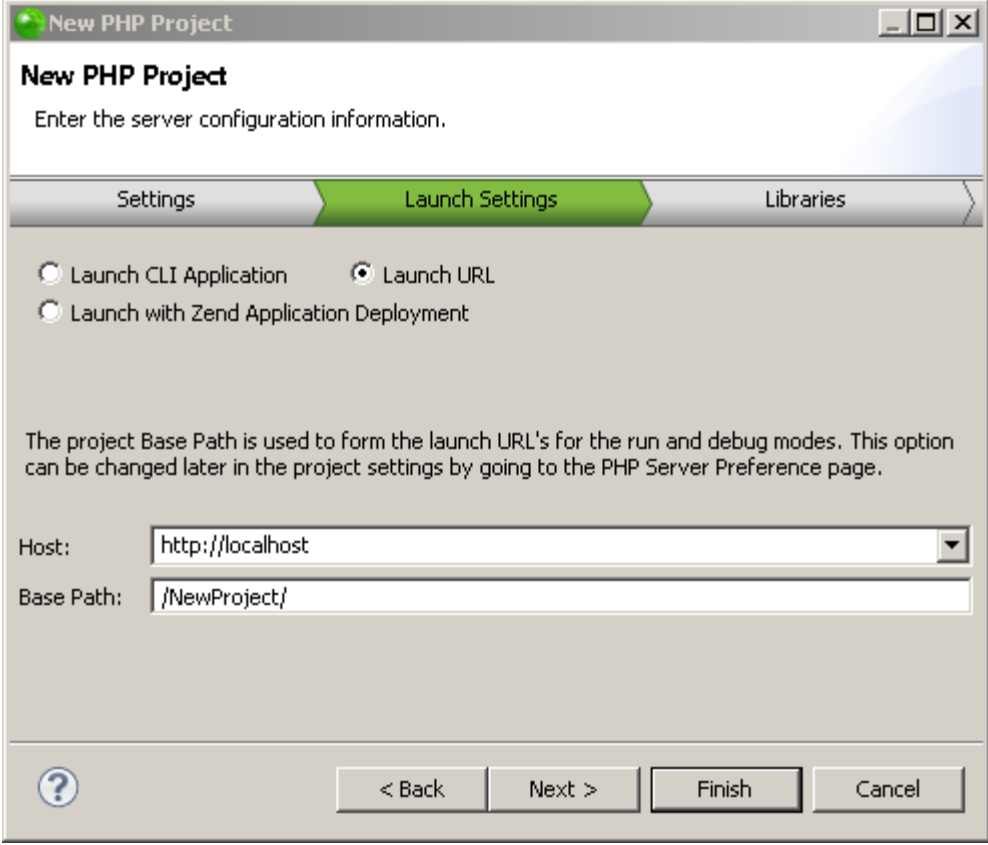
2. Enter the following information:

- Project Name: Enter the name of your project.

- Location: Enter the location folder for your project.

3. Click **Next**.
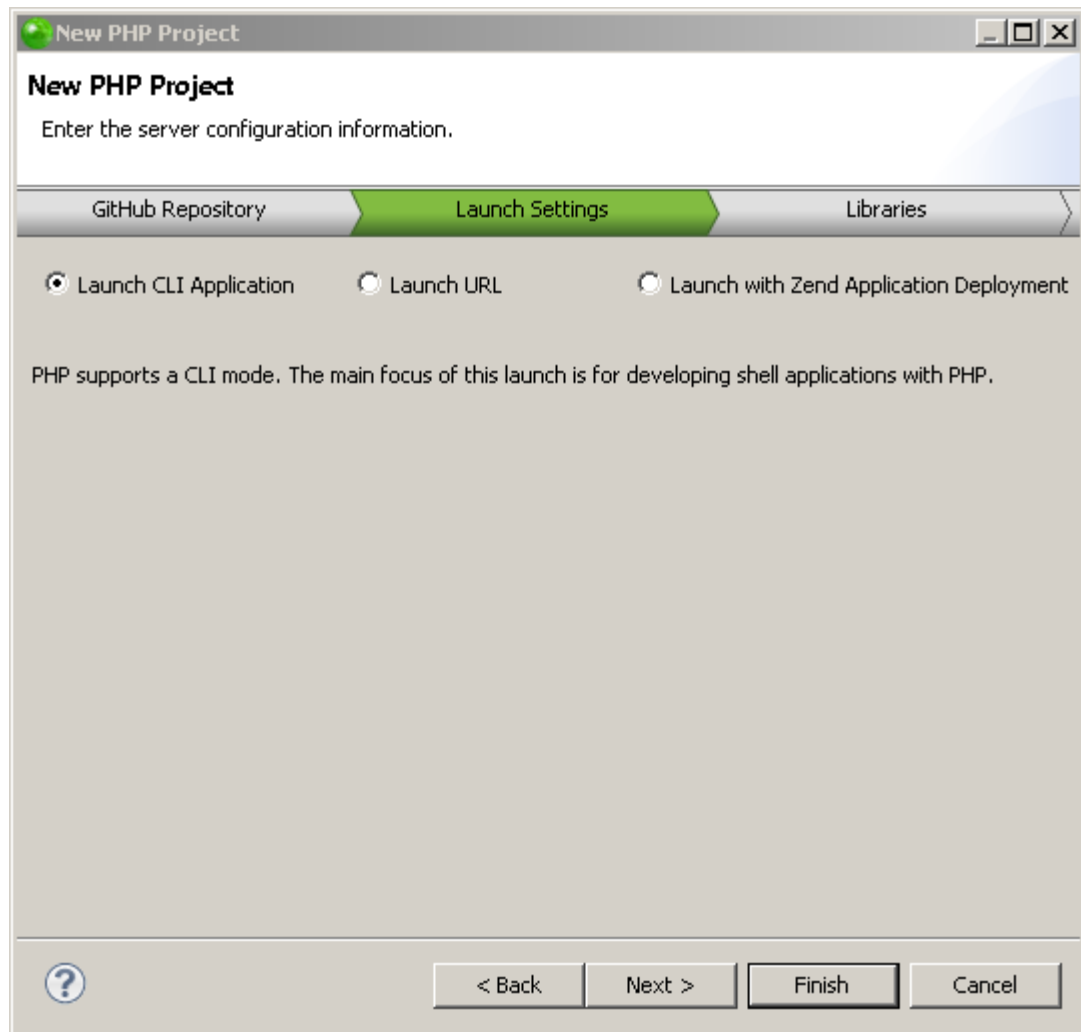
The Launch Settings dialog is displayed.

By default, the dialog displays with the Launch URL option selected.

This option connects your application to a Local Web server with no Zend Deployment Support options. The name of your Host server and the application Base Path will automatically be displayed in the relevant fields. You may select a different server by clicking on the Host dropdown menu and clicking on New Server.
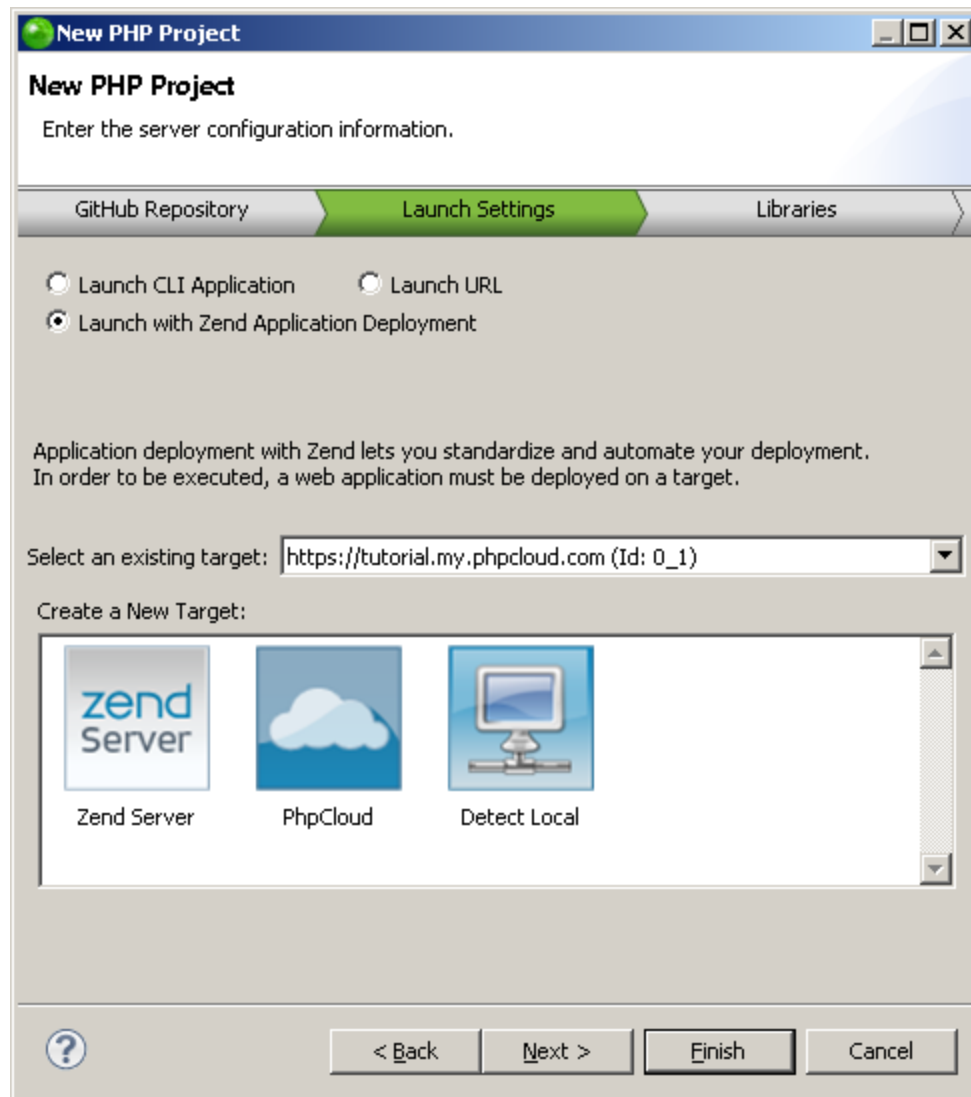
4. Two additional options are available for launching:

▪   Launch CLI Application: Select this option if you do not wish to enable deployment, and to launch your application as a Local CLI.

▪ Launch with Zend Application Deployment : Select this option to deploy your project on Zend Server, the Zend Developer Cloud or a Local server with deployment options.

For more information on Zend's Application Deployment Support, see [Working with Deployment](#).

5. Click **Next**.

   The Libraries dialog is displayed.

6. Select the libraries you wish to add to the project, and click **Finish**.

The new project will be created in your workspace and displayed in the PHP Explorer View.

Creating a New Project from a Remote Server

This procedure describes how to create a new project with remote server support.

Through Zend Studio, you can connect to a remote server and import and develop any files you wish. Any changes saved in Zend Studio, will be updated through the server and displayed after a refresh command on your browser.

To create a new PHP project from a remote server:

1. Go to **File | New | PHP Project from Remote Server**.

   -OR-

   In the PHP Explorer View, right-click and select **New | PHP Project from Remote Server.**

   The Remote Configuration dialog is displayed.

2. Enter the following information:

- Project Name: Enter the name of your project.

- Location: Enter the location folder for your project.

3. In the Remote Connection field click **Manage** to create a new remote system connection.

   The New Remote System Connection is displayed.
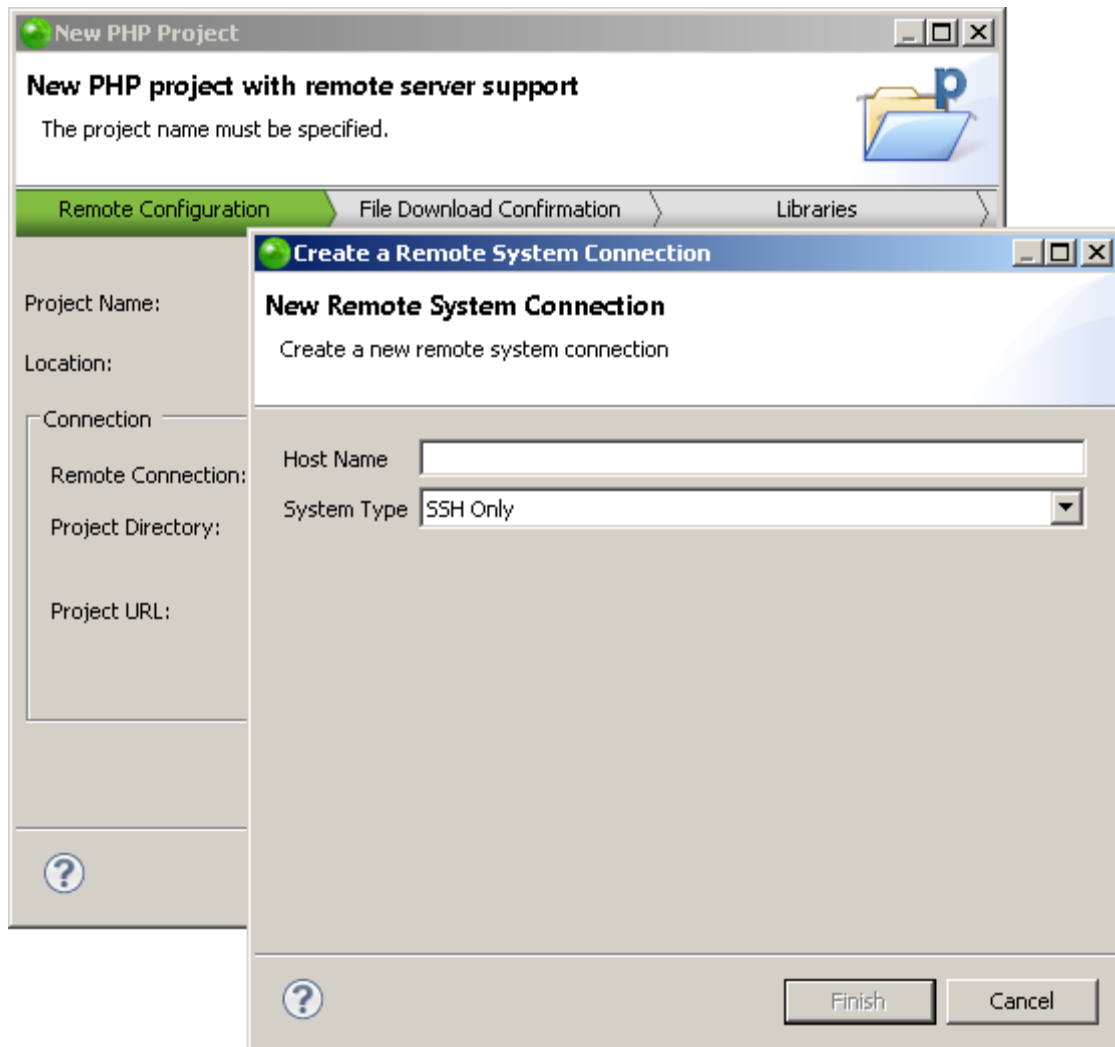
4. Enter the host server name and select system type (SSH Only, FTP Only) and click **Finish**.

The Remote Connection Profile dialog is displayed.

5. Enter the following information:

- Alias Name: Enter an alias name for the remote host server.
- Host Name: Enter the name of the remote host server.
- Username: Enter the username of the remote host server.
- Password: Enter the password for the remote host server.
- Initial Directory: Select the initial directory on the host server from which to import files.
- Timeout (sec): Select the length of time in seconds that Zend Studio waits for a reply form the host server before terminating the connection process.

6. Click **Test Connection**.

    A popup dialog is displayed announcing connection success.
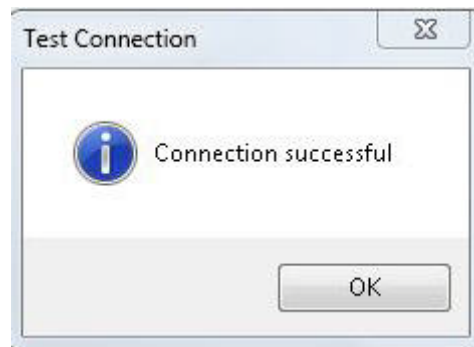
7. Click **Finish**.

    The Remote Configuration dialog is displayed again with the new connection settings selected.
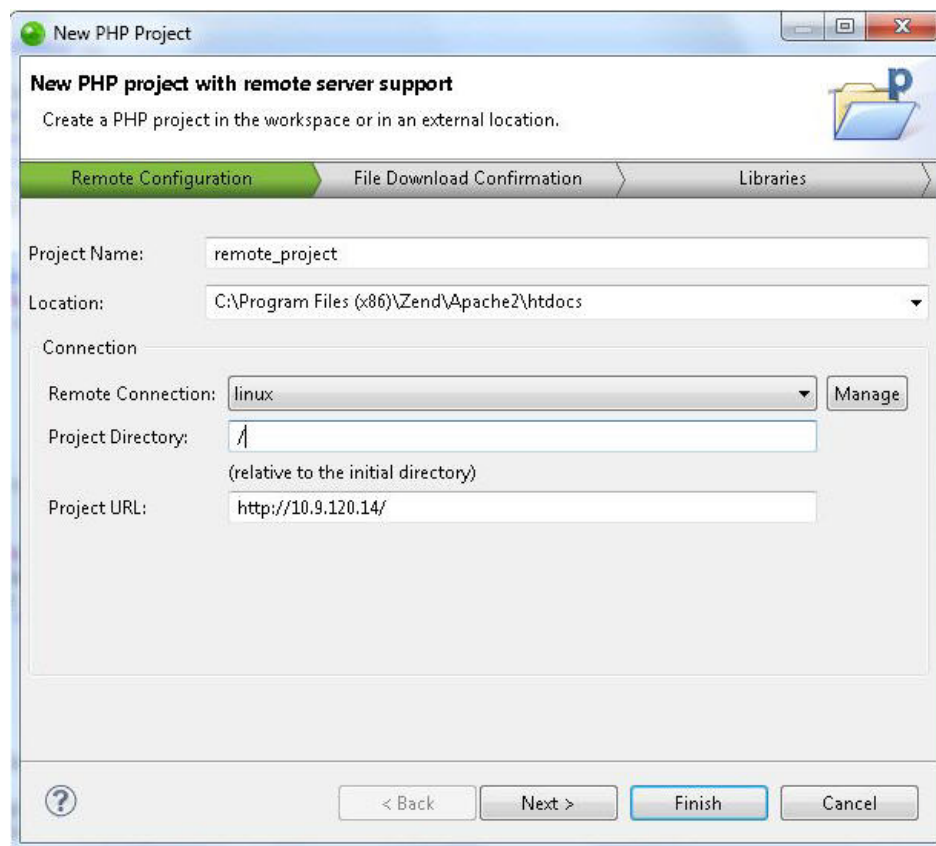
8. In the **Project Directory** field enter the project directory root on the remote host server.

    Entering **/** will instruct Zend Studio to display all folders relative to the initial directory root.

9. Click **Next**.

The Data Download Selection dialog is displayed.
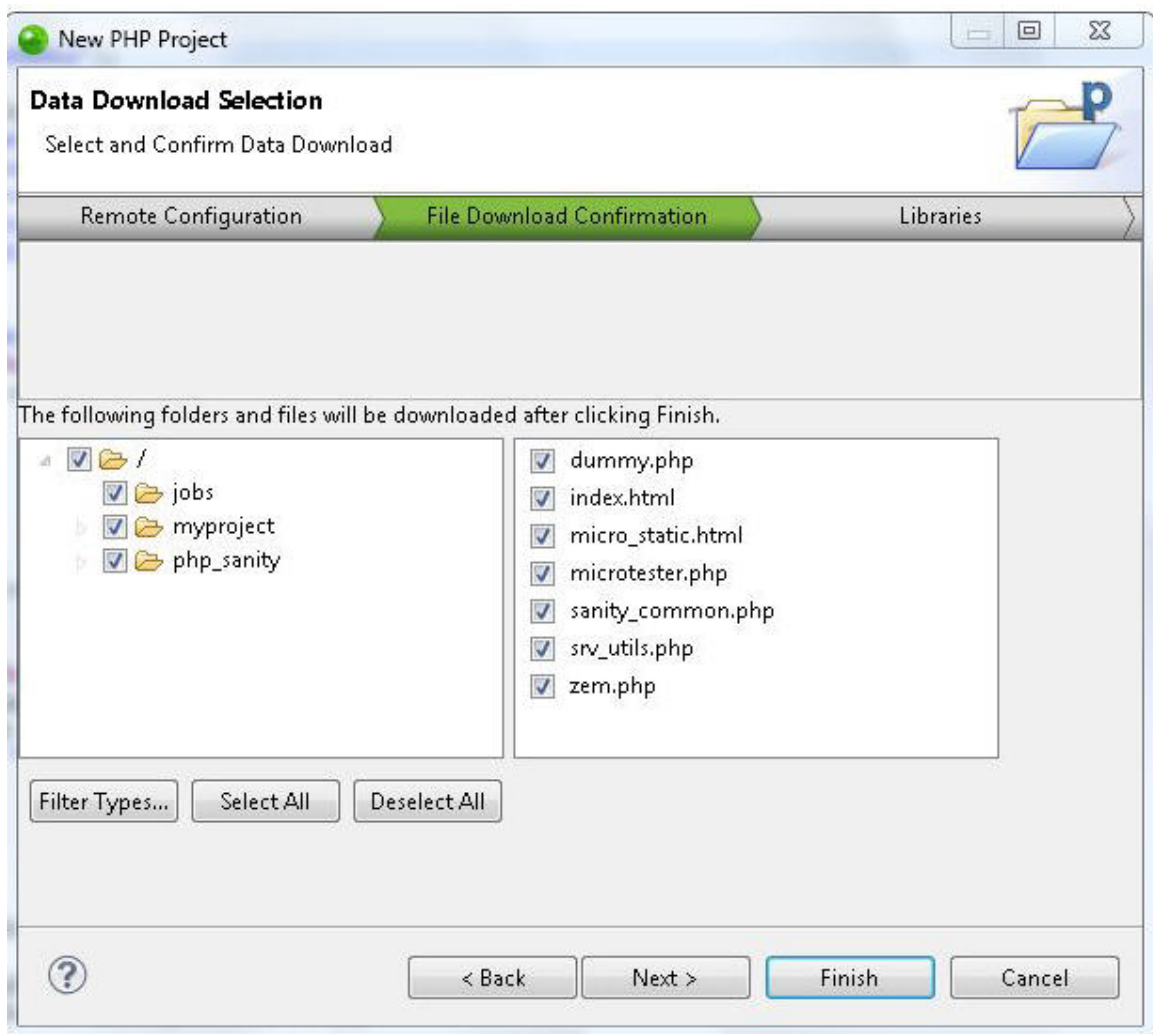
10. Mark the files you wish to import into Zend Studio, and click **Next**.

The Libraries dialog is displayed.
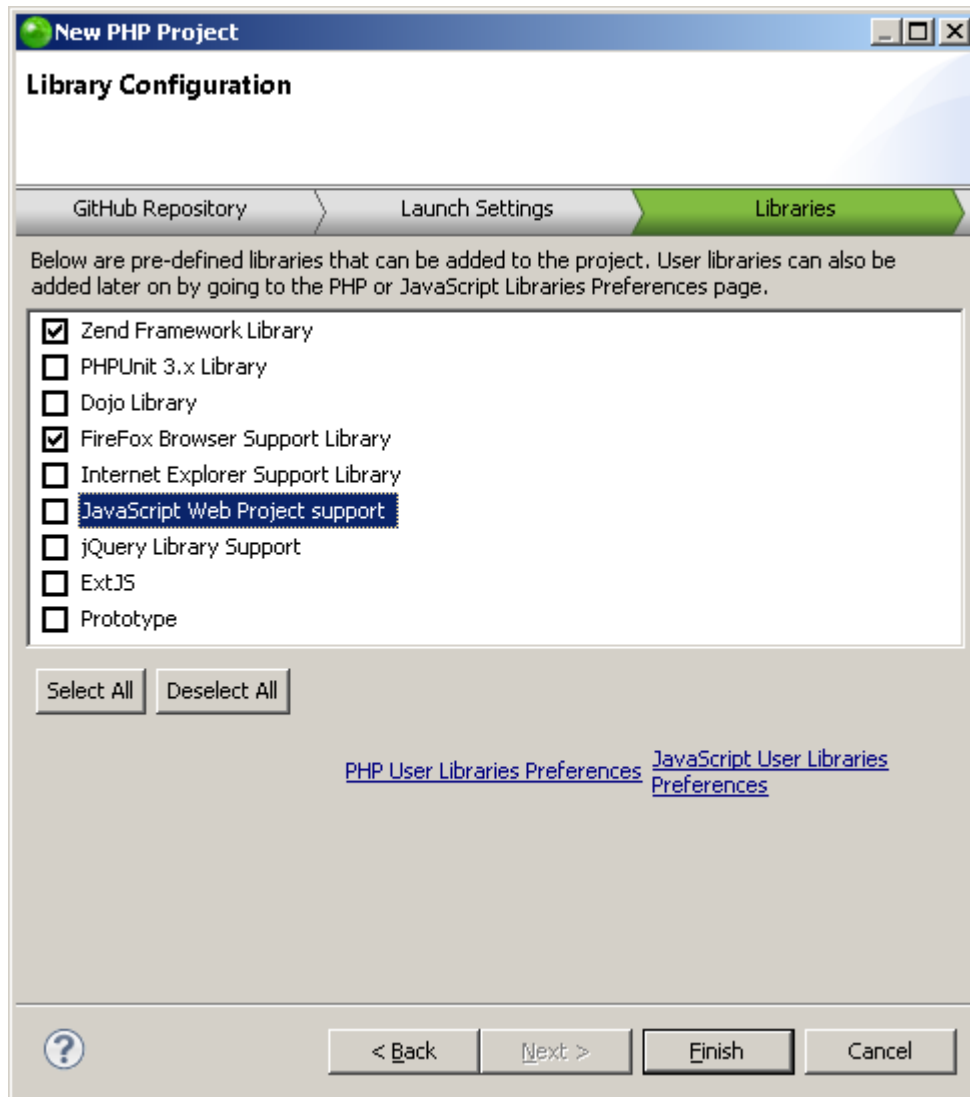
11. Select the libraries you wish to add to the project, and click **Finish**.

Note:
If not already installed, selecting to install jQuery Library Support, ExtJS and Prototype will prompt a restart of Zend Studio.

The new project will be created in your workspace and displayed in the PHP Explorer View. You now have the option to upload or download data to the remote server.

Creating a New PHP Project

Zend Studio offers various options for creating new PHP projects.

If you want to see the full range of options or if you are not sure which project you wish to create you can display all available new project options and select the New Project Wizard that best suits your preferences.

To create a new PHP project:

1. Go to **File** | **New | PHP Project.**

   -OR-

   In the PHP Explorer View, right-click and select **New | PHP Project.**

   The Select a Wizard dialog is displayed.

2. Expand the folders to see full-range of options.
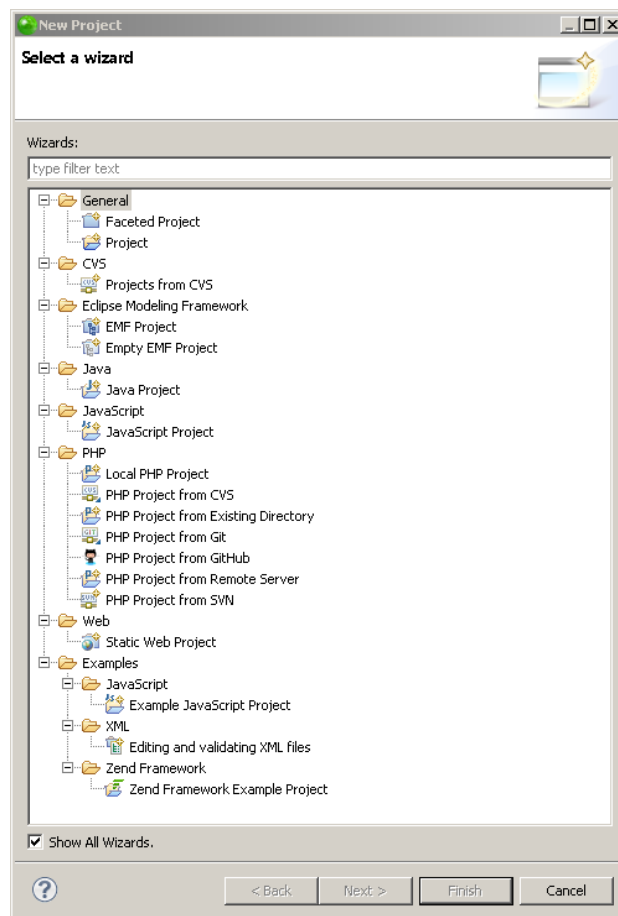
3. Open a Wizard by clicking on the project you want to create.

Creating PHP Files

PHP files can be created and opened in Zend Studio in a number of ways:

- [Creating a new PHP file associated with a project](#)
- [Creating a new PHP file not associated with a project](#)
- [Opening an external file](#)

Creating a PHP File within a Project

This procedure describes how to create a new PHP file within an existing project.

To create a new PHP file within a project:

1. In PHP Explorer view, select the Project within which you would like to place the file.
2. Right-click and select **New | PHP File** -or- go to File on the Menu Bar and select **New | PHP File**.
3. The PHP File creation dialog will be displayed.
4. Enter the name of the file and click **Next**.
5. The 'Use PHP Template' checkbox will be marked by default. This will create the new PHP file with the "<?php ?>" PHP tags.

   Select the required template or unmark the checkbox to create a blank file.
6. Click **Finish**.

Your file will open in the editor and will appear within your project in PHP Explorer and Navigator views.

Creating a PHP File Outside of a Project

Single PHP files can be created outside of a project quickly and easily for the purposes of writing short snippets of code which will not later need to be debugged or run and do not need to be associated with other files or projects.

Creating a New PHP File Not Associated with a Project

To create a new PHP file not associated with a project:

1. From the Menu Bar, go to **File | New | Other | PHP | Untitled PHP Document**.
   -OR-

   click the new Easy PHP File icon on the toolbar .
2. A new PHP file, by default called PHPDocument1, will open in the editor.

Once the file has been created, it can later be saved within a project.

Saving the File to a Project

To save the file to a project:

1. Click save on the toolbar.

   A Save As dialog will open.

2. Select the project with which you would like to associate with the file

    -OR-

    To create a new project:

    i. Click **Create New Project**.

        The New PHP Project dialog will be displayed.

    ii. Enter the Project name and click **Finish**.

        The new project will be added to the list. Select it to save your file within your new project.

    iii. Edit the file name.

    iv. Click **OK**.

Your file will be saved within the selected project and will be available for running, debugging and profiling operations.

Opening an External File

These procedures describe how to open external files in Zend Studio.

External files can be opened in Zend Studio in three ways:

- Dragging-and dropping the file into Zend Studio.
- Double-clicking the file (Windows only).
- Using the Open function in Zend Studio.

Once external files have been opened in Zend Studio, you can perform operations such as running, debugging and profiling on them.

Opening a File Using Drag and Drop

To open a file by dragging-and-dropping:

1. Find your file in your external file system.
2. Have both Zend Studio and your file system explorer open and visible on your desktop.
3. Drag and drop the file into the editor space in Zend Studio.

The file will be displayed in an editor and will be available for operations such as debugging and profiling .

Opening a File by Double Clicking

To open a file by double-clicking:

1. If the file type you are trying to open was associated with Zend Studio during installation, simply double-clicking it in your external file system will cause it to be opened in Zend Studio.
2. If the file type was not associated with Zend Studio you can:
   - Right-click the file and select **Open With | Choose Program | Zend Studio**.
   - -Or- Add the file type to the list of file types which will automatically be opened in Zend Studio by doing the following:
     a. From Zend Studio's Menu Bar, go to **Window | Preferences | General | Content Types**.
        The Content Types dialog will be displayed.

b. Select **Text | PHP Content Type** from the list.

A list of file types associated with Zend Studio will be displayed.
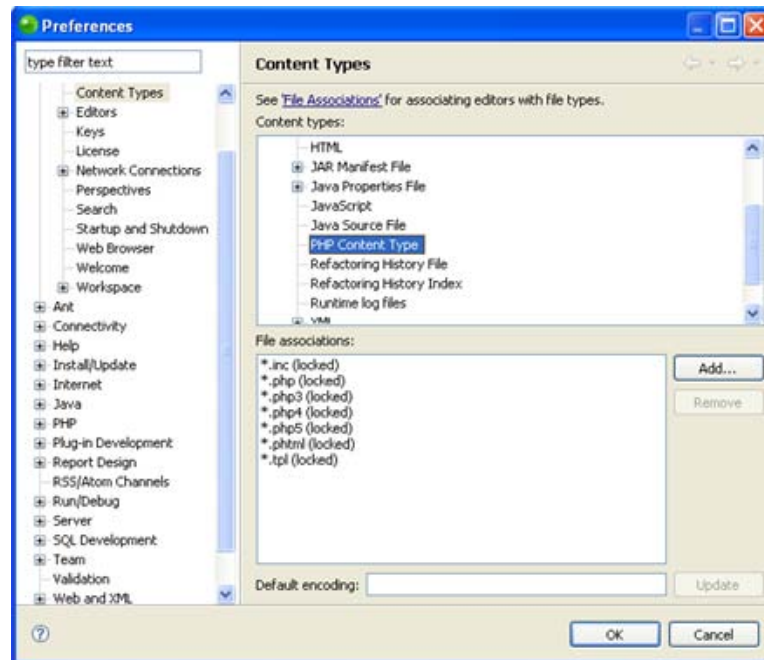
c. Click **Add** to add your file's type to the list,

d. Enter the file type (e.g. .php) and click **OK**.

e. The file type will be added to the list.

f. Open your Windows Explorer.

g. Go to **Tools | Folder Options | File Types**.

h. From the File Types list, select **PHP File**.

i. In the Opens with category click Change, browse to your Zend Studio .exe location and click **OK**.

j. Click **Apply**.

You can now double-click the file on your external file system to open it in Zend Studio.

The file will be displayed in an editor.

Opening a File Using the Open Function

To open a file using Zend Studio's file open function:

1. In Zend Studio, go to **File | Open File**.

2. Browse for your file in your file system.

3. Select the required file and click **Open**.

The file will be displayed in an editor.

Creating PHP Elements

New PHP Element wizards allow for the easy creation of PHP classes and interfaces.

The wizards let you easily configure all required parameters and give you access to Superclasses.

New PHP Element wizards allow you to [create a new PHP Class](#) and [create a new PHP Interface](#).

Creating a New PHP Class

This procedure demonstrates how to create a new PHP class using the new PHP class wizard.

To create a new PHP class:

1. In PHP Explorer view, right-click the project/file in which you want to create the new class and select **New | Class**.

   The New PHP Class wizard is displayed.

2. Enter the following details:

   - Source Folder - If necessary, click **Browse** to change the source folder.
   - Class location - Select one of the following options:
     - Create New File - A new PHP file will be created in which the new class will be inserted

- Add in existing file - The class will be created in an existing file.
  Click **Browse** to select the file in which it will be created and select whether it will be created as the 1st PHP Block in the file or as a New PHP Block at the end of the file.

- Class Name - Enter the name for the class. If you chose the Create New File option, this will also be the name of the file.

- Superclass - Click **Browse** to select a Superclass to extend.

- Interfaces - Click **Add** to select interaces to extend/implement.

- Method stubs - mark the checkboxes of the method stubs to be created (if any) from the following options:

  - Constructor

  - Destructor

  - Inherited abstract methods

- Comments - mark the checkboxes of the comments to be created (if any) from the following options:

  - PHPDpc Blocks

  - TODOs

3. Click **Finish**.

The new class will be created with the required code.
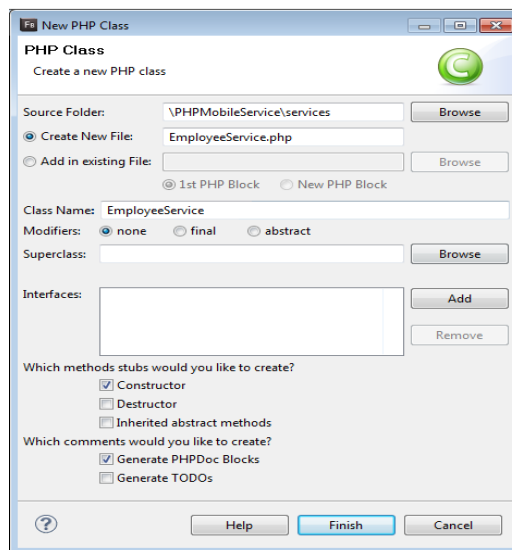
Creating a New PHP Interface

This procedure demonstrates how to create a new PHP interface using the new PHP Interface wizard.

To create a new PHP interface:

1. In PHP Explorer view, right-click the project/file in which you want to create the new interface and select **New | Interface**.

   The New PHP Interface wizard is displayed.
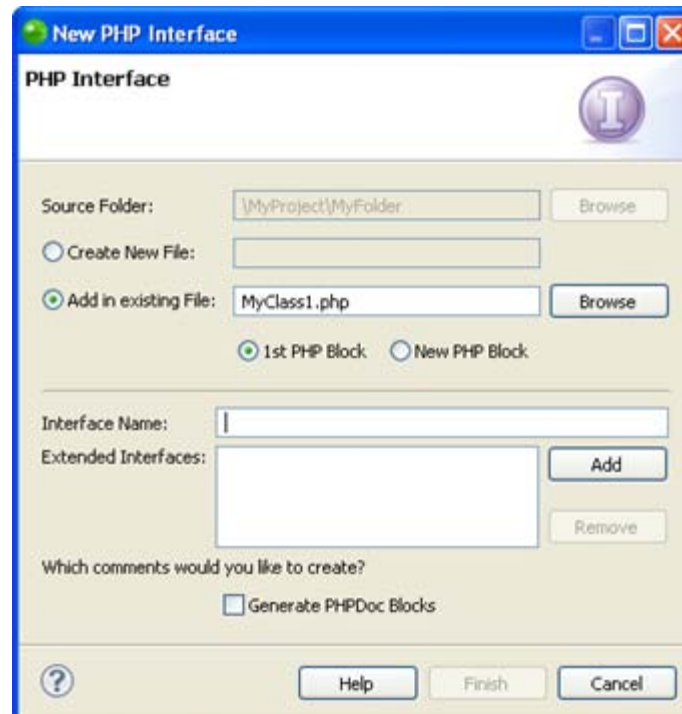
2. Enter the following details:
   - Source Folder - If necessary, click **Browse** to change the source folder.
   - Interface location - Select one of the following options:
     - Create New File - A new PHP file will be created in which the new interface will be inserted
     - Add in existing file - The interface will be created in an existing file.
       Click **Browse** to select the file in which it will be created and select whether it will be created as the 1st PHP Block in the file or as a New PHP Block at the end of the file.
   - Interface Name - Enter the name for the interface. If you chose the Create New File option, this will also be the name of the file.
   - Extended Interfaces - Click **Add** to select interfaces to extend/implement.

- Generate PHPDoc Blocks - Mark this checkbox for PHPDoc Blocks to be automatically created for this item.

3. Click **Finish**.

The new interface will be created with the required code.
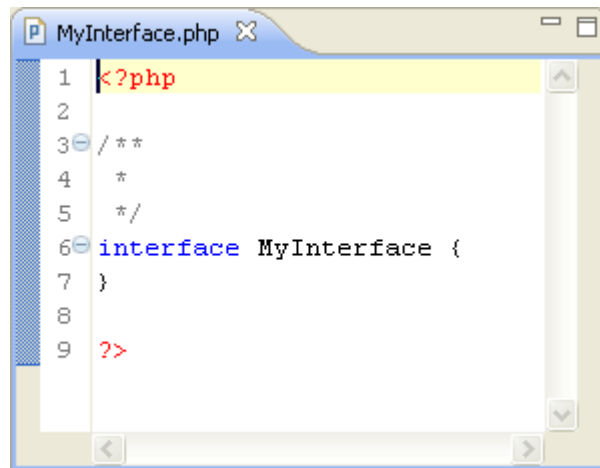
Zend Server Integration

Integrating Zend Studio with Zend Server allows you to benefit both from Zend Studio's debugging and profiling functionality and from Zend Server's PHP Intelligence event monitoring capabilities.

Zend Server monitors and constantly tests your PHP environment and programs in order to allow you to gain maximum efficiency. Instances of problematic scripts and slow execution are captured by Zend Server as 'events'.

For more on events and how to configure what constitutes an event, see the Monitor chapter in Zend Server Online Documentation.

Zend Server's events can be viewed directly from Zend Studio using the internal browser, which displays the Zend Server User Interface.
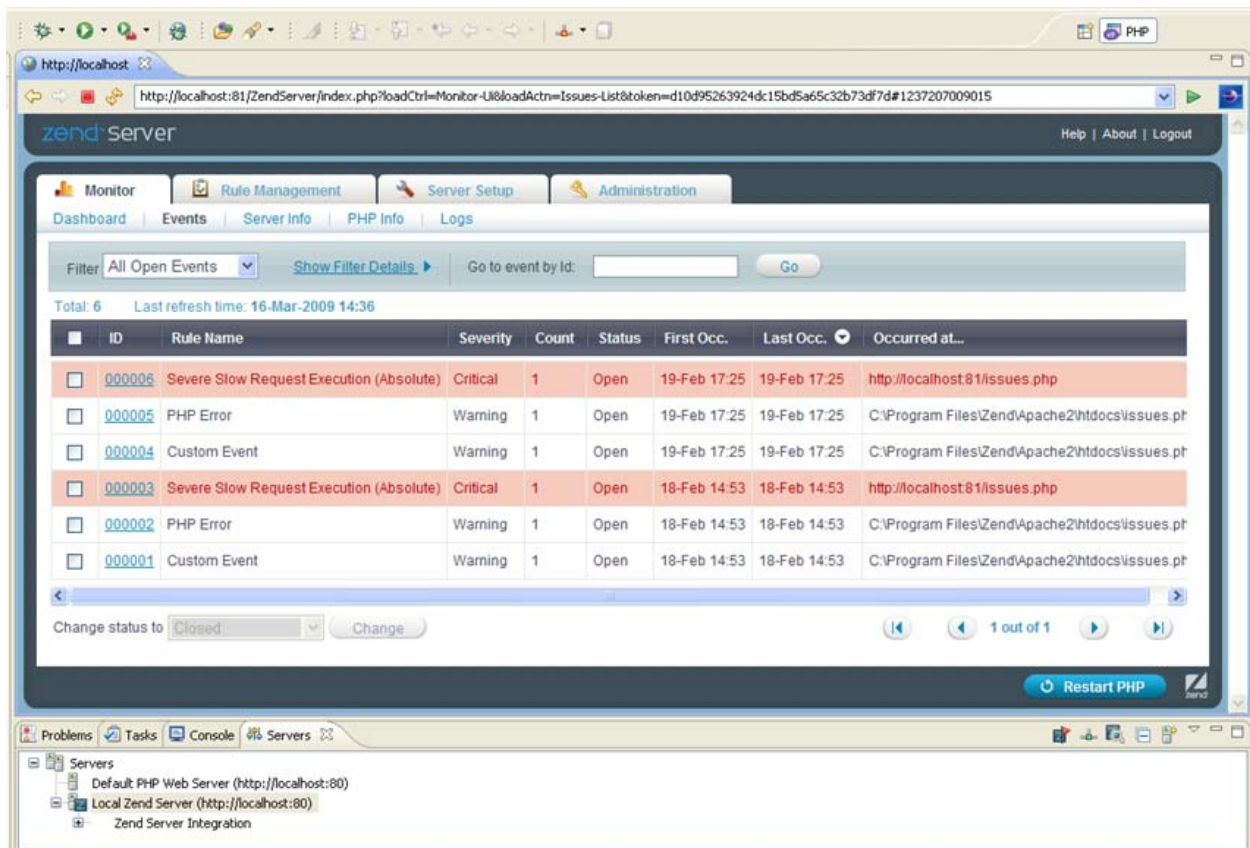
Conversely, Zend Server's integration with Zend Studio means that the problems identified by Zend Server can be viewed, tested, debugged and profiled in Zend Studio.

Note:

If your Zend Server is situated behind a firewall or other security device, you will have to set up a tunnel in order for the integration between Zend Studio and Zend Server to be enabled. See Setting Up Tunneling for more information.

For more on Zend Server go to http://www.zend.com/en/products/server.

Troubleshooting Zend Server Integration

Your Zend Server Integration might not function correctly for one of the following reasons:

- Your Zend Server is not correctly configured in Zend Studio.

  See <u>Defining Zend Server in Studio</u> for more information.

- Zend Server is situated behind a firewall or other security device.

  See <u>Setting Up Tunneling</u> for more information.

- Your Zend Studio Communication is not correctly configured in Zend Server.

  See <u>Configuring Studio Communication Settings in Zend Server</u> for more information.

- Zend Server is situated on an HTTPS Server. In this case, when trying to connect to your Zend Server from the Zend Server Events view, you will receive the following message:

  - For Windows: "Unable to verify the identity of 'http://<myserver>/ZendServer/index.php' as a trusted site."

  - For Linux: "Unable to verify the identity of 'http://<myserver>:10081/ZendServer/' as a trusted site."

Working with Code Tracing

Zend Server Code Tracing captures full execution data (trace data) of PHP applications in real time. The execution data includes function call trees, arguments and return values, function execution duration, memory usage and indication for an executed files name and line of code. This enables you to capture problems when they occur, which eliminates the need to set up environments and reproduce the steps that led up to the failure. Integrating Code Tracing into Zend Studio allows you to open the source of the execution data inside of your environment. This feature is useful in resolving performance issues, memory usage issues, and functional errors that occur in a production environment. Using Code Tracing you can import a Zend Server event file and view the execution data which it contains in an editor. Code Tracing uses the Code Tracing Perspective to allow you to view the trace data within your environment.

The Code Tracing feature allows you to do the following:

- Export Trace Information (Located in the Zend Server Online Documentation)
- Import a Zend Server Event File
- Open the Source of Trace Data

Importing Zend Server Event Data

Importing Zend Server Event Data allows you to view and analyze trace data (execution data) from your server. The execution data includes function call trees, arguments and return values, function execution duration, memory usage and indication for an executed files name and line of code. This enables you to capture problems when they occur, which eliminates the need to set up environments and reproduce the steps that led up to the failure.

Zend Studio allows you to import a .amf, .xml or .zsf file:

- A .amf file is a trace data file which can be opened without access to Zend Server. The trace file opens in the Code Tracing Perspective where you can locate errors directly in the trace data and then Open the Source of Trace Data. You may then edit the source inside of the corresponding local file.

- A .xml file is an event configuration file which allows you to recreate and debug an error when you have access to a Zend Server on which the application exists.

- A .zsf file is a unified event file which may contain .amf files and/or a .xml files. To import a .zsf file, follow the procedure Importing Zend Server Event Data if you would like to use an included .amf file to view and analyze trace data, or follow the procedure Importing a Zend Server Event File to use an included .xml file to recreate the error using the event details.

Note:

In order to import Zend Server Event Data, you must first export Zend Server Event Data from your server. For more information see Exporting Trace Information in the Zend Server Online Documentation.

Having an active connection with a licensed version of Zend Server or Zend Server Cluster Manager allows you to fully integrate the Code Tracing feature. For more information see Zend Server.

Importing Zend Server Event Data (amf)

This procedure describes how to import an amf trace data file into Zend Studio. Trace data in an amf format can be imported and used to Open the Source of Trace Data even when there is no active connection with a Zend Server which contains the application.

To import a .amf Zend Server Event File:

1. To open the Import Wizard go to **File | Import | Zend Server | Zend Server Event File**.
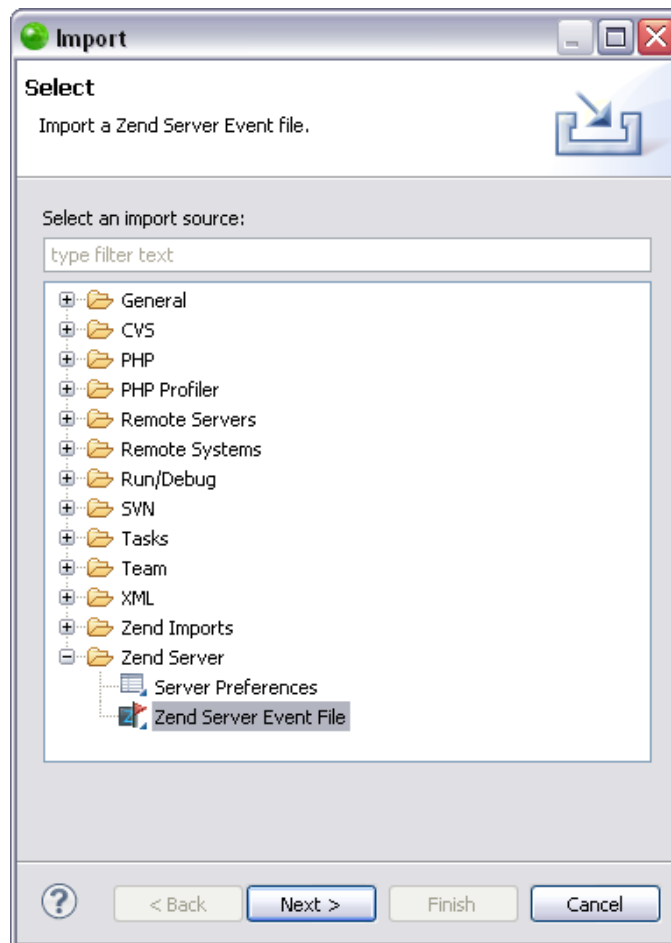
2. Click **Next**.

   The "Import Zend Server Event File" dialog opens.
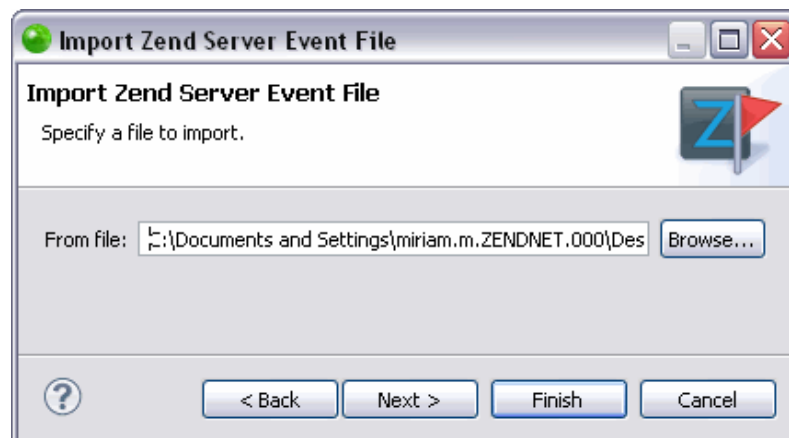
If you have downloaded a .zsf file containing both a .amf file and .xml file, a message will appear after clicking **Next** in the Import Zend Server Event File dialog asking you which action you would like to take. To use the .amf file to view and analyze trace data, click **Trace**.

3. In the "From File" text field, browse to the location of your Zend Server Event File and click **Next**.

The "Server path mapping setting" dialog opens.
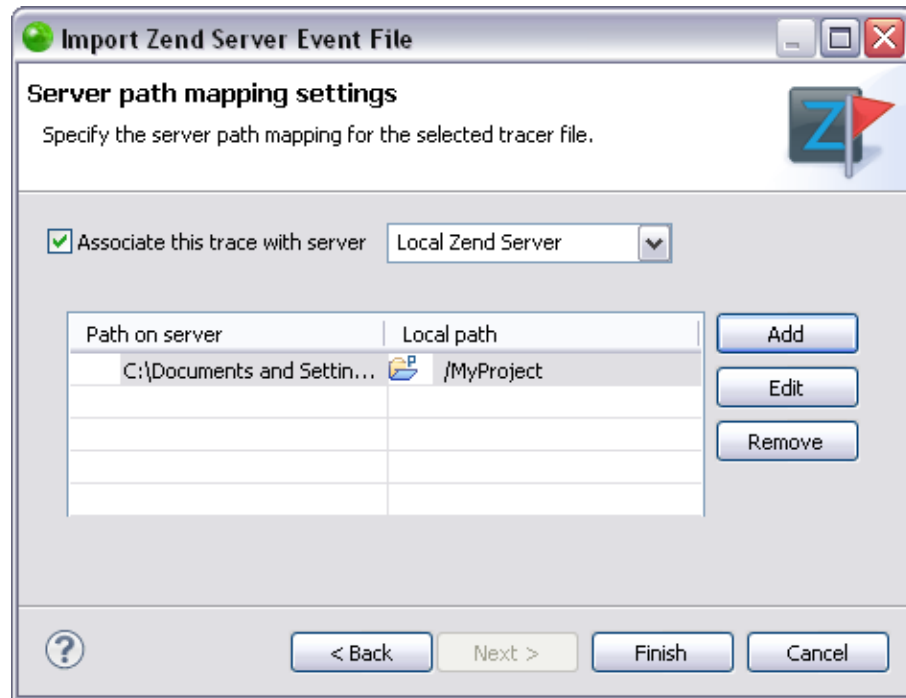
4. Select the server to associate with from the "Associate this trace with server" drop down menu.

5. You can add, edit or remove a path map from this page using the appropriate buttons. You may select to have a path in your workspace or in the file system. For more information see Adding a New Path Map for Importing a Zend Server Event File.

6. Click **Finish** to save the changes.

7. Click **Yes** when the "Confirm Perspective Switch" message appears asking your permission to open the Zend Server Code Tracer perspective.

The Code Tracing Perspective opens with the imported Zend Server Event File open in the Tracer View.

For information about the functionalities available in the Zend Server Code Tracer perspective see Working with Code Tracing.

Importing a Zend Server Event File (xml)

This procedure describes how to import an xml event file into Zend Studio. You can import an xml format Zend Server Event File to recreate and debug an error. The Zend Studio functionality for an imported xml Zend Server Event File is only available when you have access to a Zend Server on which the application exists.

To import a .xml Zend Server Event File:
1. To open the Import Wizard go to **File | Import | Zend Server | Zend Server Event File**.
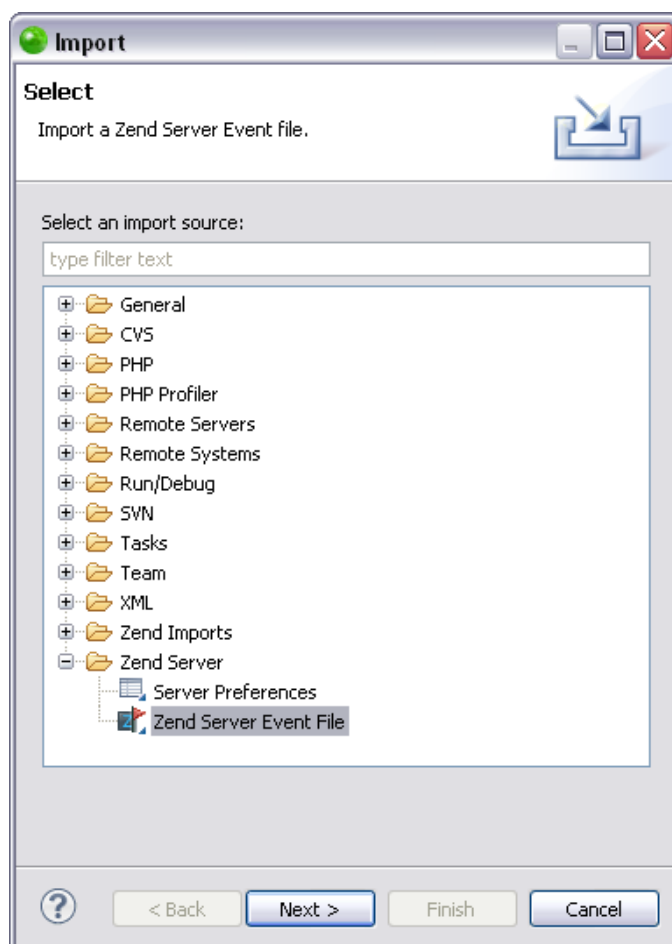
2. Click **Next**.

    The "Import Zend Server Event File" dialog opens.
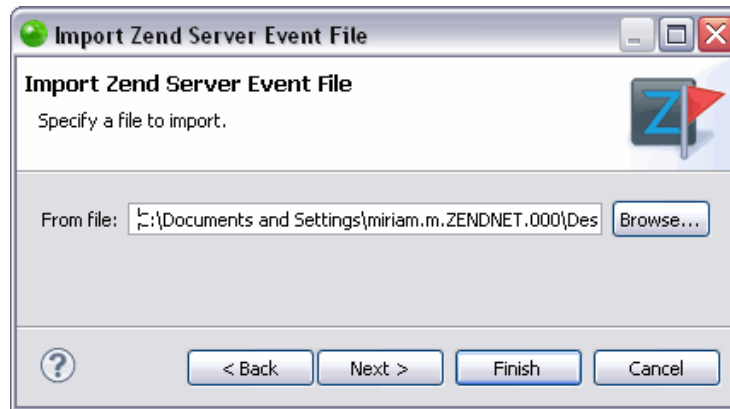
If you have downloaded a .zsf file containing both a .amf file and .xml file, a message will appear after clicking **Next** in the Import Zend Server Event File dialog asking you which action you would like to take. To use the .xml file to recreate the error using the event details, click **Event**.

2. In the "From file" text field, browse to the location of your Zend Server Event File and click **Next**.

    The Import Zend Server Event File dialog opens.
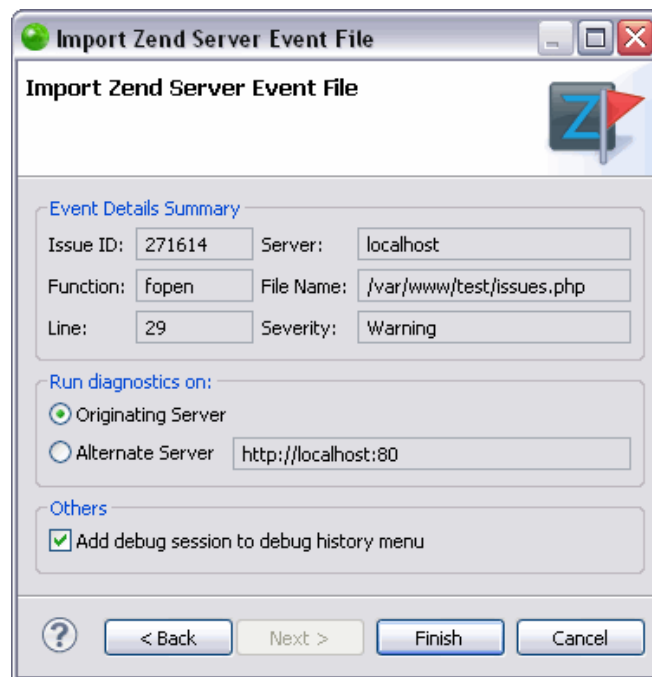
3. The dialog contains the following information:

- Event Details Summary
  - o Issue ID - A unique number assigned to each event in Zend Server. This number is displayed next to each event in the [Events](#) page.
  - o Server - The name or host of the originating server.
  - o Function - Displays information on the function that triggered the error.
  - o File Name - The original location where the event occurred.
  - o Line - The line in the file (which is specified in the File Name field) that created the event.
  - o Severity - The severity of the event (Warning or Critical). The severity is defined in the event's master settings in the Monitor tab. For more information see [Monitor](#) in the [Zend Server Online Documentation](#).

- Run Diagnostics on
  - o Originating Server - Choose this option to run the diagnostics on the originating server. You must have a working connection with the originating server if you choose this option.
  - o Alternate Server - Choose this option to run the diagnostics on an alternate server, and enter the details of your server.

Note:

In order to run the diagnostics on an alternate server, the alternate server must also contain the application that generated the events.

- Others
  - Add debug session to debug history menu - Adds your debug session to the debug history, which you can view by opening the dropdown menu of from the main toolbar.

3. Click **Finish** to save the changes.

The [PHP Debug Perspective](#) opens and the debug session begins.

For information about how to evaluate the debugging results see [Running and Analyzing Debugger Results](#).

Opening the Trace Data Source

Opening the source of trace data (execution data) allows you to see the source of the functions inside Zend Studio. This functionality allows you to pinpoint exactly where in the source the event is occurring. You may then edit the source inside of the corresponding local file.

Important Note:
To open the source of trace data you must first Import a Zend Server Event File for which the project already exists locally in your Zend Studio.

Opening the source of trace data:

1. To open the Zend Code Tracer perspective go to **Window | Open Perspective | Other | Code Tracing.**
2. In the Tracer view go to the Statics per Function tab.
   A list of the functions appears. To see the memory usage of each function mark the "Show memory usage" option in the view.
3. Double click on a function.
   The "Open Resource" dialog opens.
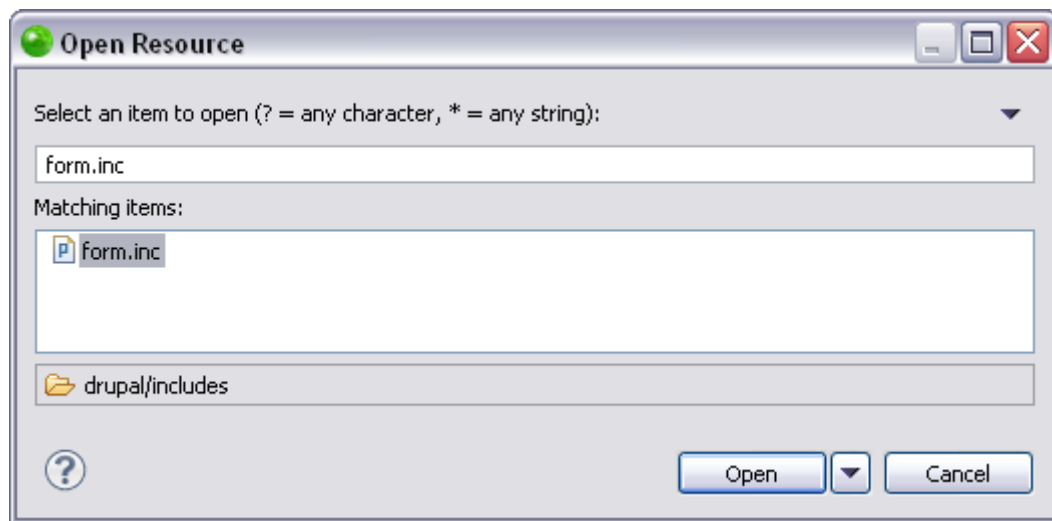
4. Use the drop down menu in the dialog to:
   ▪ Show the status line - Shows the status line in the current window.
   ▪ Show the derived resources - Resources that are not original data but can be recreated from their source files. For more information see Derived Resources in the Eclipse Online Documentation.

- Select/Deselect/Edit a working set - For information about working sets see <u>Working Sets</u> in the <u>Workbench User Guide</u> .

  You can also use the **Open** button drop down menu to open the resource in a:

- PHP editor

- Text editor

- System editor

- In-Place editor

- Default editor

- Other - Opens the "Editor Selection" dialog and allows you to choose the editor (either an internal editor or an external program) to open your resource.
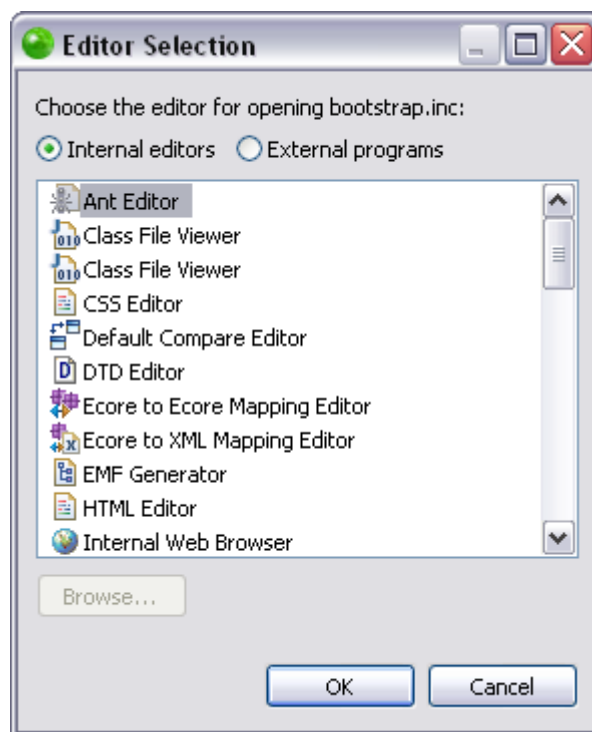
5. Select the resource and click **Open**.

   The source code of the resource opens in an editor view.

The source of the resource is read only. To edit a resource open the local project file in which it is located.

Debugging and Profiling Zend Server Events

Zend Server events can be debugged and profiled in Zend Studio either directly from Zend Server, using Zend Server integration, or by exporting an event from Zend Server and importing it into Zend Studio .

- Importing a Zend Server Event File
- Debugging Events from Zend Server

Debugging / Profiling Events from Zend Server

Zend Studio allows integration with Zend Server so that Zend Server Events can be viewed, debugged and profiled in order to detect and locate errors and issues causing slow script execution.
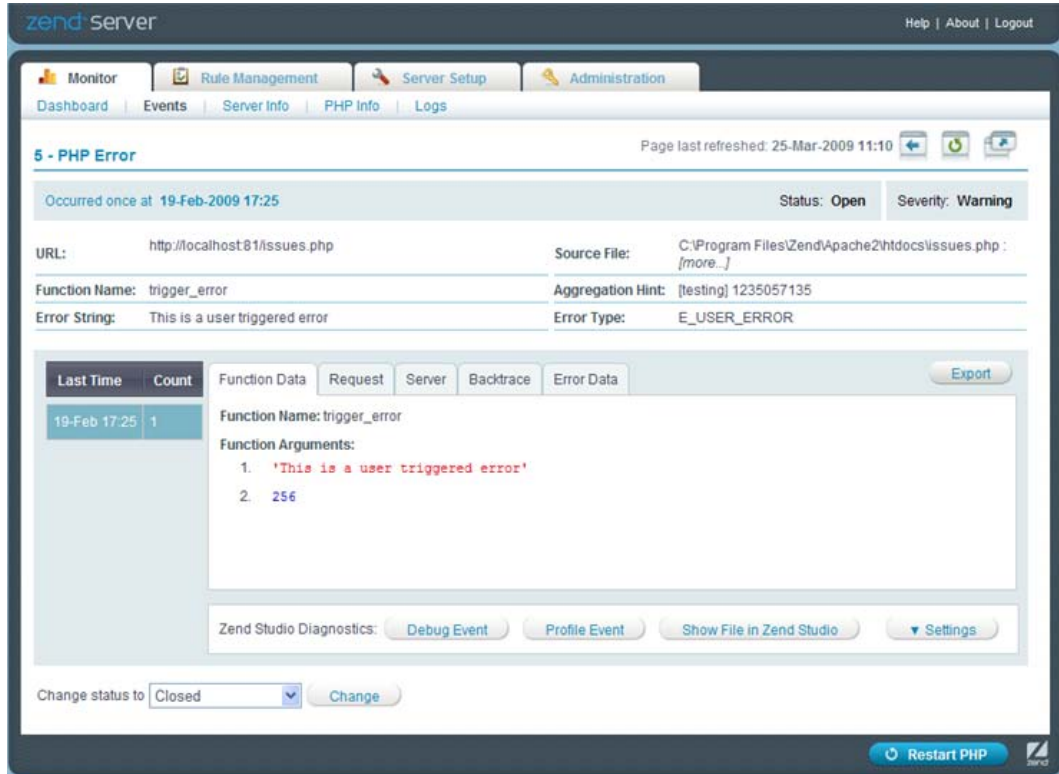
To debug an event directly from Zend Server :

1. Open your Zend Server Event List.

2. Access the Event list by browsing to the **Monitor | Events tab**.
3. Click the Event which you want to debug.

4. To select the server on which the debug/profile session will be run, click the ▼ Settings button.

    This gives you access to the following options:

    - Originating server - This will debug the event on the server from which the event originated
    - Alternate server - Allows you to debug the event on a different server (this server must also be running the Zend Debugger). Enter the IP address of the required server.

5. Click the **Set** button to save your settings.

6. Click the **Debug Event** or **Profile Event** button.

7. The relevant debug / profile session is launched in Zend Studio.

Note:
If Zend Server cannot connect to Zend Studio , see both the see the Setting Up Zend Server Integration topic and the 'Error: Failed to Communicate with Zend Studio' topic in the Zend Server Online Help (http://files.zend.com/help/Zend-Serverzend-server/htm) for more information.

See Running and Analyzing Debugger Results for more information on running a debug session or the PHP Profile Perspective topic for more on the information displayed once a Profile session has been run.

Setting Up Zend Server Integration

In order to debug events through Zend Server, the integration between the Zend Server and your Zend Studio needs to be configured in both products.

See Configuring Zend Server Settings in Zend Studio and Configuring Zend Studio Settings in Zend Server for more information.

> **Note:**
>
> If you do not have access to the Zend Server on which the events were created, you can importing a Zend Server Event File and opening the source of the Trace Data to locate errors.

Configuring Zend Server Settings in Zend Studio

About

Zend Server settings can be configured in Zend Studio in order to allow the appliance of Zend Studio functionality (Profiling, Debugging etc.) to Zend Server Events, as well as allowing access to Zend Server's Event list.

Zend Server configuration can be done either automatically, if the Zend Server is installed on the same machine on which Zend Studio is running, or manually through the PHP Servers Preferences page.

Automatically Configuring Zend Server

Zend Server s installed on the same machine as Zend Studio are automatically detected and configured in Zend Studio.

To automatically configure a Zend Server :

The auto detection is triggered when Zend Studio is launched or when the Auto Detect Zend Server button is clicked.

For auto detection when Zend Studio is launched:

1. Ensure Zend Server is installed and running on the local machine.
2. Open Zend Studio.

   A popup balloon will appear in the bottom-right corner of the window indicating that a Zend Server installation has been detected and configured.
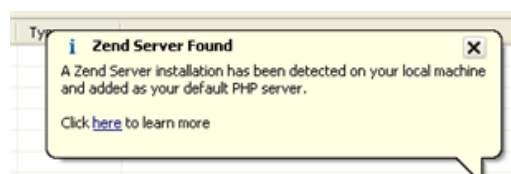
3. Click the ❌ icon to close the balloon or the 'click here' link to be taken to the Zend Server Integration help page.

A Local Zend Server configuration is configured and added to your PHP Servers Preferences page.

Configuring the Automatically Detected Zend Server

In order to access the Zend Server GUI and Zend Server Event List from Zend Studio, you will need to configure the Zend Server GUI password for the newly configured Zend Server by performing the following procedure.

To configure the automatically detected Zend Server :

1. Open the PHP Servers preferences page by going to **Window | Preferences | PHP | PHP Servers**.
2. Select the Zend Server from the list (by default, it will have been named Local Zend Server).
3. Click **Edit**.
4. Go to the Zend Server tab.
5. In the Authentication category, enter the password used to access the Zend Server GUI.
6. Click **Apply** and **OK**.

You will now be able to access the Zend Server Event list by selecting the Zend Server from the drop-down list next to the Zend Server icon on the toolbar 🇿🔽.

Manually Configuring Zend Server

To manually configure a Zend Server:

1. Open the PHP Servers Preferences page by going to **Window | Preferences | PHP | PHP Servers**.
2. Click **New** to create a new server with Server Integration.
   A "PHP Server Creation" dialog will open.
3. Configure the server as described in the PHP Server Preferences page (enter the server's name and the URL of your document root.)

   > **Note:**
   > If you are unsure of your Zend Server's document root, see the Zend Server FAQ to find out what your document root is.

4. Click **Next**.
5. If necessary, define Path Mapping. See Managing Path Maps for more information.

Zend Studio 9 User Guide

6. Click **Next**.

The "Zend Server Configuration" dialog is displayed.
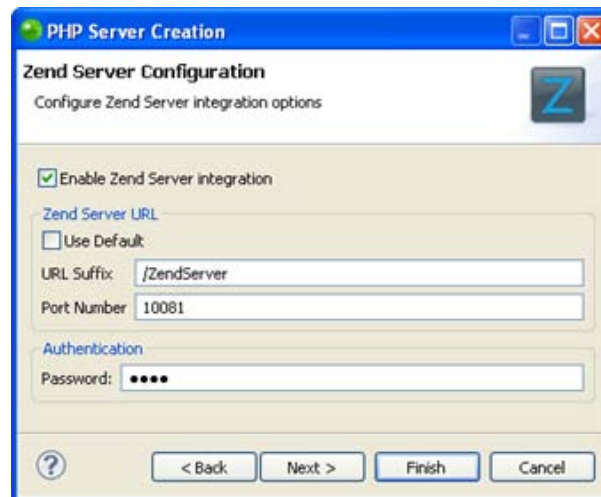
7. Mark the Enable Zend Server integration checkbox to enable Zend Server integration functionality.
8. Leaving the Use default checkbox marked will create a URL in the format <server's document root>/ZendServer>. If necessary, unmark the checkbox and edit the following:
   - URL Suffix - The suffix which should be added to the URL of your document root in order to browse to your Zend Server GUI.
   - Port number - Enter the port number you defined during Zend Server installation. See the Zend Server FAQ site for default port number settings.
9. Enter the password for your Zend Server GUI.
10. Click **Next** to configure Tunneling settings or **Finish** to create your server.

Your Zend Server is added to the Zend Server list and will allow you to use Zend Server integration features.

Your Zend Server is now available from the arrow next to the Zend Server icon on the toolbar.

Configuring Zend Studio in Zend Server

In order for you to be able to debug applications situated on Zend Server , your Zend Studio must be configured as an Allowed Host for initiating debugging settings.

To configure Zend Studio communication in Zend Server :
1. Open your Zend Server GUI.

> **Note:**
> This can be done from within Zend Studio through the Servers view or by selecting the server on which you have configured your Zend Server integration from the drop-down list next to the Zend Server icon on the toolbar.

2. Go to the Zend Server **Setup | Debugger tab**.
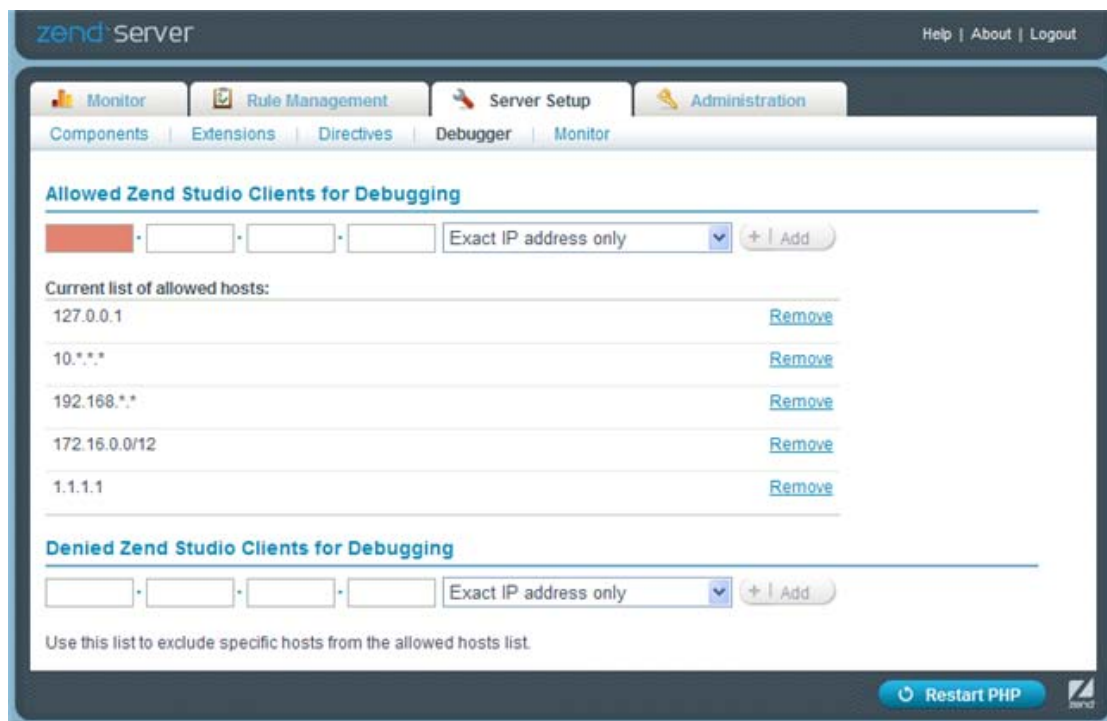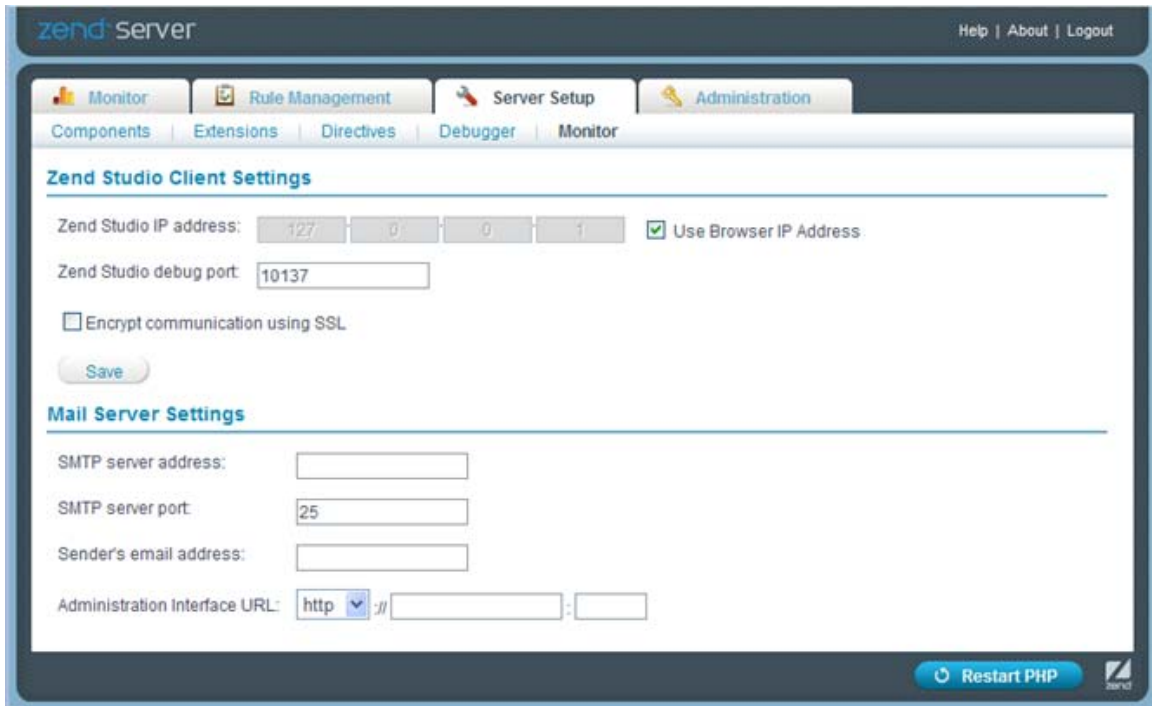
3. Ensure the IP address of your Zend Studio is included in the Allowed Hosts sections.
   To add an address to the list:
   i. Under the Allowed Zend Studio Clients for Debugging category, enter the IP of the machine on which your Zend Studio is installed
   ii. Click **Add**.
   The IP Address is added to the Allowed Hosts list.

4. Ensure your Zend Studio's IP address is not in the Denied Hosts list.

   If it is, click **Remove** next to the required address to remove it from the list.

5. Click the ⬭ **Restart PHP** button to apply your settings.

6. Browse to the **Server Setup | Monitor** tab.

7. Configure the following settings under the Zend Studio Client Settings category:

   ▪ **Zend Studio IP address -** Enter the IP address of the machine on which your Zend Studio is installed or mark the Use Browser IP address to use the IP address of the machine on which the Zend Server UI is running.

   ▪ **Zend Studio Debug Port -** This should match the debug port configured for the Zend Debugger in Zend Studio 's Installed Debugger Preferences page (**Window | Preferences | PHP | Debug | Installed Debuggers**) page. The default port number is 10137.

   ▪ **Encrypt communication using SSL** - Mark this checkbox for the communication between Zend Studio and Zend Server to be encrypted using SSL.

8. Click the ⬭ **Restart PHP** button to apply your settings.

Configuring Studio Communication Settings in Zend Server

In order for you to be able to view, debug and profile Zend Server Events in Zend Studio, you must ensure the correct communication settings are configured in your Zend Server.

To configure Zend Studio communication in :

1. Open your Zend Server GUI.
2. Go to the **Server Setup | Debugger tab**.

3. Ensure the address of your Zend Studio is included in the Allowed Hosts sections. This will ensure you can debug/profile Events.

   To add an address to the list:

   i. Enter the IP address or Net mask of the machine on which your Zend Studio is installed.In Order to enter a Net mask, enter a range by entering the beginning of an IP address and adding '0' instead of the rest of the number. To make sure you are using Wildcards (*) to specify a range of IP's, select the pattern you want from the drop-down list.

   ii. Click **Add**. Your Zend Studio machine's address will be added to the Allowed Hosts list.

4. Ensure your Zend Studio's IP address is not in the Denied Hosts list.

   If it is, click **Remove** next to the required address to remove it from the list.

5. In the Zend Server GUI, go to the Server **Setup tab | Monitor** and configure the following:

- Auto detect the Zend Studio Client Settings - Set to 'On' to inform Zend Server of the method of connection to Zend Studio. This allows Zend Server to automatically detect your Zend Studio Debug settings.

6. Click **Save**.

7. Restart your Web Server for the settings to take effect.

Configuring Zend Server to Auto Detect Zend Studio Settings

This procedure describes how to configure Zend Server so that Zend Studio's settings are automatically detected during the Debugging/ Profiling of Zend Server events.

To establish a communication tunnel between Zend Server and Zend Studio:

1. Open your Zend Server GUI.

2. Go to the **Server Setup | Monitor tab**.

3. In the Zend Server Settings section, configure the following:

- Auto detect the Zend Studio Client Settings - Set to 'On' to inform Zend Server of the method of connection to Zend Studio. This allows Zend Server to detect your Zend Studio Debug settings.

4. Click **Save**.

Using Content Assist

This procedure describes how to use Content Assist in order to quickly and easily insert code elements into your script:

To use Content Assist:

1. Enter the first few characters of the required code element into the editor.

   The Content Assist window should be automatically displayed.

2. If the Content Assist window does not pop up automatically, press **Ctrl+Space**.
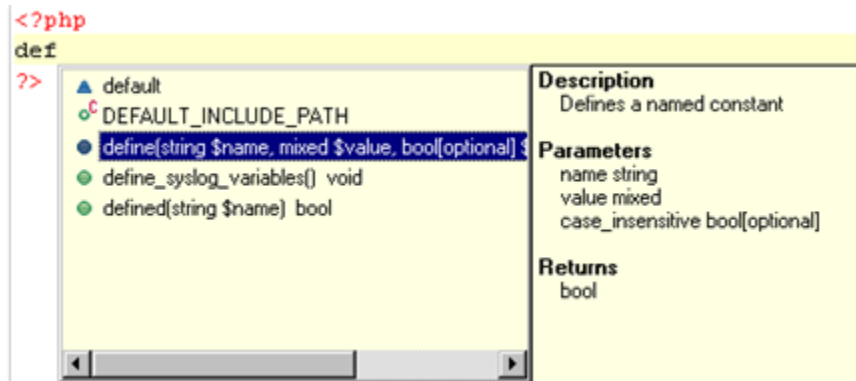
3. Use the arrow keys to scroll through the code completion options. The window on the right will display descriptions and hints for using the selected code element.

4. Select the required option by double-clicking it or selecting it and pressing **Enter**.

The selected code will be inserted into your script.

To enable the Content Assist window to open automatically, go to the Content Assist Preferences page, accessed from **Window | Preferences | PHP | Editor | Content Assist** and mark the "Enable auto-activation" checkbox. This is marked by default.

Using Templates

Templates are shortcuts used to insert a pre-defined framework of code into your scripts. The purpose is to save time and reduce the potential for errors in standard, repetitive code units. Once a template is inserted, you can complete the code quickly using manual and automated code entry methods.

Requirements:

A template must be defined in the Templates list in the Template Preferences page, accessed from **Windows | Preferences | PHP | Editor | Templates**, before it can be used. To learn how to create a new template from the Templates Preferences page, see Adding a New Template.

Note:
The i5 edition of Zend Studio includes pre-defined templates for using i5 PHP API Toolkit functions. See i5 Edition Extras for more information.

Templates are context sensitive and can be used in HTML, PHP, PHPDOC, JavaScript or CSS. The context of the current code being entered defines which templates are available. For example, PHP templates are not available if your current code is Java.

Inserting a Template into Code

This procedure describes how to insert a template into your script.

To insert a template:

1. Place your cursor at the desired insertion point.
2. Enter a character string (e.g. "Sw").
3. Click **Ctrl+Space**.

   The Content Assist box will appear, listing all available templates and completion options that begin with that combination of keys.

   Templates are marked in the content assist list with a blue square.

4. Double-click the required template from the list.

   The template will be entered into your code.

Example:

Entering "sw" and selecting the "switch statement" template from the list will give you the following code:

```
2  switch ($var) {
3      case value:
4          ;
5      break;
6
7      default:
8          ;
9      break;
10 }
11 sw
```

Note:

Templates can be created, imported and exported through the [Template Preferences page](#), accessed from **Window | Preferences | PHP | Editor | Templates**.

Drag and Drop

The Drag and Drop functionality allows you to click on a selected chunk of code and drop it anywhere in the editor. This not only helps you work more efficiently, but also helps minimize the errors that are created when editing or cutting/pasting your code.

Drag and Drop is available in PHP and JavaScript editors.

To Drag and Drop a chunk of code in an editor:

1. Highlight the chunk of code you want to move with your cursor.
2. Click and hold down your mouse within the highlighted code to grab it, and drag the chunk to the selected line in the editor.
3. Release the mouse to place the chunk of code in its new location.

```php
<?php
$releasetag = "M7";
$version = array("Windows 32-bit" => "win32.zip", "Linux x8
$sizes    = array("Windows 32-bit" => "141M", "Linux x86/GTK

print '<a href="http://www.eclipse.org/pdt/downloads/?show
```

For more information on editors see the Editors topic in the Workbench User Guide.

Formatting Code

Zend Studio can auto-format your code according to set standards in order to make it easily navigable and readable.

Your code will be automatically formatted according to the settings defined in the PHP Formatter Preferences page, accessed from **Window | Preferences | PHP Formatter**.

This procedure demonstrates how to format your scripts.

Formatting Your Whole Script

To format your whole script:

1. Open the required file.
2. Go to **Source | Format Document** or press **Ctrl+Shift+F**.

Your code will be automatically formatted.

Example:

Unformatted Code	Formatted Code

Formatting Selected Lines within the Script

To format only selected lines within the script:

1. Select the relevant lines.
2. Go to **Source | Format Active Elements** -or- press **Ctrl+I**.

Only the selected lines will be formatted.

Note:

Code Formatting will also be available in JavaScript editors.

JavaScript Formatting preferences can be configured from **Window | Preferences | Web | JavaScript | Code Style | Formatter**.

Using Code Folding

Code Folding collapses or "folds" the display of a block of code.

To enable code folding, go to the [Folding Preferences](#) page, accessible from **Window | Preferences | PHP | Editor | Using Code Folding** .

If Code Folding is enabled, minus signs will appear in the Annotation Bar next to code blocks which can be folded. In addition, certain elements will be folded by default according to the Folding Preferences settings.

Folding a Block of Code

To fold a block of code:
1. Stand within a class, function or PHPDocBlock.
2. Click the **minus sign** on the marker bar to the left of the editor.

The first line of code will remain visible but the other lines will not be displayed. A fold indicator ▢ will appear at the end of the line when the code is folded to indicate that there is hidden code.

To temporarily view folded code, hover over the **plus sign** that denotes a folded block. The folded code will be displayed in a floating block.
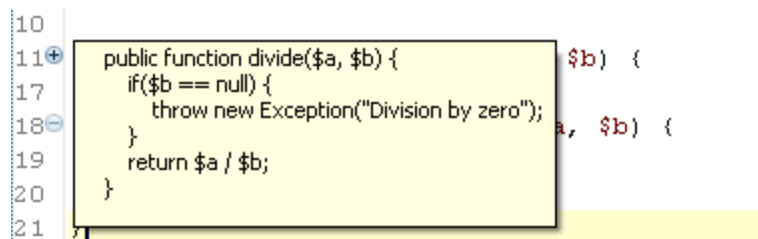
Unfolding a Block of Code

To unfold a block of code:
1. Click the **plus sign**.
2. The folded code will become visible again and the fold indicator will disappear.

To view the scope of a fold:
1. Hover over the **minus sign**.

2. A vertical line will be displayed from the first to the last line of the fold, indicating its range.

An unfolded function   A folded function

Folding/Unfolding Nested Functions

To fold/unfold nested functions:

1. Click on one of the **minus signs** of a nested function. All levels below this level will be folded into it. You can continue to fold until all levels have been folded into the topmost level.

2. To unfold nested functions, click on the **plus sign**. The folded code will open in the same order that it was folded.

Note:
Line numbers are folded together with the code. Folding and unfolding does not change line numbers, it can only hide/display them.

Note:
If the folded code contains an error, the displayed window will be syntax highlighted on both the left and right Annotation bars.

213

Searching for PHP Elements

Searching for PHP elements is a functionality that allows you to search for PHP elements with your defined specifications. PHP elements are classes, functions, constants, types, methods and references/declarations that are used in your code. Use this option if you want to locate one of these elements in your workspace, project or in a single file.

Note:

For information about the "Search" tab in the "Search" dialog, read below. To learn more about the additional tabs in the "Search" dialog see File Search in the Workbench User Guide. If you are using the Remote Search option, be aware that it does not search all of the resources of the remote server.

This procedure describes how to search for PHP elements (classes, functions and constants) within your files and projects.

Searching for a PHP Element

To search for a PHP element:

1. From the Menu Bar go to **Search | Search** -or- press **Ctrl+H**.

   The "PHP Search" dialog will open.
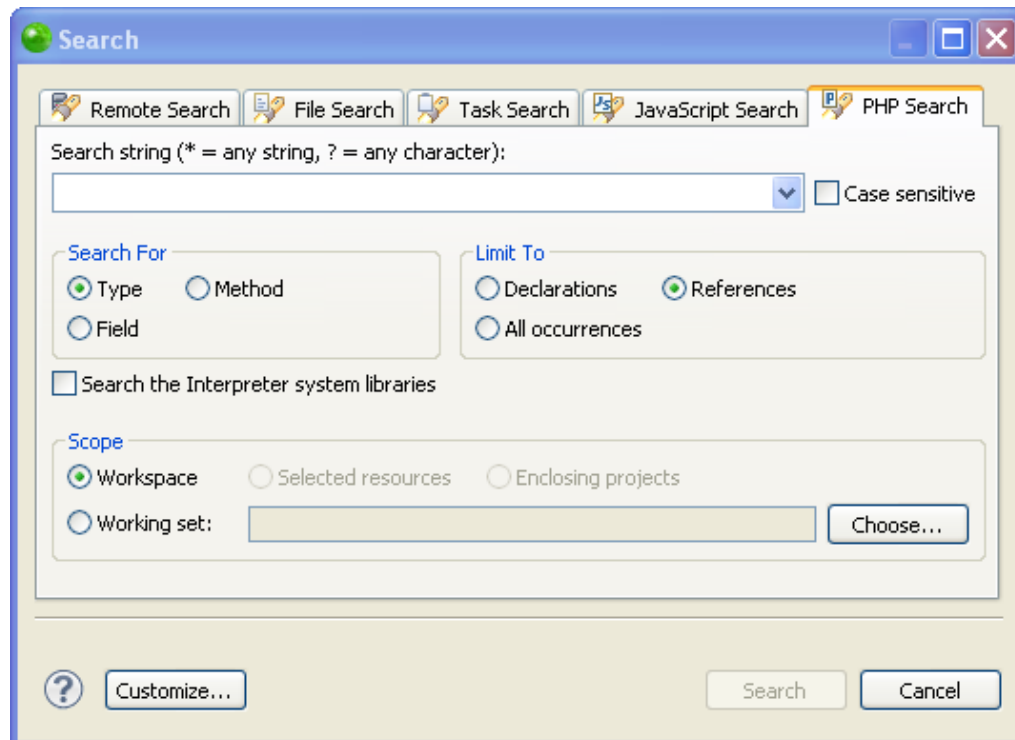
2. Enter a search string. You have the option of using Wild Cards.

   You can make your search case sensitive by selecting the "Case sensitive" box.

3. Select whether to search for a:

   - **Type** - A class (This option will search for both methods and fields.)
   - **Method** - A function
   - **Field** - A constant

4. Select whether to limit your search to:

   - **Declarations** - The location where the type, method, or field are defined.
   - **References** - Anywhere the type, method, or field are referred to.
   - **All occurrences** - Anywhere the type, method, or field are referred to or declared.

5. To extend your search to include interpreter system libraries, select the "Search the Interpreter system libraries" box. This allows Zend Studio to search in the libraries of the PHP version you have selected in the PHP Interpreter Preferences.

6. Select whether to search in your:

   - **Workspace** - The entire workspace.
   - **Selected resources** - Select these in PHP Explorer view before opening the "Search" dialog. All sub-files and sub-folders within the selected resource will be included in the search.
   - **Enclosing projects** - The projects which the selected resources are in.
   - **Working Set** - Click **Choose...** to select the required Working Set.

7. Click **Search**

The Search view will open displaying the results of the search.

To go to an element, double-click the required option from the search view.
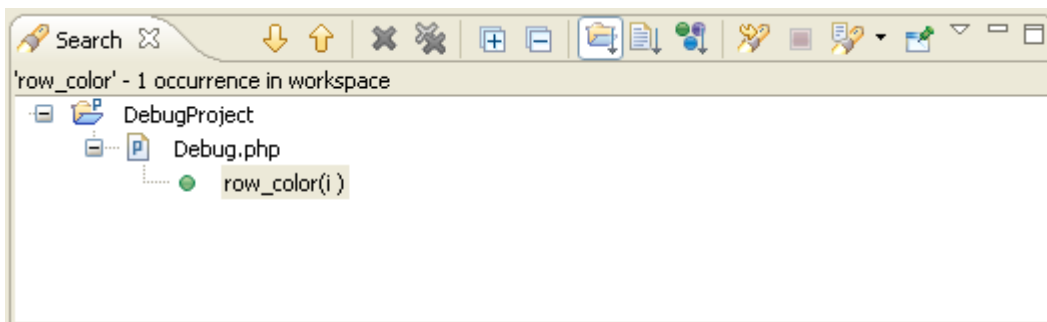
Note:

By default, the File Search, Remote Search, Task Search, and JavaScript Search dialogs will be tabbed with the PHP Search dialog. To make these unavailable, click **Customize...** within the PHP Search dialog and unmark required options.

Hotkeys

The following hotkeys are available:

Action	Shortcut
Esc / Alt+F4	Closes the Search dialog.
Alt + A	Brings the cursor to the "Search string" field.
Alt + I	Checks/unchecks the "Case Sensitive" checkbox.
Alt + T	Checks "Type" in the "Search For" box.
Alt +M	Checks "Method" in the "Search For" box.
Alt +F	Checks "Field" in the "Search For" field.
Alt +L	Checks "Declarations" in the "Limit To" field.
Alt +R	Checks "References" in the "Limit To" field.
Alt +O	Checks "All occurrences" in the "Limit To" field.
Alt +Y	Checks/unchecks the "Search the Interpreter system libraries"field.
Alt + W	Selects "Workspace" in the "Scope" field.
Alt + D	Selects "Selected resources" in the "Scope" field.
Alt + J	Selects "Enclosing projects" in the "Scope" field.
Alt + K	Selects "Working set" in the "Scope" field.
Alt + H	Opens the "Select Working Sets" dialog.
Alt + S	Performs the search and shows the results in the Search view.
Alt +Z	Opens the "Search Page Selection" dialog and allows you to customize the search page.

Opening PHP Elements

This procedure describes how to use the Open PHP Element function to navigate to a PHP element (Class, Function or Constant) in an open project.

To open a PHP Element:

1. From the Menu Bar, go to **Navigate | Open PHP Element** -or- click the Open PHP Element icon on the Toolbar .

   The "Open PHP Element" dialog will open.

2. Enter the first few characters of the element which you want to open.

   Resources that begin with those letters will appear in the 'Matching Resources' pane, listed alphabetically.
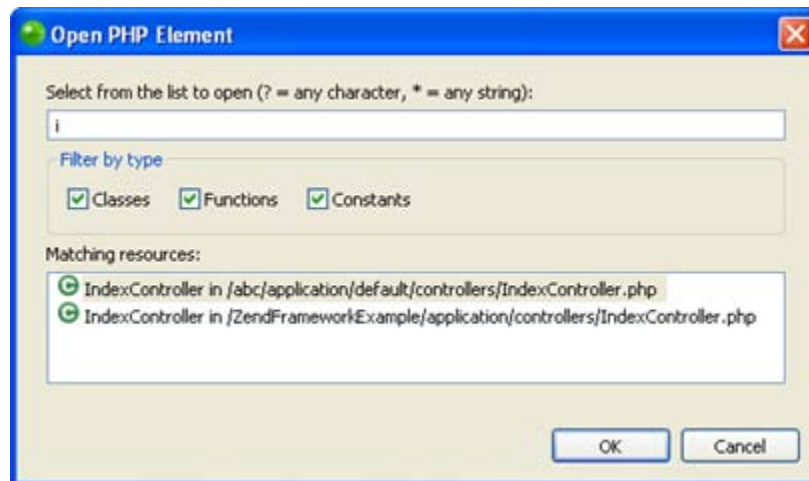
3. You can filter by element type (class, function or constant) by marking/unmarking the relevant checkboxes.

4. Select the required element and click **OK**.

The file containing the element declaration will open in the editor, with the element highlighted.

Opening Types/Methods

You can open any method or type in your workspace using the "Open Type" wizard or the "Open Method" wizard.

To open a Type or Method:

1. From the Menu Bar, go to **Navigate | Open Type or Open Method**

   -Or- press **Ctrl+Shift+T** (for a type) or **Ctrl+Shift+M** (for a method).

   The "Open Type/Method" dialog will appear. If a type/method was previously selected in the editor or outline views, it will be displayed.
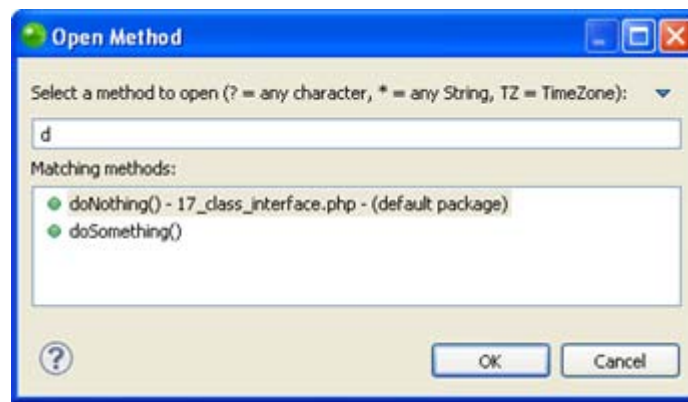
2. Begin typing the string of the required type/method to filter the results.

   You may use wild cards or CamelCase notation (e.g. DTB for DateTextBox).

3. Select the required type/method from the list and click **OK**.

   You may restrict the results displayed in the Open Type list to a particular Working Set through the wizard's menu (accessed by clicking the arrow in the top-right corner).

An editor will open on the selected type/method.

Note:

The "Open Type" dialog maintains a history of recently opened types. These are shown when the dialog is opened and stay above a separator line when you start to type a filter expression.

Using Smart Goto Source

This procedure describes how to use the Smart Goto Source function in order to easily navigate to an element's declaration.

To use the Smart Goto Source function:

1. Hover over the element whose source declaration you want to navigate to.

   A tooltip will be displayed showing the element's original location.
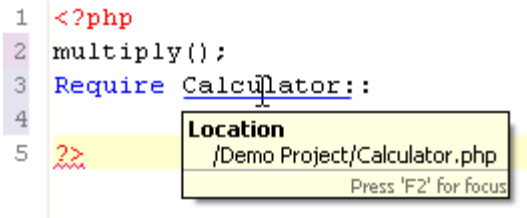
2. Hold down the **Ctrl** key and move the cursor until the element is underlined.

3. Click the element.

You will be automatically taken to the element's source code. If the declaration is in a different file, this file will be opened.

Note:

Smart Goto Source will also be available for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information.

Viewing Type Hierarchies

Type hierarchies can be viewed in either a Quick Type Hierarchy view or in the Type Hierarchy view.

Viewing Types in the Quick Type Hierarchy View

To view a type in a Quick Type Hierarchy view:

1. Select the type name in the PHP editor, or PHP Explorer or outline views.

2. Press **Ctrl+T**.

   -Or- from the Menu Bar go to **Navigate | Quick Type Hierarchy**.

The Quick Type Hierarchy view will be displayed in the editor with the selected type.
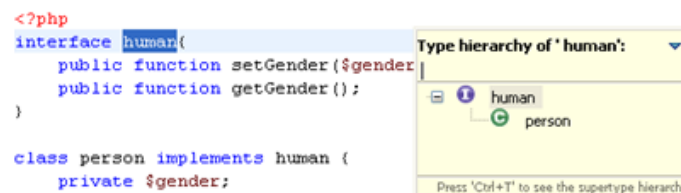
Note:

Pressing **Ctrl+T** while the quick type hierarchy view is shown will toggle between supertype hierarchy and subtype hierarchy.

Viewing Types in the Type Hierarchy View

Types can be viewed in the Type Hierarchy view by searching for them using the "Open Type in Hierarchy" dialog or by directly selecting the element in the editor or PHP Explorer.

To view a type in the Type Hierarchy view:

Through the "Open Type in Hierarchy" wizard:

1. Press **Ctrl+Shift+H** -or- from the Menu Bar go to **Navigate | Open Type in Hierarchy**.

   The "Open Type in Hierarchy" dialog is displayed.
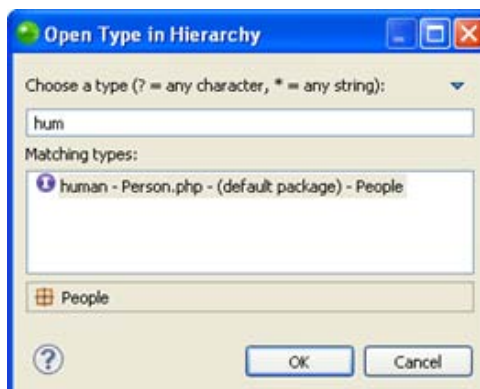
2. If a type was previously selected in the editor or outline views, it will be entered in the type field.

If a type was not selected, begin typing the string of the required type/method to filter the results.

You may use wild cards or CamelCase notation (e.g. DTB for DateTextBox).

3. Select the required type and click **OK**.

Directly from the editor or PHP Explorer:

1. Select a type in the editor or PHP Explorer.

2. Press **F4**.

Note:
The action will not be activated if the selection is not a resolvable element (i.e. if the selection is not a class name, interface name or class method, constant or field).

The type will be displayed in the Type Hierarchy view.

Creating PHP Working Sets

PHP Working Sets can be created from a variety of locations where working sets can be selected. This procedure describes how to create PHP Working Sets from the Window Menu in the PHP Perspective.

To create a PHP Working Set:

1. In the PHP Explorer view click the View Menu arrow ▽.

   The PHP Explorer menu options will open. Select 'Select Working Set...'.

   The "Select Working Set" dialog will open.
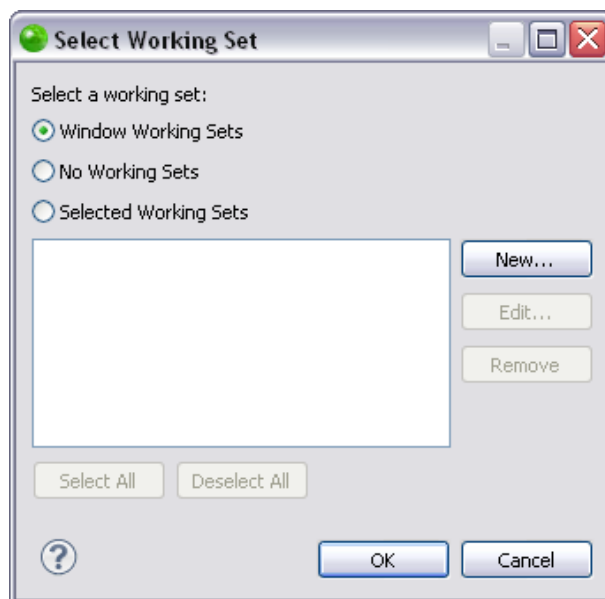
2. Click **New**.

   The "New Working Set" dialog will open.

3. Select PHP and click **Next**.

   The "New PHP Working Set" dialog will appear.

4. Enter a name for the Working Set.

5. Mark the checkbox next to the projects/files/folders to be included in the Working Set.
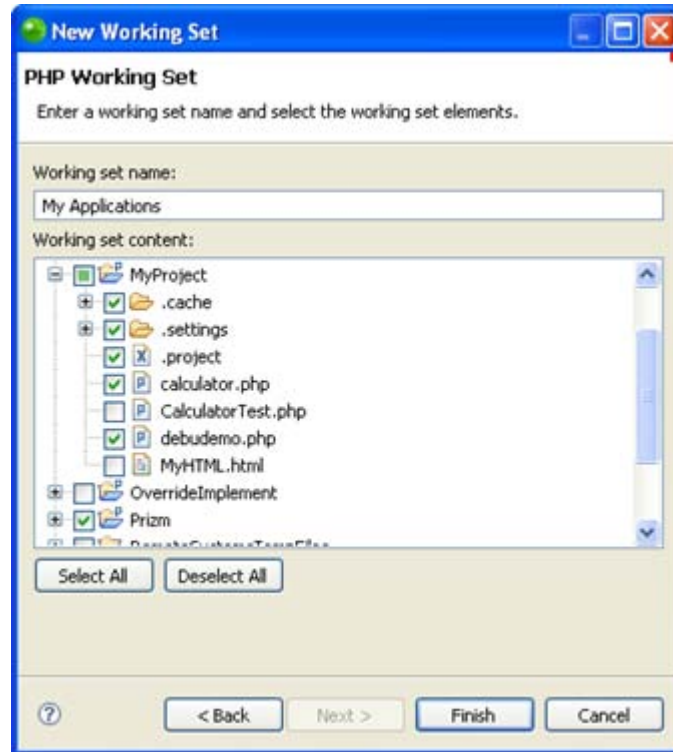
6. Click **Finish**.

Your new PHP Working Set will be added to the "Select Working Set" dialog and will be available for selection for Working Sets actions.

Using Mark Occurrences

The Mark Occurrences feature allows you to see where a variable, method or type is referenced within the active PHP file.

To use Mark Occurrences:

1. Enable the Mark Occurrences feature by clicking the Mark Occurrences icon on the toolbar

   -or- pressing **Alt+Shift+O**.

   When the Mark Occurrences feature is enabled, the icon will be highlighted.

2. Place your cursor on the required element in your file.
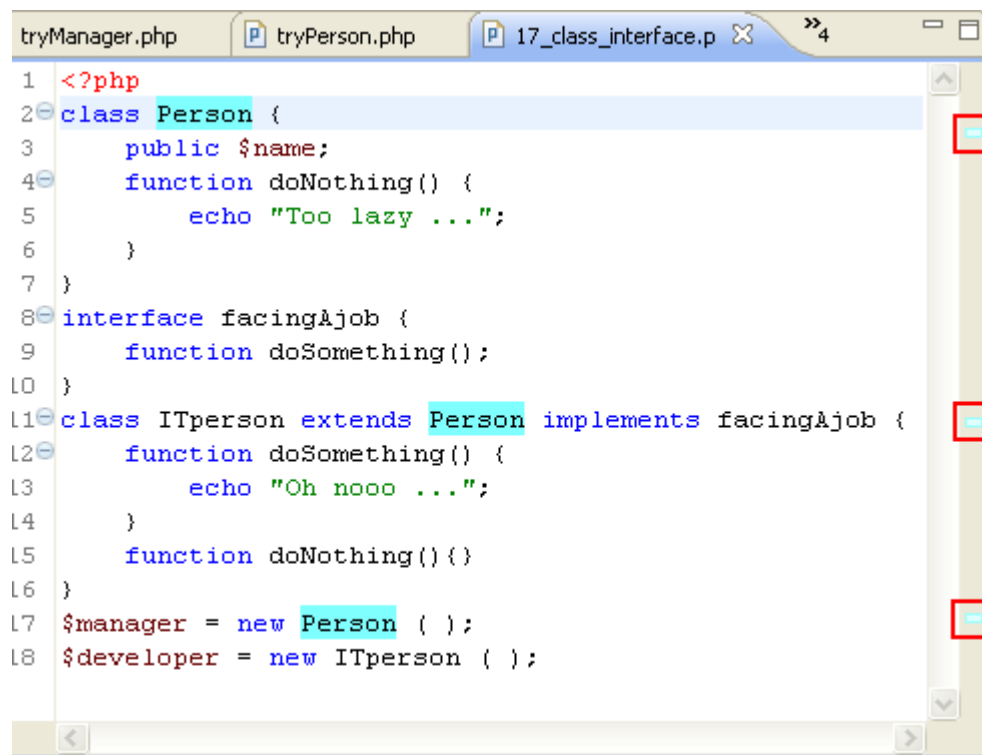
All instances where the element is referenced within the file will be highlighted and annotations will be displayed in the annotation bar.

Note:

See Mark Occurrences Preferences for information on configuring your Mark Occurrences settings.

Finding and Replacing

This procedure describes how to do a Find and Replace for a string within a file.

To find and replace a string:

1. From within the file, press **Crtl+F** or from the Menu Bar go to **Edit | Find/Replace**.
2. A "Find/Replace" dialog will appear.

3. Enter the required string to find.
4. Enter the required string to replace it.
5. If necessary, configure the settings under the direction, scope and options category.
6. Click **Replace**.

The found string will be replaced by the new string.

Applying Quick Fixes

Quick Fixes are suggestions for corrections to badly written code provided by the Semantic Analysis mechanism which can be easily applied to code to fix code errors.

To apply a quick fix suggestion:

1. Open the file containing the problematic code.

   The line(s) containing code to which a quick fix can be applied will be indicated by a lightbulb icon in the vertical marker bar to the left of the editor window.

   See Real Time Error Detection for more information on how errors are indicated.

2. Place the cursor on the problematic code section and press **Ctrl + 1** -or- click the lightbulb icon in the marker bar.

   The Quick Fix list suggests options for fixing the code.

```
1  <?php
2  $hello = "Hello, World";
3
4  echo "$hello";
5
          Change to 'hello'
```
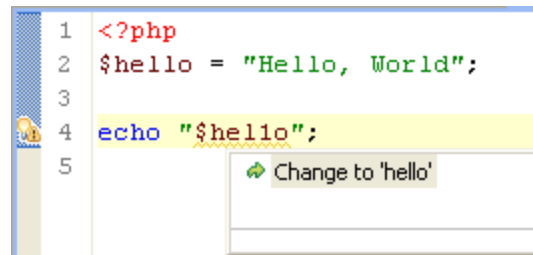
3. Select the required fix from the list by clicking on it or by using the arrow keys and pressing **Enter**.

4. The selected fix is applied and the code is changed accordingly.

Adding Comments

Zend Studio allows you to quickly and easily comment and uncomment code by selecting a line or a block of text and tagging it as a comment.

Comments can be added to single lines of code (**Ctrl + /**) or blocks of code (**Ctrl + Shift + /**).

In addition, special PHPDocBlock comments can also be added. See Adding PHP DocBlock Comments for more information.

The following procedures describe how to comment and uncomment lines and blocks of code.

Commenting a Line

To comment a line:

1. Place the cursor anywhere on the required line of code.

2. Press **Ctrl + /**

   Two slashes "//" will be added to the front of the line, causing it to be recognized as a comment.

Commenting More than One Line

To comment more than one line:

1. Select all the lines that you would like to be commented.

2. Press **Ctrl + /**

   Two slashes "//" will be added to the front of each line, causing them to be recognized as a comment.

Uncommenting a line/lines

To uncomment a line/lines:

1. Select the required line(s).

2. Press **Ctrl + /**

   The commenting formatting will be removed from the code.

Commenting a Block

To comment a block:

1. Select the required block of code.

2. Press **Ctrl + Shift + /**

   The beginning (/*) and ending (*/) characters will be added in the appropriate places in order to mark the selected block as a comment.

```
1  <?php
2  /*This
3  is a
4  block comment*/
5
```

Adding PHP DocBlock Comments

This procedure describes how to add PHPDoc Comments to PHP elements.

To create a PHPDoc Comment:

In the line above the code for the PHP element, enter the DocBlock characters /** and press **Enter**.

-Or- Right-click the relevant element in the Outline View and select **Source | Add PHPDoc**.

A PHPDoc Comment will be created with several parameters to be edited with the relevant information.

The default code comments generated for different elements can be configured through the Code Templates Preferences page.

Using Local History

The following actions can be carried out using the Local History functionality:

- [Comparing Files](#)
- [Replacing Files](#)
- [Restoring Deleted Files](#)

Comparing Files

This procedure describes how to compare a current version of a file with one from the Local History.

To compare the current file state and a previous one:

1. Right-click the file in PHP Explorer view and select **Compare with | Local history**.

   The History View will be displayed with a list of all the previous saved versions of the file.
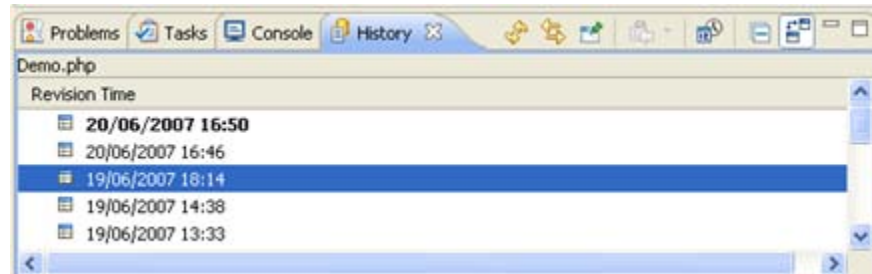
2. Double-click the version that you would like to view.

   The "Text Compare" dialog will be displayed, with the current file version displayed in the left pane and the previous version displayed in the right pane.

   Any changes that have been made between the current version and the previous version will be highlighted.
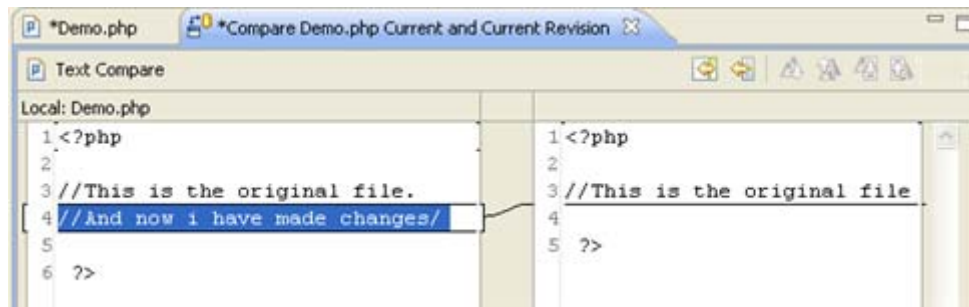
3. Click the next / previous difference buttons to scroll between the differences.

Replacing Files

This procedure describes how to replace a current file with a previous version from the Local History.

To revert a file to one of its previous states:

1. Right-click the file in PHP Explorer view and select **Replace With | Local History**.
2. The "Compare" dialog will be displayed with a list of the previous versions of the file, according to when they were last saved.
3. Double-click a previous version to view it in the Text Compare pane.
4. Once you have found the required version, click **Replace**.

The current working file will be replaced by the chosen local history file.

Note:

To replace a file with the last saved file, right-click anywhere in the editor and select **Replace With | Previous From Local History**.

Restoring Deleted Files

This procedure describes how to restore a file that has been deleted.

To restore a deleted file:

1. In PHP Explorer view, right-click the project which previously contained the file and select **Restore From Local History**.

   A "Restore from Local History" dialog will be displayed, containing a list of all files that have been deleted from the folder.

2. Select the required file to view its revisions.

3. Select the required revision and click **Restore**.

The file will be restored into the selected folder.

Using CVS

CVS is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions.

The following tasks describe some of the actions that can be done using Zend Studio's CVS functionality:

- Configuring a CVS connection
- Importing a project from CVS
- Accessing an Existing CVS Checkout
- Uploading projects to CVS

See Working in a Team Environment with CVS in the Workbench User Guide for more information on CVS.

> **Note:**
> Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Configuring a CVS Connection

This procedure describes how to configure a connection to a CVS repository:

Before you can add projects to or export projects from CVS, you must define your CVS repository settings.

Prerequisite: To access a repository, make sure that a CVS server is already configured.

To add a new CVS repository:

1. Open the CVS perspective by going to **Window | Open Perspective | Other | CVS Repository Exploring**.

2. In the CVS Repositories view, click the Add CVS Repository button [icon] on the view's toolbar -or- right-click within the view and select **New | Repository Location**.
   The "Add CVS Repository" dialog will open.

3. Enter the information required to identify and connect to the repository location:
   - Host - The host address (e.g. mycomputer.com).
   - Repository path - The path to the repository on the host (e.g /usr/local/cvsroot)
   - User - The user name with which you connect to the repository.
   - Password - The password for the user name.

- Connection Type - The authentication protocol for the CVS server.

  There are four connection methods:

i. pserver - a CVS specific connection method.

ii. extssh - an SSH 2.0 client.

iii. pserverssh2 - provides a pserver connection over ssh2.

iv. ext - the CVS ext connection method which uses an external tool such as SSH to connect to the repository.

- If the host uses a custom port, enable Use Port and enter the port number.

4. Click **Finish** to create your connection.

Your CVS repository will now be added to the CVS Repository view.

See [Creating a CVS Repository Location](#) in the Workbench User Guide for more information.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help	Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Importing Projects from CVS

Once projects are placed on the CVS repository, they can be checked out (imported) by anyone with access to that repository. CVS repository connections allow you to import projects from your repository to your workspace, which you can make and upload changes to.

This procedure describes how to import (check out) projects from a CVS repository location to your desktop.

Note:
If you have resources on your file system which are already linked to CVS, you can access them in Zend Studio by following the instructions under [Accessing an Existing CVS Checkout](#).

To import a project from an CVS repository:

1. Go to **File | Import | CVS | Projects from CVS**.
2. Click **Next**.
3. Select your repository.

   If you have not yet created a repository, enter the information required to identify and connect to the repository location:

   - Host - The host address (e.g. mycomputer.com).
   - Repository path - The path to the repository on the host (e.g /usr/local/cvsroot)
   - User - The user name with which you connect to the repository.
   - Password - The password for the user name.
   - Connection Type - The authentication protocol for the CVS server.
   - If the host uses a custom port, enable Use Port and enter the port number.
4. Click **Next**.
5. A "Select Resource" dialog will appear. Expand the nodes until you see the required project.
6. Select your project and click **Finish**.

   A "Check Out As" dialog will appear.
7. Select one of the following options:

   - Check out as project configured using the "New Project" Wizard - Imports the project as a new PHP project into your workbench with the project's existing name.
   - Find projects in the children of selected resource - Imports all folders within the project as separate projects.
   - Check out as folder into existing project - Imports the project as a folder into an existing project in your workbench.
   - Check out as project with the name specified - Imports the project as a new project into your workbench with a new name. Specify the new name in the box.

Zend Studio 9 User Guide

Note:

To enable all Zend Studio's PHP functionality for the imported projects, select the 'Check out as a project configured using the New Project Wizard' option and ensure you create the new project as a PHP project.

8. Click **Finish**.

The project will now be imported into your workspace.

Note that the project will have a CVS repository icon 📁 in your PHP explorer view.

Once you have imported a project from CVS into your workspace, you can now add files, edit existing files and commit your changes to the CVS repository.

Note:

Projects can also be checked out from CVS through the CVS Repository Exploring perspective, accessible from **Open Perspective | Other | CVS Repository Exploring**. Simply right-click the project in CVS Repositories view and select Check Out or Find/Check Out As..

See '[Checking out a project from a CVS repository](#)' in the Workbench User Guide for more information.

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Accessing an Existing CVS Checkout

This procedure shows you how to access projects in Zend Studio that you have previously checked out from CVS. This means you do not have to check out resources again to a new location in order to have access to Zend Studio's functionality.

To access previously checked out projects:

1. Launch the New PHP Project Wizard by going to **File | New | PHP Project** -or- right-clicking in PHP Explorer view and selecting **New | PHP Project**.
   The "New PHP Project" wizard will open.

2. In the Project contents category, unmark the Use default checkbox and browse to the location of your checked out CVS resources.
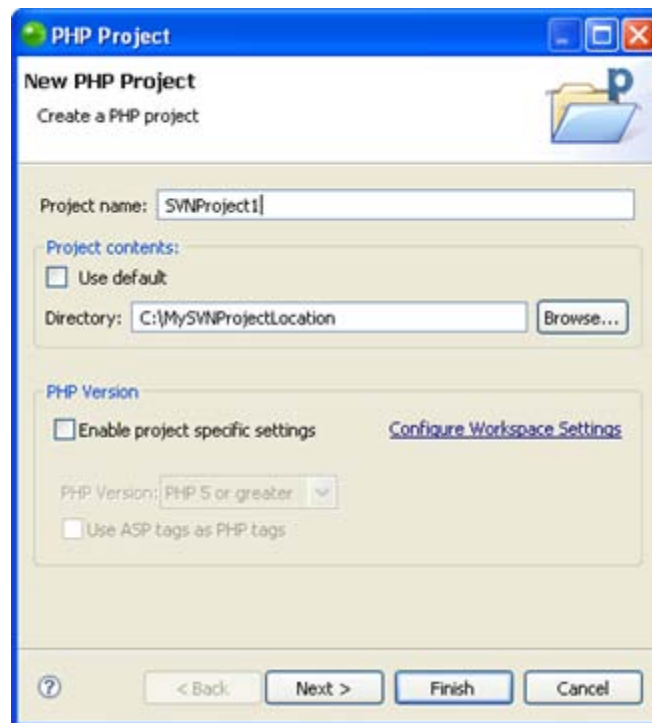
3. Click **Finish**.
   A new PHP Project will be created with the contents of the project you had checked out.

4. In PHP Explorer View, right-click your project and select **Team | Share Project**.
   A "Share Project" dialog will open.

5. From the repository list, select CVS and click **Next**.

6. If you have not yet configured a CVS repository, you will need to enter your CVS repository details. See Configuring a CVS Connection for more information.
   If you have already configured a CVS connection, mark the 'Use existing repository location' option and select your repository from the list.

7. Click **Finish**.

8. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.

   Enter your password and click **Next**.(Mark the Save Password checkbox to ensure that this screen does not reappear.)

   A "Commit" dialog will open.

9. Enter a comment if required and click **OK**.

Your project will be uploaded to the CVS repository.

Your project will have a repository icon next to it in PHP Explorer view, indicating that it is linked to an CVS repository.

Your can now perform all CVS functions (commit, update etc.) on this project.

Uploading Projects to CVS

Using CVS, you can upload projects and files which other team members can work on.

This procedure describes how to upload a project to your CVS repository location.

Prerequisites: You should have a CVS repository configured before you follow this procedure. See Configuring a CVS Connection for instructions.

To upload a project to an CVS repository:

1. In PHP Explorer View, right-click your project and select **Team | Share Project**.
   A "Share Project" dialog will open.

2. From the repository list, select CVS and click **Next**.

3. Select 'Use existing repository location', and select your repository from the list.
   If you have not yet configured a CVS repository, enter the information required to connect to your CVS Repository. See Configuring a CVS Connection for more information.

4. Click **Finish**.

5. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.
   Re-enter your password and click **Next**.(Mark the Save Password checkbox to ensure that this screen does not reappear.)
   A "Commit" dialog will open.

6. Enter a comment if required and click **OK**.

Your project will be uploaded to the CVS repository.

Your project will now have a repository icon next to it in PHP Explorer View, indicating that it is linked to a CVS repository.

Once the project has been committed other team members will be able to access and edit it.

Note:
See 'Sharing a new project using CVS' in the Workbench User Guide for more information.

Using SVN

SVN, or Subversion, is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions.

The following tasks describe some of the actions that can be done using Zend Studio's SVN functionality:

- [Configuring an SVN connection](#)
- [Importing a project from SVN](#)
- [Accessing an Existing SVN Checkout](#)
- [Uploading projects to SVN](#)

See the [Subversive User Guide](#) for more information on SVN.

> **Note:**
> Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Configuring an SVN Connection

This procedure describes how to configure a connection to an SVN repository.

Before you can add projects to or export projects from SVN, you must define your SVN repository settings.

Prerequisite: To access a repository, make sure that an SVN server is already configured.

To add a new SVN repository:

1. Open the SVN perspective by going to **Window | Open Perspective | Other | SVN Repository Exploring**.

2. In the SVN Repositories view, click the Add SVN Repository button ![icon] on the view's toolbar - or- right-click within the SVN view and select **New | Repository Location**.

   The "Add SVN Repository" dialog will open.

3. Enter the information required to identify and connect to the repository location:
   - URL - The URL on which your repository is located.
   - Label - Select whether to use the URL as the repository's name or to enter a new name.
   - Authentication - The user name and password you use to connect to SVN.
     Mark the Save password checkbox so that the password will be automatically inserted in the future.
4. Click **Finish**.

Your SVN repository will now be added to the SVN Repository view.

See the [Subversive User Guide](#) for more information on SVN.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Importing Projects from SVN

Once projects are placed on the SVN repository, they can be checked out (imported) by anyone with access to that repository. SVN repository connections allow you to import projects from your repository to your workspace, which you can make and upload changes to.

This procedure describes how to import (check out) projects from an SVN repository location to your desktop.

Note:
If you have resources on your file system which are already linked to SVN, you can access them in Zend Studio by following the instructions under Accessing an Existing SVN Checkout.

To import a project from an SVN repository:

1. Go to **File Import | SVN | Projects from SVN**.
2. Click **Next**.
3. Select your repository.

   If you have not yet created a repository, enter the information required to identify and connect to the repository location:

   - URL - The URL on which your repository is located.
   - Label - Select whether to use the URL as the repository's name or to enter a new name.
   - Authentication - The user name and password you use to connect to SVN.
     Mark the Save password checkbox so that the password will be automatically inserted in the future.

4. Click **Next**.
5. A "Select Resource" dialog will appear. Expand the nodes until you see the required project.
6. Select your project and click **Finish**.

   A "Check Out As" dialog will appear.

7. Select one of the following options:

- Check out as project configured using the "New Project" wizard - Imports the project as a new PHP project into your workbench with the project's existing name.
- Find projects in the children of selected resource - Imports all folders within the project as separate projects.
- Check out as folder into existing project - Imports the project as a folder into an existing project in your workbench.
- Check out as project with the name specified - Imports the project as a new project into your workbench with a new name. Specify the new name in the box.

Note:

To enable all Zend Studio's PHP functionality for the imported projects, select the 'Check out as a project configured using the New Project Wizard' option and ensure you create the new project as a PHP project.

8. Click **Finish**.

The project will now be imported into your workspace.

Note that the project will have an SVN repository icon 🗄️ in your PHP explorer view.

Once you have imported a project from SVN into your workspace, you can now add files, edit existing files and commit your changes to the SVN repository.

Note:

Projects can also be checked out from SVN through the SVN Repository Exploring perspective, accessible from **Window | Open Perspective | Other | SVN Repository Exploring**. Simply right-click the project in SVN Repositories view and select **Check Out** or **Find | Check Out As..**

See the Subversive User Guide for more information on SVN.

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Accessing an Existing SVN Checkout

This procedure shows you how to access projects in Zend Studio that you have previously checked out from SVN. This means you do not have to check out resources again to a new location in order to have access to Zend Studio's functionality.

To access previously checked out projects:

1. Launch the New PHP Project Wizard by going to **File | New | PHP Project** -or- right-clicking in PHP Explorer view and selecting **New | PHP Project**.
   The "New PHP Project" wizard will open.

2. In the Project contents category, unmark the Use default checkbox and browse to the location of your checked out SVN resources.

3. Click **Finish**.
   A new PHP Project will be created with the contents of the project you had checked out.

4. In PHP Explorer View, right-click your project and select **Team | Share Project**.
   A Share Project dialog will open.

5. From the repository list, select SVN and click **Next**.

6. If you have not yet configured an SVN repository, you will need to enter your SVN repository details. See Configuring an SVN Connection for more information.
   If you have already configured an SVN connection, mark the 'Use existing repository location' option and select your repository from the list.

7. Click **Finish**.

8. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.

   Enter your password and click **Next**.(Mark the Save Password checkbox to ensure that this screen does not reappear.)

   A Commit dialog will open.

9. Enter a comment if required and click **OK**.

Your project will be uploaded to the SVN repository.

Your project will have a repository icon next to it in PHP Explorer view, indicating that it is linked to an SVN repository.

Your can now perform all SVN functions (commit, update etc.) on this project.

See the Subversive User Guide for more information on SVN.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Uploading Projects to SVN

Using SVN, you can upload projects and files for other team members to access and edit.

This procedure describes how to upload a project to your SVN repository location.

Prerequisites: You should have an SVN repository configured before you follow this procedure. See Configuring an SVN Connection for instructions.

To upload a project to an SVN repository:

1. In PHP Explorer View, right-click your project and select **Team | Share Project**.
   A Share Project dialog will open.

2. From the repository list, select **SVN** and click **Next**.

3. If you have not yet configured an SVN repository, you will need to enter your SVN repository details. See Configuring an SVN Connection for more information.
   If you have already configured an SVN connection, mark the 'Use existing repository location' option and select your repository from the list.

4. Click **Finish**.

5. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.
   Re-enter your password and click **Next**.(Mark the Save Password checkbox to ensure that this screen does not reappear.)
   A Commit dialog will open.

6. Enter a comment if required and click **OK**.

Your project will be uploaded to the SVN repository.

Your project will now have a repository icon next to it in PHP Explorer View, indicating that it is linked to an SVN repository.

Once the project has been committed other team members will be able to access and edit it.

See the Subversive User Guide for more information on SVN.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help	Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Developing with Zend Framework

When developing using the Zend Framework, you should first create a Zend Framework project, which has the Zend Framework library added to it's include path.

You should then use the New Zend Framework item wizard to add Zend Framework elements to your project.

- For information on how to create Zend Framework Projects, see Creating Zend Framework Projects.
- To trial using Zend Framework, you can create a Zend Framework Example Project.
- For information on creating Zend Modules, Zend Controllers, Zend Tables, Zend Views, Zend View Helpers and Zend Action Helpers , see Creating Zend Framework Elements.
- For information on how to search for information on Zend Framework, see Searching the Zend Framework Site.

Once you have created your Zend Framework application, you can run or debug it on your server by following the instructions in Running and Debugging Zend Framework Projects.

Creating Zend Framework Projects

Zend Framework Projects are PHP projects which contain Zend Framework's libraries in their Include Path and which are organized into Framework's Controller-Model- View format (if selected). Zend Framework 1.8 projects are based on the Zend Tool, ensuring your projects comply with the latest Zend Framework standards and allow for Rapid Application Development .

Visit http://framework.zend.com for more on Zend Framework or http://framework.zend.com/manual/en for the Zend Framework Reference manual.

This procedure demonstrates how to create a new Zend Framework Project.

Important Note:

You must have Zend Framework version 1.8 installed on your machine to be able to create Zend Framework projects.

Zend Framework comes bundled with Zend Server or can be downloaded from the Zend Framework site at http://framework.zend.com/download/current .

To create a new Zend Framework Project:

1. Go to **File | New | Zend Framework Project** -or- right-click in PHP Explorer view and select **File | New | Zend Framework Project**.

   The New Zend Framework Project dialog will open.
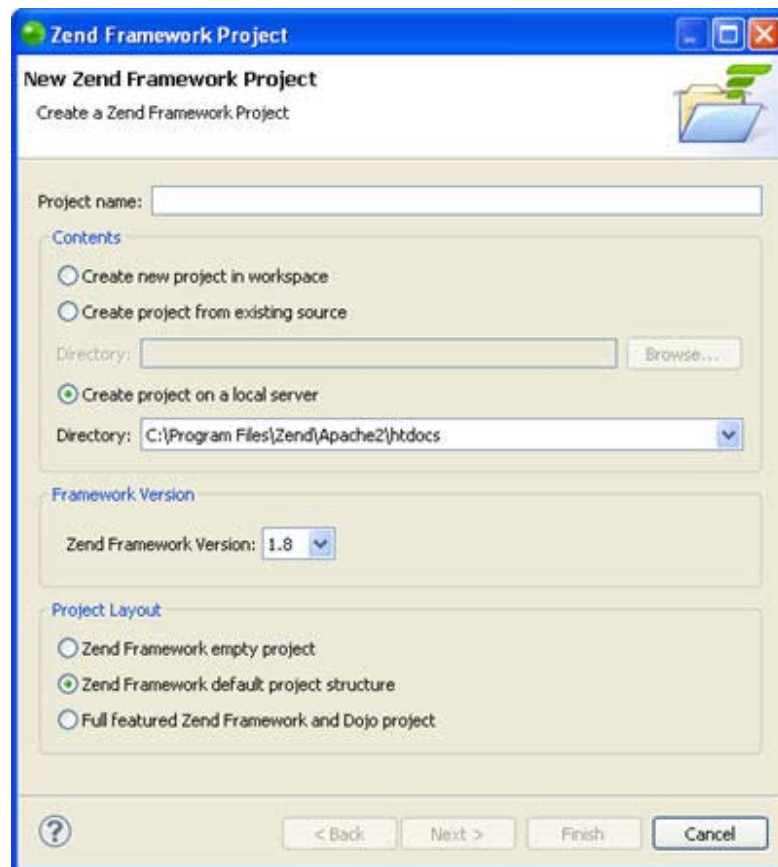
2. Enter the project name.

3. In the Framework Version drop-down list, select the Zend Framework version to use for your project. Zend Studio supports Zend Framework version **1.9**. This will include the required Zend Framework library in your project, and give you access to the relevant content assist options.

4. Select the Project Layout from the following options:
    - Zend Framework empty project - Creates a regular PHP project structure
    - Zend Framework default project structure - Creates a project organized according to Zend Tool conventions.
    - Full featured Zend Framework and Dojo project - Select this option if you are going to be developing using the JavaScript Dojo framework in this project. This will create a link to the external Dojo toolkit library. See Setting Up and Using Dojo Integration for more information.

5. Click **Finish**.

    Your Zend Framework Project will be created.

    It will be represented by a Zend Framework icon in PHP Explorer view.

6. Click **Yes** when prompted to open the Zend Framework perspective.

    Your Zend Framework project will be displayed in the MVC Outline view.

    > **Note:**
    > To open the MVC Outline view manually, go to **Window | Show View | PHP Tools | MVC Outline**.
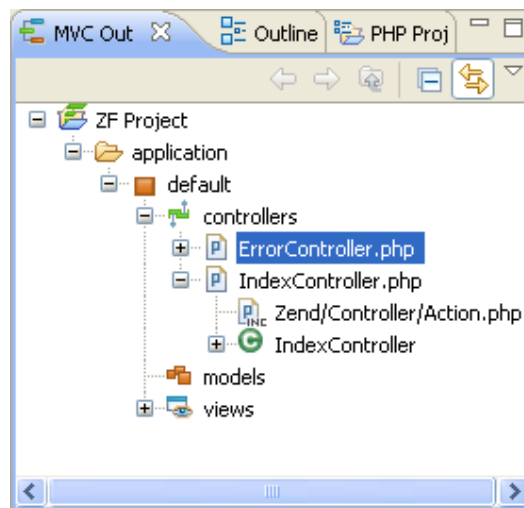
The MVC view provides an outline of all controller, model and view classes, files, variables and related functions.

You can now start to add your own Model, View and Controller files. See Creating Zend Framework Elements for more information.

Running and Debugging Zend Framework Projects

In order to run and debug Zend Framework projects on your server, you will need to set up your server to be able to execute Zend Framework applications.

Setting Up Your Server Environment

Prerequisites:

- Apache 2.2 or later*
- PHP 5*
- Zend Framework*
  Note: The Zend Framework version installed on your server should match the Zend Framework version in which you developed your application.

 * These components can be installed together by installing [Zend Server](#).

The following instructions will guide you through the process of setting up your Zend Server in order to be able to run Zend Framework-based applications.

Note:

Zend Server is the recommended environment but Zend Framework projects can be run on other servers. Different configuration information might apply.

Setting up Your Server to Run/Debug Zend Framework Projects

To set up your server in order to run/debug Zend Framework Projects:

1. If you do not have a server running a PHP distribution already installed, you can install [Zend Server](#) (downloadable from http://www.zend.com/en/products/server).
   Ensure you install with the following settings:
   i.   If you do not have a Web server, install the Apache bundled with Zend Server.
   ii. If you do not have the Zend Framework files already installed on your machine, install Zend Server in 'Complete' mode, or ensure that Zend Framework is selected in the list of additional components in 'Custom' mode.
2. If you choose not to install Zend Server make sure you install the following:
   i.   PHP 5
   ii. Apache 2
   iii. Zend Framework files (downloadable from http://framework.zend.com/download).

3. Make sure your php.ini settings are properly configured as follows: (These can be configured through Zend Server or by manually configuring your php.ini file).

<table>
<tr><td>Note:</td></tr>
<tr><td>If you are using Zend Server , these settings should be configured by default.</td></tr>
</table>

i. Your include path is pointing to your Zend Framework library (e.g. C:\Program Files\Zend\ZendServer\share\ZendFramework\library).

This is configured in the 'include_path' directive.

In Zend Server, this directive can be found under the 'Paths and Directories' category in **Server Setup | Directives**.

ii. The session settings are configured to save path points to a temporary directory writable by the Web server (e.g."C:/temp" on Windows or "/tmp" on UNIX / Linux.)

This is configured in the 'session.save_path' directive.

In Zend Server, this directive can be found under the 'session' category in **Server Setup | Directives**.

iii. PHP Short Tags are enabled.

This is configured in the short_open_tag directive.

In Zend Server, this directive is on by default. It can be found under the 'Language Options' category in **Server Setup | Directives**.

iv. If you want to display errors, set your display error setting to on.

This is configured in the display_errors directive.

In Zend Server, this can found under the 'Error Handling and Logging' category in **Server Setup | Directives**.

4. Restart your Web server after applying changes.

Once you copy your Zend Framework project files to the server you will be able to run and debug the project as you would a normal PHP project.

Configuring Zend Server to Run a Zend Framework Application

To configure Zend Server to run a Zend Framework application:

1. Define a virtual host on Zend Server that will point to the new project's public directory:

i. Find the full path to your project's public directory.

This is listed in the location field in the project's properties. To see it, in PHP Explorer view right-click the public directory and select **Properties | Resources**.

ii. Open your Apache configuration file (in most cases it will be httpd.conf and located in your Apache installation directory).

See [Where is my Apache configuration file](#) for more information.

iii.   Go to the end of the file and add the following code:

```
Listen 10089
<VirtualHost *:10089>
    DocumentRoot "DOCUMENT_ROOT"
    <Directory "DOCUMENT_ROOT">
        Order allow,deny
        Allow from all
    AllowOverride all
</Directory>
</VirtualHost>
```

2. Replace "DOCUMENT_ROOT" with the full path to the public directory, enclosed in double quotes ("DOCUMENT_ROOT").

   Replace the port number with a unique port number dedicated to this Virtual Host. The port number (10089) has to be the same value for "Listen" and "VirtualHost".

3. Zend Framework's MVC implementation makes use of the Front Controller pattern. You must therefore rewrite all incoming requests (except those for static resources, which your application need not handle) to a single script that will initialize the FrontController and route the request. If you're using mod_rewrite for the Apache web server, create the file /public/.htaccess with the following contents:

```
# public/.htaccess
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ /index.php [NC,L]
```

Note:

Some web servers may ignore .htaccess files unless otherwise configured. Make sure that your web server is configured to read the .htaccess file in your public directory.

4. Restart your Web server from the command line (windows user can use the Apache Monitor tool).

Once you copy your Zend Framework project files to the server you will be able to run and debug the project as you would a normal PHP project.

Where is My Apache Configuration File?

Apache uses a main configuration file for all its settings, typically this file is called httpd.conf or apache2.conf. The location of this file varies depending on your installation:

- **Windows:**
  - <install_dir>\Apache2.2\conf\httpd.conf
  - If you changed the location of your Zend Server installation, your document root will be located at <installation_directory>\ Apache2.2\conf\httod.conf, where <installation_directory> is the location of the directory in which Zend Server is installed.
- **Linux**:
- If you installed Zend Server from a repository (DEB or RPM packages), the location of your configuration file is defined by your distribution's Apache packages, and will vary depending on your distribution and configuration.
  - Common locations include:
    - Debian / Ubuntu - /etc/apache2/apache2.conf
    - Fedora Core / RHEL / CentOS - /etc/httpd/httpd.conf
  - If you installed Zend Server using the generic Tarball package - /usr/local/zend/apache2/conf/httpd.conf.
  - If you changed the location of your Zend Server installation, your document root will be located at <installation_directory>/ apache2/conf/httpd.conf, where <installation_directory> is the location of the directory in which Zend Server is installed.

Creating and Running a Zend Framework Example Project

The purpose of the Zend Framework Example Project is to demonstrate the capabilities and best practices of working with the Zend Framework.

Zend Framework is a high quality open source framework for developing Web Applications and Web Services with PHP. The Zend Framework Example Project is a small demo application that shows how Zend Framework can be leveraged in order to streamline writing code, by decreasing development time (through using prewritten modules) and demonstrating how to implement best practices in organizing your project.

Once the Example Project has been created, you will be able to navigate through the tree in PHP Explorer and see the different components of the project.

The Zend Framework Example provides a model called MVC (Model-View-Controller) for programming using the Framework. With this model, applications are divided into three parts to assist in making the development process more efficient.

The components are:

- Controller - includes all code that handles the logic.
- Model - contains data access commands to the raw data.
- View - contains the application's front end (User Interface).

Additional information about programming with the Zend Framework can be found in the official Zend Framework Manual at: http://framework.zend.com/manual

For more information about the MVC method see http://en.wikipedia.org/wiki/Model-view-controller.

Creating the Zend Framework Example Project

The following procedure describes how to create a Zend Framework example project.

To create a Zend Framework Example Project:

1. Go to **File | New | Example | Zend Framework Example Project**.

    The "New Zend Framework Example Project" wizard will open.

2. Enter a name for your example project.

3. You have two options for the contents of your Zend Framework Example Project:

    - Create new project in workspace - Creates the project in your current workspace.
    - Create project at existing location - Creates the project from an existing location outside of your workspace.

    You may choose your location by writing it in the "Directory" text field, or by clicking **Browse.**

4. Click **Finish**

An example project will be created and displayed in the PHP Explorer view.

You can navigate through the application source files to learn more about the source behind the application.

Running the Zend Framework Example Project

The following procedure describes how to run the Zend Framework example project. In order to run your Zend Framework Example Project, you must first create one. For more information see Creating the Zend Framework Example Project.

Note:
You must have an established connection with a server to run the Zend Framework Example project. For more information see Defining Zend Server in Zend Studio.

To run the Zend Framework example project:

1. Expand your Zend Framework Example project in the PHP Explorer View.
   The project's sub-folders will be exposed.
2. Expand the sub-folder called **public** and right click on the sub-file **index.php.** From the right click menu select **Run as... | PHP Web Page.**
   The "Run PHP Web Page" dialog will open.
   If the dialog doesn't open you will need to add this directory to the build path by right clicking on the Public folder and selecting **Build Path | Include**.
   You do not need to make any changes in order to run the example project.
3. Click **OK.**

Your project web page opens.

Creating Zend Framework Elements

Once you have created a Zend Framework Project, you can add Zend Framework elements through the Zend Framework Item wizards.

Elements should be placed within a 'Zend Module', which enables you to organize your MVC files.

When working with Zend Framework files, you should open the Zend Framework perspective by going to **Window | Open Perspective | Zend Framework**.

Zend Framework items can be created through the New Zend Framework Item dialog, accessible by going to **New | Zend Framework Item**.
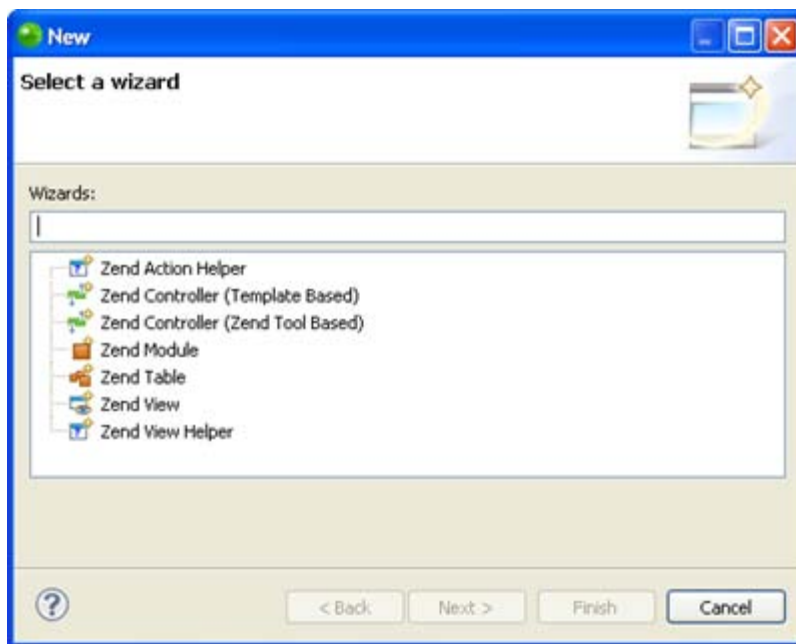
New Zend Framework Item dialog

Through this dialog, you can create the following elements:

- [Zend Action Helper](#)
- [Zend Controller](#)
- [Zend Module](#)
- [Zend Table](#)
- [Zend View](#)
- [Zend View Helper](#)

259

Creating a Zend Module File

Zend Framework MVC files should be created within a Zend Module, which enables you to easily group your MVC files according to your application's components. This will help you organize your Zend Framework Project structure.

One Zend Module, entitled 'default', will be automatically created within the Zend Framework project. Further Zend Modules can be created to group your MVC files according to your needs.

Note:

When working with Zend Framework files, you should open the Zend Framework perspective by going to **Window | Open Perspective | Zend Framework**.

To create a Zend Module:

1. In PHP Explorer view, right-click the Zend Framework project in which you want to create the Zend Module and select **New | Other | Zend Framework | Zend Module**.

    The New Zend Module wizard will open.
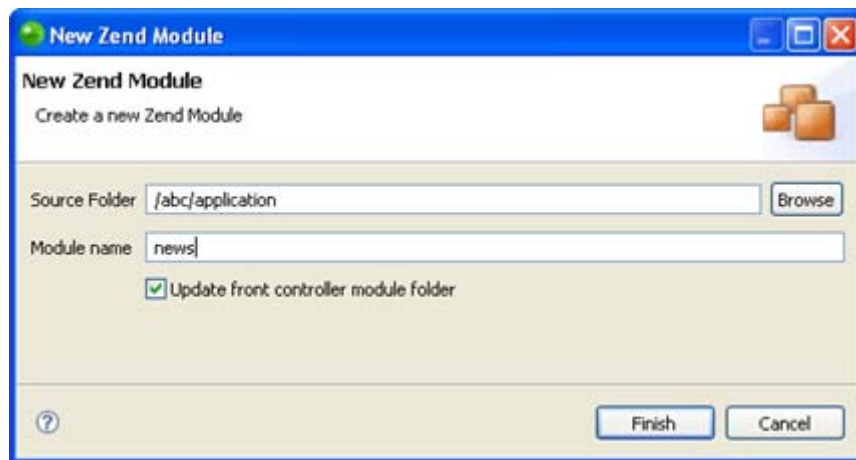
2. The folder in which the Zend Module should be created is the project's 'application' folder. If necessary, click **Browse** to change the Source Folder.
3. In the Module name category, enter a name for the new Zend Module.
4. Leave the 'Update front controller module folder' checkbox marked to add a line in the bootstrap file that specifies the new module.
5. Click **Finish**.

The new Zend Module will be created in your project and will be displayed in the MVC Outline view, represented by an orange square ▪.

The Zend Module will include controller, model and view folders in the relevant file hierarchy.
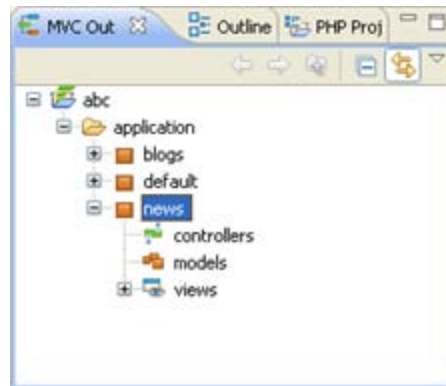
MVC Outline view - containing Zend Modules

Creating a Zend Controller File

This procedure describes how to create a new Zend Controller file.

A Zend Controller is a class for working with the "controller" portion of the model-view-controller pattern. The Zend Controller manages the communication between user actions and your application. See http://framework.zend.com/manual/en/zend.controller.html for more information on the Zend Controller.

Zend Studio offers the option to create Zend Controller files based on the Zend Controller Template (for projects based on Zend Framework version 1.7 and below) and based on the Zend Tool (for projects based on Zend Framework version 1.8 and above).

To create a new Zend Controller file:

1. In PHP Explorer view, right-click the controllers folder in your Zend Framework Project and select **New | Zend Framework Item | Zend Controller**.
   The New Zend Controller Wizard is displayed.
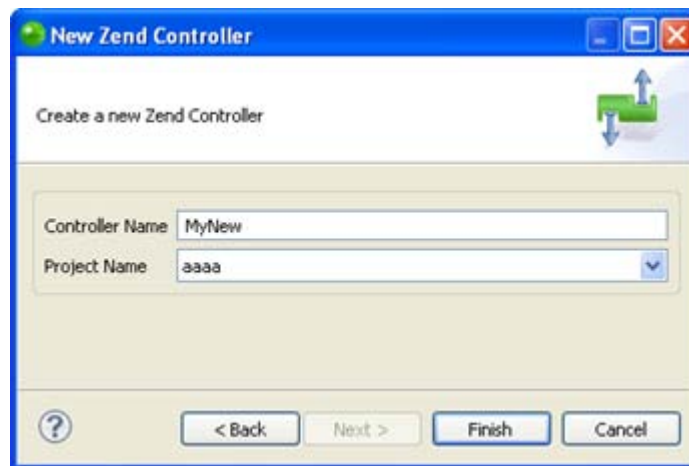
New Zend Controller Wizard

2. If you selected to create a Zend Tool based Controller:

   i. Enter a name for the Controller.
      The default name for the created controller will be <Controller Name>Controller.php (e.g. MyNewController)
      The default location for the file will be in the application's default 'controllers' folder.

   ii. Select the required containing Zend Framework project from the Project Name drop-down list.

If you selected to create a Template-based Controller:

    i.   Ensure the required containing source folder is correct.

         It is recommended to create all Zend Controllers in the project's controllers folder.

    ii.   Enter the Controller name in the File Name field.

    iii.   Click **Next** to view and edit the template on which the new Controller will be created.

    3. Click **Finish**.

A new Zend Controller file will be created with the relevant template.

The new file will be created in the project's 'controllers' folders and displayed in the MVC Outline view.

Creating a Zend Table File

This procedure describes how to create a new Zend Table file.

A Zend Table is a class for working with the "model" portion of the model-view-controller pattern. The Zend Table manages the information and data manipulation in your application.

To create a new Zend Table file:

1. In PHP Explorer view, right-click the models folder in your Zend Framework Project and select **New | Zend Framework Item | Zend Table**.

   The New Zend Table Wizard will be displayed.
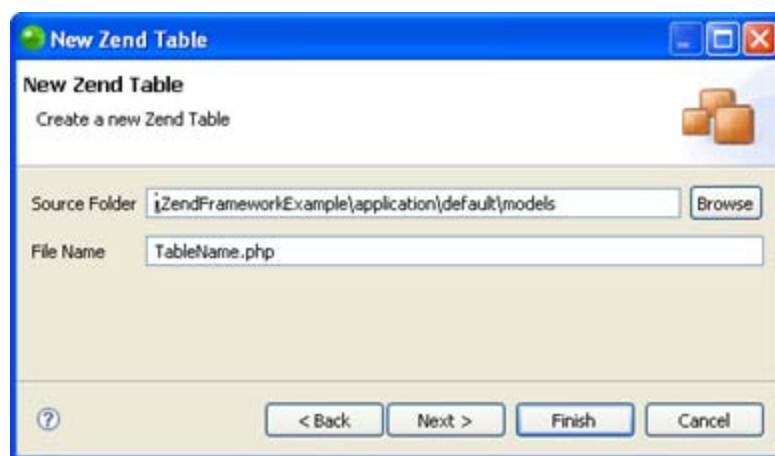
New Zend Table

2. The default location for the file will be in the application's 'models' folder.

   Click Browse next to the Source Folder field to change the location.

3. Edit the file name.

4. Click **Finish**.

A new Zend Table file will be created with the Zend Table template.

The new file will be displayed in the MVC Oultine view.

Creating a Zend View File

This procedure describes how to create a new Zend View file.

A Zend View is a class for working with the "view" portion of the model-view-controller pattern. The Zend View manages the graphic interface aspect of your application.

See http://framework.zend.com/manual/en/zend.view.html for more information on the Zend View.

To create a new Zend View file:

1. In PHP Explorer view, right-click the views folder in your Zend Framework Project and select **New | Zend Framework Item | Zend View**.

   The New Zend View Wizard will be displayed.
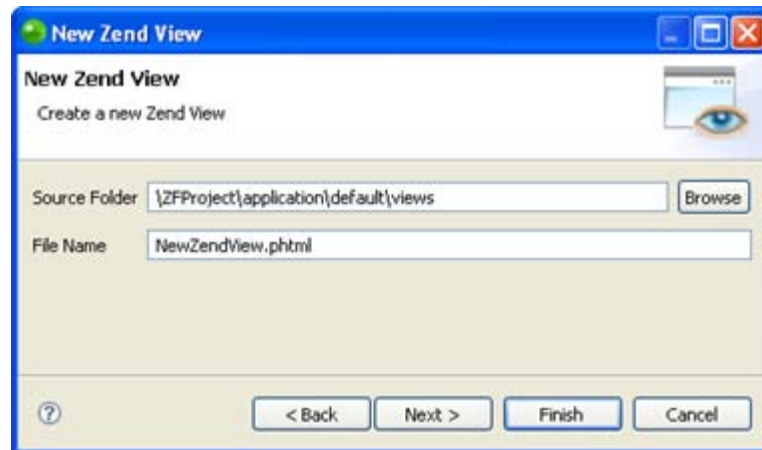
New Zend View wizard

2. The default location for the file will be in the application's default 'views' folder.

   Click Browse next to the Source Folder field to change the location.

3. Edit the File Name if required.

4. Click **Finish**.

A new Zend View file will be created with the relevant template.

The new file will be displayed in the MVC Oultine view.

Creating a Zend View Helper File

This procedure describes how to create a new Zend View Helper File. Zend View Helpers, when attached to a view object, can call the helper as if it were a method of the view object itself. The View object retains helper instances, which means that they retain states between calls.
See http://framework.zend.com/manual/en/zend.view.helpers.html for more on Zend View Helpers.

To create a new Zend View Helper file:

1. In PHP Explorer view, right-click the relevant helpers folder in your Zend Framework Project and select **New | Zend Framework Item | Zend View Helper**.
   The New Zend View Helper Wizard will be displayed.

2. Ensure the source folder is correct or click Browse to change.

3. Enter the Helper's Prefix.

4. Enter the Helper's name. This will be the name of the Helper file.
   The Helper's class name will be automatically created in the format
   <Helper's_Prefix>_<Helper's_Name>.

5. Click **Next**.

6. In the Select PHP Template dialog, ensure the New Zend View Helper template is selected.

7. Click **Finish**.

A new Zend View Helper file will be created with the relevant template. This includes phpDoc block comments which help Zend Studio to recognize that the element is a Zend View Helper.

Note:

Zend View Helper phpDoc block comments must be in the format:

```
/**
* <Helper's_Name> helper
*
* @uses viewHelper <Helper's_Prefix>
*/
```

All Zend View Helpers in your project which are correctly commented will be available in the Content
Assist list in a Zend View file. This includes the default Zend View Helpers included in Zend Framework's
libraries.
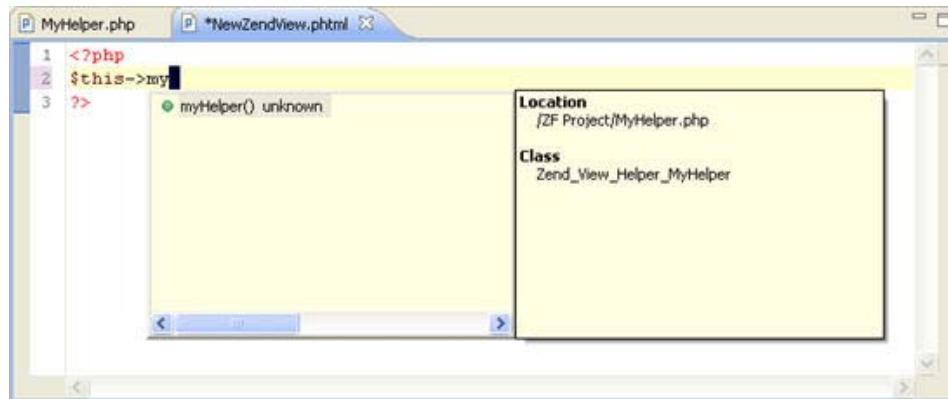
In addition, pressing Ctrl and clicking on a Zend View Helper defined in a View file will take you to the
Zend View Helper's declaration.

Creating a Zend Action Helper

Zend Action Helpers provide an easy way of extending the capabilities of Action Controllers, allowing you to extend Action Controller functionality only when it is needed. Zend Studio allows you to easily create and use Action Helpers within your Zend Framework projects.

For more information on Action Helpers, see http://devzone.zend.com/article/3350-Action-Helpers-in-Zend-Framework.

To create a new Zend Action Helper file:

1. In PHP Explorer view, right-click the relevant helpers folder in your Zend Framework Project and select **New | Zend Framework Item | Zend Action Helper**.
2. The New Zend Action Helper Wizard will be displayed.
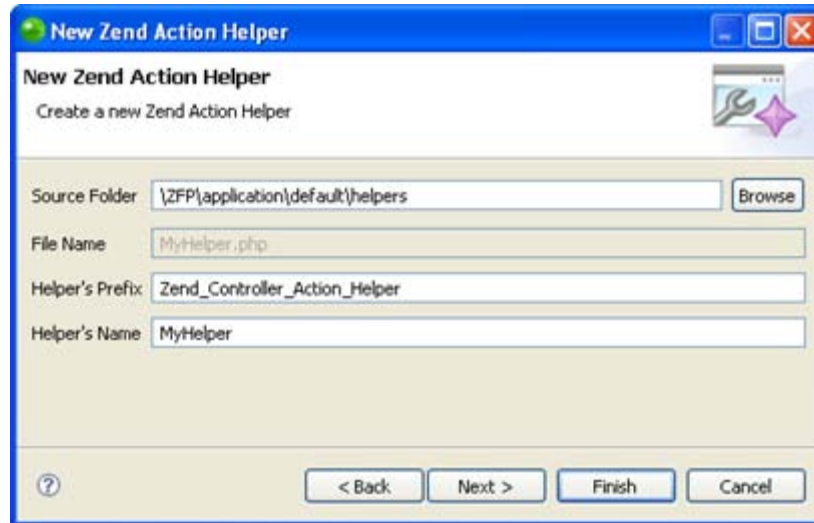
New Zend Controller Action Helper dialog

3. The default location for the file will be in the application's default 'helper' folder.
   Click **Browse** next to the Source Folder field to change the location.
4. Enter the Helper's name. This will be the name of the Helper file.
   The Helper's class name will be automatically created in the format
   <Helper's_Prefix>_<Helper's_Name>.
5. Click **Finish**.

A new Zend Action Helper file will be created with the relevant template. This includes phpDoc block comments which help Zend Studio to recognize that the element is a Zend Action Helper.

All Zend Action Helpers in your project which are correctly commented will be available in the Code Assist list in Zend Controller files. This includes both the default Zend Action Helpers included in Zend Framework's libraries and the Actions Helpers you created.
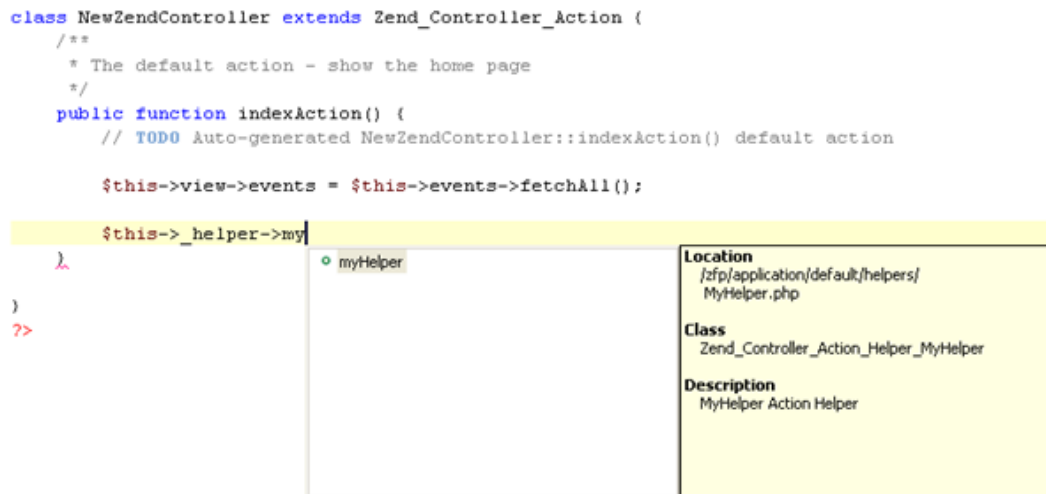
Zend Action Helper Code Assist - Example 1

The Helper broker for the Helper's members and methods are also available for Content Assist:
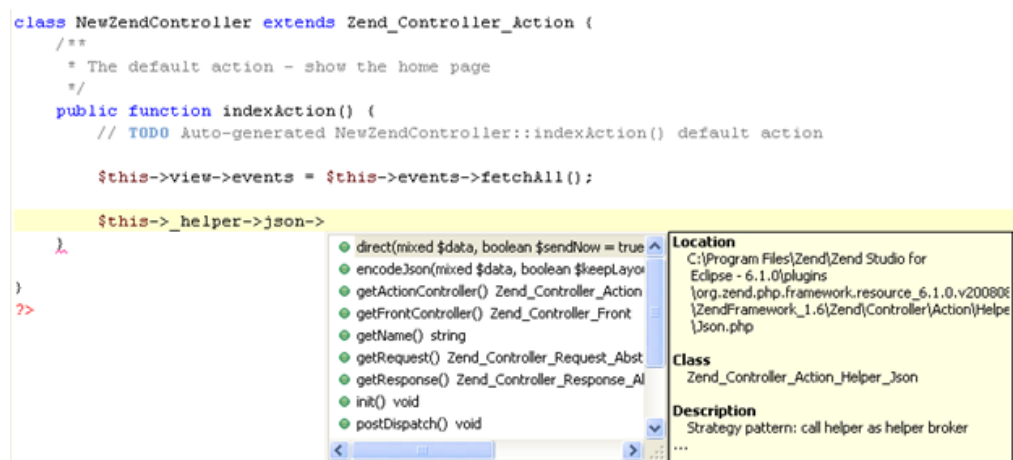
Zend Action Helper Code Assist - Example 2

In addition, pressing Ctrl and clicking on a Zend Action Helper in a Controller file will take you to the Zend Controller Action Helper's declaration.

```
class NewZendController extends Zend_Controller_Action {
    /**
     * The default action - show the home page
     */
    public function indexAction() {
        // TODO Auto-generated NewZendController::indexAction() default action

        $helper = $this->_helper->myHelper;

        $helper->
    }

}
?>
```

Zend View Helper Go To Source

Searching the Zend Framework Site

Zend Studio's workbench contains a search box which allows you to quickly and easily retrieve information from the Zend Framework site, without having to leave the application.

This procedure describes how to search for information in the Zend Framework site.

To search the Zend Framework site:

1. Open the Zend Framework Perspective by going **to Window | Open Perspective | Zend Framework**.

   The Workbench will switch to the Zend Framework Perspective and the Zend Framework search box will appear in the bottom right of the Workspace.

2. Enter a term in the Zend Framework search box.
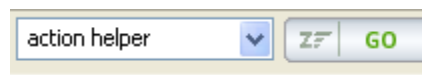
Zend Framework Search Box

3. Click the 'Go' button.

   A new browser window will open in Zend Studio, displaying a list of relevant results from the Zend Framework site.

Note:

The browser will open according to the settings defined in **Window | Preferences | Web Browser**.

4. Click the required result to be taken to the page containing the relevant information.
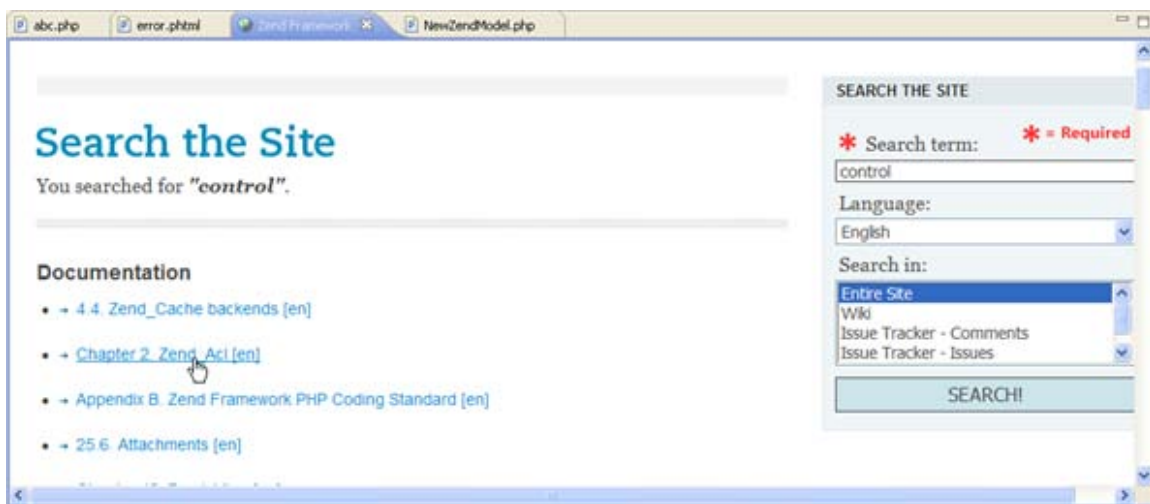
Zend Framework site search results

271

Using the Zend Tool Floating Window

Zend Studio provides you with the ability to execute Zend_Tool commands from within Zend Studio using the Zend Tool Floating Window. This replicates entering a command in your CLI (command line interface).

The Zend_Tool Initative allows PHP programmers to programme according to Rapid Application Development principles.

Note:

You must have Zend Framework version 1.8 installed and configured in your php.ini file for Zend_Tool commands to function.

The latest version of Zend Framework comes bundled with Zend Server or can be downloaded from the Zend Framework site at http://framework.zend.com/download/current.

To execute a Zend_Tool command:

1. In PHP Explorer view, select a Zend Framework project.

2. Press **Ctrl + 2** -or- from the menu bar go to **Project | Zend Tool**.

   The Zend Tool Floating Window displays.
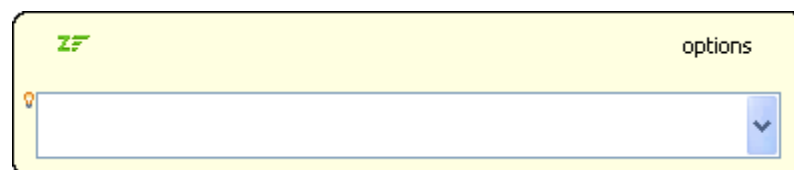
3. Enter the required command.

Note:
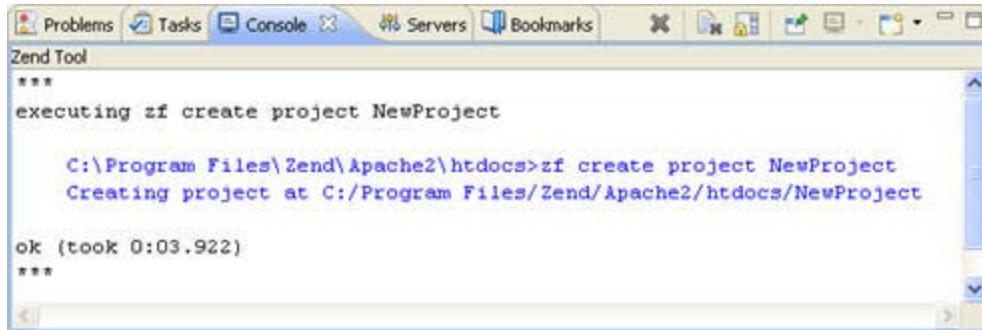
The Zend_Tool currently supports a limited number of functions.

The content assist list displays possible options as you type.
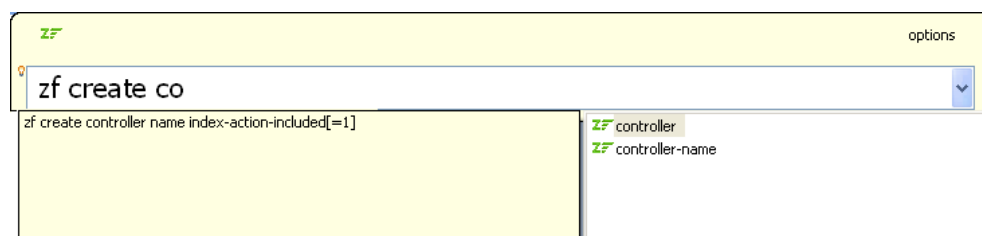
4. Press **Enter**.

The command will be executed and details will be displayed in the Zend Tool Console view.

Console view - Zend Tool

See Zend Framework Preferences for information on customizing the output displayed in the Zend Tool Console.

Connecting to Databases

The following tasks describe how to connect to and interact with a database using the Data Tools Platform:

- Creating a Database Connection Profile
- Connecting to a Database
- Viewing and Editing Database Table Content
- Creating and Executing an SQL Query

For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation .

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help	Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Creating a Database Connection Profile

This procedure describes how to create a connection profile to your database, allowing you to easily connect to it through the Database Development perspective.

The drivers for most of the database connection types are automatically included and configured in Zend Studio.

However, some drivers require you to manually add and configure the driver archive files.

The following driver types come bundled and preconfigured in Zend Studio:

- DB2
- SB2 i-series
- Derby
- MySQL
- Oracle
- Postre SQL
- SQL Server

To create a connection profile:

1. In the PHP Perspective toolbar, click the Create New SQL Connection icon ![icon]

   -Or- in the Database Development perspective (**Window | Open Perspective | Other |**

   Database Development), click the **Create New SQL Connection** icon ![icon] in the Data Source Explorer view or on the main Toolbar

   -Or- right-click the Database Connections folder in the Data Source Explorer view and select **New**.

2. The New Connection Profile wizard opens.
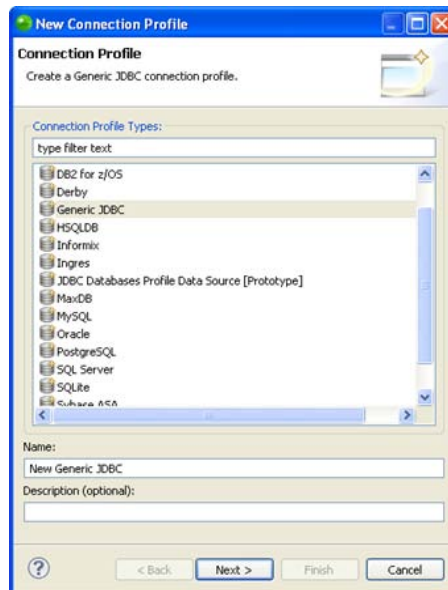
3. Select the required connection profile type from the list.

Enter a name for the connection and a description (if required).

The following driver types come bundled and preconfigured in Zend Studio:

- DB2
- SB2 i-series
- Derby
- MySQL
- Oracle
- Postre SQL
- SQL Server

4. Click **Next**.

   Select a driver from the drop-down list.

   If the required database driver has not been configured, click the New Driver Definition

   button   to the right of the drop-down list and perform the following steps:

   i. In the Name/Type tab, select the required driver type and version.
   ii. Select the Jar List tab.
   iii. Click **Add JAR/Zip**.
   iv. Browse to the JAR/Zip file containing the required driver files and click **Open**.
   v. Click **OK**.

      The new driver is added to the driver definition list.

5. The properties for the selected driver are displayed.

   These will vary depending on the database type.

6. Click the **Test Connection** button to ensure all the details have been entered correctly.

7. Click **Next** to see a summary of your Connection Profile's details.

8. Click **Finish**.

Your new connection profile will be added to your databases list in the Data Source Explorer view.

You can now use this Connection Profile to connect to your database.

See the Connecting to a Database topic for more on how to connect to your database.

For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation .

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Connecting to a Database

Once you have established your connection profile, you can connect to your database from the Data Source Explorer view.

This procedure describes how to connect to your database.

To connect to your database:

1. Open the Database Development perspective by going to **Window | Open Perspective | Other | Database Development.**

2. In the Data Source Explorer view, expand the SQL Databases node and right-click your connection profile.

   To edit the properties of the Connection Profile, right-click it and select **Properties**.

3. Click **Connect**.

4. Once the connection has been established, you can expand the tree underneath your Connection Profile to view the contents of the database.
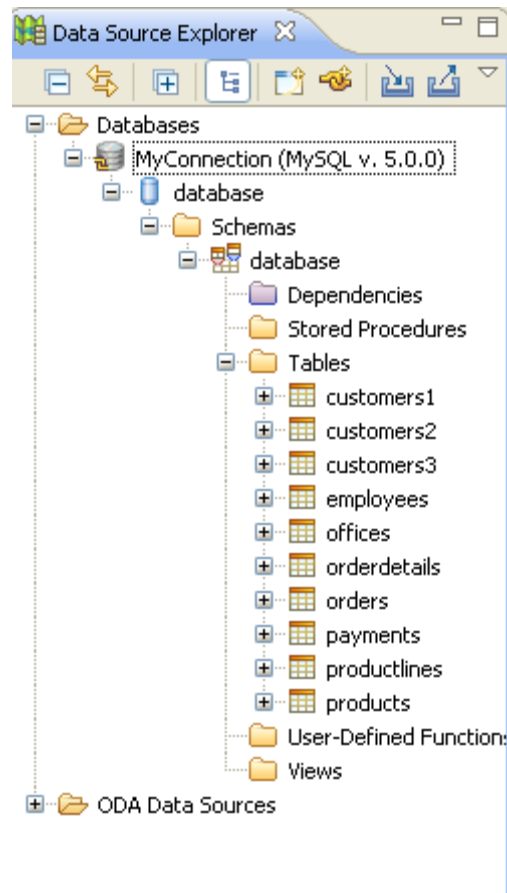
Data Source Explorer

To see a sample of the data in the tables, right-click one and select **Data | Sample Contents**.

The SQL Results view will open with a sample list of the data from your table.

SQL Results view

For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation .

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Viewing and Editing Table Content

This procedure describes how to view data from a table located in your database and edit its contents.

To view and edit table content:

1. Open the Database Development perspective by going to **Window | Open Perspective | Other | Database Development**.

2. Connect to your Database by following the steps in the "Connecting to a Database" topic.

3. In Data Source Explorer view, double-click the required table or right-click it and select **Data | Edit**.

   The table will open in a database editor displaying all the data within the table.

customerNumber ...	customerName ...	contactLastName [VA...	contactFirstName [VARCH
103	Atelier graphique	Schmitt	Carine
112	Signal Gift Stores	King	Jean
114	;lkj;lj;j	Ferguson	Peter
119	La Rochelle Gifts	Labrune	Janine
121	Baane Mini Imports	Bergulfsen	Jonas
124	Mini Gifts Distrib...	Nelson	Susan
125	Havel & Zbyszek...	Piestrzeniewicz	Zbyszek
128	Blauer See Auto...	Keitel	Roland
129	Mini Wheels Co....	Murphy	Julie
131	Land of Toys Inc.	Lee	Kwai
141	Euro+ Shopping...	Freyre	Diego
144	Volvo Model Rep...	Berglund	Christina
145	Danish Wholesal...	Petersen	Jytte

Table Contents

4. Select a cell to edit its contents.

5. Click **Save** 💾 .

   The changes made will be automatically applied to the database.

For more information on the Data Tools Platform, see the Data Tools Platform User Documentation.

Creating a MYSQL Driver Definition

This procedure describes how to create a driver definition for your database. A driver definition maps to the location of your driver files.

You must have access to MySQL driver files before attempting to create a driver definition.

To create a driver definition for MySQL:

1. Open the Driver Definitions Preferences page by going to **Window | Preferences | Connectivity | Driver Definitions**.

2. A list of driver definitions will appear in the preferences page.

3. Scroll down to the MySQL category and select the required MySQL driver.
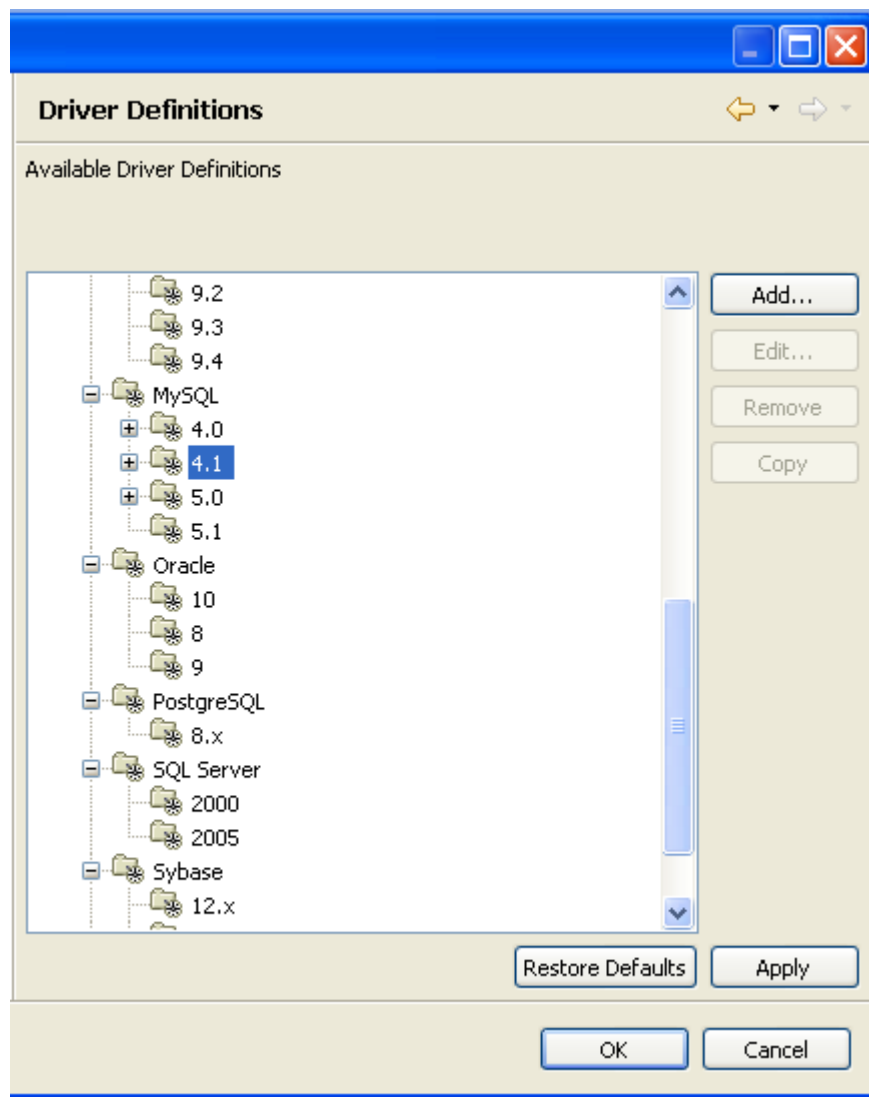
4. Click **Add**.

    The New Driver Definition wizard opens.
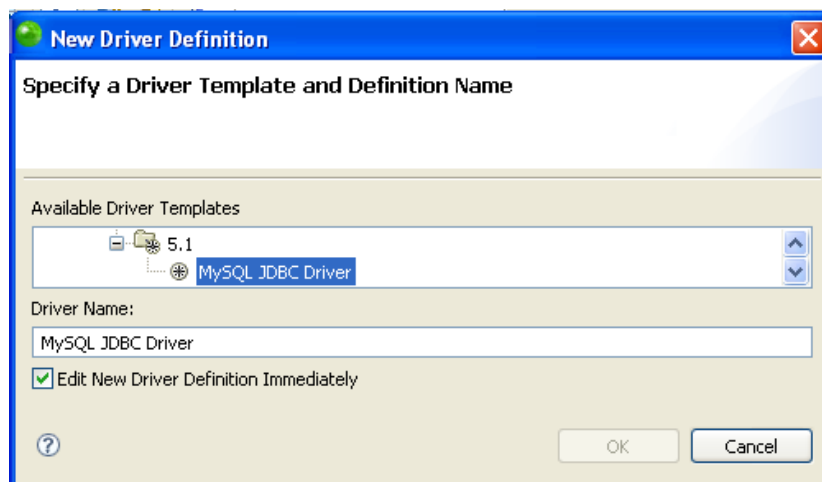
New Driver Definition dialog

5. Expand the nodes until you see the MySQL JDBC Driver.

6. Select the driver.

    The default driver name will be entered in the Driver Name field. If required, delete this name
    and enter a new name.

7. Ensure that the "Edit New Driver Definition Immediately" checkbox is marked.

8. Click **OK**.

    The Driver Details wizard opens.
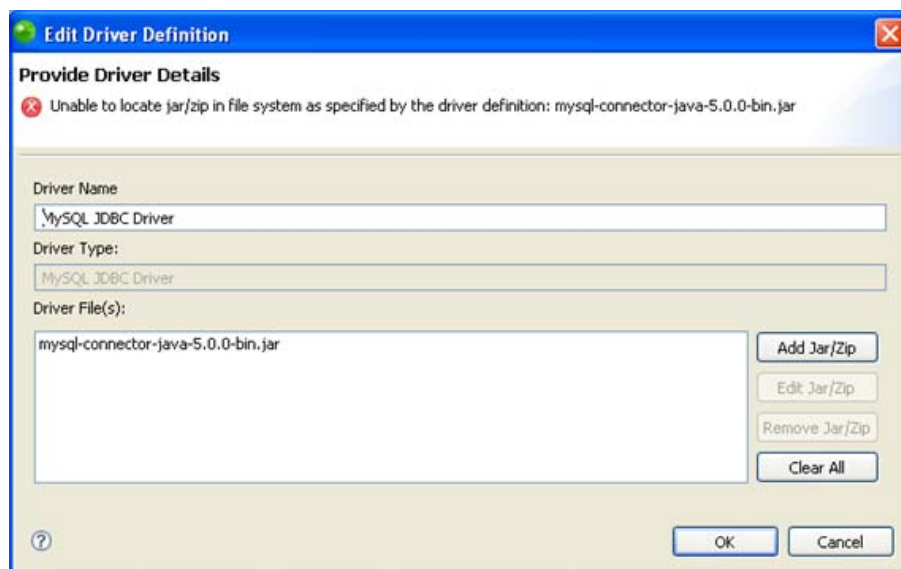
Edit Driver Definition dialog

9. Delete the sample .jar driver file by selecting it and clicking Remove Jar/Zip.

10. Click **Add Jar/Zip**.

11. Browse your file system to find your MySQL driver and press **OK**.

    Your driver will be added to the list.

12. Click **OK**.

Your new driver definition will be added to your Driver Definitions list. This will allow you to access the driver when connecting to a MySQL database.
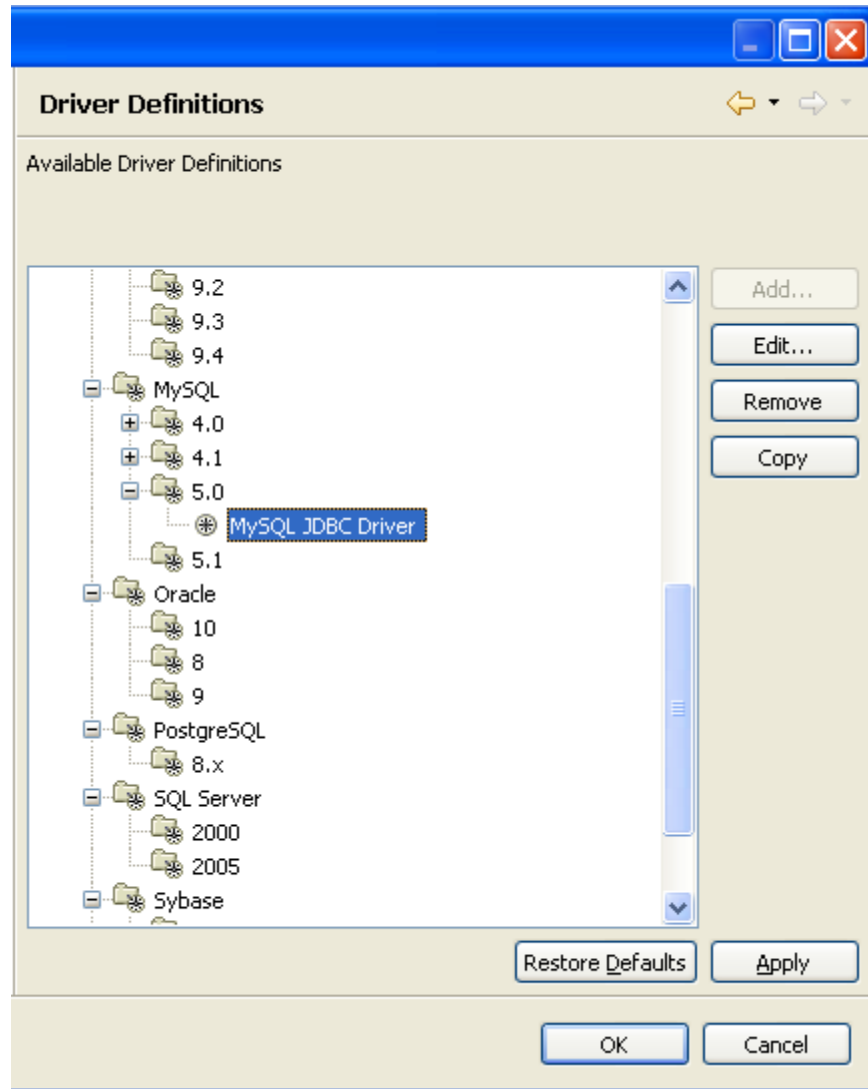
Driver Definitions preferences - with new driver

Creating and Executing an SQL Query

This procedure describes how you can run an SQL query on your database once you have created it. You must have [created a Connection Profile](#) and [connected to your database](#) before using this functionality.

To run an SQL Query on your database:

1. Open the Database Development perspective by going to **Window | Open Perspective | Other | Database Development.**

2. Connect to your Database by following the steps in the [Connecting to a Database](#) topic.

3. Click the **Open Scrapbook** icon 🖿 on the toolbar.
   A new SQL scrapbook will open.
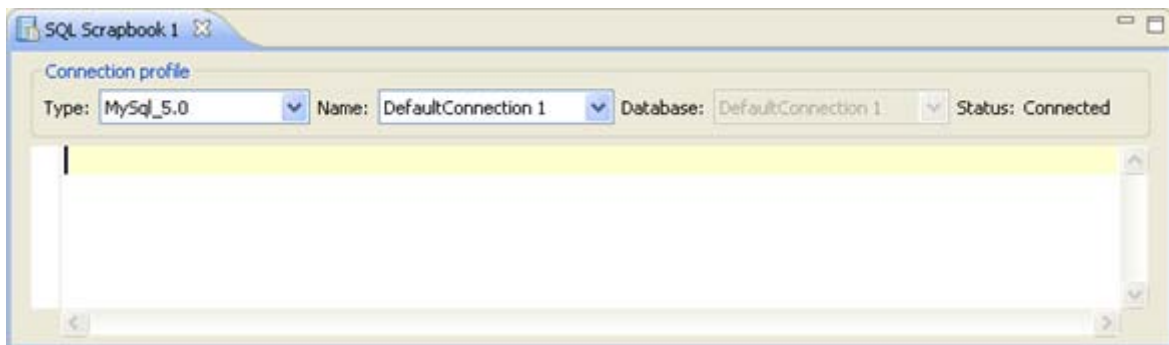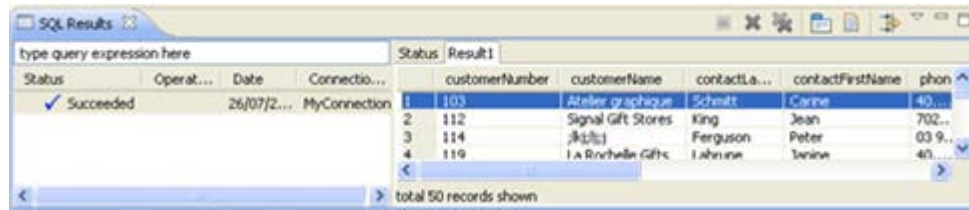
SQL File editor

4. Write your query in the scrapbook (e.g. select * from mytablename;)

5. To execute your query, right-click anywhere in the editor and select **Execute All** -or- press **Ctrl+Alt+X**.
   To execute only specified queries, highlight the relevant lines, right-click and select **Execute Selected Text**.

The query will be run and the results will be displayed in the Result1 tab in the SQL Results view.

The left pane displays the execution history. For each statement that you execute, including stored procedures, an execution history entry is added to this pane. This allows you to quickly retest the execution using slightly different values and settings. You can rename or delete the launch configurations as needed.

SQL Results view

For more information on the Data Tools Platform, please see the Data Tools Platform User

Documentation.

Running Files and Applications

The following options are available when running your files and applications:

- [Running PHP Scripts Locally](#) - Run files on your workspace using Zend Studio's internal debugger.
- [Running PHP Web Pages](#) - Run applications situated on a server.

Running PHP Scripts Locally

This procedure describes how to run a PHP Script from your workspace using Zend Studio's internal debugger.

To locally run a PHP Script:

1. Click the arrow next to the Run button [ ] on the toolbar and select **Run Configurations** -or- go to **Run | Run Configurations**.
   A Run dialog will open.
2. Double-click the PHP Script option to create a new run configuration.
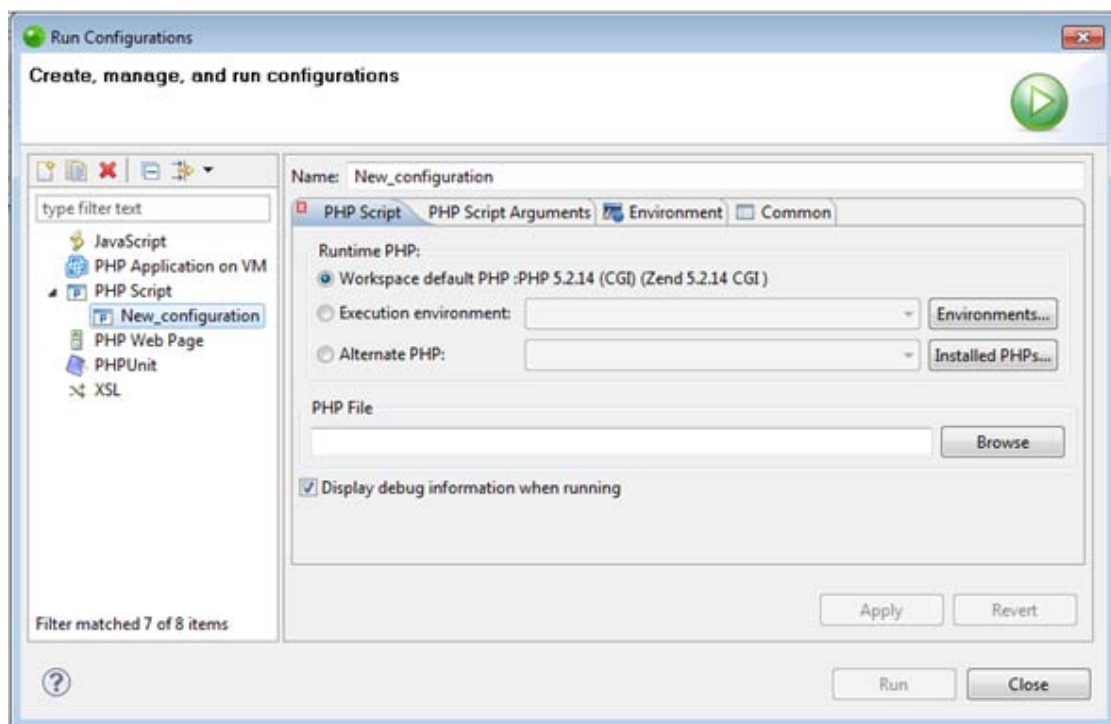
3. Enter a name for the new configuration.
4. Select from the following options:

- Workspace default PHP - Select to use the default PHP and execution environment defined in the Execution Environments Preferences page.

- Execution environment - Select the execution environment you would like to use for your debug configuration from the dropdown list.
  Click **Environments** to change the settings in the Execution Environments Preferences page.

- Alternate PHP - Select to use another PHP which can be selected from the Installed PHP's list.

5. Enter your PHP file in the "PHP File" text field, or click **Browse** and select your file.
6. Marking the 'Display debug information when running' checkbox will cause debug views to be displayed.
7. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.
8. Click **Apply** and then **Run**.

Your script will be run and displayed in a browser.

Note:

If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.

Running PHP Web Pages

This procedure describes how to run whole applications, projects, files or collections of files that are on the server. Using this process, you can run either copies of the files which are located on the server or the files located locally on your workspace (if available).

To run a PHP Web Page:

1. Click the arrow next to the Run button [▶] on the toolbar and select **Run Configurations** -or- go to **Run | Run Configurations**.

   A Run dialog will open.

2. Double-click the PHP Web Page option to create a new run configuration.

New Run Configuration

3. Enter a name for the new configuration.

4. Select the Server Debugger to be used (by default this will be the Zend Debugger).

5. Select your server from the PHP Server list.

   If you have not yet configured your server, click **New**.

   The PHP Server Creation Wizard will open.

   Configure your server by following the instructions on '<u>adding a new server</u>' under the <u>PHP Servers Preferences</u> page.

   For more information on configuring the communication between Zend Studio and your remote server, see <u>Setting Up Remote Debugging</u> .

6. To check whether your server connection is correctly configured and that Zend Studio can communicate with your server debugger, click the 'Test Debugger' button.

7. Under PHP File, click **Browse** and select the file which you would like to run.

8. The URL to be run will have been automatically created based on the file name and your server address. If the URL does not point to the file's location, unmark the Auto Generate checkbox and modify the URL.

> **Note:**
> The file to be run needs to exist on the server even if you are going to be selecting to run the local copy of your files.

9. For further options, select the Advanced tab.
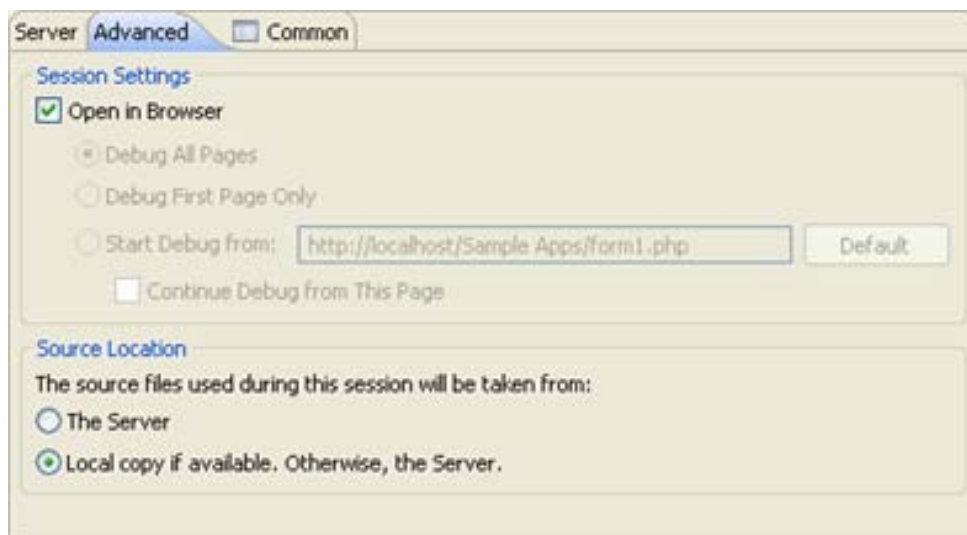
New Run Configuration - Advanced

10. Under the Source Location category you can choose whether the content of the files to be run will be taken from the server or from your workspace.

If a local copy is not available, files will be taken from the server. Selecting the 'Local Copy' option will result in the Path Mapping mechanism being applied when files are called. See the Path Mapping topic for more details.

> **Note:**
> The file to be run needs to exist on the server even if you are going to be selecting to run the local copy of your files.

11. Click **Apply and Run**.

Your application will be run and displayed in a browser.

> **Note:**
> If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.
> In addition, if a file defined with an absolute path to a server location (See 'Include Paths' for more on absolute file locations) is called, a Path Mapping dialog will appear. See Path Mapping for more information.

Debugging Files and Applications

The following Debug functionality is available in Zend Studio:

- [Local PHP Script Debugging](#) - Debug files on your workspace using Zend Studio's internal debugger.
- [PHP Web Page Debugging](#) - Debug applications situated on a server.
- [URL Debugging](#) - Enter a URL to debug an application on a server.
- [Debugging Using the Zend Debugger Toolbar](#) - Debug files and applications directly from your browser.

Once a debug session has been launched, the [PHP Debug perspective](#) is used to control the debugging process and to view and analyze the results.

See the "[Running and Analyzing Debugger results](#)" topic for more information on controlling and monitoring the debugging process.

Setting Breakpoints

Before you debug your scripts, you can set breakpoints in them to specify places in your code where the debugging process will pause.

These procedures demonstrate how to set line and conditional PHP breakpoints.

Setting a Breakpoint in Your Script

To set a breakpoint in your script:

Double-click the vertical ruler to the left of the line where you want to set the breakpoint -or- select the line and go to **Run | Toggle Breakpoint** or press **Ctrl+Shift+B**.

A blue ball will appear, indicating that a breakpoint has been set.
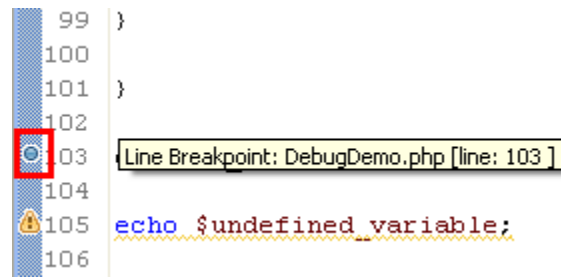
Set Breakpoint

Adding a Condition to a Breakpoint

To add a condition to a breakpoint:

1. Right-click the breakpoint in the vertical marker bar and select **Breakpoint Properties**.
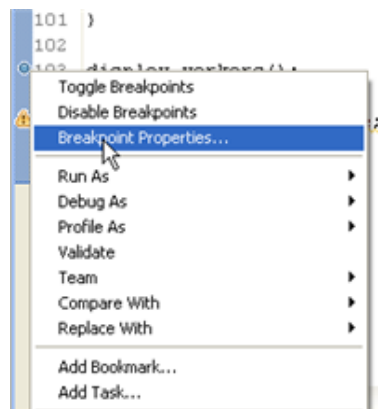
Breakpoint Properties Selection

2. Enter the required condition and ensure the Enable Set Condition checkbox is marked.
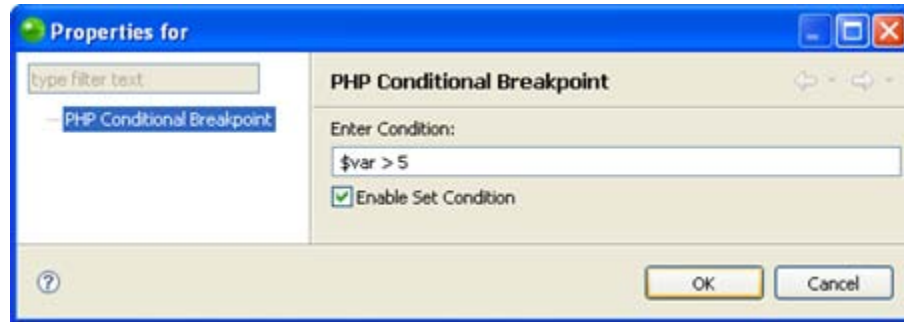
PHP Conditional Breakpoint Properties

3. Click **OK**.

The condition will be set for the breakpoint and the debugging process will pause only if the condition is met.

The breakpoint will be represented by a blue ball with a question mark.
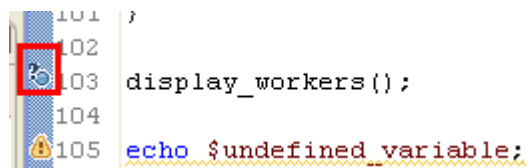
Conditional Breakpoint

Using the Inspect Action

The Inspect Action is a quick way to evaluate an expression when debugging your PHP script.

1. Set a breakpoint to your code by double-clicking the vertical ruler to the left of the line where you want to set the breakpoint -or- select the line and go to **Run | Toggle Breakpoint** or press **Ctrl+Shift+B.**

   A blue ball will appear, indicating that a breakpoint has been set.

2. Select an expression in the editor and from the Right Click Menu select **Inspect** - or - press **Ctrl+Shirft+I**.

   The evaluated expression is shown.
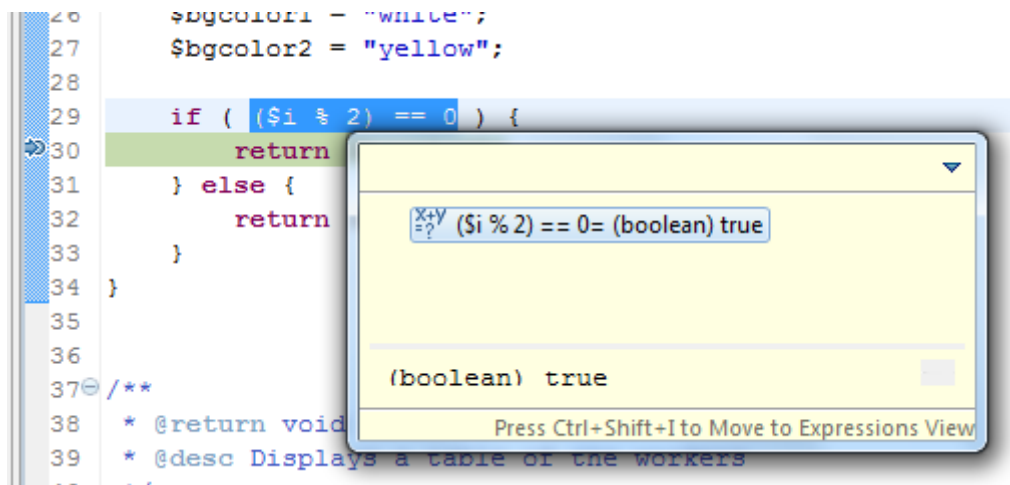
For more information on Debugging methods see Debugging.

Locally Debugging a PHP Script

This procedure describes how to debug a PHP Script from your workspace using an internal PHP
Executable.

To locally debug a PHP Script:

1. Set breakpoints at the relevant places in the file that you would like to debug by double-
   clicking the vertical marker bar to the left of the editor.
2. Save the file.
3. Click the arrow next to the debug button ![debug button] on the toolbar and select Debug
   Configurations... -or- select **Run | Debug Configurations....**
   A Debug dialog will open.
4. Double-click the PHP Script option to create a new debug configuration.
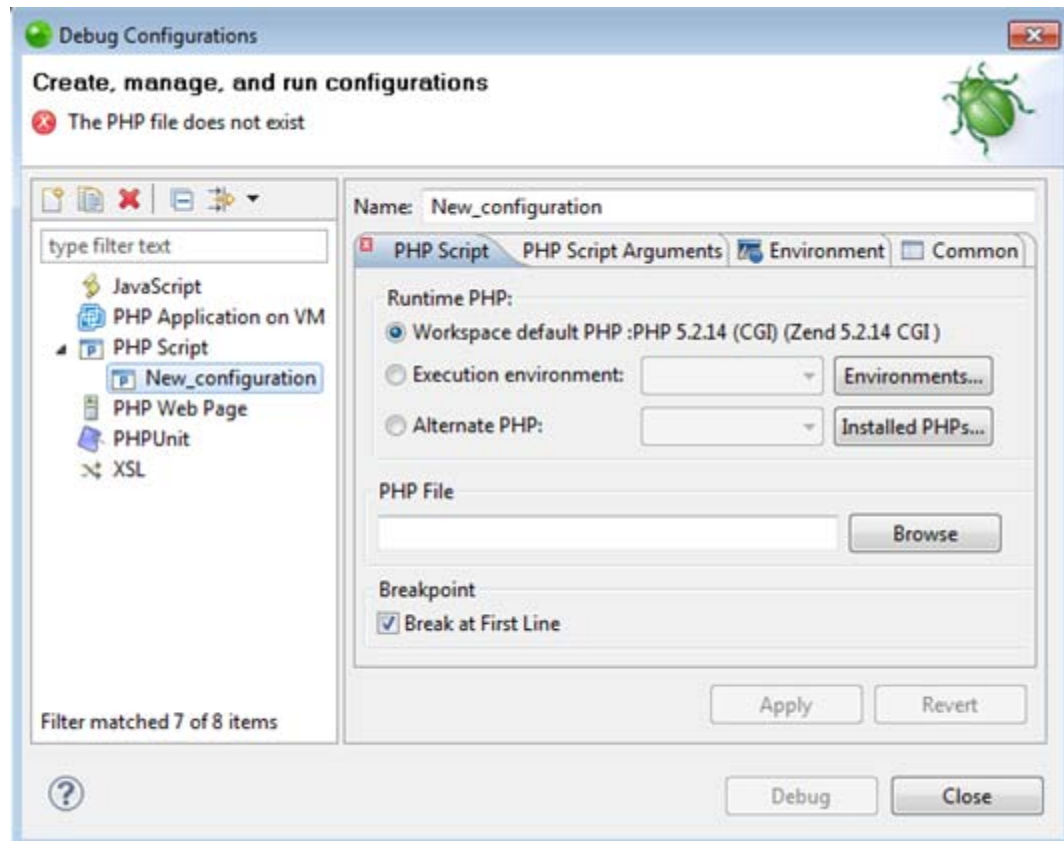
5. Enter a name for the new configuration.
6. Select from the following options:

- Workspace default PHP - Select to use the default PHP and execution environment defined
  in the Execution Environments Preferences page.

- Execution environment - Select the execution environment you would like to use for your debug configuration from the dropdown list.
  Click **Environments** to change the settings in the [Execution Environments Preferences](#) page.

- Alternate PHP - Select to use another PHP which can be selected from the Installed PHP's list.

7. Enter your PHP file in the "PHP File" text field, or click **Browse** and select your file.

8. Marking the "Break at First Line" checkbox will result in the debugging process pausing at the first line of code.

9. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.

10. Click **Apply** and then **Debug**.

11. Click **Yes** if asked whether to open the PHP Debug Perspective.

A number of views will open with relevant debug information.

See the [Running and Analyzing Debugger results](#) topic for more information on the outcome of a debugging process.

Note:
If the file contains 'include' or 'require' calls to files which are not contained within the project, you must [add them to the project's Include Path](#) in order to simulate your production environment.

Debugging a PHP Web Page

This procedure describes how to debug whole applications, projects, files or collections of files that are already on the server.

> **Note:**
> Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.
>
> The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To debug a PHP Web Page:

1. Click the arrow next to the debug button [icon] on the toolbar and select **Debug Configurations** -or- select **Run | Debug Configurations**.

   A Debug dialog will open.

2. Double-click the PHP Web Page option to create a new debug configuration.

New Debug Configuration

3. Enter a name for the new configuration.
4. Select the Server Debugger to be used (by default this will be the Zend Debugger).

5. Select your server from the PHP Server list.

   If you have not yet configured your server, click **New**.

   The PHP Server Creation Wizard will open.

   Configure your server by following the instructions on 'adding a new server' under the PHP Servers Preferences page.

   For more information on configuring the communication between Zend Studio and your remote server, see Setting Up Remote Debugging .

6. To check whether your server connection is correctly configured, and that Zend Studio can communicate with your server debugger, click the 'Check Debug Server Connection' button.

7. Under PHP File, click **Browse** and select your 'debug target' file (the file from which the debugging process will start.)

8. Select whether the Debugger should stop at the first line of code by marking/unmarking the 'Break at First Line' checkbox.

9. The URL to be debugged will have been automatically created based on the file name and your server address. If the URL does not point to your debug target's location, unmark the Auto Generate checkbox and modify the URL.

10. For further Debug options, select the Advanced tab, which has the following options:
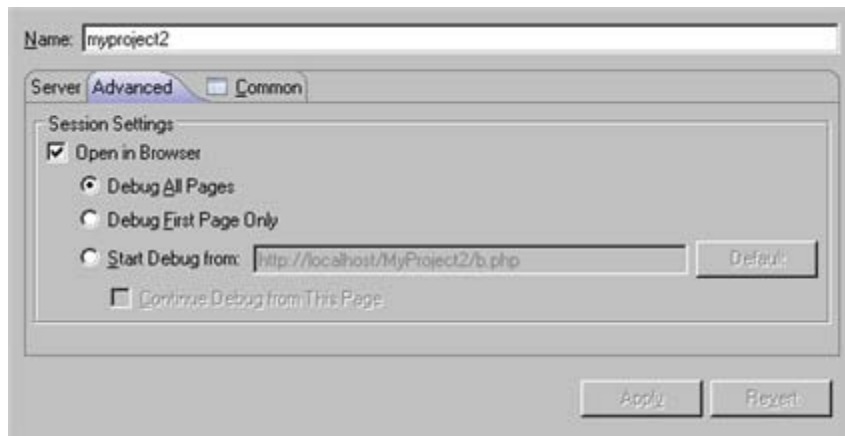
New Debug Configuration - Advanced

- Open in Browser - Mark if you would like the application to be displayed in Zend Studio's internal browser while it is debugged.

- Choose whether to:

  - Debug All Pages - The specified page and all the pages linked to it are debugged. The browser waits for the debug of each page before displaying it.

  - Debug First Page Only - Only the first page is debugged.

  - Start Debug from - Select the URL from which you would like the Debugging process to start.

- Continue Debug from this Page - Selecting this option will result in all the pages linked to the URL being debugged.

- Source Location - Choose whether the source files used during this session will be taken from the server or from a local copy.

  If a local copy is not available, files will be taken from the server. Selecting the 'Local Copy' option will result in the Path Mapping mechanism being applied when files are called. See the [Path Mapping](#) topic for more details.

11. Click **Apply and then Debug**.
12. Click **Yes** if asked whether to open the PHP Debug Perspective.

See the "[Running and Analyzing Debugger results](#)" topic for more information on the outcome of a debugging process.

Note:
If the file contains 'include' or 'require' calls to files which are not contained within the project, you must [add them to the project's Include Path](#) in order to simulate your production environment. In addition, if a file defined with an absolute path to a server location (See '[Include Paths](#)' for more on absolute file locations) is called, a Path Mapping dialog will appear. See [Path Mapping](#) for more information.

Debugging a URL

This procedure describes how to debug a URL on a server to which you have access.

Note:
Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To debug a URL:

1. Click the Debug URL button ![icon] on the main toolbar -or- go to **Run | Debug URL**.
2. The Debug URL dialog will appear.
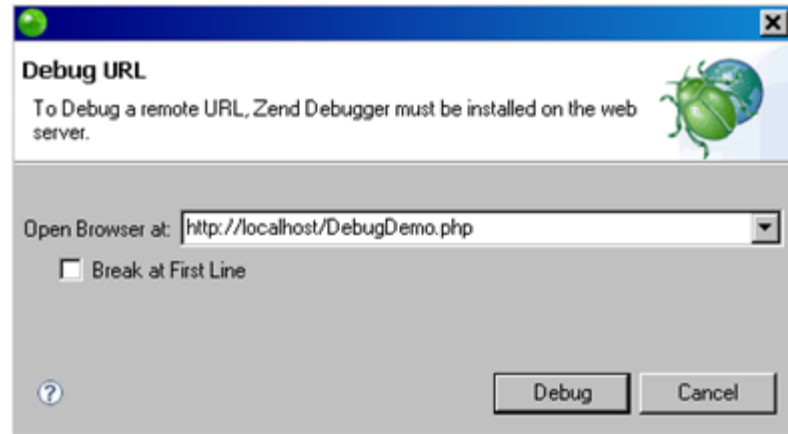
Debug URL dialog

3. In the 'Open Browser at' field, enter the URL of the first page that should be debugged.
4. Select whether the Debugger should stop at the first line of code by marking/unmarking the 'Break at First Line' checkbox.
5. Click **Debug**.

The Debug Perspective will open with a number of views detailing information about the debugging process.

See the "Running and Analyzing Debugger results" topic for more information on the outcome of a debugging process.

Debugging Using the Zend Browser Toolbar

 This procedure describes how to debug using the Zend Browser Toolbar.

Note:
In order to enable Zend Studio to debug files located on the server, remote debugging needs to be configured. See Setting Up Remote Debugging for more details.

To debug using the Zend Browser Toolbar:

1. Ensure the Zend Browser Toolbar is installed on your browser.
   If you have not yet installed the Zend Browser Toolbar, see Installing and Configuring the Zend Browser Toolbar for more information.

2. Open your browser and browse to the page from which you would like to start debugging.

3. Ensure Zend Studio is open. You can open it by clicking the Zend Studio button 🟢 Zend Studio in the toolbar.

4. Click the 🐞 Debug button on the Zend Browser Toolbar to debug the page currently displayed in the browser.
   To do this, the page is reloaded, and instructions are automatically transmitted to the server for the execution to be done in debug mode. This means that if the POST data has been transmitted to the current page, the browser will ask if the user wishes to post them again before executing the page in debug mode. In the same way, if the page uses frames (including hidden ones), the toolbar will request which frame the user wants to debug.
   -Or- select one of the following debug options by clicking the arrow to the right of the Debug button 🖼 ▾ :

   - Next page on site - The debugging session will be launched when the next link is clicked, a form is posted, or an AJAX request is executed.
   - All forms (POST) on this site - The debugging session will be launched every time a link is clicked, a form is posted or an AJAX request is executed using the POST method. The script that will be debugged will be the script designated as the action of the form or link.
   - All pages on this site - Debugs all pages from the current page.

Note:
If the files you would like to debug exist in your workspace, you can choose to debug the workspace copy of your files by going to Extra Stuff | Settings on the Toolbar and selecting the 'Debug Local Copy' option.
In Zend Studio, if path mapping has not yet been configured, a path mapping dialog will be

displayed once the debugging session is launched to determine which workspace files will be debugged. See [Path Mapping](#) for more details.

The relevant debug session will be launched in Zend Studio.

Installing and Configuring the Zend Browser Toolbar

Installing the Zend Browser Toolbar During Zend Studio Installation

The Zend Browser Toolbar can be installed during the installation of Zend Studio or can be downloaded and installed separately.

To install the Zend Browser Toolbar during Zend Studio installation:

In the 'Choose Install Set' dialog of the Zend Studio installation, select the Zend Firefox Toolbar and/or the Zend Internet Explorer Toolbar options:

Zend Studio Installation

Manually Installing the Zend Internet Explorer Toolbar

To manually install the Zend Internet Explorer Toolbar:

1. Go to the Zend Studio downloads site (http://www.zend.com/en/products/studio/downloads) and download the Studio Browser Toolbar for Internet Explorer on Windows.
2. Save the file to your file system.
3. Run the following command (by going to **Start | Run** in the Windows Start Menu):

    ```
    regsvr32.exe "<ZendIEToolbar.dll_Location>\ZendIEToolbar.dll"
    ```

    Replace <ZendIEToolbar.dll_Location> with the path to the directory in which you saved the downloaded file.
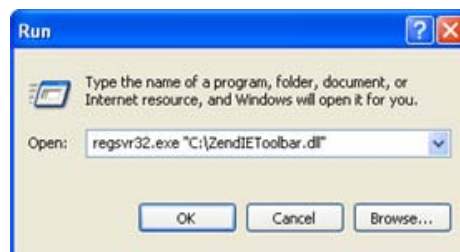
4. A dialog will be displayed confirming the successful registration of the .dll.

5. Restart Internet Explorer.

6. If the Toolbar is not automatically displayed, select the Zend Studio toolbar from the Internet Explorer Toolbars list.

Note:

In Internet Explorer 7, the Toolbar list is available from Tools | Toolbars.

You should now configure the Zend Studio communication settings for the Zend Browser Toolbar in order to be able to debug/profile.

Manually Installing the Zend Firefox Toolbar

To manually install the Zend Firefox Toolbar:

1. Go to the Zend Studio downloads site (http://www.zend.com/en/products/studio/downloads) and download the cross-platform Firefox Browser Toolbar.

2. Open the downloaded .xpi file by going to **File | Open File** from the Firefox menu bar and browsing to the downloaded file.

3. A prompt will be displayed asking whether you trust the add-on.
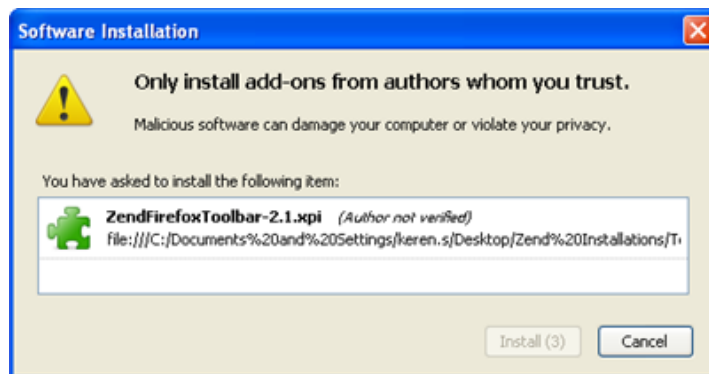
4. Click **Install Now**.

The add-on will be displayed in the Firefox Add-ons list.
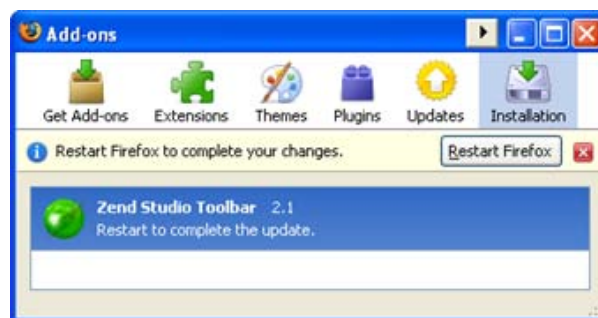
5. Click the Restart Firefox button.

   Firefox will be restarted and the toolbar will be loaded.

6. If the Toolbar is not automatically displayed, select the Zend Studio toolbar from Firefox's Toolbars list.

> **Note:**
>
> In Firefox 3, the Toolbar list is available from View | Toolbars.

You should now configure the Zend Studio communication settings for the Zend Browser Toolbar in order to be able to debug/profile.

Configuring the Zend Browser Toolbar

In order to be able to debug files and applications through the Zend Browser Toolbar, you must configure the toolbar to communicate with Zend Studio.

Toolbar debugging is a form of remote debugging and so remote debugging to the server on which your files are located must also be configured in Zend Studio and the Zend Debugger. See Setting Up Remote Debugging for more details.

This procedure describes how to configure the Zend Browser Toolbar to be able to communicate with Zend Studio.

To configure connectivity with Zend Studio:

1. From the Toolbar, go to **Extra Stuff | Settings**.

   The Zend Toolbar Settings dialog will be displayed.
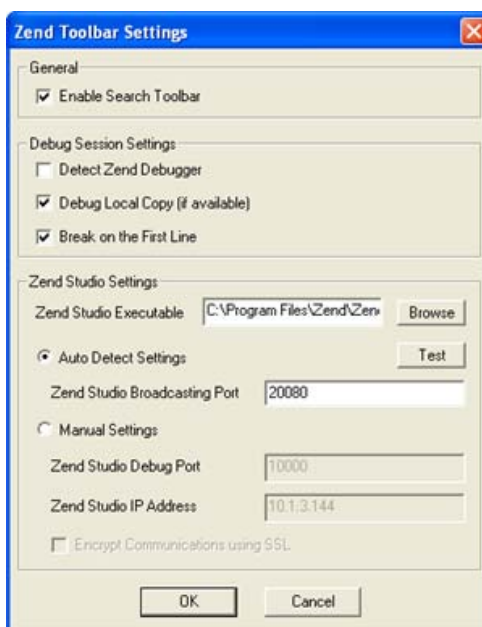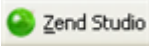
Zend Toolbar Settings

2. In the Zend Studio Settings category, configure the Zend Studio Debug Port and Zend Studio IP Address. These settings should match the settings defined for the Zend Debugger in the Installed Debuggers Preferences page.
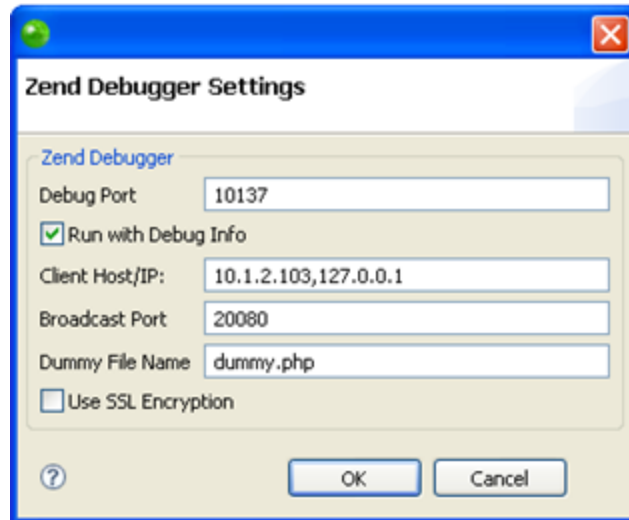
Zend Debugger Settings

To automatically synchronize your Zend Browser Toolbar settings with your Zend Studio settings:

i. Ensure Zend Studio is open.
If your Zend Studio executable is configured in the Zend Toolbar Settings dialog, you can launch Zend Studio by clicking the Zend Studio button in the toolbar.

ii. In the Zend Studio Settings category, select 'Auto Detect Settings'.

iii. Enter the Zend Studio Broadcasting Port. This must match the Broadcasting Port configured in your Zend Studio debug preferences.
The default port is 20080.

iii. Click **Test**.
The Debug Port and Zend Studio IP settings defined in Zend Studio will be automatically updated in your Zend Browser Toolbar and a message will appear confirming that the Auto Detect test was completed successfully.

To manually enter your settings:

i. Select 'Manual Settings'.

ii. Configure the following settings:

- Zend Studio Debug Port - The debug port to be used for the debug session.
  This should match the Debug Port setting configured in your Zend Studio debug

preferences.

- Zend Studio IP Address - Enter the Client Host/IP address of the machine on which your Zend Studio is installed. This should be the same machine from which you are launching your debug session.

  This should match the Client Host/IP setting configured in your Zend Studio debug preferences.

3. Click **OK** to save your settings.
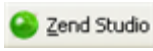
Additional Configuration Options

Additional configuration options can be accessed through the Zend Studio Toolbar Settings dialog (**Extra Stuff | Settings**) as follows:

- **General**:
  - Enable Search Toolbar - Displays the PHP information search box in the Zend Browser Toolbar.
- **Debug Session Settings**:
  - Detect Zend Debugger (only available in the Zend Internet Explorer Toolbar) - disables the Debug and Profile buttons when a Zend Debugger is not detected on the server.
  - Debug Local Copy - Choose whether the source files used during the debug session will be taken from the server or from a local workspace copy of the files in Zend Studio (if available). If a local copy is not available, files will be taken from the server.

Note:
Enabling this option will result in the Path Mapping mechanism being applied when files are called. See Path Mapping for more details.

  - Break on the first line - this will cause the debugger to pause at the first line of PHP code during debugging.
- **Zend Studio Settings**:
  - Zend Studio Executable - Browse to the location of your Zend Studio executable file. This will enable you to launch Zend Studio by clicking the Zend Studio button ![Zend Studio] in the toolbar.

Running and Analyzing Debugger Results

Once you have launched one of the debug sessions (PHP Script, PHP Web Page or URL), you can control and monitor the debugging process using the views displayed in the debugging process.

Controlling the debugging process

The debug process can be controlled using the various buttons in the Debug view.

1. The Debug process will automatically stop at each breakpoint.
2. The various views will display information about the debugging process up to that point only.
3. You can use the various buttons in the debug view to decide how to continue with the debugging process:

- Click the Resume button ▶ to continue the debugging process until the next breakpoint, or until the end of the debugging process.

- Click the Terminate button ■ to stop the debugging process.

- Click the Step Over button to step over the next method call (without entering it) at the currently executing line of code. The method will still be executed.

- Click the Step Return button to return from a method which has been stepped into. The remainder of the code that was skipped by returning is still executed.

- Click the Step Into button to step into the next method call at the currently executing line of code.

- Click the Use Step Filters button to change whether step filters should be used in the current Debug View.

- Once the debugging process has terminated, you can click the Remove Terminated Launches button to remove any terminated debug sessions from the list.

Views Provided During PHP, Web Page or URL Debugging

During the debugging process (for PHP Script, PHP Web Page or URL), various views will provide the following information:

- Debug View - Here you can control (stop, pause, and resume) the debugging process. You can also decide whether to step into, step over or step return (step out off) certain functions.
- Variables - Will display the various variables in your script.
- Breakpoints - Will display the breakpoints you have entered
- Parameter Stack - Will display the parameters through which functions are reached.

- Editor Window - Will display the code at the relevant sections, according to which line is selected in the Debug View window.
- Debug Output - Will show the textual output of the script. This will be updated as the debugging process continues.
- Browser Output - Will show the output of the script to a browser. This will be updated as the debugging process continues.
- Console View (External Link) - Displays any error and warning messages.
- Tasks View (External Link) - Displays tasks that were added to your script (if applicable).

Views Provided During JavaScript Debugging

During the debugging process (for JavaScript debugging), various views will provide the following information:

- Debug View [Debug Perspective] - Here you can control (stop, pause, and resume) the debugging process. You can also decide whether to step into, step over or step return (step out off) certain functions.
- Variables View [Debug Perspective] - Will display the various variables in your script.
- Breakpoints View [Debug Perspective] - Will display the breakpoints you have entered.
- Scripts View - Displays a list of available scripts. Double click a script to see it's code in an editor
- Editor Window - Will display the code at the relevant sections, according to which line is selected in the Debug View window.
- Internal Web Browser - Displays the code you are debugging in the Browser.
- Debug Output View [Debug Perspective] - Will show the textual output of the script. This will be updated as the debugging process continues.
- Browser Output View [Debug Perspective] - Will show the output of the script to a browser. This will be updated as the debugging process continues.
- Console View (External Link) - Displays any error and warning messages.
- Tasks View (External Link) - Displays tasks that were added to your script (if applicable).

Note:
See 'PHP Debug Perspective' for more on the views that will be displayed during Debugging.

Setting Up Remote Debugging

Before debugging on a server, when using PHP Web Page debugging, certain settings need to be configured to ensure that Zend Studio can communicate with your server.

To set up communication between Zend Studio and the server on which you are debugging:

1. Ensure the Zend Debugger is installed on your server.
   The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.
2. Ensure the machine on which your Zend Studio is installed is an allowed host for your debugger.
   The method for configuring this setting will depend on whether you have Zend Server, or the standalone Zend Debugger installed on your system.
   See Setting your Zend Studio to be an Allowed Host for more information.
3. In Zend Studio, configure your server according to the instructions under Adding Servers in the PHP Servers Preferences.
4. Ensure the correct settings are configured in your Debug Preferences and Installed Debuggers Preferences pages.
5. Ensure you have a dummy.php file in your remote server's document root.
6. If your server is situated behind a firewall or other security device, see Setting Up Tunneling for information on how to enable a communication tunnel.
   If you don't know whether your server is situated behind a firewall, contact your System Administrator.

Setting your Environment to be an Allowed Host

This procedure describes how to ensure the machine on which your Zend Studio is installed will be an Allowed Host for initiating your Debug and Tunneling session on the remote server.

The steps you need to follow will depend on whether you have Zend Server, or the standalone Zend Debugger installed on your server:

If Zend Server is Installed

To set your environment to be an allowed host if Zend Server is installed:

1. Open your Zend Server GUI.

> **Note:**
> This can be done from within Zend Studio by selecting the server on which you have configured your Zend Server integration from the drop-down list next to the Zend Server icon on the toolbar
> .

2. Go to the Server Setup | Debugger tab.
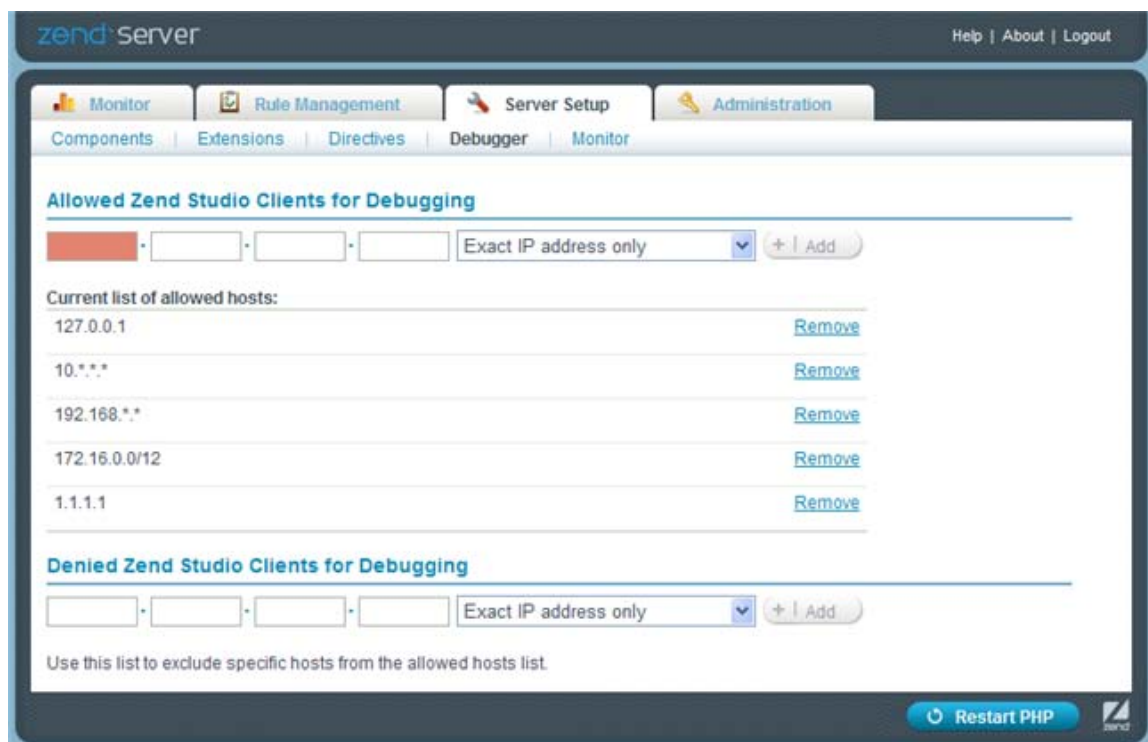
Allowed Hosts for Tunneling

3. Ensure the address of your Zend Studio is included in the Allowed Hosts sections.

   To add an address to the list:

   i. Under the Allowed Zend Studio Clients for Debugging category, enter the IP of the machine on which your Zend Studio is installed

   ii. Click **Add**.

   The IP Address is added to the Allowed Hosts list.

4. Ensure your Studio's IP address is not in the Denied Hosts list.

   If it is, click **Remove** next to the required address to remove it from the list.

5. Click the button to apply your settings.

If only the standalone Zend Debugger is installed on your server:

 To configure your debugger to allow your Zend Studio to debug:

1. Open your php.ini file.

2. Edit the zend_debugger.allow_hosts and zend_debugger.allow_tunnel (if necessary) parameters to include the IP address of the machine on which your Zend Studio is installed.

   e.g. zend_debugger.allow_hosts=127.0.0.1/32

   zend_debugger.allow_tunnel=127.0.0.1/32

3. Ensure the address is not in your zend_debugger.deny_hosts parameter list.

4. Set the Debug Server to expose itself to remote clients by setting the zend_debugger.expose_remotely parameter to Always.

   (e.g. zend_debugger.expose_remotely=always).

5. Save the file.

6. Restart your Web server for the settings to take effect.

Ensuring the Placement of dummy.php

In order for the remote server's debugger to communicate with Zend Studio, a file called dummy.php must be located in your server's document root.

With the default Zend Serverinstallation, a dummy.php file will have been automatically placed in your server's document root folder.
If you installed the standalone Zend Debugger, you must copy the dummy.php file from the Zend Debugger archive to your server's document root.

<div>

Note:

If you have set up a virtual host and it's document root is not pointed at the remote server's default document root, you will have to copy the dummy.php to the virtual host's document root in order to be able to debug on that virtual host.

</div>

You must also ensure that the Dummy File name is set to 'dummy.php' in Zend Studio.

To check your dummy file configuration in Zend Studio:

1. Open the Installed Debugger Preferences page by going to **Windows| Preferences | PHP | Debug | Installed Debuggers.**
2. In the Installed Debuggers list, select the Zend Debugger and click **Configure**.
   The Zend Debugger Settings dialog will open.
3. In the Dummy File Name setting, ensure 'dummy.php' is entered.

<div>

Note:

If you changed the name of the dummy file on the server, you must change this entry accordingly.

</div>

4. Click **OK**.

Managing Path Maps

These procedures describe how to manage Path Map settings in Zend Studio. Using Path Mapping allows Zend Studio to search for files which are called from a certain location on the server in a local location during PHP Web Page debugging /profiling.

Adding a Server Location Path Map

This procedure describes how to add a Path Map to a server so that files which are called from a certain location on the server will be searched for in a local location during PHP Web Page debugging /profiling. This will only apply when the 'use local copy' option is selected in the Advanced tab of the PHP Web Page debugging configuration).
See [Path Mapping](#) for more details.

To add a Path Map to a server:
1. Open the PHP Servers Preferences Page by going to **Window | Preferences** on the Menu Bar and selecting **PHP | PHP Servers** from the Preferences list.
2. Select the server on which you would like to create the Path Map and click **Edit**.
3. In the Edit Server dialog, select the Path Mapping tab.
4. Click **Add**.
5. The Add new Path Mapping dialog appears.
6. Enter the Server Path from which you would like to create the Path Map. Files called from this location will be searched for in the path specified below.
7. Select either the 'Path in Workspace' or 'Path in File System' option and click **Browse** to specify the location.

Edit Path Mapping

8. Click **OK**.

Your Path Map will be added to your server list.

Path Mapping Settings

The next time a file is called from the Path on Server, it will be searched for in the local location you have specified.

You can now manage your Path Map settings by Editing or Removing your Path Map.

> **Note:**
> Path Mapping can also be set automatically during Debugging / Profiling . See the Path Mapping topic for more details.

Adding a New Path Map for Importing a Zend Server Event File

This procedure describes how to add a Path Map to a server while importing a Zend Server Event File so that files which are called from a certain location on the server will be searched for in a local location during remote PHP Script debugging/profiling and PHP Web Page debugging/profiling. This will only apply when the 'use local copy' option is selected in the Advanced tab of the PHP Web Page debugging configuration).

To add a new Path Map in the Importing a Zend Server Event File wizard:

1. Open the Import Wizard go to **File | Import | Zend Server | Zend Server Event File**.

2. Click **Next**.

   The "Import Zend Server Event File" dialog opens.

3. In the "From File" text field, browse to the location of your Zend Server Event File and click **Next**.

The "Server path mapping settings" dialog opens.

4. Select the server to associate with from the "Associate this trace with server" drop down menu.

5. Click **Add**.

6. An Add a New Path Map dialog appears.

7. Enter the Server Path from which you would like to create the Path Map. Files called from this location will be searched for in the path specified below.

8. Select either the Path in Workspace or Path in File System option and click **Browse** to specify the location.

9. Click **OK** to add your path map to your server list and return to the Server Path Map Settings dialog.

   The next time a file is called from the Path on Server, it will be searched for in the local location you have specified.

You can now manage your Path Map settings by Editing or Removing your Path Map.

See Importing a Zend Server Event File for information on how to continue importing a Zend Server Event File once your Path Map settings are configured.

Editing or Removing Your Path Map

This procedure describes how to edit or remove your Path Map. Editing your Path Map allows you to change the location on the server that will be searched for in a local location (or to change the local location that will be searched) during PHP Web Page debugging /profiling.

Before editing or removing a Path Map you must first Add a Server Location Path Map or Add a New Path Map while Importing a Zend Server Event File.

- To edit your Path Map select the Path Map you would like to edit, click **Edit** and change the relevant information.

- To remove your Path Map select the Path Map you would like to delete and click **Remove**.

Setting Up Tunneling

To establish a tunneling connection for remote debugging and Zend Server integration, the following tasks need to be performed:

If you have followed the instructions under <u>Setting Up Remote Debugging</u>, you will have already performed some of these tasks.

1. <u>Ensure you have a dummy.php file</u> in your remote server's document root.
2. <u>Ensure your Zend Studio is an allowed host</u> for your server debugger. This can be done through Zend Server or through your php.ini file.
3. <u>Configuring you Tunneling Connection in Zend Studio</u> - From the PHP Servers Preferences page (Window | Preferences | PHP | PHP Servers).
4. <u>Activate the tunnel</u> - By selecting your Tunneling server from the list next to the Tunneling icon on the toolbar.

Configuring Zend Server to Auto Detect Zend Studio Settings

This procedure describes how to configure Zend Server so that Zend Studio's settings are automatically detected during the Debugging/ Profiling of Zend Server events.

To establish a communication tunnel between Zend Server and Zend Studio:
1. Open your Zend Server GUI.
2. Go to the **Server Setup | Monitor tab**.
3. In the Zend Server Settings section, configure the following:
 ▪ Auto detect the Zend Studio Client Settings - Set to 'On' to inform Zend Server of the method of connection to Zend Studio. This allows Zend Server to detect your Zend Studio Debug settings.
4. Click **Save**.

Setting Up a Tunneling Server

This procedure describes how to configure a server to allow Tunneling in Zend Studio.

You can configure several servers to allow tunneling.

To configure your server for Tunneling with Zend Studio:

1. Open the PHP Server Preferences page by going to **Window | Preferences | PHP | PHP Servers** from the Menu Bar.

2. Click **New** to define a New Server (or Edit if the server has already been defined).

3. Give the server a unique name and enter the URL of the server to which you would like to create a tunnel.
   Click **Next** to continue or go to the next Tab.

4. You can ignore the Path Mapping option. If necessary, you will be automatically prompted to define path mapping during debugging and profiling sessions. See "Path Mapping" for more details.
   Click **Next** to continue or go to the next Tab.

5. If you want to enable Zend Server integration mark the Enable Zend Server Integration checkbox.
   See Configuring Zend Server Settings in Zend Studio for more information.

6. Click **Next** to continue or go to the next Tab.
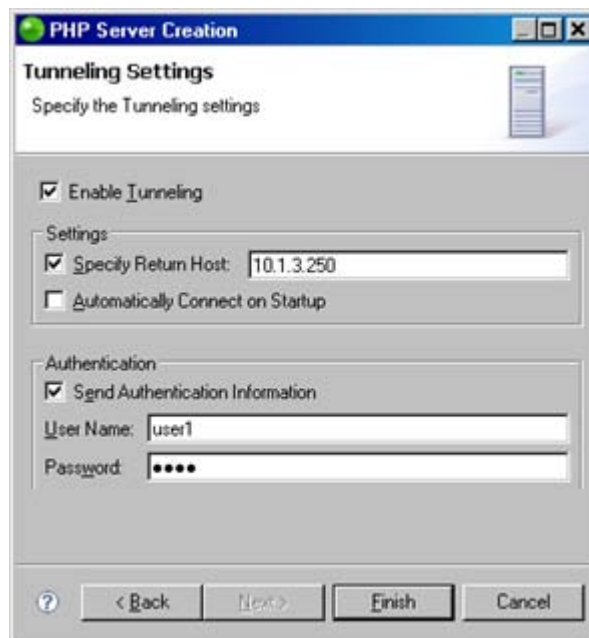
Tunneling Settings

7. In the Tunneling Settings section, check the "Enable Tunneling" option.

8. In some system configurations, a dedicated tunneling server is used which allows debug connectivity between Zend Studio and the server which is being debugged. In this case, unmark the 'Specify Return Host' checkbox and enter the IP address of the tunneling server to which the tunneling connection will be opened.

9. If your Web server requires HTTP authentication, enter your User Name and Password in the Authentication category. Zend Studio sends the authentication information in the header.

> Note:
> This assumes the user account is set up on the Web server

10. Click **Finish** or **OK**.

Your server is now configured to allow tunneling in Zend Studio.

The next step is to [Activate your Tunneling Connection](#).

Activating Tunneling

Once you have configured all the necessary settings in Zend Studio and on your server, you can activate your Tunnel connection.

This procedure describes how to open a tunnel between Zend Studio and your remote server.

To activate Tunneling in Zend Studio:

1. Click the arrow next to the Tunneling icon on the toolbar and select the server which you configured for tunneling.

Tunneling server list

2. The tunneling icon will turn green to show that a tunneling server is connected:

You can now debug/profile on the selected remote server.

> **Note:**
> Several Tunneling sessions can be configured in Zend Studio. Therefore, if the debug session is not working, check to see that the Tunnel to the correct server is connected by clicking on the drop-down arrow next to the Tunneling Icon and verifying that the name of the connected server is correct.

Troubleshooting the Communication Tunnel

If Zend Studio is unable to connect to the target server, you will get an error message with the response from the server. The table below describes the most likely causes and a recommended action for successfully establishing a connection with the target server.

Possible Cause	Recommended Action
The server address you entered is incorrect	Enter the correct server information in the Tunneling Settings dialog.
HTTP authentication is required	Enter authentication information in the Tunneling Settings dialog box; then click the 'Send authentication information' checkbox.
The dummy file content or location on the server is incorrect	The dummy file on the server side was either changed or does not exist. You will need to ensure that the correct dummy file with the correct content is placed in the correct directory on the target server. The correct dummy file is created and located properly as part of the installation procedure. The problem here is post-installation.
You are not allowed to connect with the server via the communication tunnel	You must have tunneling permissions in the php.ini file. Make sure that the zend_debugger.allow_tunnel variable is properly configured. For any other cause, or additional information, use one of our support options.

Profiling Files and Applications

The following Profile functionality is available in Zend Studio:

- [Locally Profiling a PHP Script](#) - Profile a PHP Script using Zend Studio's internal debugger

- [Profiling a PHP Web Page](#) - Profile PHP files on a remote server

- [Profiling a URL](#) - Profile a URL

- [Profiling Using the Zend Debugger Toolbar](#) - Profile directly from your web browser.

Once a Profile session has been launched, the PHP Profile perspective is used to view and analyze the results of the profiling process.

See the [PHP Profile Perspective](#) topic for more on the information displayed once a Profile session has been run.

Locally Profiling a PHP Script

This procedure describes how to Profile a PHP Script from your workspace using Zend Studio's internal debugger:

To locally Profile a PHP Script:

1. Click the arrow next to the Profile button [icon] on the toolbar and select **Open Profile Dialog** -or- from the main menu go to **Run | Open Profile Dialog** -or-right-click in PHP Explorer view and select **Open Profile Dialog**.
2. A Profile dialog will appear.
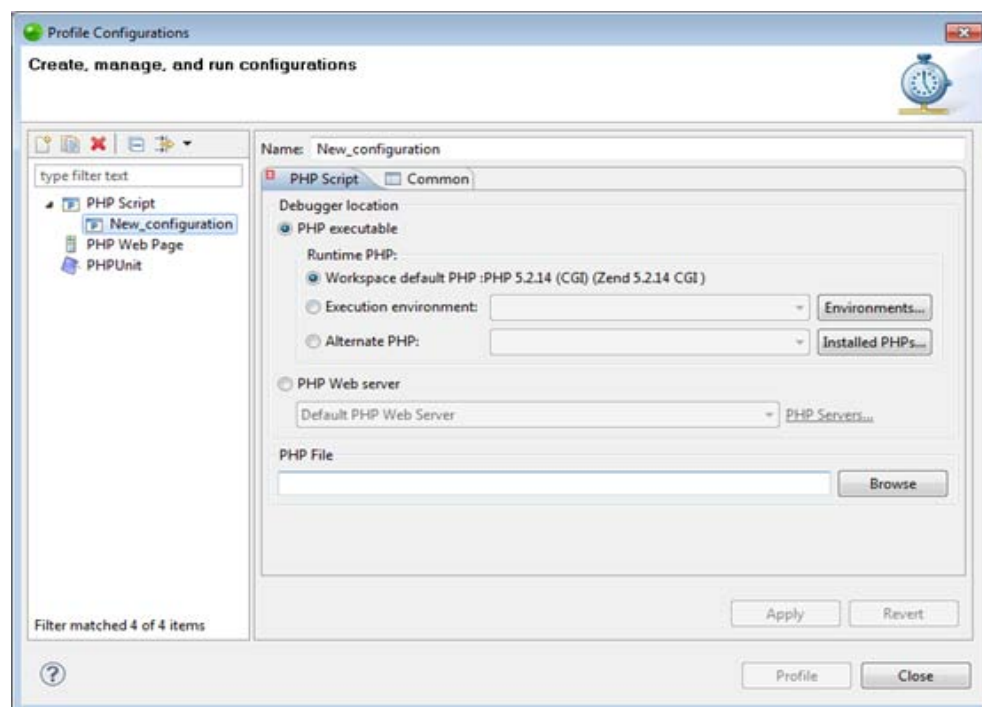
3. Double-click the PHP Script option to create a new Profile configuration.
4. Enter a name for the new configuration.
5. Select from the following options:
   - Workspace default PHP - Select to use the default PHP and execution environment defined in the [Execution Environments Preferences](#) page.
   - Execution environment - Select the execution environment you would like to use for your debug configuration from the dropdown list.
     Click **Environments** to change the settings in the [Execution Environments Preferences](#) page.

- Alternate PHP - Select to use another PHP which can be selected from the Installed PHP's list.

6. Enter your PHP file in the "PHP File" text field, or click **Browse** and select your file.

7. Marking the 'Display debug information when running' checkbox will cause debug views to be displayed.

8. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.

9. Click **Apply** and then **Profile**.

10. A confirmation dialog will be displayed asking whether you want to open the Profiling Perspective.

    Click **Yes**. (If you would like the Profiling Perspective to open by default in the future, mark the 'Remember my decision' checkbox.)

The Profiling Perspective will open, displaying the Profiling Monitor window with various Profiling views. See PHP Profile Perspective for more on the information displayed once a profile session has been run.

| **Note:** |
|---|
| If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment. |

Profiling a PHP Web Page

This procedure describes how to profile whole applications, projects, files or collections of files that are already on the server.

To profile a PHP Web Page:

1. Click the arrow next to the Profile button on the toolbar and select **Open Profile Dialog** -or- go to **Run | Open Profile Dialog** from the main menu -or- right-click in PHP Explorer view and select Open Profile Dialog.
2. A Profile dialog will appear.
3. Double-click the PHP Web Page option to create a new Profile configuration.

Profile PHP Web Page Configuration

4. Enter a name for the new configuration.

5. Select your server from the list.

   If you have not yet configured your server, click **New**. The PHP Server Creation Wizard will open.

   Configure your server by following the instructions on '<u>adding a new server</u>' under the PHP Servers Preferences page.

   For more information on configuring the communication between Zend Studio and your remote server, see <u>Setting Up Remote Debugging</u>.

6. Under PHP File, click **Browse** and select your 'debug target' file (the file from which the profiling process will start.)

7. For further profiling options, select the Advanced tab, which has the following options:

   - Open in Browser - Mark if you would like the application to be displayed in Zend Studio's internal browser

   - Source Location - Choose whether the source files used during this session will be taken from the Server or from a local copy (if a local copy is not available, files will be taken from the server according to the search mechanism detailed in the <u>Path Mapping</u> topic).

8. Click **Apply** and then **Profile**.

9. Click **Yes** if asked whether to open the PHP Profile Perspective. (If you would like the Profiling Perspective to open by default in the future, mark the 'Remember my decision' checkbox.)

The Profiling Perspective will open, displaying the Profiling Monitor window with various Profiling views. See the <u>PHP Profile Perspective</u> for more on the information that will be displayed once a profile session has been run.

Profiling a URL

This procedure describes how to profile a URL.

<table>
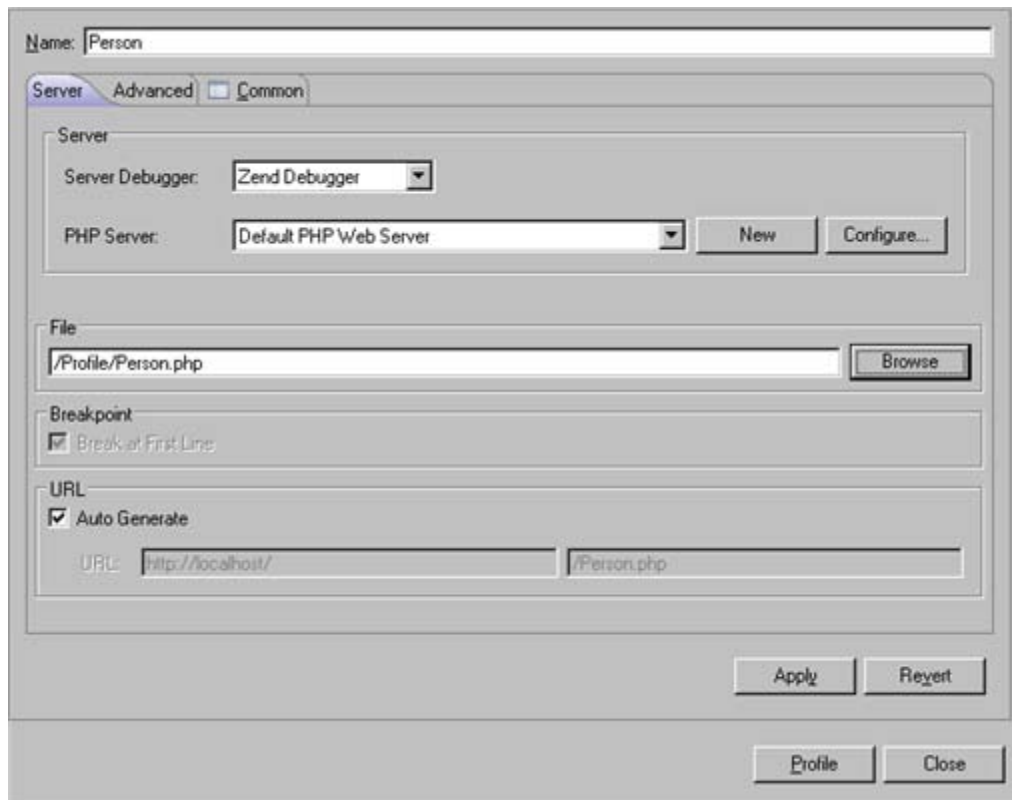<tr><td>Note:</td></tr>
<tr><td>Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.
The Zend Debugger comes bundled with Zend Server, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.</td></tr>
</table>

To Profile a URL:

1. Click the profile URL button on the main toolbar -or- go to **Run | Profile URL**.

2. The Profile URL dialog will appear.
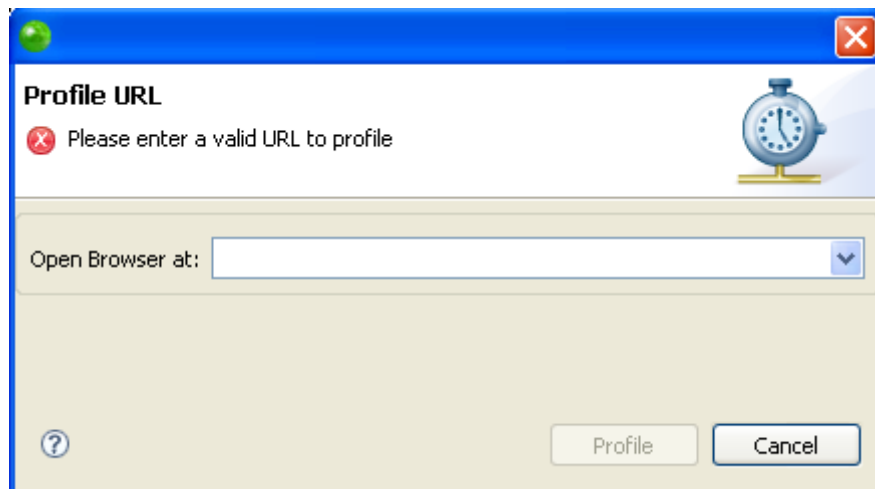
Profile URL dialog

3. In the 'Open Browser at:' field, enter the URL of the page that should be profiled.

4. Click **Profile**.
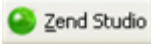
The Profile Perspective will open with a number of views detailing information about the profiling process. See the PHP Profile Perspective for more on the information that will be displayed once a profile session has been run.

Profiling Using the Zend Browser Toolbar

This procedure describes how to Profile using the [Zend Browser Toolbar](#).

To Profile using the Zend Browser Toolbar:

1. Ensure the Zend Browser Toolbar is installed on your browser.

2. Open your browser and browse to the page from which you would like to start profiling.

3. Ensure Zend Studio is open. You can open it by clicking the Zend Studio button in the toolbar.

4. Click the button on the Zend Browser Toolbar to profile the page currently displayed in the browser.

The relevant Profile session will be launched in Zend Studio.

If Zend Studio is not open, you will be prompted to open it before the profiling session is launched.

Managing PHP Libraries

PHP Libraries allow you to create and maintain an external code library.

Enabling PHP Libraries in your project allows libraries to be referenced by the project and makes the elements within these resources available for operations such as Content Assist and Refactoring.

Note:
You may change the order your user libraries are in by using the **Up** and **Down** buttons. The order in which the libraries are arranged in this page defines the order they are available in Zend Studio's functionality, such as content assist.

The PHP Libraries Preferences page allows you to do the following:

- Add PHP Libraries
- Add External Folders to PHP Libraries
- Export PHP User Libraries
- Import PHP User Libraries
- Edit PHP User Libraries
- Edit PHP Library Components or Folders
- Remove a PHP Library or Library Folder

The PHP Libraries Preferences page is accessed from **Window** | **Preferences** | **PHP** | **PHP Libraries**.

Note:

Once you have added a user library in the PHP preferences page, you must also add it to your PHP

Include Path of the project in which you would like to have it available. For more information see

Configuring a Project's PHP Include Path.

Adding a PHP Library

This procedure describes how to add a user library to Zend Studio. Including user libraries in your project or environment saves you time in writing and debugging code, as you are re-using debugged code.

To add an additional PHP Library to your project:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page click **New**.

   The "New User Library" Dialog will open.
3. In the "New User Library" dialog, enter the name of your user library
4. Select the "Add to environment" checkbox if you would like this library to be added to your entire environment instead of a specific project.
5. To apply changes click **OK**.

A new empty library will be added to the list. Next to the name in brackets indicates if it is shared by the environment or only related to a project.

Adding a PHP library creates a place folder in which you can place external files that contain pre-written code. For more information see Adding External Folders to PHP Libraries.

Note:
Once you have added a user library in the PHP preferences page, you must also add it to your PHP Include Path of the project in which you would like to have it available. For more information see Configuring a Project's PHP Include Path.

Adding External Folders to PHP Libraries

This procedure describes how to add external folders to a user library. This allows you to compile a user library using folders from varying places on your computer. In order to add an external folder, you must have already created a user library. For more information see Adding a PHP Library.

If you are interested in using compressed files, click the **Add ZIPs....** button and select a file instead of following the procedure below.

To add an external folder to your user library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page click **Add External folder...**
   The "External folder Selection" dialog will open.
3. Select a folder in the "External folder Selection" dialog or create a new folder by clicking **Make New Folder**.
4. To apply changes click **OK**.

The folder will be added to the PHP library.

The folders that are included in the user defined libraries allow you to control their access rules. The access rule default setting is "No restrictions." See Defining Access Rules below for more information.

Defining Access Rules

This procedure describes how to define access rules. Access rules allow you to control where your libraries will be available. From here you can control what code completion options will be available in features such as Content Assist on a project level.

In order to define access rules you must have an external PHP folder added to your project. For more information Adding External Folders to PHP Libraries.

To define an access rule for your user defined library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page, double click "Access rules:" located in the hierarchal tree below your added folder. The default setting is "No restrictions." The "Type Access Rules" dialog will open.

3. Click **Add.**

The "Add Access Rule" dialog will open.
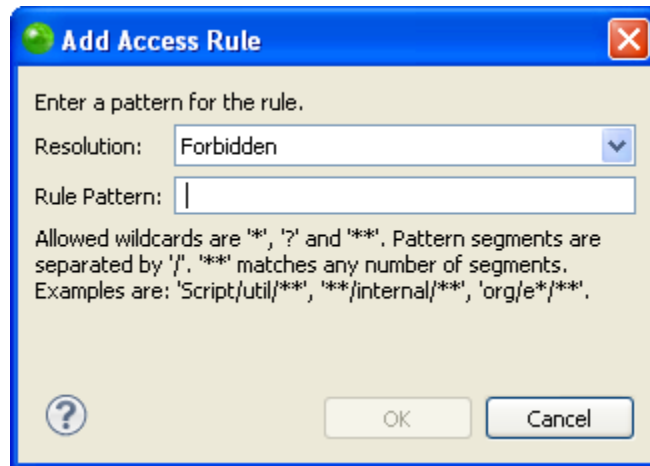
4. From the "Resolution:" drop down menu select one of the following:

- Forbidden - This will make the file completely unaccessible in the relevant project.
- Discouraged - This will make the file available but not highly suggested.
- Accessible - This will allow the project to access the file and its contents freely.

4. In the "Rule Pattern" field, insert a rule pattern to define a pattern for the rule. Use wildcards to create a pattern. The accepted wildcard values are:

- '*'
- '**'
- '?'

This will automatically apply the rule to any of the files that match the pattern.

To apply changes click **OK**.

5. In the "Type Access Rules" dialog, you also have the option to edit, remove, or change the order of your defined access rules by clicking the buttons. Before a file is allowed access into your project, it will first confirm that none of the access rules apply to it. These rules are checked in the order they are placed here.

Your PHP Library now has an access rule defined in the hierarchal tree in the PHP Libraries Preferences page. You may edit a file's access rule at any time by repeating this procedure.

Importing PHP User Libraries

This procedure describes how to import existing user libraries that are on the disk or in a repository. This allows you to take an already built library and use it in your project, as well as share a library with other users of the same repository. Importing a library will only import a description of the library in .xml format, and will not include any of the library's content. Before importing a user library, you must first have access to an exported user library. For more information see Exporting PHP User Libraries.

To import a user library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page click **Import...**
   The "Import User Libraries" dialog will open.
3. To choose where you would like to import your library from, fill in the "File location:" text field with the URL or click **Browse...** and select the location.
4. Select the libraries you would like to import from the options in the "Libraries contained in the selected file:" box, or press **Select All** or **Deselect All**.
5. To apply changes click **OK**.
   Your library's description in .xml format has now been imported into Zend Studio.

If the library of the user who imports it is stored in the same location on the disk as the user who exported it, Zend Studio will automatically find the libraries content and store it accordingly. If the user library is stored in a different place for the two users, the **Edit...** button allows you to replace the location URL. See Editing PHP User Libraries for more information.

Exporting PHP User Libraries

This procedure describes how to export user libraries, making them accessible to whoever has access to the repository where it is stored. Exporting a user library will only export a description of the library in .xml format, and will not include any of the library's content.

To export a user library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page select the user library you would like to export and click **Export...**
   The "Export User Libraries" dialog will open.
3. Select the library you would like to export by clicking the check box beside it. You may also use the **Select All** or the **Deselect All** buttons.
4. To choose where you would like to export your library to, fill in the "File location:" text field with the URL, or click **Browse...** and select the location.
5. To apply changes click **OK**.

Your user library has now been exported to the location you specified.

You may now import the exported user libraries from any location that has access to the location in which it is stored. For more information see Importing PHP User Libraries.

Editing PHP Library Components or Folders

This procedure describes how to edit user library folders. Edit a user library folder when you have imported a folder who's original location is different from its content's location on your disk. For more information see [Importing PHP User Libraries](#).

To edit a library component:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page select the user library folder to edit and click **Edit...**

   The "Edit External folder" dialog will open.
3. Select the folder that contains the user library content, or create a new folder by clicking **Make a New Folder**.
4. To apply changes click **OK**.

You have now edited your user library's URL which will import the content from the proper location on the disk.

Note:

If the imported location of the selected file you are editing is different than the new location, the new location will overwrite it.

Editing PHP User Libraries

This procedure describes how to edit user libraries. You will need to edit a user library when you would like to change it's name. Changing a library's name allows you to create descriptive differentiations between your libraries. In order to edit a library, there must be libraries available. For more information see Importing PHP User Libraries or Adding a PHP Library.

To edit a user library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page select the user library you would like to edit and click **Edit...**
   The "Edit User Library" dialog will open.
3. Enter the user library name in the "User Library Name:" text field.
4. In order to add the given user library to your environment you must select the "Add to environment" box. Otherwise your library will be added on a project level.
5. To apply changes click **OK**.

You have now edited your user library. To edit a folder within your library see Editing PHP Library Components or Folders.

Removing a PHP Library or Library Folder

This procedure describes how to remove a user library or library folder from Zend Studio. Removing a user library or library folder means that its contents will no longer be available in Zend Studio, including in it's functionality such as Content Assist and Refactoring.

Important Note:
If your PHP user library or library folder is associated with specific projects, removing it will delete it from the projects as well.

To remove a PHP library/library folder:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page select the user library or library folder to remove and click **Remove**. (Removing a user library will remove all of the folders within it as well.) Your user library or library folder is deleted.
3. To apply changes click **OK**.

Your user library or library folder has been removed. If you would like to import a library see Importing PHP User Libraries. To add a folder to an existing library, see Adding External Folders to PHP Libraries.

Configuring a Project's PHP Include Path

Adding external resources to a project's include path makes resources outside the project available for operations such as debugging, refactoring and content assist. See PHP Include Paths for more information.

This procedure describes how to configure your project's Include Path.
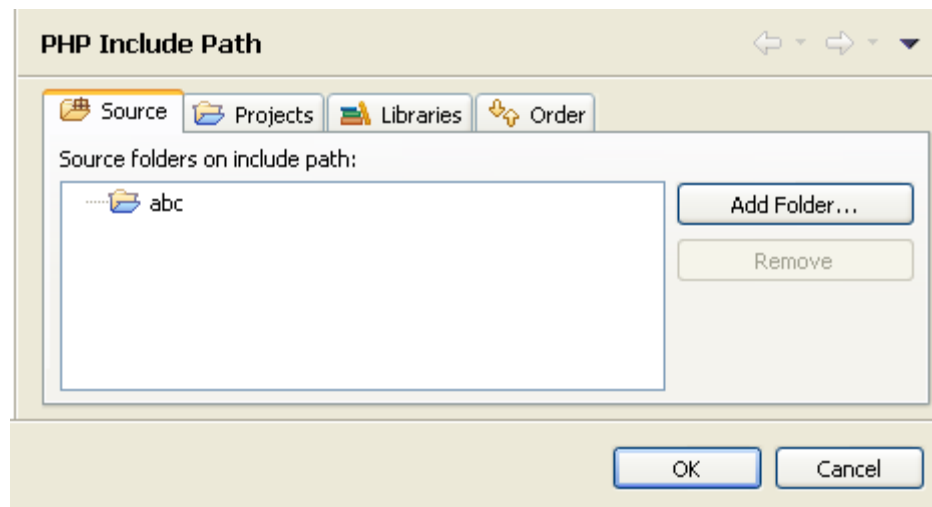
To configure your project's Include Path:

1. In PHP Explorer view, right-click the required project and select **Include Path | Configure Include Path...**

   -Or- right-click the project and select **Properties | Include Path**.

   The project's PHP Include Path properties page will appear.

From this page you can configure the following:

To configure source folders on your Include Path:

i. Select the Source tab.

   By default, all folders which are on your Include Path will be added to your Build Path.

> **Note:**
> If your project root is on your Include Path, all folders and resources contained within your project will also be on the Include Path. To add only certain folders, remove the project root from the Include Path and add only the required folders.

ii. To remove a folder, select it and click **Remove**.

iii. To add a folder from your project, click the **Add Folder** button and select the required folder.

A prompt will display, asking whether you would also like to add the selected folder(s) to your [Build Path](#).

It is recommended that the resources on your Build Path match the resources on your Include Path.

Click **Yes** to add the folder(s) to your Build Path or **No** for your Build Path to not be affected.

To add another project from your workspace to your Include Path:

i. Select the "Projects tab".

ii. Click **Add**.

The Required Project Selection dialog appears.

iii. Select the projects you would like to add and click **OK**.

The selected project(s) will be added to your project's Include Path.

To add a library to your Include Path:

i. Select the "Libraries tab".

ii. Click **Add Library**.

The Add Library dialog appears.

iii. Select the required Library (if available) and click **OK**.

To add external folders to your Include Path:

i. Select the "Libraries tab".

ii. Click **Add** External Folder.

The Add Include Path dialog appears.

iii. Browse to and select the required folder.

iv. Click **OK**.

The folder will be added to your project's Include Path.

> **Note:**
> The library is a read-only file and will not be available for editing.

To configure the order of elements on your Include Path:

i. Select the "Order tab". This determines the order in which resources will be searched for in require/include calls.

See [Include Paths](#) for more information on the order for which files are searched for.

If applicable, it is recommended that elements appear in the same order as they do in your php.ini.

ii. If necessary, rearrange the order of the entries. To do so, select an element and click **Up** or **Down** to move it in the list.

2. Once all the elements are added and are in the right order, click **OK**.

All the selected elements will be added to the project's include path.
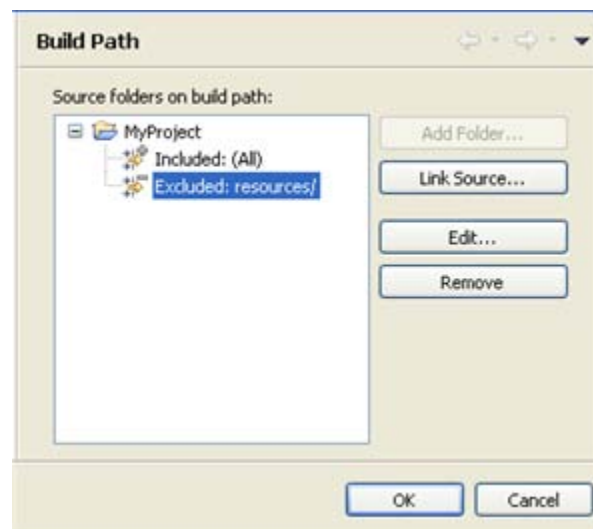
Configuring a Project's PHP Build Path

By configuring your project's PHP Build Path, you can select resources which will be included or excluded from the Build process.

The project's PHP Build Path can be configured through the project's Build Path properties page, accessed by right-clicking the required project and selecting Build Path | Configure Build Path.

Build Path properties page

This page allows you to add folders and resources to your Build Path and configure rules for including / excluding certain resources in the project.

You can configure your Build Path according to the options listed below.

Once you have made changes, click OK to save.

Configuring Inclusion/Exclusion Patterns for the Project

By configuring inclusion/exclusion patterns for the Build Path, you can select to include or exclude all resources in your project which match a defined pattern.

To include/exclude resources from the Build Path:

1. In the Build Path Properties page, expand the node next to the folder whose Inclusion/Exclusion pattern you want to configure and click **Edit**.

   -Or- In PHP Explorer view, right-click the folder whose Inclusion/Exclusion pattern you want to configure and select **Build Path | Configure Inclusion / Exclusion Filters**.

   The Inclusion and Exclusion patterns dialog is displayed.

2. Click **Add** next to the Inclusion or Exclusion patterns panes.

   The Add Inclusion/Exclusion Pattern dialog is displayed.

3. Enter or select the required resource(s) or pattern to include / exclude and click **OK**.

4. Click **Finish**.

All resources in the project which match an inclusion pattern but do not match an exclusion pattern will be added to the Build Path.

Configuring Different Inclusion/Exclusion Patterns for Folders within Your Project

You can configure different inclusion/exclusion rules for child folders within a selected directory. However, you will need to exclude the selected child folder from the parent directory's Build Path and add it as a separate Build Path source folder. This is done to resolve any conflicts which may arise from configuring one pattern for the parent folder and another for the child.

Source folders can be created and/or added from the PHP Build Path properties page or from PHP Explorer view.

**To add a directory as a separate source folder:**

From the PHP Build Path Properties page:

1. Click **Add Folder**.

   The Source Folder Selection dialog will display.

2. If necessary, click **Create New Folder** to create a new folder.

3. Select the required folder and click **OK**.

4. The folder will be added as a separate source folder in the Build Path list.

> **Note:**
> In order to overcome the nesting conflict, you should exclude the folder you have just added from the parent folder's Build Path. See 'Configuring Inclusion/Exclusion Patterns', above, for information on how to exclude the folder and all resources contained within it.
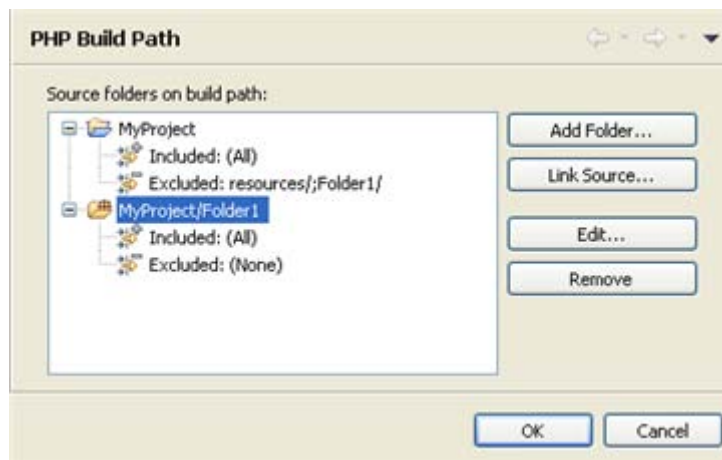
PHP Build Path properties - Add Folder

<u>From PHP Explorer view:</u>

1. Right-click the project and select **Build Path | New Source Folder**.

   The New Source Folder dialog will be displayed.

New Source Folder dialog

2. Enter the name for the new folder in the Folder name field.

3. Adding a folder to a project's Build Path could cause a conflict between the inclusion/exclusion patterns configured for the project root and those configured for the selected folder.

In order to avoid these nesting conflicts, select one of the following options:

- Replace existing project source folder entry to solve nesting - This will remove your parent directory as a source folder from the Build Path configuration and replace it with the currently selected folder.
- Update exclusion filters in other source folders to solve nesting - This will exclude the currently selected folder from the parent directory's Build Path but will add it as a separate entity so that different inclusion/exclusion patters can be configured for it.

4. Click **Finish**.

You can now configure a pattern for including/excluding resources for the source folder by following the instructions under '[Configuring Inclusion/Exclusion Patterns](#)', above.

Adding External Source Folders to the Build Path

You can add an external source folder as a link to your project which will be scanned during the Build process.

To add an external source folder to the project's Build Path:

1. In the Build Path Properties page, click the 'Link Source...' button.

   -Or- In PHP Explorer view, right-click the project and select **Build Path | Link Source**.

   The Link Source dialog is displayed.

Link Source dialog

2. Click the 'Browse..' button and browse to the location of the folder you want to add to the Build Path.

3. Alternately, click **Variables** and select the variable which points to the required resource. The Folder name field will have been automatically populated with the name of the original folder.

4. Edit this entry if required.

5. Adding a folder to a project's Build Path could cause a conflict between the inclusion/exclusion patterns configured for the project root and those configured for the selected folder.
    In order to avoid these nesting conflicts, select one of the following options:

    ▪ Replace existing project source folder entry to solve nesting - This will remove your parent directory as a source folder from the Build Path configuration and replace it with the currently selected folder.

    ▪ Update exclusion filters in other source folders to solve nesting - This will exclude the currently selected folder from the parent directory's Build Path but will add it as a separate entity so that different inclusion/exclusion patters can be configured for it.

    ▪ Ignore nesting conflicts - The selected folder will be added as a separate entity to the Build Path list but will need to be manually excluded from the project root's Build Path to avoid nesting conflicts.

6. Click **Finish**.

The folder will be added to your Build Path and displayed in the Build Path list.

You can now configure a pattern for including/excluding resources for the linked folder by following the instructions under '[Configuring Inclusion/Exclusion Patterns](#)', above.

Managing Path Maps

These procedures describe how to manage Path Map settings in Zend Studio. Using Path Mapping allows Zend Studio to search for files which are called from a certain location on the server in a local location during PHP Web Page debugging /profiling.

Adding a Server Location Path Map

This procedure describes how to add a Path Map to a server so that files which are called from a certain location on the server will be searched for in a local location during PHP Web Page debugging /profiling. This will only apply when the 'use local copy' option is selected in the Advanced tab of the PHP Web Page debugging configuration).

See Path Mapping for more details.

To add a Path Map to a server:

1. Open the PHP Servers Preferences Page by going to **Window | Preferences** on the Menu Bar and selecting **PHP | PHP Servers** from the Preferences list.
2. Select the server on which you would like to create the Path Map and click **Edit**.
3. In the Edit Server dialog, select the Path Mapping tab.
4. Click **Add**.
5. The Add new Path Mapping dialog appears.
6. Enter the Server Path from which you would like to create the Path Map. Files called from this location will be searched for in the path specified below.
7. Select either the 'Path in Workspace' or 'Path in File System' option and click **Browse** to specify the location.

Edit Path Mapping

349

8. Click **OK**.

Your Path Map will be added to your server list.

Path Mapping Settings

The next time a file is called from the Path on Server, it will be searched for in the local location you have specified.

You can now manage your Path Map settings by <u>Editing or Removing your Path Map</u>.

> **Note:**
>
> Path Mapping can also be set automatically during Debugging / Profiling . See the <u>Path Mapping</u> topic for more details.
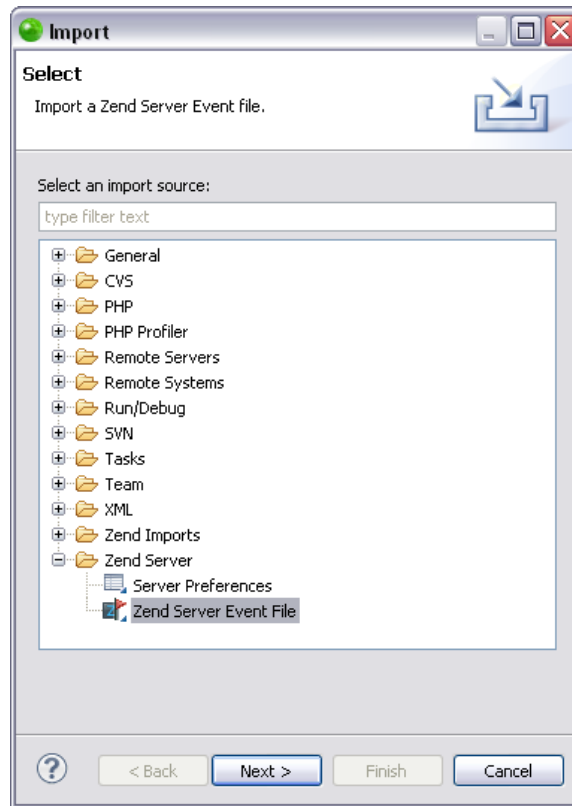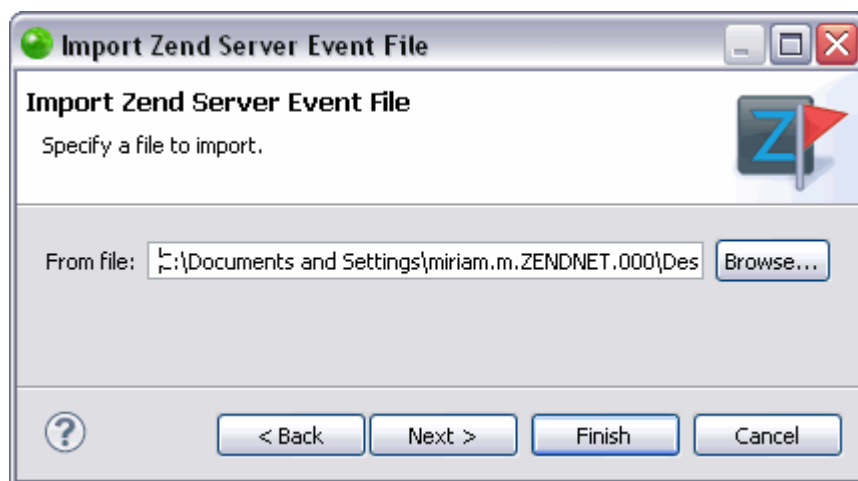
Adding a New Path Map for Importing a Zend Server Event File

This procedure describes how to add a Path Map to a server while importing a Zend Server Event File so that files which are called from a certain location on the server will be searched for in a local location during remote PHP Script debugging/profiling and PHP Web Page debugging/profiling. This will only apply when the 'use local copy' option is selected in the Advanced tab of the PHP Web Page debugging configuration).

To add a new Path Map in the Importing a Zend Server Event File wizard:

1. Open the Import Wizard go to **File | Import | Zend Server | Zend Server Event File**.

2. Click **Next**.

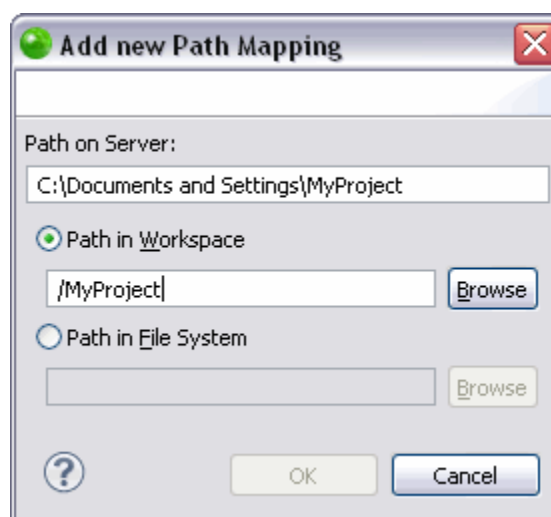   The "Import Zend Server Event File" dialog opens.

3. In the "From File" text field, browse to the location of your Zend Server Event File and click **Next**.

The "Server path mapping settings" dialog opens.

4. Select the server to associate with from the "Associate this trace with server" drop down menu.

5. Click **Add**.

6. An Add a New Path Map dialog appears.

7. Enter the Server Path from which you would like to create the Path Map. Files called from this location will be searched for in the path specified below.

8. Select either the Path in Workspace or Path in File System option and click **Browse** to specify the location.

9. Click **OK** to add your path map to your server list and return to the Server Path Map Settings dialog.

   The next time a file is called from the Path on Server, it will be searched for in the local location you have specified.

You can now manage your Path Map settings by Editing or Removing your Path Map.

See Importing a Zend Server Event File for information on how to continue importing a Zend Server Event File once your Path Map settings are configured.

Editing or Removing Your Path Map

This procedure describes how to edit or remove your Path Map. Editing your Path Map allows you to change the location on the server that will be searched for in a local location (or to change the local location that will be searched) during PHP Web Page debugging /profiling.

Before editing or removing a Path Map you must first Add a Server Location Path Map or Add a New Path Map while Importing a Zend Server Event File.

- To edit your Path Map select the Path Map you would like to edit, click **Edit** and change the relevant information.

- To remove your Path Map select the Path Map you would like to delete and click **Remove**.

Using PHPUnit Testing

The following tasks will guide you through the process of creating, running and reporting on PHPUnit Test

Cases and Suites:

- [Creating a PHPUnit Test Case](#)
- [Running a PHPUnit Test Case](#)
- [Creating a PHPUnit Test Suite](#)
- [Running a PHPUnit Test Suite](#)
- [Reporting on PHPUnit Test Results](#)

Creating a PHPUnit Test Case

This procedure describes how to create a PHP Unit test case.

Zend Studio will automatically create test case files which can be run in order to check the functionality of your code.

You must first create a file containing a class with functions which will be tested when the PHPUnit Test case is run.

To create a PHPUnit Test Case:

1. In PHP Explorer view, right-click the file containing the classes you would like to test and select **New | Other | PHP | PHPUnit | PHPUnit Test Case**.

   The PHPUnit Test Case dialog will open, with relevant information already entered into the various fields.

   Note that a new file will be created called "FileName"Test.php
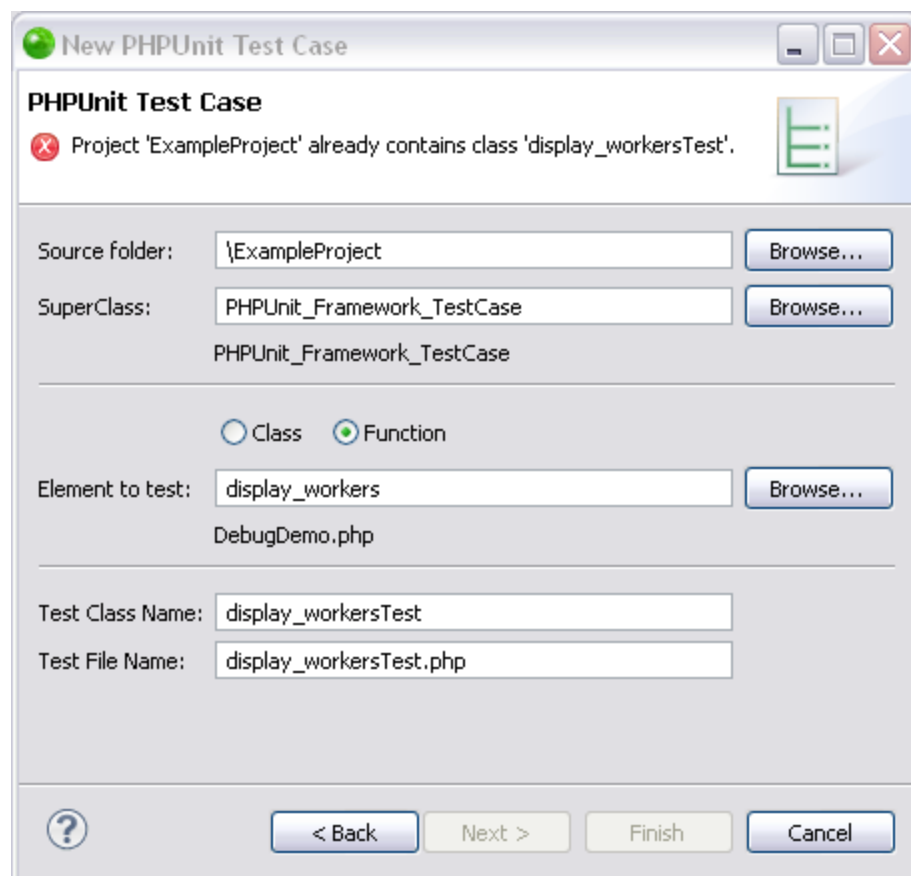
New PHPUnit Test Case dialog

2. A SuperClass is a class from which the new PHPUnit Test Case will inherit functionality (e.g. setup and constructors). If necessary, click Browse next to the 'SuperClass' field to select a different PHPUnit Framework SuperClass.

3. Click **Browse** next to the 'Element to test' field to select the Class or Function which will be tested in the new PHPUnit Test Case.

4. If this is the first PHPUnit Test created for the project, a warning will appear stating that the PHPUnit is not on the include path of your project.
   To add it to the include path, click the underlined "Click here" link at the bottom of the dialog screen. This will enable PHPUnit Content Assist options in the PHPUnit Test.
   Once it has been clicked, the link and the warning message will disappear.

5. Click **Finish** to create your test case.

6. The new test file, containing tests for the selected elements, will be added to your project.
   Note that all relevant functions in the original class will have a corresponding test function in the test file.
   However, test functions will have been created with no parameters.

7. Before you can run your test file, you must create relevant tests and parameters for each of your functions, depending on the results you expect to see when the function is run. For each function, write a test with demo input parameters and the expected result. When the test is run, Zend Studio will insert these parameters into your original file's functions to see if the result is as expected.

Once you have completed the file by creating relevant test functions and inserting parameters, your PHPUnit test case is ready to be run.

Running and Debugging a PHPUnit Test Case

This procedure describes how to run a PHPUnit Test Case and how to analyze the results.

Before running a PHPUnit Test Case, one needs to be created by following the instructions under Creating a PHPUnit Test Case.
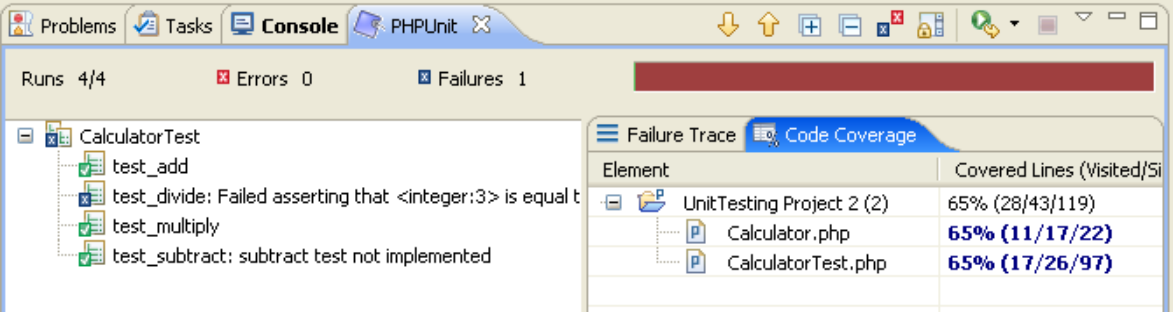
Running a PHPUnit Test Case

To Run a PHPUnit Test Case:

1. Open your PHPUnit Test Case file in the editor.

2. To run the PHPUnit Test Case, click the arrow next to the Run on the toolbar and select **Run As | PHP Unit Test** -or- from the Main Menu, go to Run and select **Run As | PHP Unit Test** -or- right-click the file in PHP Explorer view and select **Run As | PHP Unit Test**.

   -Or- to debug the PPHUnit Test Case, click the arrow next to the debug button on the toolbar and select **Debug As | PHP Unit Test** -or- from the Main Menu, go to Run and select **Debug As | PHP Unit Test** -or- right-click the file in PHP Explorer view and select **Debug As | PHP Unit Test**.

   The PHPUnit view will be displayed, with a section showing all the tests run and the results, and two extra tabbed views showing code coverage and failure trace.

3. In the main area of the PHPUnit test view, the results for each of the tests run will be displayed.

   Tests that have passed successfully will be displayed with a green tick icon.

   Tests that have failed will be displayed with a blue X icon.

   Functions with tests that have not been implemented (i.e. functions that tests have not been created for), will have passed but will have a note indicating that they have not been implemented.

4. The number at the top of the view indicates how many tests have been run. Tests may not be run if an 'exit' command is given or if a fatal error is encountered.

5. Click the 'Show failures only' icon to only view failed results.

6. Select a failed result to view it in the Failure Trace view. Click the Filter Stack Trace icon to display only functions relevant to your application and not PHPUnit functions.

7. Double-click on a failed result to be taken to the test function in the test file.
   To correct the failed result, either fix the test function or the original function on which it was run.

8. The Code Coverage display indicates how much of the code in both the original file and the test file was run:
   - The percentage in the Covered Lines column displays the percentage of lines executed out of the total number of executable lines.
   - The number of 'visited' lines are the number of executable code lines.
   - The number of 'significant' lines are the number of significant (i.e. executable) lines.
   - The number of 'total' lines is the total number of lines in the file.

9. Click on the code coverage statistics next to each file to open the Code Coverage view displaying the code with the lines of code that were run.
   'Visited' lines will be highlighted in blue.
   'Significant' lines will be highlighted in pink.

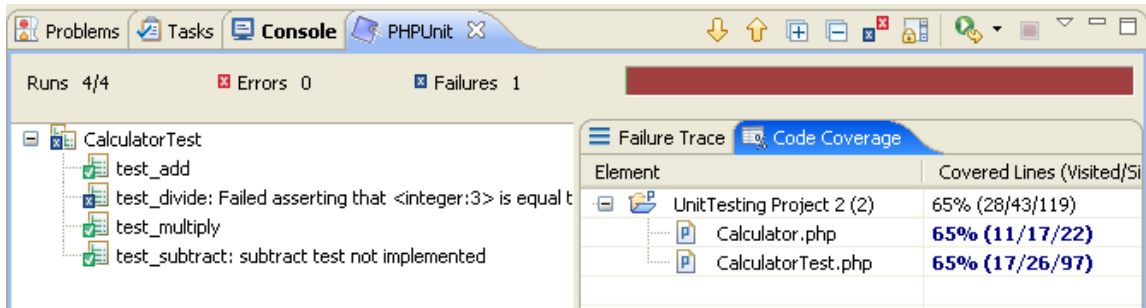Once you have corrected errors, you can re-run the PHPUnit Test by clicking the Run Last Test button in the PHPUnit view until all tests pass successfully.

Debugging a PHPUnit Test Case

To Debug a PHPUnit Test Case:

1. Open your PHPUnit Test Case file in the editor.

2. Click the arrow next to the Run button on the toolbar and select **Run As | PHP Unit Test** –or- from the Main Menu, go to Run and select **Run As | PHP Unit Test** .
   -Or- to debug the PPHUnit Test Case, click the arrow next to the debug button on the toolbar and select **Debug As | PHP Unit Test** –or- from the Main Menu, go to Run and select **Debug As | PHP Unit Test** .

The PHPUnit view will be displayed, with a section showing all the tests run and the results, and two extra tabbed views showing code coverage and failure trace.

3. In the main area of the PHPUnit test view, the results for each of the tests run will be displayed.

    Tests that have passed successfully will be displayed with a green tick icon.

    Tests that have failed will be displayed with a blue X icon.

    Functions with tests that have not been implemented (i.e. functions that tests have not been created for), will have passed but will have a note indicating that they have not been implemented.

4. The number at the top of the view indicates how many tests have been run. Tests may not be run if an 'exit' command is given or if a fatal error is encountered.

5. Click the 'Show failures only' icon to only view failed results.

6. Select a failed result to view it in the Failure Trace view. Click the Filter Stack Trace icon to display only functions relevant to your application and not PHPUnit functions.

7. Double-click on a failed result to be taken to the test function in the test file.
    To correct the failed result, either fix the test function or the original function on which it was run.

8. The Code Coverage display indicates how much of the code in both the original file and the test file was run:

    ▪ The percentage in the Covered Lines column displays the percentage of lines executed out of the total number of executable lines.

    ▪ The number of 'visited' lines are the number of executable code lines.

    ▪ The number of 'significant' lines are the number of lines which were executed.

    ▪ The number of 'total' lines is the total number of lines in the file.

9. Click on the code coverage statistics next to each file to open the Code Coverage view displaying the code with the lines of code that were run.
    'Visited' lines will be highlighted in blue.
    'Significant' lines will be highlighted in pink.

Once you have corrected errors, you can re-run the PHPUnit Test by clicking the Run Last Test button

in the PHPUnit view until all tests pass successfully.

You can transform the xml file created with while running the PHPUnit Teasr into an HTML report using

the Report Generator icon . See [Reporting on PHPUnit Test Results](#) for more details.
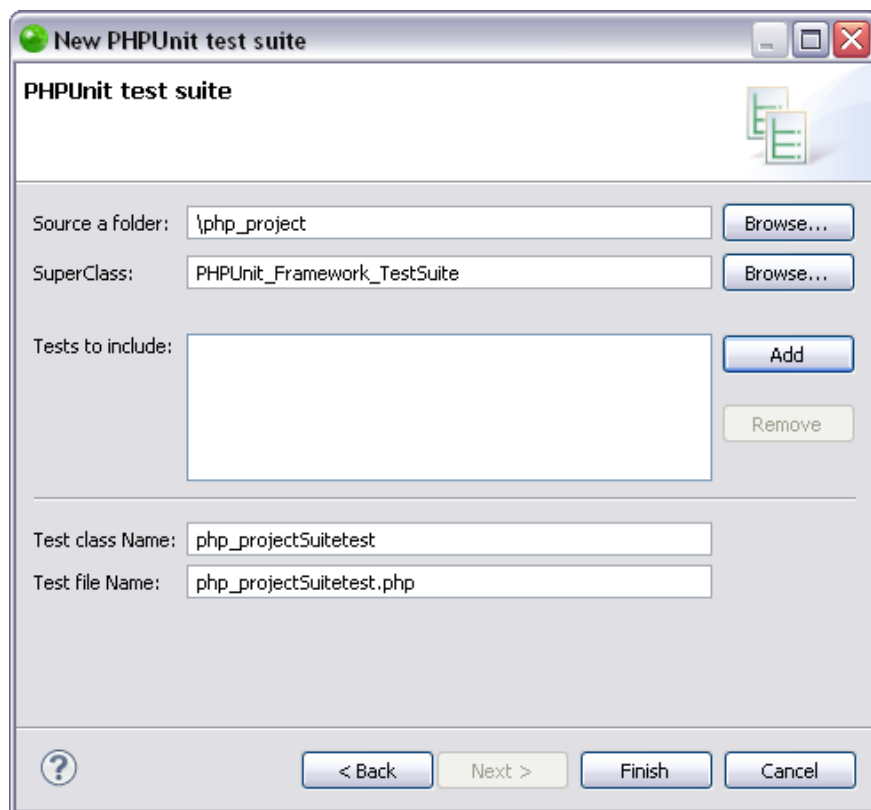
Creating a PHPUnit Test Suite

This procedure demonstrates how to create a PHPUnit Test Suite for running a number of PHPUnit Test Cases at once. This function is useful if you have a number of tests which you would like to unify into one. Before creating the PHPUnit Test Suite, you must have created all your separate PHPUnit Test Cases.

To create a PHPUnit Test Suite:

1. In PHP Explorer View, right-click the project which contains your PHPUnit Test Cases and select **New | Other | PHP | PHPUnit | PHPUnit Test Suite**.

   The "New PHPUnit Test Suite" dialog appears.

2. The 'Tests to include' category will show the available test cases within the project. Cilck **Add** to choose the test cases you would like to include in the Test Suite.

3. Click **Finish**.

A PHPUnit Test Suite will be created, integrating all the separate PHPUnit test cases, and will be added as a file to your project.

Running a PHPUnit Test Suite

This procedure describes how to run a PHPUnit Test Suite and how to analyze the results.

Before running a PHPUnit Test Suite, one needs to be created by following the instructions under [Creating a PHPUnit Test Suite](#).

To run a PHPUnit Test Suite:

1. Open your PHPUnit Test Suite.

2. Click the arrow next to the Run button ![Run button] on the toolbar and select **Run As | PHP Unit Test** ![icon] -or- from the Main Menu, go to Run and **select Run As | PHP Unit Test** ![icon] -or- right-click the file in PHP Explorer view and select **Run As | PHP Unit Test**.

   All the PHPUnit Test Cases contained inside the PHPUnit Test Suite will be run.
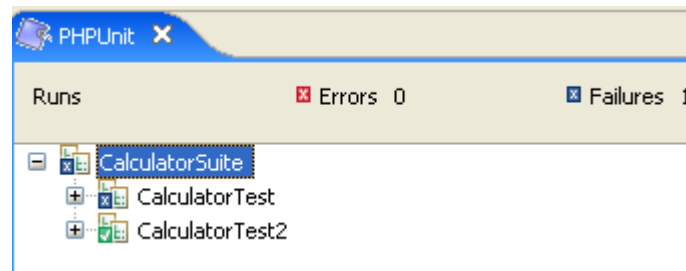
3. The PHPUnit view will be displayed, with a section showing all the tests run and the results, and two extra tabbed views showing code coverage and failure trace.

   The results of the individual PHPUnit Test Cases will be displayed in a tree diagram.

4. Expand the nodes to see the results for each of the individual test cases.

   Tests that have passed successfully will be displayed with a green tick icon. ![green tick icon]

   Tests that have failed will be displayed with a blue X icon. ![blue X icon]

   Tests that have not been implemented (i.e. that tests have not been written for), will have passed but will have a note indicating that they have not been implemented.

5. Double-click on a failed result (if applicable) to be taken to the test function in the test file.

   To correct the failed result, either fix the test function or the original function on which it was run.

6. The Code Coverage display indicates how much of the code in both the original file and the test file was run.

   Click on the code coverage statistics next to each file to open the Code Coverage view displaying the code with the lines of code that were run highlighted in blue.

7. Once you have corrected errors, you can re-run the PHPUnit Test by clicking the Run Last Test button ![Run Last Test button] in the PHPUnit view until all tests pass successfully.

Reporting on PHPUnit Test Results

Once you have run a PHPUnit Test Case/Suite, you can quickly and easily create a report to view the results of your test.

To generate a report:

1. Run a PHPUnit Test Case/Suite. (See Running a PHPUnit Test Case or Running a PHPUnit Test Suite for more information).

2. In the PHPUnit view, click the arrow next to the Report Generator icon ▦ on the view's toolbar to select a report type -or- click the Report Generator icon itself to generate the last generated report.
   See PHPUnit Testing for more on the different types of reports.
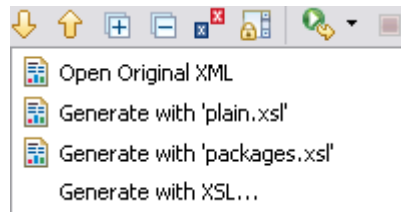
3. A report will be automatically generated and opened in a browser window.

Unit Test Results Report

Note:
Reports will be generated in the location defined in the PHPUnit Preferences page.

4. Clicking the link beneath a failed test result will take you to the relevant test.

Using Refactoring

The Refactoring feature allows you to:

- Rename Files
- Rename Elements
- Move Files
- Extract Variables
- Extract Methods

Note:

Refactoring options will only be available from within PHP Explorer view and not from Navigator view.

Using the Navigator view's move/rename functions will not update any referenced instances of the file/element.

Renaming Files

This procedure describes how to rename files and update all instances where that file is referenced within the project.

To rename a file using the Refactoring feature:

1. In PHP Explorer view, right-click the file which you would like to rename and select **Refactor | Rename** -or- select it and go to **Refactor | Rename** from the Menu Bar.

   A Rename File dialog will appear.
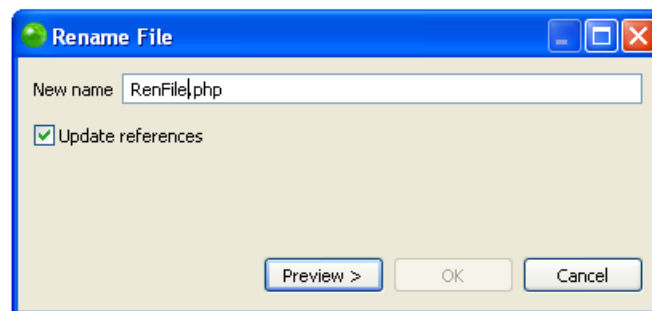
Rename File dialog

2. Enter the file's new name.

3. Check the "Update references" box and click **Preview**.

   A preview window will open with a change tree showing all the changes which will be made to reflect the rename of the file, sorted according to the files in which the changes will be made.
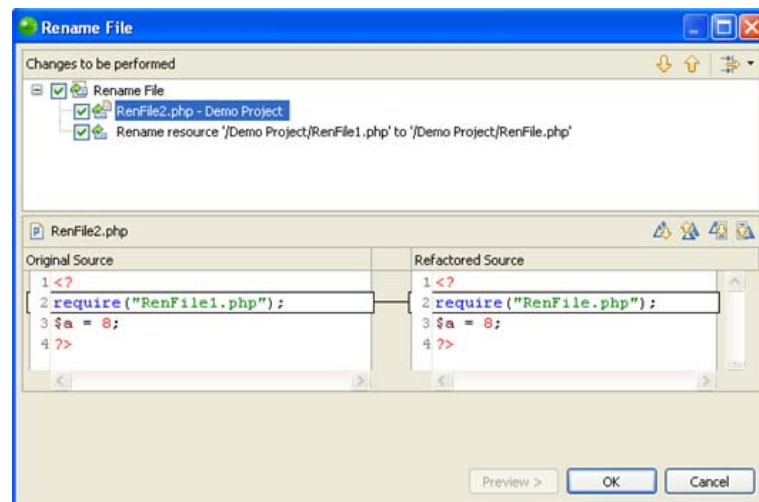
Rename File change tree

Note that, if the file has been referenced (required, included etc.) in other files, the name of the file will also be updated in those instances.

4. You can scroll through the different changes using the Select **Next / Previous Change** scrolling arrows ⬇ ⬆ .

5. If you are satisfied with the changes, press **OK**.

The file will be renamed and all instances where that file is referenced will be updated to reflect the change.

Renaming Elements

This procedure describes how to rename a PHP element and ensure that all references to that element are updated.

All PHP elements can be renamed and refactored from PHP Explorer view. The following is a list of applicable PHP elements:

- Classes
- Interfaces
- Variables
- Methods
- Functions
- Constants
- Class Members

Note:

Ensure that you save any changes to the file before applying the refactoring feature.

Elements can either be renamed from within the editor itself if the In-place refactoring feature is activated (will not display a preview of changes) or from the Rename Element dialog.

Renaming Elements within the Editor

To rename an element within the editor:

1. Ensure the 'Rename in editor without dialog' checkbox is marked in the Code Refactor Preferences page (accessible from **Window | Preferences | PHP | Code Refactor**).

2. In the editor, place your cursor on the element to be renamed.

3. From the menu bar select **Refactor | Rename** -or- right-click and select **Refactor | Rename** -or- press **Alt-Shift-R**.

   All occurrences of the element are put in a frame and the Refactor popup is displayed.

4. Type the new element name in the box.

All occurrences of the element name are automatically updated.

> **Note:**
>
> To preview the changes before applying them, click the arrow in the right-hand corner of the Refactor popup and select Preview.

5. Click **Enter** to apply the refactoring.

A dialog is displayed prompting you to save the file before the refactoring is applied.

6. Mark the 'Always save all modified resources automatically prior to refactoring' checkbox so that the dialog will not be displayed again.

7. Click **OK**.

The element will be renamed and all instances where that element is referenced will be updated to reflect the changes.

Renaming Elements through the Refactor Dialog

To rename an element through the Refactor dialog:

1. Ensure the 'Rename in editor without dialog' checkbox is unmarked in the Code Refactor Preferences page (accessible from **Window | Preferences | PHP | Code Refactor**).

2. In the editor, place your cursor on the element to be renamed or select it in the PHP Explorer.

3. From the menu bar select **Refactor | Rename** -or- right-click and select **Refactor | Rename** -or- press **Alt-Shift-R**

The Rename dialog box will be displayed. The name of the dialog will be dependent on the element type.
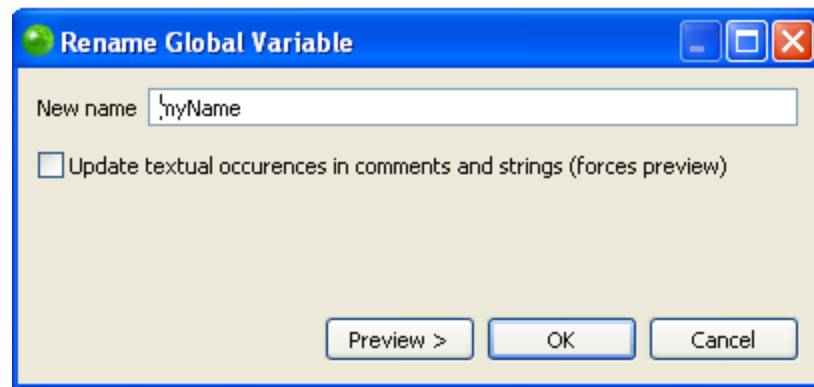
Rename Global Variable

4. Enter the element's new name. You must enter a valid name for the required element - i.e. one that starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

5. Check the "Update textual occurrences in comments and strings" box if you want the element's name to be updated in all comments and strings where it is referenced.
   This will force you to preview the changes before applying them.

6. Click **OK** to apply your changes or click Preview if you want to see a preview of the changes that this refactoring will create.

7. If you clicked preview a preview window will open with a changes tree showing all the changes which will be made to reflect the rename of the element.
   The changes will be listed according to the context within which they appear. You can therefore expand the nodes to see all changes within particular files, classes or functions.

8. Use the Next / Previous Change arrows ⇩ ⇧ to scroll through all possible changes.
   Unmarking the checkboxes next to the changes will cause those changes not to take effect.
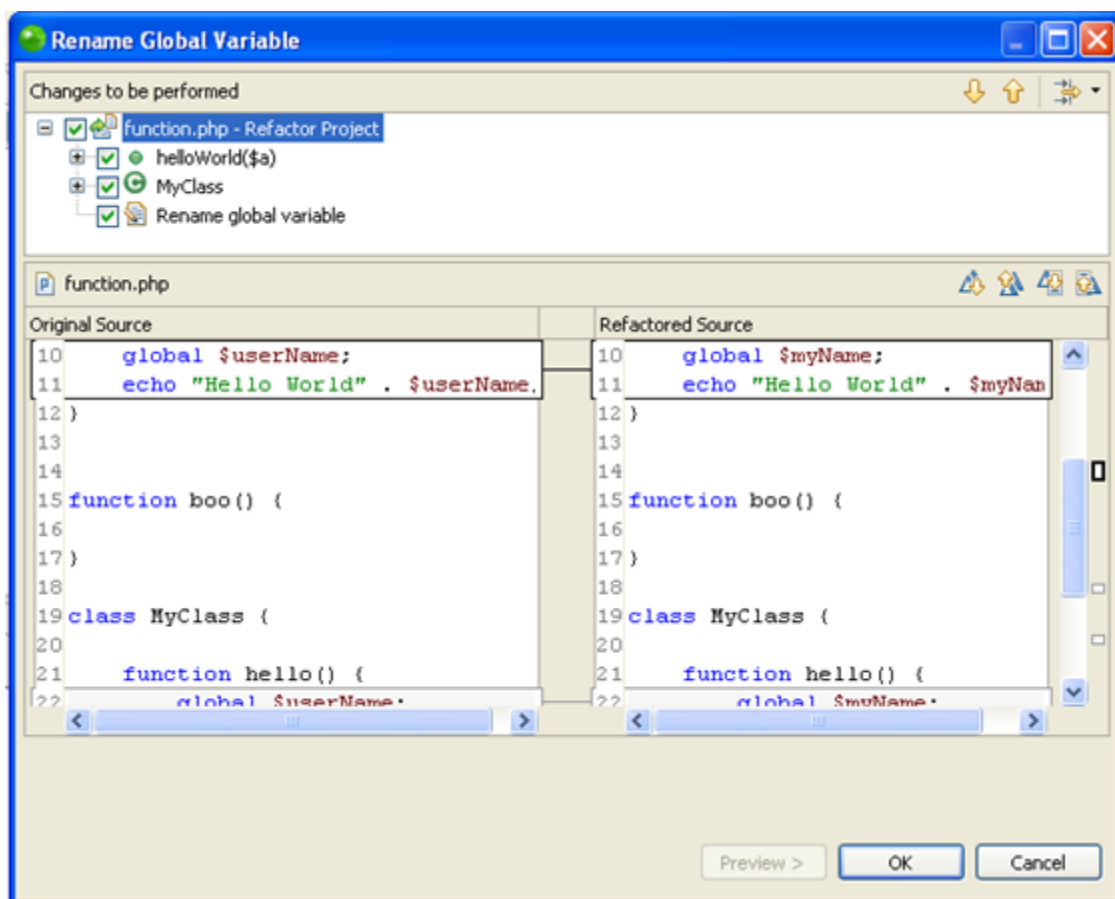
Rename Global Variable changes tree

Note that if changes will be made in other files which reference the element being refactored, the changes will also be listed here under the file name.

9. The changes to be applied will be displayed in the bottom pane.
   You can scroll through the different changes using the scrolling arrows:

   - Next / Previous Difference scrolling arrows - Scroll through changes to be applied within the element selected in the top pane.

   - Next / Previous Change scrolling arrows - Scroll through all changes to be applied. If you unmarked changes in the top pane, these will not be displayed when using these arrows.

10. Once you are satisfied with the changes, click **OK**.

The element will be renamed and all instances where that element is referenced will be updated to reflect the changes.

Moving Files

This procedure describes how to move a file, which will result in the automatic updating of all instances where that file is referenced (required, included etc.) within the project to reflect its change of location.

To move a file:

1. In PHP Explorer view, right-click the file which you would like to rename and select **Refactor | Move** -or- press **Alt-Shift-V**.

   A Move File dialog will appear.
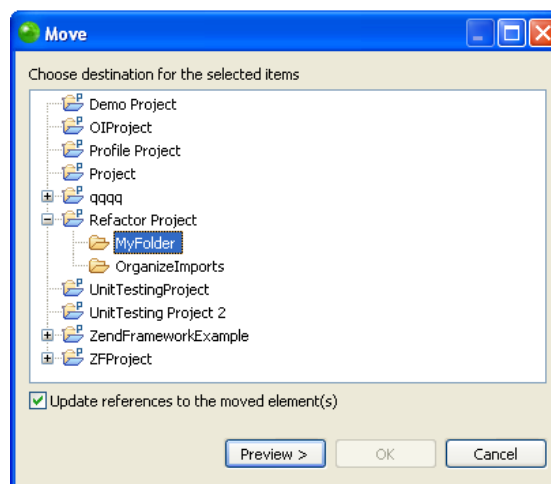
2. Select the new location of the file.
3. Check the "Update references" box and click **Preview**.

   A preview window with a changes tree will open showing all the changes which will be made to reflect the move of the file.
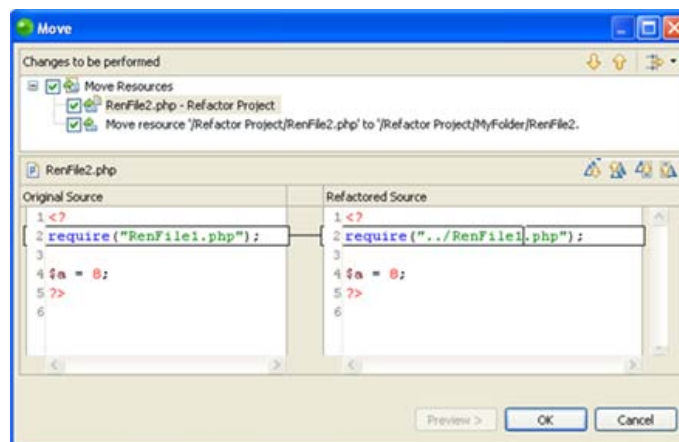
4. You can scroll through the different changes using the scrolling arrows ⬇ ⬆ .

   Note that, if the file has been referenced (required, included etc.) in other files, the reference to the location of that file in other files will also have been changed.

5. Unmark the checkbox of changes which you do not want applied.

6. If you are satisfied with the changes, press **OK**.

The file will be moved and the file's new location will be updated in all instances where that file is referenced.

Extracting Variables

The extract variable feature can create a local variable to replace all occurrences of a given expression.

To create a new variable from an expression:

1. Place your cursor on the expression which you would like to replace with a variable.
2. Right-click and select **Refactor | Extract Variable**

   -Or- press **Alt+Shift+L**.

   The Extract Local Variable dialog is launched.
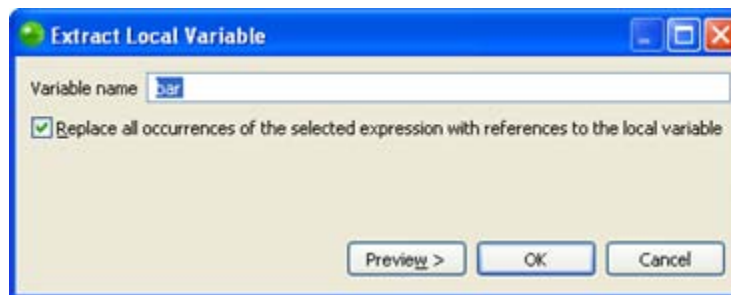
Extract Local Variable dialog

3. Enter the name of the new variable in the Variable name field.
4. Mark the checkbox so that all occurrences of the selected expression will be replaced by references to the local variable.
5. Click **OK** to apply your changes or click Preview if you want to see a preview of the changes that this refactoring will create.
6. If you clicked preview a preview window will open with a changes tree showing all the changes which will be made to reflect the extracting of the variable.
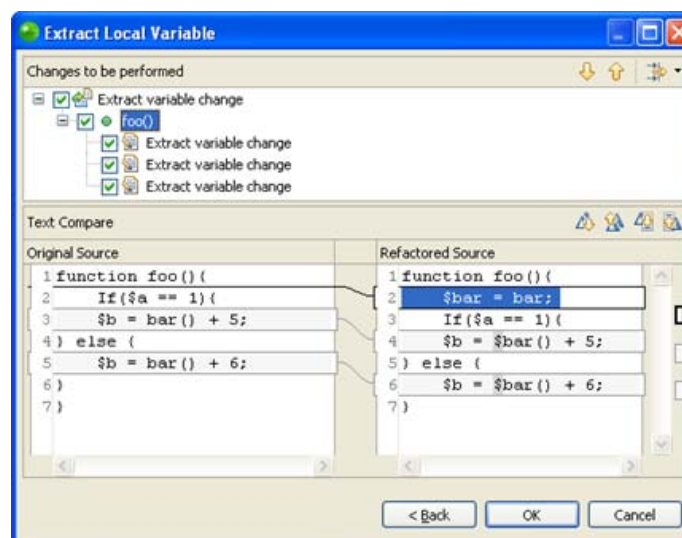
7. The changes will be listed according to the context within which they appear. You can expand the nodes to see all changes within particular files, classes or functions.

8. Use the Next / Previous Change arrows ⬇ ⬆ to scroll through all possible changes. Unmarking the checkboxes next to the changes will cause those changes not to take effect.

9. Click **OK** to apply the changes.

The variable will be extracted and the relevant changes made to the code.

Extracting Methods

The extract method feature can create a method to replace all occurrences of a given code fragment.

To create a new method from an expression:

1. In the editor, select the the code fragement which you would like to replace with a method.
2. Right-click and select **Refactor | Extract Method** -or- click **Alt+Shift+M**.

   The Extract Method dialog is launched.
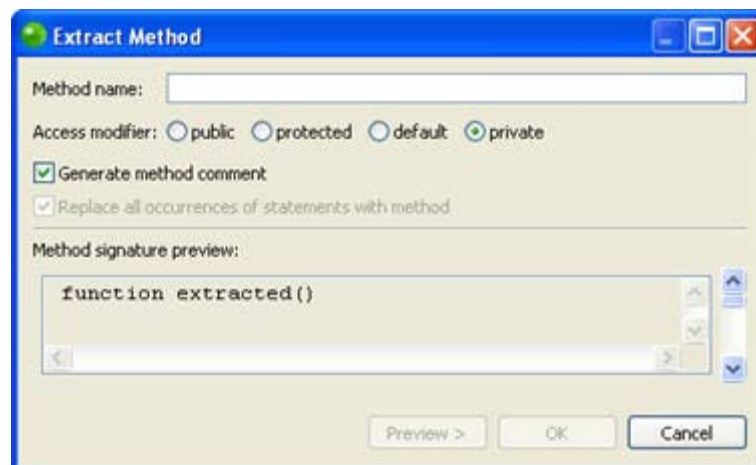
Extract Method dialog

3. Enter the name of the new method in the Method name field.
4. Select the Access modifier for your method.
5. If multiple occurrences of the code fragement appear in your code, mark the 'replace all occurrences of statements with method' checkbox for all occurrences to be replaced with the new method.
6. Mark the Generate method comments checkbox for comments to be created for your method.
7. Click **OK** to apply your changes or click Preview if you want to see a preview of the changes that this refactoring will create.
8. If you clicked preview a preview window will open with a changes tree showing all the changes which will be made to reflect the extracting of the method.
9. The changes will be listed according to the context within which they appear. You can expand the nodes to see all changes within particular files, classes or functions.
10. Use the Next / Previous Change arrows ⇩ ⇧ to scroll through all possible changes. Unmarking the checkboxes next to the changes will cause those changes not to take effect.
11. Click **OK** to apply the changes.

The method will be extracted and the relevant changes made to the code.

Generating Getters and Setters

Zend Studio can automatically create getter and setter functions in order for 'Get' and 'Set' function calls to be easily created.

This procedure describes how to generate getter and setter functions for all variables within a class.

To generate getters and setters:

1. In a PHP file, place your cursor within the class for which you would like to generate the getters and setters.

2. Right-click within the class's source code and select **Source | Generate Getters and Setters** -Or- from the Menu Bar go to **Source | Generate Getters and Setters** -Or- Right-click the required class in PHP Explorer view and select **Source | Generate Getters and Setters**

3. The 'Generate Getters and Setters' dialog will open, displaying all variables that might require a getter/setter.
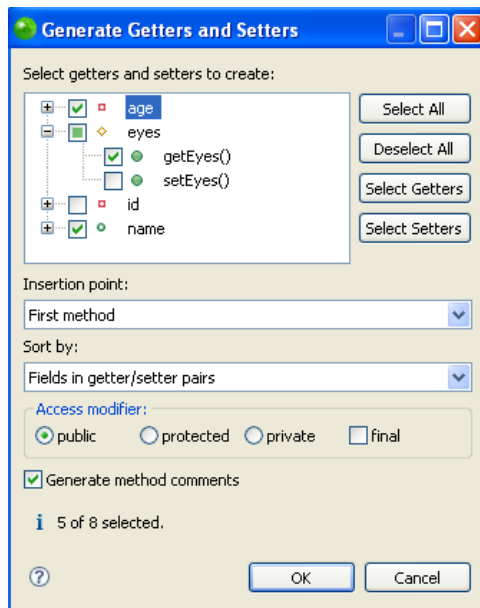
4. Select:

- Which variables getter and setter functions should be created for.

  Expand the list under each function by clicking the + icon in order to select to generate only a getter or a setter for each variable.

  If your cursor was originally placed on a certain variable, this will automatically be selected.

- Insertion Point - Select the location in the class where you want the entries to be added from the drop-down list.

  Options are:

  - First method - Getters and setters will be placed as the first methods within the class.

  - Last method - Getters and setters will be placed as the last methods within the class.

  - Cursor position - Getters and setters will be placed at the cursor position (only available if cursor was placed in a valid position).

  - After function ... - Getters and setters will be placed after the selected function (depending on the functions available within the class).

- Sort by - Determine the order in which the entries are entered.

  The options are:

  - First getters, then setters - All the getters will be grouped together, followed by the setters.

  - Fields in getter/setter pairs - Pairs of getters and setters relating to the same variable will be generated together.

- Access modifier (Not available in PHP 4 projects) - Selects whether the functions will be public, protected, private or final.

- Generate method comments - Select whether to generate a PHP Docblock for each entry.

5. Click **OK**.

The relevant getter and setter functions will be generated for the selected functions.


```php
<?php
class Person{
        private $age;
        private $id;
        public $name;

    /**
     * @return unknown
     */
    public function getAge () {
        return $this->age ;
    }

    /**
     * @param unknown_type $age
     */
    public function setAge ( $age ) {
        $this->age = $age ;
    }
```

Generated getter and Setter

Overriding / Implementing Methods

The override / implement method feature provides a mechanism for overriding/implementing methods defined in a class's parent or interface.

To override / implement methods:

1. Place your cursor within the class for which you would like to override/implement methods.
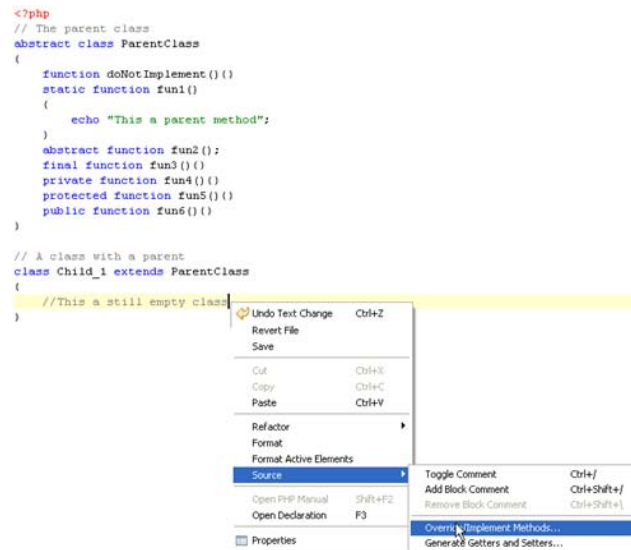
2. Right-click within the class's source code and select **Source | Override/Implement Methods** -Or- Right-click the required class in PHP Explorer view and select **Source | Override/Implement Methods**.

   The ' Override/Implement Methods' dialog will open, displaying all methods that can be overridden.
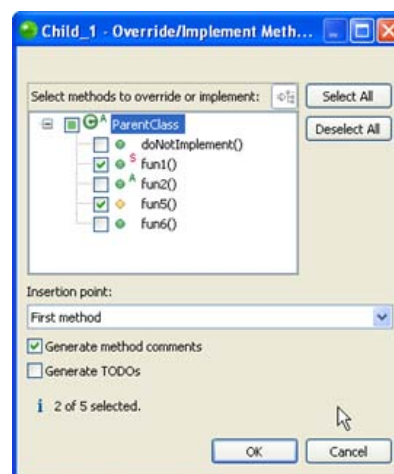
3. Select the elements which you would like to override or implement by marking the checkbox next to the element.

4. Select the insertion point from the drop-down list.

5. Select whether to generate method comments or TODOs (tasks) by marking the relevant checkboxes.

6. Click **OK**.

The relevant code to implement / override the selected methods will be created in your script.

```
34  class Child_1 extends ParentClass
35  {
36
37      /**
38       * @see ParentClass::fun1()
39       *
40       */
41      static function fun1() {
42      }
43
44      /**
45       * @see ParentClass::fun5()
46       *
47       */
48      protected function fun5() {
49      }
50      //This a still empty class
51  }
```

Creating a PHPDoc

This procedure describes how to create a PHPDoc from your PHP files.

To create a PHPDoc:

1. Right-click the project from which you would like the PHPDoc to be generated and select
   Generate PHP Doc -or- go to **Project | Generate PHPDoc** -or- press **Alt+D**.
   The PHPDoc Generation dialog will open.
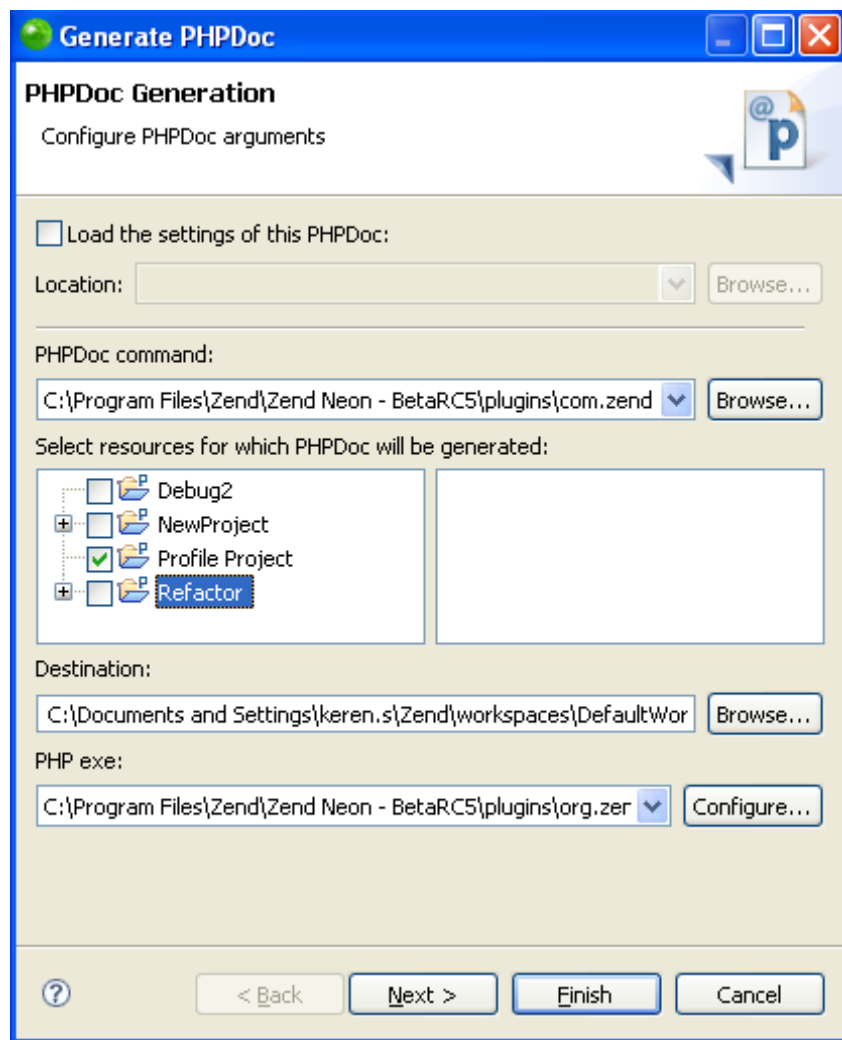
PHPDoc Generation dialog 1

2. If you have previously created PHPDoc settings which you would like to apply, mark the
   checkbox. To create a settings configuration, see point 8.
3. Ensure that the required project, destination for the PHPDoc and PHP Executable are
   selected.

4. Click **Next**.
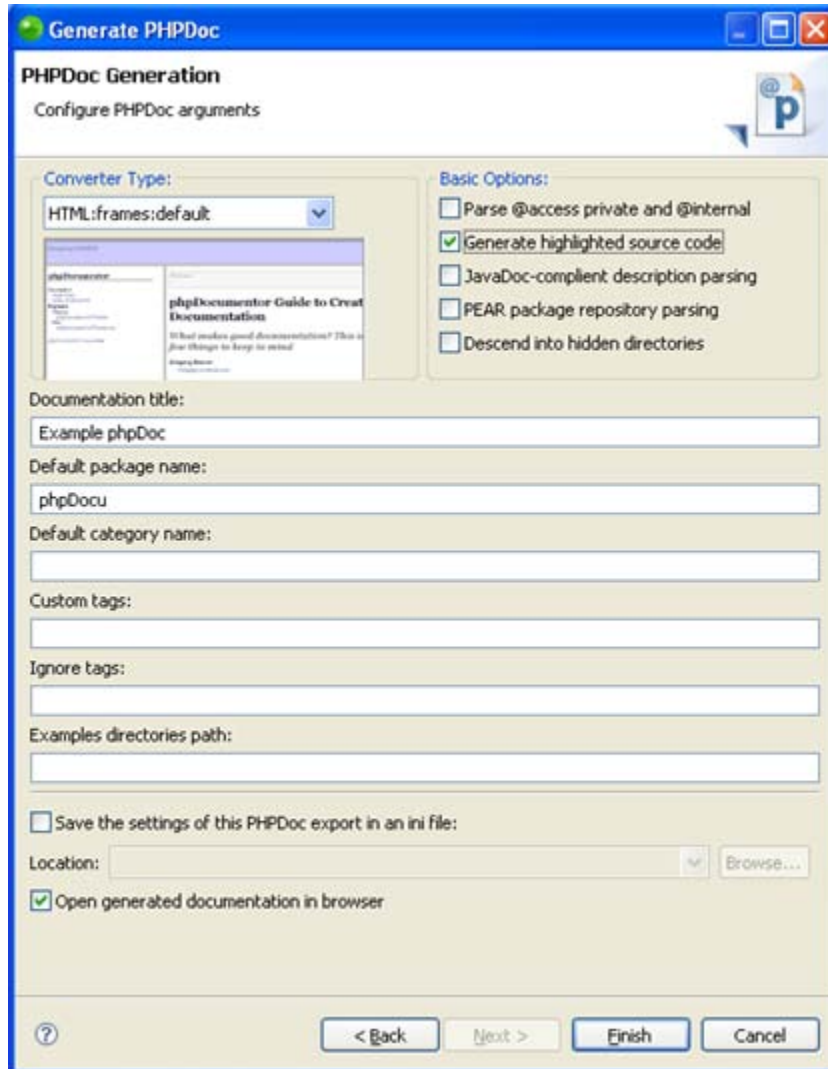
PHPDoc Generation dialog 2

5. Choose the style for your PHPDoc from the 'Converter Type' drop-down list. This will affect the layout and format of your PHPDoc.

6. Select which basic options you want to apply: (Refer to the phpDoc Manual online at http://www.phpdoc.org for complete descriptions of the options.)

- Parse @acess private and @internal
- Generate highlighted source code
- JavaDoc-compliant description parsing
- PEAR package repository parsing
- Descend into hidden directories

7. Enter the following fields: (Refer to the phpDoc Manual online at http://www.phpdoc.org for complete descriptions of the options.)

    - Default package name
    - Default category name
    - Custom tags (if required)
    - Ignore tags (if required)
    - Examples directories path (if required)

8. To save these settings so that they can be reused when creating new PHPDocs, mark the 'Save the settings of this PHPDoc export in an ini file" checkbox and specify where the ini file should be saved.

9. Ensure the 'Open generated documentation in browser checkbox is marked to view your PHPDoc once created.

10. Click **Finish**.

    A Running PHPDocumentor dialog will appear.

Your PHPDoc will be automatically created and will be opened in a browser.

By default, your phpdoc is created as an index.html file in a folder entitled 'docs' in the root of you Workspace. (e.g. C:\Documents and Settings\bob\Zend\workspaces\DefaultWorkspace\docs\index.html).

Creating HTML files

These procedures describe how to create new HTML files, allowing implementation of Zend Studio's full HTML editing functionality.

To create a new HTML file:

1. In PHP Explorer view, select the folder into which you would like to create the file and from the Menu Bar go to **File | New | HTML Page** -or- right-click the folder and select **File | New | HTML. Page**.

   The new HTML Page dialog will appear.

2. Enter the file's name and click **Next**.

3. Select which HTML template to use for the new file and click **Finish**.

The new HTML file will be created and will open in a standard HTML editor.

If you are interested in editing your code in a WYSIWYG view, you can install the WST Web Page Editor plugin. This plugin provides functionality while allowing you to view something very similar to your end outcome before the project has been completed. To download the necessary plugin, follow this link: http://kb.zend.com/index.php?View=entry&EntryID=419.

For more information on adding a plugin see Update Manager.

Note:

More information about writing HTML can be found at: http://www.w3schools.com/html

Using Code Galleries

The Code Gallery view gives you access to code snippets that you pre-defined or that are available through a Code Gallery site.

The Code Gallery view can be accessed from **Window | Show View | Code Gallery**.
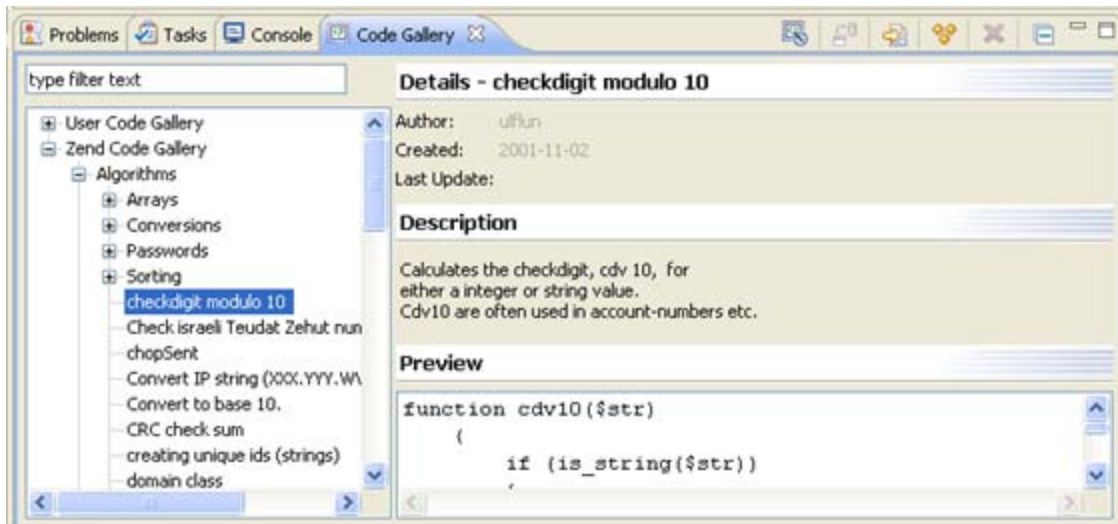
Code Gallery view

See Inserting Code Snippets into your Script, Creating and Editing Code Gallery Entries, and Interacting with Code Gallery Sites for more on working with Code Galleries.

Inserting Code Snippets into your Script

This procedure describes how to insert existing code snippets into your script.

To insert a code snippet into your script:

1. Open the Code Gallery view by going to **Window | Show View | Code Gallery**.

2. Place the cursor at the place in the editor into which you would like the code snippet to be entered.

3. Expand the nodes next to the required User Code Gallery in the Code Gallery view -or- enter a string into the 'type filter text box' to search for a particular category or snippet.

4. Select the required code snippet.

   The details of the code snippet, including its description and a preview of the code, will appear in the right-hand pane.

5. Right-click the snippet and select Insert -or- click the Insert button on the view's toolbar.

The selected snippet will be entered into your script.

Note:

Many snippets contain PHP tags. If necessary, ensure that you remove existing PHP tags from your script to avoid duplication.

Creating and Editing Code Gallery Entries

Adding a Code Snippet to Your List

This procedure describes how to create a new code snippet and add it to your User Code Gallery so that you can quickly and easily insert it into your script.

To add a code snippet to your list:

1. Open the Code Gallery View by going to **Window | Show View | Code Gallery**.

2. Right-click the 'User Code Gallery' from the list and select **New Entry** -or- click the New
   Entry ![icon] icon on the Code Gallery view's toolbar.
   A New code gallery entry dialog will open.

3. Enter the code snippet's name, author and description in the relevant fields.

4. In the 'Code' box, enter the code snippet.

5. Click **OK**.

Your new code snippet will be added to the list.

Editing an Existing Snippet

To edit an existing snippet:

1. Open the Code Gallery View by going to **Window | Show View | Code Gallery**.
2. Expand the nodes next to the required User Code Gallery in the Code Gallery view -or- enter a string into the 'type filter text box' to search for the particular category or snippet.
3. Select the required code snippet.
   The details of the code snippet, including its description and a preview of the code, will appear in the right-hand pane.
4. Right-click the snippet and select **Edit**.
5. The New code gallery entry dialog will open containing the snippet's details.
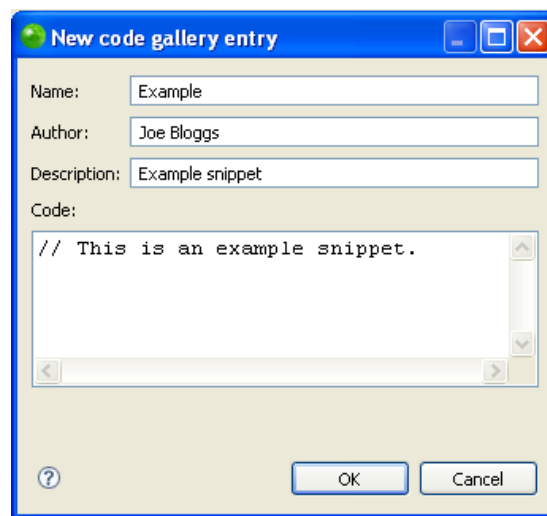
Code Gallery Edit

6. Make the required edits and click **OK**.

The code snippet will be updated with the edits you have made.

Note:

You can only edit snippets in the User Code Gallery.

Interacting with Code Gallery Sites

The Code Gallery view allows access to Code Gallery sites that contain a variety of code snippets.

These procedures describe how to connect to the Zend Code Gallery site to download Zend's code

snippets,how to add a new Code Gallery site, how to update your Code Gallery list, how to suggest code

snippets you have created to the site for use by others and how to give a rating to downloaded code

snippets.

Accessing the Zend Code Gallery

To access the Zend Code Gallery:

1. Open the Code Gallery view by going to **Window | Show View | Code Gallery**.

2. Expand the node next to the Zend Code Gallery list.

   If no node appears, click the 'synchronize with site' button to reactivate it.

3. A login dialog will appear.

Zend Network Login

4. Enter your Zend Network User name and Password.

   If you don't have a Zend Network ID, click the 'Sign up' link to be taken to the Zend

   Developer Zone registration site, or follow this link:

   http://www.zend.com/user/register?redirect=/member/login&sub=devzone.

Your Zend Code Gallery list will be updated with all the code snippets from the Zend Code Gallery site, divided into categories.
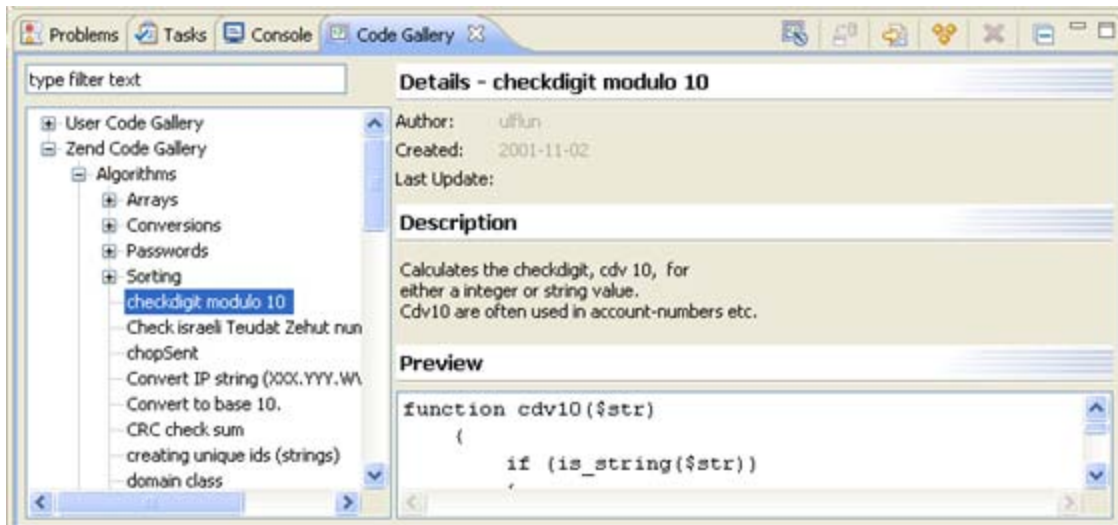
Adding a Code Gallery Site to the Code Gallery List

To add a Code Gallery site to the Code Gallery list:

1. Open the Code Gallery preferences page by going to **Window | Preferences | PHP | Code Gallery**, or clicking the 🖼 'Configure Code Gallery' button on the Code Gallery view's toolbar.

2. Follow the instructions under 'Adding a Code Gallery' in the 'Code Gallery preferences' help page.

Updating Your Code Gallery

To update your Code Gallery:

1. Open the Code Gallery view by going to **Window | Show View | Code Gallery**.
2. Select the Code Gallery which you would like to update.
3. Click the 'synchronize with site' button 🖼 .

Your code gallery list will be updated with all the latest changes from the Code Gallery site.

Suggesting a Code Snippet be Added to a Code Gallery Site

To suggest that a code snippet you have created be added to a Code Gallery site:

1. Open the Code Gallery view by going to **Window | Show View | Code Gallery**.

2. Expand the User Code Gallery node and right-click the code snippet you have created.

3. Click **Suggest**.

    The Suggest to Code Gallery dialog will appear.
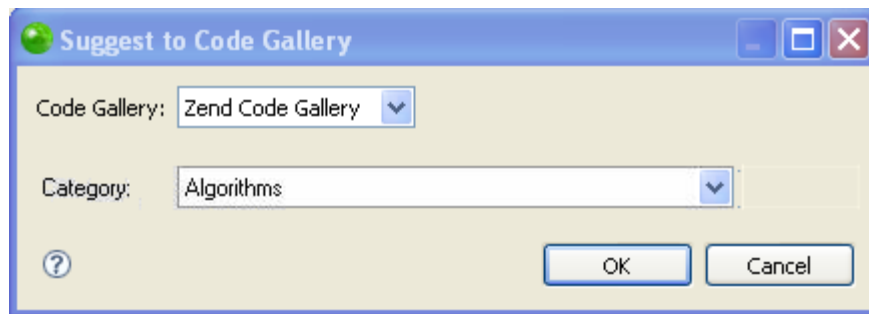
Code Gallery Suggestion

4. Select the Code Gallery to which you would like to suggest your snippet from the Code Gallery drop-down list.

5. Select the category to which you would like to associate it.

6. Click **OK**.

Your code snippet will be sent to the chosen site for consideration.

Rating a Snippet Which You Have Downloaded from a Code Gallery

To rate a snippet which you have downloaded from a Code Gallery:

1. Open the Code Gallery view by going to **Window | Show View | Code Gallery**.

2. Expand the relevant Code Gallery node and right-click the required code snippet.

3. Select 'Rate...'.

    The 'Send your rating' dialog will appear.

4. Select your rating for the snippet (1 being the lowest and 5 being the highest).

5. Press **OK**.

Your rating will be sent to the relevant Code Gallery site.

Configuring Studio Communication Settings in Zend Server

In order for you to be able to view, debug and profile Zend Server Events in Zend Studio, you must ensure the correct communication settings are configured in your Zend Server.

To configure Zend Studio communication in :

1. Open your Zend Server GUI.

2. Go to the **Server Setup | Debugger tab**.

Allowed Zend Studio Clients for Debugging

				Exact IP address only	▾	+	Add

Current list of allowed hosts:

127.0.0.1	Remove
10.*.*.*	Remove
192.168.*.*	Remove
172.16.0.0/12	Remove
245.234.234.*	Remove

Denied Zend Studio Clients for Debugging

				Exact IP address only	▾	+	Add

Use this list to exclude specific hosts from the allowed hosts list.

3. Ensure the address of your Zend Studio is included in the Allowed Hosts sections. This will ensure you can debug/profile Events.

   To add an address to the list:

   i. Enter the IP address or Net mask of the machine on which your Zend Studio is installed.In Order to enter a Net mask, enter a range by entering the beginning of an IP address and adding '0' instead of the rest of the number. To make sure you are using Wildcards (*) to specify a range of IP's, select the pattern you want from the drop-down list.

   ii. Click **Add**. Your Zend Studio machine's address will be added to the Allowed Hosts list.

4. Ensure your Zend Studio's IP address is not in the Denied Hosts list.

   If it is, click **Remove** next to the required address to remove it from the list.

5. In the Zend Server GUI, go to the Server **Setup tab | Monitor** and configure the following:

- Auto detect the Zend Studio Client Settings - Set to 'On' to inform Zend Server of the method of connection to Zend Studio. This allows Zend Server to automatically detect your Zend Studio Debug settings.

6. Click **Save**.

7. Restart your Web Server for the settings to take effect.

Defining Zend Server in Zend Studio

Integration of Zend Server with Zend Studio allows the appliance of Zend Studio functionality (Profiling, Debugging etc.) to Server Events, as well as allowing access to Zend Server's Event list.

Before configuring Zend Server in Zend Studio, Zend Server must be installed and running.

To define Zend Server :

1. Open the PHP Servers Preferences page by going to **Window | Preferences | PHP | Editor | PHP Servers**.
2. Click **New** to create a new server with Zend Server Integration or select an existing server and click Edit to add Zend Server integration to a previously configured server.
   A PHP Server Creation dialog will open.
3. Configure the server as described in the [PHP Server Preferences page](#) (enter the Server's name and document root's URL.)
4. Click **Next**.
5. If necessary, define Path Mapping. See [Managing Path Maps](#) for more information.
6. Click **Next**.
   A Zend Server Integration dialog will appear.
7. Mark the Enable Zend Server Integration checkbox to enable Zend Server Integration features.
8. Insert the desired Zend Server GUI URL suffix and Port Number.
   Leaving the Use default checkbox marked will create a URL in the format <server's document root>/ZendServer>. If necessary, unmark the checkbox to edit the suffix to point to your Zend Server GUI URL.
9. Enter your Password.
10. Click **Next** to [configure Tunneling settings](#) or **Finish** to create your server.

The new server will be added to the Server list and will allow you to use integration features. It will also now be available from the toolbar and the Zend Server Events view.
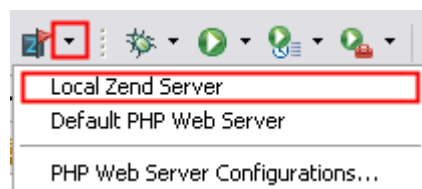
Integrating with Zend Guard

Zend Guard's Integration with Zend Studio allows you to:

- Encode your Zend Studio projects using Zend Guard.
- Open and edit Zend Guard projects in Zend Studio

See the Zend Guard product site (http://www.zend.com/en/products/guard) or the Zend Guard Online Documentation (http://files.zend.com/help/Zend-Guard/zend-guard.htm) for more information on Zend Guard.

Encoding Projects Using Zend Guard

This procedure describes how to open your Zend Studio Projects in Zend Guard in order to encode them.

Zend Guard must be configured in Zend Studio before Zend Guard integration is accessible. This can be configured through the Zend Guard preferences page, accessible from **Window | Preferences | PHP | Zend Guard**.
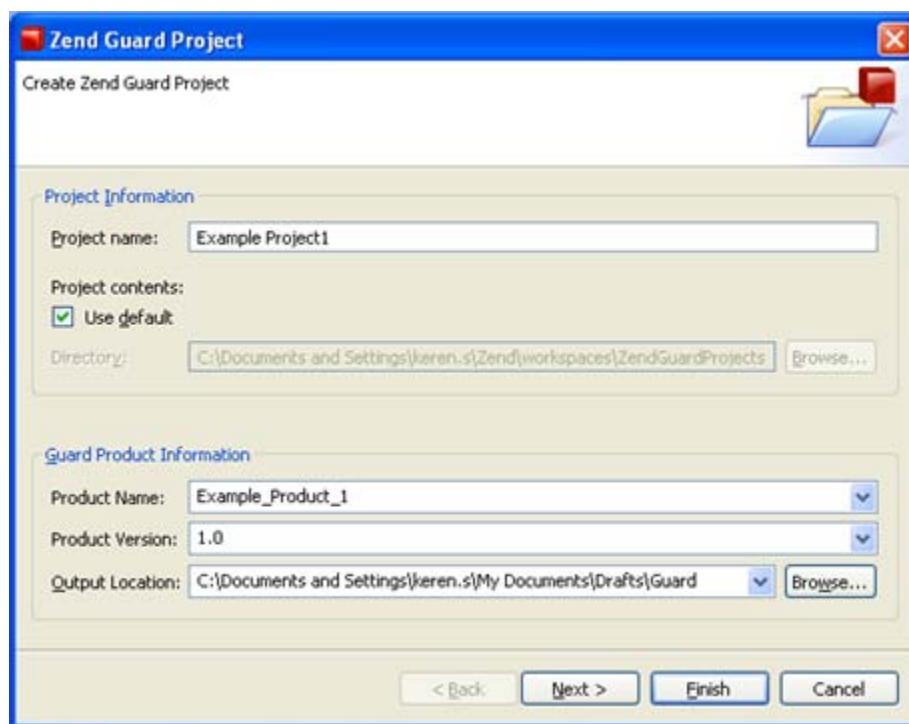
To open a project in Zend Guard:

1. In PHP Explorer view, right-click the required project and select **Encode Project** -or- select the required project and from the Menu Bar go to **Project | Encode Project** -or- click the Encode Project button on the toolbar.

   If you have not configured your Zend Guard location in Zend Studio, a prompt will appear stating that Zend Guard preferences have not been defined. Click on the 'Define Zend Guard.exe' link to be taken to the Zend Guard preferences page or go to **Window | Preferences | PHP | Zend Guard**.

   If you have defined your Zend Guard location, Zend Guard will now be opened and a Zend Guard Project dialog will open.

Zend Guard Project creation dialog

2. Enter the following information in the relevant fields:

- Project Name - This will automatically be taken from the name of your Zend Studio project.
  This cannot be the same name as an existing project in Guard.
- Project contents - The directory into which the project will be placed. Unmark to browse to a different location.
- Product Name - Enter the product's name.
- Product Version - Enter the product's version.
- Output Location - Enter the location to which you would like the encoded files to be created.

3. Click **Next**.

4. If you want additional files and folders to be added to the project, click Add Folder or Add File and browse to the required source.

5. Click **Next**.

6. Select:

- The PHP version
- Whether to enable Short Tag Support - Enable recognition of short PHP tags. Recognizes <? as a valid PHP start tag. When this option is not selected, Zend Guard will not encode short tags, which will be treated as regular HTML.
- Whether to enable ASP Tag Support - Enables recognition of ASP tags. Recognizes <% as a valid PHP start tag. When not selected, code within ASP tags is treated as regular HTML.
- Resolve Symlinks - Resolves Symbolic Links before encoding (not applicable in Windows). A symbolic link (often shortened to symlink and also known as a soft link) consists of a special type of file that serves as a reference to another file or directory. Unix-like operating systems in particular often feature symbolic links.
- Which files to encode - Lists the file extensions for Guard to encode (extensions not listed will not be encoded). File extensions that are not listed here and in "Patterns to Ignore" will be sent as-is to the output folder.
- Which patterns to ignore - Files matching these patterns will not be encoded when encoding a directory, nor will they be copied as-is to the target directory. By default, the list contains the CVS directory and cvsignore files (includes Wildcards '*').

7. Click **Finish**.

Your project will be opened in Zend Guard and can be encoded using Zend Guard's functionality.
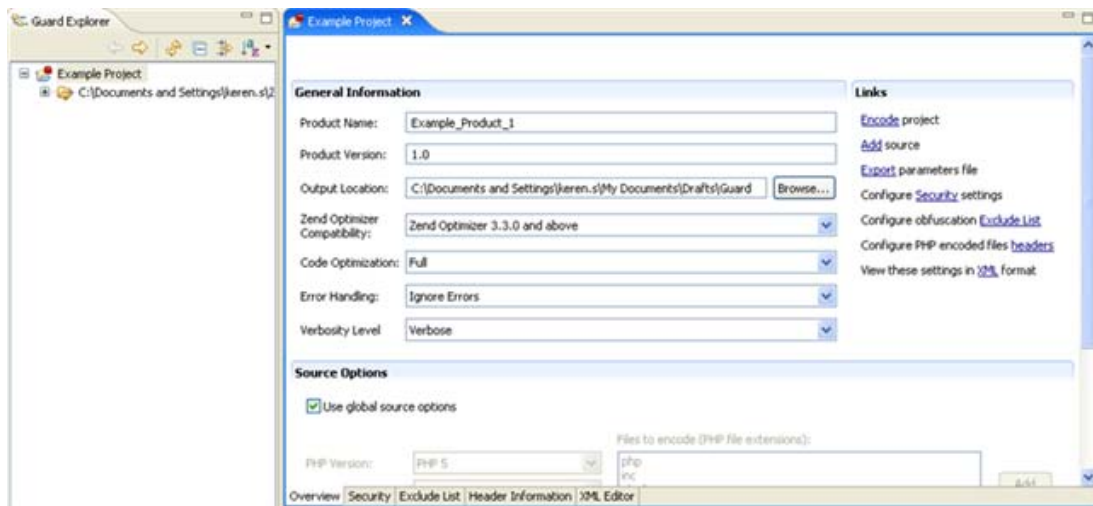
Zend Guard

See the Zend Guard Online Help for more information on encoding your projects.

Note:

Your production server must be running the Zend Optimizer in order for your PHP to be able to run the files encoded by Zend Guard. The Zend Optimizer comes bundled with Zend Server or Zend Core or can be downloaded from the Zend Guard product page (http://www.zend.com/en/products/guard/optimizer).

Note:

Always test encoded files before uploading them to your production server.

Opening and Editing Zend Guard Projects in Zend Studio

This procedure describes how to open Zend Guard's file in Zend Studio, enabling users to extend Zend Guard with the rich editing features included in Zend Studio. This provides users with a seamless workflow for fixing and modifying code through Zend Guard.

To open a Zend Guard file in Zend Studio:

1. In Zend Guard, go to **Edit | Preferences | Zend IDE**.

2. Enter the path to Zend Studio and click **Apply** and **OK**.

3. Right-click the required file in Guard Explorer view and select "Open with Zend IDE."
   This option will only be available once Zend Studio integration has been configured as described in steps 1 and 2.

4. Zend Studio will open and the file will be displayed in an editor.
   All of Zend Studio's editing capabilities will now be available to the file.

5. Once you have made the required edits, click Save on Zend Studio's Menu Bar to save the edits made to the file. Changes will be updated in the file accessible through Zend Guard.

Working with WSDL

Using Zend Studio, you can reference and view existing WSDL files in your PHP Code.

See [Incorporating WSDL Files](#) for more on referencing WSDL files in your code.

Incorporating WSDL Files

Incorporating a WSDL file is the process of taking a service or services from an existing WSDL file and integrating their capabilities (functions) into your PHP code.

To reference a WSDL file and benefit from full integration into Zend Studio (code completion and function display in the PHP Project Outline tab) the file must be present in your local file system (Method 1), or can be referenced by using the WSDL file's URL (Method 2).

Creating a SOAP Client

To create a SOAP Client:

Method 1

1. Use an external browser to open the URL.
2. Save the contents of the URL as a WSDL file. e.g "C:\GoogleSearch.wsdl".
3. Create the soap client by referencing the WDSL file in your PHP script.

   e.g:

```
$a = new SoapClient("C:\GoogleSearch.wsdl")
```

The WSDL file's methods, classes, functions, etc. will now be available in the Code Assist Menu and the SOAP Client now appears in the PHP Project Outline tab.

To create a SOAP Client:

Method 2

1. Create a new SoapClient instance in your PHP code, containing the URL of the WSDL file which you want to incorporate.

   e.g.:

```
$a = new SoapClient("http://api.google.com/GoogleSearch.wsdl");
```

2. Save the file.

The WSDL file's methods, classes, functions, etc. will now be available in the Code Assist Menu and the SOAP Client now appears in the PHP Project Outline tab.

After Referencing a WSDL File

Once a WSDL file has been referenced in your project, the following will be affected:

Outline and PHP Project Outline views

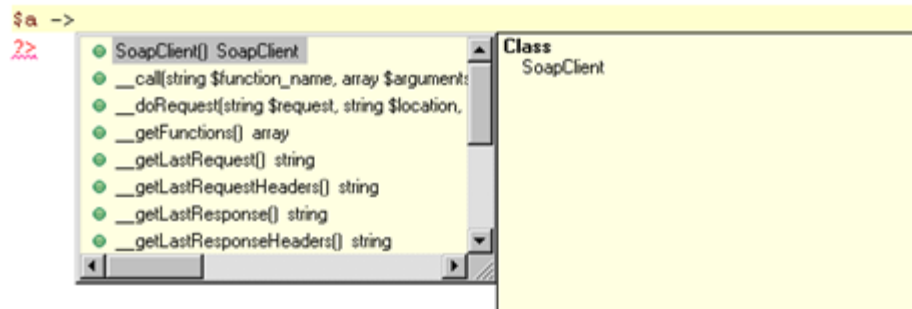The PHP Project Outline will now include all the functions from the referenced WSDL file.
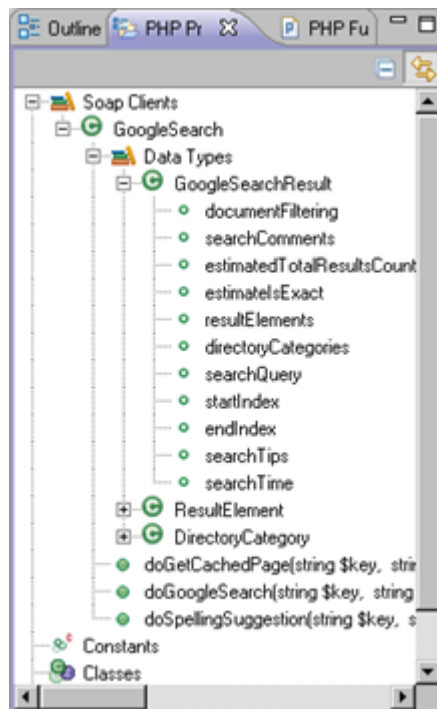
PHP Project Outline view - with the GoogleSearch classes

Content Assist

Code completion is automatically updated with all the functions included in the referenced WSDL file.

Note:

Code completion for Web Services is supported for PHP 5.

Auto Link to WSDL files

Transform the name of the referenced WSDL file into a link by hovering over the file's name and pressing CTRL. Clicking the link (while CTRL is still pressed) will jump to the WSDL file if it is already open in the editor. Auto Link to Files exists for every string containing a file name in the editor.

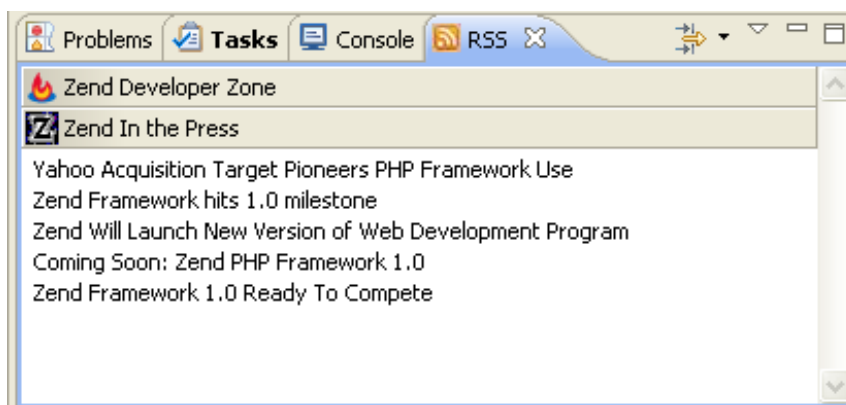# Viewing RSS Feeds and Adding RSS Channels

Viewing RSS Feeds

These procedures describe how to view RSS feeds within Zend Studio's RSS view and how to add additional RSS channels.

To view RSS feeds:

1. Open the RSS view by going to **Window | Show View | RSS**.

RSS view

2. The RSS reader comes pre-loaded with the Zend Developer Zone feed and the Zend In the Press feed.

3. Click the required heading to expand the list and see all the news items underneath it.

   You can group the items by channels or by time by clicking the view's menu icon ▽ and selecting the relevant option (Group by Channels / Group by Time).

4. Double-click the item you want to view to open it in Zend Studio's internal browser.

Adding an RSS Channel

To add an RSS channel:

1. Open the RSS view by going to **Window | Show View | RSS**.

2. Click the view's menu icon ▽ and select Subscribe.

3. The Add New Channel dialog will be displayed.
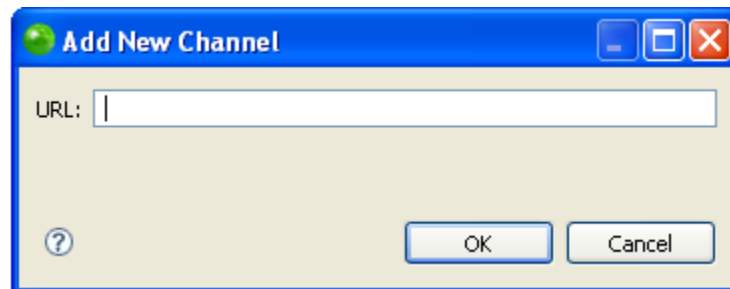
Add New RSS Channel dialog

4. Enter the URL of the site from which the RSS feeds will come and click **OK**.

The new RSS feed will be displayed in your RSS view.

Working with Remote Server Support

Remote Server Support allows you to transparently access your remote server and remote resources. This provides an easy way to upload and download files from your remote server, as well as allowing you to develop your code in one environment, while in parallel executing it in a different environment. In addition, you can create and manage connections to your FTP and SSH remote systems through Remote Server Support. This will allow you to work on projects locally while keeping them updated on your remote server.

Remote Server Support allows you to perform the following tasks:

- [Work with Remote Connection Profiles](#)
- [Add a Remote Connection Profile](#)
- [Edit a Remote Connection Profile](#)
- [Remove a Remote Connection Profile](#)
- [Creating a New PHP Project with Remote Server Support](#)
- [Enable/Disable a PHP Project as a Remote Project](#)
- [Upload Folders/Files to a Remote Server](#)
- [Download Folders/Files from a Remote Server](#)
- [Work with Inclusion/Exclusion Patterns](#)

Important Note;
If you have a project that was associated with a remote server in Zend Studio previous to version 8.0, you can [enable](#) your project to be a remote project, or [create](#) a new project and download the data from the remote server.

Remote Server Support can be used with the existing Eclipse version control (CVS or SVN), or another version control you are using. For more information on built-in Eclipse version control options see the [Subversive User Guide](#), or the [Team CVS tutorial](#) topic in the [Workbench User Guide](#).

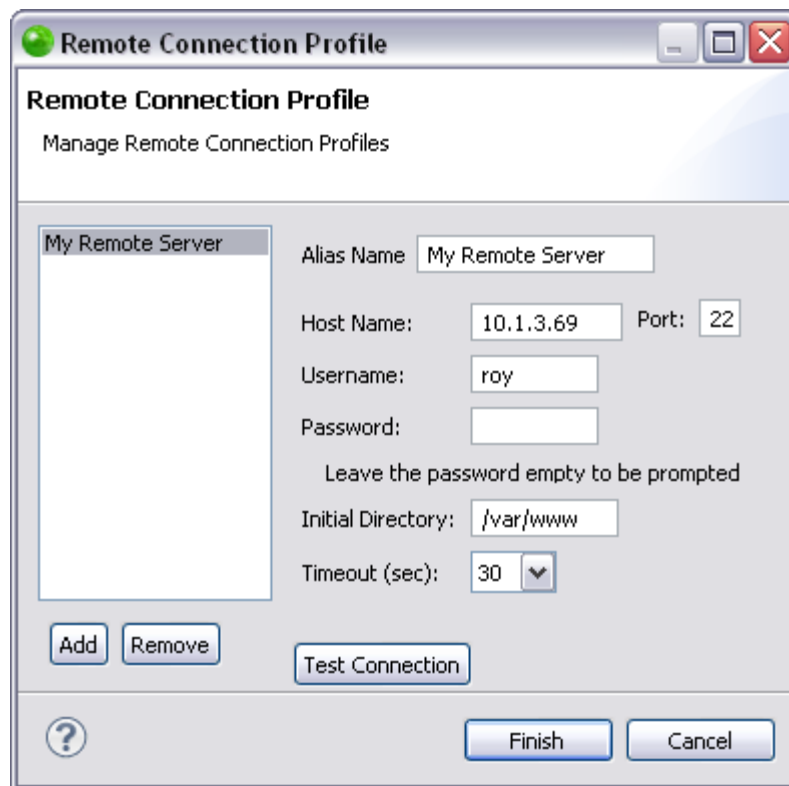**Working with Remote Connection Profiles**

Remote Connection Profiles allow you to define the properties of a remote server to associate with a project. Zend Studio transfers data with the Remote Server Transfer Mode you select in the Remote Server Support Properties page to/from the remote server you specify in the Remote Connection Profile dialog. This makes the upload and download process easier as you do not have to spend time defining the remote server every time you would like to perform a data transfer.

From the Remote Connection Profile dialog you can:

- Add a Remote Connection Profile
- Edit a Remote Connection Profile
- Remove a Remote Connection Profile

Accessing the Remote Server Support Properties Management Dialog (Known as the Remote Connection Profile Dialog)

The Remote Connection Profile dialog is accessed by clicking **Manage** in the Remote Server Support Properties page (which is accessible by selecting a remote PHP project and going to **Project | Properties | Remote Server Support** - Or - Selecting **Properties | Remote Server Support** from the Right Click Menu in your project directory).

This dialog includes the following options:

- Alias Name - A name you give to your Remote Connection Profile. The remote server is displayed with this name in the Remote Connection dropdown menu in the Remote Project Properties page.

- Host Name - A pre-defined name for the remote server. This can also be the IP address of the remote server.

- Port - The port of your remote server.

    o Port 22 is used for SSH servers.

    o Port 21 is used for FTP servers.

- Username - The username associated with the remote server. This is defined in your remote server.

- Password - The password associated with the remote server. Leave this field empty to be prompted for the password each time you connect to the remote server.

- Initial Directory - The folder in the remote server which contains the Project Directory you would like to associate with your project.

- Timeout -The time (in seconds) Zend Studio will attempt to connect to the remote server before returning an answer.

- Test Connection - Click to test if Zend Studio can connect to the remote server according to the details you have entered in the Remote Connection Profile.

- Add - Add a new Remote Connection Profile.

- Delete - Delete a Remote Connection Profile.

When connecting to an FTP server there are two additional options:

- FTP Parser - Choose from the dropdown menu according to the operating system you are working with. By default the setting is AUTO, which will automatically detect which operating system is being used.

- Passive Mode - Select the checkbox to run your FTP connection in passive mode.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.
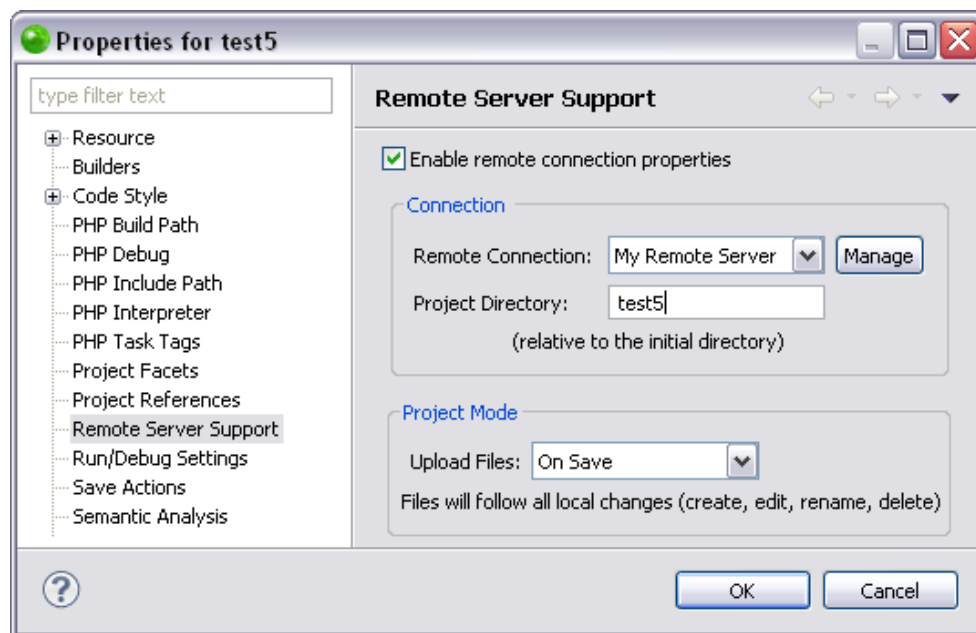
Adding a Remote Connection Profile

This procedure describes how to add a new Remote Connection Profile. Before adding a profile, you must [enable your PHP project as a remote project](). Adding a Remote Connection Profile allows you to define the connection settings for a remote server which you can then associate with your project.
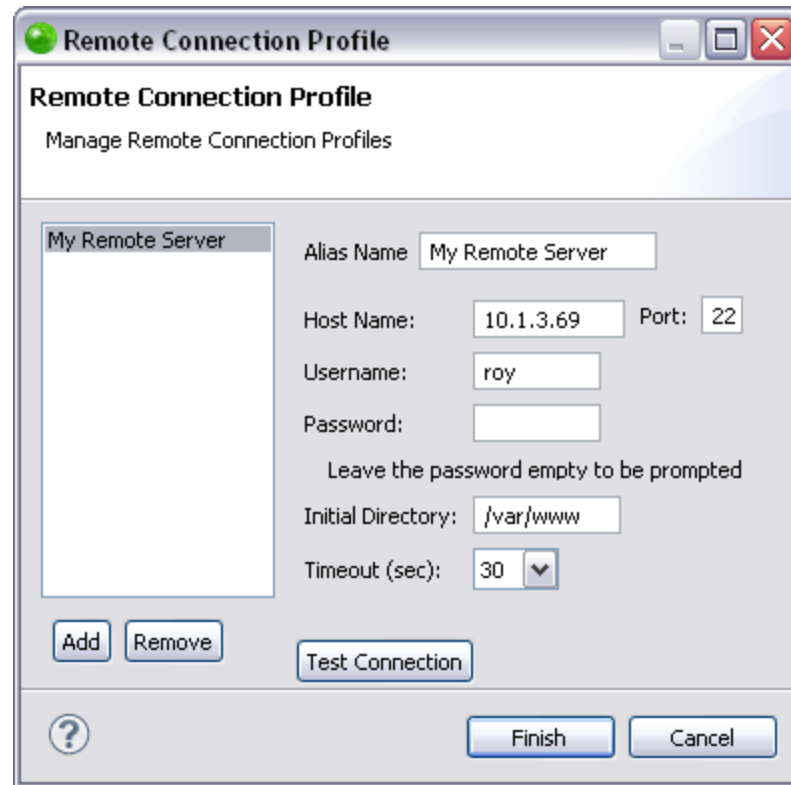
To add a Remote Connection Profile:

Go to **Project | Properties | Remote Server Support** - Or - Select **Properties | Remote Server Support** from the Right Click Menu in your project directory.
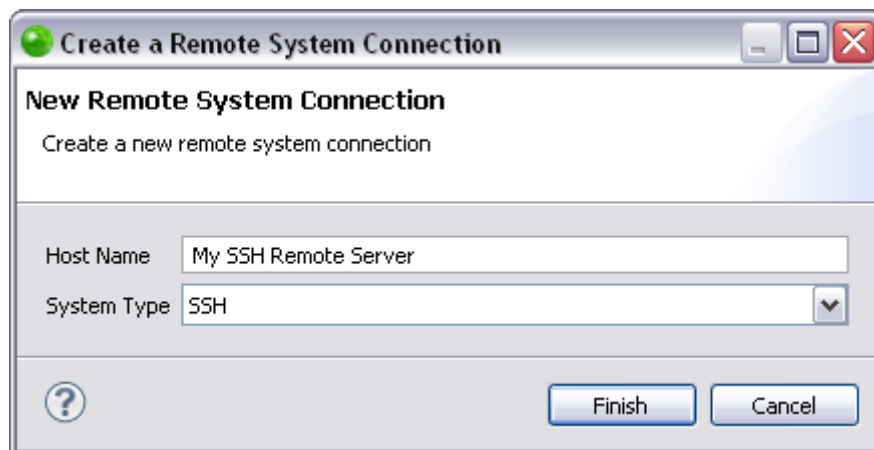
The Remote Project Properties page opens.

2. Click **Manage**.
   The Remote Connection Profile dialog opens.

3. Click **Add**.
   The Create New Remote System Connection dialog opens.

4. Enter the Host Name in the Host Name text field.
   The Host Name can be the IP address of the remote server, or a pre-defined name the server has been given. The Host Name cannot be defined in Zend Studio as it is already defined in the remote server.

5. Select a System Type in the System Type dropdown menu.

   The options in this menu are:

   - SSH

   - FTP

6. Click **Finish**.

   Return to the Remote Connection Profile dialog to define the additional required properties of your new Remote Connection Profile by editing your Remote Connection Profile.

Once you have defined a Remote Connection Profile, you can associate it with any PHP remote project. This is done by selecting the associated Alias Name from the Remote Connection dropdown menu in the Remote Server Support Properties page of the project.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.
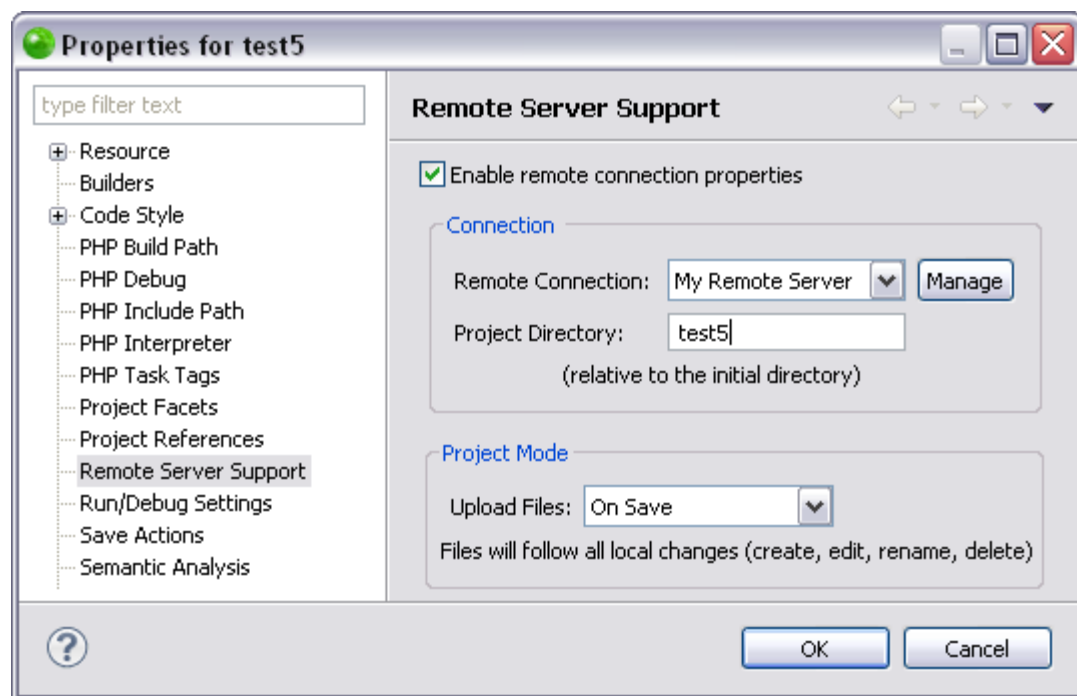
Editing a Remote Connection Profile

This procedure describes how to edit a Remote Connection Profile. Editing a Remote Connection Profile allows you to manage the connection settings to your remote server. Before editing a profile, you must first add a Remote Connection Profile.
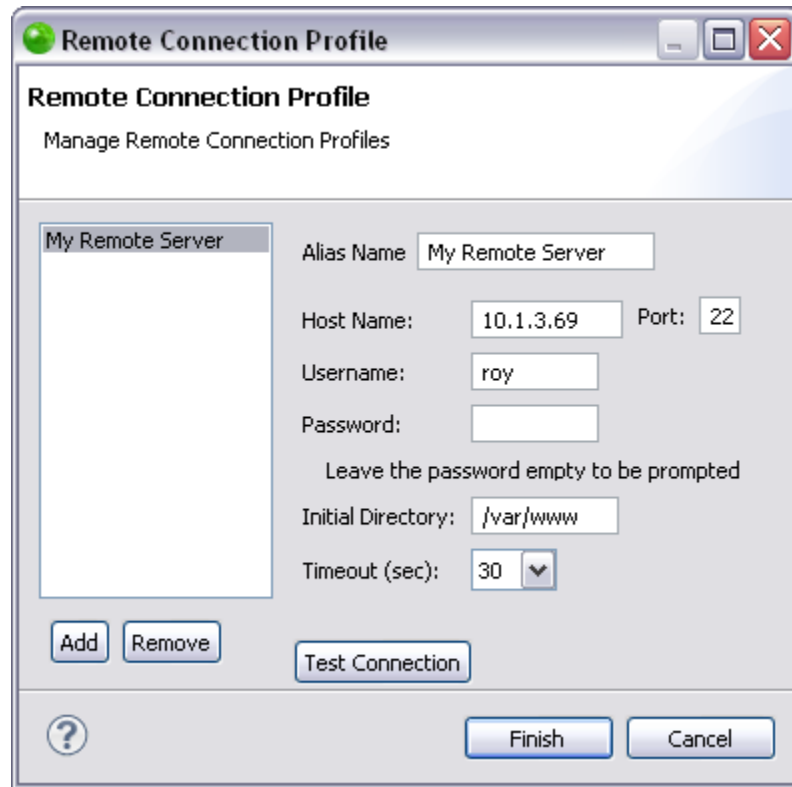
To edit a Remote Connection Profile:

1. Go to **Project | Properties | Remote Server Support** - Or - Select **Properties | Remote Server Support** from the Right Click Menu in your project directory.
   The Remote Project Properties page opens.

2. Click **Manage**.

The Remote Connection Profile dialog opens.

3. The following fields are available for editing:

- Alias Name - A name you give to your Remote Connection Profile. The remote server is displayed using this name in the Remote Connection dropdown menu in the Remote Project Properties page.

- Host Name - A pre-defined name for the remote server. This can also be the IP address of the remote server. The Host Name is what defines which remote server you are connecting to.

Note:
Altering the Host Name field causes every project which is associated with that specific Remote Connection Profile to now be associated with the new remote server.
If you would like to add a new remote server with its own Remote Connection Profile without affecting any existing projects, folders or files, you must add a Remote Connection Profile.

- Port - The port of your remote server.
  - Port 22 is used for SSH servers.
  - Port 21 is used for FTP servers.

- Username - The username associated with the remote server. This is defined in your remote server.
- Password - The password associated with the remote server. This is defined in your remote server.
  Leave this field empty to be prompted for the password each time you connect to the remote server.
- Initial Directory - The folder in the remote server which contains the Project Directory you would like to associate with your project.
- Timeout - The time (in seconds) Zend Studio will attempt to connect to the remote server before returning an answer.

When connecting to an FTP server there are two additional options:

- FTP Parser - Choose from the dropdown menu according to the operating system you are working with. By default the setting is AUTO, which will automatically detect which operating system is being used.
- Passive Mode - Select the checkbox to run your FTP connection in passive mode.
- Click **Finish** to apply and save the changes to your Remote Connection Profile.

Once your Remote Connection Profile has been edited, click **Test Connection** to test if Zend Studio can connect to the remote server according to the details you have entered in the Remote Connection Profile.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Removing a Remote Connection Profile

This procedure describes how to remove a Remote Connection Profile. You will want to remove a Remote Connection Profile when you no longer want to use the remote server associated with it. In order to remove a profile you must first add a Remote Connection Profile.

Note:

Removing a Remote Connection Profile causes any project associated with that Remote Connection Profile to be disabled as a Remote Project. You must enable the PHP project as a Remote Project to associate it with a new Remote Connection Profile.

To remove a Remote Connection Profile:

1. Go to **Project | Properties | Remote Server Support** - Or - **Select Properties | Remote Server Support** from the Right Click Menu in your project directory.

   The Remote Project Properties page opens.
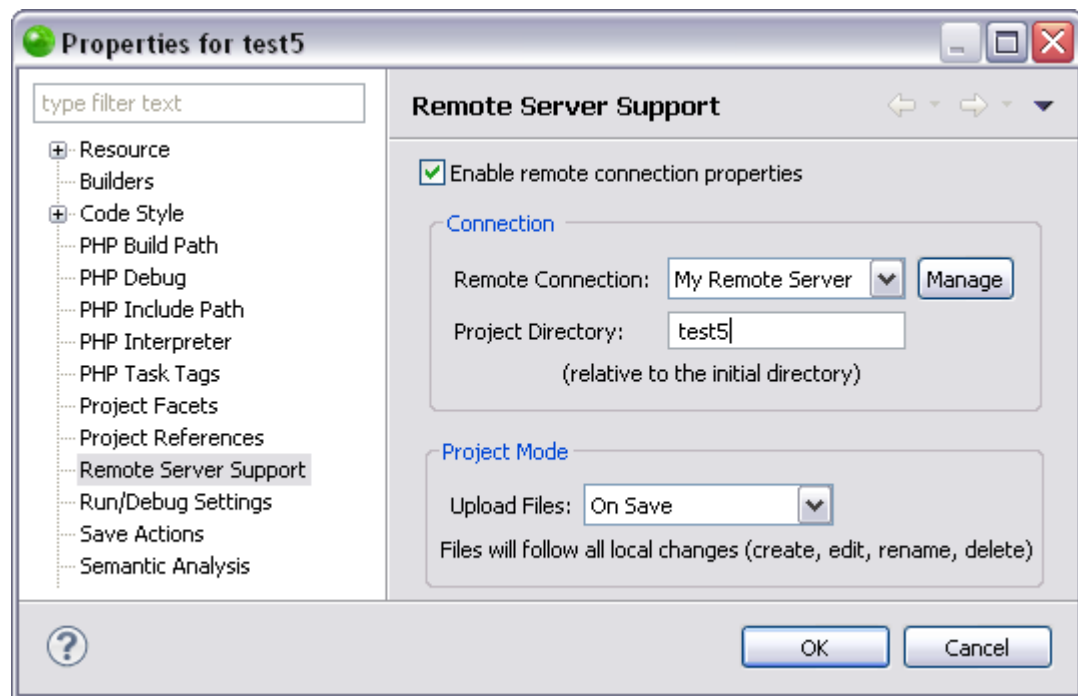
2. Click **Manage**.

The Remote Connection Profile dialog opens.
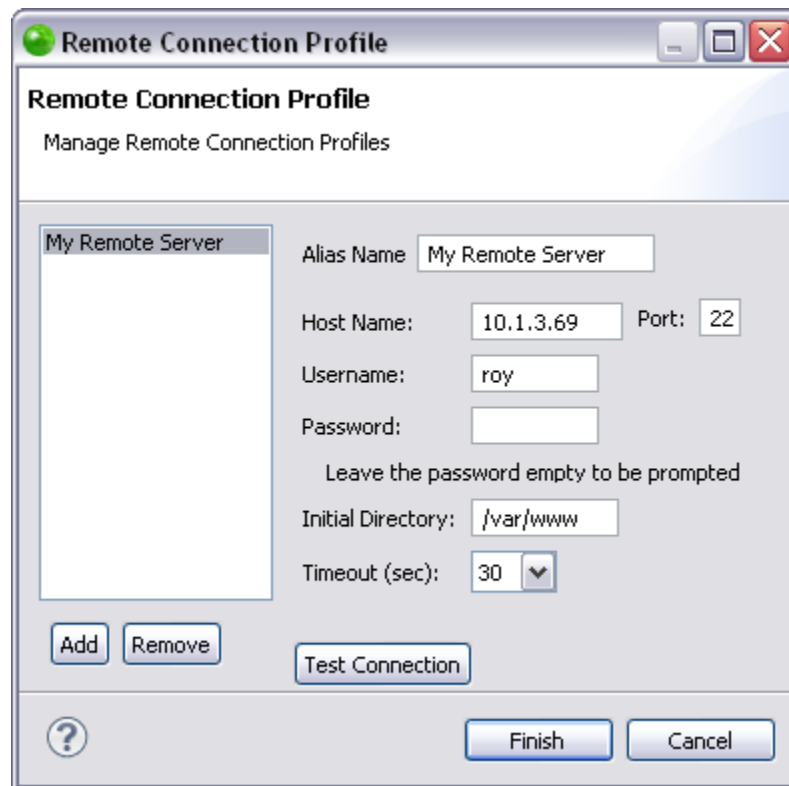
3. Select the Remote Connection Profile you would like to delete and click **Remove**.

The Remote Connection Profile is deleted.

You can now add a Remote Connection Profile to associate your project with a remote server.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Creating a New PHP Project with Remote Server Support

This procedure describes how to create a new PHP remote project. A PHP remote project allows you to develop your project locally, with the option of remote testing and integration.

Important Note:

Creating a new PHP Remote Project can only be carried out if the project already exists on the remote server. To create a new project that does not exist on the remote server yet, create a PHP project and then enable it as a PHP Remote Project.

To create a new PHP remote project:

1. Go to **File | New | Other | PHP | PHP Project from Remote Server**

   - Or - In the PHP Explorer view, right-click and select **New | Other | PHP | PHP Project from Remote Server**.

   The New PHP Project dialog opens.

2. Enter the following information:

- Project name - The required project name.

- Contents - Select whether to:

   o Create a new project in the workspace - Creates a new PHP project in the workspace directory.
   By default a workspace will have been created in @user.home/Zend/workspaces/DefaultWorkspace8 when you first launched Zend Studio.

   o Create a project from existing source - Creates a PHP project pointing to files situated outside of the workspace. Click **Browse** to select the required source content.

- PHP Version - Select whether to:

   o Use default PHP settings - Uses the default PHP Interpreter settings. For more information on PHP Interpreter and PHP version compatibility see Execution Environments Preferences.

   o Use project specific settings - Select the PHP version to be used for the project. See PHP Version Support for more information.

- Project Layout - Select whether to:

   o Use project as source folder - All resources within the project will be added to the Build Path by default.

   o Create separate folders for source files and public resources - Separate folders will be created in which you can place resources which should be included or excluded from the Build Path. See Configuring a Project's PHP Build Path for more information.
   The default setting for this option can be configured from the New Project Layout Preferences page.

- JavaScript Support - Mark the 'Enable JavaScript support for this project' check-box for JavaScript functionality (e.g. JavaScript Content Assist options) to be available to the project. See Enabling JavaScript Support in PHP Projects in PHP Projects for more information.

3. Click **Next**.

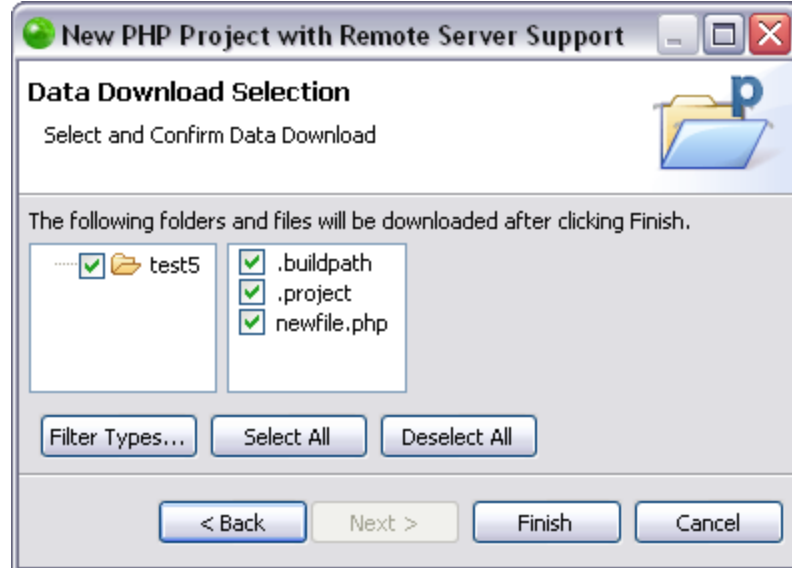   The Remote Configuration dialog opens.

4. Select a Remote Connection from the drop down menu.

   If you do not have an existing Remote Connection Profile you can configure one by clicking **Manage**. For more information see Working with Remote Connection Profiles.

   The Project Directory is configured automatically based on your project name.

5. Make sure the Project URL is correct and click **Next**.

   The Data Download Selection dialog opens.

6. Expand the Project Directory to see all the folders within it that will also be transferred.

7. Select the folders you want to download from the remote server by selecting the check-box next to the folder name.

8. Click on a selected folder to view the files included in the folder.

   The individual files can also be selected/unselected to download from the remote server.

By default, all files within a folder are selected.

On this page you can also:

- Filter Types - Opens the Inclusion and Exclusion Patterns dialog. For more information see [Working with Inclusion and Exclusion Patterns](#).
- Select All - Selects all the folders and files.
- Deselect All - Deselects all the folders and files.

9. Click **Finish**.
   The selected files are downloaded and the new PHP Remote Project with Remote Server Support is created.

The new PHP project will be created in your workspace and displayed in the [PHP Explorer View](#).

You can now start to develop your application by [creating PHP Files](#) or adding other resources to your project.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Enabling/Disabling a PHP Project as a Remote Project

This procedure describes how to enable a local PHP project as a remote project by applying a Remote Connection Profile to it, and therefore associating it with a remote server. Working with remote projects allows you to develop your code in one environment, while in parallel executing it in a different environment.

Enabling/disabling a project is done through the Remote Server Support Properties.

Enabling a Project as a Remote Project

This procedure describes how to enable your local project as a remote project by associating it with a remote server. This allows you to transparently access your remote server and remote resources, providing an easy way to upload and download files from your remote server.

To enable a project as a remote project:

1. Select the project you would like to enable and go to **Project | Properties | Remote Server Support** - Or - Select **Properties | Remote Server Support** from the Right Click Menu in your project directory.

   The Remote Project Properties page opens.
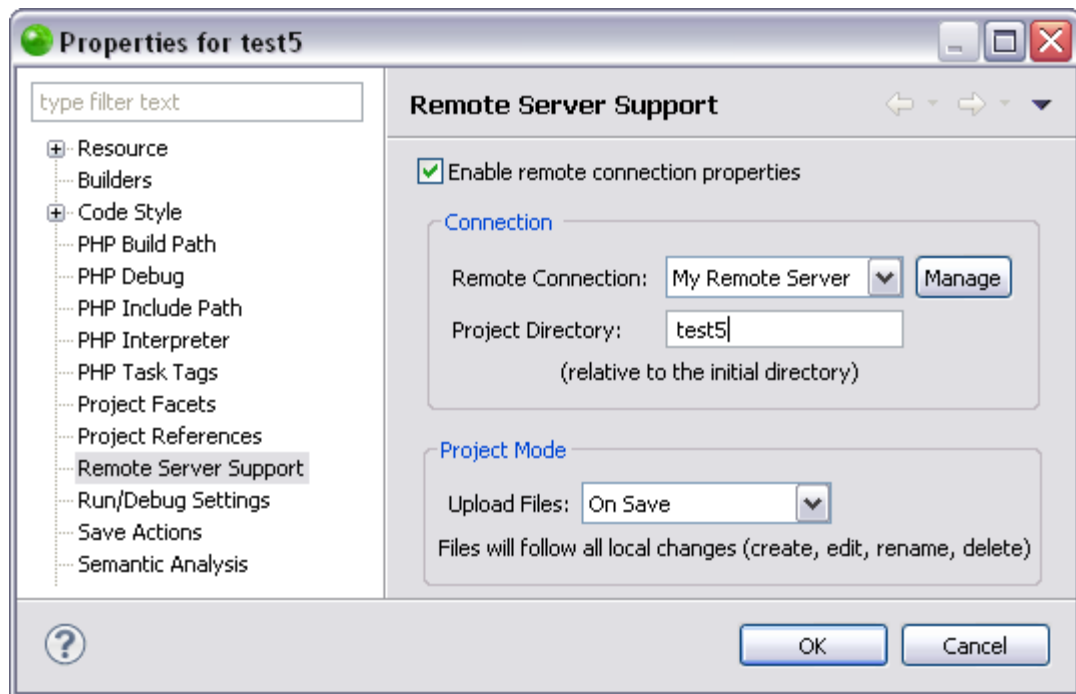
2. Select the 'Enable remote connection properties' check-box to modify the current settings on the page.

3. Select a remote connection from the Remote Connection drop down menu.

    You can manage the remote connection you have associated with your project by clicking **Manage**. If you do not have any remote connections already defined, you must add a Remote Connection Profile. See Working with Remote Connection Profiles for more information.

4. Click **OK** to enable your project as a remote project.

5. A message appears asking if you would like to upload all files in your project to the remote server. Click **Yes** to upload your project's files in the workspace to the remote server. To skip this step, click **No** and upload data from your workspace to the remote server when you would like to.

6. If you click **Yes** the Remote Server Upload dialog opens.
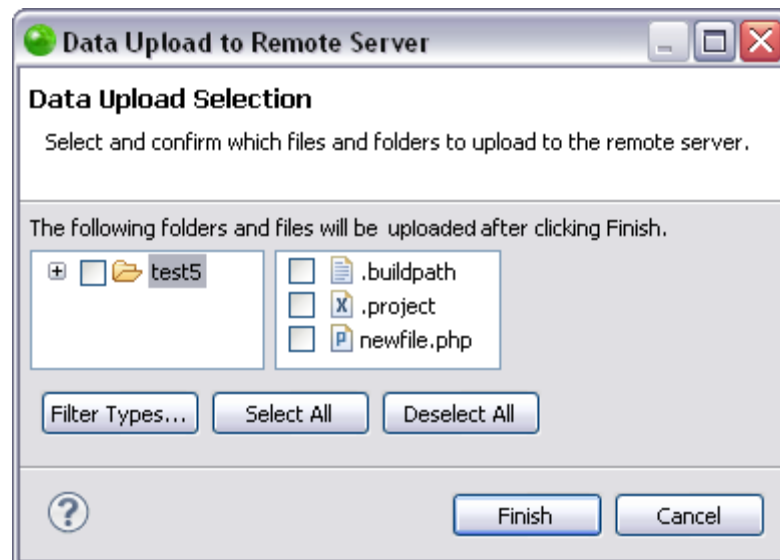
7. Expand the Project Directory to see all the folders within it that will also be uploaded.

8. Select the folders you want to upload to the remote server by selecting the check-box next to the folder name.

    By default all folders are selected.

Note:

If you open the Right Click Menu from a specific folder/file within the project instead of the project name, only the folders/files contained within those resources will be selected by default.

9. Click on a selected folder to view the included files.

    The individual files can also be selected/unselected to upload to the remote server. By default, all files within a folder are selected.

On this page you can also:

- Filter Types - Opens the Inclusion and Exclusion Patterns dialog. Defining exclusion/inclusion patterns allows you to filter which folders/files will be uploaded. For more information see Working with Inclusion and Exclusion Patterns.
- Select All - Selects all the folders and files.
- Deselect All - Deselects all the folders and files.

10. Click **Finish**.

The selected files are uploaded to the server.

To find out how to disable your project as a remote project see Disabling a Project as a Remote Project. Now that you are working with a remote project, you can Download Folders/Files from the Remote Server and Upload Folders/Files to the Remote Server.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Disabling a Project as a Remote Project

This procedure describes how to disable a project as a remote project. Once disabled, a project will no longer be associated with a remote server and will no longer allow you to transparently access your remote server and remote resources.

To disable a project as a remote project:

1. Select the project you would like to disable and go to **Project | Properties | Remote Project**. - Or - Select **Properties | Remote Project** from the Right Click Menu in your project directory.
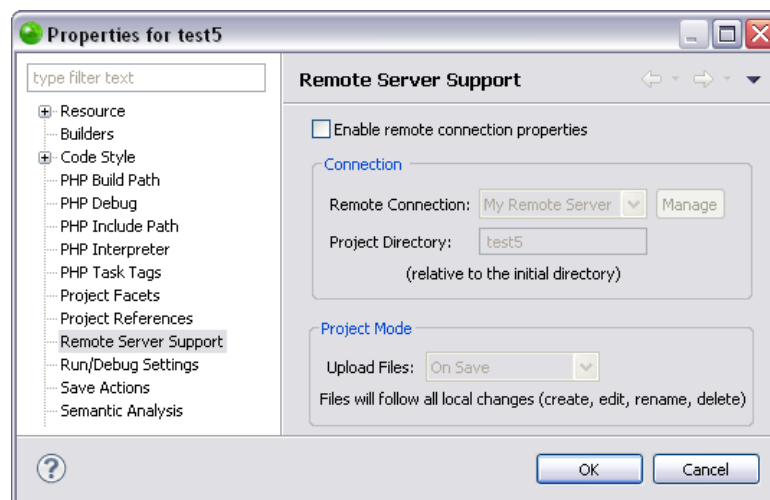
2. Deselect the 'Enable remote connection properties' check-box to disable the project as a remote project.

Your project is no longer associated with a remote server.

To enable your project as a remote project see <u>Enabling a Project as a Remote Project</u>.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Uploading Folders/Files to a Remote Server

Working with an enabled remote project allows you to upload folders, and the files contained within them, to the remote server specified in the associated Remote Connection Profile. This functionality allows you to ensure that your local project corresponds to the remote copy. Uploading data can be done manually (Manual mode) or automatically (On Save mode and On Run mode) according to your settings in the Remote Server Support Properties page.

Uploading to a Remote Server in Manual Mode

Manual mode allows you to upload data to the remote server only when you manually select to perform a data upload. If your project is set to Manual mode, data will never be transferred without you manually uploading the folders/files. This option is useful when you do not want your local and remote projects to always be synchronized.

Before beginning this procedure you must create a new project with Remote Server Support or enable your project as a remote project.

To upload folders and files in Manual mode:

1. From the Right Click Menu of your project select **Properties | Remote Server Support** - OR- go to **Project | Properties | Remote Server Support**.

   The Remote Server Support Properties page opens.
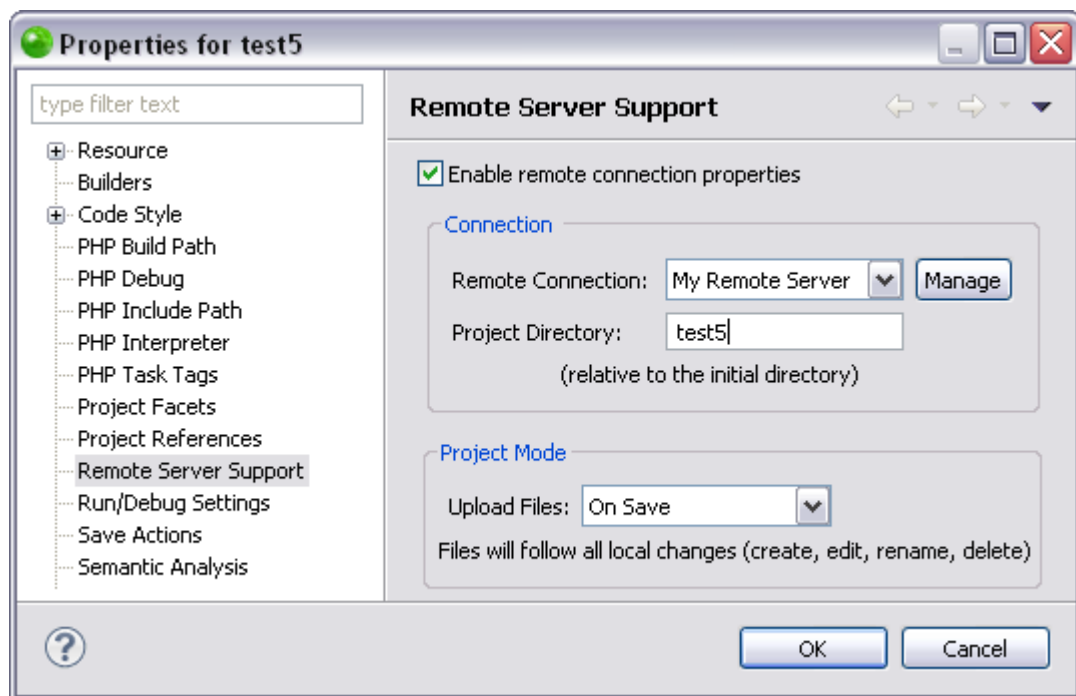
2. Select Manually from the Upload Files dropdown menu and click **OK**.

    Your project is set to Manual mode.

3. From the Right Click Menu of your project select **Remote Servers | Upload from Server**.

    The Data Upload Selection dialog opens.
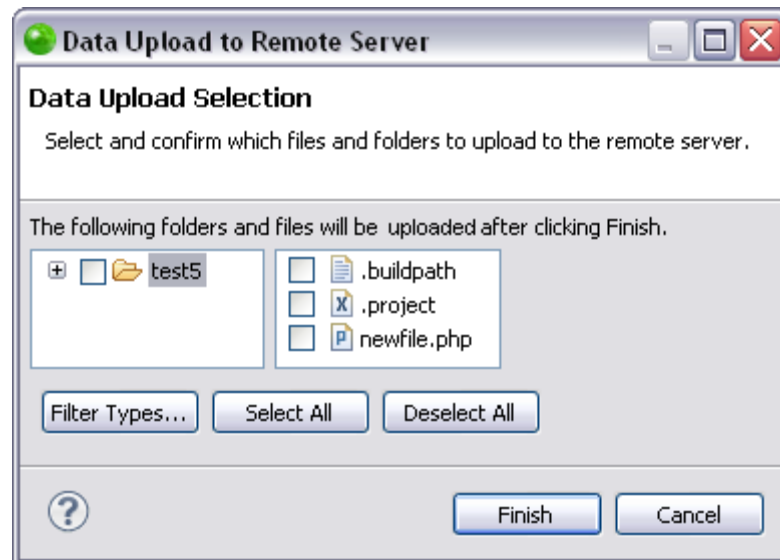
4. Expand the Project Directory to see all the folders within it that will also be uploaded.

5. Select the folders you want to upload to the remote server by selecting the check-box next to the folder name.

    By default all folders are selected.

6. Click on a selected folder to view the included files.

    The individual files can also be selected/unselected to upload to the remote server. By default, all files within a folder are selected.

    On this page you can also:

- Filter Types - Opens the Inclusion and Exclusion Patterns dialog. Defining exclusion/inclusion patterns allows you to filter which folders/files will be uploaded. For more information see <u>Working with Inclusion and Exclusion Patterns</u>.

- Select All - Selects all the folders and files.

- Deselect All - Deselects all the folders and files.

7. Click **Finish**.

    The selected files are uploaded to the server.

To keep your local project in sync with the remote copy on the remote server, you can also download folders/files from the remote server or change the remote server associated with your project. For more information see <u>Enabling a Project as a Remote Project</u> or Managing Remote Connections Profiles.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Uploading to a Remote Server in On Save Mode

Setting your project to On Save mode allows you to know that every time you save your project (or folder/file), or perform a change event, it will be uploaded to the remote server. This option is useful when you want to ensure that your local and remote project are always synchronized.

Important Note:

Deleting folders/files when in On Save mode automatically deletes them from the remote server. This will occur even if you have not saved the file.

**To upload folders and files in On Save mode:**

1. From the Right Click Menu of your project select **Properties | Remote Server Support** - OR- go to **Project | Properties | Remote Server Support**.
   The Remote Server Support Properties page opens.
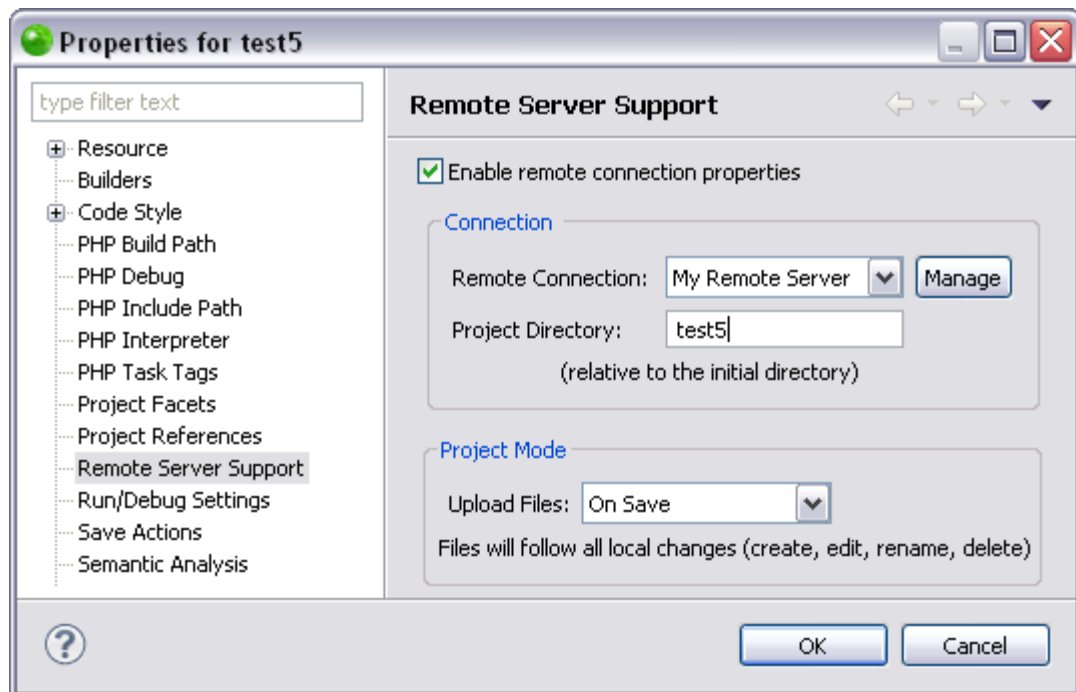
2. Select On Save from the Upload Files dropdown menu and click **OK**.

Your project is set to On Save mode.

3. Your project will be automatically uploaded to the remote server every time you save your project, or when a change event occurs. A change event is when:

- An event is changed
- A folder/file is changed
- A new folder/file is created
- A folder/file is deleted

Important Note:

Deleting folders/files when in On Save mode automatically deletes them from the remote server. This will occur even if you have not saved the file.

If you save only a specific file instead of the entire project, only the saved file will be uploaded to the remote server. To save the entire project click the save all icon from the menu bar.

Note:

The On Save mode does not allow you to filter which folders/files you are uploading.

To keep your local project in sync with the remote copy on the remote server, you can also download folders/files from the remote server.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Uploading to a Remote Server in On Run Mode

Setting your project to On Run mode allows you to know that your project is always uploaded to the remote server before you execute it. This option is useful when you do not want the project to always be automatically synced, but you would like an updated version of your project to be transferred to the remote server before you execute it.

When using On Run, Zend Studio will upload your selected project resources when you choose to execute your project, allowing you to remotely execute an updated copy of your project.

Your project specific debug settings can be configured in the PHP Debug Properties page.

To upload folders and files in On Run mode:

1. From the Right Click Menu of your project select **Properties | Remote Server Support** - OR- go to **Project | Properties | Remote Server Support**.

   The Remote Server Support Properties page opens.
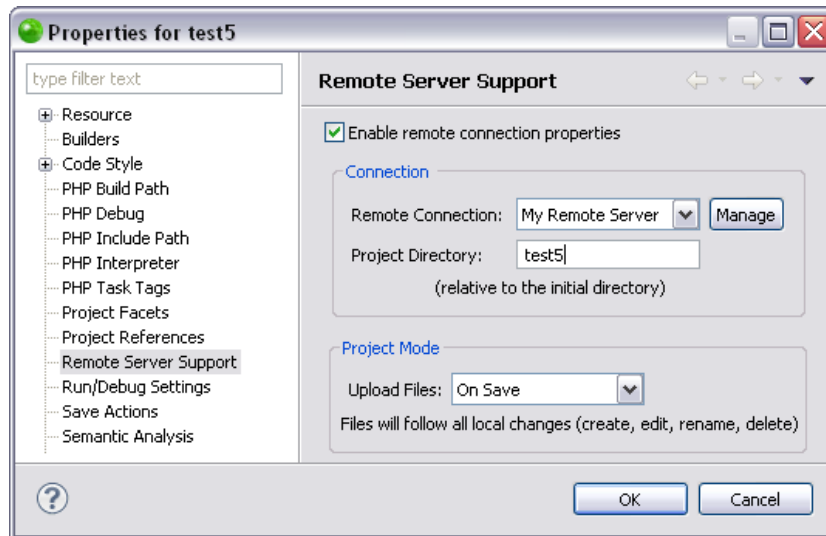
2. Select On Run from the Upload Files dropdown menu and click **OK**.

   Your project is set to On Run mode.

3. Select one of the available methods to debug or run your project. See Debugging Files and Applications or Running Files and Applications for more information.

   The Data Upload Selection dialog opens.
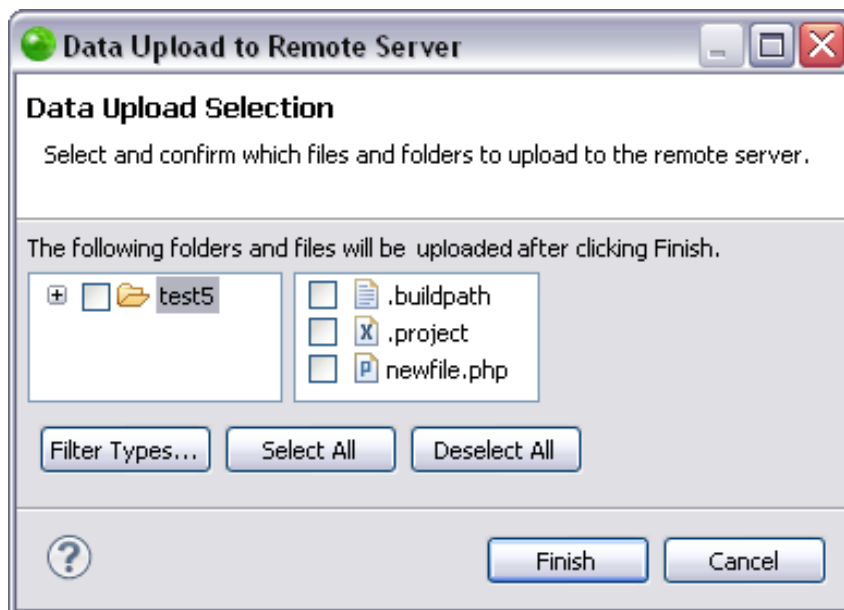
4. Expand the Target folder to see all the folders within it that will also be uploaded.

5. Select the folders you want to upload to the remote server by selecting the check-box next to the folder name.

6. Click on a selected folder to view the included files.

   The individual files can also be selected/unselected to upload to the remote server. By default, all files within a folder are selected.

   On this page you can also:

   - Filter Types - Opens the Inclusion and Exclusion Patterns dialog. Defining exclusion/inclusion patterns allows you to filter which folders/files will be uploaded For more information see <u>Working with Inclusion and Exclusion Patterns</u>.

   - Select All - Selects all the folders and files.

   - Deselect All - Deselects all the folders and files.

7. Click **Finish**.

   The selected files are uploaded to the remote server and your run/debug session starts.

To keep your local project in sync with the remote copy on the remote server, you can also <u>download folders/files from the remote server</u>.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Downloading Folders/Files from a Remote Server

Working with an <u>enabled remote project</u> allows you to download folders, and the files contained within them, to the remote server specified in the associated <u>Remote Connection Profile</u>. This functionality allows you to ensure that your local project corresponds to the remote copy. Data can only be downloaded from the remote server manually (Manual mode).

Downloading from a Remote Server in Manual Mode

Manual mode allows you to download data from the remote server only when you manually select to perform a data download. Data can only be downloaded manually.

To download folders and files in Manual mode:

1. From the Right Click Menu of your project select **Properties | Remote Server Support** -OR- go to **Project | Properties | Remote Server Support**.

   The <u>Remote Server Support Properties</u> page opens.
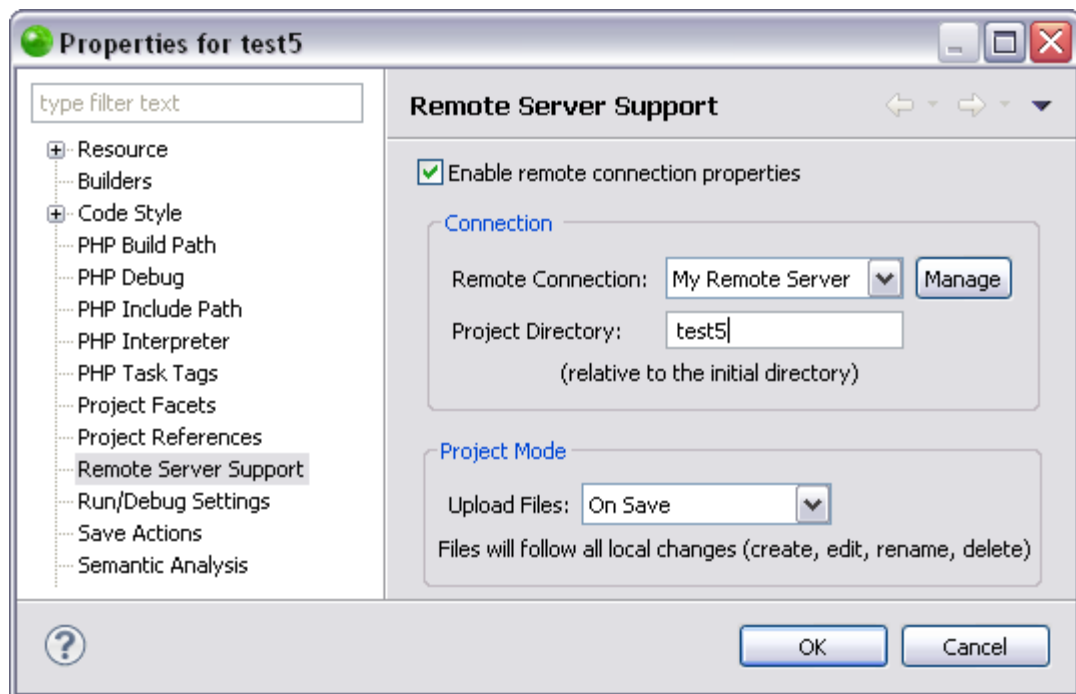
2. Select Manual from the Remote Server Transfer Mode dropdown menu and click **OK**.

   Your project is set to Manual mode.

3. From the Right Click Menu of your project select **Remote Servers | Download from Server**.

    The Data Download Selection dialog opens.
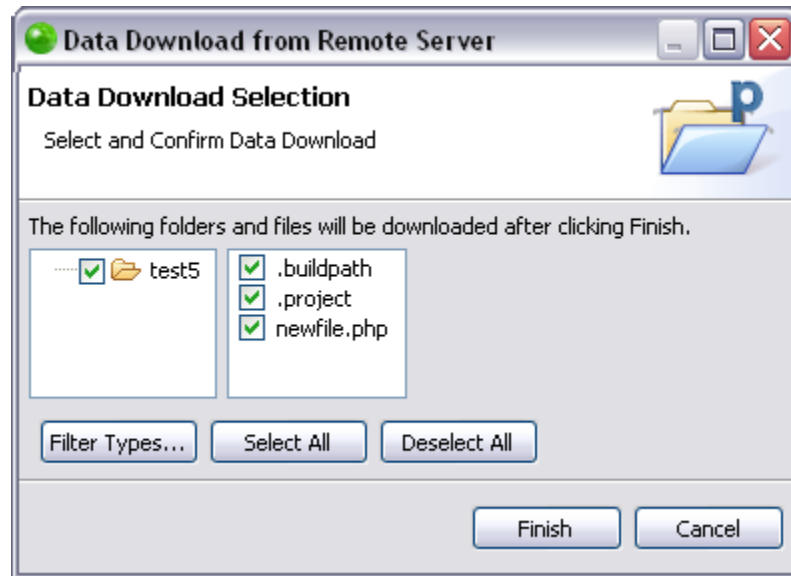
4. Expand the Project Directory to see all the folders within it that will also be downloaded.
5. Select the folders you want to download from the remote server by selecting the check-box next to the folder name.
6. Click on a selected folder to view the included files.

    The individual files can also be selected/unselected to download from the remote server.

    By default, all files within a folder are selected.

    On this page you can also:

*   Filter Types - Opens the Inclusion and Exclusion Patterns dialog. Defining exclusion/inclusion patterns allows you to filter which folders/files will be downloaded. For more information see *Working with Inclusion and Exclusion Patterns*.
*   Select All - Selects all the folders and files.
*   Deselect All - Deselects all the folders and files.

7. Click **Finish**.

    The selected files are downloaded from the server.

To keep your local project in sync with the remote copy on the remote server, you can also upload folders/files to the remote server or change the remote server associated with your project. For more information see Enabling a PHP Project to be a Remote Project or Working with Remote Connections Profiles.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Working with Inclusion/Exclusion Patterns

Inclusion and exclusion patterns allow you include or exclude all resources which match a defined pattern when transferring data to/from a remote server. This gives you the option of filtering the data you are transferring from your remote server to your project, or vice versa.
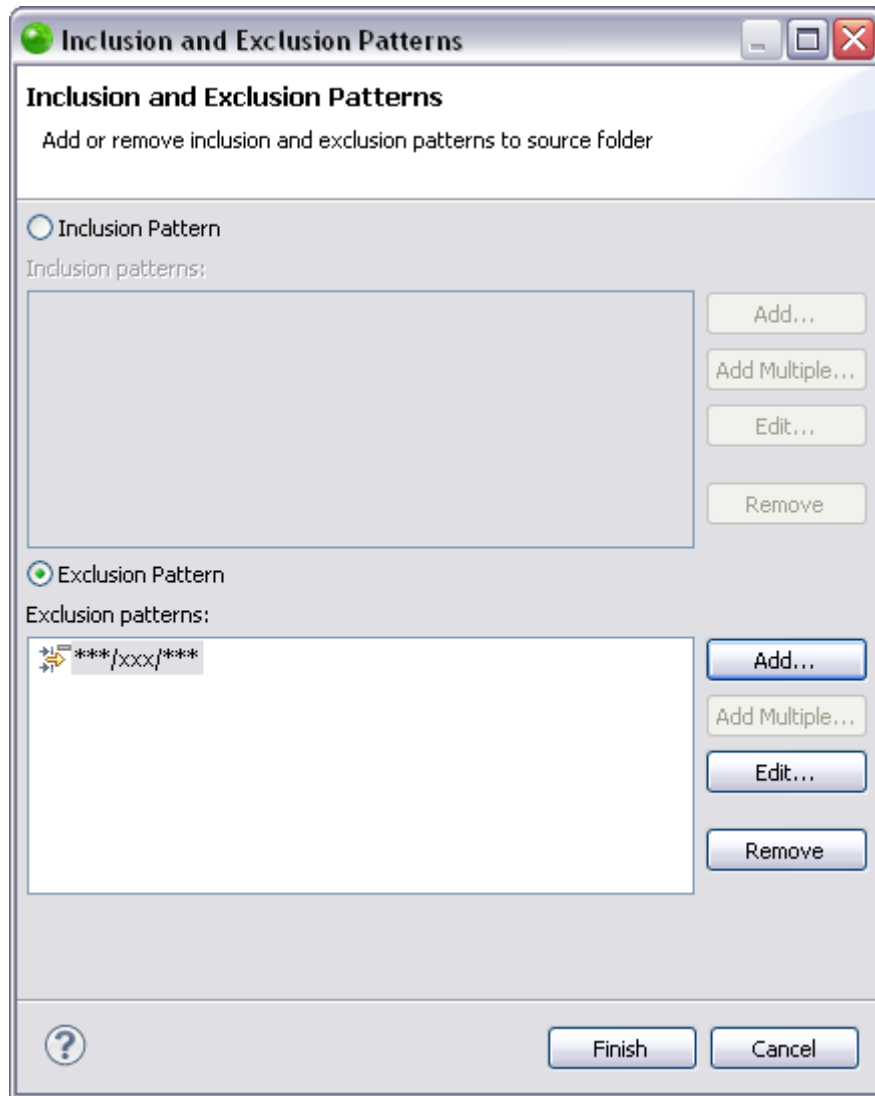
Selecting Inclusion/Exclusion Patterns

This procedure describes how to select an inclusion/exclusion pattern which filters your data transfer to/from a remote server. Selecting an inclusion or exclusion pattern can be accessed from the Data Upload Selection dialog or the Data Download Selection dialog. These dialogs appear when Creating a New PHP Project with Remote Server Support, Uploading Folders/Files to a Remote Server, or Downloading Folders/Files from a Remote Server.

To select an inclusion/exclusion pattern:

1. In the Data Download Selection dialog or the Data Upload Selection dialog click **Filter Types**.

   The Inclusion and Exclusion Patterns dialog opens.
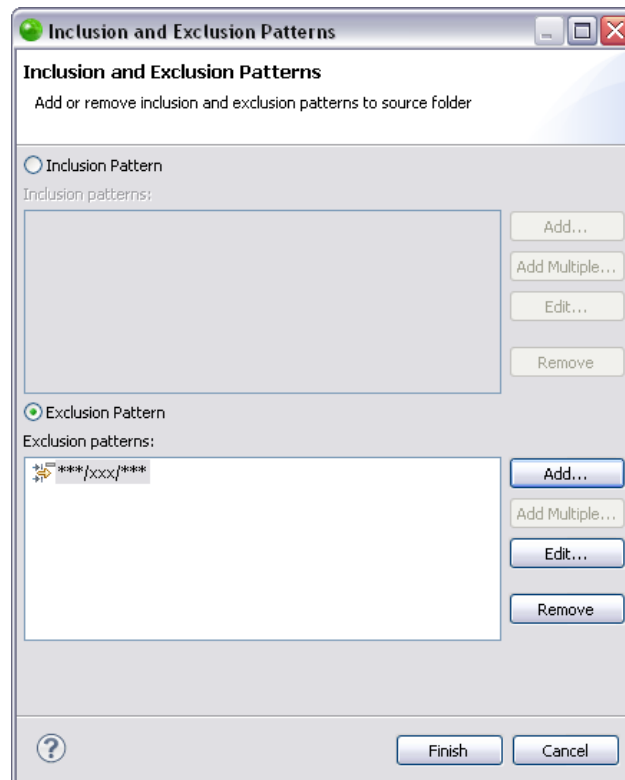
2. In the dialog you can select:

- Inclusion Pattern - The patterns you configure will be included in the data transfer.

- Exclusion Pattern - The patterns you configure will be excluded from the data transfer.

3. Select the patterns you would like to apply to your data transfer.

4. Click **Finish** to apply and save the changes.

You will be returned to the Data Download Selection or the Data Upload Selection dialog. For more information see [Creating a New PHP Project with Remote Server Support](#), [Downloading Folders/Files from a Remote Server](#) or [Uploading Folders/Files to a Remote Server](#).

Adding an Inclusion/Exclusion Pattern

This procedure describes how to add a new inclusion/exclusion pattern, which filters your data transfer to/from a remote server.

To add an inclusion/exclusion pattern:

1. In the Data Download Selection dialog click **Filter Types**.

   The Inclusion and Exclusion Patterns dialog opens.
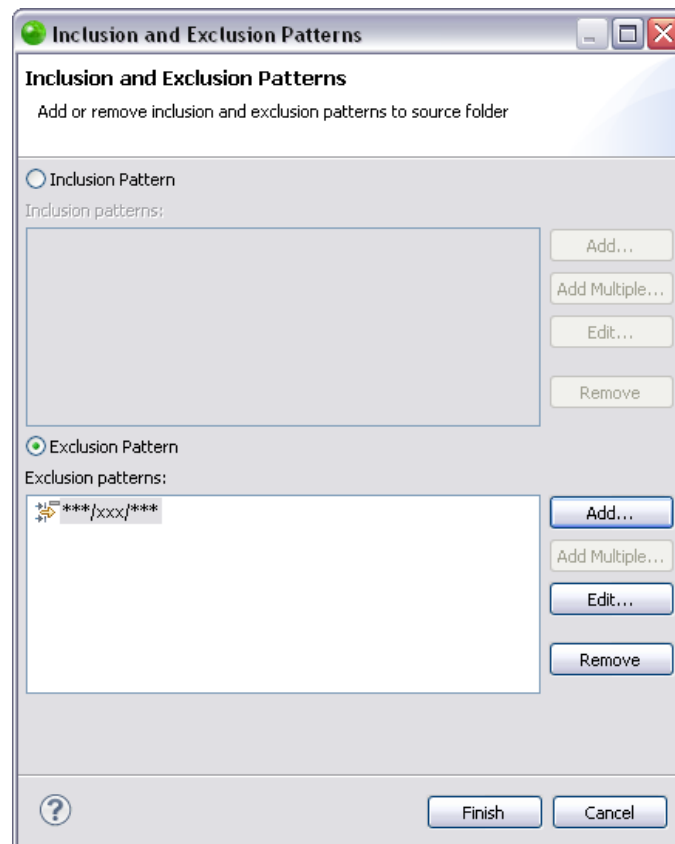
2. Select one of the following:

- Inclusion Pattern - The patterns you configure will be included in the data transfer.
- Exclusion Pattern - The patterns you configure will be excluded from the data transfer.

3. Click **Add**.

   The Add Inclusion Pattern or Add Exclusion Pattern dialog opens respectively.
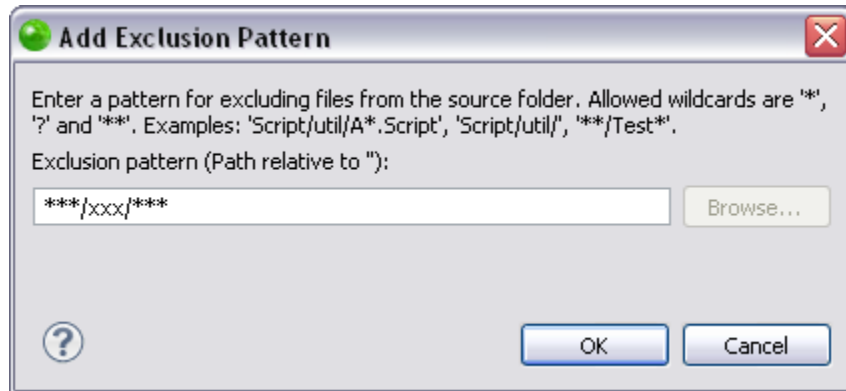
4. Enter the pattern you would like to use.

   The allowed wildcards are '*', '?', and '***'. For more information see Wildcards.

5. Click **OK** to save changes and return to the Inclusion and Exclusion Patterns dialog.

   To add another pattern, repeat this procedure.

6. In the Inclusion and Exclusion Patterns dialog click **Finish** to save the changes and return to the Data Download Selection or the Data Upload Selection dialog.

You can now edit a pattern, remove a pattern, continue creating a new PHP remote project, upload folders/files to the remote server, or download folders/files from the remote server.

Editing an Inclusion/Exclusion Pattern

This procedure describes how to edit an inclusion/exclusion pattern which filters your data transfer to/from a remote server. To edit a pattern you must have already added an inclusion/exclusion pattern.

To edit an inclusion/exclusion pattern:

1. In the Data Download Selection dialog click **Filter Types**.

    The Inclusion and Exclusion Patterns dialog opens.
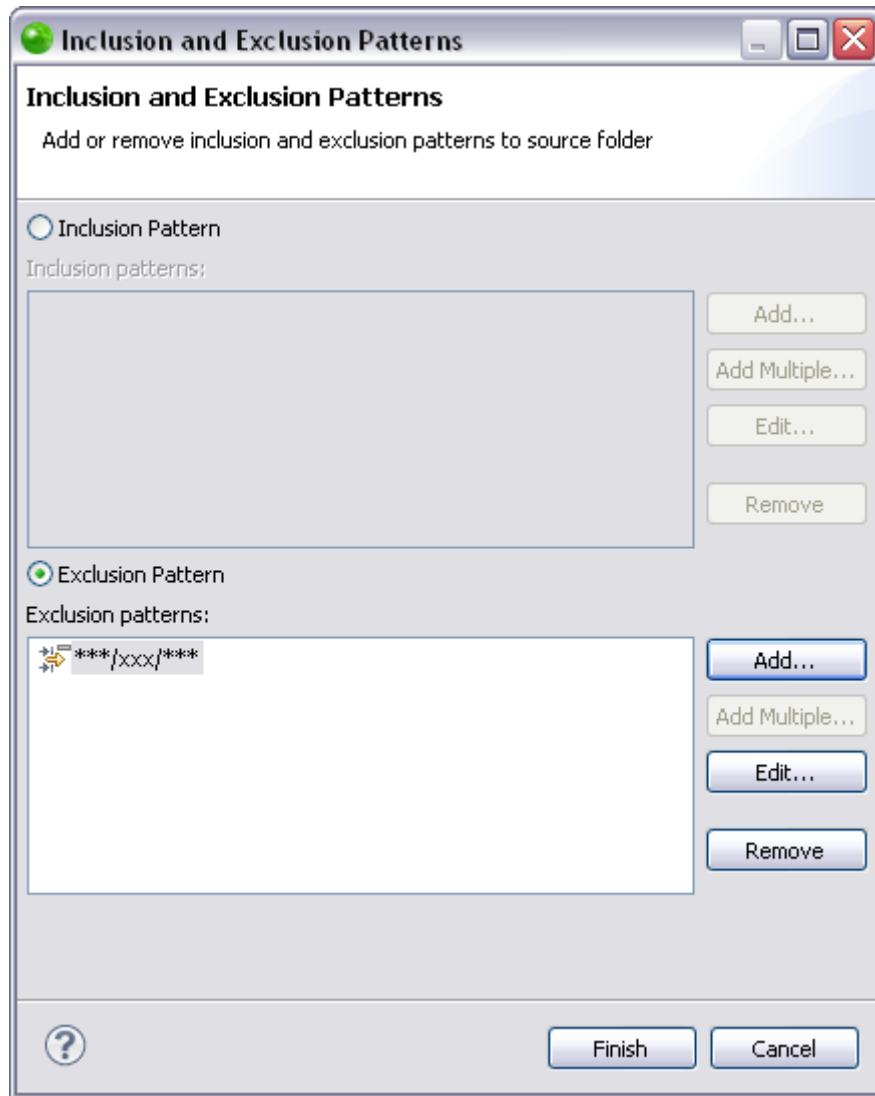
2. Select one of the following:

- Inclusion Pattern - The patterns you configure will be included in the data transfer.

- Exclusion Pattern - The patterns you configure will be excluded from the data transfer.

3. Select the pattern you would like to edit and click **Edit**.

The Edit Inclusion Pattern or Edit Exclusion Pattern dialog opens respectively.

4. Edit the pattern to your specifications and click **OK** to apply and save changes.

You are returned to the Inclusion and Exclusion Patterns dialog.

5. In the Inclusion and Exclusion Patterns dialog click **Finish** to save the changes and return to the Data Download Selection or the Data Upload Selection dialog.

You can now added an inclusion/exclusion pattern, edit a pattern or continue creating a new PHP remote project, upload folders/files to the remote server, or download folders/files from the remote server.

Removing an Inclusion/Exclusion Pattern

This procedure describes how to remove an inclusion/exclusion pattern which filters your data transfer to/from a remote server. To remove a pattern you must have already added an inclusion/exclusion pattern.

To remove an inclusion/exclusion pattern:

1. In the Data Download Selection dialog click **Filter Types**.

   The Inclusion and Exclusion Patterns dialog opens.

2. Select one of the following:

- Inclusion Pattern - The patterns you configure will be included in the data transfer.

- Exclusion Pattern - The patterns you configure will be excluded from the data transfer.

3. Select the pattern you would like to remove and click **Remove**.

   The pattern is deleted and will not be applied to the data transfer.

4. Click **Finish** to save the changes and return to the Data Download Selection or the Data Upload Selection dialog.

You can now added an inclusion/exclusion pattern, remove a pattern or continue creating a new PHP remote project, upload folders/files to the remote server, or download folders/files from the remote server.

Working with Mylyn Integration

The Task List contains two types of tasks:"Local Tasks" and shared "repository tasks" that are stored in a task repository such as Bugzilla or Jira. See how to create new tasks. Local tasks are typically contained in categories, which you can create by right-clicking on the task list and selecting **New** | **Category**. Repository tasks are contained in special categories that represent queries.

At the top of the Task List, you will find the following buttons and features:

- New Task - Create a new local or repository task.
- Synchronize - Update repository tasks with changes from the server.
- Task Presentation - Toggle between Scheduled and Categorized presentations.
- Focus on Workweek - See only tasks scheduled for this week.
- Find - search for a task by typing in words from the task summary
- Working set indicator - Indicates the currently active working set. Use the black arrow on the left to change the working set.
- Current task indicator - Indicates the currently active task. Use the black arrow on the left to re-activate a recently active task.

Task List Presentation

The task list supports several ways to present tasks. You can toggle between the following modes by using the "Task Presentation" button in the toolbar. Categorized - View tasks grouped by their category Scheduled - View tasks grouped by the "scheduled date" .

Icon Legend and Color Coding

See the legend below to interpret the icons and color coding in the task list. You can view this legend by selecting "Show UI Legend" from the menu that appears when you click the white down arrow next to the minimize button in the top right corner of the Task List view.

Creating new Tasks

You can create new tasks by clicking on the "New Task" button in the Task List's toolbar. This will open the "New Task" dialog and prompt you to select a repository. There are two main types of tasks:

- Local tasks
- Repository tasks

Local Tasks

You can use local tasks if you do not have a shared task repository or if you would like to create a private personal task that is local to your workspace.

To create a local task:

1. Select Local Task | **Finish** from the **New Task** dialog. You can then provide the following details about the task:

- Task Description - Your task is called New Task by default. Replace this with a brief description of your task.
- Priority - Set the priority of your task. This will affect the tasks' icon and order in the task list.
- Status - Set your task to "complete" or "incomplete". In your task list, completed tasks have a strike-through font and will appear lower in the list.
- URL - You can associate a URL with this task.
- "Retrieve Task Description from URL" button - Set the task description to the title of the associated URL (page)
- "Open with Web Browser" button - Open the URL in the integrated web browser
- Scheduled For - Set the date when you will work on this task. Tasks scheduled for today or a date in the past will appear in blue in your task list. Tasks scheduled for future days will appear in black. If your task list is in focused mode, only tasks for the current week will be visible (unless they have unread changes).
- Due - Set the date when your task must be completed. Overdue tasks and tasks due today will appear in red in your task list.
- Estimated Hours - Estimate the number of hours it will take to complete this task.
- Active - Displays the total time that you have worked on this task. Time is only recorded when this task is active and you are actively interacting with the system.
- Notes - Record your personal notes about this task.

Repository Tasks

You can create a new repository task when you would like to share information about the task with your team using a task repository such as Bugzilla or JIRA. To create a new repository task, click on the "New Task" button in the Task List's toolbar. You can then select the repository where you would like to create a task. If you don't see your team's task repository, you will need to configure it in the task repositories view. Once you have selected a repository, click "Next". If you are connecting to a Bugzilla repository, select a **Product** as a top-level category for your task and click "Finish". A new task editor will appear. If you are using Bugzilla, you can enter the following required information:

- Description - Enter a brief task description in the text box at the top (this box does not have a label).
- Component - Specify a "Component" to further categorize this task within the previously selected "Product".
- Description - Describe the task in detail. Optional
- You can specify additional information about your tasks in the "Attributes" section.
- Personal Planning - You can enter information in this section that will be local to your workspace and not available on your team's task repository. See Local Tasks" for more information about the personal planning fields.
- Assigned to - Specify who should work on the task. Type the first several characters of the person's email address, and then press ctrl+space to select the address from a list. A task can be assigned to only one person at a time.
- Add CC - Add the addresses of people who should be notified of changes to this task. You can add multiple addresses, separated by a comma, e.g. (mik.kersten@tasktop.com, steffen.pingel@tasktop.com).

When finished, click **Submit** to add your new task to your team's shared task repository.

Context

The context tab allows you to manage the context of resources associated with the task. You can view the context tab by selecting it in the lower left of the editor window.

Elements

This section lists the resources that are part of the tasks' context. Because the number of elements may be large, you can adjust the level of detail using the slider at the top of the Actions section. Sliding the control all the way to the left will show you all elements in your task context. As you slide it to the right, only the elements with a high level of interest will be displayed. You can manually remove elements from your task context by doing the following:

1. Right-Click **Remove From Context**.

You may choose to view all elements and exclude irrelevant items in this way before attaching the context to the task so that others can download it.

Actions

Element Detail Slider - Adjusts the minimum level of interest required for an element to be displayed in the Elements section.

- Attach Context - Attaches the context to the task so that it is available for download from the shared task repository. The context consists of the elements shown on the right.
- Retrieve Context - Replaces the current task context with one that is attached to the task in the shared task repository.

- ▪ Copy Context to... - Copy the task context to another task. That task will then have the same context as the current task.
- ▪ Clear Context - Removes all context information from the task.

Planning

Use the planning tab to access local information about the task that is private to your workspace. You can view the planning tab by selecting it in the lower left of the editor window. This tab contains a large area where you can enter personal notes about the task. See the local task section for more information about fields in the Personal Planning section.

Task-Focused Interface

The task-focused interface is oriented around tasks and offers several ways to focus the interface on only what is relevant for the currently active task. You can focus navigator views (e.g. Package Explorer, Project Explorer, Navigator) by toggling the "Focus on Active Task" button in the toolbar. When focused, the view will show only the resources that are "interesting" for the currently active task.

Alt+Click Navigation

To navigate to a new resource that is not a part of the active task's context, you can toggle "Focus on Active Task" off, browse to the resource, and then click "Focus on Active Task" again to see only relevant resources. A more efficient way to add new resources is to use Alt+Click navigation (Clicking the mouse while holding the Alt key). When a view is in Focused mode, you can Alt+Click a node to temporarily show all of its children. Once an element that was previously not interesting is selected with the mouse, it becomes interesting the other child elements will disappear. The clicked element is now a part of the task's context. Alt can be held down while clicking to drill down from a top-level element to a deeply nested element that is to be added to the task context. Multiple Alt+Click are supported so that you can add several elements to the task context. As soon as a normal click is made, uninteresting elements will disappear. Ctrl+Click (i.e. disjoint selections, use Command key on Mac) are also supported and will cause each element clicked to become interesting. The first normal click will cause uninteresting elements to disappear. Note that Ctrl+click elements will become interesting (turn from gray to black) but only the most recently-clicked one will be selected while Alt is held down. Focusing Editors

Some editors such as the Java editor support focusing. Clicking the Focus button in the toolbar will fold all declarations that are not part of the active task context.

Task-focused Ordering

When a task is active, elements that are interesting are displayed more prominently. For example, when you open the Java Open Type dialog (Ctrl+Shift+T), types that are interesting for the active task are shown first. Similarly, when you use ctrl+space to autocomplete a method name in a Java source file, methods that are in the task context are displayed at the top. Working Set Integration

When Focus is applied to a navigator view, the working sets filter for that navigator view will be disabled. This ensures that you see all interesting elements when working on a task that spans working sets. To enforce visibility of only elements within one working set, do the following: Set the view to show working sets as top-level elements. Use the Go Into action on the popup menu of the working set node in the view to scope the view down to just the working set. Open Task dialog

An Open Type style dialog is available for opening tasks (Ctrl+F12) and for activating tasks (Ctrl+F9). The list is initially populated by recently active tasks. The active task can also be deactivated via Ctrl+Shift+F9. This can be used as a keyboard-only alternative for multi-tasking without the Task List view visible. These actions appear in the Navigate menu.

Task Hyperlinking

In the task editor, comments that include text of the form bug#123 or task#123 or bug 123 will be hyperlinked. Ctrl+clicking on this text will open the task or bug in the rich task editor. To support hyperlinks within other text editors such as code or .txt files, the project that contains the file must be associated with a particular task repository. This is configured by right-clicking on the project and navigating to "Properties" > "Task Repository" and selecting the task repository used when working with this project. Reporting Bugs from the Error Log

Bugs can created directly from events in the Error Log view. This will create a new repository task editor with the summary and description populated with the error event's details. If the Connector you are using does not have a rich editor, the event details will be placed into the clipboard so that you can paste them into the web-based editor that will be opened automatically.

Developing with JavaScript

A range of JavaScript features are available within PHP files and projects, as well as in standalone JavaScript files.

The following tasks describe how to:

- Enable JavaScript Support in PHP Projects
- Set the JavaScript Build Path
- View JavaScript Elements in the Outline View
- Use and Configure JavaScript Content Assist

Note:

In order for Content Assist options to include elements from JavaScript libraries and referenced files and folders from outside of your project, you must first enable JavaScript Support in your project, and then set the JavaScript Build Path.

- Use and Configure JavaScript Mark Occurrences
- Use and Configure JavaScript Syntax Coloring
- Open JavaScript Types
- Set Up and Use Dojo Integration
- Work with jQuery JavaScript Library
- Work with Prototype JavaScript Library
- Work with ExtJS Library
- Work with JSDoc
- Debug JavaScript

Enabling JavaScript Support in PHP Projects

Enabling JavaScript support in PHP projects allows JavaScript libraries and external files to be referenced by the project and makes the elements within these resources available for operations such as Content Assist and Refactoring .

> **Note:**
> Once JavaScript support has been enabled for a project, you should set the project's JavaScript Build Path in order for the required resources to be made available to the project.

These procedures describe how to enable JavaScript support for new PHP projects, add support to existing projects, or disable JavaScript support.

Enabling JavaScript Support for New PHP Projects

To enable JavaScript support in new PHP Projects:

1. Go to File Menu and select **New | PHP Project**.

   -Or- In PHP Explorer view, right-click and select **New | PHP Project**.

   The new PHP Project wizard will launch.

2. Enter the required information in the various fields.

3. To enable JavaScript support, mark the 'Enable JavaScript support for this project' checkbox.

4.   Click **Finish**.

A new PHP Project will be created with full JavaScript support.

Enabling JavaScript Support for Existing PHP Projects

JavaScript libraries and features can be added to existing PHP projects in your workspace.

To enable JavaScript Support for existing PHP Projects:
In PHP Explorer view, right-click the project for which you want to enable JavaScript support and select **Configure | Add Java Script Support**.

JavaScript support will be enabled for the project.

Removing JavaScript Support

If you are not using JavaScript libraries or files in your project, you can remove JavaScript support for that project.

To remove JavaScript Support for existing PHP Projects:
In PHP Explorer view, right-click the project for which you want to enable JavaScript support and select **Configure | Remove JavaScript Support**.

JavaScript support will be removed from the project and no JavaScript libraries or external files will be available to the project.

Setting the JavaScript Build Path

The JavaScript build process scans all files, folders, projects and libraries that are on the project's JavaScript Build Path so that resources which are referenced in the project can be made available for Content Assist options and Refactoring operations . The required resources must therefore be included in the JavaScript Build Path.

JavaScript's Build Path allows you to select JavaScript resources to include/exclude from this process. Rather than automatically scanning all referenced projects and libraries, configuring the JavaScript Build Path allows you to select which resources will be scanned, and so can significantly speed up the build process.

Note:

JavaScript Support must be enabled for the project before you can set the JavaScript Build Path.

Configuring the Project's JavaScript Build Path

The JavaScript Build Path is determined by the resources included in the project's JavaScript Libraries properties page.

To configure the project's JavaScript Build Path:

1. In PHP Explorer view, right-click the required project and select Properties | JavaScript | Include Path.

   In this properties page you can add the following elements to the project's JavaScript Build Path:

   ▪ Libraries - Built-in and user defined libraries contain pre-written collections of JavaScript source files that have prototyped object/class definitions and JsDoc.

   ▪ Global Super Types - Each object/field/function in the added libraries will be added to the project's Global Scope. Every JavaScript file in the project will then have access to these objects/fields/functions.

   ▪ Source - Other folders/files/variables from your file system or linked from an external source.

   ▪ Projects - Other projects in the workspace or on your file system

   ### Libraries Tab

   If your project references a JavaScript library, it should be added to this list so that it's elements can be available for Content Assist options.

   Through the Libraries tab you can add Zend Studio 's built-in libraries, or create and add your own User Libraries.

> **Note:**
> The default Runtime Libraries (Script Language Libraries, Web Project support for JSDT and
> ECMA 3 Browser Support Library) are added to the JavaScript Build Path automatically.
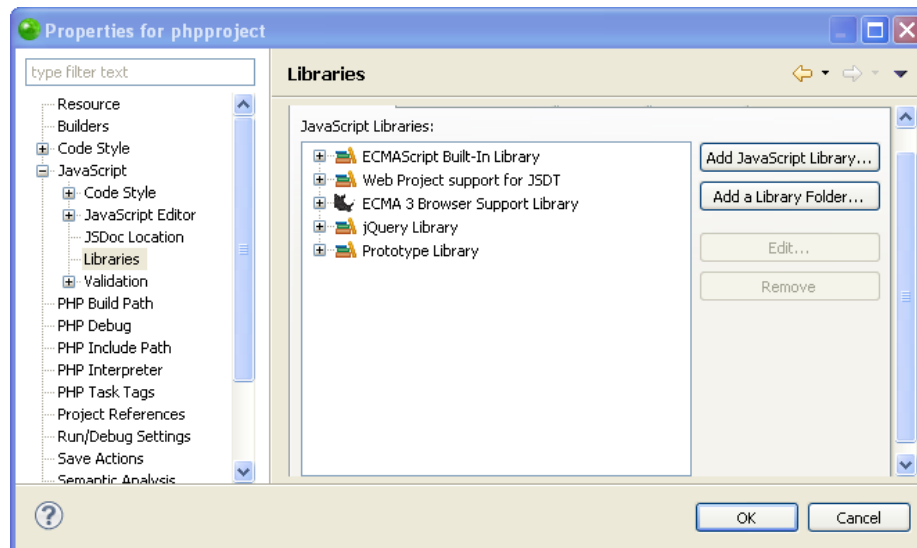
To add a built-in library to the JavaScript Build Path:

1. Click 'Add a Runtime Library...'.

   The Add Library dialog will appear.

2. Select the required library and click **Next** and **Finish**.

To add a User Library to the JavaScript Build Path:

1. Click 'Add a Runtime Library...'.

   The Add Library dialog will appear.

2. Select 'User Library' and click **Next**.

   A list of previously configured User Libraries will appear.
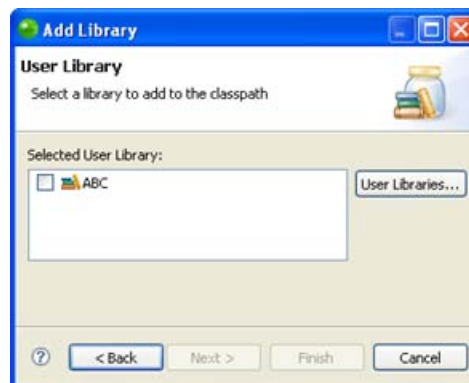
User Library Selection

3. If you have not yet configured your User Libraries:

   i. Click the User Libraries button to be taken to the JavaScript User Libraries preferences page (also accessible from **Window | Preferences | JavaScript | Include Path | User Libraries).**

   ii. Click **New**.

      The New User Library dialog will open.
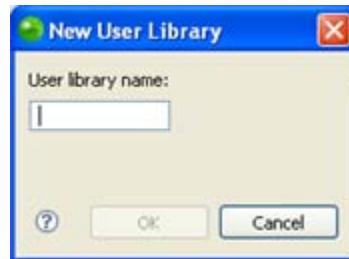
New User Library dialog

   iii. Enter a name for the library you would like to add and click **OK**.

      The library name will be added to the User Libraries list. This is just a placeholder name to which you will add the required files in order to create your own library.

   iv. Select the name from the list and click 'Add .js.'

   v. In the .js File Selection dialog, browse to the required JavaScript files on your file system and click **Open**.

   vi. The JavaScript files will be added to your User Library.
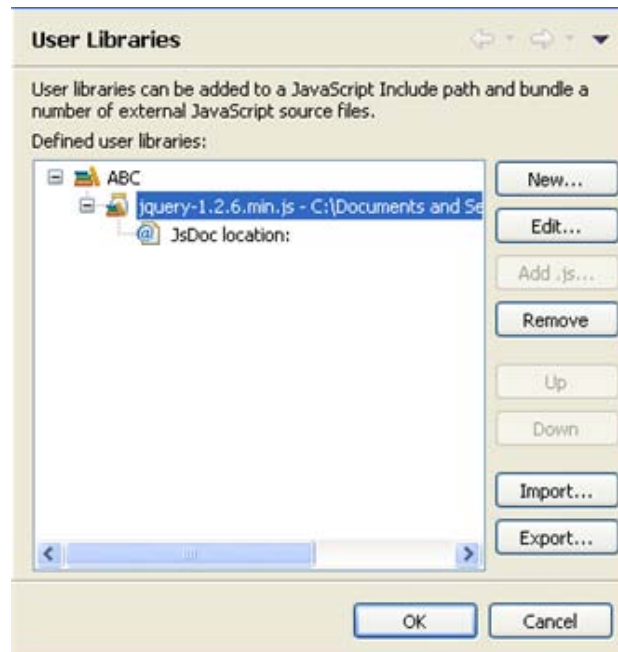
User Libraries Preferences

vii.   Click **OK** to return to the User Library selection list.

4. Mark the checkbox of the User Library you would like to add to the project's Build Path and click Finish.

The User Library will be added to the project's JavaScript Libraries Build Path list.

To add a library folder from your Workspace to the JavaScript Build Path:

1. Click the 'Add a Library Folder' button.

The library folder selection dialog will display a list of available folders.

2. Select the required folder(s) and click **OK**.

Global SuperType Tab

The project's Global Scope contains all the objects, fields and functions contained in the libraries which have been added to the JavaScript Build Path. Elements in the Global SuperType will be available to all PHP and JavaScript files within the project.
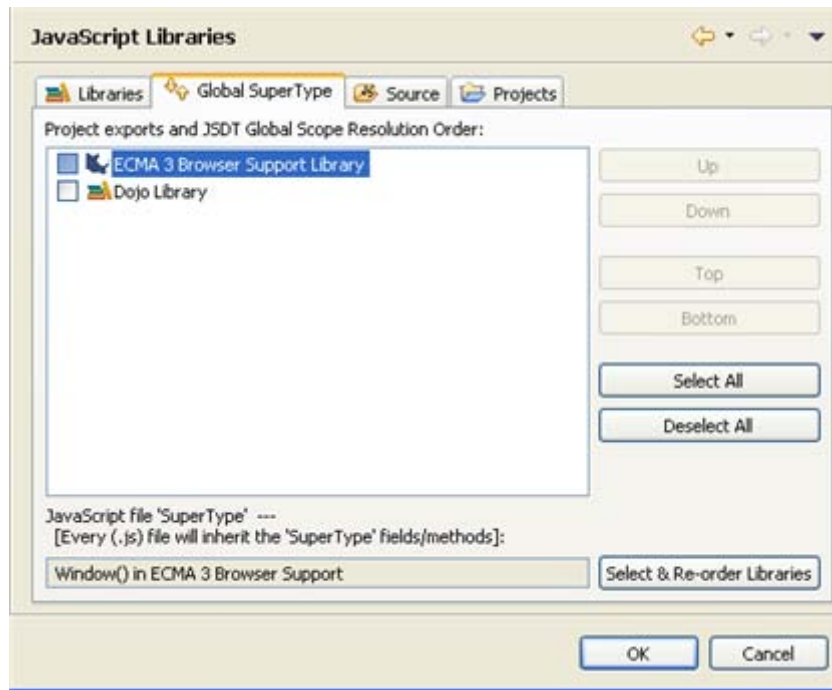
JavaScript Build Path - Global SuperType tab

In addition, the Global SuperType tab allows you to select a 'SuperType' class. All JavaScript elements which are not defined in their own class will automatically be added to this class. This will allow you to select these elements from the Content Assist list according to the SuperType Class name you selected.

To configure the SuperType class:

1. Click the 'Select & Re-order Libraries' button.

The JavaScript Sourcefile SuperType Selection dialog will open.

2. Select the class name for the SuperType class and click **OK**.

   All elements which are not defined in a class will be added to the selected class. Entering the class name will display these elements in the Content Assist list.

Source Tab

The Source tab allows you to include/exclude specific files, folders and variables from the JavaScript Build Path.
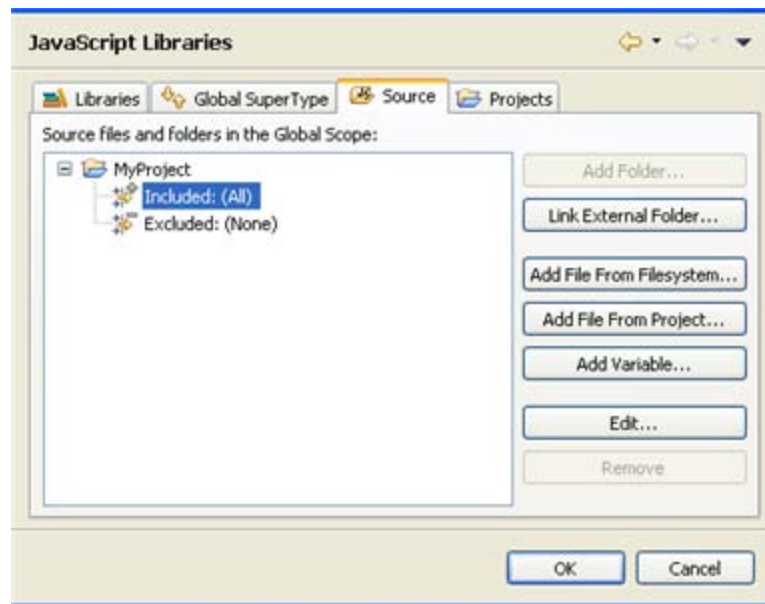
JavaScript Build Path - Source tab

To add a folder from your Workspace:

1. Click **Add folder**.

   The Source Folder Selection dialog will display.

2. Select the required folders and click **OK**.

To add an external folder:

1. Click **Add External Folder**.

   The Link Source dialog is displayed.

2. Click Browse to select a folder from your file system and click **OK**.

To add a JavaScript file from your filesystem:

1. Click **Add File from Filesystem**.

2. Select a JavaScript file and click **Open**.

To add JavaScript files from your Workspace:

1. Click **Add File From Project**.

2. Select the required file(s) and click **OK**.

Using a Classpath Variable allows you to point to a JavaScript library without having to specify it's local file system location. This is important when sharing resources in a team.

To add a Variable Classpath Entry:

1. Click **Add Variable**.
2. The list of Configured Classpath Variables will be displayed.

   If you have not configured any Classpath Variables:

   i. Click the Configure Variables button to open the JavaScript Include Path Variables preferences page (also accessible from **Window | Preferences | JavaScript | Include Path | Include Path Variables**).

   ii. Click **New**.

   The New Variable Entry dialog will appear.

   iii. Enter the Name and Path of the variable (click the File or Folder buttons to browse to the path).

   iv. Click **OK**.

   The Variable will be added to the Include Path Variables list.

   v. Click **OK** to return to the Variable Classpath Entry selection list.

3. Select the variable you would like to add to the project's JavaScript Build Path and click **OK**.

> **Note:**
> The Variable points to a folder. To choose an archive inside the folder, click Extend.

4. Click **OK**.

You can choose to include or exclude specific resources or file name patterns from within source folders which you have added.

To include/exclude specific resources from the JavaScript Build Path:

1. Expand the node next to the folder from which you would like to include/exclude resources.

   A list of the resources which are included and excluded will be displayed (by default, all resources within the folder are included).

2. Select the Included or Excluded list and click **Edit**.

   The Inclusion and Exclusion patterns dialog is displayed.

3. The Add Inclusion/Exclusion Pattern dialog is displayed.

4. Enter or select the required resources / pattern to Include/Exclude and click **OK**.

5. Click Finish.

All resources in the folder which match an Inclusion pattern but do not match an Exclusion pattern will be added to the JavaScript Build Path.
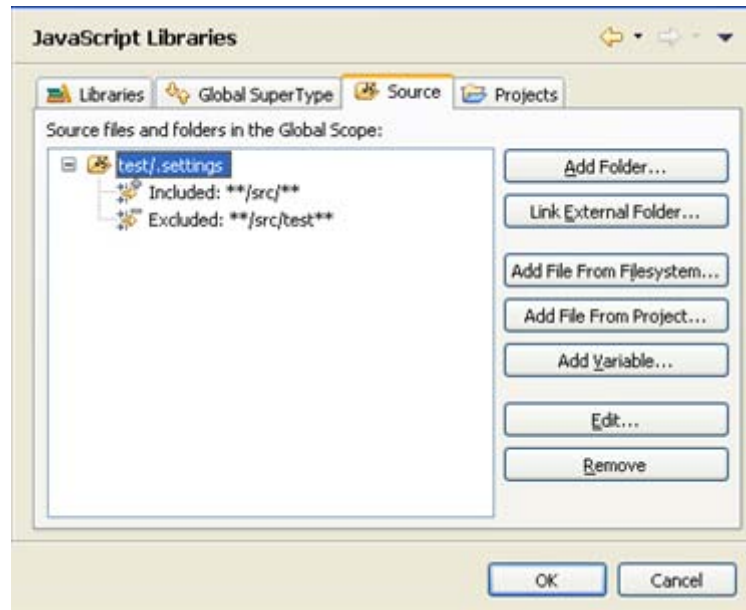
JavaScript Build Path - Source Tab - Exclusion List

Projects Tab

The Projects tab allows you to select projects from your Workspace to add to the JavaScript Build Path.
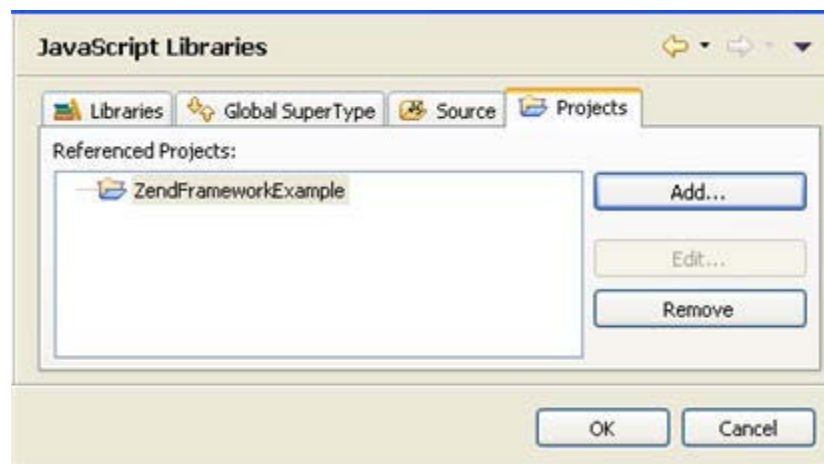
JavaScript Build Path - Projects Tab

To add a project to the JavaScript Build Path:

1. Click the 'Add..' button.

The Required Project Selection dialog is displayed.

2. Select the required projects and click **OK**.

The selected projects will be added to the JavaScript Build Path.

Note:

During the Build process, the referenced projects will be built before the project itself.

Saving your changes

Once you have made all necessary changes to your JavaScript Build Path, click **OK**.

Click **Add** next to the Inclusion or Exclusion patterns panes.

Your project will be rebuilt to reflect the changes, according to the settings you configured.
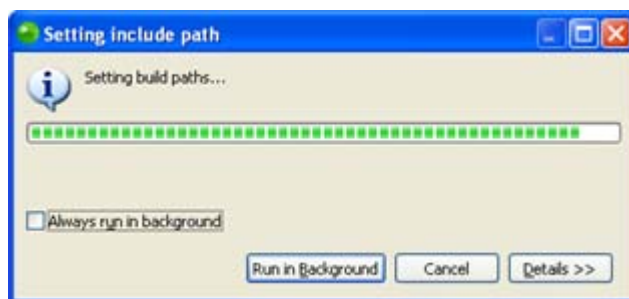
Setting Include Path dialog

Viewing JavaScript Elements in the Outline View

This procedure describes how to view JavaScript objects and elements in the Outline view.

To view your JavaScript objects in the Outline view:

1. Go to the Outline view.

   If it is not displayed, go to **Window | Show View | Outline**.

2. If your JavaScript objects are contained within a PHP file, click the Menu arrow ▽ on the Outline view's toolbar and select **html**.

   HTML and JavaScript objects contained within the file will be displayed in a tree view.
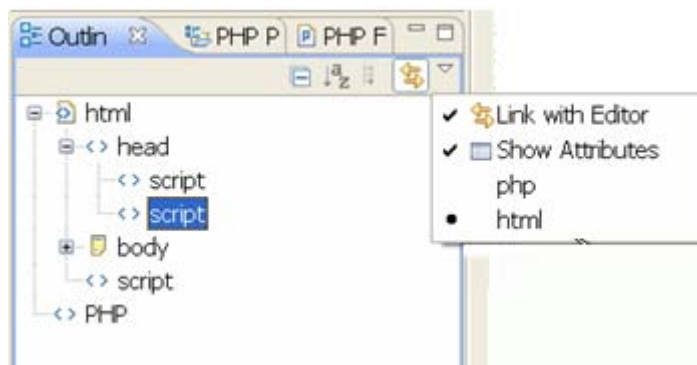
Outline view - HTML

3. Double-Clicking the <Script> node in the outline view will select the entire <script> element in the Editor.

Using JavaScript Content Assist

These procedures describe how to enable and configure JavaScript Content Assist options.

Accessing JavaScript Content Assist Options

To access JavaScript Content Assist options:

1. Type the relevant HTML and JavaScript tags:

```
<HTML>
<script type="text/javascript">

</script>
</HTML>
```

2. JavaScript Content Assist options will now be available.
3. Type the first few letters of the required element.
   The Content Assist list will be displayed (if the Content Assist list is not automatically displayed, press **Ctrl + Space** or go to your JavaScript Content Assist preferences page to configure your auto activation preferences).
4. Select the relevant option from the Content Assist window by double-clicking or pressing **Enter**.
5. If you selected a JavaScript class, type a period "." after the name of the class to display a Content Assist window with the classes' relevant functions and methods.

```
<HTML>
<script type="text/javascript">
window.
</scrip    alert()
</HTML>    atob()
           attachEvent()
           back()
           blur()
           btoa()
           captureEvents()
           clearInterval()
           clearTimeout()
           clientInformation
```

6. Select the required option to complete your JavaScript code.

JavaScript Content Assist Configuration

JavaScript Content Assist options can be configured from the JavaScript Content Assist preferences page.

To configure JavaScript Content Assist options:

1. Go to the JavaScript Content Assist preferences page, accessed from **Window | Preferences | Web | JavaScript | Editor | Content Assist**.

JavaScript Content Assist preferences page

2. Configure the following options, according to your preferences:

Insertion

- Completion Inserts/Completion Overwrites - Select whether choosing an item from the Content Assist list will cause new code to be entered or existing code to be overwritten.

- Insert single proposals automatically -If enabled, the content assist suggestion will be inserted automatically when only one content assist option exists

- Insert common prefixes automatically - If enabled, Content Assist will automatically insert the common prefix of all possible completions similar to Unix shell expansion. This can be used repeatedly, even while the Content Assist window is being displayed.

- Fill argument names on completion - If enabled, Content Assist will add arguments when completing a method.

- Guess filled function arguments - If enabled, Content Assist will fill the arguments with the best matching function, according to the context.

Sorting and Filtering

- Sort proposals - Select how the proposals should be sorted in the Content Assist list.

- Hide proposals not visible in the invocation context - If enabled, the Java element proposals are limited by the rules of visibility. For example, private field proposals of other classes would not be displayed.

- Show camel case matches - If enabled, camel case matches are displayed (e.g. NPE is expanded to NullPointerException).

- Hide forbidden references - If enabled, references to JavaScript elements forbidden by access rules are not displayed.

- Hide discouraged references - If enabled, references to JavaScript elements discouraged by access rules are not displayed.

- Hide deprecated references - If enabled, references to deprecated JavaScript elements are not displayed.

Auto-activation

- Enable auto activation - If enabled, the Content Assist list will automatically be displayed when the first letters of an element are typed.

Note:
If this is unmarked, you can display the Content Assist list by pressing Ctrl+Space.

- Auto activation delay - Determines the delay before the Content Assist box is automatically displayed.

- Auto-activation triggers for JavaScript - Sets the characters that determine JavaScript context.

- Auto-activation triggers for JSDoc: Sets the characters that determine JSDoc context.

3. Click **Apply** to apply your settings.

Using JavaScript Syntax Coloring

Enabling JavaScript Syntax Coloring

JavaScript Syntax Coloring can be applied to JavaScript code within a PHP file or a JavaScript file.

To enable JavaScript syntax coloring:

Enter the relevant JavaScript tags:

```
<HTML>
<script type="text/javascript">

</script>
</HTML>
```

Any code enclosed in the JavaScript tags will have Syntax Coloring applied to it, according to the configuration in the JavaScript Syntax Coloring Preferences page

```
<HTML>
<script type="text/javascript">
/**
 * This is a multiple-
 * line comment
 */
var index = 0;
var arr = [];

function push(elem) {
    // This comment may span only this line
    arr[index++] = elem;
}
</script>
</HTML>
```

JavaScript Syntax Coloring

JavaScript Syntax Coloring Configuration

JavaScript Syntax Coloring options can be configured from the JavaScript Syntax Coloring preferences page.

To configure JavaScript Syntax Coloring options:

1. Go to the JavaScript Syntax Coloring preferences page, accessed from **Window | Preferences | JavaScript | Editor | Syntax Coloring**.

2. Select the required item from the Syntax element list.
   To enable Syntax Coloring for the element, ensure the "Enable" checkbox is marked.
3. Select a color to apply to the text.
4. Select what formatting, if any, you would like to apply to the text (Bold, Italic, Strikethrough, Underline).

Note:

The Sample text box displays a preview of the different elements.

4. Click **Apply** and **OK** to apply and save your settings.

Using JavaScript Mark Occurrences

The JavaScript Mark Occurrences feature allows you to see where a variable, method or type is referenced within a JavaScript file.

To use Mark Occurrences:

Stand on a variable, method or type in your file.

All instances where the element is referenced within the file will be highlighted.

To configure Mark Occurrences settings:

1. To toggle mark occurrences, click the Toggle Mark Occurrences button on the toolbar - or- press **Alt+Shift+O**.

2. To configure further Mark Occurences preferences, go to the JavaScript Mark Occurrences preferences page, accessed from **Window | Preferences | Web | JavaScript | Editor | Mark Occurrences**.
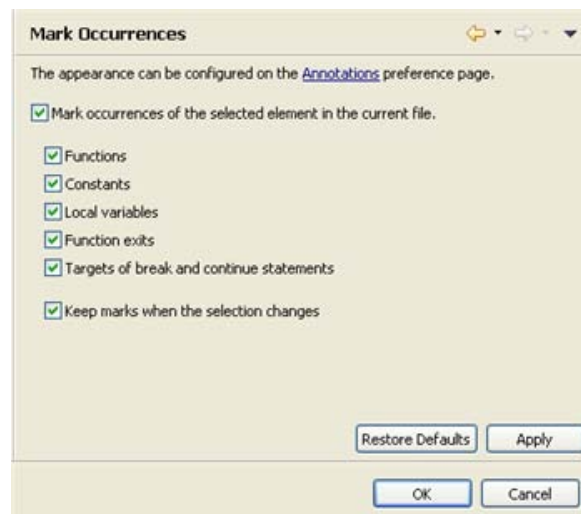
3. Mark the "Mark occurrences of the selected element" checkbox to enable the Mark Occurrences feature.

4. Select which element's occurrences will be marked by marking the checkboxes next to the required elements.

5. Mark the "Keep marks when the selection changes'" checkbox for marks to continue to be displayed once the cursor has been moved from the selected element.

6. Click **Apply** to apply your changes.

Note:

The appearance of marked occurrences can be configured in the Annotations preferences page (**Window | Preferences | General | Editors | Text Editors | Annotations**).

Opening JavaScript Types

The JavaScript Open Type functionality allows you to search for any JavaScript type in your workspace and opens an editor with the type's declaration.

To open a JavaScript Type:

1. Open the JavaScript Perspective by going to **Window | Open Perspective | Other | Javascript**.

2. Click the Open JavaScript Type ![icon] icon on the toolbar - or - from the menu bar go to **Navigate | Open JavaScript type**.

   The Open JavaScript Type dialog will open.

3. Enter the first few letters of the class which you would like to open. (You can also use wildcards such as * or ?, or camel case.)

   The list of available types will be filtered to match the expression you are entering.

Note:
If you have previously opened a Type which matches your expression, this will be displayed first above a separator line.
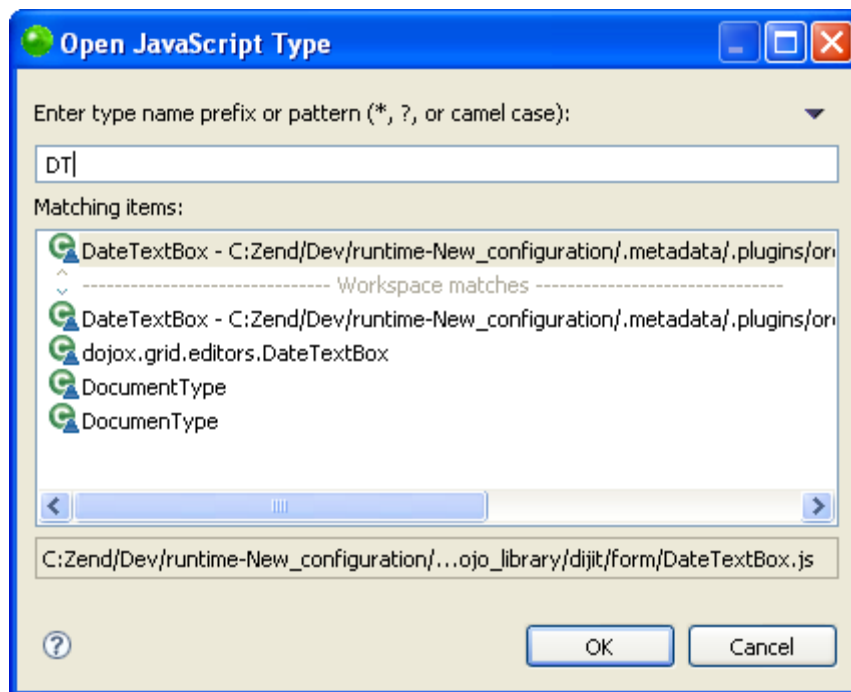
Open JavaScript Type dialog

4. Select the required class and click **OK**.

An editor will open with the required class's declaration.

Setting Up and Using Dojo Integration

Zend Studio's Dojo Integration functionality allows you to add the Dojo toolkit library to your project's Build Path so that its functions, classes and elements will be available for operations such as code completion and hover help.

See http://dojotoolkit.org/ for more information on the Dojo toolkit.

Setting Up Dojo Integration in PHP Projects

To enable Dojo integration you must first enable JavaScript Support in your PHP Project.

You then need to add the Dojo library to your project's Build Path by following the instructions under the Adding Built-in Libraries section of Setting the JavaScript Build Path topic, and selecting the Dojo library.

Setting Up Dojo Integration in Zend Framework Projects

The Dojo library can also be added to Zend Framework projects through the New Zend Framework Project wizard.

When creating a Zend Framework project, mark the 'Enable Dojo Support' checkbox in the Dojo Support category. This will create a link to the external Dojo toolkit library, allowing for Dojo development functionality.

Note:

This option will only be available if Zend Framework version 1.6 was selected.
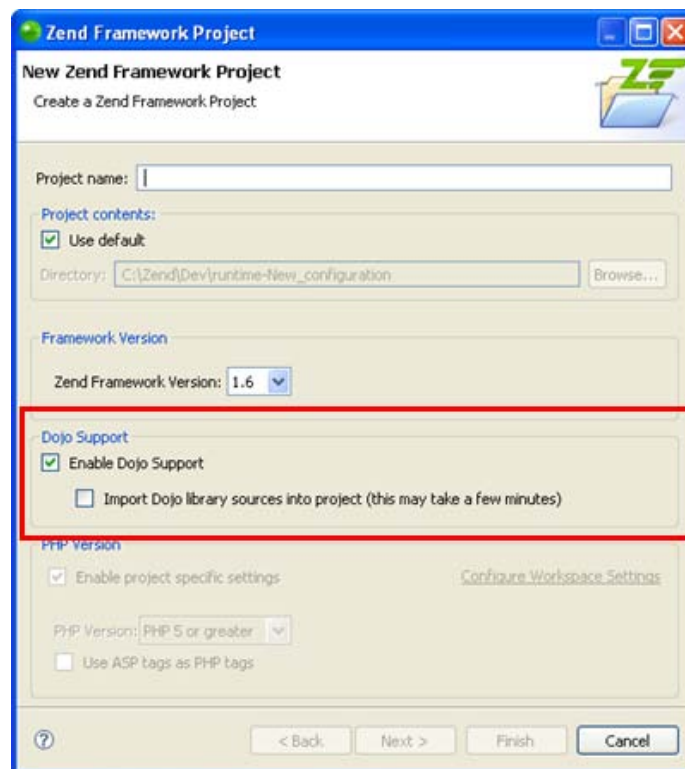
In addition, you can select to import the JavaScript library into your project. The link to the external library will still be maintained, so you do not have to add this Dojo library to your [JavaScript Build Path](#) (this will speed up the JavaScript build process).

Note:
The Dojo library may take a couple of minutes to import. If you are using Dojo for development purposes only, we recommend that you do not import it into your project.

If you did not enable Dojo support for the project during its creation, you can do so by adding the Dojo for Zend Framework library to the project's build path.

See the [Adding Built-in Libraries](#) section of [Setting the JavaScript Build Path](#) topic for more information.

Using Dojo Integration

Once Dojo libraries are included in your project's JavaScript Build Path, Dojo code completion options will be available to files within that project.

The following Dojo elements will be available in the content assist list:

- Dojo classes (including namespaces)
- Dojo object methods
- Dojo object properties
- Dojo global variables (e.g. dojo)
- Classes in dojo.require('r;')

Example:
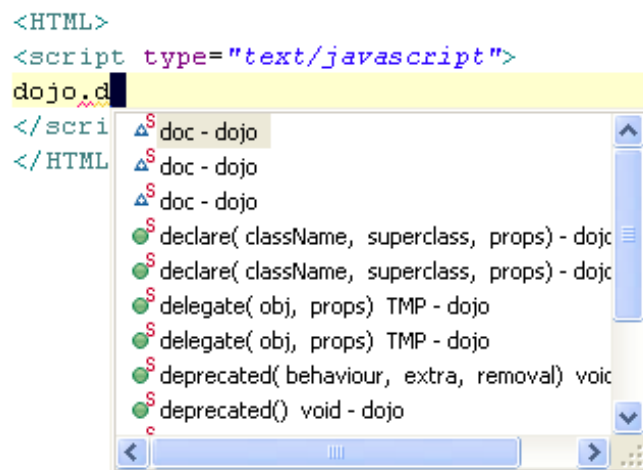
Dojo elements will also be displayed in the Outline view:
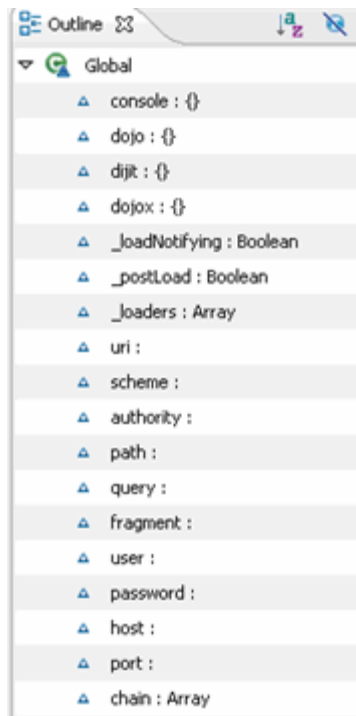
Dojo Elements in Outline view

Dojo requireModule

The dojo() view helper is intended to simplify setting up the Dojo environment and can be accessed from within Zend View files.

One of the methods available from the Dojo view helper is requireModule, which sets up a require method. Within Zend View files, content assist options will be available to this method:

Example:
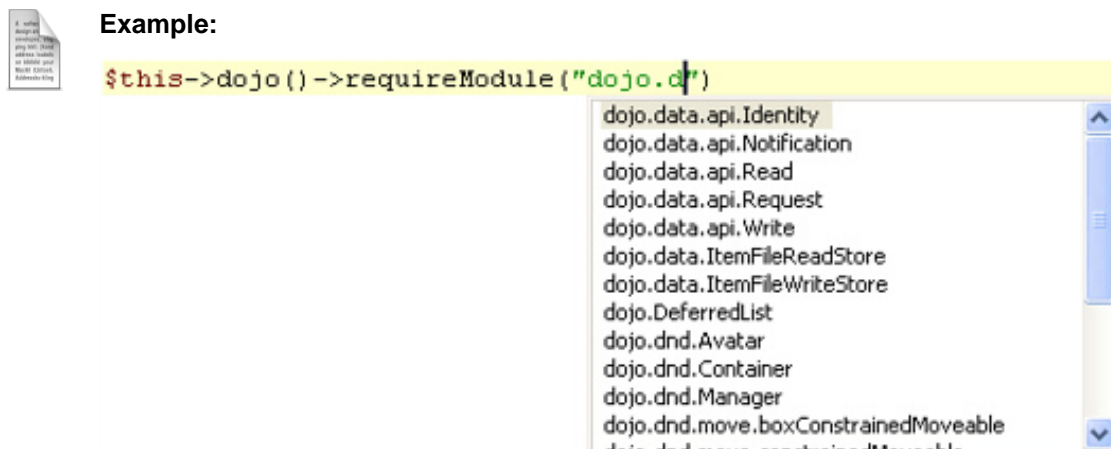
See http://framework.zend.com/manual/en/zend.dojo.view.html for more information.

Adding the Dojo JavaScript Library

This procedure describes how to add the Dojo JavaScript library to your project. Including JavaScript libraries in your project saves you time in writing and debugging code, as you are re-using debugged code.

JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see <u>Creating PHP Projects</u>.

To add the Dojo JavaScript Library to your project:

Select **Configure** from the right click menu of you project and select **Add Dojo**.
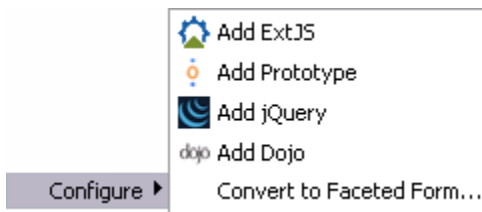
Functionalities such as Content Assist will now be able to access the Dojo JavaScript library. You can see the library in the "JavaScript Resources" node in your Project directory.

For information on Managing libraries see <u>Managing JavaScript Libraries</u>.

Working with jQuery JavaScript Library

Zend Studio's jQuery Integration functionality allows you to add the jQuery library to your project's JavaScript resources so that its functions, classes and elements will be available for functionalities such as Content Assist.

See http://jquery.com/ for more information on the jQuery library.

Adding the jQuery JavaScript Library

This procedure describes how to add the jQuery JavaScript library to your project. Including JavaScript libraries in your project saves you time in writing and debugging code, as you are re-using debugged code.

JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects.

**To add the jQuery JavaScript Library to your project:**

Select **Configure** from the right click menu of you project and select **Add jQuery**.
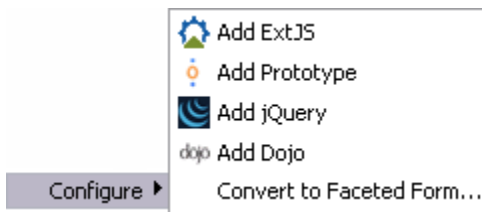
Functionalities such as Content Assist will now be able to access the jQuery JavaScript library. You can see the library in the "JavaScript Resources" node in your Project directory.

For information on Managing libraries see Managing JavaScript Libraries.

Working with Prototype JavaScript Library

Zend Studio's Prototype Integration functionality allows you to add the Prototype toolkit library to your project's JavaScript resources so that its functions, classes and elements will be available for functionalities such as Content Assist.

See http://www.prototypejs.org/ for more information on the Prototype toolkit.

Adding the Prototype JavaScript Library

This procedure describes how to add the Prototype JavaScript library to your project. Including JavaScript libraries in your project saves you time in writing and debugging code, as you are re-using debugged code.

JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects.

To add the Prototype JavaScript Library to your project:

Select **Configure** from the right click menu of you project and select **Add Prototype**.
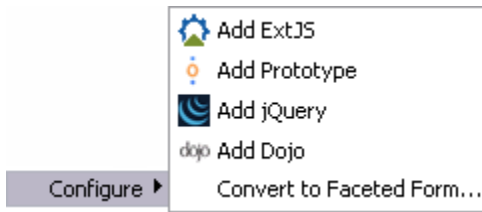
Functionalities such as Content Assist will now be able to access the Prototype JavaScript library. You can see the library in the "JavaScript Resources" node in your Project directory.

For information on Managing libraries see Managing JavaScript Libraries.

Working with ExtJS Library

Zend Studio's jQuery Integration functionality allows you to add the ExtJS library to your project's JavaScript resources so that its functions, classes and elements will be available for functionalities such as Content Assist.

See http://www.extjs.com/ for more information on the ExtJS library.

Adding the ExtJS Library

This procedure describes how to add the ExtJS library to your project. Including JavaScript libraries in your project saves you time in writing and debugging code, as you are re-using debugged code. JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects.

**To add the ExtJS JavaScript library to your project:**

Select **Configure** from the right click menu of you project and select **Add ExtJS**.
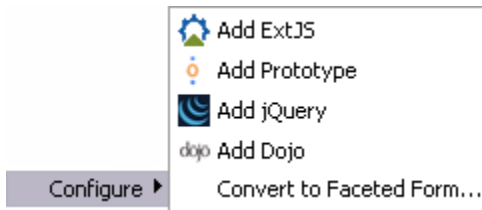
Functionalities such as Content Assist will now be able to access the ExtJS JavaScript library. You can see the library in the "JavaScript Resources" node in your Project directory.

For information on Managing JavaScript libraries see Managing JavaScript Libraries.

Working with JSDoc

The JSDoc functionality allows you to parse inline documentation for JavaScript source code in your project. The Documentation view shows the documentation of the JavaScript code including the parameters, returns, and exceptions. These are defined using tags ('@' - attributes). This documentation will also be added to functionalities such as Content Assist.

For more information on JSDoc see the JSDoc documentation (external link).

Opening the Documentation View

This procedure describes how to open the Documentation view, allowing you to use its functionality. The JSDoc functionality is only available for JavaScript files.

To open the Documentation view:

1. Open a JavaScript file by going to **File | New | JavaScript File**.

   The "New JavaScript File" dialog opens.

2. After selecting the parent folder for your file, enter a name in the "File name" text field and click **Finish**.

   A JavaScript file opens in the editor.

3. Go to **Window | Show View | Other | JavaScript | Documentation** and press **OK**.

   The Documentation view appears.

JSDoc allows you to add comments to your JavaScript code and makes the comments available in the Documentation view and in functionalities such as Content Assist.

Adding JSDoc Comments

This procedure describes how to add JSDoc Comments to JavaScript functions.

To add a JSDoc Comment:

1. In the line above the code for the JSDoc function, enter the JSDoc characters '/**' and press **Enter**.

2. Use the JSDoc tags to describe the properties of the JavaScript function (parameters, method, exceptions, etc.).

3. End the code for the JSDoc function with the standard '*/'
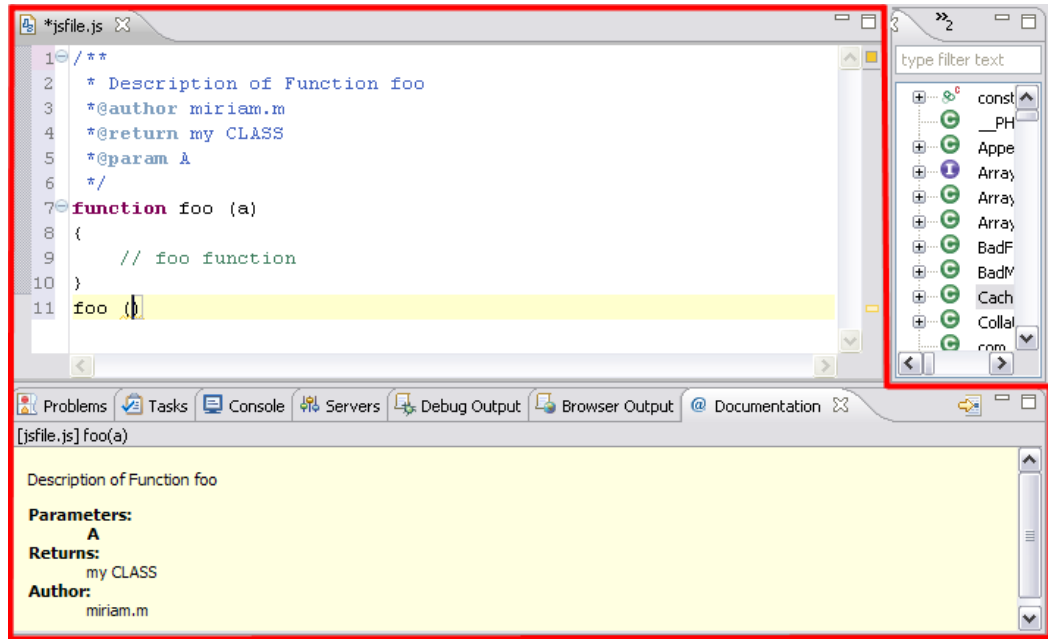
A JSDoc Comment will be created with the properties appearing in the Documentation View and in functionalities such as Content Assist. Whenever you use a function previously defined by the JSDoc tool, the properties will appear in the Documentation view.

When you use the previously defined JavaScript function, you can see the JSDoc documentation in the Documentation view by highlighting the function.

Managing JavaScript Libraries

Enabling JavaScript Libraries in your project allows libraries to be referenced by the project and makes the elements within these resources available for operations such as Content Assist and Refactoring.

Zend Studio allows you to define a list of libraries which have to be loaded before analyzing JavaScript code in your project. This is the equivalent to HTML users writing directly in the script what JavaScript should be loaded by the browser.

Each library can be different; It can be a plain list of JavaScript files, a zip file with JavaScript files, a database of JavaScript function signatures, or a running JavaScript engine. Therefore, each library has an ID associated with it. For example: "com.zend.jsdt.support.jquery" or "org.eclipse.jsdt.system". The ID's uniquely identify the kind of JavaScript library being referred to so that Zend Studio can internally locate the right mechanism to load the library's contents.

JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects.

 JavaScript Libraries allows you to do the following:
- Quickly Add a Predetermined JavaScript Library
- Add a JavaScript Library
- Add a Library Folder to JavaScript Libraries
- Edit JavaScript Libraries
- Remove JavaScript Libraries

The supported JavaScript libraries are:

- [Dojo Library](#)
- ECMA 3 Browser Support Library - A standard JavaScript library.
- Internet Explorer Library - A JavaSript library specialized for Internet Explorer users.
- [jQuery Library](#)
- [ExtJS Library](#)
- Mozilla Firefox Library - A JavaSript library specialized for Mozilla Firefox users.
- [Prototype Library](#)
- User Library - Allows you to create or import a JavaScript library into your project.
- ECMAScript Built-In Library - A standard JavaScript library.
- Web Project support for JSDT - A standard JavaScript library.

Adding a JavaScript Library

This procedure describes how to add a JavaScript library to your project. Including JavaScript libraries in your project saves you time in writing and debugging code, as you are re-using debugged code, and is available in functionalities such as Content Assist.

JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects.

Adding a Predetermined JavaScript Library

This procedure describes how to add a predetermined JavaScript library to your project, allowing you to include debugged code for features such as Content Assist. This time saver allows you to add a JavaScript library without using the properties menu.

To add a predetermined JavaScript Library to your project:
Select **Configure** from the right click menu of you project in your project directory and select from the list of available libraries.
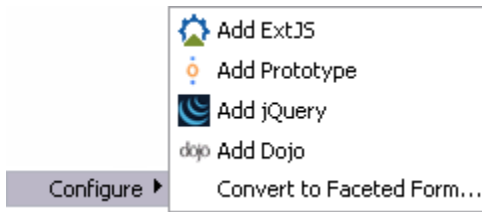
For more information on one of the predetermined JavaScript libraries see Setting Up and Using Dojo Integration, Working with jQuery JavaScript Library, Working with Prototype JavaScript Library, or Working with ExtJS Library.

Adding a JavaScript Library

This procedure describes how to add a JavaScript library using the Properties page. Adding a library through the Properties page allows you to add predetermined libraries and user libraries to your project.

To add a JavaScript Library to your project:
1. Select **Configure** from the right click menu of you project in your project directory and select **Convert to JavaScript Project**.

2. Go to **Project | Properties | JavaScript | Include Path - or -** Select **Properties | JavaScript | Include Path** from the Right Click Menu of the project folder in your project directory.

Your project's properties page opens.

2. In the JavaScript Libraries Properties page click **Add JavaScript Library**.

The "Add JavaScript Library" dialog opens with a list of the available JavaScript libraries.
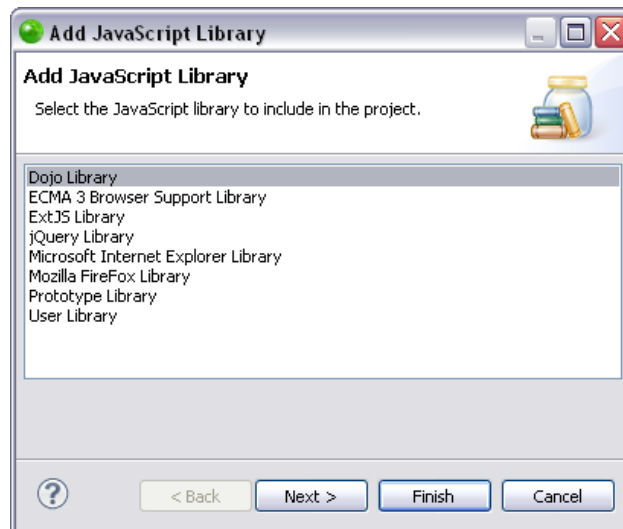
3. Select a JavaScript Library and click **Next**.

   Your JavaScript library has been added to your project.

4. To apply changes click **Finish**.

Functionalities such as Content Assist and Refactoring will now be able to access the JavaScript library.

You can see the library in the "JavaScript Resources" node in your Project directory.

For information on Managing libraries see [Managing JavaScript Libraries](#).

Adding a JavaScript User Library

This procedure describes how to add a JavaScript User library to your project. Including a JavaScript User library in your project saves you time in writing and debugging code, as you are re-using debugged code, and is available in functionalities such as Content Assist.

JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects. .

To add a new JavaScript Library to your project:

1. Select **Configure** from the right click menu of you project in your project directory and select **Convert to JavaScript Project**.

2. Go to **Project | Properties | JavaScript | Include Path - or -** Select **Properties | JavaScript | Include Path** from the Right Click Menu of the project folder in your project directory.

   Your project's properties page opens.
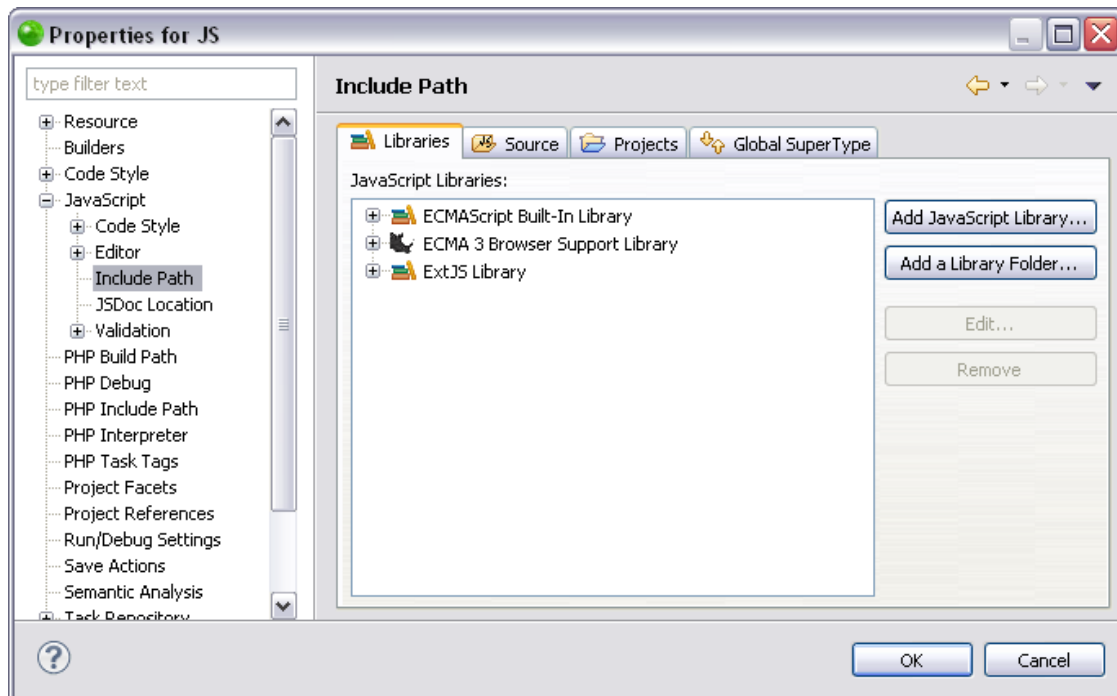
2. In the JavaScript Libraries Properties page click **Add JavaScript Library**.

   The "Add JavaScript Library" dialog opens with a list of the available JavaScript libraries.
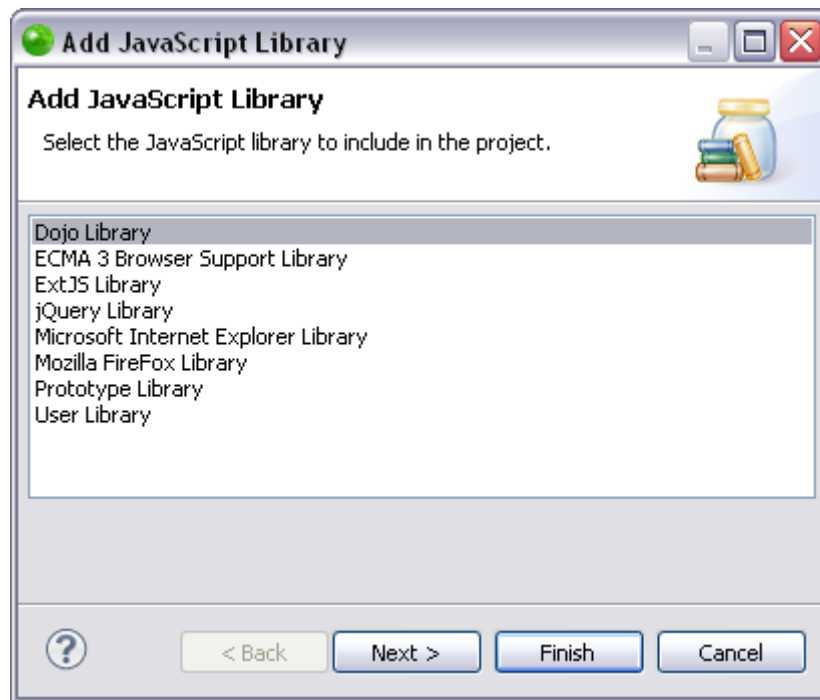
3. Select "User Library" and click **Next**.

   The "User Library" dialog opens.

4. Click **Configure User Libraries**.

   The "User Libraries" preferences page opens.
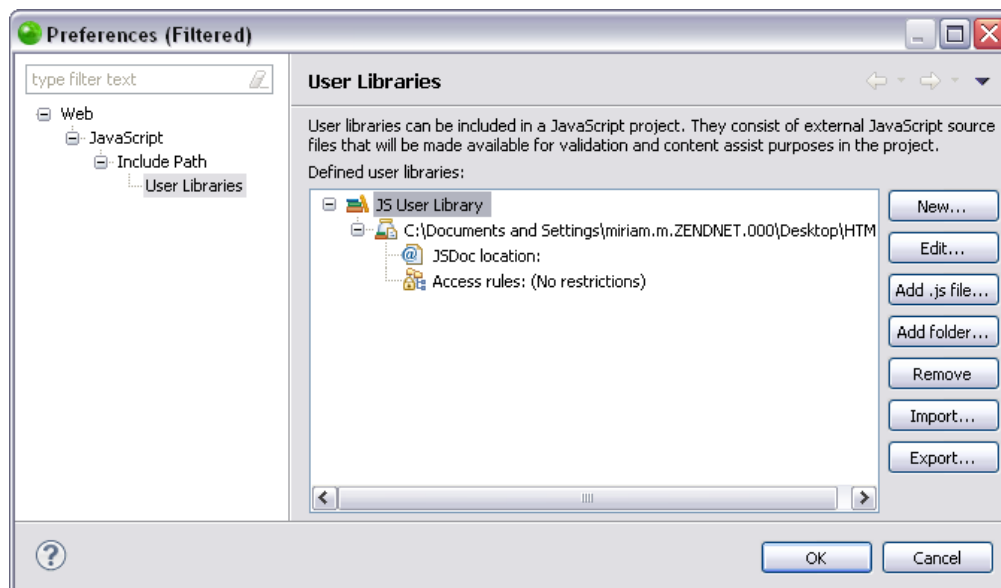
5. Click **New**.

    The "User library name" dialog opens.

6. Enter the name of your user library and click **Ok**.

    Your user library now appears in the list of JavaScript user libraries.

7. The User Libraries preferences page allows you to do the following using the buttons in the screen:

- Use the **Edit** button to edit the name of the JavaScript user library.

- Use the **Add .js file** to browse and select a .js file to add to your JavaScript user library.

- Use the **Add folder** button to browse and select a folder to add to your JavaScript user library.

- Use the **Remove** button to remove a JavaScript user library.

- For information about the **Import** and **Export** buttons see Exporting JavaScript User Libraries and Importing JavaScript User Libraries.

- For information about Access rules see Editing Access Rules.

8. To apply changes click **Finish**.

Functionalities such as Content Assist will now be able to access the JavaScript user library. You can see the library in the "JavaScript Resources" node in your Project directory.

For information on Managing JavaScript libraries see Managing JavaScript Libraries.

Adding a Library Folder to JavaScript Libraries

This procedure describes how to add a library folder to a the JavaScript libraries in your project. Library folders are resources containing source code that you would like to be available to functionalities such as Content Assist, but which are contained in files rather than libraries. Including JavaScript libraries and library folders in your project saves you time in writing and debugging code, as you are re-using debugged code.

JavaScript libraries can only be added to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects.

1. To add a JavaScript library folder to your project:

1. Select **Configure** from the right click menu of you project in your project directory and select **Convert to JavaScript Project**.

2. Go to **Project | Properties | JavaScript | Include Path - or -** Select **Properties | JavaScript | Include Path** from the right click menu of the project folder in your project directory.

Your project's properties page opens.
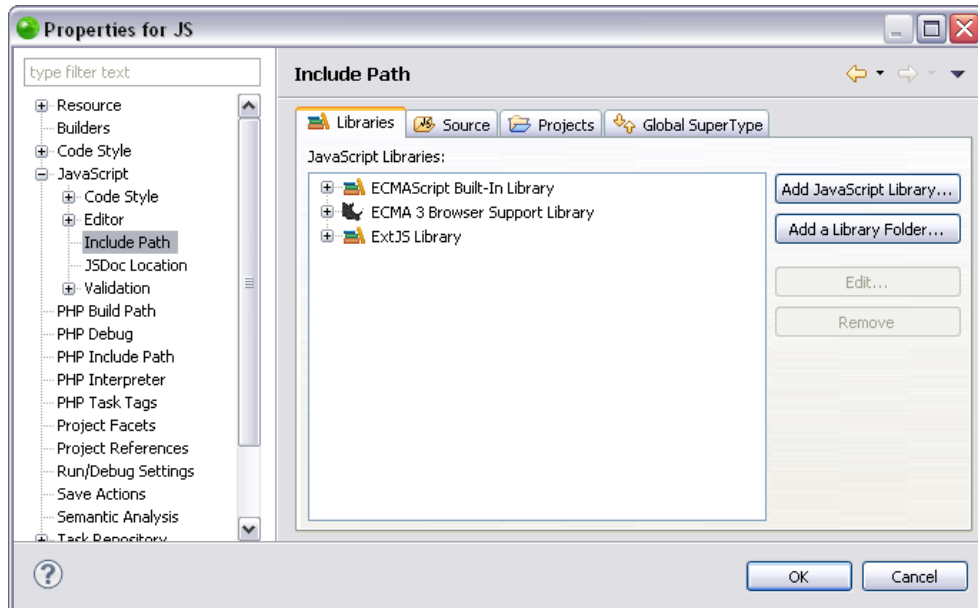
3. In the JavaScript Libraries Properties page click **Add a Library Folder**.
The "Library Folder Selection" dialog opens.

4. Select a folder in the "Library Folder Selection" dialog or create a new folder by clicking **Create New Folder**.

5. To apply changes click **OK**.

Functionalities such as Content Assist will now be able to access the JavaScript library folder. You can see the library folder in the "JavaScript Resources" node in your Project directory.
For information on Managing libraries see Managing JavaScript Libraries.

Exporting JavaScript User Libraries

This procedure describes how to export JavaScript user libraries, making them accessible to whoever has access to the repository where it is stored. Exporting a JavaScript user library will only export a description of the library in .xml format, and will not include any of the library's content.

Important Note:

Before configuring any settings for JavaScript libraries make sure JavaScript support is enabled in your project. Select **Configure** from the right click menu of you project in your project directory and select **Convert to JavaScript Project**. If that option does not appear then JavaScript support has already been enabled.

To export a user library:

1. Go to **Window | Preferences | JavaScript | Include Path | User Libraries**.

   The User Libraries Preferences page opens.

2. In the JavaScript Libraries Preferences page select the user library you would like to export and click **Export...**

   The "Export User Libraries" dialog will open.

3. Select the library you would like to export by clicking the check box beside it. You may also use the **Select All** or the **Deselect All** buttons.

4. To choose where you would like to export your library to, fill in the "File location:" text field with the URL, or click **Browse...** and select the location.

5. To apply changes click **OK**.

Your user library has now been exported to the location you specified.

You may now import the exported user libraries from any location that has access to the location in which it is stored. For more information see .

Importing JavaScript User Libraries

This procedure describes how to import existing JavaScript user libraries that are on the disk or in a repository. This allows you to take an already built library and use it in your project, as well as share a library with other users of the same repository. Importing a library will only import a description of the library in .xml format, and will not include any of the library's content. Before importing a JavaScript user library, you must first have access to an exported JavaScript user library. For more information see [Exporting JavaScript User Libraries](#).

Important Note:

Before configuring any settings for JavaScript libraries make sure JavaScript support is enabled in your project. Select **Configure** from the right click menu of you project in your project directory and select **Convert to JavaScript Project**. If that option does not appear then JavaScript support has already been enabled.

To import a JavaScript user library:

1. Go to **Window | Preferences | JavaScript | Include Path | User Libraries.**

   The User Libraries Preferences page opens.

2. In the JavaScript Libraries Preferences page click **Import...**

   The "Import User Libraries" dialog will open.

3. To choose where you would like to import your library from, fill in the "File location:" text field with the URL or click **Browse...** and select the location.

4. Select the libraries you would like to import from the options in the "Libraries contained in the selected file:" box, or press **Select All** or **Deselect All**.

5. To apply changes click **OK**.

   Your library's description in .xml format has now been imported into Zend Studio.

If the library of the user who imports it is stored in the same location on the disk as the user who exported it, Zend Studio will automatically find the libraries content and store it accordingly. If the JavaScript user library is stored in a different place for the two users, the **Edit...** button allows you to replace the location URL. See Editing JavaScript Libraries for more information.

Editing JavaScript Libraries

This procedure describes how to edit JavaScript libraries, library folders and their components. The edit functionality is used to customize each library according to it's specifics. For example, if the library is a list of JavaScript files then clicking **Edit** will open a dialog allowing you to select which files are provided by this library. If library is a zip file, then clicking **Edit** will open a dialog allowing you to select the proper zip file.

In order to edit a library, there must be libraries available. For more information see Adding a JavaScript Library or Adding a Library Folder to JavaScript Libraries .

The Edit functionality is only for user libraries.

Editing a JavaScript User Library

This procedure describes how to edit JavaScript user libraries for user libraries only. You will need to edit a JavaScript user library when you would like to change it's name. Changing a library's name allows you to create descriptive differentiations between your libraries. In order to edit a library, there must be libraries available. For more information see Importing JavaScript User Libraries or Adding a JavaScript Library.

Important Note:
Before configuring any settings for JavaScript libraries make sure JavaScript support is enabled in your project. Select **Configure** from the right click menu of you project in your project directory and select **Convert to JavaScript Project**. If that option does not appear then JavaScript support has already been enabled.

To edit a JavaScript user library:

1. Go to **Project | Properties | JavaScript | Include Path - or -** Select **Properties | JavaScript | Include Path** from the right click menu of the project folder in your project directory.

2. In the JavaScript Libraries page select the JavaScript user library you would like to edit
    and click **Edit...**

    The "Edit JavaScript Library" dialog will open.
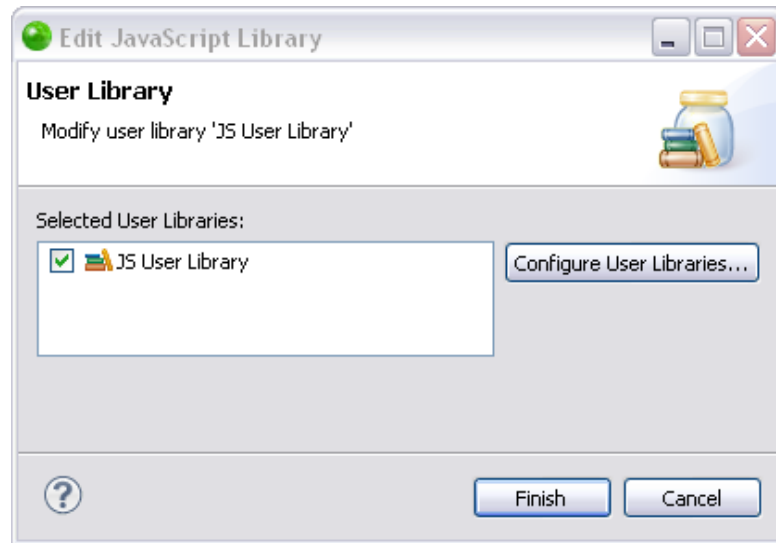
3. Select the JavaScript user library to edit by selecting the check box beside it and click
    Configure User Libraries.

    The User Libraries Preferences page opens.
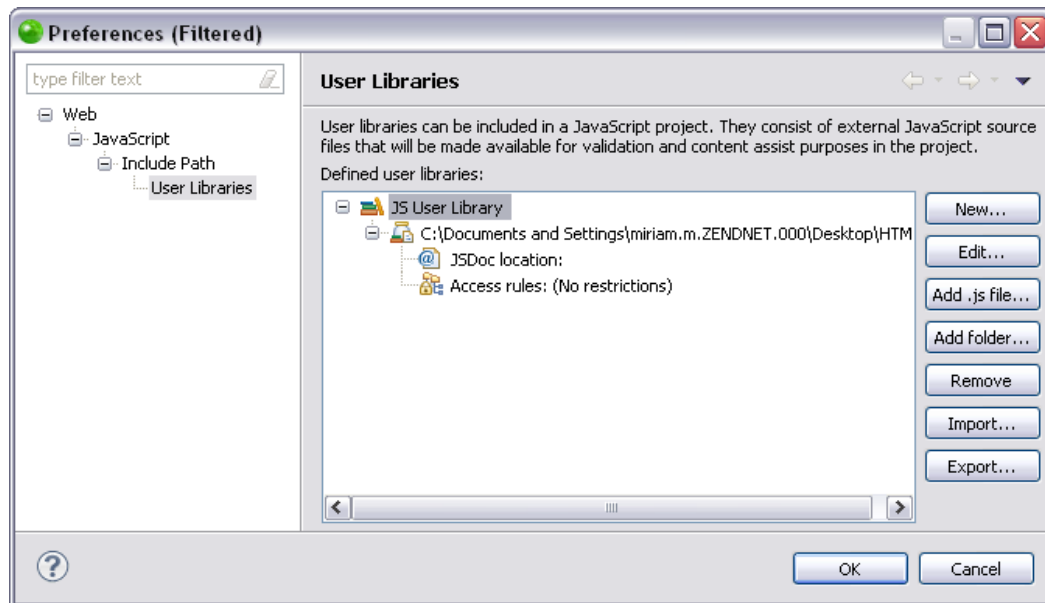
4. This page allows you to do the following:

- Highlight the JavaScript user library name and click **Edit** to edit the name of the user library.
- Highlight the path of the added folder and click **Edit** to edit which folder is included in the JavaScript user library.
- Highlight the Access rules and press **Edit** to edit the current access rules. For more information see Editing Access Rules.

Note:

If you have added a library folder, the only component that can be edited is the access rules.

5. To apply changes click **OK**.

For information on Managing libraries see Managing JavaScript Libraries.

You have now edited your user library. To learn more about using JavaScript libraries see Developing with JavaScript.

Note:

The Edit functionality is only for user libraries.

Editing a JavaScript Library

This procedure describes how to edit a JavaScript library. This allows you to change the classpath container path of a JavaScript library. JavaScript libraries are only relevant to PHP projects with JavaScript support enabled.
This procedure is not for JavaScript User Libraries. See Editing a JavaScript User Library for more information.

Important Note:

Before configuring any settings for JavaScript libraries make sure JavaScript support is enabled in your project. Select **Configure** from the right click menu of you project in your project directory and select **Convert to JavaScript Project**. If that option does not appear then JavaScript support has already been enabled.

To edit a JavaScript Library:

1. Go to **Project | Properties | JavaScript | Include Path - or -** Select **Properties | JavaScript | Include Path** from the right click menu of the project folder in your project directory.

2. In the JavaScript Libraries Properties page select a library and click **Edit** .

The "Edit JavaScript Library" Dialog will open.
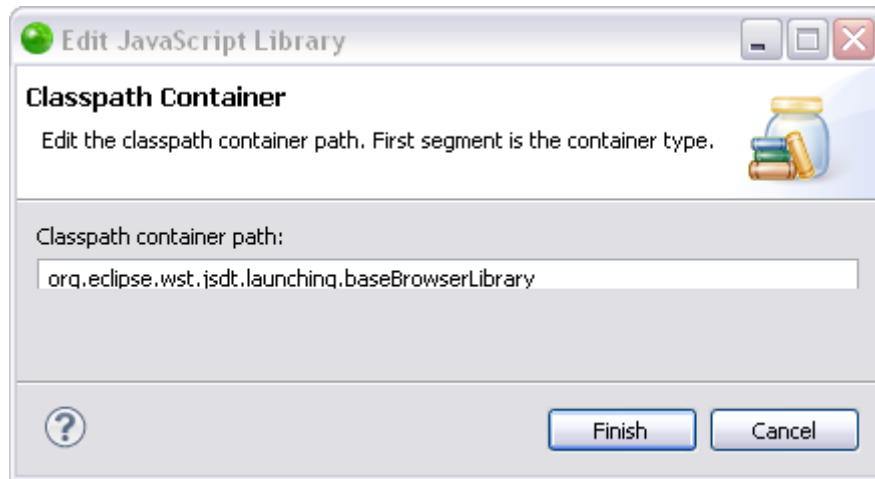
Note:

The **Edit** button is disabled for the jQuery and Prototype JavaScript libraries. This is because they are pre-configured and therefore do not offer the option to customize.

3. Enter the classpath container path in the "Classpath container path" text field. For more information see JavaScript Libraries.
4. To apply changes click **Finish**.

You have now defined what files will be loaded before analyzing JavaScript code in your project.

For information on Managing libraries see Managing JavaScript Libraries.

Editing Access Rules for JavaScript Libraries and Library Folders

This procedure describes how to manage access rules in your JavaScript libraries. Access Rules allow you to customize which libraries will be accessible when accessing a type. For more information see Access Rules in the Java Development User Guide .JavaScript libraries are only relevant to PHP projects. For more information see Creating PHP Projects.

Important Note:

Before configuring any settings for JavaScript libraries make sure JavaScript support is enabled in your project. Select **Configure** from the right click menu of you project in your project directory and select **Convert to JavaScript Project**. If that option does not appear then JavaScript support has already been enabled.

To edit an access rule in a JavaScript library or library folder:

1. Go to **Project | Properties | JavaScript | Include Path - or -** Select **Properties | JavaScript | Include Path** from the right click menu of the project folder in your project directory.

2. In the JavaScript Libraries Properties page expand a library and select an access rule.

3. Click **Edit**.

   The "Type Access Rules" dialog opens.
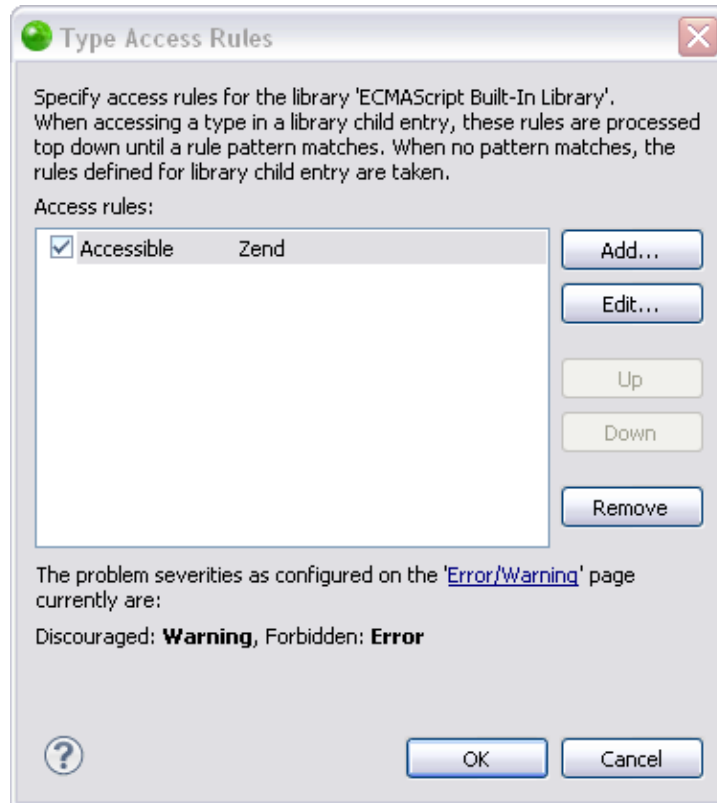
4. The options you have in this dialog are:

   ▪ **Add** - Add a new access rule. In the "Add Access Rule" dialog you can select a resolution (Forbidden, Discouraged, Accessible) and rule pattern, including wildcards, for the rule.

   ▪ **Edit** - Edit an existing access rule. In the "Edit Access Rule" dialog you can edit the resolution and rule pattern of the rule.

   ▪ **Remove** - Remove the Access Rule.

5. Use the **Up** and **Down** buttons to move between existing Access Rules.

6. To apply changes click **OK**.

If you choose "Discouraged" or "Forbidden"you can configure the problem severities in the "Error/Warning" page. For more information see Java Compiler Errors/Warnings Preferences.

For information on Managing libraries see Managing JavaScript Libraries.

Removing JavaScript Libraries

This procedure describes how to remove a JavaScript library or library folder from Zend Studio. Removing a JavaScript library or library folder means that its contents will no longer be available in Zend Studio, including in it's functionality such as Content Assist, and in any project it is associated with. JavaScript libraries are only relevant to PHP projects with JavaScript support enabled. For more information see Creating PHP Projects.

<table>
<tr><td>Important Note:</td></tr>
<tr><td>Before configuring any settings for JavaScript libraries make sure JavaScript support is enabled in your project. Select Configure from the right click menu of you project in your project directory and select Convert to JavaScript Project. If that option does not appear then JavaScript support has already been enabled.</td></tr>
</table>

To remove a JavaScript library/library folder:

1. Go to **Project | Properties | JavaScript | Include Path - or -** Select **Properties | JavaScript | Include Path** from the right click menu of the project folder in your project directory.

   Your project's properties page opens.

2. In the JavaScript Libraries Properties page select a library/library folder and click **Remove**.

    Your JavaScript library/library folder has been removed.

3. To apply changes click **OK**.

If you would like to add a JavaScript library see [Adding a JavaScript Library](). To add a library folder see [Adding a Library Folder to JavaScript Libraries]().

Working with Ajax Tools

Ajax Tools is a set of features based on the Web browser incorporated into Zend Studio. This provides the advantage of having a fully functioning web browser in your environment as well as the ability to edit, debug, and monitor your projects live, thus improving and simplifying the process for you.
The additional functionality provided by Ajax Tools can be applied to HTML, CSS, JavaScript, and XML.
To use Ajax Tools functionalities go to the Web Browser Tools Perspective which can be manually accessed by going to **Window | Open Perspective | Other | Web Browser Tools Perspective**.

Ajax Tools provides the following Views:

- DOM Inspector View - The *DOM* Inspector view provides a pre-defined hierarchal tree of HEAD and BODY elements. The attributes and values of the selected node appear in the view as well.

- Browser Console View - The Browser Console view is an aggregative list of the execution errors, warnings, and information messages that occurred in the time the page open in the Internal Web Browser was running.

- Request Monitor View - The Request Monitor view allows you to analyze the requests that occur in the browser open in the Internal Web Browser. The request is separated into different components (request, waiting, and response), and allows you to see the exact time each component is active, in seconds. This view should be used when profiling your application.

- DOM Source View - The DOM Source view shows the content and structure, including the attributes and values, of the highlighted node (and its child nodes) in HTML format.

- CSS View - CSS style rules determine the formatting of an element. The CSS view provides four different tabs, each with a different approach to the CSS style rules in the browser, both active and inactive.

- JavaScript View - The JavaScript view allows you to evaluate JavaScript expressions. This is useful to test, check, and debug your JavaScript code.

- DOM Watcher View - The DOM Watcher view is a way to record events occurring in the node selected in the DOM Inspector view. This allows you to see what exact events are occurring live.

- DOM Compare View - The DOM Compare view compares DOM attributes, child nodes, and CSS properties of a node. This view will not appear automatically when selecting the Web Browser Tools Perspective. To open it go to **Window | View | DOM Compare**.

Debugging JavaScript

The Zend Studio JavaScript debugging function allows you to test your files and applications and detect errors in your code. These procedures describe how to debug JavaScript code live from your workspace using an internal JavaScript Debugger.

Debugging JavaScript in an HTML File

This procedure describes how to debug JavaScript code that is in an HTML file. To open an HTML file go to **File | New | HTML Page**.

To debug JavaScript in an HTML file:

1. To open the PHP perspective go to **Window | Open Perspective | PHP**.
   The PHP perspective opens.
2. In your project directory select **Debug As | Debug JavaScript** from the Right Click Menu.
   The "Confirm Perspective Switch" dialog opens to inform you that the launch will open the Debug Perspective.
3. Click **Yes**.
   The Debug Perspective opens.
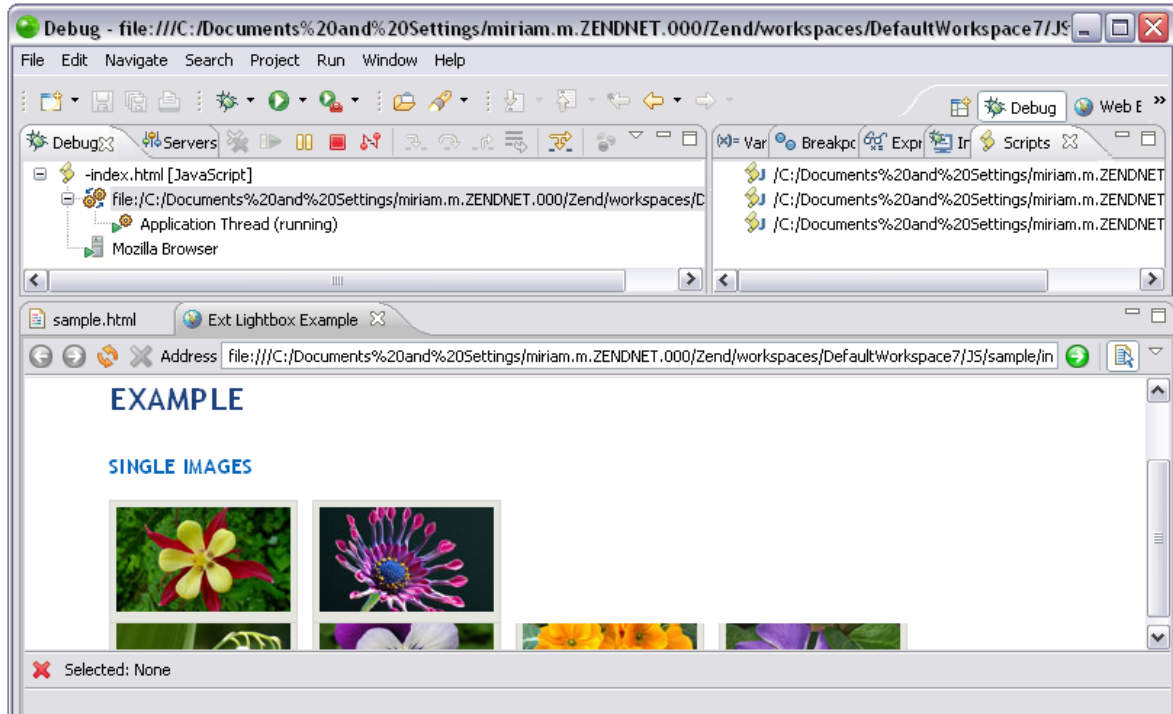
4. From the Scripts view double click a script from the list of available scripts.
   The script opens in the editor.

5. In the editor, set breakpoints at the relevant places in the file by double-clicking the vertical marker bar to the left of the editor.

6. Save the file.

7. Click in the Debug View to start debugging.

Use the controls in the Debug view to manage your debugging session.

A number of views will open with relevant debug information.

See the Running and Analyzing Debugger results topic for more information on the outcome of a debugging process.

Note:

If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.

Debugging a URL

This procedure describes how to debug a URL. This allows you to debug code according to a URL instead of a file name.

To Debug a URL:

1. From the menu bar select **Run | Run Configurations** - or - From the Right Click Menu in your project directory select **Run As | Run Configurations**.

The "Run Configurations" dialog opens.
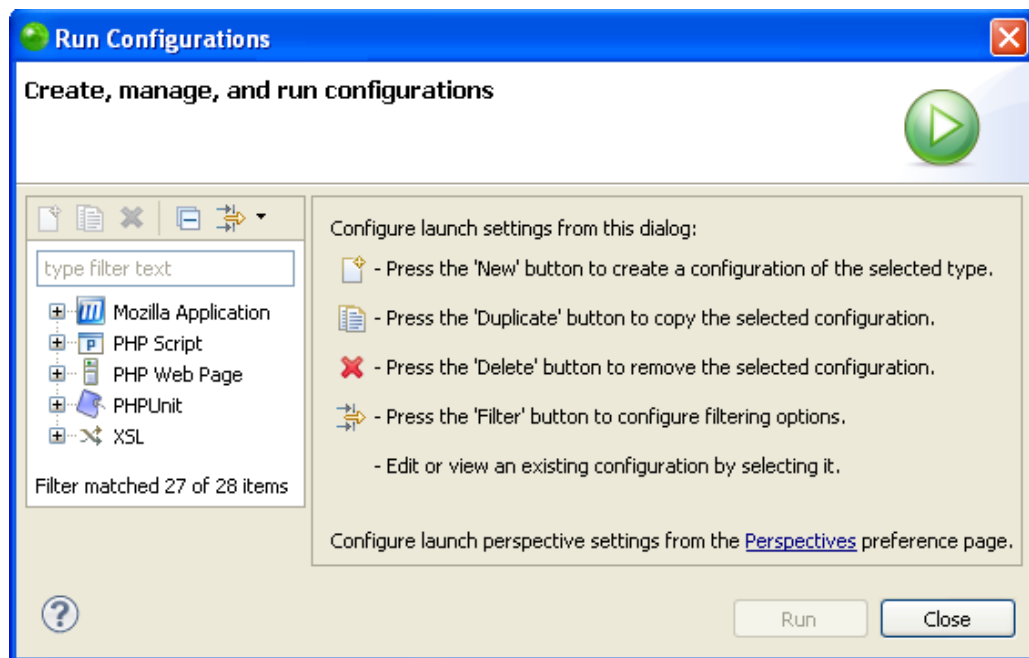
2. Double click the Web Application node to create a new debug configuration.

3. Open the Web Configuration tab.

   In the "Debug Target" field select "URL".

4. In the empty text field enter the complete URL.

5. Click **Apply** to apply the changes.

6. Click **Debug** to debug the URL.

   The URL opens in the [Internal Web Browser](#) and the debugging process begins.

A number of views will open with relevant debug information.

See the [Running and Analyzing Debugger results](#) topic for more information on the outcome of a debugging process.

Working with the Internal Web Browser

The Internal Web Browser allows you to open a web browser inside of your environment. This provides the advantage of having a fully functioning web browser in your environment as well as added functionality which allows you to carry out live editing, debugging, and monitoring in your projects. This improves and simplifies the process for you. The additional functionality provided by this feature can be applied to HTML, CSS, JavaScript, and XML.

The Internal Web Browser is only available in the Web Browser Tools perspective which can be opened manually by going to **Window | Open Perspective | Web Browser Tools**.
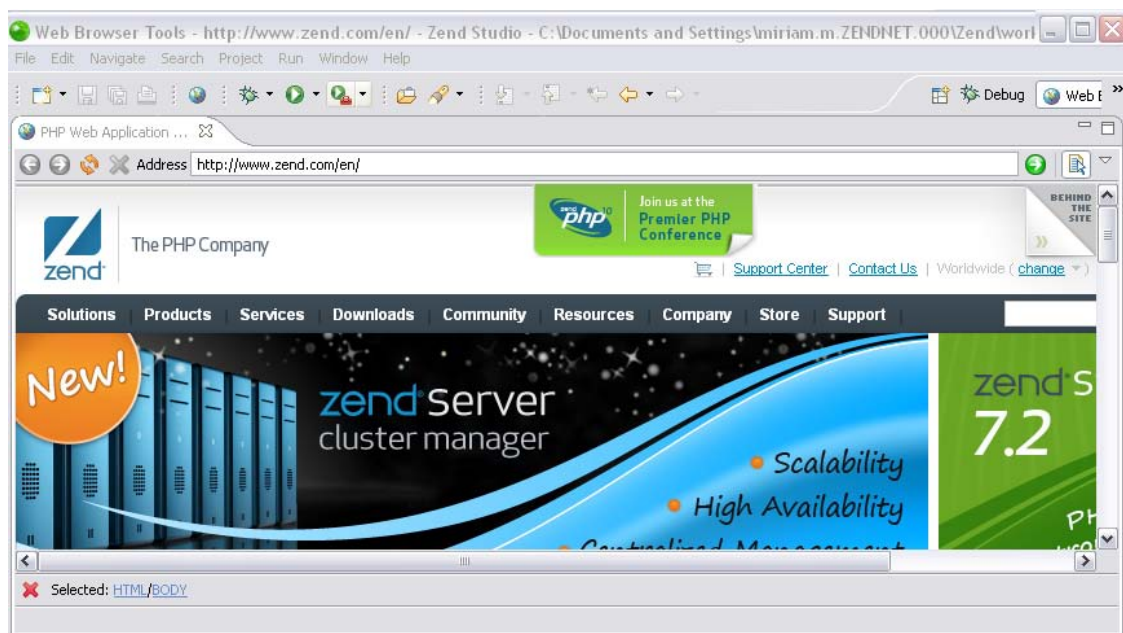
Opening a URL the Internal Web Browser

This procedure describes how to open a URL in the Internal Web Browser. This is the first step to using the Internal Web Browser.

To open the Internal Web Browser:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click from the main toolbar.

   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and click **OK**.

   The URL opens in the Internal Web Browser.

You can use the open browser as a fully functioning web page, or incorporate the functionalities of Ajax Tools.

Enabling the Ctrl+Click Element Functionality

This procedure describes how to enable the Ctrl+Click functionality in the Internal Web Browser. This functionality allows you to select a node in the DOM Inspector view by clicking on it inside of the browser.

To enable the Ctrl+Click functionality:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click ⊕ from the main toolbar.

   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and click **OK**.

   The URL opens in the Internal Web Browser.

4. Use 🔲 to enable the CTRL + Click element selection in the browser.

   Disabled icons will be greyed-out.

5. Hold down **Ctrl** and click on the element in the browser.

   The element is highlighted in the *DOM* Inspector view hierarchal tree.

You can also highlight an element in your browser using the DOM Inspector view hierarchal tree. For more information see Highlighting a Node in the Internal Web Browser.

Internal Web Browser Icons

The Internal Web Browser includes the following icons:

🔲	Enable/disable CTRL + Click element selection in the browser.
▽	View menu. Expands to allow you to "Clear Cache" and select/deselect to "Show Selection Path Bar".

Working with the Ajax DOM Inspector View

The DOM Inspector view provides a pre-defined hierarchal tree of HEAD and BODY elements. The information is according to the URL displayed in the <u>Internal Web Browser</u>. The nodes can be expanded to view their child nodes thus allowing you to see as many or as few of the nodes at any given time. The attributes and values of the selected node appear in the view as well.

T**he DOM Inspector view allows you to do the following:**

- <u>Highlight the node in the Internal Web Browser</u>
- <u>Add, remove, or edit DOM attributes and values</u>
- <u>Evaluate a node</u>
- <u>Compare a node</u>
- <u>Collapse the DOM tree to the body element using the icon</u>
- <u>Search the hierarchal tree using the icon</u>
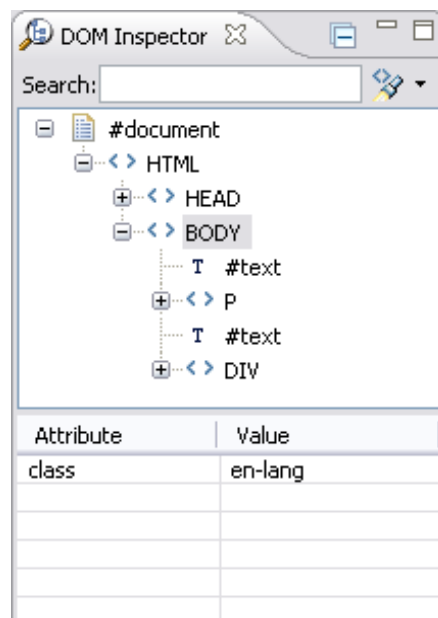
Highlighting a Node in the Internal Web Browser

This procedure describes how to highlight a node in the Internal Web Browser using the DOM Inspector hierarchal tree. This allows you to see how a node appears in the browser.

To highlight a node in the Internal Web Browser:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click ⊕ from the main toolbar.

The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

The URL opens in the <u>Internal Web Browser</u>.

4. Open the DOM Inspector view.

A hierarchal tree of the nodes appears with the attributes and values table below.
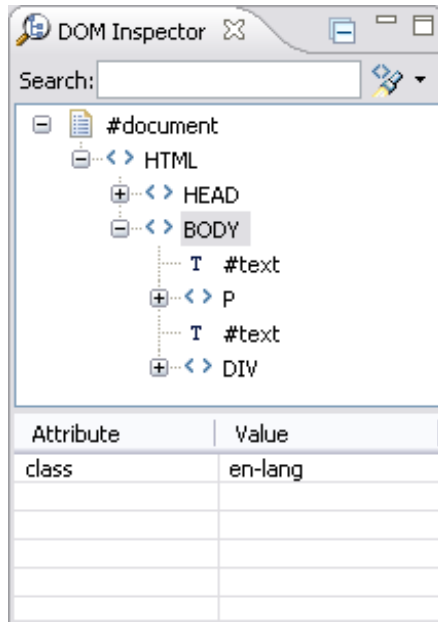
5. Select a node.

If the node is visually expressed, it will be highlighted in the Internal Web Browser.

If it is not a visually expressed DOM element you can investigate it further by opening the <u>DOM Source view</u> to see the element in HTML format.

You can also highlight a DOM element in the DOM Inspector hierarchal tree by selecting it in the browser. For more information see <u>Enabling the Ctrl + Click functionality</u>.

Managing DOM Attributes and Values

This procedure describes how to manage DOM attributes and values in the DOM Inspector view. This allows you to add, edit, and remove the attributes and values of a node.

To manage DOM attributes and values:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click from the main toolbar.

    The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

    The URL opens in the <u>Internal Web Browser</u>.

4. Open the DOM Inspector view.

    A hierarchal tree of the nodes appears with the attributes and values table below.
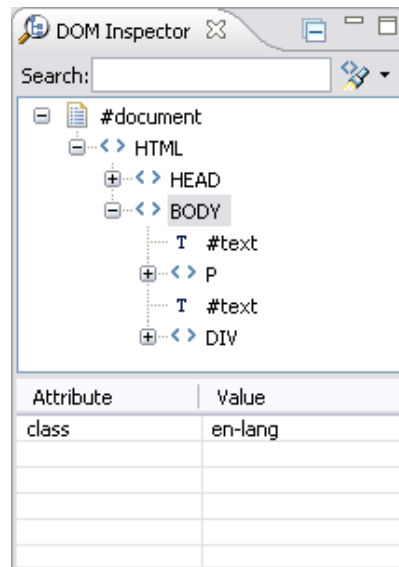
5. Select a node. If the node has attributes and values they will appear in the table.

Note:

Any node is capable of having attributes. As it is not required, not all DOM elements do.

6. Select the value or attribute you would like to manage.

    If you would like to add an attribute or value, it is not required to select one in the table.

7. Click on the attribute column and from the "right click menu" select **Add**, **Remove**, or **Edit**.

▪   If you have chosen **Add**, the "Add DOM Attribute" dialog opens. Enter the relevant information in the dialog and click **OK**.

▪   If you have chosen **Remove**, the component is removed.

▪   If you have chosen **Edit**, the "Edit DOM Attribute" dialog opens. Enter the relevant information in the dialog and click **OK**.

The attribute or value has been managed.

Evaluating a Node

This procedure describes how to evaluate a node in the DOM Inspector view using the JavaScript view in the Web Browser Tools perspective.

To evaluate a node:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click ![globe icon] from the main toolbar.

   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

   The URL opens in the Internal Web Browser.

4. Open the DOM Inspector view.

   A hierarchal tree of the nodes appears with the attributes and values table below.
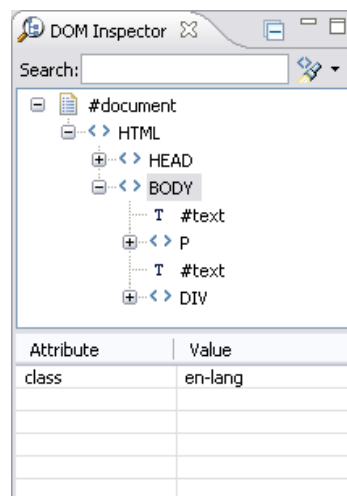
5. Select a node and right click.

   The "right click menu" opens.

6. Select "Evaluate Node".

   The JavaScript view opens with the node evaluation output.
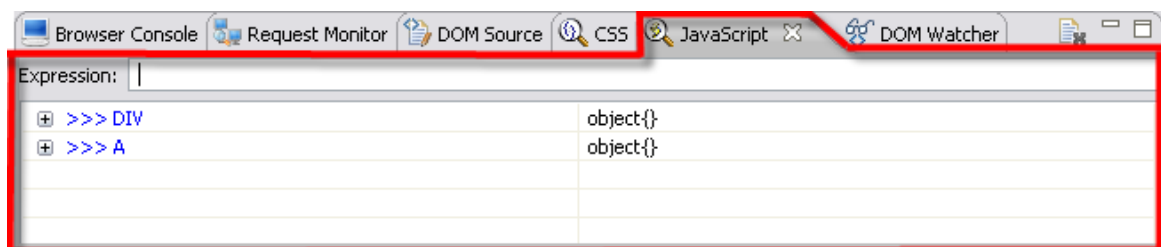
For more information on evaluating nodes see the JavaScript view.

Comparing a Node

This procedure describes how to compare nodes in the DOM Inspector view using the DOM Compare view in the Web Browser Tools perspective.

To Compare a node:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click from the main toolbar.

    The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

    The URL opens in the Internal Web Browser.

4. Open the DOM Inspector view.

    A hierarchal tree of the nodes appears with the attributes and values table below.
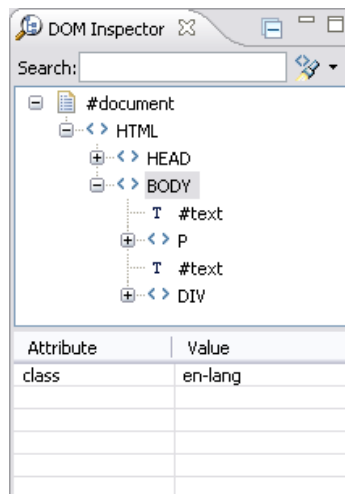
5. Select a node and from the Right Click Menu select "Compare Node".

    The DOM Compare view opens with the node already inserted in the "Compare List:" field. (To add another node to compare, simply repeat step 5 with the additional node or nodes.)

6. Highlight the nodes to compare in the "Compare List:" field.

7. In the "Compare:" field choose from the following:

-   **DOM Attributes** - To compare the DOM attributes of the selected nodes.

-   **Child Nodes** - To compare the child nodes of the selected nodes.

-   **CSS** - To compare the CSS properties of the selected nodes.

8.   Press **Compare.**

The details appear in the Dom Compare view organized by columns. Each column is for a different node.
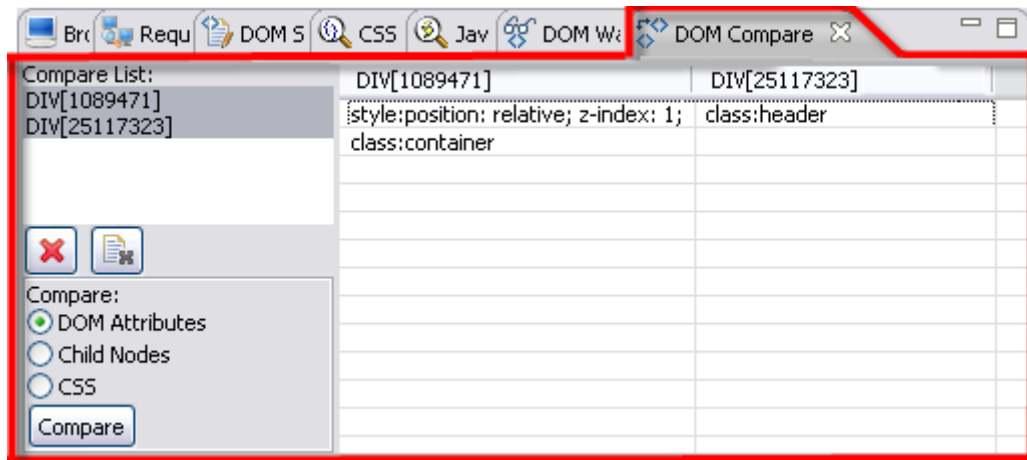
To clear nodes from the "Compare List:" field, click ![x] to remove only one node or ![list-x] to remove the entire list.

For more information on comparing nodes see the DOM Compare view.

DOM Inspector View Icons

The DOM Inspector View includes the following icons:

![collapse]	Collapse the DOM tree Body element.
![filter-name]	Filter DOM elements by name. Use the drop-down menu to change the filtering criteria.
![filter-id]	Filter DOM elements by ID. Use the drop-down menu to change the filtering criteria.
![filter-class]	Filter DOM elements by class. Use the drop-down menu to change the filtering criteria.

Working with the Ajax Browser Console View

The Browser Console view is an aggregative list of the execution errors, warnings, and information messages that occurred in the time the open page in the Internal Web Browser was running.

The Browser Console view shows the following information:

- Message - The type of message and its content.
- File - The file name of the error.
- Line Number - The location where the error occurred.

Opening a DOM element in an HTML Editor

This procedure describes how to open a *DOM* element in an HTML editor. This allows you to see the source code of an element.

To open the element in an HTML editor:

1. Go to **Window | Open Perspective | Web Browser Tools.**

2. Click [icon] from the main toolbar.

   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

   The URL opens in the Internal Web Browser.

4. Open the Browser Console view.

   A list of the execution errors, warnings, and information messages appears.

5. Select the message you would like to open in an HTML editor and double click.

   The message opens in an HTML editor.

The HTML editor is not for editing or updating. It is only intended as a way to see the message in HTML format to allow an alternative method of viewing it. For information on editing a node see Editing the Source Code of a Node.

Browser Console View Icons

The Browser Console View includes the following icons:

	Show only errors in the view.
	Show only warnings in the view.
	Show only information messages in the view.
	Clears all messages in the view.
	Expands the View drop-down Menu allowing you to show/hide CSS messages, JavaScript messages, and XML messages.

Working with the Ajax Request Monitor View

The Request Monitor view allows you to analyze the requests that occur on the open browser in the Internal Web Browser. The request is separated into different components (request, waiting, and response), and allows you to see the exact time each component is active, in seconds. This feature is especially useful in investigating what elements in a page are taking a long time to load, and how long. The view shows the URL and its correlated request analysis, as well as any messages associated with the request. You can also see the URL and component of the request by hovering over the analysis.

Showing the Response/Request Monitor View

This procedure describes how to show the request/response Content panel. This panel provides additional information to the Request Monitor view. For more information see Response/Request Panel.

To show the request/response content panel:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Open the Request Monitor view.

3. Click 🌐 from the main toolbar.

   The "Open URL…" dialog opens.

4. Enter a URL in the "Enter URL…" text field and press **Enter**.

   The URL opens in the Internal Web Browser and the response/request details open in the Request Monitor view.

5. Click [icon].
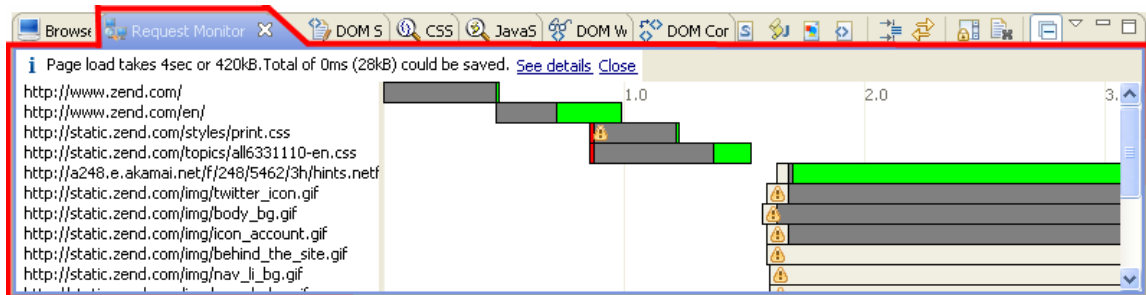
The request/response panel opens.
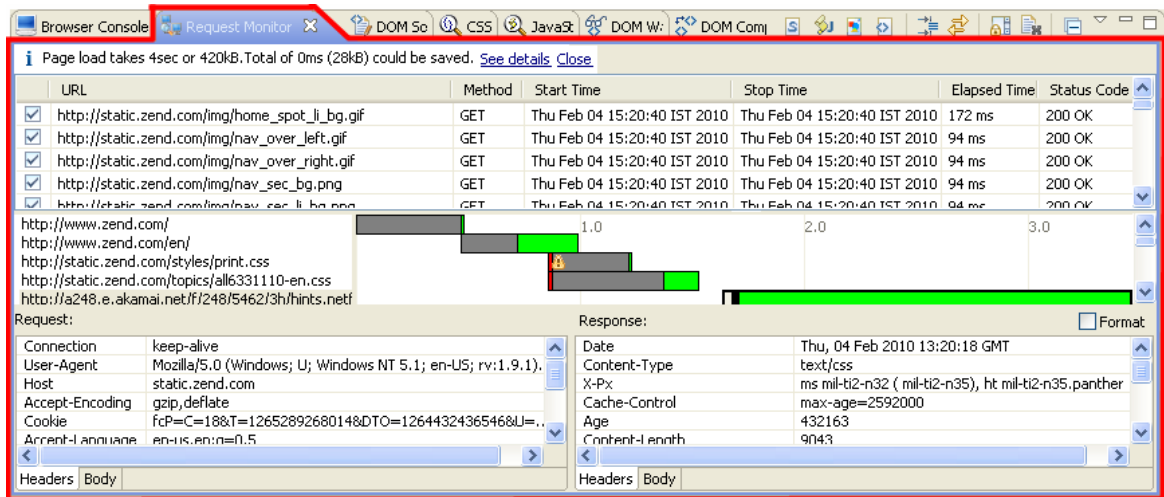
To close the request/response panel click [icon] again.

The Response/Request Content Panel in Request Monitor View

The request/response content panel adds the following information to the view:

- The URL
- The Method
- The Start Time
- The Stop Time
- The Elapsed Time
- The Status Code
- The Request information:
  - Headers - A set of parameters.
  - Body - The content of the page.
- The Response information

The request/response panel adds a window to the top and to the bottom of the view. The top window is the same information as the original view but in columns.

The columns in the request/response content panel are:

- URL - A list of all the request/response URL's.
- Method - The method used.
- Start Time - The start time of the monitoring including the date.
- Stop Time - The stop time of the monitoring including the date.

- Elapsed Time - The elapsed time of the monitoring.
- Status Code - The status code of the monitoring.

The bottom window separates the request and response into two tables. If you select a URL in the middle window you can see the details of the request and response separately. The details are separate into Headers and Body.

Request Monitor View Rules

In the analysis of a request you may see the ⚠ icon. If you hover over the icon you will see the type of rule it is expressing. The rules are found in the [Ajax Errors/Warnings Preferences](#) page.
The types of rules are:

Rule	Description
Avoid Redirects	Detects a response with a 301 or 302 status. If the browser has to follow any redirects before entering the main page, it cannot do anything else simultaneously. The extreme case is a "redirect chain", where one URL redirects to another redirect. The example below illustrates a "redirect chain and the extra cost of it": google.com -> www.google.com -> www.google.pl (2 requests and 0,4sec, 2kB total transfer in/out).
Combine External CSS, Images, or JavaScript	Detects more than one download of a CSS, Image or JavaScript file. Many small resources of the same type may take longer to load than a single bigger resource. Browsers try to minimize the time required to load many resources by parallelizing downloads as much as possible. Parallel downloads do save time, however they don't save bandwidth. Assuming that a typical small resource is 1kb big and a typical GET request/response headers size is 1kb, a download of 10 small resources costs 10*(resourceSize+headersSize) = 10*(1+1) = 20kb of bandwidth. If all resources were replaced into a single header, it would instead cost 11kb. This saves 45% of bandwidth. Merging multiple images into a single sprite could result in even smaller images because of how images are represented internally. For example, if all images use similar colors, they would use a single shared palette instead of many separate palettes.
CSS Expression or Filter Use	Check if "expression(...)" or "filter: alpha(...)" is used. They slowdown rendering because an expression has to be evaluated at all times (on scroll, re-size, and load). The Alpha filter is just slow, according to

	YSlow.
Unefficient CSS Selector	Check if any used selector uses global qualifier ("*"). Universal selectors take more time to apply because they have to be applied to all "document" *DOM* elements. Based on https://developer.mozilla.org/en/Writing_Efficient_CSS
Unused CSS Rule	Walks over DOM "document" to find the CSS rules referenced in DOM nodes (via Mozilla API). Next parses all loaded CSS files to find all loaded rules. List rules that were loaded but never used. For a large web site it's easy to lose control of CSS and keep constantly adding styles, without removing them to not break something. This rule would help maintain minimal CSS rules.
Unused CSS File	Check if any of the rules defined in the CSS file are referred to in the HTML document.
Gzip Contents	Check if the response uses Content-Encoding: gzip header. gZip compression saves bandwidth.
Leverage Browser Caching	Check if the response contains an "Expires" or "Cache-Control" header. Caching significantly reduces the amount of necessary downloads.
Minimize Cookie Size	Check the length of a requests' "Cookie" header. The average request should be no bigger than 1500 bytes which allows it to fit into one packet. Too big of a cookie can easily break that number, causing the request take more packets. Google suggests to use cookies no longer than 1000 bytes and recommends up to 400 bytes. See http://code.google.com/intl/pl-PL/speed/page-speed/docs/request.html#MinimizeCookieSize
Minimize the Number of IFrames	Yahoo recommends reducing up to 5 IFrames per web page. There is no clear evidence on how more IFrames contribute to performance loss.
No 404s	Detects responses with a 404 status. When opening a web page it's not immediately visible if parts of it are missing due to a 404 response from the server.
Optimize CSS and JavaScript Order	Parses DOM "document" to find out if there are any LINK tags referring to CSS after SCRIPT tags referring to external JavaScript files. This is not a problem for modern browsers anymore because they are able to download both CSS and JavaScript resources at the same time. Still, when JavaScript is executed, any other actions is blocked because usually JavaScript execution occurs in the main

	browser thread. Below is the link to a Google Page-Speed diagram that shows a hypothetical situation: http://code.google.com/intl/pl/speed/page-speed/docs/rtt.html#PutStylesBeforeScripts They are in the following order: CSS file, JavaScript file, JavaScript file, CSS file. Requests are handled by a servlet that by default delays the response for about 1sec in order to simulate the network load.
Parallelize Downloads Across Domains	Checks if the requests are more or less equally split to all domains (using a user-defined threshold factor). Reports the problem for every domain that responds to significantly more requests than others. Using more domains helps browsers more effectively parallelize downloads because usually browsers have hard-set limits of maximum parallel downloads per hostname. HTTP 1.1 recommends up to 2 parallel connections. Popular browsers use up to 6.
Reduce DNS Lookups	Check if the number of unique hostnames is less or equal to 5. Using too many hostnames can cause times needed for resolving a hostname's IP addresses to be too long. Google recommends up to 5 domains.
Uncompacted Resource	Detect any extra whitespaces, comments, or otherwise redundant information, that could be removed to make the resource smaller and therefore faster to download. This rule analyzes JavaScript, CSS and HTML files.
Use GET for Ajax Calls	Detects the XML/HTTP requests that use a request method other than "GET". According to Yahoo, many browsers need 2 packets for POST requests, compared to 1 when using GET. See http://developer.yahoo.com/performance/rules.html.

Request Monitor View Icons

The Request Monitor View includes the following icons:

	Show only CSS analysis.
	Show only JavaScript analysis.
	Show only image analysis.
	Show only HTML/HXML/XML analysis.
	Show only XHR analysis.
	Show only HTTP analysis.
	Lock/Unlock the scroll of the content page.
	Clear the call list.
	Hide the request/response content panel.

Working with the Ajax DOM Source View

The *DOM* Source view shows the content and structure, including the attributes and values, of the highlighted node (and its child nodes) in HTML format.

Editing the Source Code of a Node

This procedure describes how to make changes in the source code of a node. This allows you to take the analysis capabilities of Ajax Tools and utilize the information to optimize element performance.

To make changes in the source code of a node:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click ![globe icon] from the main toolbar.

   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

   The URL opens in the Internal Web Browser.

4. Select a node from the DOM Inspector hierarchal tree.
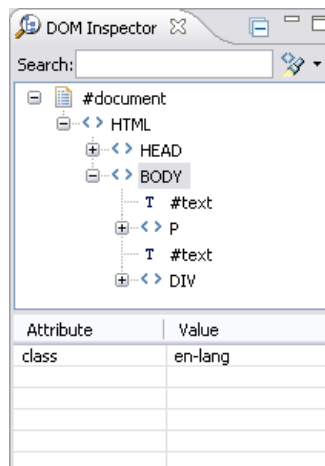
5. Open the DOM Source view.

The source code of the selected node appears in HTML format.
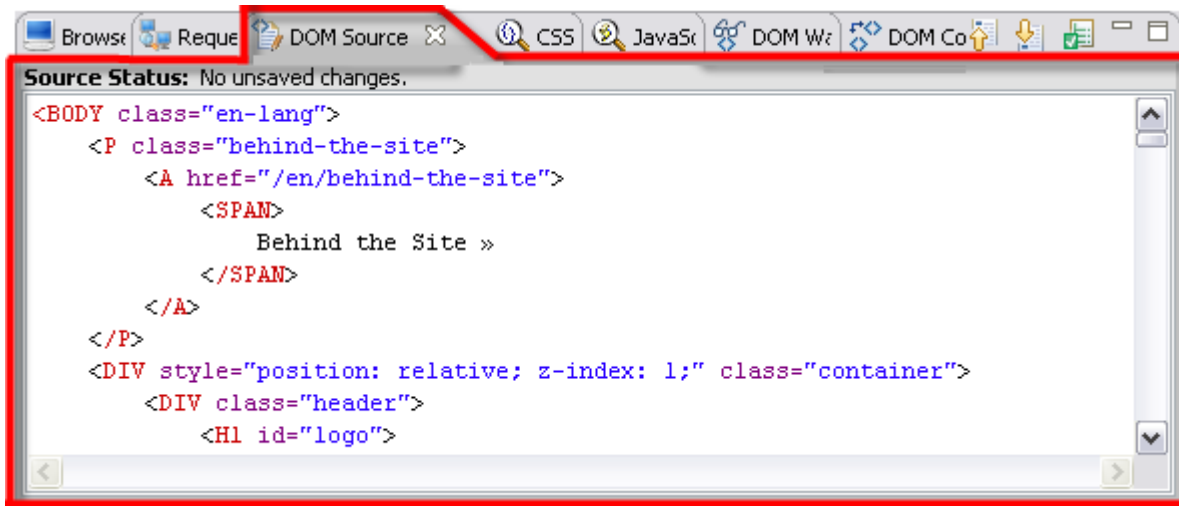
6. Make the necessary changes in the source code.

7. Click [icon].

The browser is modified according to your changes.

If you would like to revert back to the original browser source code before clicking [icon], click [icon]. You will lose any changes made to the source code that were not updated completely according to the instructions above.

Validating a DOM Source

This procedure describes how to validate a DOM source. The DOM source is validated by running an XML validator. This will detect problems such as missing closing tags and unpaired quotes in tag attributes.

To validate DOM source:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click [icon] from the main toolbar.

The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

The URL opens in the Internal Web Browser.

4. Open the DOM Source view.

The source code of the selected node appears in HTML format.

5. Click [icon].
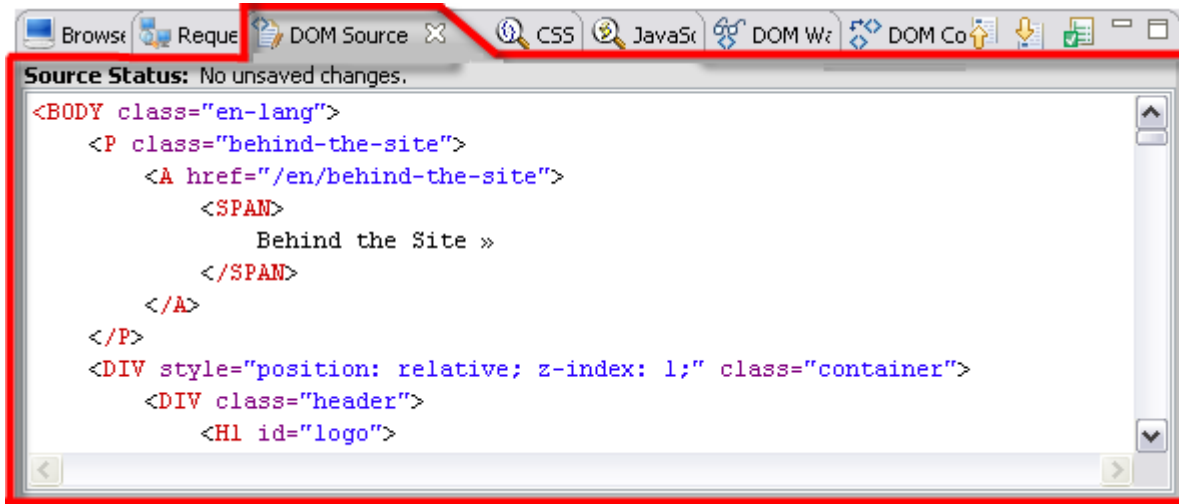
The source status appears at the top of the view.

If there is a validation error, the reason for the error will appear beside it in the "Source Status" field.

If you make a change that leads to a validation error in the DOM Source view, you will not be able to update the browser source with the changes.

DOM Source View Icons

The DOM Source View includes the following icons:

[icon]	Update the browser source with changes.
[icon]	Refresh the source code according to the browser page.
[icon]	Validate the DOM source.

Working with the Ajax CSS View

CSS style rules determine the formatting of an element. The CSS view provides four different tabs, each with a different approach to the CSS style rules in the browser, both active and inactive.

Style Rules Tab

The Style Rules tab is a list of the style rules for the selected node. This tab shows only the CSS rules that were defined by the author of the web page, or predefined internally in the web browser. The rules appear in the order they were defined.

The following are the columns that make up the view:

- Rule - The name of the style rule. You can expand the properties of the rule.
- Property - The style rule property. Each property is marked with a red or green square. Green indicates it is an active rule, while red indicates it is an inactive rule.
- Value - The value of the property. Some values can be edited by clicking on them. See Editing a Value.
- URL - The URL in which the rule applies.
- Line Number - The location of the rule.
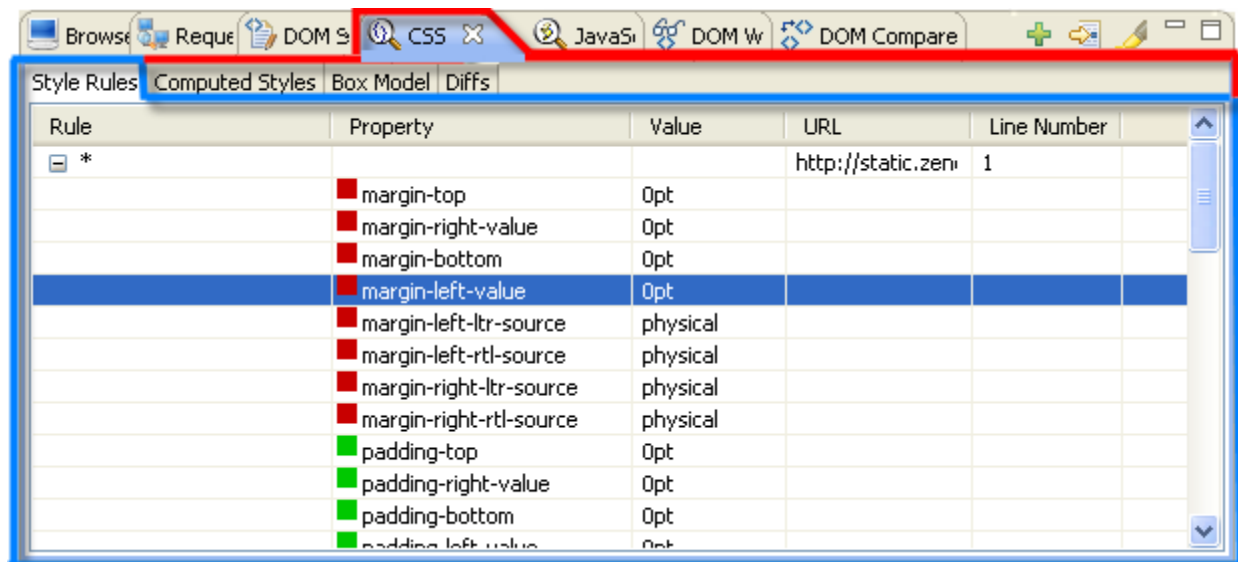
The following functionalities are available for the style rules tab:

- Edit an attribute's value
- Add a property
- Open CSS file
- Toggle highlighting using the icon in the view

Editing a Value

This procedure describes how to edit the value of an attribute. Attributes are properties of a node. Editing their values also effects their expression in the browser.

To edit a value:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click ⊕ from the main toolbar.
   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.
   The URL opens in the [Internal Web Browser](#).

4. Select a node from the *DOM* Inspector view.

5. Open the CSS view and within it, the Style Rules tab.
   The style rules and their properties appear.

6. Click on the value of the style rule in the "Value" column and enter the new value.
   The value of the style rule has been edited.

Adding a Property

This procedure describes how to add a property to a node.

To add a property:

1. Go to **Window | Open Perspective | Web Browser Tools.**

2. Click ⊕ from the main toolbar.
   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

   The URL opens in the <u>Internal Web Browser</u>.

4. Select a node from the DOM Inspector.

5. Open the CSS view and within it the Style Rules tab.

   The style rules and their properties appear.

6. Select a style rule from the "Rule" column and from the "Right Click Menu" select "Add

   Property"   - or - click ✚ .

   The "Add Property" dialog opens.

7. Enter the property name and value separated by a colon in the text field and click **OK**.

   The property appears in the "Property" column and the browser has been modified

   accordingly.

Opening a CSS File

This procedure describes how to open a CSS file. A CSS file will be a source file of the rule selected in
the "Style Rules" tab.

To open a CSS file of the style rule:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click 🌐 from the main toolbar.

   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

   The URL opens in the <u>Internal Web Browser</u>.

4. Select a node from the DOM Inspector view.

5. Open the CSS view and within it, the Style Rules tab.

   The style rules and their properties appear.

6. Select a style rule from the "Rule" column and from the "Right Click Menu" select "Open

   CSS file", or click 📝 .

   The style rule opens in a CSS file.

Computed Styles Tab

The Computed Styles tab is a list of the style rules for the selected node. This tab shows only those that
will be used to render the particular tag; This is the list of all styles supported by the browser with their
values calculated based on the definitions provided by page author.

The following are the columns that make up the view:

- Rule - The name of the style rule. You can expand the properties of the rule.
- Property - The style rule property. Each property is marked with a red or green square. Green indicates it is an active rule, while red indicates it is an inactive rule.
- Value - The value of the property.
- URL - The URL in which the rule applies.
- Line Number - The location of the rule.
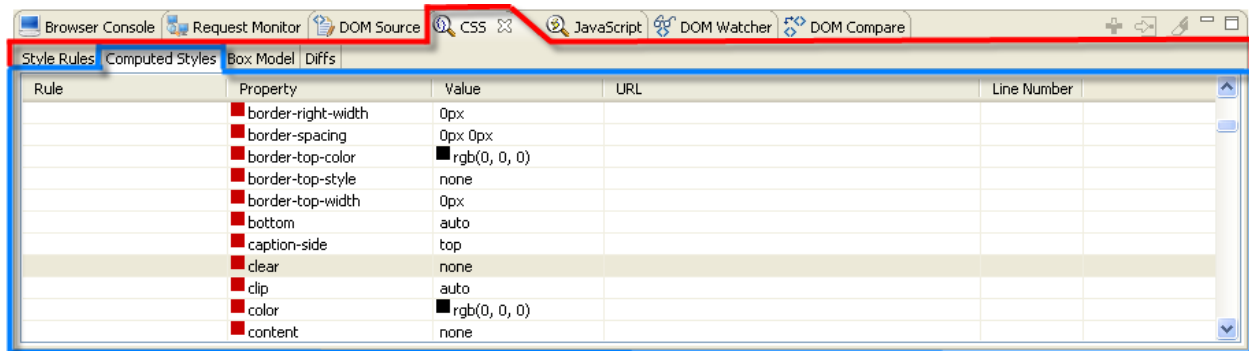
Box Model Tab

The box model in the Box Model tab provides the visual approach to the properties of the selected DOM element. This tab shows the width and height of the innermost dimension, the padding, the border, and the margin of the DOM element You can control which element you are analyzing using the Navigation Controls.

The following are the components that make up the view:

- Rule - The name of the style rule. You can expand the properties of the rule.
- Property - The rule property. Each property is marked with a red or green square beside it. Green indicates it is an active rule, while red indicates it is an inactive rule.
- Value - The value of the property. Some values can be edited by clicking on them. See Editing a value.
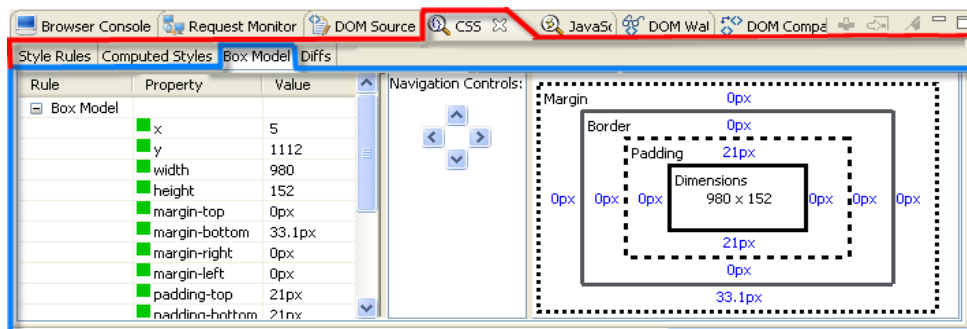
The Navigation Controls control which element you are analyzing by moving throughout the hierarchal tree in the DOM Inspector.

The controls are used as follows:

- Up - Navigates to the parent DOM element.
- Down - Navigates to the first child DOM element.
- Right - Navigates to the next sibling DOM element.
- Left - Navigates to the previous sibling DOM element.

The box model in the view is made up of the following components:

- Dimensions - The dimensions of the content of the element.
- Padding - The width of the box padding. If the width of the padding is 0, it will be the same as the content edge.
- Border - The width of the box border. If the width of the border is 0, it will be the same as the padding edge.
- Margin - The width of the margin border. If the width of the border is 0, it will be the same as the border edge.

For more information on the Box Model see http://www.w3.org/TR/CSS2/box.html.

Diffs Tab

The Diffs tab allows you to view the changes made so far in the Style Rules tab. Some of the "Vaues" in the Style Rules tab can be edited, but the history of their values is not stored there. The Diffs tab is a way to compare the original style rules with the current (modified) style rules. If you have modified style rules in the "Style Rules" tab, the differences will automatically show in the "Diffs" tab. For more information see Editing a Value.

There are two options for comparing the style rules:

- Show selection CSS diff - Compares the differences for the selected style rule only (selected in the Style Rules tab).
- Show global CSS diff - Compares the differences for all the modified style rules in the node.
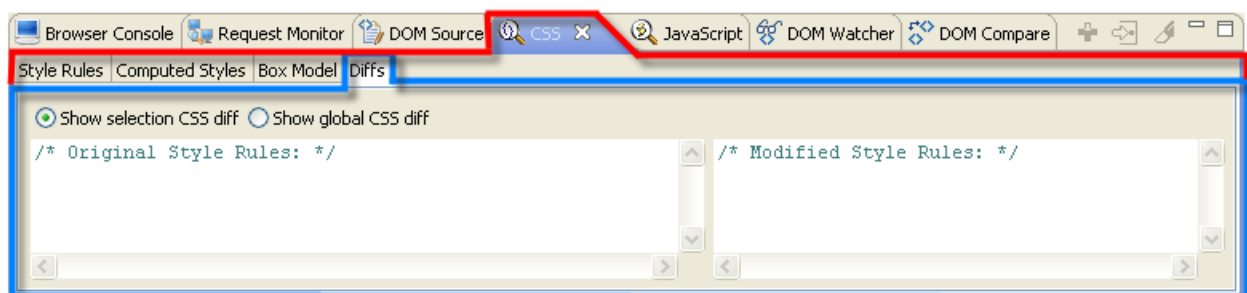
CSS View Icons

The CSS View includes the following icons:

✚	Add a property.
	Open a CSS file.
	Toggle highlighting.

Note:

Icons are enabled only for those CSS rules (entries in the "Style Rules" tab) that can be edited. You can edit only those rules that were added to the website using an external stylesheet (embedded <style> tag or in style="" parameter in any tag). Icons will not be enabled if:

- You have not selected anything in the "Style Rules" tab.

- You have selected a browser's internal rule. Internal rules can be recognized by a URL starting with "resource://..." instead of "http://..."

Working with the Ajax JavaScript View

The JavaScript view allows you to evaluate JavaScript expressions. This is useful to test, check, and debug your JavaScript code.

Evaluating JavaScript Expressions

This procedure describes how to evaluate JavaScript expressions.

To evaluate a JavaScript expression:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click ![icon] from the main toolbar.

   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

   The URL opens in the <u>Internal Web Browser</u>.

4. Go to the JavaScript view.

5. Enter your JavaScript expression in the "Expression:" text field and press **Enter**.

   Your JavaScript expression evaluation output appears.
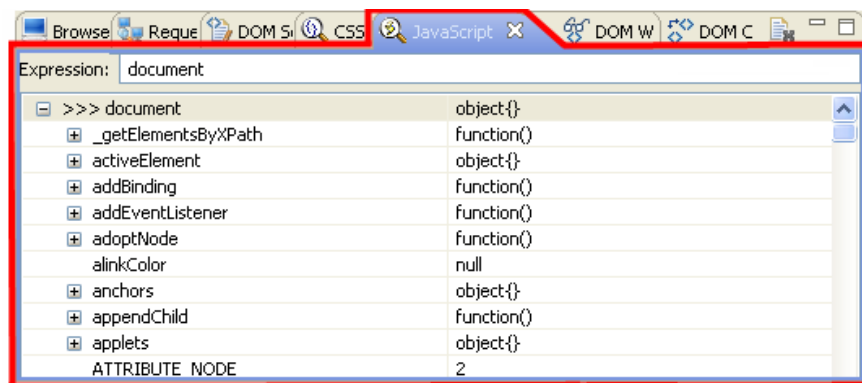
6. Expand a line to see the complete output of the evaluation.

The output does not clear automatically, even if you launch a new evaluation. To clear previous output click the ![icon].

You can also evaluate JavaScript expressions in the *DOM* Inspector view. For more information see <u>Evaluating a Node</u>.

JavaScript View Icons

The JavaScript View includes the following icons:

![icon]	Clear variables and their references.

Working with the Ajax DOM Watcher View

The *DOM* Watcher view is a way to record events occurring in the node selected in the DOM Inspector view. This allows you to see what exact events are occurring live.

The following components make up the view:

- Type - The type of action.
- TimeStamp - The exact time the event occurred.
- Details - The details of the event. These are separated into "screen" and "client".

Recording Events Live

This procedure describes how to record events occurring live in your browser. This allows you to see which events are occurring in a node as you are actively using the browser.

To record events:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click ![icon] from the main toolbar.
   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.
   The URL opens in the Internal Web Browser.

4. Go to the DOM Watcher view.

5. Select a node in the DOM Inspector view or use the Ctrl + Click option in the Internal Web Browser to select a node. (For more information see Highlighting a Node.)
   The node recording timestamp and details appear in the "Node:" text field.

6. Click ![icon] to begin recording.

7. Use the browser.

> **Note:**
> If you click on a link while recording, the DOM Watcher will automatically stop the recording and follow the link.

8. To stop recording click ![icon].
   The recording stops and the events and their properties remain.

The event list from a recording is not automatically deleted. If you Start recording for a node you have previously recorded, the new event list will begin below the previous event list. To delete an event list click ![icon].

> **Note:**
> While recording for one node, you can simultaneously record another node. This is achieved by using the Ctrl + Click option to select another node while actively recording, and again pressing ![icon] to begin recording. To switch between the nodes use the Ctrl + Click option.

DOM Watcher View Icons

The DOM Watcher View includes the following icons:

	Start watching events for the selected node.
	Stop watching events for the selected node.
	Clear the event list.
	View menu. Select **Settings...** to open the "Supported DOM Event types" dialog.

Working with the Ajax DOM Compare View

The *DOM* Compare view compares DOM attributes, child nodes, and CSS properties of a node.

This view will not appear automatically when selecting the Web Browser Tools Perspective. To open it go to **Window | View | DOM Compare**.

Comparing Nodes

This procedure describes how to compare nodes. This is allows you to visually see the differences between nodes. This feature compares DOM attributes, child nodes, and CSS properties of a node.

To compare nodes:

1. Go to **Window | Open Perspective | Web Browser Tools**.

2. Click 🌐 from the main toolbar.

   The "Open URL…" dialog opens.

3. Enter a URL in the "Enter URL…" text field and press **Enter**.

   The URL opens in the Internal Web Browser.

4. Open the DOM Compare view.

5. Select a node from the DOM Inspector view and drag it into the "Compare List:" field.

   Repeat step 3 for as many nodes as you want to compare.

6. Highlight the nodes to compare in the "Compare List:" field.

7. In the "Compare:" field choose from the following:

   ▪ DOM Attributes – To compare the DOM attributes of the selected nodes.

   ▪ Child Nodes – To compare the child nodes of the selected nodes.

   ▪ CSS – To compare the CSS properties of the selected nodes.

8. Press **Compare**.

   The details will appear in the view organized by columns. Each column is for a different node.
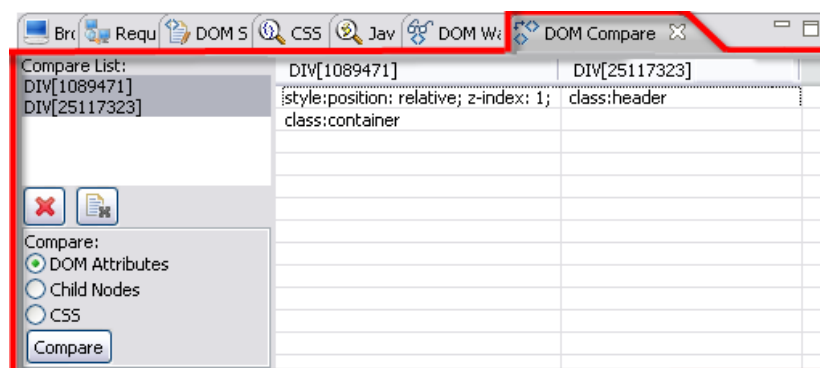
To clear nodes from the "Compare List:" field, click the ![X] to remove only one node or ![remove list icon] to remove the entire list.

Integrating with VMWare Workstation

Zend Studio allows you to integrate with VMware Workstation so that you can easily execute your project on a virtual machine. Working with a virtual machine allows you to develop your code on one operating system and execute it on a different one, all while working on one machine.

Prerequisites

The following components must be installed prior to the integration:

- Zend Studio 9.0 or above
- VMware Workstation 7.x or above, available for download at
  http://www.vmware.com/products/workstation/
- A configured virtual machine.
  The Zend Server image can be downloaded at
  http://www.zend.com/en/products/studio/downloads. See Importing the ZendServer.zip Image File into VMware Workstation to learn how to import the downloaded image, or see Setting Up a Custom Virtual Machine to learn how to create your own.

Once you have met all the required prerequisites you can work with your VMware virtual machine to:

- Manage Virtual Machine Connections
- Define a VMware Run/Debug Configuration
- Work with Multiple Virtual Machines
- Debug a PHP Application on a Virtual Machine
- Run a PHP Application on a Virtual Machine

Importing the ZendServer.zip Image File into VMware Workstation

Zend Studio allows you to integrate with VMware Workstation so that you can easily execute your project on a virtual machine. Working with a virtual machine allows you to develop your code on one operating system and execute it on a different one, all while working on one machine.

Before importing the ZendServer.zip image file into VMware Workstation, make sure Zend Studio and VMware Workstation are open and running, and that you have downloaded and extracted the ZendServer.zip file. See Prerequisites for more information on where to download these components.

Note:

If you would like to create your own image file instead of using the provided ZendServer.zip file, see Setting Up a Custom Virtual Machine Image. After creating your image file you do not need to import the image into VMware Workstation.

To import the ZendServer.zip image file into VMware Workstation:

1. Extract the ZendServer.zip file. The file is available for download at http://www.zend.com/en/products/studio/downloads under the name "Virtual development server (Ubuntu Linux)".
   This extracted file provides you with the OVF and VMDK files you need.

2. From the Main menu of your VMware screen select **File | Import or Export**.
   The Conversion wizard opens.
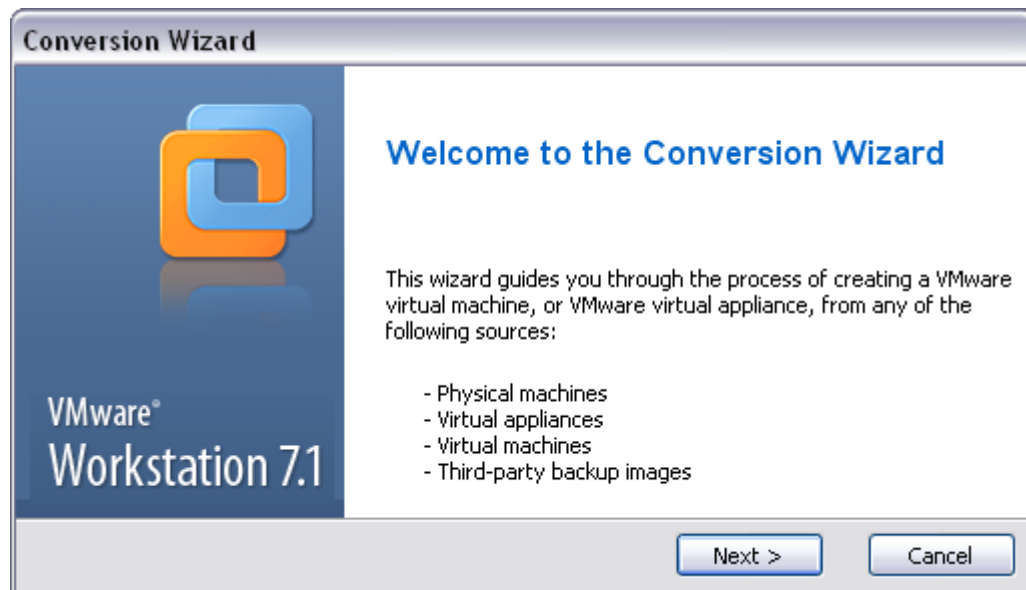
3. Click **Next** to open the Source dialog and click **Next**.

The Source Type dialog opens.
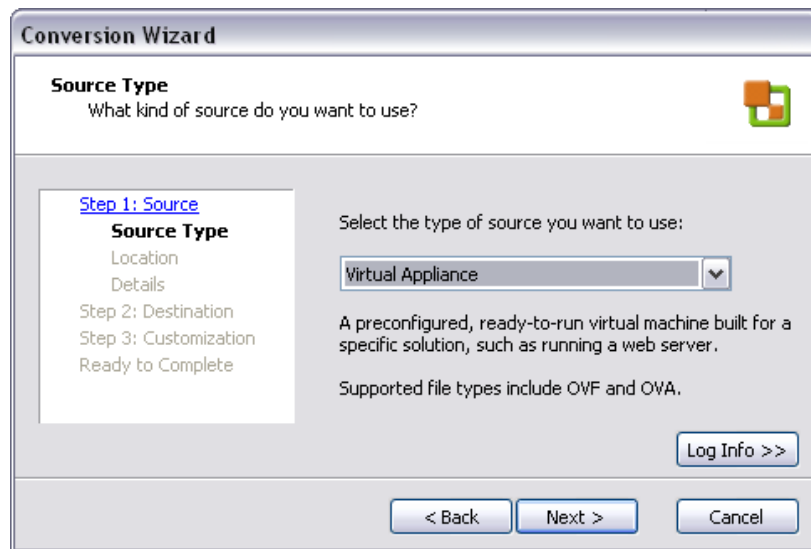
4. In "Step 1: Source" of the Conversion wizard select **Virtual Appliance** form the Source Type dropdown menu.

The Virtual Appliance dialog opens.

5. Select File System and browse to the OVF file in your extracted ZendServer.zip folder and select it.

6. Click **Next**.

The Virtual Appliances Details dialog opens.

7. Review the details of the virtual appliance you have selected. To change something click **Back**. If the details are correct click **Next** to open the Destination dialog.

8. In the Destination dialog click **Next** to open the Destination Type dialog.

9. In the Select the destination type drop-down menu select Other Virtual Machine and click **Next**.

    The Virtual Machine Name and Location dialog opens.

10. Fill in the following components in the dialog:

- Virtual Machine Name - Enter the name you would like to use for your virtual machine.

- Location - Enter the location where your virtual machine will sit or click **Browse** to browse to a location.

- Type of virtual machine to create - Select VMware Workstation from the drop-down menu.

- Version - Select version 7.x or above.

11. Click **Next** to open the Virtual Machine Options dialog.

12. The default settings in this dialog are:

- How do you want to convert your disks? - Import and convert (full-clone)

- Disk Allocation - Allow virtual disk files to expand.

    If you would like to keep the default settings, click **Next**.

    The Networks dialog opens.

13. The default setting in the Destination Network drop-down menu is Bridged.

   If you wish to keep the default setting, click **Next** to open the Customization dialog.

14. Click **Next** to open the Ready to Complete dialog.

15. Review the details of your virtual appliance. If you would like to make any changes click **Back** until you get to the relevant window.

   If the details are all correct click **Finish**.

   Your virtual machine has been created and installed in VMware Workstation.

After importing the image file, your virtual machine is created, and is represented by a ZendServer.vmdx file.

You can now [run a PHP application on your virtual machine](#) or [debug a PHP application on your virtual machine](#).

Note:

A virtual machine created with the ZendServer.zip image file has a pre-defined user ID and password that you will need to enter in order to use the virtual machine:

   - User ID - Studio

   - Password - logitech

Creating a Custom Virtual Machine

This procedure describes how to create your own virtual machine. A virtual machine is necessary in order to integrate with VMware Workstation. If you would like to use the pre-configured Zend Server image to create your virtual machine see Importing the ZendServer.zip Image File into VMware Workstation. Before creating a custom virtual machine in VMware Workstation, make sure Zend Studio and VMware Workstation are open and running. See Prerequisites for more information on where to download these components.

Note:
This procedure applies to Linux virtual machines only.

To create a custom virtual machine:
1. Open VMware Workstation and create a Linux virtual machine that is compatible with Zend Server.
2. Power on your virtual machine in VMware Workstation.
3. Install Zend Server 5 or above on your virtual machine. For information on Zend Server installation see http://static.zend.com/topics/Zend-Server-5-Installation-Guide-100421.pdf.
4. Edit the document root according to your operating system:
   - Ubuntu - Use a text editor to open the file located in "/etc/apache2/sites-available/default", and edit the document root both times that it appears in the file from "/var/www" to "/mnt/hgfs".
   - RPM and Fedora - Use a text editor to open the file located in "/etc/httpd/conf/httpd.conf", and edit the document root in the file from "/var/www/html" to "/mnt/hgfs".
5. Save the file and restart your virtual machine.

Your custom virtual machine has been created.

Important Note:
After installing your virtual machine, shared folders are disabled by default. To enable shared folders go to the VMware Workstation main toolbar and select **VM | Settings | Options tab | Shared Folders** and select 'Always enabled'. Click **OK** to save the settings.

Once you have a custom virtual machine created, you can access it through your VMware Workstation or you can run a PHP application on your virtual machine or debug a PHP application on your virtual machine in Zend Studio.

Working with VMware Virtual Machines

Zend Studio allows you to integrate with a local VMware Workstation so that you can easily execute your project on a virtual machine. Working with a virtual machine allows you to develop your code on one operating system and execute it on a different one, all while working on one machine.

Working with VMware Virtual Machines allows you to:

- Manage Virtual Machine Connections - Add and delete virtual machine connections for run and debug purposes.
- Define a VMware Run/Debug Configuration - Define a VMware run/debug configuration, which allows you to work with multiple virtual machines.
- Work with Multiple Virtual Machines - Choose a virtual machine to use for run/debug purposes when you have more than one virtual machine defined in your Zend Studio.
- Debug a PHP Application on a Virtual Machine - Debug your PHP application on a VMware virtual machine.
- Run a PHP Application on a Virtual Machine - Run your PHP application on a VMware virtual machine.

Managing Virtual Machine Connections

This procedure describes how to manage your virtual machine connections, allowing you to control which virtual machine connections are defined and accessible in your Zend Studio. If you would like to define your virtual machine connection during the run or debug process see Running a PHP Application on a Virtual Machine or Debugging a PHP Application on a Virtual Machine.

Defining a Virtual Machine Connection

This procedure describes how to define a virtual machine connection in Zend Studio without running or debugging your PHP application. Defining a virtual machine connection allows you to Run or Debug a PHP application on a virtual machine.

To define a virtual machine connection you must have already created a VMware Workstation virtual machine by Importing the ZendServer.zip Image File into VMWare Workstation or Creating a Custom Virtual Machine Image.

To define a virtual machine connection:

1. From the main toolbar click ⊞ .

    The VMware Virtual Machine Connection wizard opens.
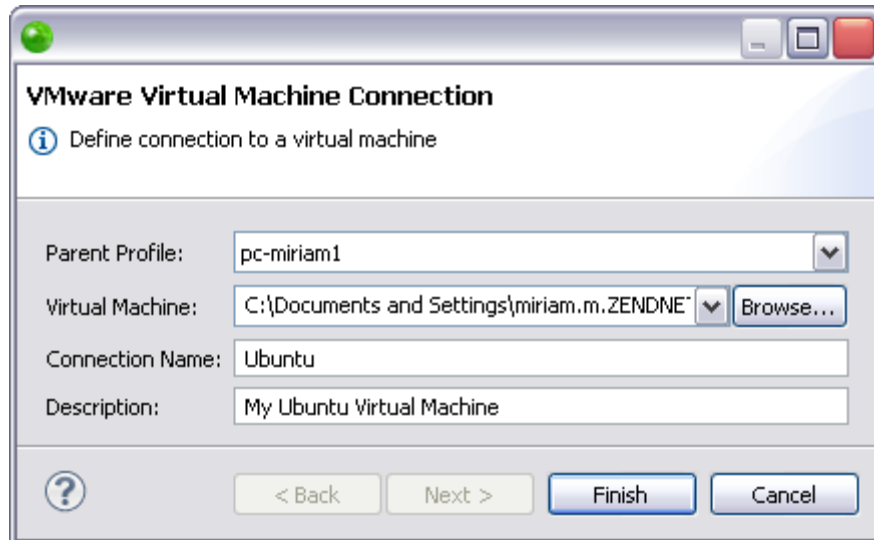
2. The dialog contains the following components:

    - Parent Profile - The host computer.

    - Virtual Machine - The .vmx file that represents your virtual machine. This is produced after importing the ZendServer.zip image file into VMWare Workstation or creating a custom virtual machine image.

    - Connection Name - The name you would like to give the connection.

    - Description - A field where you can insert an explanation of the defined virtual machine connection.

3. Click **Finish**.

    The virtual machine connection is defined in Zend Studio. To see the virtual machines you have defined go to the Remote Systems view.

Once you have defined you virtual machine you can define more specific settings by defining a VMware run/debug configuration, or you can run a PHP application on a virtual machine, or debug a PHP application on a virtual machine.

Deleting a Virtual Machine Connection

This procedure describes how to delete a virtual machine connection. Deleting a virtual machine connection only deletes it from your Zend Studio, not from VMware Workstation. PHP applications cannot run or debug on virtual machines that do not have an active connection with Zend Studio. Before deleting a virtual machine connection, you must first Define a Virtual Machine Connection.

To delete a virtual machine connection:

1. Go to the Remote Systems view, which can be manually accessed by going to **Window | Show View | Other | Remote Systems | Remote Systems.**

   The Remote Systems view opens.
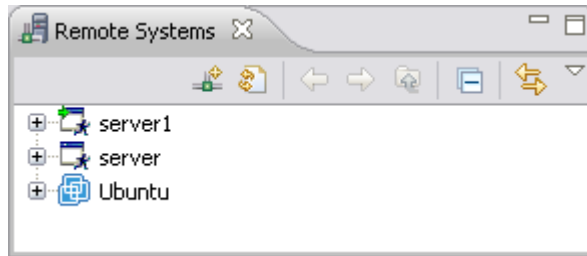
2. From the Right Click Menu of your virtual machine select **Delete**.

   Your virtual machine connection has been deleted from Zend Studio.

You can now define a new or previously deleted virtual machine. See Defining a Virtual Machine Connection for more information.

All virtual machines which are defined in your Zend Studio can be viewed in the Remote Systems view, which can be accessed by going to **Window | Show View | Other | Remote Systems | Remote Systems**.

Defining a VMware Run/Debug Configuration

This procedure describes how to define a VMware run or debug configuration. This configuration can be used when choosing to run or debug your PHP application, and is necessary in order to work with multiple virtual machines. The Configurations dialog allows you to specify additional information for your run or debug configuration.

Before defining a VMware run/debug configuration you must define a virtual machine connection, which can also be done while running a PHP application on a virtual machine or debugging a PHP application on a virtual machine.

To define a VMware run or debug configuration:

1. Click in the main toolbar and select **Debug Configurations** -or- Go to **Run | Debug Configurations**.

   The Debug Configurations dialog opens.

2. Double-click the PHP Application on VM option to create a new debug configuration.
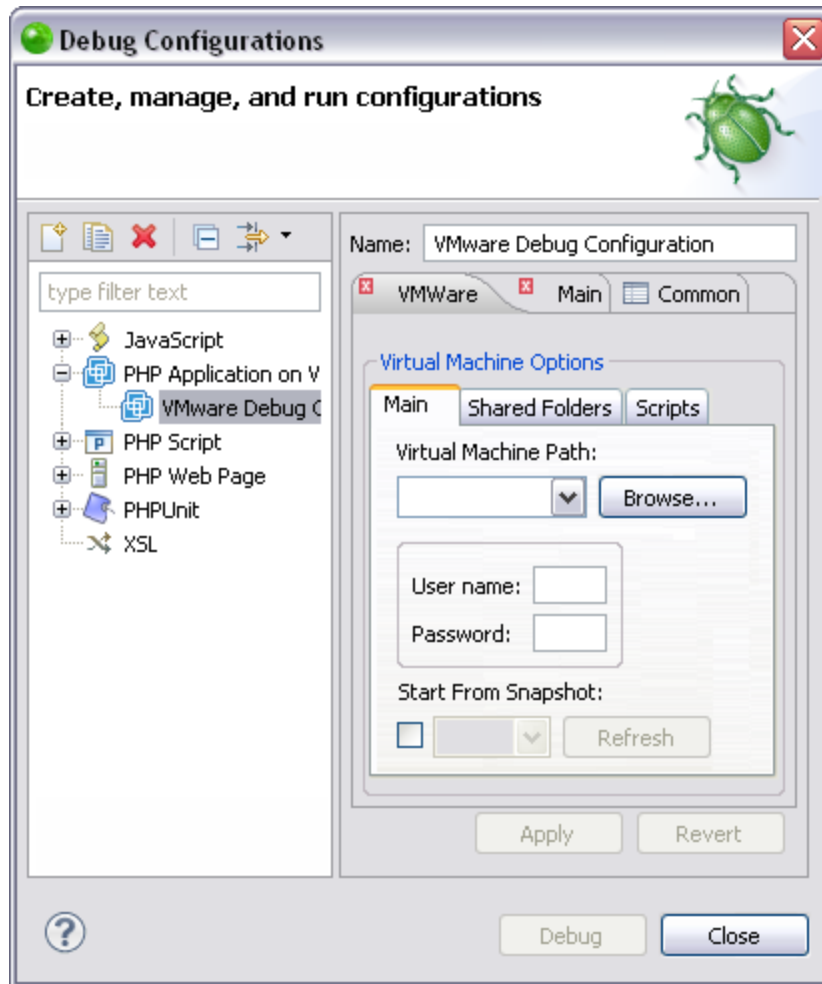
3. The dialog consists of the following fields:

- Main tab

    o Name - Enter a name for the new configuration.

    o Virtual Machine Path - Click **Browse** to browse to your virtual machine.

    o Username - The username of your virtual machine. This is defined in your virtual machine.

    o Password - The password of your virtual machine. This is defined in your virtual machine.

    o Start from Snapshot - Starts the debug session from a selected snapshot in your virtual machine. For more information see VMware KB - Working with snapshots.

- Shared Folders tab - This page allows you to define a folder as a shared folder between your local machine and your virtual machine. This will allow both machines to have access to the folder.

- o Name - The name you define for the shared folder.
- o Host Path - The path to the shared folder on its original machine.
- o Add - Click **Add** to define a new shared folder by defining the name and host path in the Add a New Shared Folder dialog.
- o Edit - Click **Edit** to change the name or host path of a shared folder.
- o Remove - Select the shared folder you would like to delete and click **Remove**.
- Scripts tab - This page allows you to run a specific script before and/or after the debug session.
  - o In the "Before Launch, Run Script" text field, paste the script you would like to run before the debug session begins.
  - o In the "After Termination, Run Script" text field, paste the script you would like to run after the debug session is terminated.

4. Click **Apply** and then **Close**.

   Your run or debug configuration has been saved.

You can now run a PHP application on a virtual machine or debug a PHP application on a virtual machine with your new configuration.

All virtual machines which are defined in your Zend Studio can be viewed in the Remote Systems view, which can be accessed by going to **Window | Show View | Other | Remote Systems | Remote Systems**.

Working with Multiple Virtual Machines

This procedure describes how to specify which virtual machine you would like to run or debug your PHP application on, when you have more than one virtual machine defined in your Zend Studio.
Before specifying which virtual machine you would like to use, you must first define VMware run/debug configurations for the virtual machines you would like to choose from.

To work with multiple virtual machines:

1. Select the file you would like to run/debug in the PHP Explorer view.

2. Click **Run | Run Configurations** or **Run | Debug Configurations** depending on which action you would like to take.

The Run Configurations or Debug Configurations dialog opens.
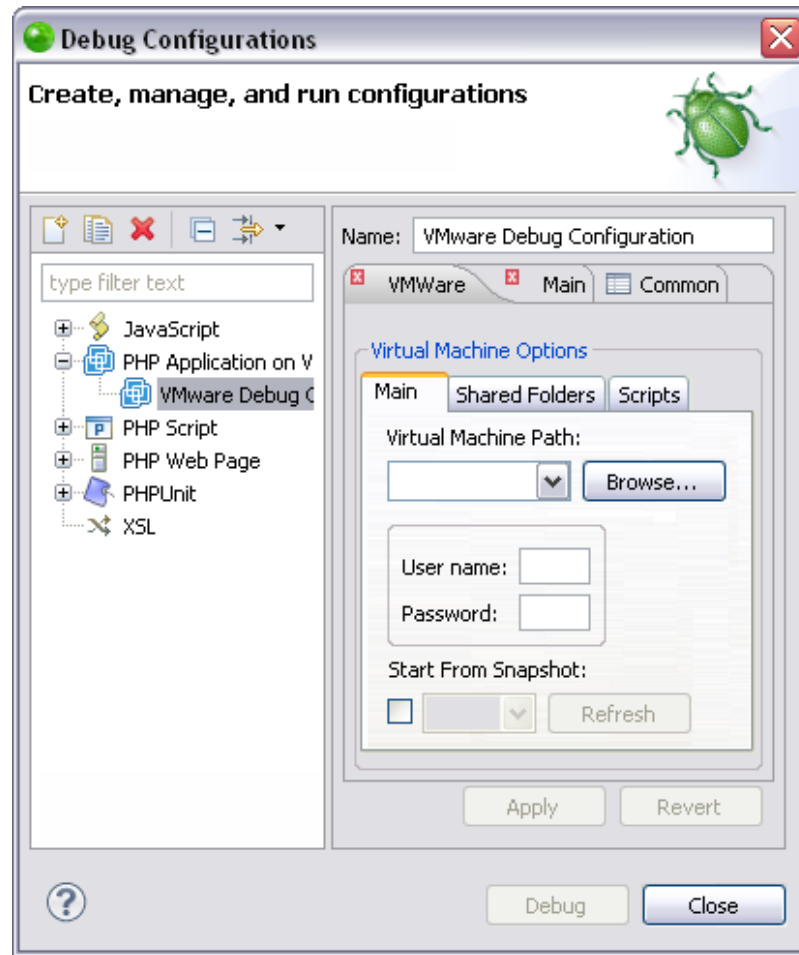
3. Select the configuration for the virtual machine you would like to use in this instance and click **Apply** and then **Debug** or **Run**.

Your run or debug session begins. For more information see Running a PHP Application on a Virtual Machine or Debugging a PHP Application on a Virtual Machine.

Next time you execute your application, you can select which virtual machine you would like to use by repeating this procedure.

All virtual machines which are defined in your Zend Studio can be viewed in the Remote Systems view, which can be accessed by going to **Window | Show View | Other | Remote Systems | Remote Systems**.

Debugging a PHP Application on a Virtual Machine

This procedure describes how to debug a PHP application on your virtual machine. Debugging on a virtual machine allows you to debug your PHP application on different operating system than the operating system it was developed on, all from your Zend Studio interface.

Before debugging your PHP application on a virtual machine you must first create a VMware Workstation virtual machine by Importing the ZendServer.zip Image File into VMware Workstation or Creating a Custom Virtual Machine.

To find out how to define a virtual machine connection without debugging your PHP application see Managing Virtual Machine Connections.

> **Note:**
>
> If you have already defined a VMware run/debug configuration or defined a virtual machine in a previous VMware debug or run session, clicking **Debug As | Debug as PHP Application on VM** will automatically open the previously defined VMware Workstation virtual machine and begin the debug session. For information on how to select a different virtual machine see Working with Multiple Virtual Machines.

To debug a PHP application on a virtual machine:

1. Set breakpoints in the desired lines of your project. For more information see Setting Breakpoints.

2. From the Right Click Menu of your project select **Debug As | Debug as PHP Application on VM**.

   The VMware Virtual Machine Connection wizard opens.

> **Note:**
>
> If you have already defined a virtual machine connection, choosing to Debug as a PHP Application on VM will open the Enter Password dialog (step 5).
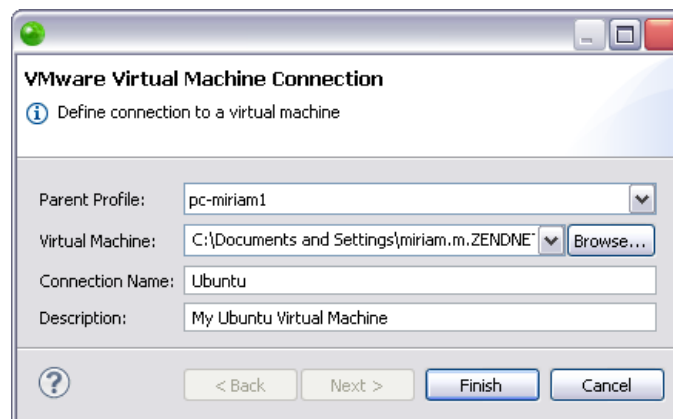
3. The dialog contains the following components:

- Parent Profile - The host computer.
- Virtual Machine - The .vmx file that represents your virtual machine. This is produced after importing the ZendServer.zip image file into VMWare Workstation or creating a custom virtual machine image.
- Connection Name - The name you would like to give the connection.
- Description - A field where you can insert an explanation of the defined virtual machine connection.

4. Click **Finish** to open the Enter Password dialog.

5. Enter the user ID and password of virtual machine that you are using. The dialog also allows you to see which host name and system type the dialog is referring to.

Note:

A virtual machine created with the ZendServer.zip image file has a pre-defined user ID and password:

- User ID - studio
- Password - logitech

6. Click **Apply** and then **Debug**.
   VMware Workstation opens and powers on your virtual machine.

7. Click **Yes** if asked whether to open the PHP Debug Perspective.
   The Debug session begins.

See the "Running and Analyzing Debugger results" topic for more information on the outcome of a debugging session.

You can now run the PHP application on the virtual machine to ensure that all bugs have been fixed.

All virtual machines which are defined in your Zend Studio can be viewed in the Remote Systems view, which can be accessed by going to **Window | Show View | Other | Remote Systems | Remote Systems**.

Running a PHP Application on a Virtual Machine

This procedure describes how to run your PHP application on a virtual machine. Running on a virtual machine allows you to run your PHP application on different operating system than the operating system it was developed on, all from your Zend Studio interface.

Before running your PHP application on a virtual machine, you must first create a VMware Workstation virtual machine by Importing the ZendServer.zip Image File into VMware Workstation or Creating a Custom Virtual Machine. To find out how to define a virtual machine connection without running your PHP application see Managing Virtual Machine Connections.

Note:

If you have already defined a VMware run/debug configuration or defined a virtual machine in a previous VMware run or debug session, clicking **Run As | Run as PHP Application on VM** will automatically open the previously defined VMware Workstation virtual machine and begin the debug session. For information on how to select a different virtual machine see Working with Multiple Virtual Machines.

To run a PHP application on a virtual machine:

1. From the Right Click Menu of your project select **Run As | Run as PHP Application on VM**.

   The VMware Virtual Machine Connection wizard opens.

Note:

If you have already defined a virtual machine connection, choosing to Run as PHP Application on VM will open the Enter Password dialog (step 4).
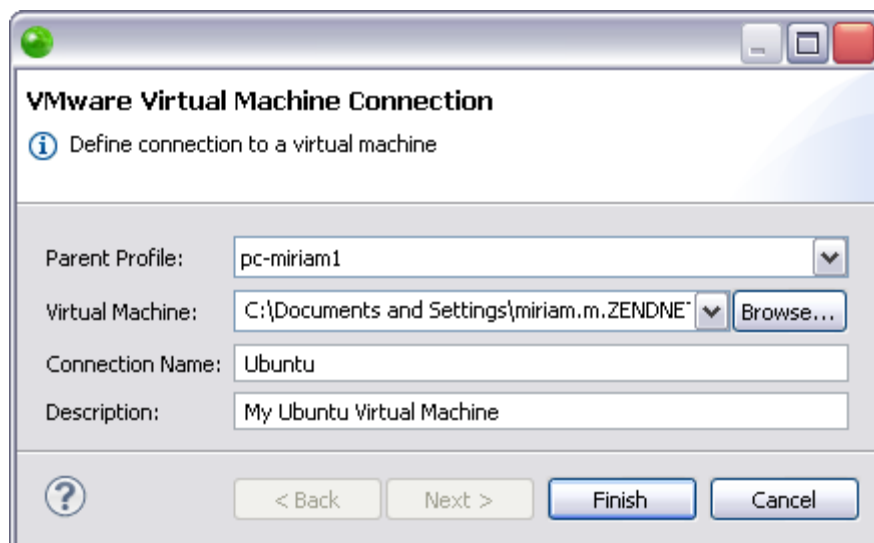
2. The dialog contains the following components:
    * Parent Profile - The host computer.
    * Virtual Machine - The .vmx file that represents your virtual machine. This is produced after importing the ZendServer.zip image file into VMWare Workstation or creating a custom virtual machine image. Click **Browse** to locate the correct file.
    * Connection Name - The name you would like to give the connection.
    * Description - A field where you can insert an explanation of the defined virtual machine connection.
3. Click **Finish** to open the Enter Password dialog.

4. Enter the user ID and password of virtual machine that you are using. The Host name field allows you to see which system type and virtual machine the dialog is referring to.

Note:

A virtual machine created with the ZendServer.zip image file has a pre-defined user ID and password:

      - User ID - studio

      - Password - logitech

The virtual machine powers on and the Run PHP Web Page dialog opens.

5. A URL is configured based on the default URL of a new server in the Run PHP Web Page dialog.
6. Click **OK**.

    Your application runs on the virtual machine and the output is open in your Zend Studio.

To delete a virtual machine from your Zend Studio go to the Remote Systems view and from the Right Click Menu of your virtual machine select **Delete.**

You can now [debug your application on your virtual machine](#).

All virtual machines which are defined in your Zend Studio can be viewed in the [Remote Systems view](#), which can be accessed by going to **Window | Show View | Other | Remote Systems | Remote Systems**.

Using the Amazon AWS Toolkit

The Amazon AWS Toolkit for Zend Studio simplifies the development cycle when using remote Amazon EC2 Instances by allowing PHP applications to be developed and debugged while working with Amazon and Zend Server DevPay Instances.

Installing the Amazon AWS Toolkit Plugin

This procedure describes how to install the Amazon AWS Toolkit plugin on top of your Zend Studio. You will need to install the Amazon AWS Toolkit plugin in order to be able to work with Amazon AWS Toolkit for Zend Studio.

System Requirements

- Zend Studio 9.0 and above

To install the Amazon AWS Toolkit plugin:

1. When Zend Studio opens, the Welcome Page appears. You can also open the Welcome Page by going to **Help | Welcome** if your product is already open.
2. In the Welcome Page's Plugin List mark the 'Amazon Web Services support' checkbox and click **Apply changes**.
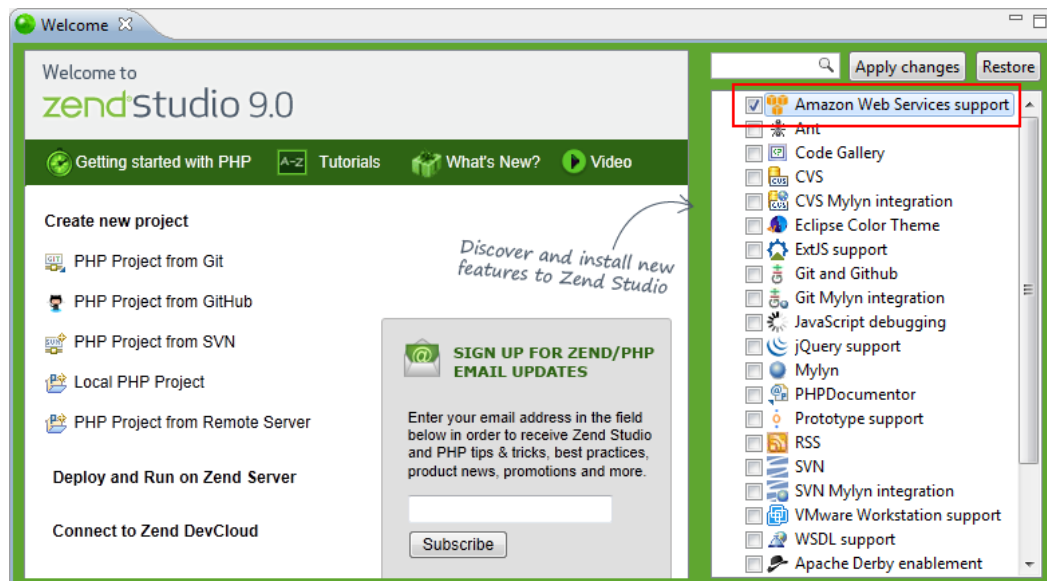
3. Your Zend Studio restarts with the mazon AWS Toolkit plugin installed.
   For information on uninstalling the plugin see Uninstalling the Amazon AWS Toolkit Plugin.

You can now begin to set up the Amazon AWS Toolkit for Zend Studio by meeting the prerequisites.

Uninstalling the Amazon AWS Toolkit Plugin

To uninstall the mazon AWS Toolkit plugin:

1. When Zend Studio opens, the Welcome Page appears. You can also open the Welcome Page by going to **Help | Welcome** if your product is already open.
2. In the Welcome Page's Plugin List unmark the 'Amazon Web Services support' checkbox and click **Apply changes**.

Your Zend Studio restarts without the mazon AWS Toolkit plugin installed.

Without the mazon AWS Toolkit plugin installed, the features of Amazon AWS Toolkit for Zend Studio will not be available. To begin using the Amazon AWS Toolkit for Zend Studio, you must install the Amazon AWS Toolkit plugin.

Prerequisites

See the Zend Server AMI User Guide, available from the Zend Server on Amazon EC2 Forum, for more information on the Amazon AWS Toolkit for Zend Studio prerequisites and getting started with your Zend Server AMI.

Before meeting the prerequisites you must install the Amazon AWS Toolkit plugin.

The prerequisites for Amazon AWS Toolkit for Zend Studio are:

1. Create the following accounts:
    a. An Amazon Web Services account (http://aws.amazon.com/)

       All the information you will need in order to create an Amazon EC2 account and understand how to work with the Amazon EC2 web service can be found in Amazon's Getting Started Guide.

       After opening an account with Amazon AWS you will receive the following credentials:
         i. An Access Key ID
         ii. A Secret Key ID
    b. A Zend Account

2. Purchase and Activate Your Zend Server AMI - For more information see the Zend Server AMI User Guide, available from the Zend Server on Amazon EC2 Forum.

3. Adding an SSH Security Group for your Zend Server AMI - Configuring Security Groups for your Zend Server AMI allows you to specify which connection methods can communicate with your Amazon EC2 Instance.

   You can also add a new Security Group from inside of Zend Studio.

4. Create a Key Pair and launch your Zend Server AMI

   It is also possible to create a Private Key and launch your Zend Server Instance from inside of Zend Studio.

Once you have met all of the prerequisites you can set up your Zend Server Amazon EC2 Instance.

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

Setting Up Your Zend Server Amazon EC2 Instance

Once you have met all the prerequisites of Amazon AWS Toolkit for Zend Studio you can set up your Zend Server Amazon EC2 Instance, which allows you to use the functionalities of the mazon AWS Toolkit in your Zend Studio.

Once you have met the prerequisites you can:

1. Set Your Environment As An Allowed Host on Your Zend Server Amazon EC2 Instance
2. Set Up Your Amazon EC2 Instance in Zend Studio
3. Add an Amazon EC2 Remote Connection Profile
4. Enable Tunneling for Your Zend Server Amazon EC2 Instance

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

Setting Your Environment as an Allowed Host on Your Zend Server Amazon EC2 Instance

Setting your environment as an allowed host on your Zend Server Amazon EC2 Instance allows Zend Studio's debugger to communicate with your Zend Server. This allows you to debug your PHP application as a PHP web page on your Amazon EC2 Instance directly from your Zend Studio GUI.
Before setting your environment as an allowed host on your Zend Server, you must launch your Zend Server Amazon EC2 Instance.

To configure your Zend Server AMI settings:

1. In your Zend Server GUI go to **Server Setup | Debugger**.
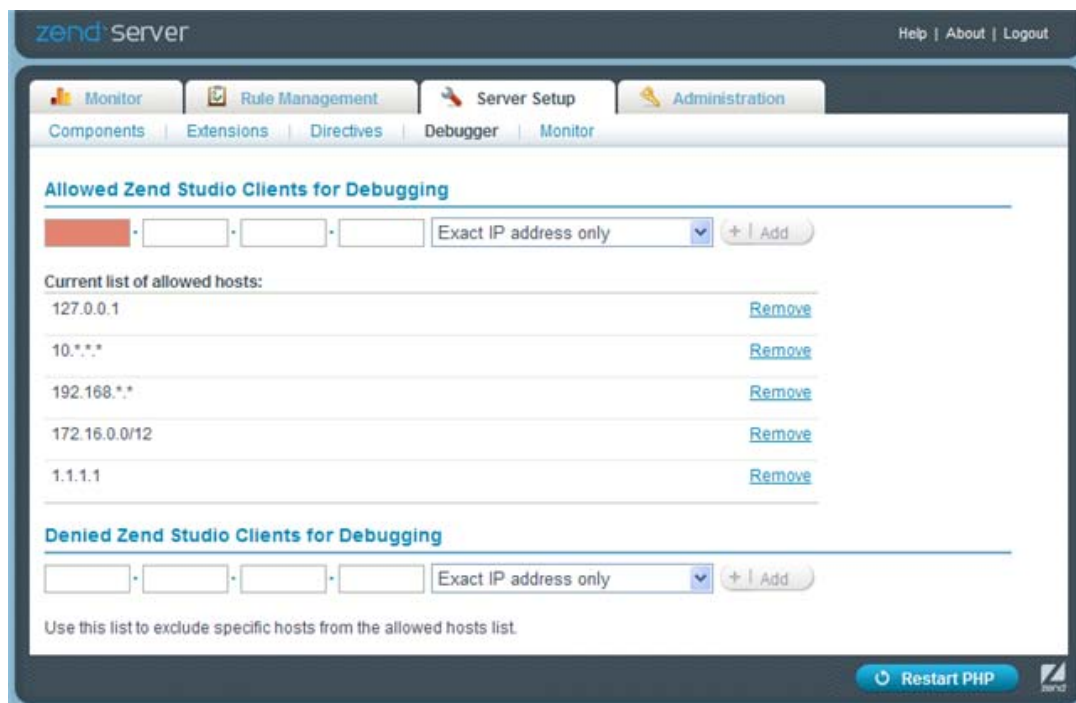
2. In the Allowed Zend Studio Clients for Debugging section, enter your machine's public IP and click **Add**.

Note:
To find out your machine's public IP, go to http://www.whatsmyip.org/.

3. Restart your PHP by clicking **Restart PHP** in the Zend Server GUI.
   Your Zend Server AMI is now configured as an allowed host.

For more information on the Zend Server GUI see the Zend Server Online Documentation.

You can now set up your Amazon EC2 Instance in Zend Studio.

Setting Up Your Amazon EC2 Instance in Zend Studio

This procedure describes how to set up your Zend Server Amazon EC2 Instance in your Zend Studio to activate the functionalities of Amazon AWS Toolkit for Zend Studio.

Before setting up your Amazon EC2 Instance in Zend Studio, you must meet all the prerequisites and set your environment as an allowed host on your Zend Server.

Entering Your Amazon EC2 Credentials into Zend Studio

This procedure describes how to enter the Amazon credentials you were given when opening an Amazon AWS Account into Zend Studio. Inputting these credentials is necessary in order to use the Amazon AWS Toolkit for Zend Studio.

To enter your Amazon EC2 credentials in Zend Studio:

1. In Zend Studio open the AWS Toolkit Preferences by going to **Window | Preferences | AWS Toolkit**.

   The AWS Toolkit Preferences page opens.
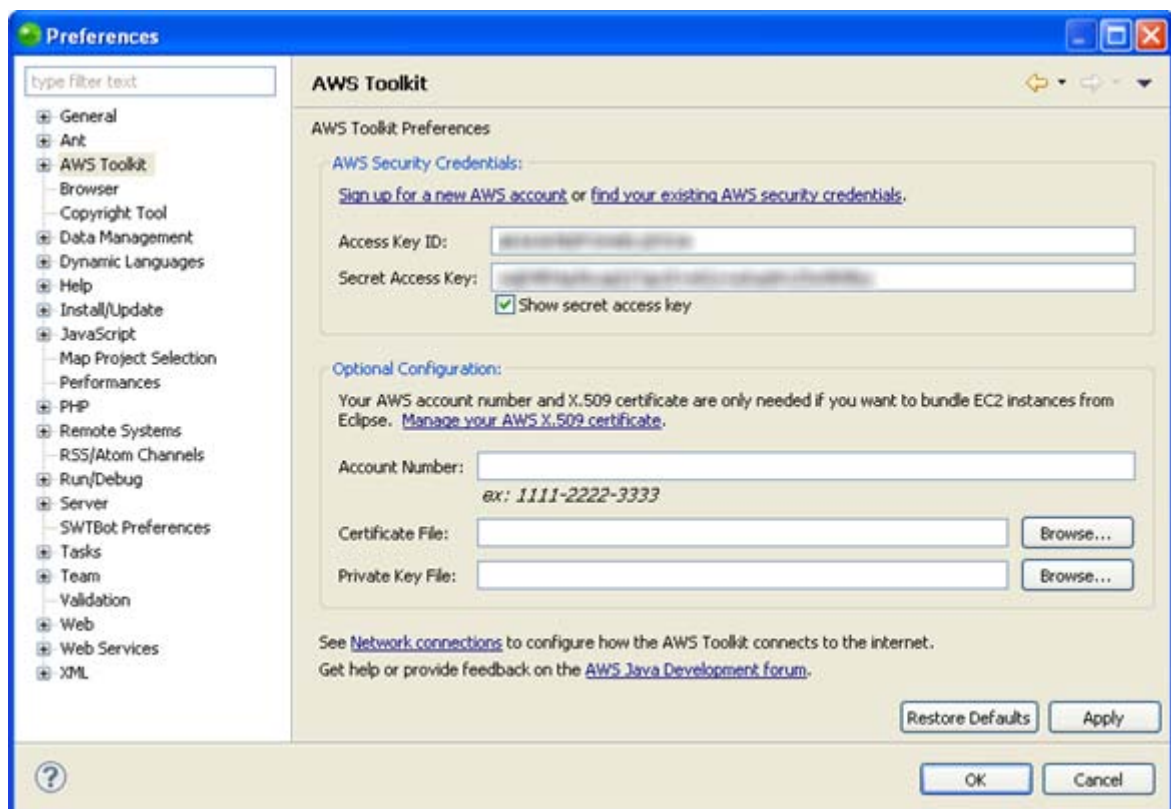
2. In the AWS Security Credentials section insert your Access Key ID and Secret Access Key and click **Apply** to save your settings.

3. In the Preferences page go to **General | Network Connections | SSH2**.

    The SSH2 Preferences page opens.
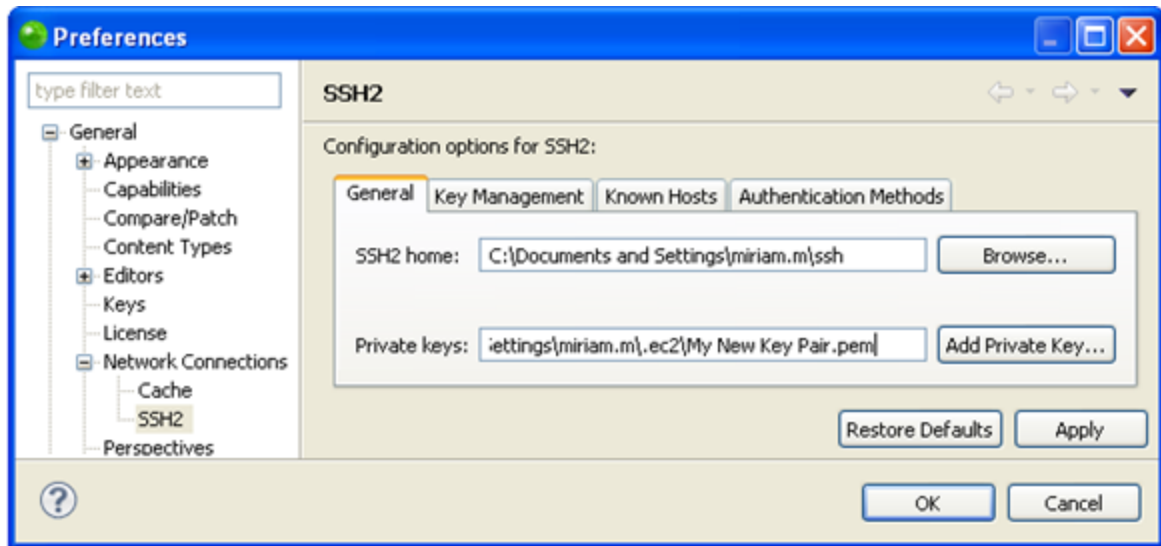
4. In the General tab click **Add Private Key**.
5. Browse to the location of your private key and select it. For more information on private keys see Creating a Private Key in Zend Studio.
6. Click **Apply** and **OK** to apply and save the settings.

    Your Amazon Instance is now identified by Zend Studio, and the Amazon Toolkit views are available.

Before using your Amazon EC2 Instance in Zend Studio you must add an Amazon EC2 Remote Connection Profile and enable Tunneling for your Zend Server Amazon EC2 Instance.

Defining Your Region for AMI's

This procedure describes how to define your region for Amazon AWS Toolkit for Zend Studio. The region you are in will change which AMI's are available to you, which you can see in the EC2 AMIs view.

To define your region for Amazon AWS Toolkit for Zend Studio:

1. Go to the Regions Preferences page which is manually accessible by going to **Window | Preferences | AWS Toolkit | Regions**.

   The Regions Preferences page opens.
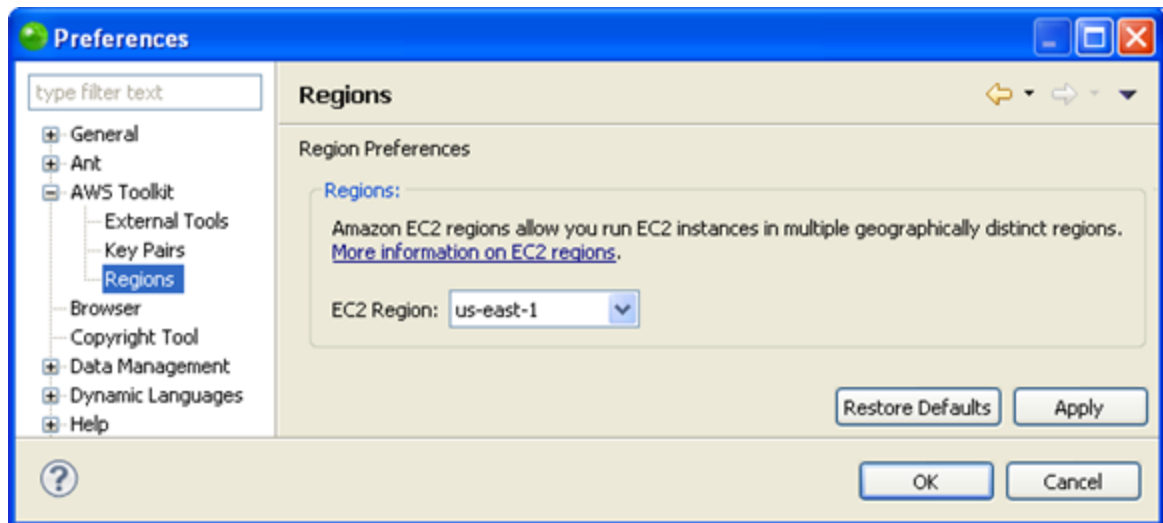
2. Select your region from the EC2 Region dropdown menu.
3. Click **Apply** and **OK** to apply and save your settings.

   Your region has been set.

The AMI's that are available in your region now appear in the EC2 AMIs view.

For information on which AMIs are available in each region see the Zend Server on Amazon EC2 section of the Zend website.

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

Adding an Amazon EC2 Remote Connection Profile

This procedure describes how to add a new Remote Connection Profile for your Amazon EC2 Instance. Adding a Remote Connection Profile allows you to define the connection settings for a remote server, in this case your Amazon EC2 Instance, which you can then associate with your project. Adding an Amazon EC2 Remote Connection Profile is necessary before enabling Tunneling for Your Zend Server Amazon EC2 Instance.

Before adding a profile, you must set up your Amazon EC2 Instance in Zend Studio.

To add a new Amazon EC2 Remote Connection Profile:

1. Go to **Project | Properties | Remote Server Support** - Or - Select **Properties | Remote Server Support** from the Right Click Menu in your project directory.

   The Remote Project Properties page opens.
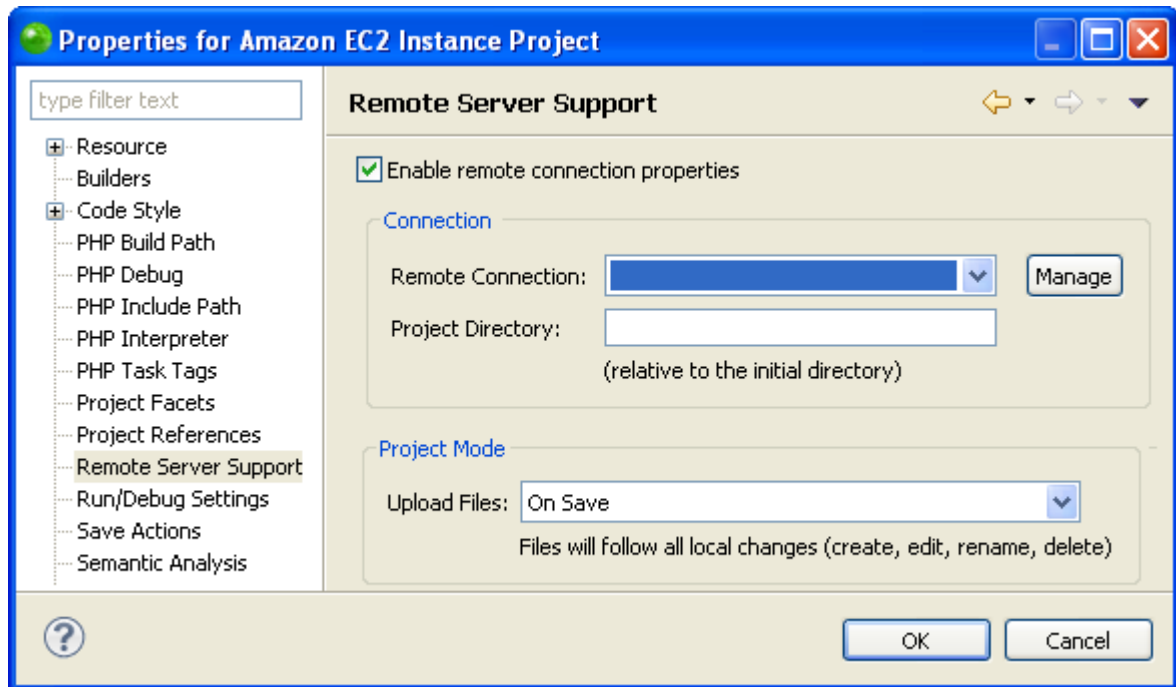
2. Mark the 'Enable remote connection properties' check-box and click **Manage**.

   The Create a Remote System Connection dialog opens.
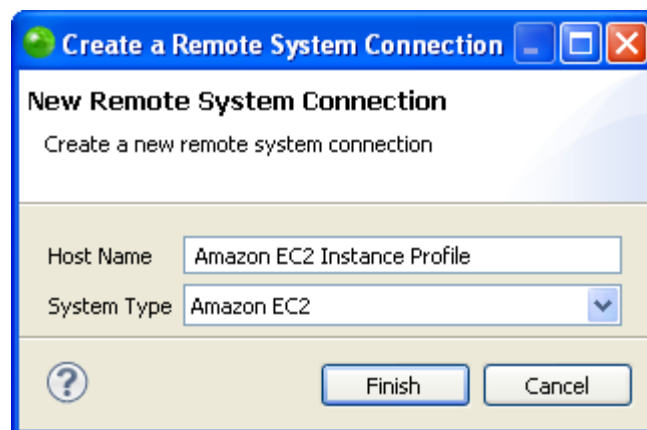
3. Enter a host name for your new remote system connection and select **Amazon EC2** from the System Type dropdown menu.

4. Click **Finish** to open the Remote Connection Profile dialog.
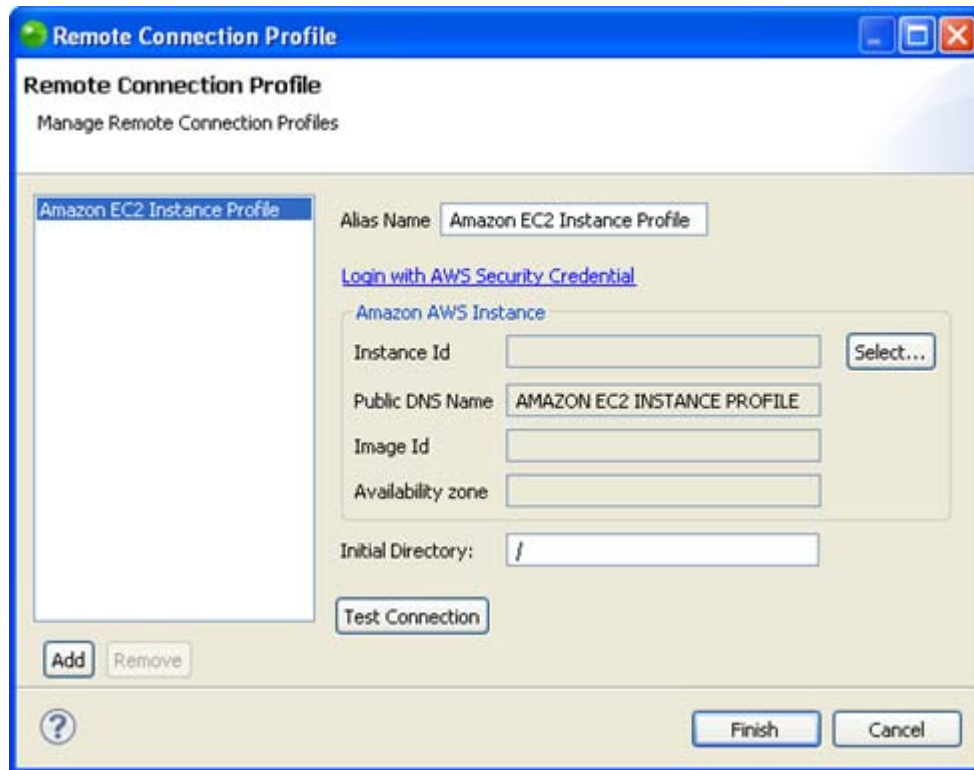
3. Click **Select**.

The Amazon AWS Instance Selection dialog opens.
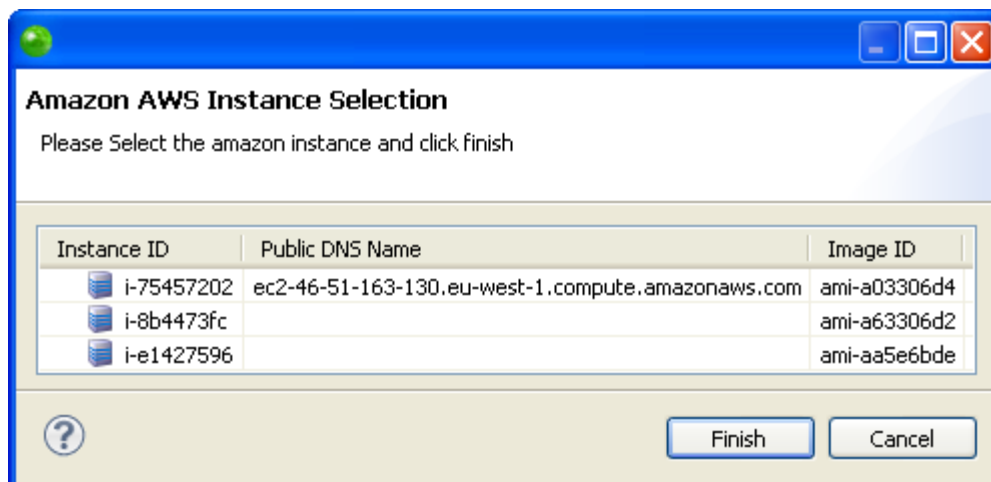
4. Select the Amazon EC2 Instance you would like to use and click **Finish**.

The details are automatically configured in the Remote Connection Profile dialog according to the Instance you have chosen.
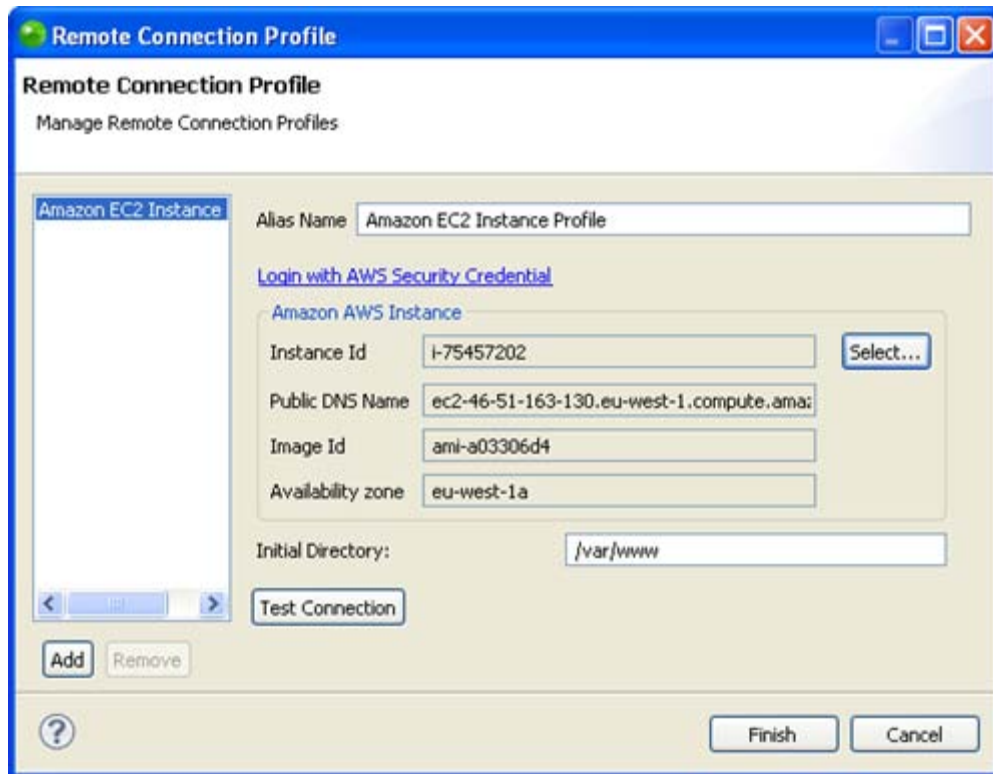
5. Click **Test Connection** to check that Zend Studio can connect to your Amazon EC2 Instance according to the current settings.

Note:
If the test connection fails, see Setting Up Your Zend Server Amazon EC2 Instance to review all the settings that need to be in place in order to for Zend Studio to be able to connect to your Instance.

6. Click **Finish** to save your new Remote Connection Profile and return to the Remote Server Support Properties page with your configured Remote Connection Profile selected.
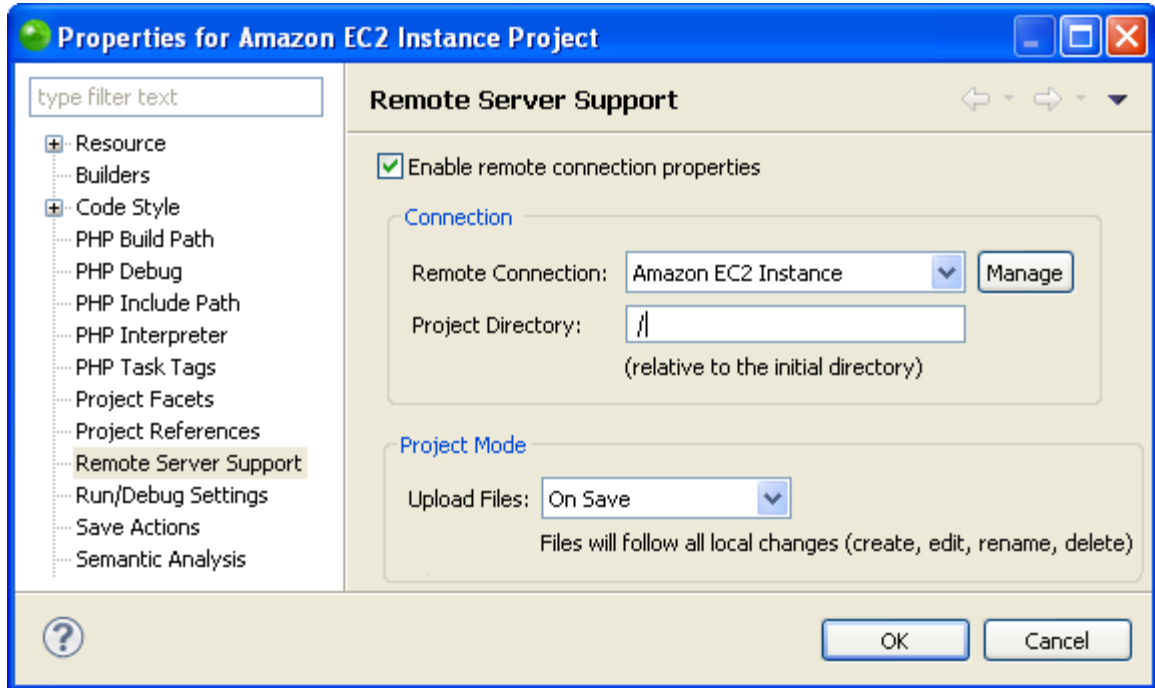
7. Select a Project Directory - A Project Directory is a folder within the Initial Directory that you would like to work with.

    The Project Directory is often the same as the project name.
8. Select an Upload Files mode - The Upload Files dropdown menu allows you to select to transfer files to the remote server in three ways:

    - Manually - Data is only transferred to/from the remote server when you manually perform the procedure. This option is available for uploading and downloading data to/from the remote server. See Uploading Manually or Downloading Manually for more information.
    - On Save - Data is transferred to the remote server every time you save your project or perform a change event. This option is only available for uploading data to the remote server. See Uploading On Save for more information.
    - On Run - Data is transferred to the remote server before you run or debug your project. This option is only available for uploading data to the remote server. See Uploading On Run for more information.

9. Click **OK** to save and apply your changes.

You can now enable Tunneling for your Zend Server Amazon EC2 Instance.

Once you have defined a Remote Connection Profile, you can associate it with any PHP remote project. This is done by selecting the associated Alias Name from the Remote Connection dropdown menu in the Remote Server Support Properties page of the project.

Enabling/Disabling Tunneling for Your Amazon EC2 Instance

This procedure describes how to enable and disable Tunneling for your Zend Server Amazon EC2 Instance. Tunneling provides a means of persistent connection between Zend Studio and a remote server, in this case your Amazon EC2 Instance, situated behind a Firewall or NAT. After creating a tunnel, all communication between Zend Studio and your Zend Server can go through that tunnel, instead of assigning more communication ports for the Debug sessions made on the Amazon EC2 Instance.

Enabling Tunneling for Your Amazon EC2 Instance

This procedure describes how to enable tunneling for your Zend Server Amazon EC2 Instance. Before enabling Tunneling for your Zend Server you must add an Amazon EC2 Remote Connection Profile for your Instance.

To enable Tunneling for your Zend Server Amazon EC2 Instance:
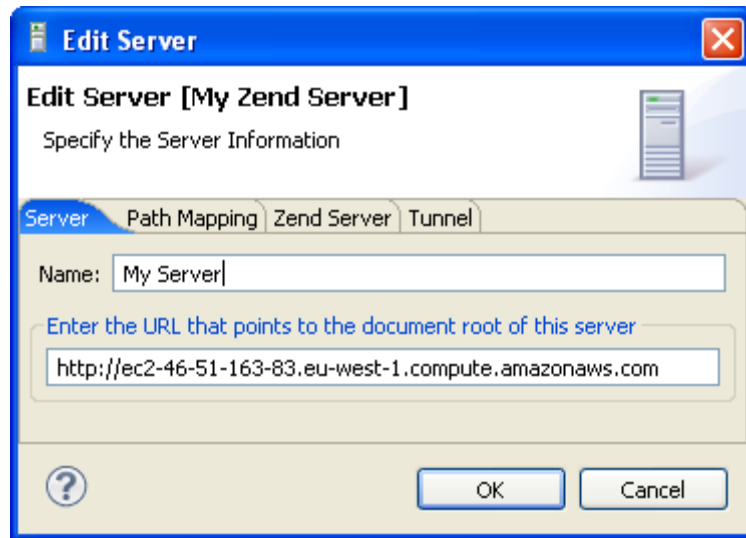
1. Go to **Window | Preferences | PHP | PHP Servers**.
   The PHP Servers Preferences page opens.
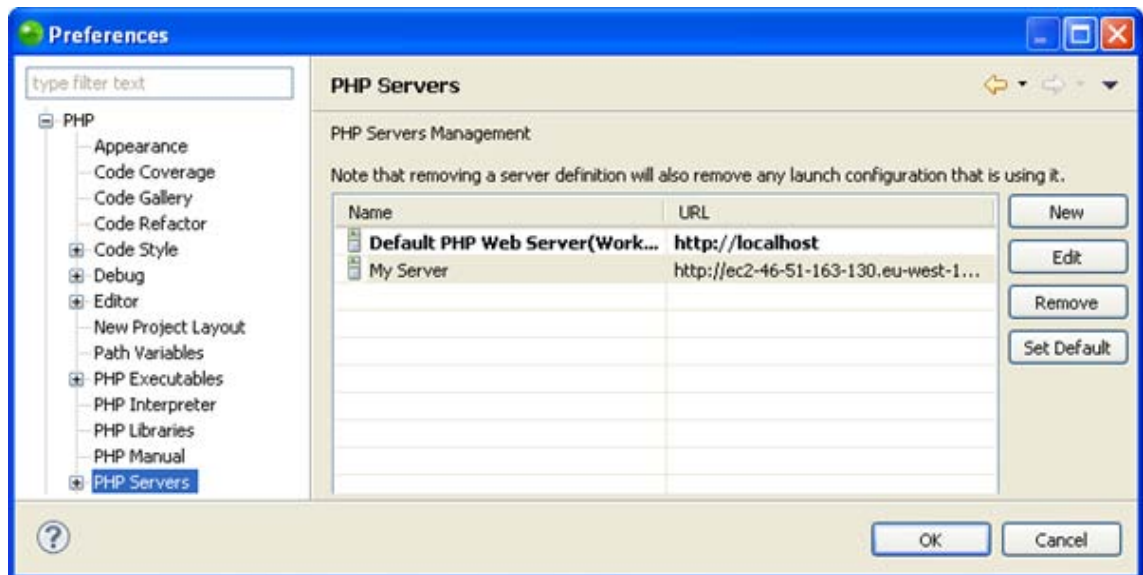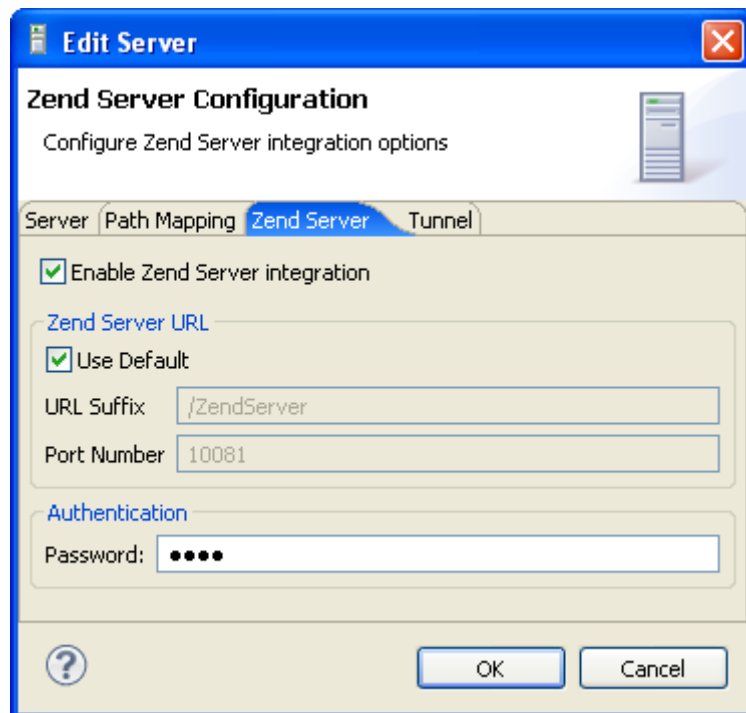
2. Select your Amazon Instance and click **Edit**.

   The Edit Server dialog opens.

3. In the Zend Server tab mark the 'Enable Zend Server Integration' check-box and the 'Use Default' check-box.

4. In the Password text field enter the password you defined in your Zend Server.

5. In the Tunnel tab mark the 'Enable Tunneling' check-box and click **OK**.

For more information on the Tunneling settings dialog see <u>Setting Up a Tunneling Server</u>.
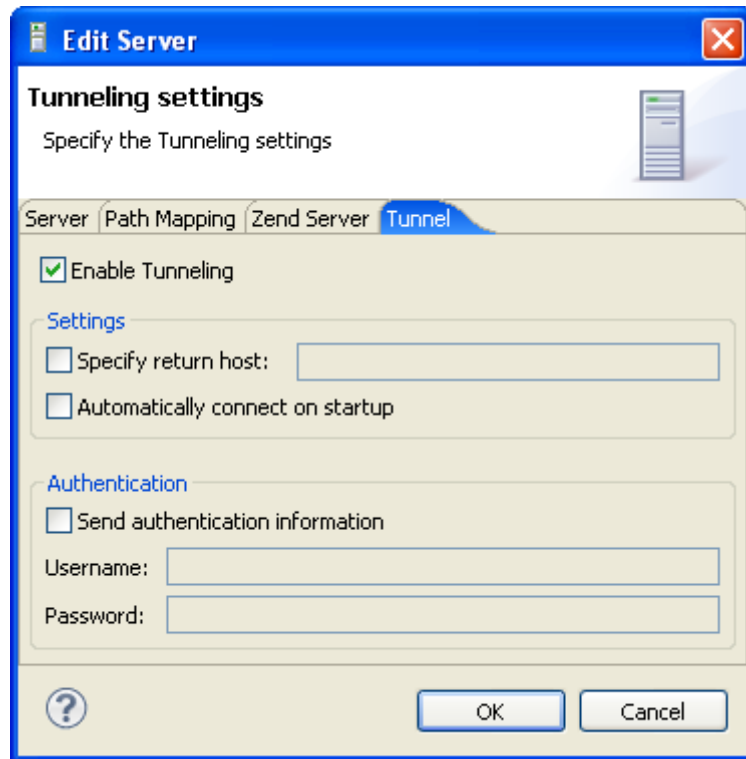
To confirm that Tunneling is enabled for your Amazon Instance, make sure that the icon in your
GUI is green ( ). If it is red ( ), select your Amazon EC2 Instance from the icon's
dropdown menu to enable Tunneling.

You can now <u>create a new PHP project with Remote Server Support with a Zend Server Amazon EC2
Instance</u> or <u>debug your PHP application as a PHP web page on your Zend Server Amazon EC2
Instance</u>.

Disabling Tunneling for Your Amazon EC2 Instance

Disabling Tunneling for your Amazon EC2 Instance will not allow Zend Studio to <u>debug your PHP
application as a PHP web page on your Amazon EC2 Instance</u>.

To disable Tunneling for your Amazon Instance:

1. Go to [icon] and from the dropdown menu select your Amazon Instance.

[icon]

✔ My Server (connected)
Ronen

Servers

2. If the icon is green ([icon]) click on it and Tunneling will be disabled.

If the icon is red ([icon]) Tunneling has already been disabled.

To enable Tunneling see Enabling Tunneling for Your Amazon EC2 Instance.

Debugging a PHP application on an Amazon Instance is not possible when Tunneling is disabled.

Working with Amazon AWS Toolkit for Zend Studio

Once you have met the prerequisites and set up your Zend Server Amazon EC2 Instance, you can work with your Instance. The Amazon AWS Toolkit for Zend Studio allows you to enjoy the benefits of the Amazon Elastic Compute Cloud and use them to help you develop, test, and stage your PHP application in Zend Studio.

Working with your Zend Server Amazon EC2 Instance allows you to do the following:

1. Create a New PHP Project with Zend Server Amazon EC2 Instance Remote Server Support
2. Create a Private Key in Zend Studio
3. Add a New Security Group in Zend Studio
4. Launch An Amazon EC2 Instance in Zend Studio
5. Terminate an Amazon EC2 Instance in Zend Studio
6. Debug as a PHP Web Page on Your Amazon EC2 Instance

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

Creating a New PHP Project with Remote Server Support with a Zend Server Amazon EC2 Instance

This procedure describes how to create a new PHP project with Remote Server Support, when an Amazon EC2 Instance is your remote server. A PHP remote project allows you to develop your project locally but debug it remotely on your Zend Server Amazon EC2 Instance.
Before creating a new PHP project with Remote Server Support with a Zend Server Amazon EC2 Instance as your remote server, you must add an Amazon EC2 Remote Connection Profile and enable tunneling for your Zend Server Amazon EC2 Instance.

![Clipboard icon] **To create a new PHP project with Remote Server Support with a Zend Server Amazon EC2 Instance:**

1. Go to **File | New | Other | PHP | PHP Project from Remote Server**

   - Or - In the PHP Explorer view, right-click and select **New | Other | PHP | PHP Project from Remote Server**.

   The New PHP Project with Remote Server Support dialog opens.

2. Enter the following information:

- Project name - The required project name.

- Contents - Select whether to:

  - Create a new project in the workspace - Creates a new PHP project in the workspace directory.
    By default a workspace will have been created in @user.home/Zend/workspaces/DefaultWorkspace8 when you first launched Zend Studio.

  - Create a project from existing source - Creates a PHP project pointing to files situated outside of the workspace. Click **Browse** to select the required source content.

- PHP Version - Select whether to:

  - Use default PHP settings - Uses the default PHP Interpreter settings. For more information on PHP Interpreter and PHP version compatibility see Execution Environments Preferences.

  - Use project specific settings - Select the PHP version to be used for the project. See PHP Version Support for more information.

- Project Layout - Select whether to:

  - Use project as source folder - All resources within the project will be added to the Build Path by default.

  - Create separate folders for source files and public resources - Separate folders will be created in which you can place resources which should be included or excluded from the Build Path. See Configuring a Project's PHP Build Path for more information.
    The default setting for this option can be configured from the New Project Layout Preferences page.

- JavaScript Support - Mark the 'Enable JavaScript support for this project' check-box for JavaScript functionality (e.g. JavaScript Content Assist options) to be available to the project. See Enabling JavaScript Support in PHP Projects in PHP Projects for more information.

3. Click **Next**.

The Remote Configuration dialog opens.

4. Select your Amazon EC2 Remote Connection Profile.

If you do not have an existing Amazon EC2 Remote Connection Profile you can configure one by clicking **Manage**. For more information see Adding an Amazon EC2 Remote Connection Profile.

The Project Directory is configured automatically based on your project name.

5. Make sure the Project URL is correct and click **Next**.

The Data Download Selection dialog opens.

6. Expand the Project Directory to see all the folders within it that will also be transferred.

7. Select the folders you want to download from the remote server by selecting the check-box next to the folder name.

8. Click on a selected folder to view the files included in the folder.

The individual files can also be selected/unselected to download from the remote server.

By default, all files within a folder are selected.

On this page you can also:

- Filter Types - Opens the Inclusion and Exclusion Patterns dialog. For more information see Working with Inclusion and Exclusion Patterns.

- Select All - Selects all the folders and files.

- Deselect All - Deselects all the folders and files.

10. Click **Finish**.

The selected files are downloaded and the new PHP Remote Project with Remote Server Support is created.

You can now debug your project as a PHP web page on your Amazon EC2 Instance.

The new PHP project will be created in your workspace and displayed in the PHP Explorer View.

You can now start to develop your application by creating PHP Files or adding other resources to your project.

Creating a Private Key in Zend Studio

A Private Key is necessary in order to launch and access an Amazon EC2 Instance. Private Keys supply security to ensure that no unauthorized users can access your Instance. Creating a Private Key in Zend Studio allows you to create the private and public key which allow your Amazon EC2 Instance to identify you as an authorized user.

Before launching an Amazon EC2 Instance, you must create Private Key for that Instance or apply an existing Private Key.

To create a Private Key in Zend Studio:

1. Open the EC2 AMIs view by going to **Window | Show View | Other | AWS Toolkit | EC2 AMIs**.

2. Find your AMI in the EC2 AMIs view or use criteria to search for a specific AMI in the search field.

3. From the Right Click Menu of the desired AMI select **Launch**.

The Launch Amazon EC2 Instances dialog opens.

3. Open the Right Click Menu in the Key Pair section and select **New Key Pair**.

The Create New Key Pair dialog opens.

4. Enter a new Key Pair Name and click **Browse** if you would like to change the location where your Private Key Pair is saved.
5. Click **OK** to save your new Private Key.
6. Click **Cancel** to save the changes.

Before using your Private Key you must add it in the SSH2 Preferences page. For more information see Setting Up Your Amazon EC2 Instance in Zend Studio.

You can now launch your Amazon EC2 Instance in Zend Studio.

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

Adding a New Security Group in Zend Studio

This procedure describes how to add a new Security Group for your AMI. Configuring Security Groups for your Zend Server AMI allows you to specify which connection methods can communicate with your Amazon EC2 Instance.

Having a configured Security Group is necessary to launch an Amazon EC2 Instance.

To add a new Security Group in Zend Studio:

1. Go to **Window | Show View | Other | AWS Toolkit | EC2 Security Groups**.

   The EC2 Security Groups view opens.

2. From the Right Click Menu of the view select **New Group**.

   The New Security Group dialog opens.

3. Enter a name and description for your Security Group and click **OK**.

   Your Security Group is added in the EC2 Security Groups view.

4. Select your Security Group and from the Right Click Menu in the Protocol column click **Add Permission**.

   The Add Permissions dialog opens.

5. In the dialog you must do the following:

- Mark the 'Protocol, port and network' field

- Select **TCP** from the Protocol dropdown menu.

- Enter "22" in the Port of Port Range text field.

6. Click **OK**.

Your Security Group has been configured.

For more information on Security Groups see Amazon's Getting Started Guide.

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

Launching an Amazon EC2 Instance in Zend Studio

This procedure describes how to launch your Amazon EC2 Instance in Zend Studio. The Amazon AWS Toolkit for Zend Studio allows you to launch EC2 Instances of your AMI from within Zend Studio. For information on which Zend Server AMIs are available in your region see the Zend Server on Amazon EC2 section of the Zend website.

Before launching an Amazon EC2 Instance in Zend Studio you must meet all the prerequisites and set up your Zend Server Amazon EC2 Instance.

To launch an Amazon EC2 Instance in Zend Studio:

1. Open the EC2 AMIs view by going to **Window | Show View | Other | AWS Toolkit | EC2 AMIs**.

2. Find your AMI in the EC2 AMIs view or use criteria to search for a specific AMI in the search field.

3. From the Right Click Menu of the desired AMI click **Launch**.

   The Launch Amazon EC2 Instances dialog opens.

4. The dialog is made up of the following components:
- Number of Hosts
- Instance Type - The size of the Instance you have selected.
- Availability Zone - Displays your current region.
- Key Pair - Select an existing Key Pair, or to create a new Key Pair see Creating a Private Key in Zend Studio.
- Security Group - Select a Security Group. For more information see Amazon's Getting Started Guide or Adding a New Security Group in Zend Studio.
- User Data
5. Click **Finish**.

6. Open the EC2 Instances view by going to **Window | Show View | Other | AWS Toolkit | EC2 Instances**.

   The EC2 Instances view opens.
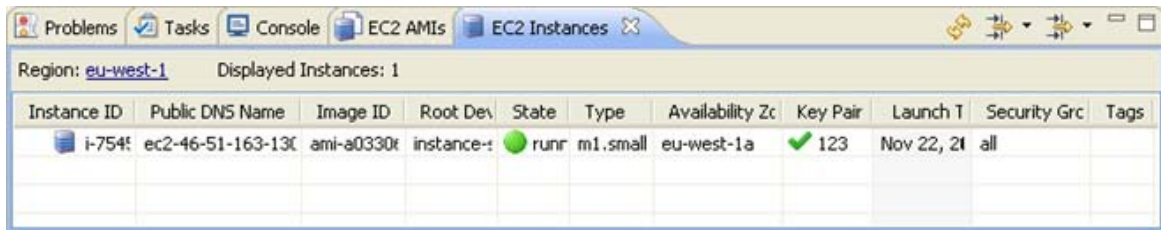
7. From the Right Click Menu of your running EC2 Instance click **Copy Public DNS Name**.

8. Open a browser and paste the Public DNS Name with "http://" added in front of it, and ":10081" added to the end.

   For example, if your Public DNS Name is "ec2-79-125-63-250.eu-west-1.compute.amazonaws.com", the full URL will be "http://ec2-79-125-63-250.eu-west-1.compute.amazonaws.com:10081".

   Your Zend Server Amazon EC2 Instance opens.

9. If this is your first time launching the Zend Server Amazon EC2 Instance you will have to do the following:

   a. Agree to the Terms of Use.

   b. Set your password.

   c. Enter your Zend Server license details.

      These were sent to you in an invoice email when you purchased and activated your Zend Server Amazon EC2 Instance. For more information see Prerequisites.

   d. Set your environment as an allowed host on your Zend Server Amazon EC2 Instance.

For more information on first time registration of Zend Server see Registration in the Zend Server Online Documentation.

You can now debug as a PHP web page on an Amazon EC2 Instance. If you would like to no longer use your Instance you can terminate it in Zend Studio.

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

Terminating an Amazon EC2 Instance in Zend Studio

This procedure describes how to terminate your Amazon EC2 Instance. Once an Instance is terminated it can not be accessed again.

Before terminating an Instance you must launch an Amazon EC2 Instance.

To terminate an Amazon EC2 Instance in Zend Studio:

1. Open the EC2 Instances view by going to **Window | Show View | Other | AWS Toolkit | EC2 Instances**.

   The EC2 Instances view opens.

2. From the Right Click Menu of your running EC2 Instance click **Terminate Instance**.

   The Terminate Selected Instances dialog opens.

3. Click **OK** to terminate the Instance.

   Your Instance has been terminated.

Once an Instance is terminated, it cannot be accessed again. To launch another Instance see Launching an Amazon EC2 Instance in Zend Studio.

Debugging as a PHP Web Page on Your Amazon EC2 Instance

Debugging as a PHP web page on your Zend Server Amazon EC2 Instance allows you to debug your PHP application on a Zend Server Instance. This will allow you to use the functionalities of Zend Server to find problems in your code and fix them more efficiently. This is especially useful for the staging phase of development as it most similarly resembles the environment your PHP application will run on. For more information on Zend Server and its functionalities see the Zend Server Online Documentation.
Before debugging on your Zend Server Amazon EC2 Instance you must enable tunneling for your Zend Server Amazon EC2 Instance.

To debug as a PHP web page on your Zend Server Amazon EC2 Instance:

1. Click the arrow next to the debug button ![debug button] on the toolbar and select Debug Configurations -OR- go to **Run | Debug Configurations**.
   The Debug Configurations page opens.
2. Double-click the PHP Web Page option to create a new debug configuration.
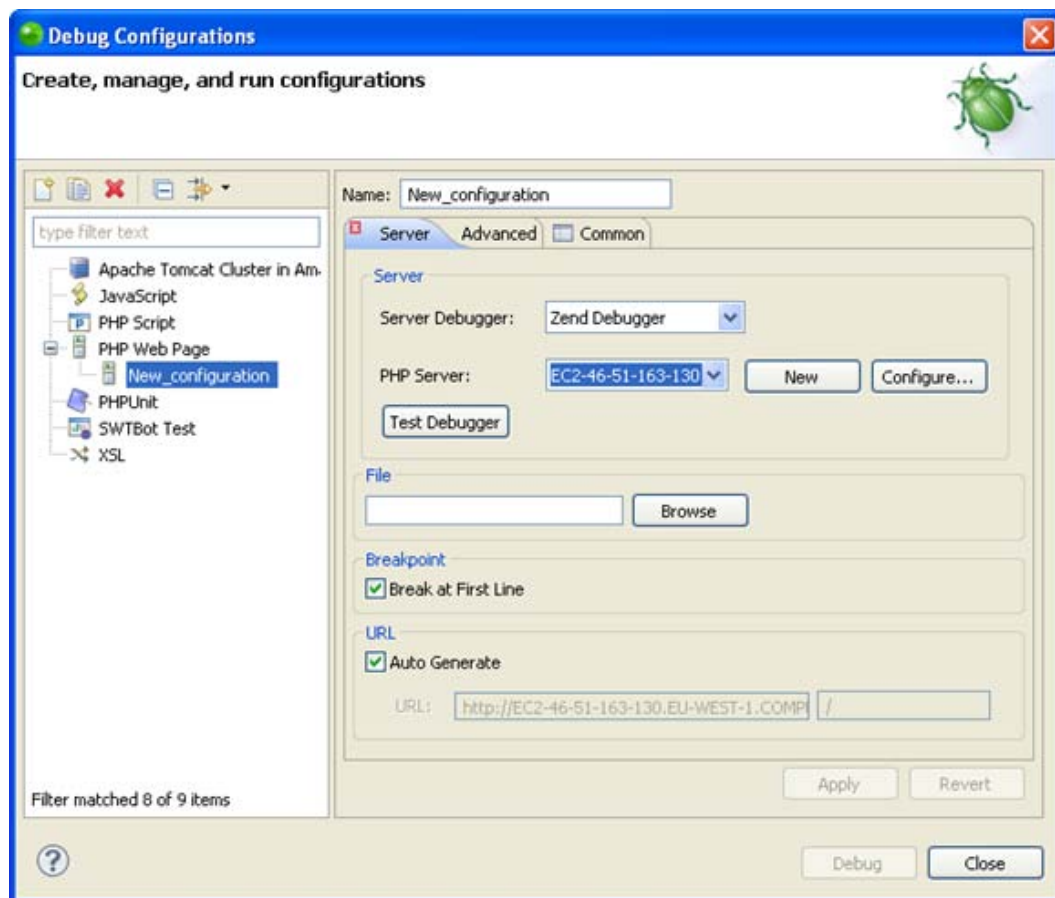
3. Specify the following information in the dialog:

▪ Enter a name for the new configuration.

▪ Select the Server Debugger to be used (by default this will be the Zend Debugger).

▪ Select your Zend Server Amazon EC2 Instance from the PHP Server list.

4. In the File section click **Browse** and select your debug target file (the file from which the debugging process will start.)

5. Select whether the Debugger should stop at the first line of code by marking/unmarking the 'Break at First Line' check-box.

6. The URL to be debugged will have been automatically created based on the file name and your server address. If the URL does not point to your debug target location, unmark the Auto Generate check-box and modify the URL.

7. To check whether your server connection is correctly configured, and that Zend Studio can communicate with your server debugger, click **Test Debugger**.

8. Click **Apply** and then **Debug**.

9. Click **Yes** if asked whether to open the PHP Debug Perspective.

See Running and Analyzing Debugger Results for more information on the outcome of a debugging process.

Working with Deployment

Zend Studio enables you to develop, test, debug, launch and deploy your application within the same IDE making the process of application development a whole lot easier and faster.

Working with deployment includes the following topics:

- [Enabling Deployment Support for Your Application](#)
- [Editing the Deployment XML](#)
- [Launching an Application](#)
- [Using the Deployment Actions](#)
- [Managing Targets](#)

For more information on the general concept of deployment and its benefits, see Deployment.

Enabling Deployment Support for Your Application

This procedure describes how to enable Zend Deployment Support. Enabling Zend Deployment Support creates the deployment.xml file that allows you to manage and define the deployment of your application.

Zend Studio gives users two options to enable Deployment Support:

- Enabling Deployment Support while creating a project.
- Enabling Deployment Support for an existing project.

Enabling Deployment Support while Creating a Project

To enable Deployment Support while creating a project:

1. Open a New Project Wizard.
2. In the **Launch Settings** dialog, select **Launch with Zend Application Deployment Support**.

After completing the remainder of the Wizard, your new project will appear in the PHP Explorer View with the deployment.xml file. This file holds all the deployment settings and definitions.

For more information on how to work with the deployment.xml file, see Creating the Deployment XML.

For more information on how to deploy the application, see Deploying an Application.

Enabling Deployment for an Existing Project

To enable Deployment Support for an existing project:

1. In the PHP Explorer, select the project you wish to enable deployment for.
2. Right-click and select **Configure | Add Application Deployment Support.**

Your new project will appear in the PHP Explorer View with the deployment.xml file. This file holds all the deployment settings and definitions.

For more information on how to work with the deployment.xml file, see Creating the Deployment XML.

For more information on how to deploy the application, see Deploying an Application.
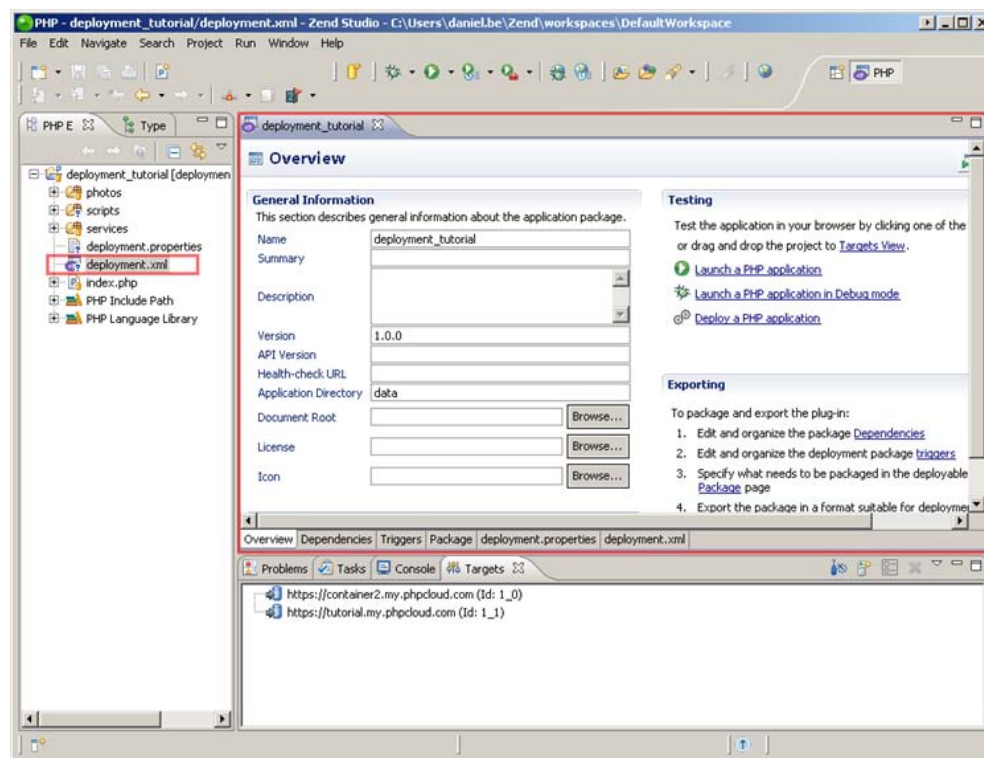
Editing the Deployment XML

After enabling deployment on your application, Zend Studio creates a deployment.xml file that is displayed in the PHP Explorer view. This file is used to customize and define your deployment and to package the application.

Prerequisites: A project with enabled deployment. For more information on how to enable deployment for a project, see Enabling Deployment Support for Your Application.

To edit your deployment.xml file:

Click the deployment.xml file in the PHP Explorer. The file is displayed in the main view in your workspace.
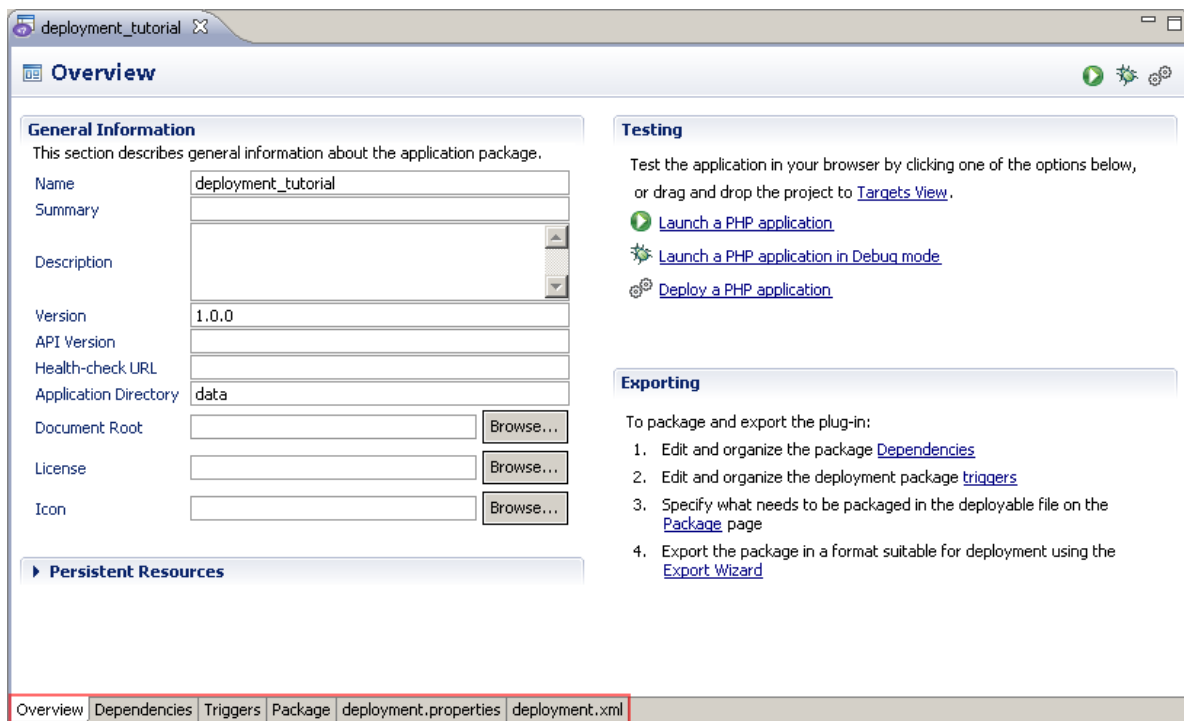
Components

The deployment.xml file is composed of the following components that are displayed on tabs:

- Overview tab
- Dependencies tab
- Triggers tab
- Package tab
- deployment.properties tab
- deployment.xml tab

Overview Tab

The Overview tab is the first tab in the deployment.xml file, and is displayed automatically when opening the file.
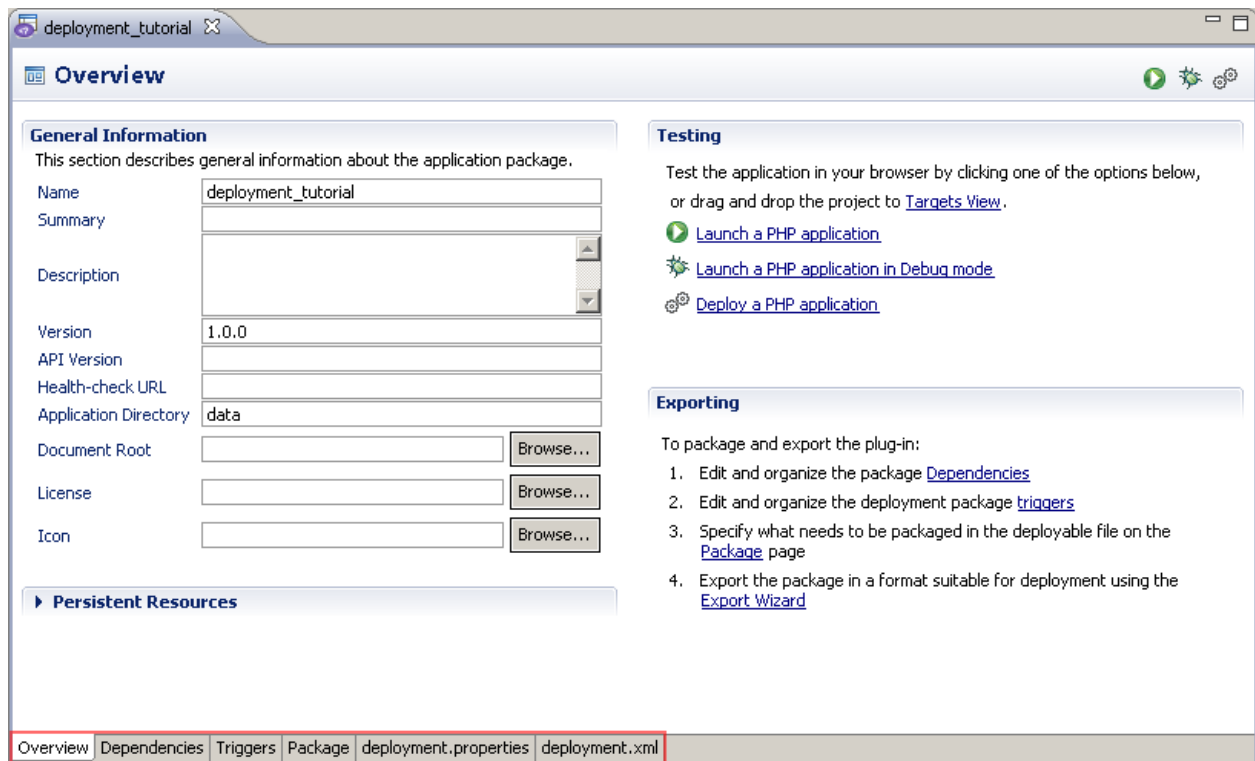
Tab Areas

The overview tab includes the following areas:

- General Information
- Persistent Resources
- Testing
- Exporting

General Information

In this area, enter general information on the project:

- Name: Enter the name of the project.
- Summary: Enter a short description of the application.
- Description: Enter a description of the project.
- Version: Enter version number for the application.
- API Version: Enter an API version number for future use.

- Health-check URL: Add the relative URL you would like Zend Studio to check against for Application Health Check. This checks if the URL returns a 200 (OK) response or an error response.
- Application Directory:
- Document Root: Enter the location root of the project.
- License: Enter which license to display in the browser.
- Icon: The icon of the application.

Persistent Information

This area lists persistent resources to be kept during upgrade, relative to the package root.

To add a resource:

1. Click **Add.**

   The Add Path dialog opens.
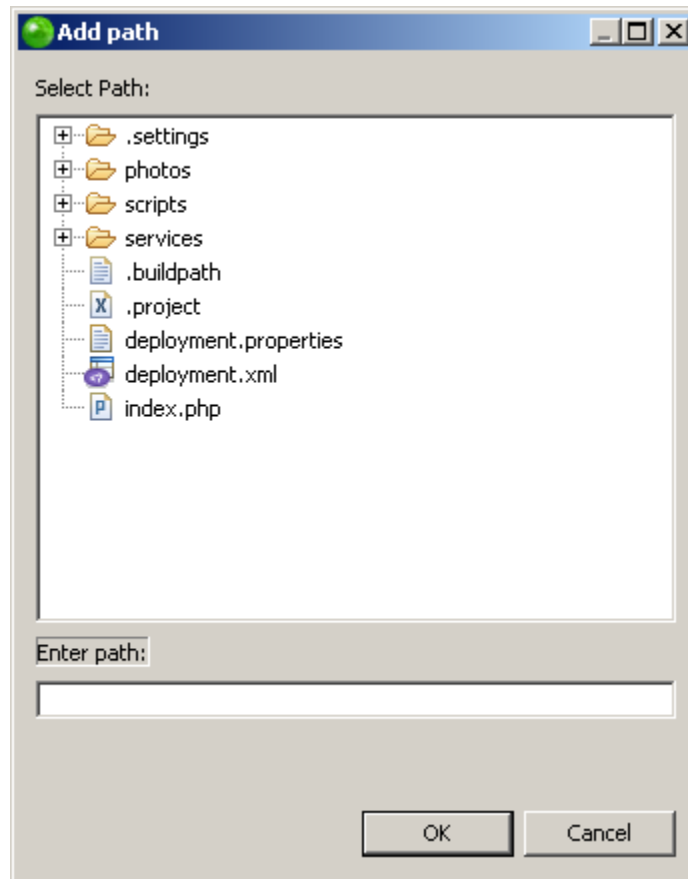
2. Select the resource and click **OK**.

   The resource is added the list.

You can edit and remove resources by clicking on the **Edit** and **Remove** commands.

Testing

In this area, you can launch and deploy your application.

- Click **Launch a PHP Application** to run your application.

  Your application deploys and launches in a browser.

  For more information, see <u>Launching an Application</u>.

- Click **Launch a PHP Application in Debug mode** to launch your application in the debugging mode.

  Your application deploys and launches in a browser.

  For more information, see <u>Launching an Application in Debug Mode</u>.

- Click **Deploy a PHP Application** to deploy.

  Your application deploys.

  For more information, see <u>Deploying an Application</u>.

Exporting

In this area, you can package and export the application.

For more information on how to package and export an application, see <u>Exporting the Application Package</u>.

Dependencies Tab

The Dependencies tab is the second tab in the deployment.xml file and contains dependencies which need to be satisfied in order for the application to function correctly. For example, PHP version and PHP directive values.
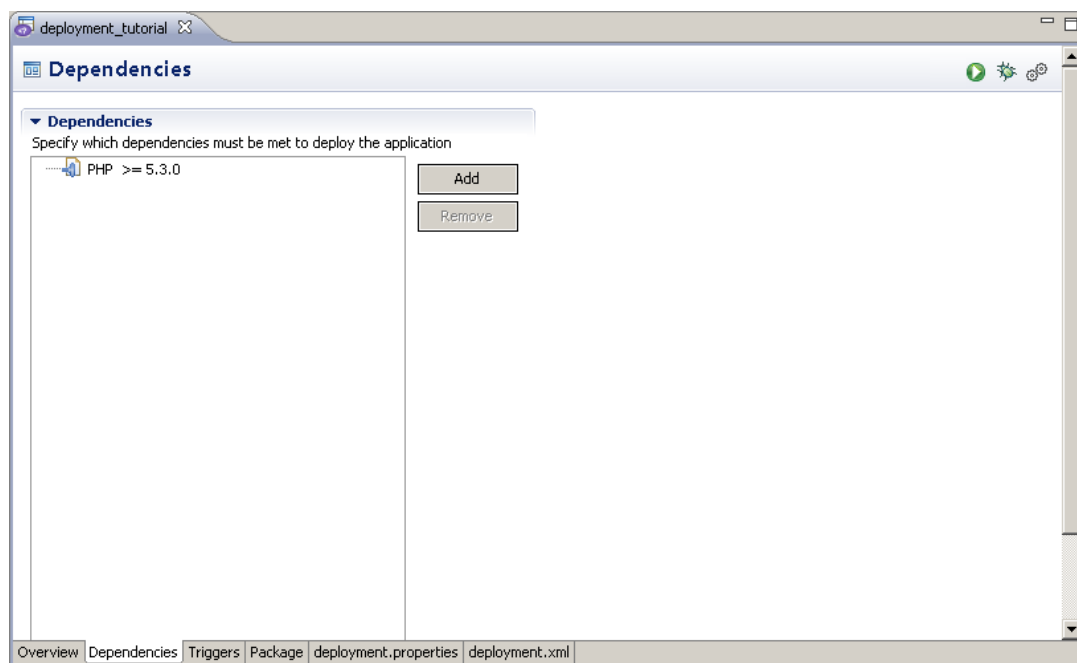
Adding Dependencies

To add a dependency:

1. Click **Add**.

    The Add Dependency dialog is displayed.
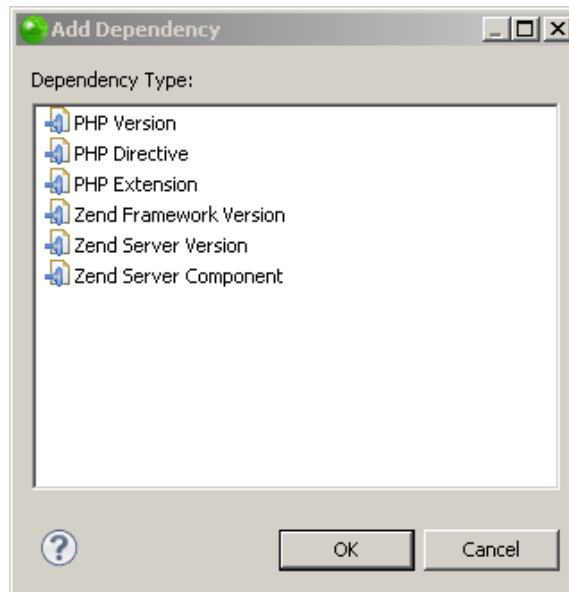
2. Select a dependency to add to the deployment.xml, and click **OK**.

    The Add New Dependency dialog is displayed.
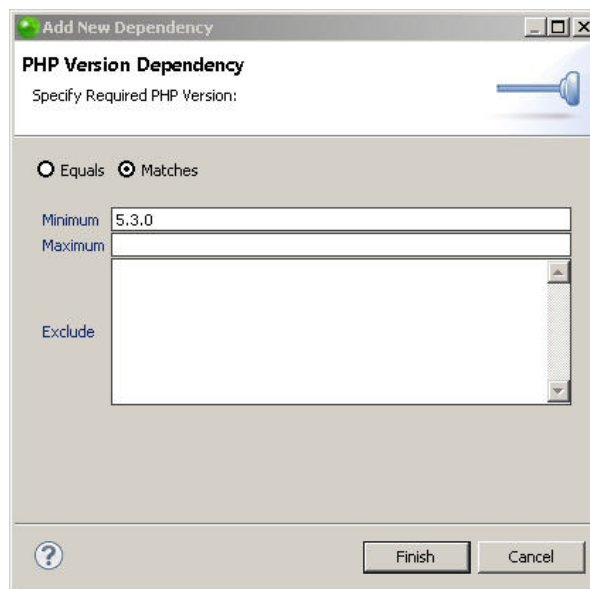
3. Define the dependency.

For example: PHP Version Dependency- Select **Matches** and enter 5.3.0 to command

Zend Studio to check that the PHP Version is at least 5.3.0.

4. Click **Finish**.

The dependency is displayed in the dependency list.

Removing the dependencies is easy. Just select the dependency and click **Remove** .

Triggers Tab

The Triggers tab is the third tab in the deployment.xml file, and allows you to define user parameters to be used in selected deployment scripts.

You may use scripts and parameters to automate the application's installation process across any defined targets.
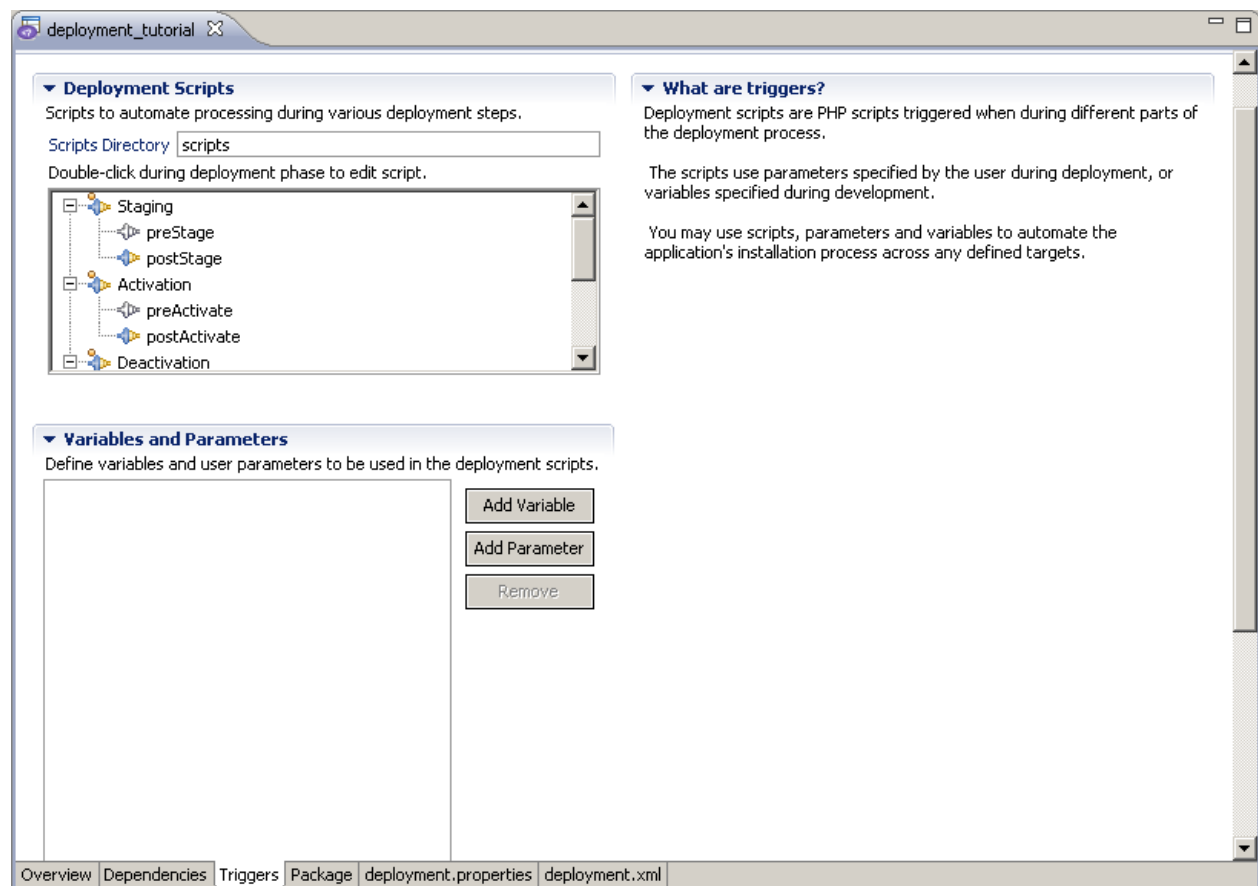
Tab Areas

The tab consists of two areas:

- Deployment Scripts
- Parameters

Deployment Scripts

Deployment scripts are PHP scripts triggered when during different parts of the deployment process. The scripts use parameters specified during development.

For more information on the trigger types, see the [Zend Server Help](#).

To add a deployment script:
1. Double-click the trigger you wish to add.
2. When prompted, click **Yes** to confirm.

   The script file is added to the Scripts folder in your project and is displayed in the PHP Explorer.

Variables and Parameters

You can define parameters that will be used by the deployment scripts. These parameters will later be presented during the deployment process.

To add a parameter:
1. Click **Add Parameter**.

   The Parameter Details area is displayed in the tab.

2. Enter the following information:
   - ID: Enter a name for the parameter.
   - Type:Define the type of the parameter. For example: Password.
   - This Parameter is Required: Mark this check-box to define it as mandatory for deployment.

- Display Text:Enter text to be displayed.
- Default Value:Enter text to be displayed by default.
- This Parameter is read-only:Mark this check-box to define parameter as read-only.
- Identical: Select a parameter to be duplicated and displayed for confirmation. For example: Confirm password.
- Description: Enter the parameter description.
3. The parameter is updated automatically and will appear in the Variables and Parameters area.

The defined parameters will be activated during the deployment process as prescribed by the deployment scripts. You can remove parameters by clicking **Remove.**

Package Tab

The Package tab is the fourth tab in the deployment.xml file, and allows you to define the contents of your application package by including the project components that you wish to package and excluding the non-essential components.

To add content to your application package, mark the relevant files. Expand the folders to view the full listing of files.
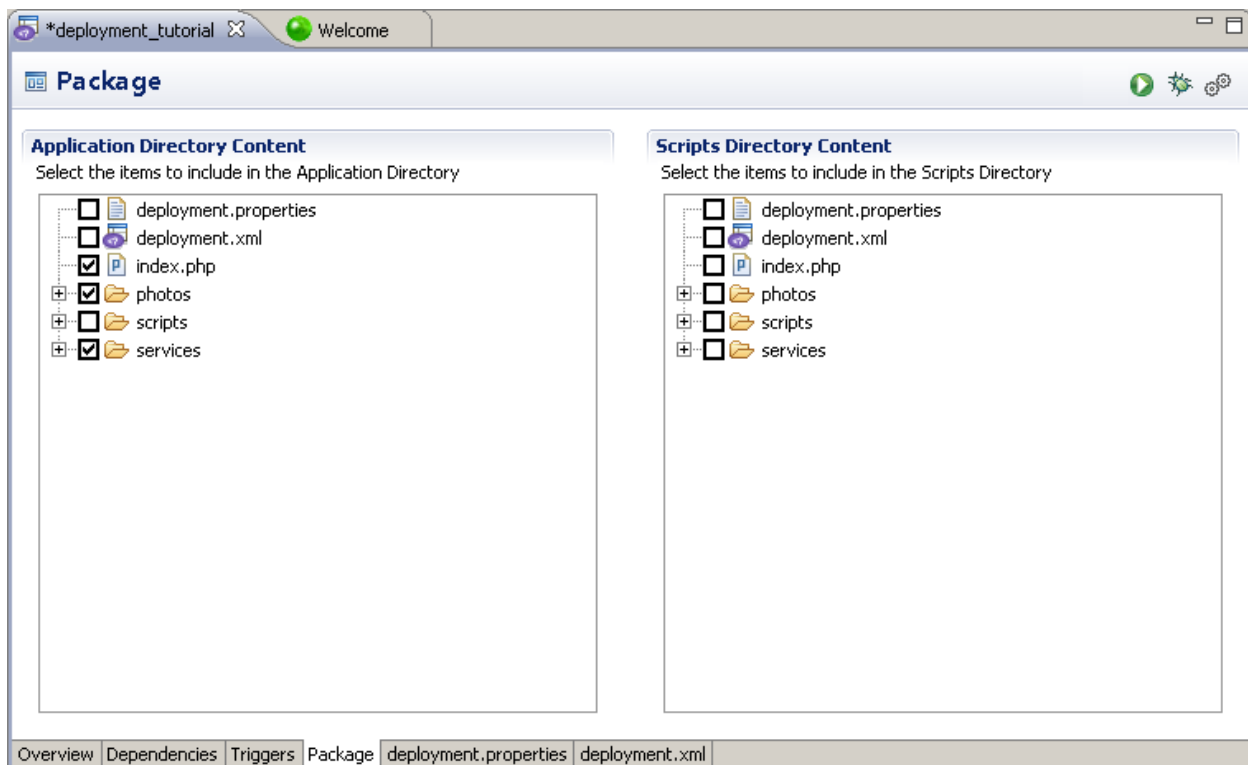
deployment.properties Tab

This tab holds internal paths for the deployment features and reflects the settings defined in the package tab.

Example:

appdir.includes = index.php,\

photos,\

services

scriptsdir.includes = scripts/post_activate.php,\

scripts/post_stage.php

deployment.xml Tab

The deployment.xml tab is the XML source file and includes the definitions you have inserted in the previous tabs.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<package xmlns="http://www.zend.com/server/deployment-descriptor/1.0" version="1.0">
   <name>deployment_tutorial</name>
   <version>
      <release>1.0.0</release>
   </version>
   <appdir>data</appdir>
   <scriptsdir>scripts</scriptsdir>
   <dependencies>
   <required>
    <php>
```

```
        <min>5.3.0</min>
      </php>
    </required>
  </dependencies>
   <parameters>
   <parameter id="Parameter" readonly="false" required="false" type="string"/>
   <parameter display="sdsd" id="sParameter" readonly="false" required="false" type="string">
     <defaultvalue>sdsd</defaultvalue>
   </parameter>
   <parameter id="Parameter" readonly="false" required="false" type="string"/>
  </parameters>
   </package>
```

Launching Applications

This procedure describes how to run your application in Zend Studio. Launching will display your application in a Web browser. If it is the first time you are launching, deployment will also be performed.

Prerequisites: A project with enabled deployment. For more information on enabling deployment, see Enabling Deployment Support for Your Application.

To launch your application:

1. Select the project you wish to launch.
2. Open the deployment.xml file and select the Overview tab.
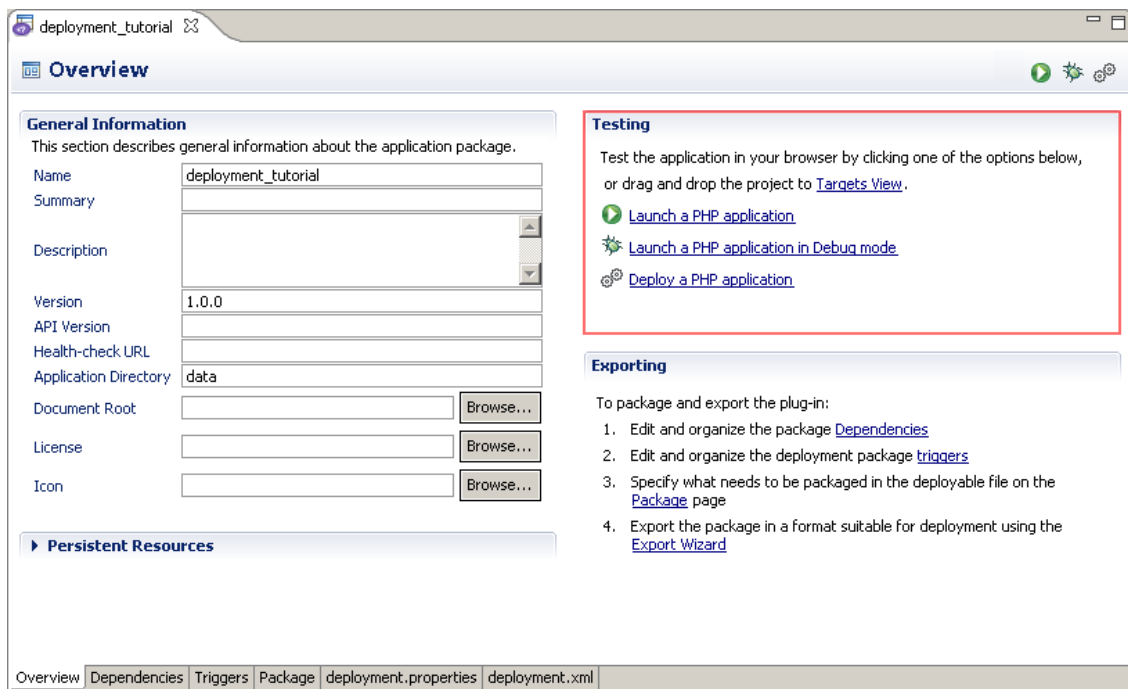
3. In the Testing area, click **Launch a PHP Application**.

    The Deploy PHP Application dialog is displayed.

    For more information on Deployment, see [Deploying an Application](#).
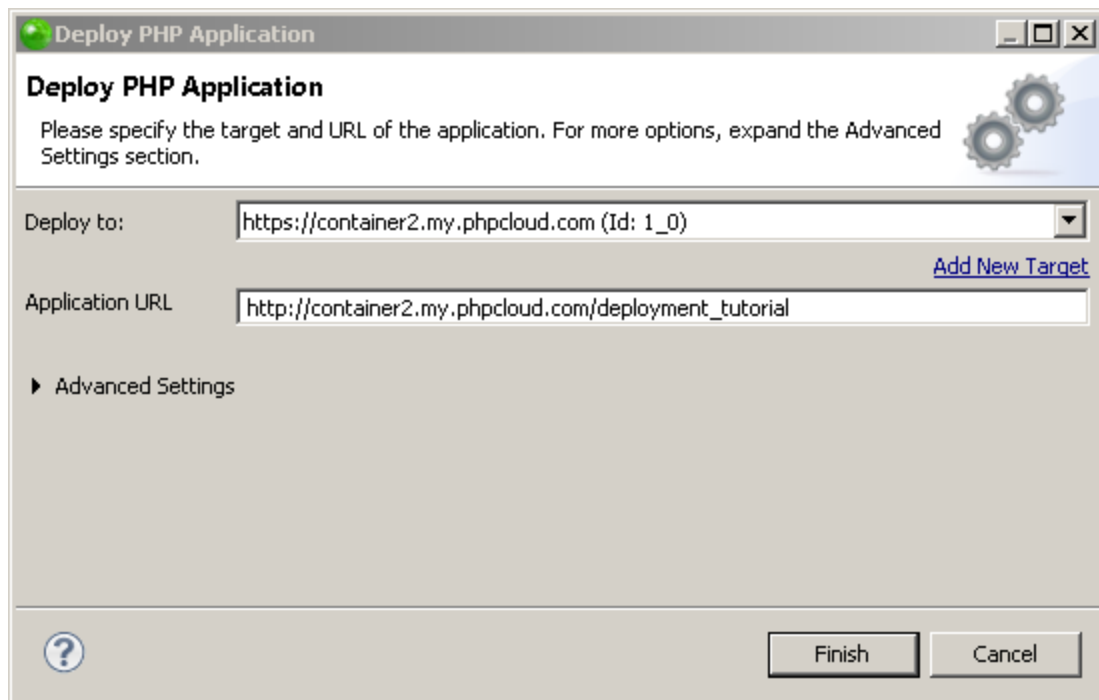
4. Click **Finish**.

    Your application will deploy and launch in a browser.

> **Note:**
> If this is not the first time you are launching, no deployment dialog is displayed and the application will launch in a browser.

You may also launch in Debug Mode. See [Debugging an Application](#) for more information.

Launching an Application in Debug Mode

Zend Studio gives you the option to run your application and launch in a debugging mode.

To launch in Debug mode:

1. Select the project you wish to launch in Debug mode.

2. Open the deployment.xml file and select the Overview tab.
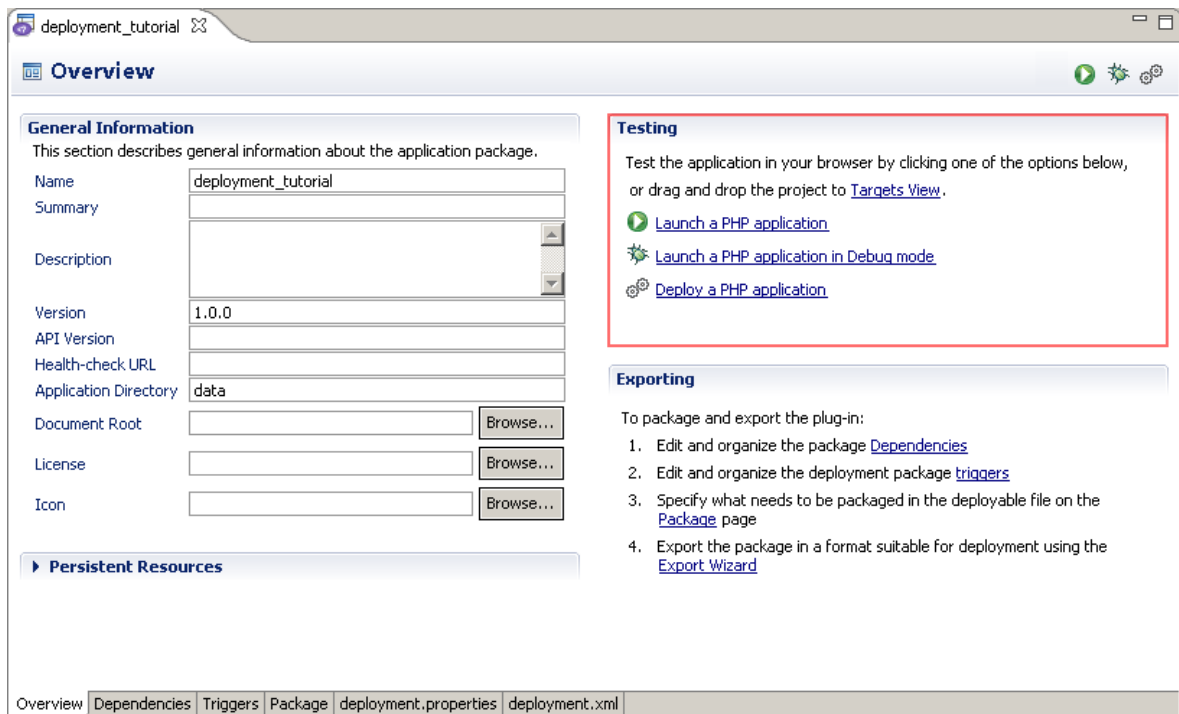
3. In the Testing area, click **Launch a PHP Application in Debug mode**.

   You will be prompted to confirm switching perspectives to Debug perspective.
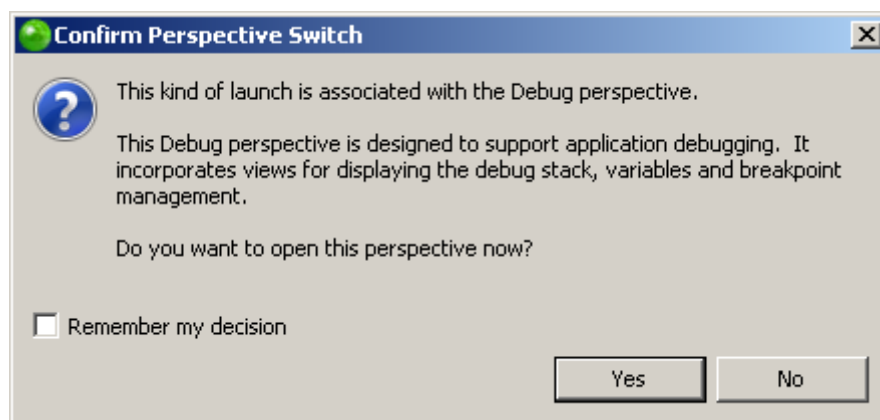
4. Click **Yes**.

Your application launches in Debug perspective and stops at the first breakpoint.
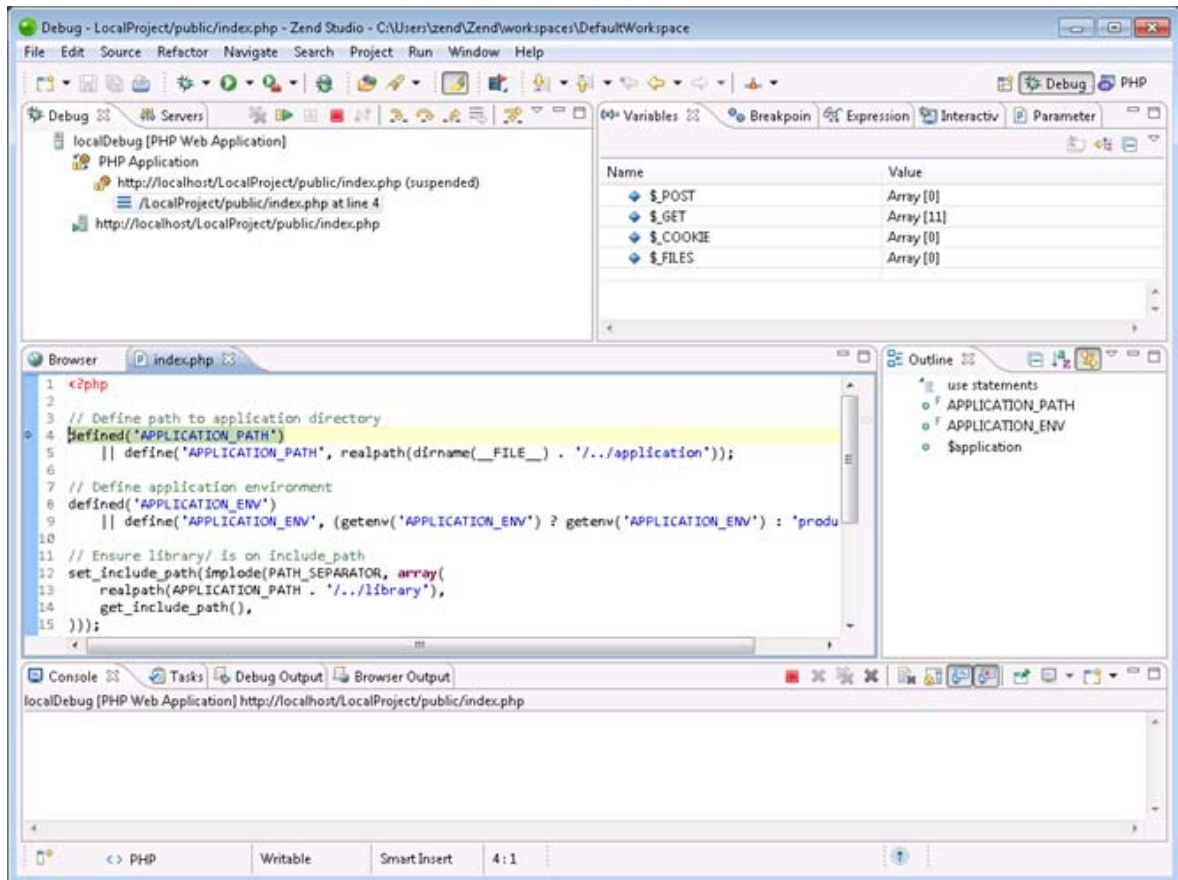
To continue the debugging, click .

Using the Deployment Actions

Deployment in Zend Studio includes the following actions:

- [Exporting the Application Package](#)

- [Deploying an Application](#)

- [Updating an Application](#)

- [Removing an Application](#)

For more information, see [Deployment](#).

Exporting the Application Package

After developing your application, Zend Studio gives you the option to create a ZPK file that holds the application.
Once this package is created, you can easily export the file to other members of the work process making your application portable and accessible.

To create and export the ZPK file:
1. Click the **Overview** tab in the deployment.xml file.
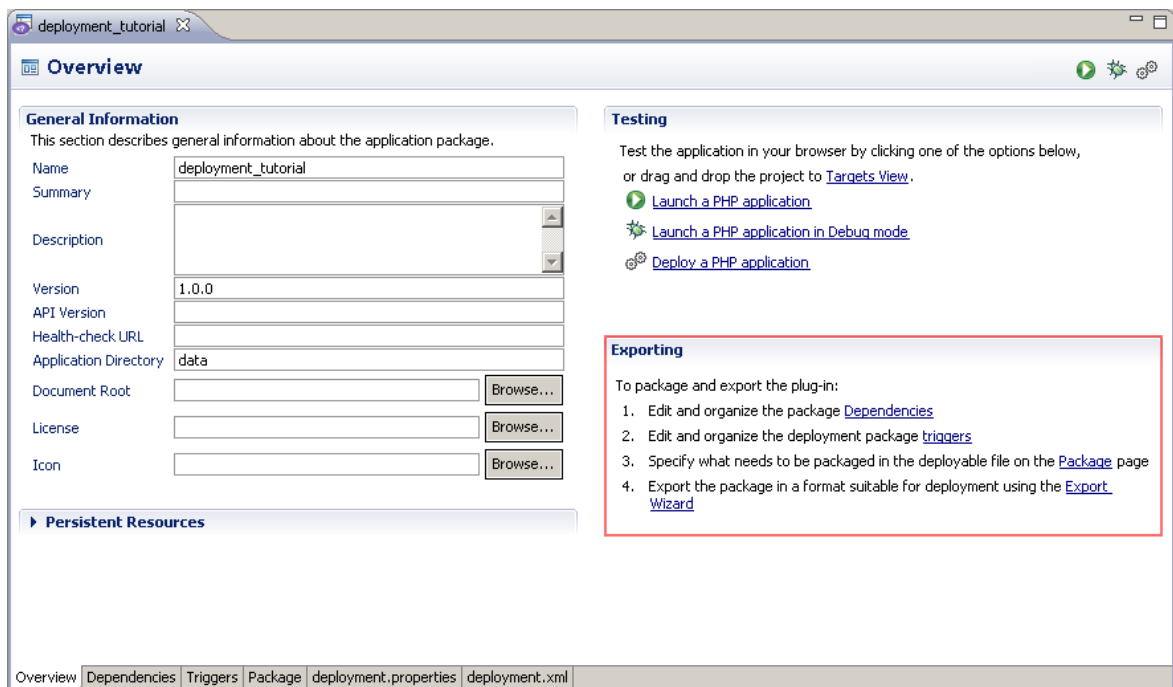
2.   In the Exporting area, click **Export Wizard**.

The Export Deployment Package dialog is displayed.
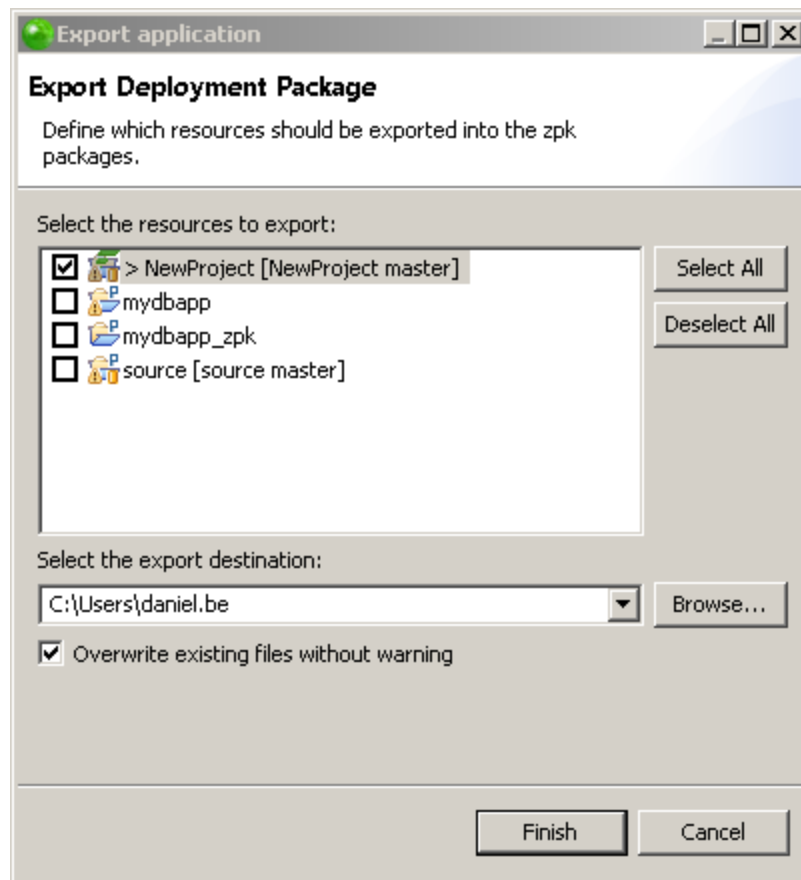
3.   Select the project to be packaged.
4.   Select the export destination by clicking **Browse**.
5.   Click **Finish**.

Zend Studio creates the ZPK package and exports it to the designated destination folder.

You can now send the application package to other members of the work process.

Deploying an Application

This procedure describes how to deploy your application. Deploying takes the application from the local development environment and enables external view and usage on Web browsers.

For more information on what deployment means, see <u>Deployment</u>.

Prerequisites: To deploy an application in Zend Studio, a project needs enabled deployment. For more information on how to enable deployment for a project, see <u>Enabling Deployment Support for Your Application</u>.

To deploy an application:

1. Select the project you wish to deploy.

2. Open the deployment.xml file and select the Overview tab.
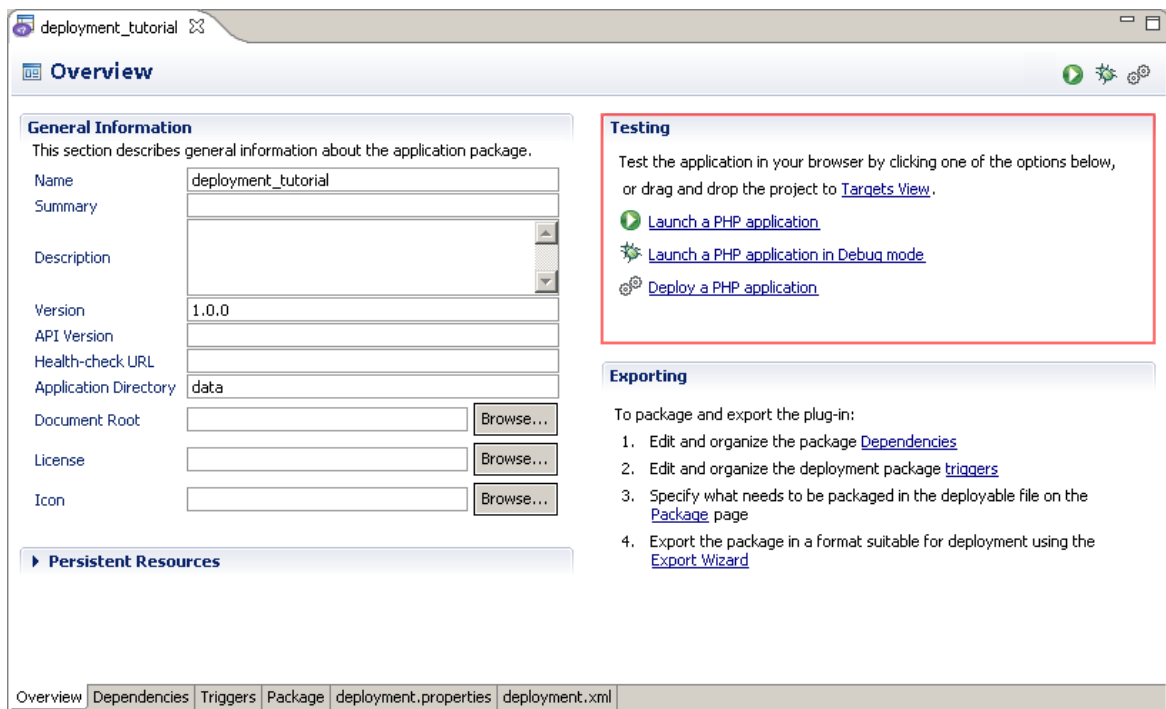
3. In the Testing area, click **Deploy a PHP Application**.

    -OR-

    Drag the project from the PHP Explorer onto the target in the Target view and select
    Deploy a PHP Application.

    The Deploy PHP Application dialog is displayed.
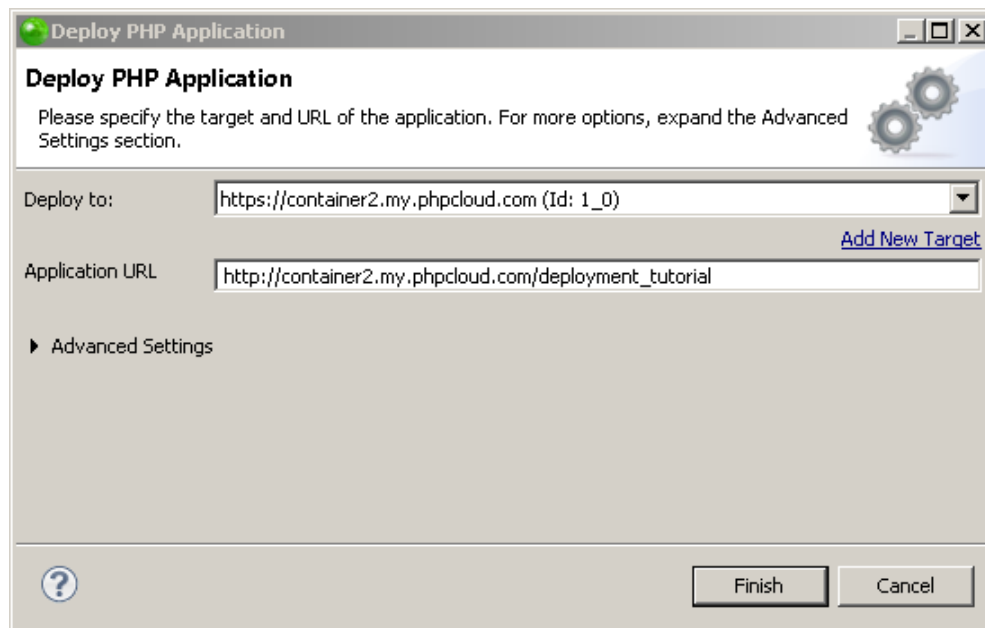
4. Enter the following information:

▪   Deploy to: Select the target you wish to deploy the application on. If you wish to add a
    new target, click on **Add New Target**.

    For more information on adding new targets, see [Managing Targets](#).

▪   Application URL: Enter the URL for your application.

▪   Click **Advanced Settings** for additional settings:

    •   By default, **Deploy** is selected.

    •   To take no action and open a browser, mark **No Action**.

    •   To update an existing application, mark **Update** and select application to update
        by clicking on dropdown menu.

    •   To automatically update an applcaition, mark **Automatic Deploy** and select
        application clicking on dropdown menu.

    •   To ignore deployment configuration failures, mark **Ignore Failures**.

5. Click **Finish**.

    Your application is deployed on the target selected.

Deployment also occurs the first time applications are launched. For more information on launching applications, see Launching an Application.

Updating an Application

This procedure describes how to update your application. Any changes that are applied to your project will be updated via the target.

Prerequisites: A project with enabled deployment. For more information on enabling deployment, see Enabling Deployment Support for Your Application.

To update your application:

1. Select the project you wish to update.
2. Open the deployment.xml file and select the Overview tab.

3. In the Testing area, click **Deploy a PHP Application**.

-OR-

Drag the project from the PHP Explorer onto the target in the Target view and select **Deploy a PHP Application**.

The Deploy PHP Application dialog is displayed.

4. Click on **Advanced Settings**.

   The Advanced Settings area is displayed.

4. Mark **Update** and click the dropdown menu to select the application you wish to update.

5. Click **Finish**.

   Your application will be deployed and updated.

Removing an Application

This procedure describes how to remove an application. Removing an application will detach the application from the target defined and can be reattached at a later stage.

To remove an application:

1. In the Target view, expand the relevant target to view the application.
2. To remove the application from the target, right-click and select **Uninstall**.
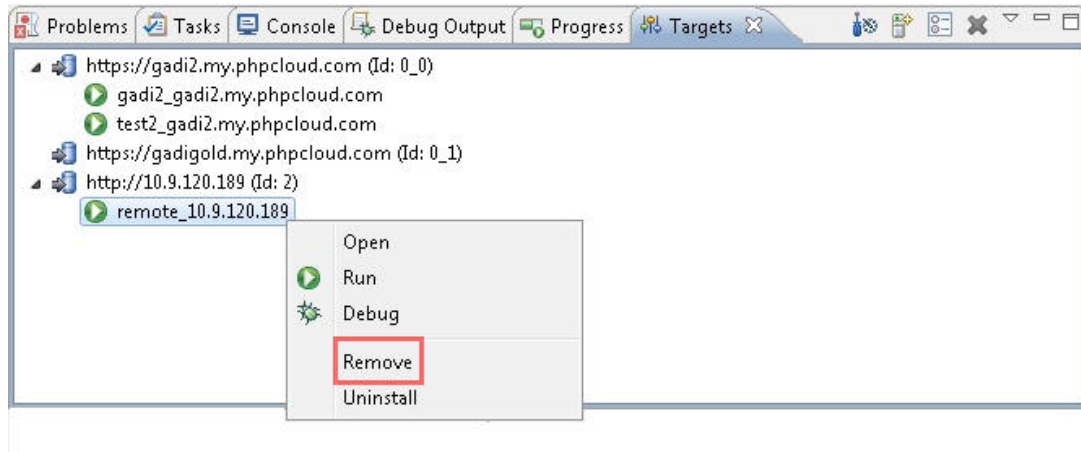
Right-clicking and selecting **Remove** will only remove the application from the Target view.

Deleting an application from the PHP Explorer, will remove the application from the workspace.

Managing Targets

Zend Studio allows you to easily access and manage your targets for deployment.

The targets define the way that your application is deployed.

Viewing Targets

To view your targets, select **Window | Show View | Targets** from the Menu bar.

The Targets view is added to your View tabs at the bottom of your screen.
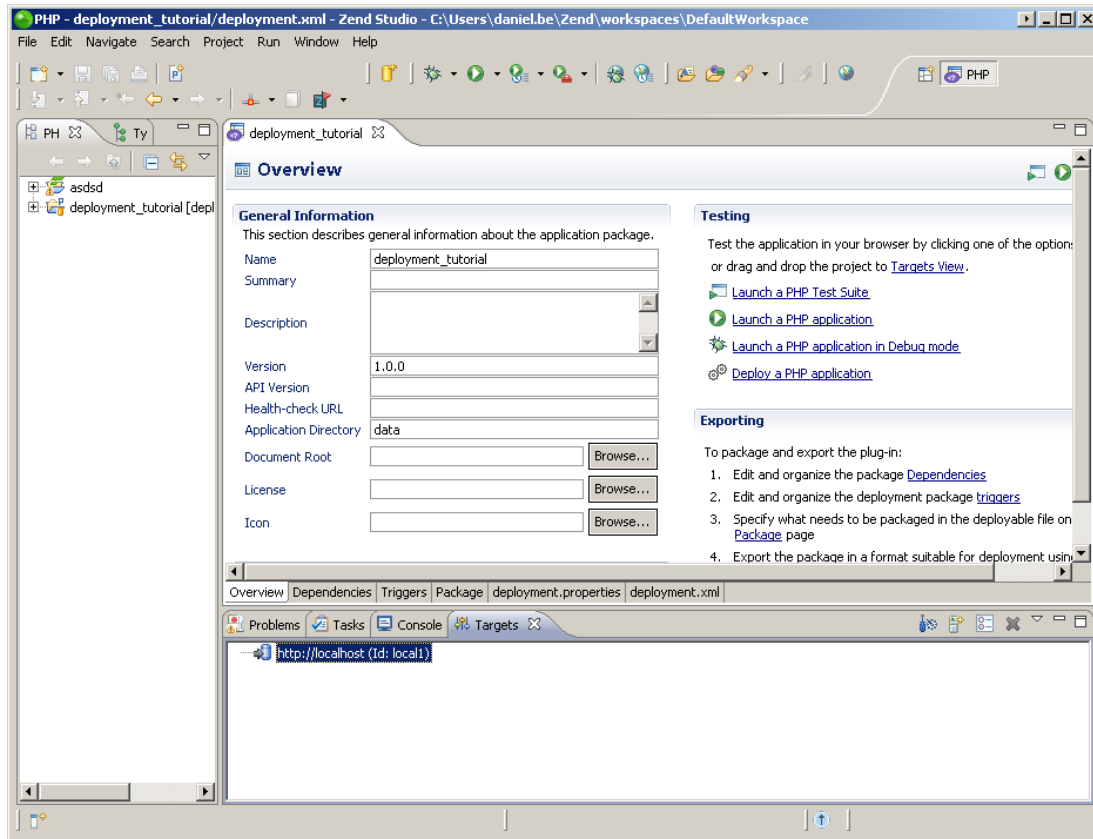
Adding Targets

In the Target view, you can add a new target.

To add a target:

1. In the Target view, click on the **Add Target** icon 　.
   The Add Target dialog is displayed.

2. Select which target you wish to create by clicking on the icons in the Create a New Target window:

- Creating a Local Zend Server Target
- Creating a Remote Zend Server Target
- Creating a Zend Developer Cloud Target

Once your new target is added, you can deploy a project by dragging an existing project from the PHP Explorer onto the new target. For more information on deploying, see Deploying an Application.

Editing Targets

To change the settings of a target, select the target in the Target view and click the Edit Target icon 　.

Detecting Targets

To detect an existing target, click the Detect Target icon 　.

Removing Targets

To remove targets, right-click the target you wish to remove and select **Remove**.

Note:

Make sure you have selected the right target to remove, there is no warning dialog before the target is removed.

Creating a Local Zend Server Target

Creating a Local Zend Server target allows you to deploy your application on your local server.
Prerequisite: Zend Server and Zend Studio installed on the same machine.

To create a Local Zend Server target:

1. From the Menu bar, select **Window | Show View | Targets** to add the Target view to your workspace.

2. In the Target view, click on the **Add Target** icon .

The Add Target dialog is displayed.

3. Click on the **Detect Local** | **Next**.

The Target Details dialog is displayed.
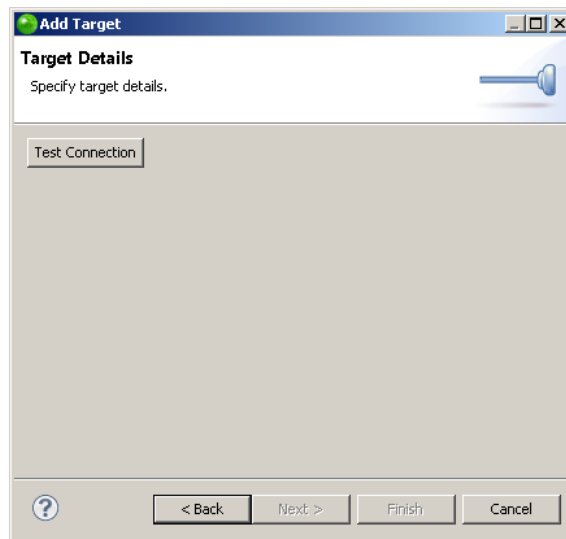
4. Click **Test Connection**.

5. If no problems arise, click **Finish**.

The new local target appears in the Target view.

You can now deploy a project on the Zend Developer Cloud by dragging an existing project from the PHP Explorer onto the new target. For more information on deploying, see Deploying an Application.

Creating a Remote Zend Server Target

Creating a Remote Zend Server target allows you to deploy your application on a remote server.

To create a Remote Zend Server target:

1. From the Menu bar, select **Window | Show View | Targets** to add the Target view to your workspace.

2. In the Target view, click on the **Add Target** icon .

The Add Target dialog is displayed.

3. Click on the **Zend Server** icon.

The Target Details dialog is displayed.
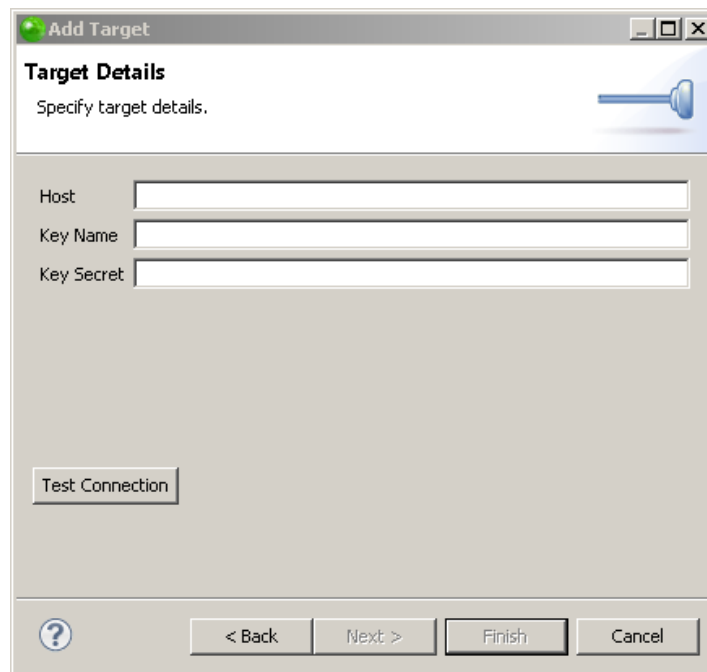
4. Enter the following information:

- Host: Address of the host server.

- Key Name: Enter name of API key.

- Key Secret: Enter API key.

   For more information on how to get API key, see <u>Adding API Keys</u>.

5. Click **Test Connection**.

6. If no problems arise, click **Finish**.

   The new Remote Zend Server target appears in the Target view.

You can now deploy a project on the Zend Developer Cloud by dragging an existing project from the PHP Explorer onto the new target. For more information on deploying, see <u>Deploying an Application</u>.

Creating a Zend Developer Cloud Target

Creating a Zend Developer Cloud target allows you to deploy your application on the Cloud.

For more information on deploying applications on the Zend Developer Cloud, see

http://www.phpcloud.com.

Prerequisite: To create a new Zend Developer Cloud target, you need a Zend Developer Cloud account and container.

To create a Zend Developer Cloud target:

1. From the Menu bar, select **Window | Show View | Targets** to add the Target view to your workspace.

2. In the Target view, click on the **Add Target** icon .

    The Add Target dialog is displayed.
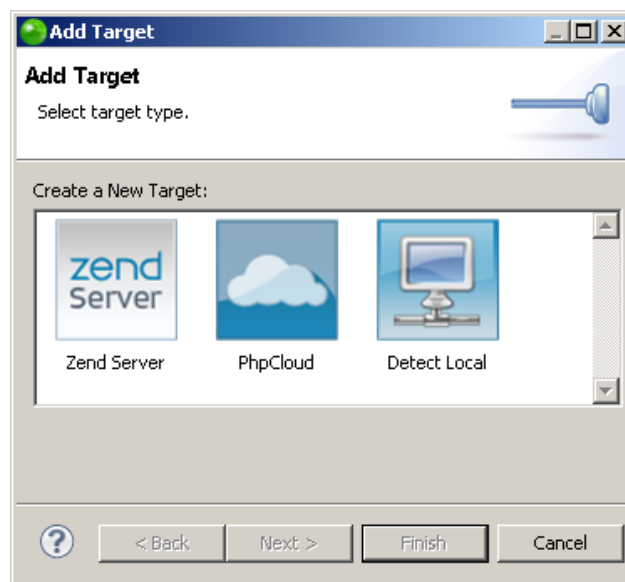
3. Click on the **PhpCloud** icon.

The Target Details dialog is displayed.

4. Enter the following information:
- Username: Your Zend Account username.
- Password: Your Zend Account Password.
- SSH Private Key: Your Zend Develoepr Cloud container SSH Keypair.

5. Click **Test Connection**.

6. If no problems arise, click **Finish**.

The new Zend Developer Cloud target appears in the Target view.

You can now deploy a project on the Zend Developer Cloud by dragging an existing project from the PHP Explorer onto the new target. For more information on deploying, see Deploying an Application.

Reference

[PHP Perspectives and Views](#)	[PHP Project Properties](#)
[JavaScript Debug Perspective](#)	[PHP Icons](#)
[Documentation View](#)	[Keymap](#)
[Targets View](#)	[Useful Links](#)
[Web Browser Tools Perspective](#)	[Contribute to the Documentation](#)
[PHP Perspective Menus](#)	[Support](#)
[PHP Perspective Main Toolbar](#)	[Registering Your License](#)
[PHP Preferences](#)	

PHP Perspectives and Views

Zend Studio incorporates a number of Perspectives and Views for managing all aspects of your PHP code, files, project and application creation.

The following PHP perspectives are used for developing PHP:

- PHP Perspective
- PHP Debug Perspective
- PHP Profile Perspective
- Code Tracing Perspective

See the Workbench User Guide for more on Perspectives and Views.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

PHP Perspective

The PHP Perspective is Zend Studio's default perspective. It incorporates all Zend Studio's PHP project/file creation, inspection and editing functionality.
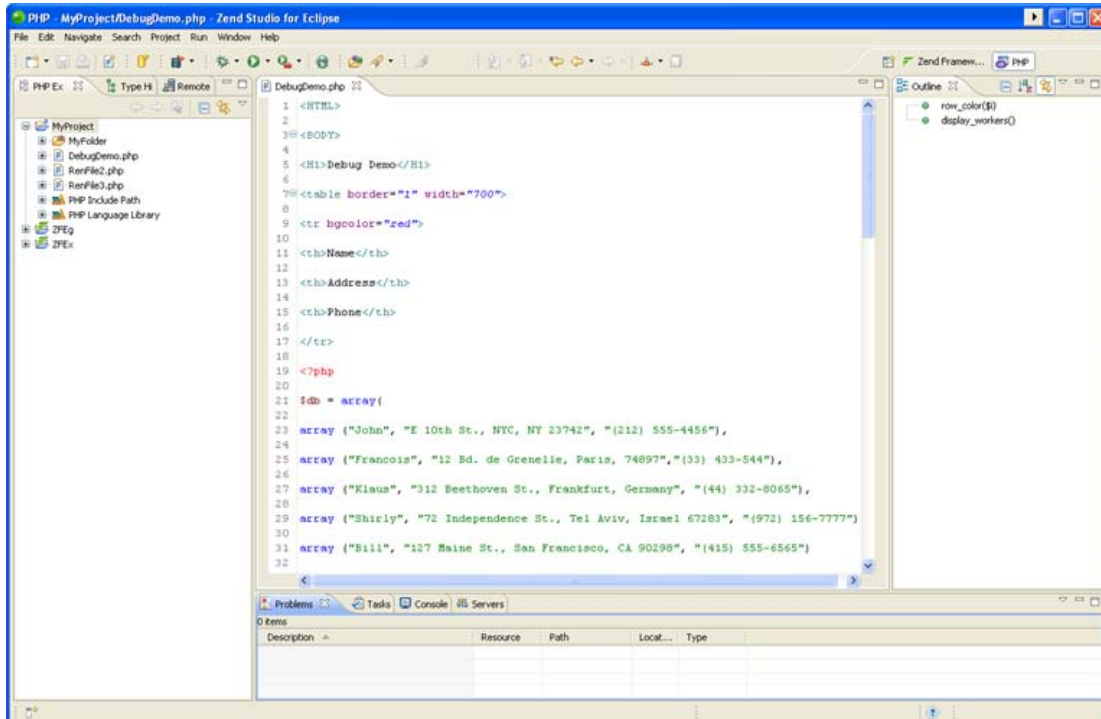
PHP Perspective

The PHP Perspective contains the following views:

- **PHP Explorer View**
- **Outline View**
- **Type Hierarchy View**
- **Problems View** (External Link)
- **Tasks View** (External Link)
- **Console View** (External Link)

PHP Explorer View

The PHP Explorer view is an internal file system browser, allowing you to view all PHP projects and files in your Workspace. It shows the PHP element hierarchy of PHP projects in the Workbench and provides you with a PHP-specific view of your available resources. Within each project, source folders and referenced libraries are shown in the tree. In addition, this view shows all PHP code elements (classes, functions, variables, etc.). Clicking an element or declaration will cause the corresponding code to be displayed in the PHP editor.

See PHP Icons for a description of the icons displayed in the PHP Explorer view.
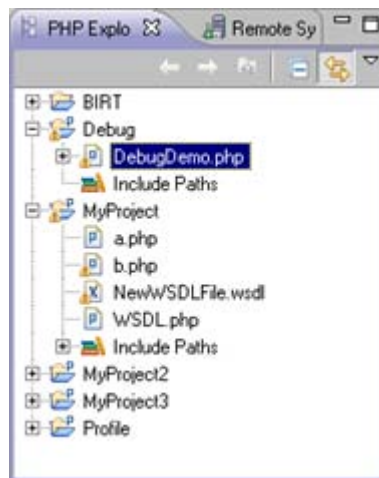
PHP Explorer view

Toolbar Commands

Icon	Name	Description
	Back/Forward	Scrolls backwards and forwards through your recently navigated resources. These icons will only be active if the 'Go into the selected element' option is selected in the PHP Preferences Page (available from Window | Preferences | PHP).
	Up	Navigates up one level. This icon will only be active if the 'Go into the selected element' option is selected in the PHP Preferences Page (available from Window | Preferences | PHP).
	Collapse All	Collapses the list of elements
	Link with Editor	If selected, elements will immediately be displayed in the editor when selected. If unselected, elements will be displayed in the editor when they are double-clicked.

Menu Commands

The view's menu can be accessed through the view menu icon ▽.

Icon	Name	Description
Show ► • Projects / Working Sets	Show	Select to view your projects grouped by Project or Working Set.
	Select Working Set	If Show Projects was selected (above), allows you to select a specific Working Set to view. See PHP Working Sets for more information.
	Deselect Working Set	Deselects the Working Set (if selected).
	Edit Active Working Set	Allows you to edit the selected Working Set. See PHP Working Sets for more information.
�=">	Filters..	Opens the PHP Elements filters dialog which allows you to select which elements will be excluded from being displayed in the view. You select to exclude elements according to name or type.
	Group by namespaces	Sorts elements by namespaces (for projects using PHP 5.3 only).
⇔	Link With Editor	If selected, elements will immediately be displayed in the editor when selected. If unselected, elements will be displayed in the editor when they are double-clicked.

Note:

The PHP Explorer View is displayed by default as part of the PHP Perspective. To manually open the view, go to Window | Show View | Other | PHP Tools | PHP Explorer.

Outline View

The Outline view displays all PHP elements and element types in the current active file. The elements are grouped according to type and are displayed in a tree-like browser.

See [PHP Icons](#) for a description of the icons displayed in the Outline view.
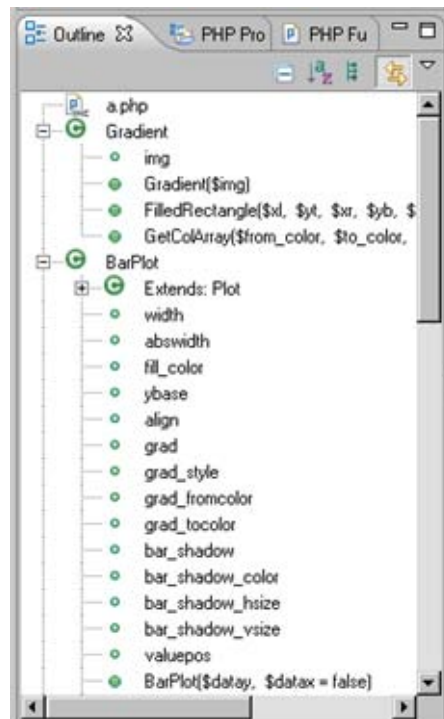
PHP Outline view

Features

- The Outline View is updated interactively according to changes made in the files.
- Each type of PHP element is represented by a unique icon:

Reserved PHP Words

Functions

Templates

Classes

Interfaces

Constants

Variables (public)

Namespaces (PHP 5.3 only)

Use Statements (PHP 5.3 only)

- The Outline view is integrated with the Editor. Therefore, if you select a PHP element in the view, the Editor will jump to the element's declaration in the file in which it is declared.

Note:
Toggle the link to Editor on/off using the Link with Editor button ⇆ .

- The View enables you to add PHPdoc blocks and, if available, Getters and Setters to any PHP element:

To generate a PHP DocBlock or Getter and Setter :
1. Right-click the element in Outline view.
2. Select Source | Add PHP Doc -or- Generate Getters and Setters .

The relevant PHP DocBlock or Getter/Setter will be created above the code for the selected element in the editor.
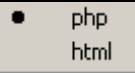
Toolbar Commands

Icon	Name	Description
	Collapse All	Collapses the list of elements
	Sort	Sorts the list alphabetically
	Show Groups	If selected, elements will be displayed in Groups (include files, constants, classes, functions)
	Link with Editor	If selected, elements will immediately be displayed in the editor when single-clicked. If unselected, elements will be displayed in the editor when they are double-clicked.

Menu Commands

The view's menu can be accessed through the view menu icon ▽.
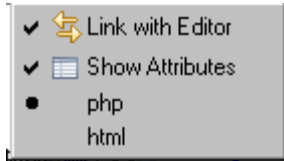
Icon	Name	Description
⇆	Link with Editor	If selected, elements will immediately be displayed in the editor when single-clicked. If unselected, elements will be displayed in the editor when they are double-clicked.
▤	Show Attributes	If selected, element attributes will be displayed. These are defined by the element's [PHP Doc Block]().
● php html	PHP/HTML selection	Toggles the view to display PHP or HTML elements.

Note:

The Outline View is displayed by default as part of the PHP Perspective. To manually open the view, go to Window | Show View | Other | PHP Tools | Outline.

Remote Systems View

The Remote Systems view helps you create, view and manage your connections to remote systems such as FTP and SSH.
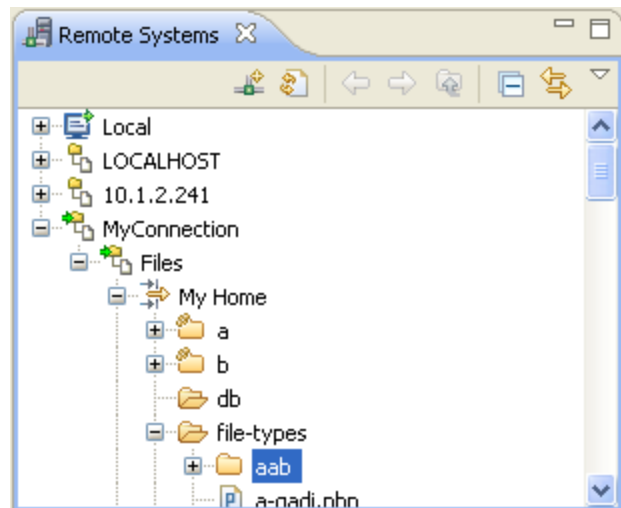
Remote Systems view

Toolbar Commands

Icon	Name	Description
	Define a connection to remote system	Opens the 'New Connection' dialog.
	Refresh information of selected resource	Refreshes the connection information of selected resources.
	Back/Forward	Scrolls backwards and forwards through your projects.
	Up	Navigates up one level
	Collapse All	Collapses the list of elements
	Link with Editor	If selected, files will immediately be displayed in the editor when selected. If unselected, files will be displayed in the editor when they are double-clicked.

Menu Commands

The Remote Systems view menu can be accessed through the view menu icon ▽ .

Name	Description
New Connection...	Opens the 'New Connection' dialog.
Import connection definition	Browse and select the connection definition you would like to import.
Work with Profiles	Opens the Team profile view. See Remote System Explorer Profiles in the RSE User Guide for more information. **Note:** Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).
Refresh All	Refreshes all connections.
Quality Connection Names	Displays the connection names.
Show Filter Pools	Displays filter pools. See Filters, filter pools, and filter pool references in the RSE User Guide for more information. **Note:** Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).
Restore Previous State	Select this option to use locally cached information instead of connecting immediately if you are automatically opening the previously expanded connections when starting RSE.
Preferences	Opens the Remote Systems Preferences page.

Note:
The Remote Systems View is displayed by default as part of the PHP Perspective. To manually open the view, go to **Window \| Show View \| Other \| Remote Systems \| Remote Systems**.

See the RSE User Guide for more on FTP/SSH connectivity.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help \| Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

Type Hierarchy View

The Type Hierarchy view display the hierarchy for a given type (a class name, interface name or class methods, constants and fields.). This allows you to view an element's supertypes (types higher in the hierarchy) or subtypes (lower in the hierarchy) within a tree structure, providing you with an overview of your element's structure.

See Viewing Types in the Type Hierarchy View for information on how to open a type in the Type Hierarchy view.
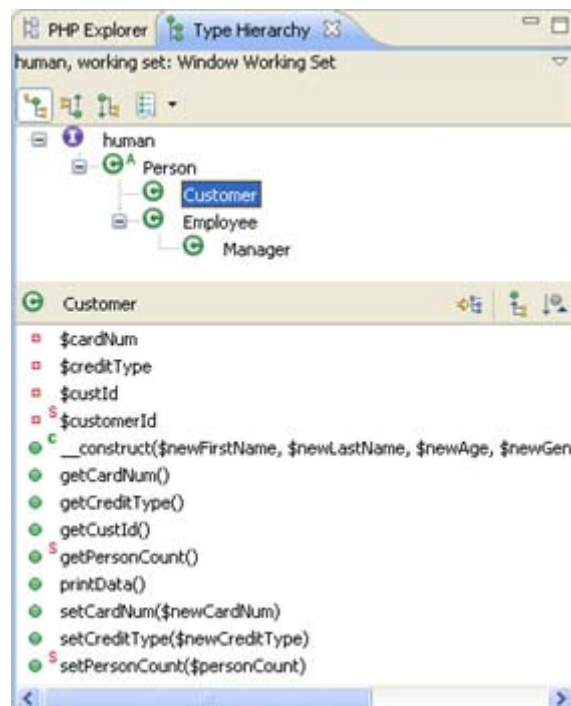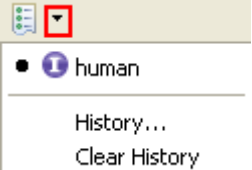
Type Hierarchy view

The Type Hierarchy view consists of two panes:

- The Type Hierarchy Tree - Displays the type's supertypes and/or subtypes.
- Member list pane - Displays the type's members.

Tree Pane Toolbar Commands

Icon	Name	Description
	Show the Type Hierarchy	Displays the type in its full context (i.e., superclasses and subclasses) in the Type Hierarchy view.
	Show the Supertype Hierarchy	Displays the supertypes and the hierarchy of all implemented interfaces of the type. Note: The selected type is always at the top level, in the upper-left corner.
	Show the Subtype Hierarchy	Displays the subtypes of the selected class and/or all implementors of the interface in the Hierarchy view. Note: The selected type is always at the top level, in the upper-left corner.
	Previous Hierarchy Inputs	Displays a history of previously displayed type hierarchies.

Member List Pane Toolbar Commands

Icon	Name	Description
	Lock View and Show Members in Hierarchy	Only displays the members implementing the selected method. When the view is locked, the member list pane no longer tracks the selection in the Type Hierarchy Tree Pane.
	Show All Inherited Members	Shows or hides all methods and fields inherited by base classes. When this option is enabled, the name of the type that defines the method is appended to the method name.
	Sort Members by the Defining Type	Sorts the members according to the type in which they are defined.

Menu Commands

The view's menu can be accessed through the view menu icon ▽.

Icon	Name	Description
	Show the Type / Supertype or Subtype Hierarchy	Selects whether to display the type, supertype or subtype hierarchy in the Type Hierarchy Tree Pane (see above).
	Select/Deselect/Edit Working Set	Allows you to view only types from within a specific Working Set. See PHP Working Sets for more information.
	Layout	Allows you to select the display of the panes within the Type Hierarchy view. The options are: ▪ Vertical View Orientation ▪ Horizontal View Orientation ▪ Automatic View Orientation ▪ Hierarchy View Only
	Show Qualified Type Names	Displays only qualified type names
	Link With Editor	If selected, elements will immediately be displayed in the editor when selected. If unselected, elements will be displayed in the editor when they are double-clicked.

PHP Debug Perspective

The PHP Debug Perspective can be launched automatically when a Debug session is run. It contains views which allow you to control and monitor the debugging process.
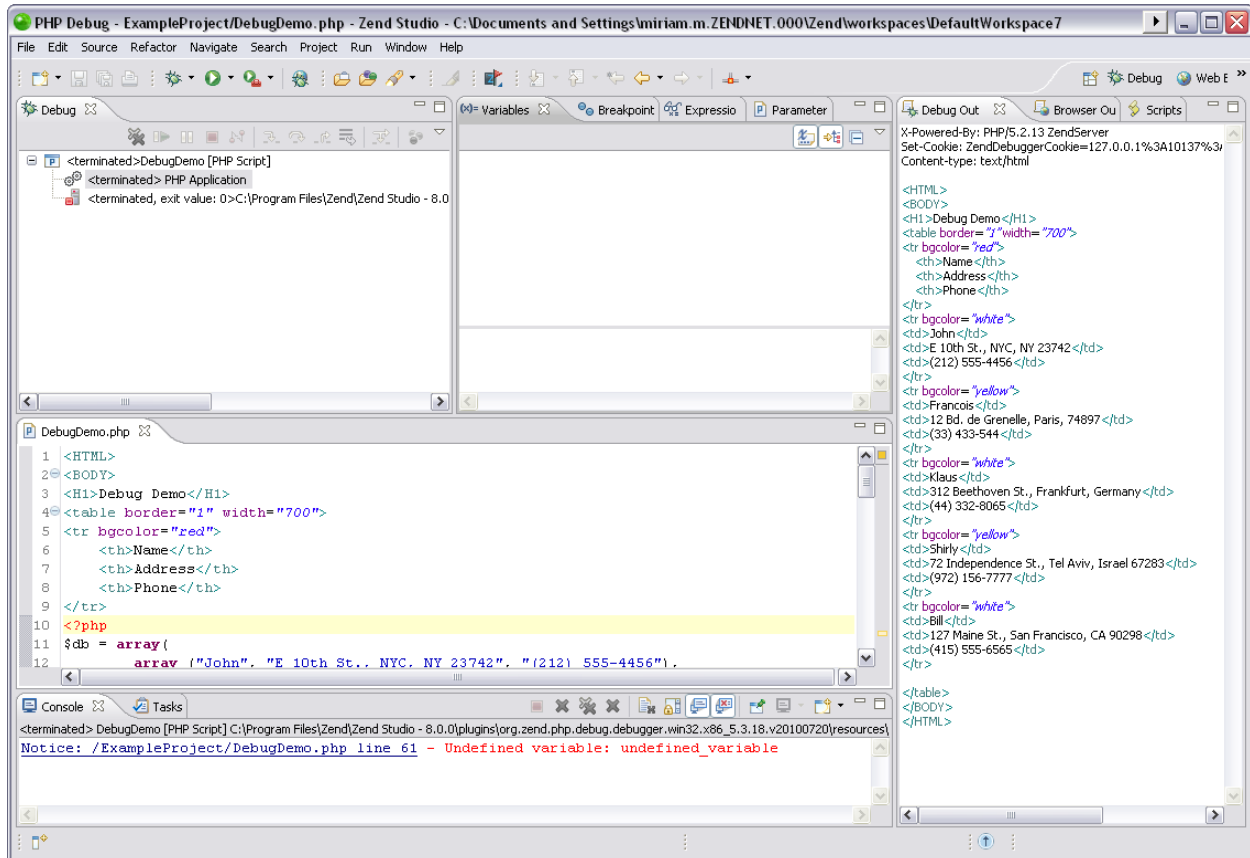
The PHP Debug Perspective contains the following views:

- **Debug View** - Here you can control (stop, pause, and resume) the debugging process. You can also decide whether to step into, step over or step return (step out of) certain functions.

- **Variables View** - Displays the various variables in your script.

- **Breakpoints View** - Displays the breakpoints you have entered.

- **Parameter Stack View** - Displays the parameters through which functions are reached.

- **Debug Output View** - Displays the textual output of the script. This will be updated as the debugging process continues.

- **Browser Output View** - Displays the output of the script to a browser. This will be updated as the debugging process continues.

- **Expressions View** - Displays the progress of selected variables. The view will only be displayed if you have selected to watch a variable.

- **Scripts View** - Displays a list of available scripts.

- Editor - Displays the code at the relevant sections, according to which line is selected in the Debug View window.
- <u>Console View</u> (External Link) - Displays any error and warning messages.
- <u>Tasks View</u> (External Link) - Displays tasks that were added to your script (if applicable).

Note:
By default, a dialog will appear asking whether you want to open the Debug Perspective when a debugging session is run. To change this behavior, open the Perspectives Preferences dialog by going to Window	Preferences	Run/Debug	Perspectives and select Always, Never or Prompt in the 'Open the associated perspective when launching' category.

Debug View [PHP Debug Perspective]

The Debug view displays the stack trace and allows you to monitor and control the Debugging process.

Toolbar Commands

Icon	Name	Description
	Remove Terminated Launches	Remove any terminated debug sessions from the list.
	Resume	Continue the debugging process until the next breakpoint, or until the end of the debugging process.
	Terminate	Stop the debugging process.
	Step Into	Step into the next method call at the currently executing line of code.
	Step Over	Step over the next method call (without entering it) at the currently executing line of code. The method will still be executed.
	Step Return	Return from a method which has been stepped into. The remainder of the code that was skipped by returning is still executed.
	Use Step Filters	Enables/disables the <u>step filters</u> functionality.

629

Variables View [PHP Debug Perspective]

The Variables view displays information about the variables associated with the stack frame selected in the Debug View. Selecting a variable will display details in the detail pane below the view. Expanding the list under a variable will display its fields.

Toolbar Commands

Icon	Name	Description
	Show Type Names	If selected, type names will be displayed.
	Show Logical Structure	Shows the logical structure.
	Collapse All	Collapses the list.

Menu Commands

The view's menu can be accessed through the view menu icon .

Name	Description
Layout	Defines the view's layout: ▪ Vertical View Orientation - The details pane will be displayed at the bottom of the Variables view. ▪ Horizontal View Orientation - The details pane will be displayed to the right of the Variables view. ▪ Variables View Only - Only the Variables view will be displayed. ▪ Show columns - Divide the view into columns. ▪ Set Columns - Only available if "Show columns" is selected. Allows you to choose which of the following columns to display: • Name • Declared Type • Value • Actual Type

Note:

The Variables View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective.

To manually open the view, go to **Window | Show View | Other | Debug | Variables**.

Breakpoints View [PHP Debug Perspective]

The Breakpoints view displays and allows you to monitor and control the breakpoints set in the files being debugged.

Toolbar Commands

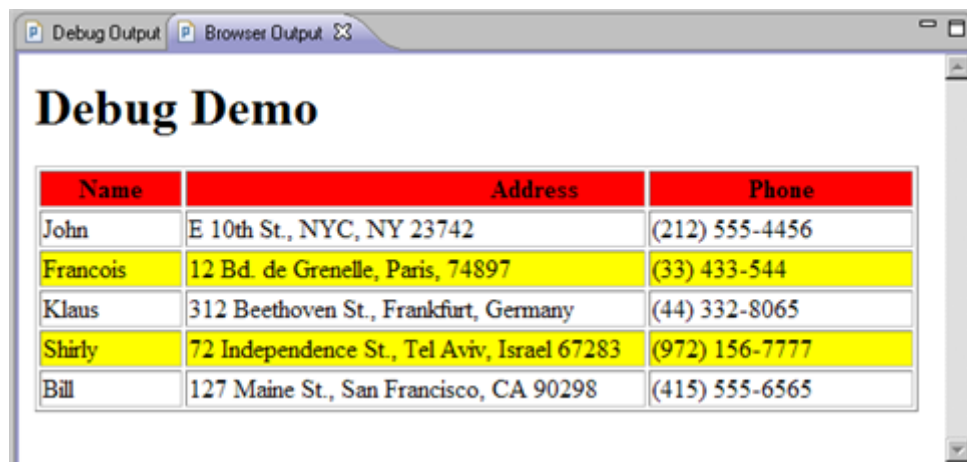| Icon	Name	Description
✖	Remove Selected Breakpoints	Removes the selected Breakpoints from the file.
✖	Remove All Breakpoints	Removes all Breakpoints from the file.
⚙	Show Breakpoints Supported By Selected Targets	If selected, only breakpoints supported by the current 'debug target' will be displayed. For example, iIf a PHP file is being debugged, only PHP breakpoints will be displayed.
⬐	Go to File for Breakpoint	Opens the resource in which the breakpoint is located.
⊘	Skip All Breakpoints	If selected, all breakpoints will be skipped and execution will not stop.
⊞	Expand All	Expands all items in the list.
⊟	Collapse All	Collapses all items in the list.
⇆	Link with Debug View	If selected, clicking a breakpoint will link with the Debug view.

Menu Commands

The view's menu can be accessed through the view menu icon ▽.

Name	Description

Group By	• Breakpoints
	• Breakpoint Types
	• Breakpoint Working Sets
	• Files
	• Projects
	• Resource Working Sets
	• Advanced
Select/Deselect Default Working Set	Allows you to choose the default breakpoint working set from the Default Working Set dialog.
Working Sets	Opens the Working Sets dialog.

Note:

The Breakpoints View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective.

To manually open the view, go to **Window | Show View | Other | Debug | Breakpoints**.

Parameter Stack View [PHP Debug Perspective]

The Parameter Stack view displays the parameters executed when stepping into a function during the debugging process.
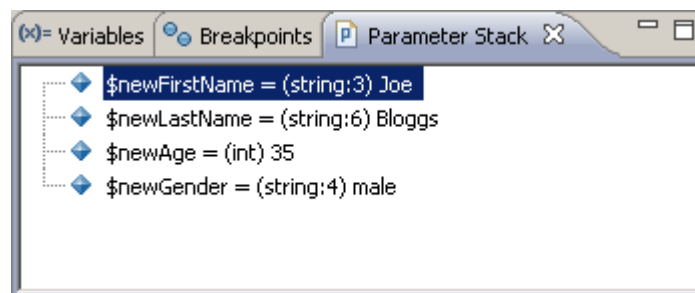
The following information can be gathered from the Parameter Stack view:

- Called Parameters - The called parameters as written in the line or code.
- The Main Calling Line of Code - The line number in which the calling statement occurred (in parentheses).
- Parameter Values - Shows the parameter values that were passed in the function call.

Note:

The Parameter Stack View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | PHP Tools | Parameter Stack**.

Expressions View [PHP Debug Perspective]

The Expressions view will not open by default when a debugging session is launched, but only when you have selected to watch a variable or create a watch expression.
To watch a variable, right-click a variable in the editor or from the variables view and select **Watch** or **Add Watch Expression**. The Expressions view will open and the variable will be added to it. The variable's information will be updated as the debugging process continues.

Note:

To manually open the Expressions view, go to **Window | Show View | Debug | Expressions**.

The Expressions view allows you to monitor certain variables which you have decided to 'watch' during the debugging process. Selecting a variable will display details in the detail pane below the view. Expanding the list under a variable will display its fields.

Toolbar Commands

Icon	Name	Description
	Show Type Names	Displays type names
	Show Logical Structure	Displays the logical structure.
	Collapse All	Collapses the list.

Menu Commands

The view's menu can be accessed through the view menu icon ▽.

Name	Description
Layout	Defines the view's layout: • Vertical View Orientation - The details pane will be displayed at the bottom of the Variables view. • Horizontal View Orientation - The details pane will be displayed to the right of the Variables view. • Expressions View Only - Only the Watched Variables pane will be displayed.

Debug Output View [PHP Debug Perspective]

The Debug Output view shows the textual output of the script. This will be updated as the debugging process continues.

Note:

The Debug Output View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | PHP Tools | Debug Output**.

Browser Output View [PHP Debug Perspective]

The Browser Output view will show the output of the script to a browser. This will be updated as the debugging process continues.

Note:

The Browser Output View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | PHP Tools | Browser Output**.

PHP Profile Perspective

The PHP Profile Perspective can be launched automatically when a Profile session is run. It allows you to view all the information relevant to your scripts.
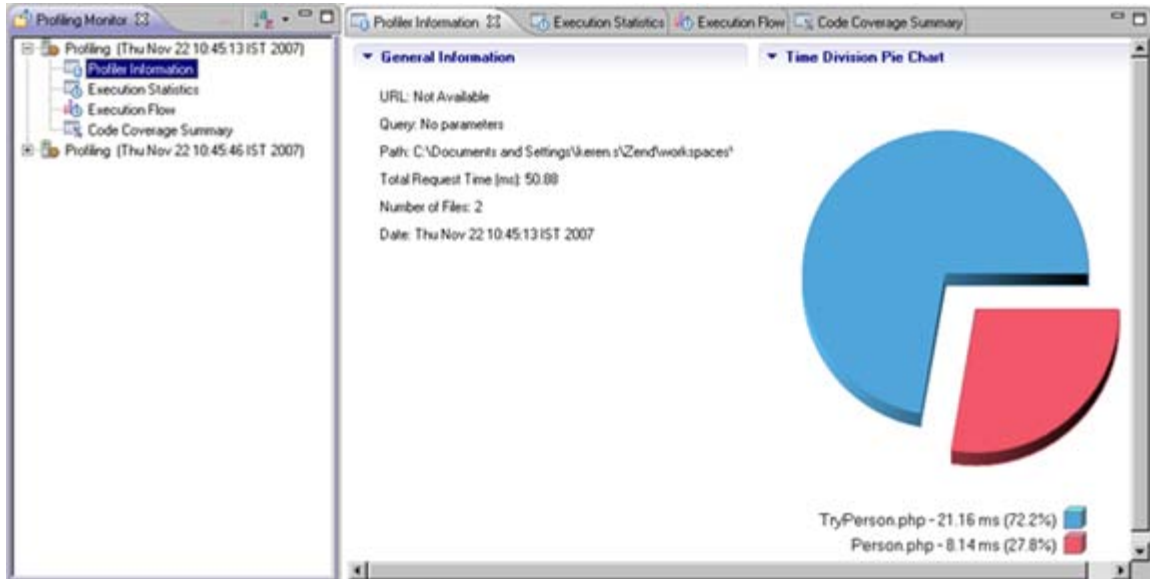
Profiling Perspective

The PHP Profile Perspective contains the following views:

- Profiling Monitor
- Profiler Information
- Execution Statistics
- Execution Flow

Profiling Monitor View

The Profiling Monitor view displays a list of previously run Profiling sessions.

Expanding the list under a Profiling session allows you to select a Profiling view to display.
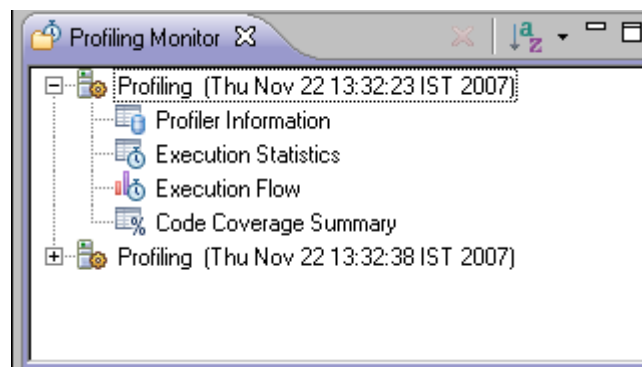
Profiling Monitor view

Icon	Name	Description
✖	Delete Session	Deletes a Profiling session from the list. This will be enabled if a profiling session is selected.
↓a z	Sort Profile Sessions	Click the arrow next to the Profile Session to sort the Profile Session list by date or time.

Note:

The Profiling Monitor view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Profiling Monitor.

Profiler Information View

The Profiler Information view provides general information about profiling sessions such as: duration, date, number of files constructing the requested URL and more. In addition, a Pie Chart is generated that shows the time division of the files in the URL.
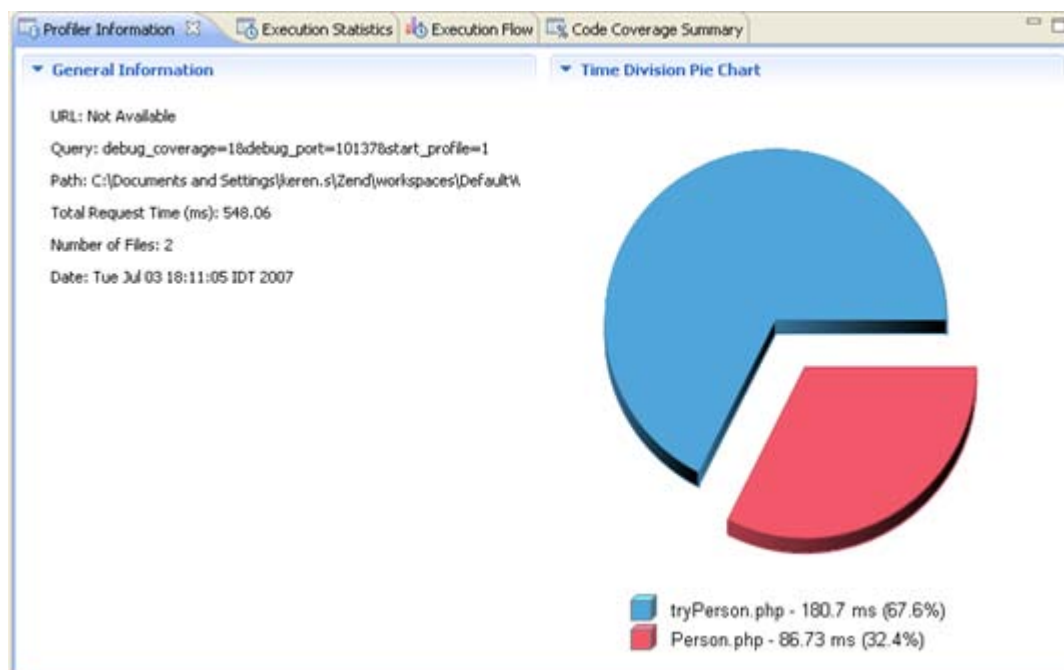
Profiler Information view

The right side of the view displays time division in a pie chart and the left side provides the following information:

- URL - The URL analyzed (if applicable)
- Query - The specific query parameters

638
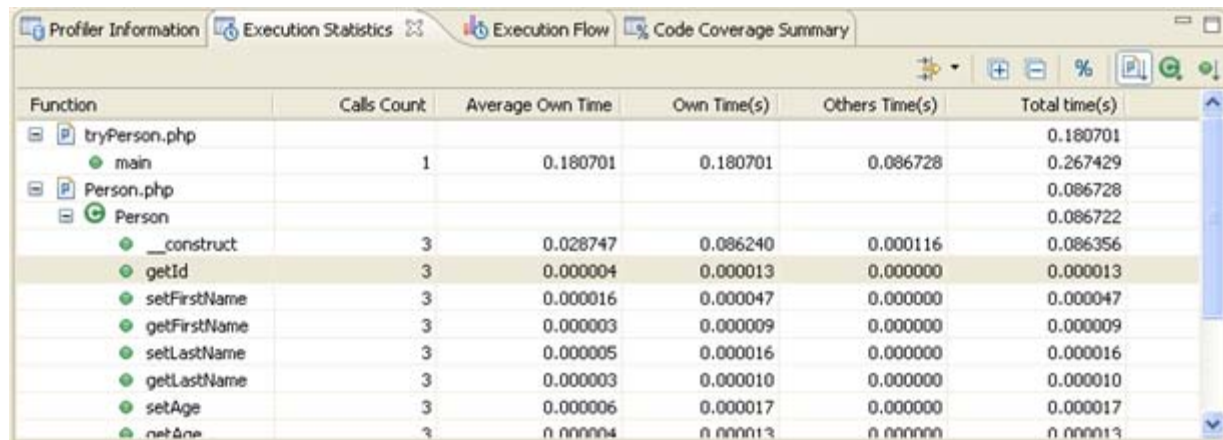
- Path - The location of the first file called
- Total Request Time - Total processing time for the entire page
- Number of Files - Number of files processed
- Date - Date and time that the profiling took place

Note:

The Profiler Information View view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Profiling Monitor.

Execution Statistics View

The Execution Statistics view displays the list of files that were called during the profiling process and detailed information on processing times for elements within the files.

Execution Statistics

The window contains statistics relevant to each element as follows:

- Function - The name and location of the function.
- Calls Count - The number of times that the function was called.
- Average Own Time - The average duration without internal calls.
- Own Time(s) - The net process duration without internal calls.
- Others Time(s) - Time spent on calling other files.
- Total Time(s) - The total time taken to process.

Note:

Click the 'Show as percentage' button on the toolbar to see the statistics as percentages rather than times.

Right- clicking a function in the list gives you the option to 'Open Function Invocation statistics'. This will open a view with statistics about the selected function, the functions it was invoked by and functions that it invoked.

Toolbar Commands

| Icon	Name	Description
	Filters...	Click the arrow next to the icon to select to display only the results with: • Highest 10 own time • Highest 10 calls • Highest 10 total time • Highest 10 average time • -Or- No filter. Click the icon itself or select Manage Filters from the list to launch the Edit filter dialog which allows you to create or edit your own filter conditions.
	Expand/Collapse all	Expands/collapses the list.
	'Show as Percentage'	Toggles the view to show your times in seconds or percentages.
	Group by File	Sorts the list by file.
	Group by Class	Sorts the list by class.
	Group by Function	Sorts the list by function.

Note:

The Execution Statistics view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Execution Statistics.

Execution Flow View

The Execution Flow view shows the flow of the execution process and summarizes percentages and times spent on each function.

Function Information

The view displays the following information for each function:

- Function - Function name.
- File - The file in which the function is located.

- Total Execution Time - Percent of time taken per function.

- Duration Time - Time taken per function. In milliseconds.

Profiler Execution Flow

Right Click Menu

Right-clicking a function in the list gives you the option to:

- View Function Call - Opens the selected function call in the editor.
- View Function Declaration - Opens the selected function declaration in the editor.
- Open Function Invocation statistics - Opens a view with statistics about the selected function, the functions which the selected function was invoked by, and the functions invoked by the selected function.

Toolbar Commands

Icon	Name	Description
	Expand/Collapse all	Expands/collapses the list.
%	'Show as Percentage'	Toggles the view to show your times in seconds or percentages.
	Sort Profile Sessions	Click the arrow next to the Profile Session to sort the Profile Session list by the Order in which the functions were executed or by Duration Time.

Note:

The Execution Flow view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Execution Flow.

Code Tracing Perspective

The Zend Server Code Tracer perspective allows you to use the Zend Server Code Tracing feature. Integrating Zend Server Code Tracing into Zend Studio allows you to open the source of the execution data inside of your environment. This feature is useful in resolving performance issues, memory usage issues, and functional errors that occur in a production environment.

The perspective is accessed by going to **Window | Open Perspective | Other | Code Tracing**.
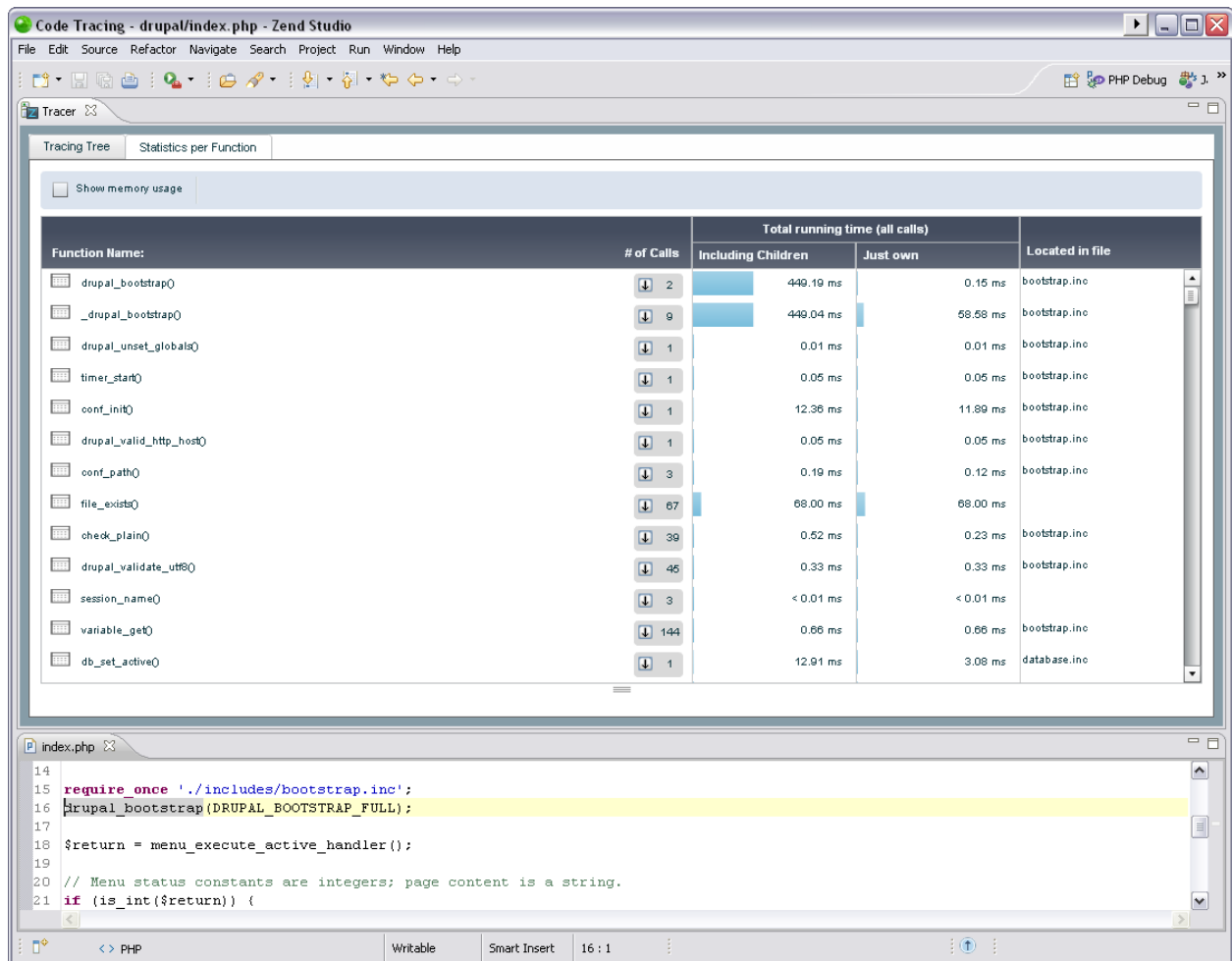
The Zend Server Code Tracer perspective contains the following views:

- Tracer View

Tracer View

The Tracer view displays the Imported Zend Server Event File.

The Tracer view includes the following tabs:

- Tracing Tree - Displays the call tree for a selected event or trace file. For more information see Code Tracing Tree in the Zend Server Online Documentation.
- Statistics per Function - A table based display that provides a statistical perspective of the data captured in the request. For more information see Code Tracing Statistics in the Zend Server Online Documentation.

Note:
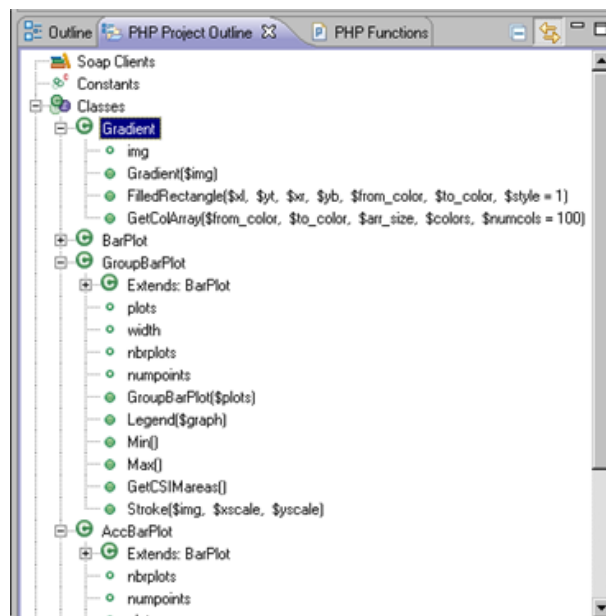
The Tracer View is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | Zend Servers | Tracer**.

Additional Views

PHP Functions View

The PHP Functions view lists most commonly used PHP Classes, Constants and Iterators. The PHP Functions view can be used in order to easily add functions into your scripts. To add a function to your code, simply place the cursor in the required position in the Editor and double-click the required element from the list.
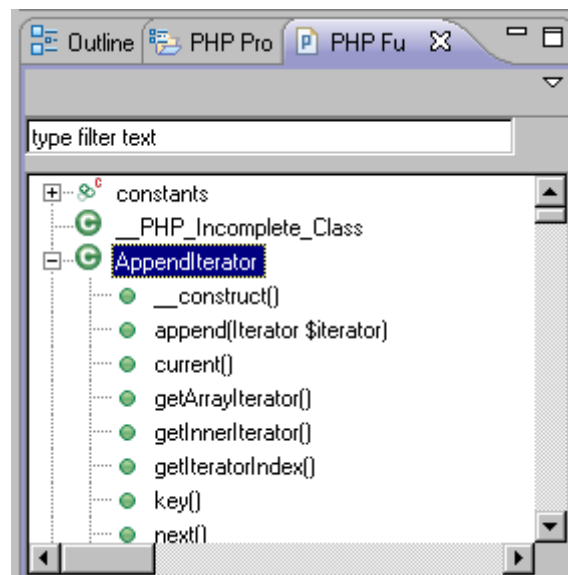
PHP Functions view

Right-clicking a function in PHP Functions view and selecting Open Manual will open an online version of the PHP manual with an explanation about most of the functions on the list.

A new browser window will open with an explanation of the function from the PHP Manual.

If the browser opens with a 'Cannot find server' error message, it means the function does not have a description assigned to it in the PHP Manual.

Sites for viewing the PHP Manual can be added and edited from the [PHP Manual Preferences](#) page.

Toolbar commands

Icon	Name	Description
type filter text	Filter Text box	Allows you to find a particular function. Start typing the function name. Relevant results will be displayed below it.

The PHP Functions View is displayed by default as part of the PHP Perspective. To manually open the view, go to Window | Show View | Other | PHP Tools | PHP Functions.

PHP Project Outline View

The PHP Project Outline view displays a list of Soap Clients, Namespaces (PHP 5.3 only), Constants, Classes and Functions for all files within the selected project.
Selecting an element in the PHP Project Outline view will open the relevant file in the editor.

To access the view, go to Window | Show View | Other | PHP Tools | PHP Project Outline.

Toolbar commands

Icon	Name	Description
	Collapse All	Collapses the list of elements
	Link with Editor	If selected, elements will immediately be displayed in the editor when they are single-clicked. If unselected, elements will be displayed in the editor when they are double-clicked.

JavaScript Debug Perspective

The Debug Perspective will be launched automatically when a Web Debug session is run. It contains views which allow you to control and monitor the debugging process.
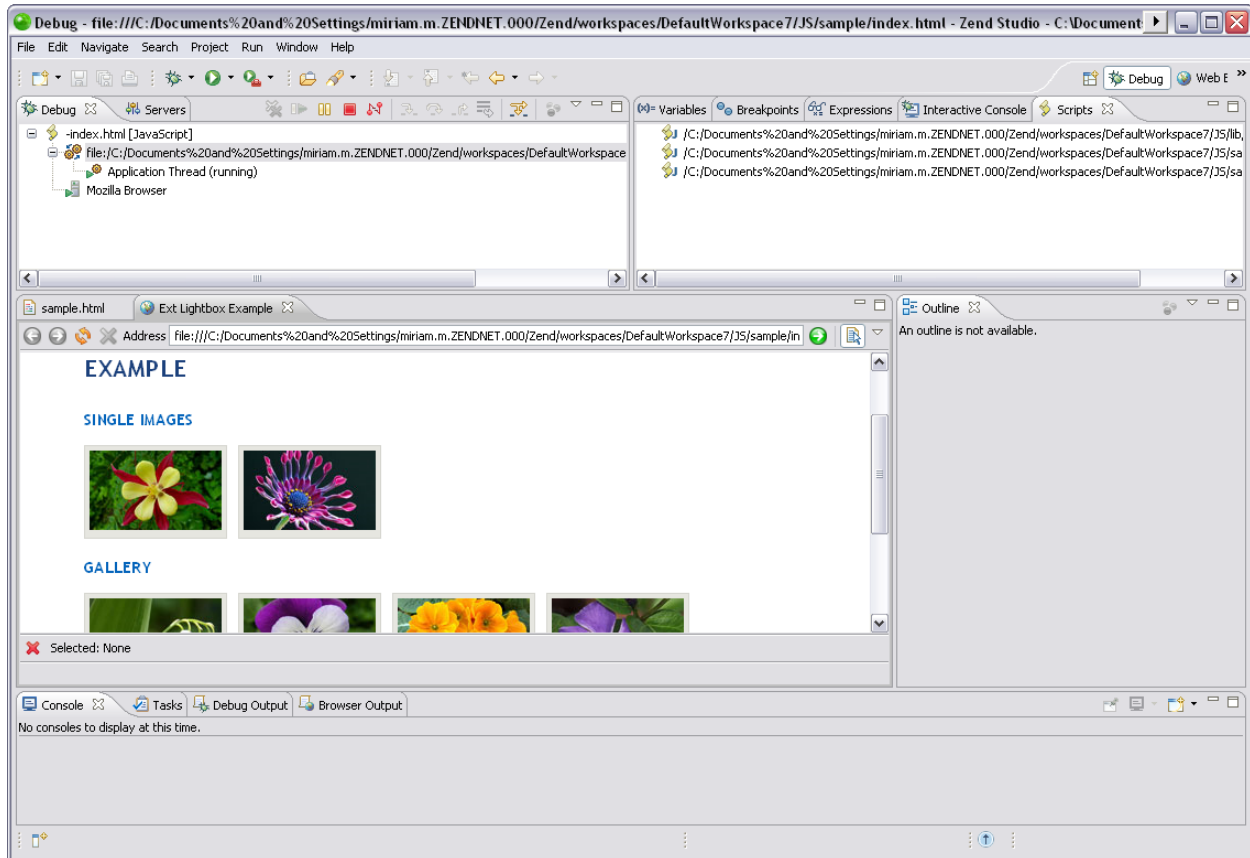
The Debug Perspective contains the following views:

- **Debug View [Debug Perspective]** - Here you can control (stop, pause, and resume) the debugging process. You can also decide whether to step into, step over or step return (step out of) certain functions.
- **Variables View [Debug Perspective]** - Displays the various variables in your script.
- **Breakpoints View [Debug Perspective]** - Displays the breakpoints you have entered.
- **Expressions View [Debug Perspective]** - Displays the progress of selected variables. The view will only be displayed if you have selected to watch a variable.
- **Scripts View** - Displays a list of available scripts.
- Editor - Displays the code at the relevant sections, according to which line is selected in the Debug View window.
- **Internal Web Browser** - Displays the URL of the code you are debugging.
- **Console View** (External Link) - Displays any error and warning messages
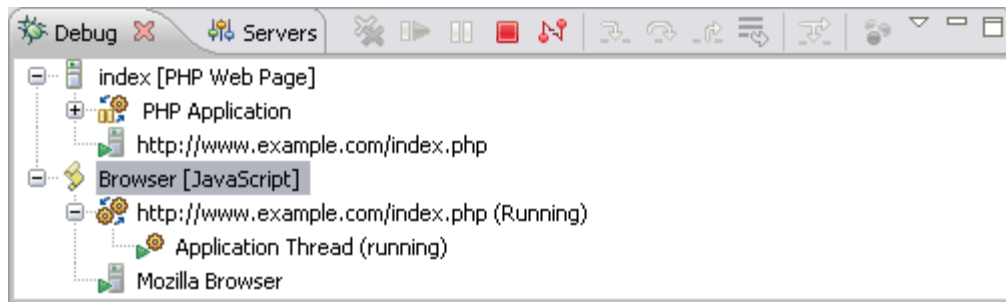
- Tasks View (External Link) - Displays tasks that were added to your script (if applicable).
- Debug Output View [Debug Perspective] - Displays the textual output of the script. This will be updated as the debugging process continues.
- Browser Output View [Debug Perspective] - Displays the output of the script to a browser. This will be updated as the debugging process continues.

Note:

By default, a dialog will appear asking whether you want to open the Debug Perspective when a debugging session is run. To change this behavior, open the Perspectives Preferences dialog by going to **Window | Preferences | Run/Debug | Perspectives** and select 'Always', 'Never' or 'Prompt' in the "Open the associated perspective when launching"category.

Debug View [Debug Perspective]

The Debug view displays the stack trace and allows you to monitor and control the Debugging process.
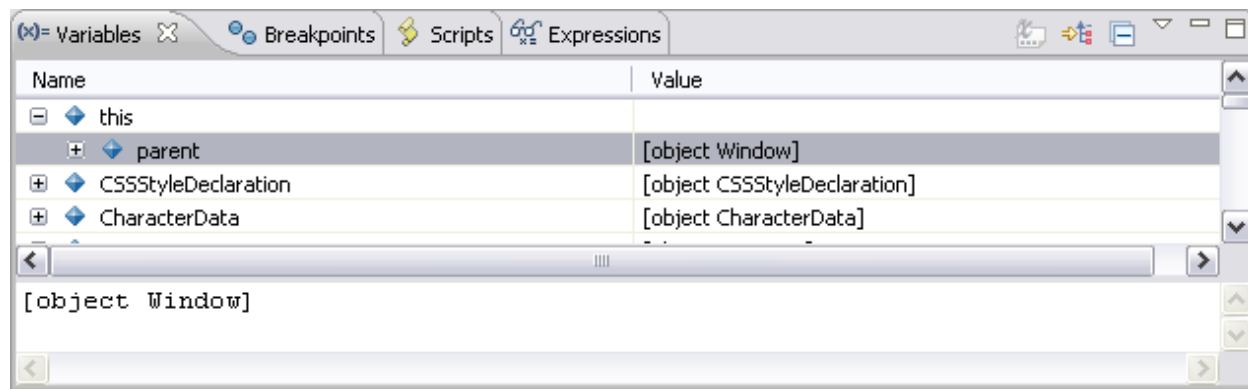

Debug View Toolbar Commands

Icon	Name	Description
	Remove Terminated Launches	Remove any terminated debug sessions from the list.
	Resume	Continue the debugging process until the next breakpoint, or until the end of the debugging process.
	Pause	Pause the debugging process.
	Terminate	Stop the debugging process.
	Step Into	Step into the next method call at the currently executing line of code.
	Step Over	Step over the next method call (without entering it) at the currently executing line of code. The method will still be executed.
	Step Return	Return from a method which has been stepped into. The remainder of the code that was skipped by returning is still executed.
	Use Step Filters	Enables/disables the step filters functionality.

Note:

The Debug View [Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | Debug | Debug**.

Variables View [Debug Perspective]

The Variables view displays information about the variables associated with the stack frame selected in the Debug View. Selecting a variable will display details in the detail pane below the view. Expanding the list under a variable will display its fields.
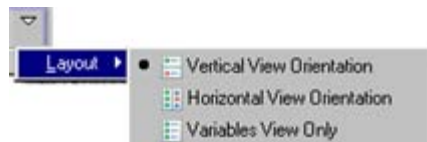

Note:
Right-click a variable and select **Watch** to add the variable to the Expressions View.

Variables View Toolbar Commands

Icon	Name	Description
	Show Type Names	If selected, type names will be displayed.
	Show Logical Structure	Shows the logical structure.
	Collapse All	Collapses the list.

Variables View Menu Commands

The view's menu can be accessed through the view menu icon .
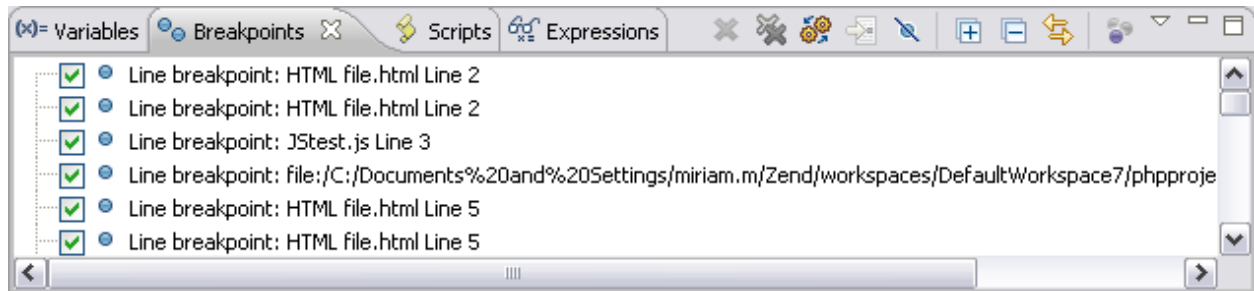

Name	Description

Layout	Defines the view's layout:
	▪ Vertical View Orientation - The details pane will be displayed at the bottom of the Variables view.
	▪ Horizontal View Orientation - The details pane will be displayed to the right of the Variables view.
	▪ Variables View Only - Only the Variables view will be displayed.

Note:

The Variables View [Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | Debug | Variables**.

Breakpoints View [Debug Perspective]

The Breakpoints view displays and allows you to monitor and control the breakpoints set in the files being debugged.
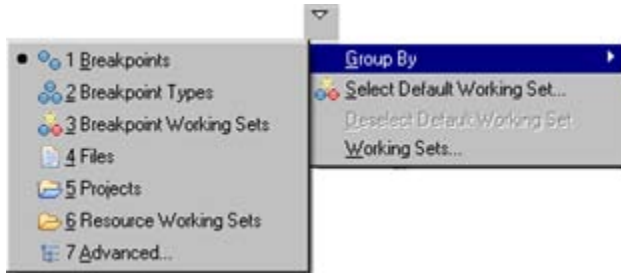

Breakpoints View Toolbar Commands

Icon	Name	Description
	Remove Selected Breakpoints	Removes the selected Breakpoints from the file.
	Remove All Breakpoints	Removes all Breakpoints from the file.
	Show Breakpoints Supported By Selected Targets	If selected, only breakpoints supported by the current 'debug target' will be displayed. For example, iIf a PHP file is being debugged, only PHP breakpoints will be displayed.
	Go to File for Breakpoint	Opens the resource in which the breakpoint is located.
	Skip All Breakpoints	If selected, all breakpoints will be skipped and execution will not stop.
	Expand All	Expands all items in the list.
	Collapse All	Collapses all items in the list.
	Link with Debug View	If selected, clicking a breakpoint will link with the Debug view.

Breakpoints View Menu Commands

The view's menu can be accessed through the view menu icon ▽.

Name	Description
Group By	▪ Breakpoints ▪ Breakpoint Types ▪ Breakpoint Working Sets ▪ Files ▪ Projects ▪ Resource Working Sets ▪ Advanced
Select/Deselect Default Working Set	Allows you to choose the default breakpoint working set from the Default Working Set dialog.
Working Sets	Opens the Working Sets dialog.

Note:

The Breakpoints View [Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | Debug | Breakpoints**.

Scripts View

The Scripts view displays an expandable list of the available scripts of the HTML file you are debugging in the [Debug Perspective](#).
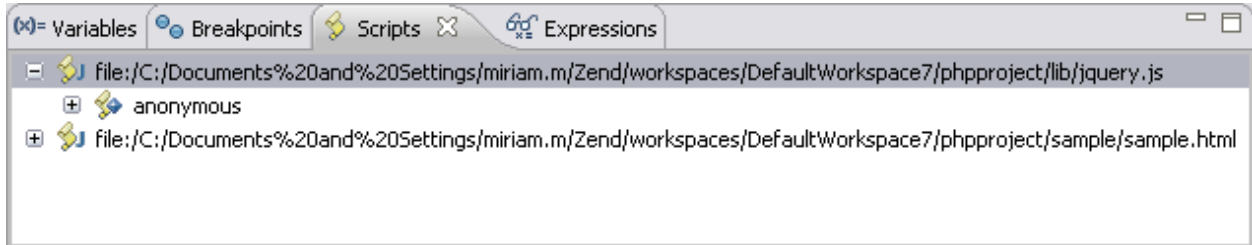
Double clicking on a script will open its code in an editor and an outline of the elements in the Outline view. Select an element in the Outline view to highlight it in the editor.

Expressions View [Debug Perspective]

The Expressions view will not open by default when a debugging session is launched, but only when you have selected to watch a variable.

To watch a variable, right-click a variable in the editor or from the Variables View and select **Watch**. The Expressions view will open and the variable will be added to it. The variable's information will be updated as the debugging process continues.

Note:

To manually open the Expressions view, go to **Window | Show View | Debug | Expressions**.

The Expressions view allows you to monitor certain variables which you have decided to 'watch' during the debugging process. Selecting a variable will display details in the detail pane beside the view. Expanding the list under a variable will display its fields.
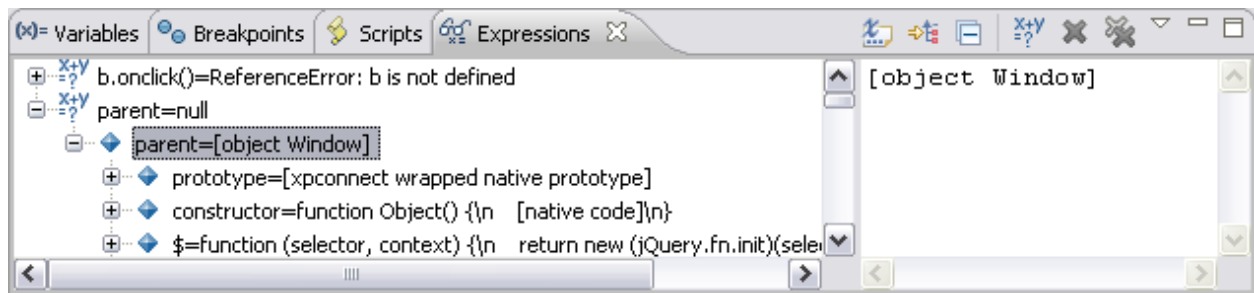
Expressions View Toolbar Commands

Icon	Name	Description
	Show Type Names	Displays type names
	Show Logical Structure	Displays the logical structure.
	Collapse All	Collapses the list.
	Create a New Watch Expression	Allows you to define a new watch expression to add to the list.
	Remove Selected Expression	Removes the expression selected in the view.
	Remove All Expressions	Removes all expressions in the view.

Expressions View Menu Commands

The view's menu can be accessed through the view menu icon ▽.

Name	Description
Layout	Defines the view's layout: ▪ Vertical View Orientation - The details pane will be displayed at the bottom of the Variables view. ▪ Horizontal View Orientation - The details pane will be displayed to the right of the Variables view. ▪ Expressions View Only - Only the Watched Variables pane will be displayed.

Debug Output View [Debug Perspective]

The Debug Output view shows the textual output of the script. This will be updated as the debugging process continues.
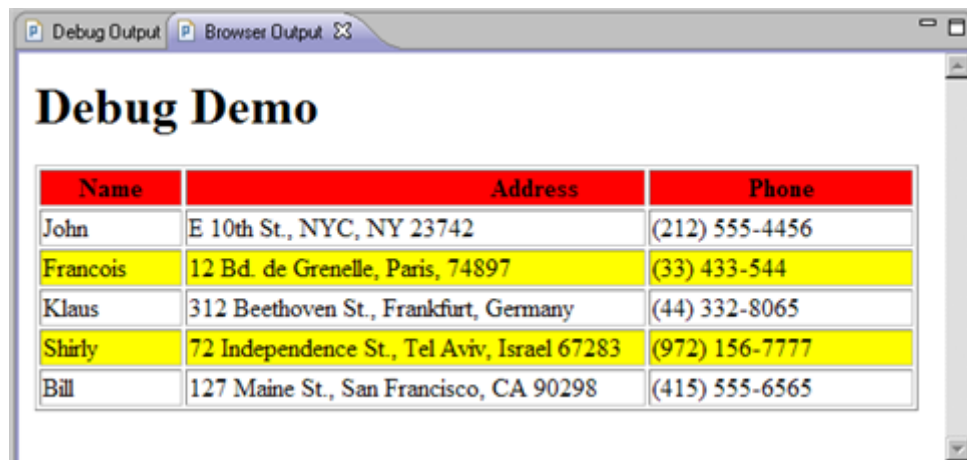

Note:

The Debug Output View [Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | PHP Tools | Debug Output**.

Browser Output View [Debug Perspective]

The Browser Output view will show the output of the script to a browser. This will be updated as the debugging process continues.

Documentation View

The JSDoc functionality allows you to parse inline documentation for JavaScript source code in your project by opening it with '/**' and closing it with the standard */'. When using the function previously identified, the Documentation view shows the documentation of the JavaScript code including the parameters, returns, and exceptions. These are defined using tags ('@' - attributes). This documentation will also be added to functionalities such as Content Assist.

For more information on JSDoc see the JSDoc documentation (http://jsdoc.sourceforge.net/).
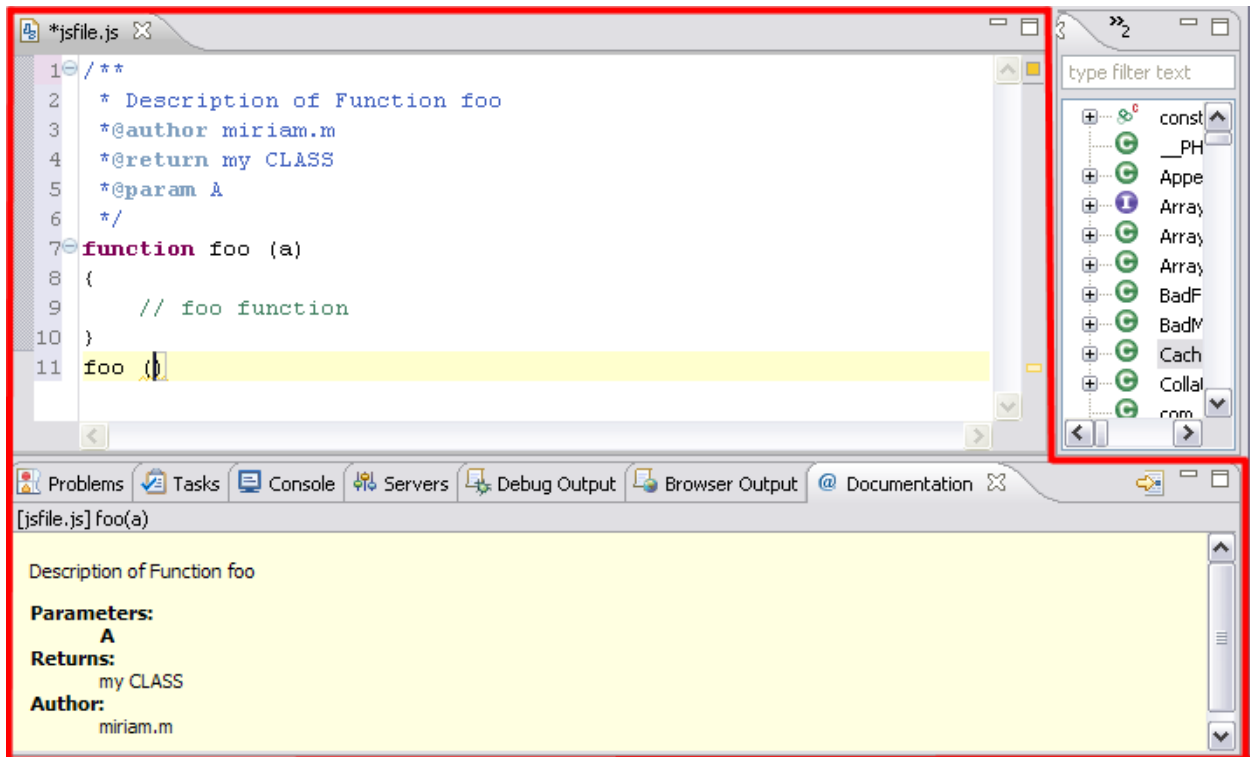
Targets View

Defining targets is a crucial stage in deploying your application. To help developers manage targets in a more efficient way, Zend Studio includes the option to add a Target view to the workspace. Using this view, developers can manage targets by adding new targets, editing or removing existing targets and detecting new targets.

For more information, see Managing Targets.

To view your targets, select **Window | Show View | Targets** from the Menu bar.

The Targets view is added to your View tabs at the bottom of your screen.
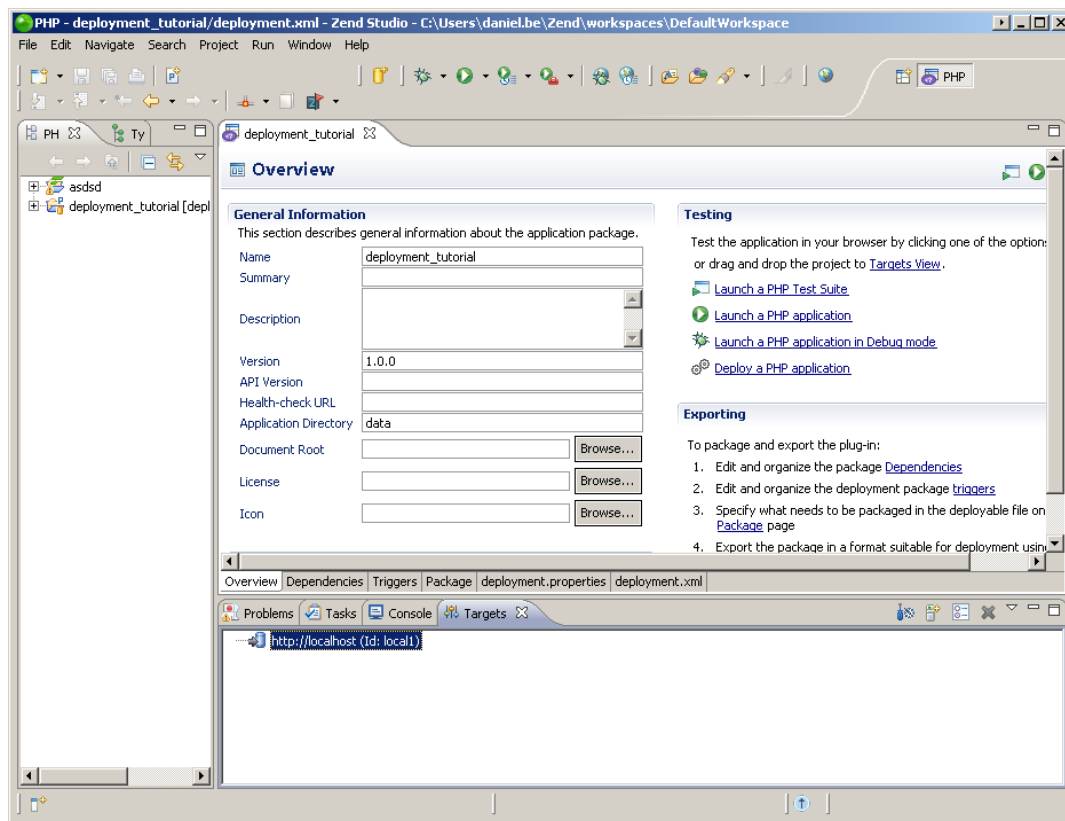
Web Browser Tools Perspective

The Web Browser Tools perspective opens the Internal Web Browser inside of your environment. To

open this perspective go to **Window | Open Perspective | Web Browser Tools** and click from the

main toolbar to open a URL.
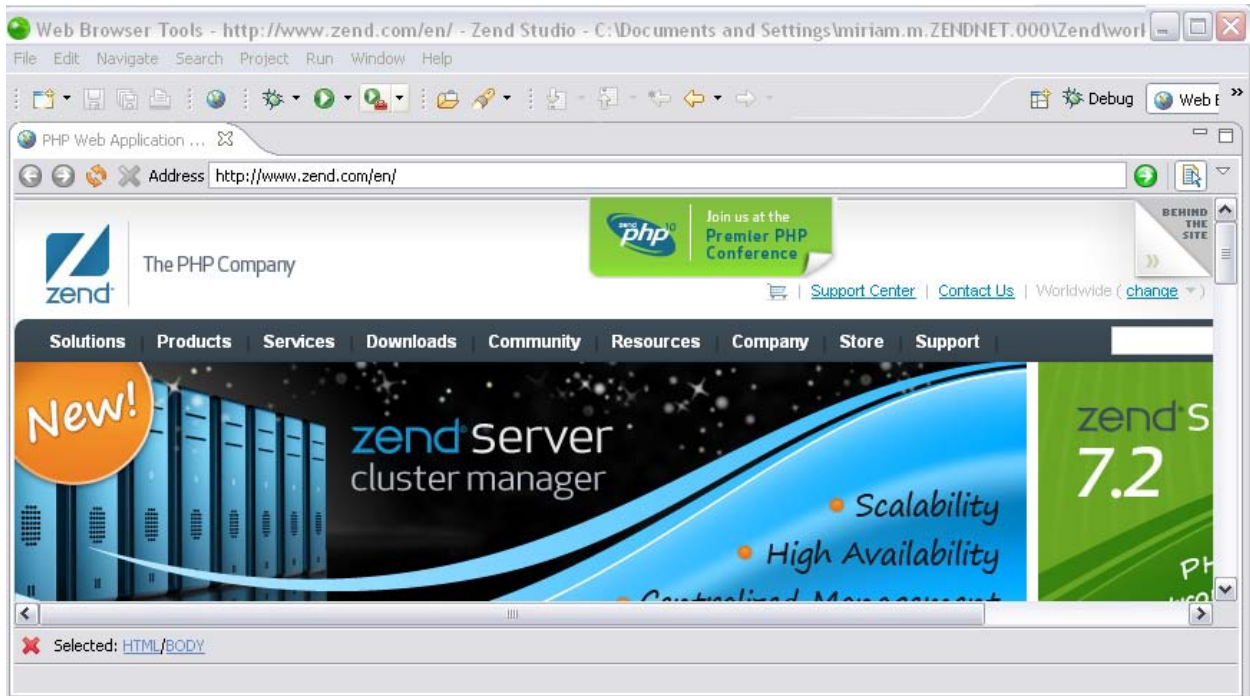
This perspective includes the following views:

- DOM Inspector View
- Browser Console View
- Request Monitor View
- DOM Source View
- CSS View
- JavaScript View
- DOM Watcher View
- DOM Compare View

This perspective includes the following functionalities:

- The Internal Web Browser

PHP Perspective Menus

Zend Studio's menu bars and toolbars offer a range of useful and easily accessible functionality.

Note:

The options available through the menu and toolbars will vary depending on which perspective is currently active.

To configure the menu options for the active perspective, go to Window menu and select Customize Perspective.

The PHP Perspective will by default display the following menu options:

Menu Option	Description
File	Carries out various functions on active files and folders, as well as organizing current files and creating new items.
	The File menu options are:
	New, Open File, Close, Close All, Save, Save As, Save All, Revert, Move, Rename, Refresh, Convert Line Delimeters To, Print, Switch Workspace, Import, Export, Properties, Last Viewed Files and Exit.
Edit	Normal text editing functionality, as well as features such as tasks and bookmarks which are specifically related to editing code.
	The Edit menu options are:
	Undo Text Change, Redo Text Change, Cut, Copy, Paste, Delete, Select All, Find/Replace, Find Next, Find Previous, Incremental Find Next, Incremental Find Previous, Add Bookmark, Add Task, Show Tooltip Description, Word Completion, Quick Fix and Set Encoding
Source	Organizes your scripts by adding or removing comments and formatting the script to make it more easily viewable.
	The Source menu options are:
	Toggle Comment, Add Block Comment, Remove Block Comment, Format Document and Format Active Elements.
Refactor	Edits names and locations of files and resources while maintaining the links between the files .
	The Refactor menu options are:
	Organize Imports, Rename and Move.

Navigate	Navigates through your scripts in order to find relevant resources, information and text.
	The Navigate menu options are:
	Go Into, Go To, Open Declaration, Open PHP Element, Open Resource, Show In, Next Annotation, Previous Annotation, Last Edit Location, Go to Line, Back and Forward.
Search	Searches for text or PHP elements in your workspace.
	The Search menu options are:
	Search, File and Text.
Project	Carries out different functions on your projects.
	The Project menu options are:
	Open Project, Close Project, Build All, Build Project, Build Working Set, Clean, Build Automatically, Generate PHPDoc.., Encode Project , and Properties.
Run	Gets maximum efficiency and accuracy from your files and projects through analyzing and testing your code using the Debugging , Profiling and Run functionality.
	The Run menu options are:
	Toggle Breakpoint, Toggle Line Breakpoint, Toggle Method Breakpoint, Toggle Watchpoint, Skip All Breakpoints, Remove All Breakpoints, Run History, Run As, Open Run Dialog, Debug History, Debug As, Open Debug Dialog, Run, Debug, Profile, Profile History, Profile As, Profile, External Tools, Debug URL and Profile URL.
Window	Customizes your workspace display.
	The Window menu options are:
	New Window, New Editor, Open Perspective, Show View, Customize Perspective, Save Perspective As, Reset Perspective, Close Perspective , Close All Perspectives, Navigation, Working Sets, Web Browser and Preferences.
Help	Gives access to the most updated information on all aspects of Zend Studio , as well as allowing access to software updates and registration locations so that you can get the most out of the product.
	The Help menu options are:
	Welcome, Help Contents, Search Dynamic Help, Key Assist, Tips and Tricks, Software Updates, Register, Tip of the Day and About Zend Studio.

File Menu

The File Menu allows you to carry out various functions on active files and folders, as well as organizing current files and creating new items.

The options available from the File menu are:

Name	Shortcut	Description
New	Alt+Shift+N	Creates various items and types of files. To see the list of new items that can be created through this menu, see the "New" subtopic. Note: The options available in the New Menu for the current perspective can be configured through the Customize Perspectives option in the Window Menu.
Open File		Opens a previously created file in an editor view. If the file is stored in your workspace's active projects, the file will be displayed in PHP Explorer / Navigator views.
Close	Ctrl+W	Closes the active file's editor. If there are unsaved changes in the file, you will be prompted to save the file before closing.
Close All	Ctrl+Shift+W	Closes all open editor windows. If there are unsaved changes in the file, you will be prompted to save them before closing.
Save	Ctrl+S	Saves changes made to the active file.
Save As		Allows you to specify the file name/location when saving the active file.
Save All	Ctrl+Shift+S	Saves all open files.
Revert		Reverts the state of the active file back to its last saved version.

Move		Moves the currently selected file to a different folder / project. Marking the 'Update References' check box in the Move dialog will apply the refactoring feature which will automatically updates all references to the file in other files. Click Preview to see all changes that will be made as a result of the move. Click OK to implement your changes and move the file. All references to the file will be automatically updated to reflect its new location. For more on moving files using the refactoring feature, see "[Moving Files](#)".
Rename	F2	Renames the currently selected file. Marking the 'Update References' check box in the Move dialog will apply the refactoring feature which will automatically update all references to the file in other files. Click Preview to see all changes that will be made as a result of the rename. Click OK to implement your changes. All references to the file will be automatically updated to reflect the change.
Refresh	F5	Refreshes the Navigator views when external changes have been made.
Convert Line Delimeters To		Selects the preferred line ending style. Choices of Windows, Unix and Mac styles.
Print	Ctrl+P	Prints the active file.

Switch Workspace		Allows you to open an alternate workspace. Using this feature will restart Zend Studio with an alternate workspace displayed in Navigator / PHP Views. This is useful if you want to open files and projects situated in a different location or if you want to save files and projects to a different location.
Import		Imports various types of items into your workspace. For a list of the different types of items you can import, divided into categories, see "Import".
Export		Exports and creates different types of items from your workspace into various locations. For a list of the different items you can export from your workspace, divided into categories, see "Export".
Properties	Alt+Enter	Displays a screen with information on the active file, including it's path, type, location, size and when it was last modified. From here the text file encoding type can also be configured, and whether the file is read-only, archive or derived can be set.
Last viewed files		Lists the last viewed files for easy access.
Exit		Shuts down Zend Studio. If there are unsaved changes in the file, you will be prompted to save them before exiting.

New Submenu

The New submenu is available under File | New from the Menu Bar.

The options available under the New submenu are:

Name	Description
PHP Project	Creates a new PHP project within your workspace, with PHP configuration settings allowing full PHP functionality.
Zend Framework Project	Creates a new Project with Zend Framework's libraries in the Include Path and files to create a basic "Hello, World!" application. For more on Zend Framework, visit the Zend Framework site at http://framework.zend.com For more on using Zend Framework with Zend Studio, see "Zend Framework Integration ".
Project ...	Creates a new project in your workspace.
PHPUnit Test Case	Creates a new PHPUnit Test Case. See PHPUnit Testing for more information.
Task	Creates new repositories using the Task Repository view.
PHPUnit Test Suite	Creates a new PHPUnit Test Suite.. See PHPUnit Testing for more information.
PHP File	Creates a new file with PHP tags. Allows full PHP functionality.
Folder	Creates a new folder within a project. The new folder can be linked to a folder in the files system by clicking on the Advanced button in the new folder creation dialog. Using this option will insert an existing folder into your workspace folder. Any changes made to the files and folders in your workspace will automatically be reflected in the local versions of the files in your file system.
File	Creates a new file resource.
CSS	Inserts a cascading style sheet into a project.
HTML Page	Creates a new HTML file within a project, which allows the utilization of HTML functionality.

XML	Creates a new XML file within a project, which allows the utilization of XML functionality.
Zend Framework Item	Opens a New Zend Framework Item wizard which allows you to launch the Wizards for creating a Zend Controller , Zend Model , Zend Module , Zend View and Zend View Helper .
Class	Inserts a new PHP Class within existing or new files, including the required modifiers, Superclasses, interfaces, method stubs, comments etc.
Interface	Creates a new PHP Interface within existing or new files. The PHP Interface creation wizard allows you to include PHP Doc Blocks in your interface, as well as extending other interfaces.
Remote Folder	Creates a new Remote Folder with a remote system connection
Java Script	Creates a new JavaScript file.
Example	Creates the following example projects in your workspace: ▪ **XML** - Inserts an XML example project into the workspace. ▪ **Zend Framework** - Zend Framework is a high quality open source framework for developing Web Applications and Web Services with PHP. The Zend Framework is a collection of common PHP classes and infrastructure which sits above the PHP layer. It packages classes and code, used for common functions such as connecting to databases and creating PDF's, into one easy-to use application. For more on using the Zend Framework example, see the Zend Framework tutorial. This is contained in a readme.html file within the readme folder of the ZendFrameworkExample project. To view the file, right-click it in PHP Explorer view and select **Open With | Web Browser**. For more information on Zend Framework, see http://framework.zend.com.

Other	Allows access to all other types of items not in the main list.
	To configure which items will be available from the main list, go to **Window menu | Customize Perspective**.
	The options in the list are divided into categories:
	▪ Class
	▪ Folder
	▪ Interface
	▪ Java Project
	▪ Java Project from Existing Ant Buildfile
	▪ Plug-in Project
	▪ **General** - File, Folder, Project, Remote Folder, Unitled Text File
	▪ **Connection Profiles** - Connection Profile, Connection Profile Repository
	▪ **CVS** - CVS Repository Location, Projects from CVS
	▪ **Eclipse Modeling Framework** - EMF Model, EMF Project, Empty EMF Project
	▪ **Example EMF Model Creation Wizards** - Ecore Model, Ecore to Ecore Model, Ecore to XML Model, XSD Model, ZSD to Ecore Model
	▪ **Java** - Annotation, Class, Enum, Interface, Java Project, Java Project from Existing Ant Buildlife, Package, Source Folder, DBUnite Test Case, Scrapbook Page, JUnit Test Case, JUnite Test Suite, Servlet Test Case
	▪ Java - DbUnit - DbUnit Test Case
	▪ Java Emitter Templates - Convert Projects to JET Projects
	▪ JavaServer Faces - JSF Library
	▪ JPA - JPA Project
	▪ **PHP** - PHP Class, PHP File, PHP Interface, PHP Project, Untitled PHP Document, Zend Controller, Zend Framework Project, Zend Model, Zend Module, Zend View
	▪ PHP Unit - PHPUnit Test Case, PHPUnit Test Suite
	▪ Remote System Explorer - Connection
	▪ **SQL Development** - SQL File
	▪ SVN - Projects from SVN, Repository Location
	▪ **Web** - CSS, HTML, JavaScript, Static Web Project
	▪ Web Services – WSDL
	▪ **XML** - DTD, XML, XML Schema
	▪ **Examples** - GEF (Graphical Editing Framework) Plug-ins XML, Zend Framework

669

Import Submenu

The Import submenu is available under <u>File | Import</u> from the Menu Bar.

The options available under the Import submenu are:

Name	Description
General	**Archive File** - Extracts files from the archive file into the workbench. **Existing Projects into Workspace - Projects from the local file system.** **File System - Files from the local file system. Browse to the folder in which the file is sitting and click OK. A list of files within that folder will be displayed to allow you to choose the required ones.** **Preferences** - Import preferences from a preferences file on the local file system into the workbench.
CVS	**Projects from CVS** - Imports projects by connecting to a CVS repository. Once a project has been imported from CVS, changes can be made to projects and files which can then be committed to update the CVS repository. A CVS repository connection needs to be configured before using this function. For more information, see <u>Using CVS</u> .
PHP Profiler	**Profile Session** - Imports a profile session file into the workspace.
Remote Systems	**Remote file system-** Imports resources from a remote file system.
Run/Debug	**Breakpoints -** Imports a brakpoint working set. Launch Configurations - Imports a debug launch configurations file.
SVN	**Projects from SVN -** Imports projects by connecting to an SVN repository. Once a project has been imported from SVN, changes can be made to projects and files which can then be committed to update the SVN repository. An SVN repository connection needs to be configured before using this function.
Team	**Team Project Set** - Imports a description of the repository and version control information for a set of projects. For more on Team Project Sets, see <u>Sharing your workspace setup using Project Sets</u>.
XML	**XML Catalog -** Imports an XML Catalog file.
Zend Server	**Server Preferences -** Imports server preferences from a server preferences (.zsc) file. **Zend Server Event File -** Imports a Zend Server Event File and allows you to recreate an event and <u>open the source of the trace data</u>. See <u>Importing a Zend Server Event File</u> for more information.

Export Submenu

The Export submenu is available under File | Export from the Menu Bar.

The options available under the Export submenu are:

Name	Description
General	**Archive File** - Exports files from the Workbench to an archive file in the local file system. **File System** - Export files from your workspace to your local file system. Preferences - Export preferences from the Workbench. In the Export Preferences wizard, select the preferences to export and the location of the preferences (.epf) file to which you want to export them.
PHP	**PHP Doc** - Creates a PHPDoc from your projects or files. See PHPDocs for more information.
PHP Profiler	**HTML Report** - Create an HTML report from a profile session. To use this feature, a profile session must first be run on a file by going to Run menu	Profile As and selecting the relevant profiling configuration. For more on profiling, see the Profiling topic **Profile Session** - Creates an xml document from your profile session. To use this feature, a profile session must first be run on a file by going to Run menu	Profile As and selecting the relevant profiling configuration. For more on profiling, see the Profiling topic.
Remote Systems	**Remote file system** - Exports resources to a remote file system.
Run/Debug	**Breakpoints** - Exports breakpoints from the workbench to a breakpoint file. A list of available breakpoints will be displayed in the Export Breakpoint wizard. Select the relevant breakpoints and the name and location of the file to which they should be exported. Note: The Breakpoint (.bkpt) file will not appear in Navigator / PHP Explorer views. **Launch Configurations -** Imports a debug launch configurations file.
Tasks	**Task List and Contexts** - Export all Task List data, useful for migrating workspaces**.**
Team	**Team Project Set** - Exports Imports a description of the repository and version control information for a set of projects. For more on Team Project Sets, see Sharing your workspace setup using SetsSharing
XML	**XML Catalog** - Exports an XML Catalog file.

Zend Servers	**Server Preferences** - Imports server preferences from a server preferences (.zsc) file.

Edit Menu

The Edit menu contains normal text editing functionality, as well as features such as tasks and bookmarks which are specifically related to editing code.

The options available from the Edit menu are:

Name	Shortcut	Description
Undo Text Change	Ctrl+Z	Undoes the last text edit in the active file
Redo Text Change	Ctrl+Y	Redoes the last text edit in the active file.
Cut	Ctrl+X	Cuts the selected section of text.
Copy	Ctrl+C	Copies the selected section of text to the clipboard.
Paste	Ctrl+V	Pastes text from the clipboard.
Delete	Delete	Deletes the selected section of text.
Select All	Ctrl+A	Selects all text within a file.
Find / Replace	Ctrl+F	Finds and replaces text within the active file.
Find Next	Ctrl+K	Goes to the next instance of an item selected in the editor.
Find Previous	Ctrl+Shift+K	Goes to the next instance of an item selected in the editor.
Incremental Find Next / Previous	Ctrl+J Ctrl+Shift+J	Finds character strings after / before the cursor within the active file. To use this feature, press Ctrl+J and the first few letters of the required string. The relevant text will be highlighted in the file. While in this mode, the up and down cursor keys can be used to navigate between matches. The search can be cancelled by pressing left, right, Enter or Escape.

Add Bookmark...		Inserts a bookmark into your script.
		Bookmarks are used to easily navigate to specific sections in your scripts.
		You can attach a descriptive name to each Bookmark which can be later seen in a tooltip next to the Bookmark.
		Bookmarks are indicated by a bookmark icon ⬛ in the left margin.
		Open the Bookmark view (Window | Show View | Other |General | Bookmarks) to navigate between existing Bookmarks.
Add Task...		Inserts a task into your script.
		Tasks are used as reminders to the programmer. For maximum effectiveness, tasks should be placed next to the section of code on which the action will be implemented.
		Open the tasks view (Window | Show View | Tasks) to navigate between existing tasks.
Show Tooltip Description		Shows the value of a hover that would appear at the current cursor location. The dialog shown is scrollable and does not shorten descriptions.
Word Completion	Alt+/	Completes a word being typed. Enter the first few letters of the word and press Alt+/ to complete the word. Completes a prefix to a word occurring in all currently open editors or buffers.
Quick Fix	Ctrl+1	Displays possible quick fix options for problems in the Problems view.
		To use this option, first select a problem in the Problems view.
		Note: This option will not always be available.

Source Menu

The Source menu allows you to organize your scripts by adding or removing comments and formatting the script to make it more easily viewable.

The options available from the Source menu are:

Name	Shortcut	Description
Override/Implement Methods		Launches the Override/implement Method Wizard to override/implement methods defined in the selected class's parent or interface. See Overriding / Implementing Methods for more information.
Generate Getters and Setters		Launches the Generate Getters and Setters Wizard to automatically create getter and setter functions for variables within the selected class. See Generating Getters and Setters for more information.
Toggle Comment	Ctrl+/ -or- Ctrl+7	Comments or uncomments a line by adding or removing "//" characters. Comments are used for adding text to your script to explain sections of code. Commented text will not be run as part of your code. To use this feature, select the line and press Ctrl+/. See Commenting Code for more information.
Add Block Comment	Ctrl+Shift+/	Comments a block by adding "/*" and "*/" characters to either side of the code. To use this feature, select the block and press Ctrl+Shift+/. See Commenting Code for more information.
Remove Block Comment	Ctrl+Shift+\	Removes a block comment. To use this feature, place the cursor anywhere within the comment and click Ctrl+Shift+\. See Commenting Code for more information.

Format Document	Ctrl+Shift+F	Auto formats a script to organize it into an easily readable format. To format your code, place your cursor anywhere within the editor view and press Ctrl+Shift+F. Appropriate line breaks and indents will be added. You can configure your auto-formatting options through the Formatter Preferences page , accessible from Window | Preferences | PHP | Formatter. See Formatting Code for more information.
Format Active Elements	Ctrl+I	Only formats selected code. To format active elements, select the required code to format and press Ctrl+I. Appropriate line breaks and indents will be added to the active elements. You can configure your auto-formatting options through the Formatter Preferences page , accessible from Window | Preferences | PHP | Formatter. See Formatting Code for more information.

Refactor Menu

The Refactor menu allows you to edit names and locations of files and resources.

For more on refactoring, see "Using Refactoring ".

The options available from the Refactor menu are:

Name	Shortcut	Description
Move	Alt+Shift+V	Moves a file to a different folder. To move a file, select it from the PHP Explorer view. A Move dialog will open. Select the required folder and click Preview to see all changes that will be made as a result of the Move. Click OK to implement your changes and move the file. All references to the file will be automatically updated to reflect its new location.
Rename	Alt+Shift+R	Renames a file or element. To rename a file, select it from the PHP Explorer view. To rename an element within a file select it from the PHP Explorer view or highlight it in the editor view. A rename dialog will open. Enter the new name and click Preview to see all changes that will be made as a result of the rename. Click OK to implement your changes. All references to the file / element will be automatically updated to reflect the change.
Extract Method	Alt+Shift+M	Extracts methods from selected code.
Extract Local Variable	Alt+Shift+L	Extracts variables from selected code. See Extracting Variables for more information.

Note:

Refactoring options will only be available from within PHP Explorer view and not from Navigator view.
Using the Navigator view's move/rename functions will not update any referenced instances of the file/element.

Navigate Menu

The Navigate menu allows you to navigate through your scripts in order to find information and text.

The options available from the Navigate menu are:

Name	Shortcut	Description
Go Into		Goes into a selected folder so that only that folder's contents will be displayed in the Navigator view. Note: This will not work in PHP Explorer view.
Go To		**Back** Displays the previously displayed hierarchy in the Navigator view.
		Forward **Returns to the display from which the back button was pressed in the Navigator view.**
		Up one level Will go up one level in the hierarchy in the Navigator view.
		Resource Goes to a resource within the files and folders displayed in the Navigator view. Enter the first few letters of the a resource in the Go To Resource dialog, and select the required one from the list. Note: This functionality will not work in the PHP Explorer view.
	Ctrl+Shift+P	**Matching Bracket** Jumps to a bracket's pair. Clicking to the right of a bracket will highlight its matching pair. To jump to the matching bracket, press Ctrl+Shift+P.
Open Declaration	F3	Goes to the declaration of an item selected in the editor.
Open Method		Opens a Method in the workspace. See Opening Types/Methods for more information.
Open Type		Opens a Type in the workspace. See Opening Types/Methods for more information.
Open Type in Hierarchy		Displays a selected Type in a hierarchy. See Viewing Type Hierarchies for more information.

Open Resource...	Ctrl+Shift+R	Opens files within the same project as the active file. An Open Resource dialog will appear. Enter the first few letters of the required file to see a list of matching files. Select the required file and click OK to open it in an editor window.
Show In	Alt+Shift+W	**PHP Explorer -** Displays the current active file in the PHP Explorer view. **Navigator** - Displays the current active file in the Navigator view. **Outline -** Displays an element in the Outline view.
Quick Type Hierachy		Displays a selected Type in a Quick Type Hierarchy. See Viewing Type Hierarchies for more information.
Next / Previous Annotation	Ctrl+. Ctrl+,	Goes to the next / previous annotation in the script. Possible annotations are: Bookmarks, Diff additions, Diff changes, Errors, Info, Search Results, Spelling Errors, Tasks and Warnings. Click the arrow next to the next / previous annotation icon on the toolbar to configure which types of annotations should be included. Possible annotation types are Bookmarks, Diff Additions, Diff Changes, Errors, Info, Search Results, Spelling Errors, Tasks and Warnings.
Show Zend Server Event List		Opens the Zend Server Event List. This is only applicable if a Zend Server has been configured in the PHP Servers Preferences page. See Zend Server Integration for full information on the Zend Server .
Last Edit Location	Ctrl+Q	Jumps to the last location that was edited.
Go to Line...	Ctrl+L	Allows you to go to a specific line in the active editor.
Back / Forward	Alt+Left Alt+Right	Scrolls through last viewed sections in active and previously edited files in the current session.

Search Menu

Allows you to Search for text or PHP elements in your workspace.

The options available from the Search menu are:

Name	Shortcut	Description
Search	Ctrl+H	Opens the PHP Search dialog. PHP Search enables you to locate declarations of PHP Classes, Functions and Constants. For more, see "[Searching for PHP Elements](#)".
File		Opens the File Search dialog. File Search enables you to locate text in all files in your workspace. To run a File Search or Replace: 1. Enter the required text to be searched for. 2. Select the types of files to search in. Click 'Choose' for a full list of file type extensions. 3. Select whether to search in: ■ Workspace - The entire workspace. ■ Selected resources - Select these in PHP Explorer view before opening the Source dialog. ■ Enclosing projects - The projects which the selected resources are in. ■ Working Set - Click 'Choose' to select the required Working Set. 4. To search for the string, click Search. Search results will be displayed in the Search view. To replace the string, click Replace. The Replace dialog will open. **Note:** By default, the PHP Search dialog will be tabbed with the File Search dialog. To make the PHP Search dialog unavailable, click Customize within the File Search dialog and unmark the PHP Search dialog option.
Text	Ctrl+Alt+G	**Workspace** - Searches the Workspace for text selected in the editor. **Project -** Searches the current active project for text selected in the editor. **File -** Searches the current active file for text selected in the editor. **Working Set -** Searches the current active Working Set for text selected in the editor.

Project Menu

The Project menu allows you to carry out different functions on your projects, including open, close, build and encode.

The options available from the Project menu are:

Name	Shortcut	Description
Open Project		Opens the currently selected project. This option is enabled when a closed project is selected.
Close Project		Closes the currently selected project. Closing a project does not cause it to be deleted from the file system. A closed project will still be displayed in PHP Explorer view with a closed project icon , but its resources are no longer accessible from within the Workbench. Closing projects takes up less memory and speeds up the build process.
Build All	Ctrl+B	This command manually invokes an incremental build on all projects in the Workbench. This is only available if automatic build is not selected (see below).
Build Project		This command manually invokes an incremental build on any resources in the currently selected project that have been affected since the last build. This is only available if automatic build is not selected (see below).
Build Working Set		This command manually invokes an incremental build on any resources in a working set that have been modified since the last build. This is only available if automatic build is not selected (see below).
Clean...		Invokes a clean build. This will discard all previous build results.

Build Automatically		Performs an incremental build whenever resources are saved. Selecting this option will disable all other manual build options. **Note:** The build function can be configured by selecting General | Workspace from the preferences dialog (Windows | Preferences).
Zend Tool	Ctrl+2	Opens the Zend Tool Floating WIndow for executing Zend_Tool commands. See Using the Zend Tool Console more information.
Generate PHPDoc		Creates a PHPDoc from your projects or files. A PHPDoc is an online document, organized in a book format, allowing easy viewing of all elements within your code.
Encode Project		Encodes the selected project using Zend Guard. This option will be enabled once Zend Guard is installed and configured in Zend Studio. Zend Guard integration can be configured through the Zend Guard preferences dialog, accessed from Window | Preferences | PHP | Zend Guard.
Properties		Opens the project's properties dialog which allows you to view and configure various settings for the project, including resource information, Builders, Code Analyzer Properties, Formatter, Includes Mapping, PHP Debug, PHP Include Path, PHP Interpreter, PHP Java Bridge, PHP Task Tags, Project References, Run/Debug Settings, Task Tags and Validation. See PHP Project Properties for more information on all the properties available for a PHP project.

Run Menu

The Run menu allows you to get maximum efficiency and accuracy from your files and projects through analyzing and testing your code using the Debugging , Profiling and Run functions.

Running a file or application will display the output in the Browser and Debug Output views, as well as displaying any error or warning messages in the console view.

Debugging a file or application allows you to view the output and any error notices, as well as information about various elements, at various stages while the file is run. For more information on Debugging, see Debugging Files and Applications.

Profiling a file or application allows you to detect bottlenecks in scripts by locating problematic sections of code.

For more information on Profiling, see Profiling Files and Applications .

The options available from the Run menu are:

Name	Shortcut	Description
Toggle Breakpoint	Ctrl+Shift+B	Adds / removes breakpoints from your script. Breakpoints are used to stop the debugging process at certain key places throughout your code.
Toggle Line Breakpoint		Adds / removes line breakpoints from your script.
Toggle Method Breakpoint		Adds / removes method breakpoints from your script. Method breakpoints are used to add conditions to breakpoints.
Toggle Watchpoint		Adds / removes a field watchpoint for the current selected variable in the Expressions View.
Skip All Breakpoints		Temporarily removes all breakpoints from your script so that the debugging process will not stop at them. Select this option again to return all breakpoints to the script.
Remove All Breakpoints		Removes all breakpoints from the current active file.
Run History		Displays and allows access to a list of previously launched Run configurations.
Run As		Lets you choose from running: on the server, as a PHP Script, or as a PHP WebPage.

Run Configurations...		Launches the Run dialog to create / edit Run configurations.
Debug History		Displays a list of Debug configurations so that they can be used for debugging.
Debug As		Lets you choose from debugging: on the server, as a PHP Script, or as a PHP WebPage. For more on these options, see Debugging.
Debug Configurations...		Launches the Debug dialog to create /edit debugging configurations.
Run	Ctrl+F11	Launches the last Run configuration run.
Debug	F11	Launches the last Debug session run.
Profile		Launches the last Profile session run.
Profile History		Displays a list of Profile configurations so that they can be used for profiling.
Profile As		Lets you choose from profiling: on the server, as a PHP Script, or as a PHP WebPage. For more on these options, see "Profiling".
Profile Configurations...		Launches the Profile dialog to create / edit Profiling configurations.
External Tools		**Run As** If applicable, allows you to run the file using External tools. **Open External Tools Dialog** Opens the configuration dialog for running a file using external tools. **Organize Favorites** Opens a dialog allowing you to organize your external tools. See the External Tools topic in the Workbench User Guide for more information.
Debug URL		Launches a Debug session for a specified URL.
Profile URL		Launches a Profile session for a specified URL.

Navigation Submenu

The Navigation submenu is available under Window | Navigation from the Menu Bar.

The options available under the Navigation submenu are:

Name	Shortcut	Description
Show System Menu	**Alt + -**	Shows the system menu for the active view. This contains options on how the view is displayed (Fast View, Detached, Restore, Move, Size, Minimize, Maximize, and Close.)
Show View Menu	Ctrl+F10	Shows the active view's menus, containing functionality for each view.
Quick Access	Ctrl+3	Gives you quick access to a range of Eclipse options.
Maximize / Minimize Active View or Editor	Ctrl+M	Toggles between full screen and minimal views.
Activate Editor	F12	Switches to the editor view.
Next / Previous Editor	Ctrl+F6 Ctrl+Shift+F6	Switches to the next/previous open editor. Hold down the Ctrl key and press F6 to scroll between the editors. This command is similar to the Alt+Tab functionality in Windows.
Switch to Editor	Ctrl+Shift+E	Opens a dialog displaying all the open editors. Allows you to choose which editors to open, close or save.
Next / Previous View	Ctrl+F7 / Ctrl+Shift+F7	Switches to the next open view. Hold down the Ctrl key and press F7 / Shift + 7 to scroll between the views. This command is similar to the Alt+Tab functionality on Windows.
Next Perspective / Previous Perspective	Ctrl+F8 / Ctrl+Shift+F8	Switches to the last / next open Perspective.

Window Menu

The Window menu allows you to customize your workbench display.

The options available from the Window menu are:

Name	Description
New Window	Opens the workspace in a new window. Clicking on the window's X icon will close only that window and not the whole workspace.
New Editor	Opens the active editor in a new window.
Open Perspective	Opens a selected perspective, containing a selection of views. See the Workbench User Guide for more on <u>Perspectives</u>. **Note:** Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (<u>http://help.eclipse.org/helios/index.jsp</u>).
Show View	Displays a selected view. See the Workbench User Guide for more on <u>Views</u>. **Note:** Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (<u>http://help.eclipse.org/helios/index.jsp</u>).
Customize Perspective	Configures settings for the active perspective, including settings for: - The quickly accessed settings on the New, Open Perspective and Show View submenus. - Which options appear in the menu and toolbar.
Save Perspective As	After configuring the perspective, you can select to save it under a different name for future use.
Reset Perspective	Resets the perspective to its default view, menu bar and toolbar settings.
Close Perspective	Closes the active perspective and reverts back to the last viewed perspective.
Close All Perspectives	Closes all perspectives. No views or functionality will be available. Click the open perspective icon to open a perspective.

Navigation	Allows quick access and Navigation between views and perspectives. For a list of commands available from this menu option, see the "Navigation" subtopic.
Working Sets	Opens the working sets dialog, allowing you to edit or create Working Sets See PHP Working Sets for more information.
Preferences	Opens the preferences dialog to configure all aspects of the workspace. See the PHP Preferences section for on PHP Preferences.

Help Menu

The Help menu allows access to the most updated information on all aspects of Zend Studio, as well as allowing access to software updates and registration locations so that you can get the most out of the product.

The options available from the Help menu are:

Name	Shortcut	Description
Welcome		Opens the welcome page containing the latest news from Zend as well as access to a range of tutorials for using Zend Studio functionality.
Help Contents		Opens the Online Help page.
Search		Opens a Search view for searching the online Help manual.
Dynamic Help		Opens Help Topics relevant to the current action in the workspace.
Key Assist...	Ctrl+Shift+L	Opens a list of all shortcut keys for Zend Studio functionality.
Tips and Tricks...		Opens the Tips and Tricks Help page for the Eclipse Platform and Eclipse Plug-In Development Environment.
Cheat Sheets...		Allows you to open a cheat sheet with quick explanations on how to carry out a variety of functions.
Register		Allows you to enter your Zend Studio license key to enable Zend Studio functionality. For more on purchasing a Zend Studio license, see the Zend Studio site at http://www.zend.com/en/products/studio. **Note**: System i users can register for a free i5 Edition of Zend Studio. See i5 Edition Extras for more information.
Zend Studio Support Center		Redirects you to the Zend Online Support Center.
Tip of the Day		Opens the Tip of the Day window.
Check for Updates		Searches for available software updates.
Install New Software...		Installs available software updates.
Get a License		Connects you to the Zend Studio site at http://www.zend.com/en/products/studio to learn more about purchasing a license and enabling Zend Studio functionality.
Protect your PHP Code!		Directs you to the Zend Guard site at http://www.zend.com/en/products/guard. Zend Guard protects your applications from reverse engineering and unauthorized customization by providing encoding and obfuscation. See Zend Guard Integration for more information.

Speed up your PHP Site!		Directs you to the Zend Server site at http://www.zend.com/en/products/server.Zend Server is the only PHP Web application server that supports the enterprise reliability and comprehensive performance features organizations need for business-critical applications. See Zend Server Integration for more information.
About Zend Studio		Opens the About dialog, displaying information about the current version of Zend Studio.

PHP Perspective Main Toolbar

The PHP Perspective's Main Toolbar offers shortcuts to frequently used functionality:

Shortcut Icon	Shortcut Keys	Name	Description
		New	Opens the New Wizard dialog. Clicking the arrow next to the icon lets you select to create a new Zend Framework Project, PHP Project, Project, Zend Module, Zend Controller, Zend Model, Zend View, PHPUnit Test Case, PHPUnit Test Suite, PHP Class, PHP Interface, Remote Folder, PHP File, Folder, CSS, HTML, XML, Example Project or Other resource.
	Ctrl+S	**Save**	Saves the active file.
	Ctrl+P	**Print**	Prints the active file.
	Ctrl+Alt+N	**New Untitled PHP Document**	Creates a new untitled PHP Document.
		Create New SQL Connection	Opens the New SQL Connection dialog.
		Generate PHPDoc	Opens the PHPDoc Generation wizard.
		Zend Server Event List	Displays the Event List for the configured Zend Servers. Clicking the arrow next to the icon allows you to select the required Zend Server. See Zend Server Integration for more information.

		Debug	Clicking the Debug Button executes the last run configuration. Clicking the arrow next to the icon gives access to the following options: • Debug a previously executed launch configuration. • Debug As... - Debug the active file as a PHP Script, PHP Web Page, or, when applicable, a PHPUnit Test. • Debug Configurations - Opens the Debug dialog. • Organize Favorites - Allows you to select which launch configurations should be added to your Favorites list. Your Favorite launches will be listed first in the launch configuration list.

		Run	Clicking the Run Button executes the last run configuration. Clicking the arrow next to the icon gives access to the following options: - Run a previously executed launch configuration. - Run As... - Run the active file as a PHP Script, PHP Web Page, or, when applicable, a PHPUnit Test. - Run Configurations - Opens the Run dialog. - Organize Favorites - Allows you to select which launch configurations should be added to your Favorites list. Your Favorite launches will be listed first in the launch configuration list.

| | | **Profile** | Clicking the Profile Button executes the last run configuration. Clicking the arrow next to the icon gives access to the following options:

▪ Profile a previously executed launch configuration.
▪ Profile As... - Profiles the active file as a PHP Script, PHP Web Page, or, when applicable, a PHPUnit Test.
▪ Profile Configurations - Opens the Debug dialog.
▪ Organize Favorites - Allows you to select which launch configurations should be added to your Favorites list. Your Favorite launches will be listed first in the launch configuration list. |

		External Tools	Clicking the External Tools Button opens the External Tools Configuration dialog. Clicking the arrow next to the icon gives access to the following options:
			- Run As - If applicable, allows you to run the file using External tools. - External Tools Configurations - Opens the configuration dialog for running a file using external tools. - Organize Favorites - Opens a dialog allowing you to organize your external tools. See the External Tools topic in the Workbench User Guide for information on configuring your Builder. **Note:** Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).
		Debug URL	Launches the Debug URL dialog.
		Profile URL	Launches the Profile URL dialog.
	Ctrl+H	Search	Launches the Search dialog.

	Ctrl+. Ctrl+,	**Next/Previous Annotation**	Navigates to the next / previous annotation in the script. Possible annotations are: Bookmarks, Diff additions, Diff changes, Errors, Info, Search Results, Spelling Errors, Tasks and Warnings. Click the arrow next to the next / previous annotation icon on the toolbar to configure which types of annotations should be included.
	Ctrl+Q	**Last Edit Location**	Jumps to the last location that was edited.
	Alt+Left Alt+Rigt	**Back/forward to last edited file**	Scrolls through the previous/next edited locations in all edited file in the current session.
		Enable Tunneling	Creates a Tunneling connection. Click the arrow next to the Tunneling icon to select the server for which you want to enable a Tunneling connection.
		Encode Project with Zend Guard	Encodes your project with Zend Guard. This will only be available when Zend Guard integration is enabled through the [Zend Guard Preferences](#) page.

PHP Preferences

Below is a list of the different PHP preferences which can be configured.

To access this menu, go to Window menu and select **Preferences | PHP**.

Preference	Description
PHP	Configures the display in PHP Explorer view.
Appearance	Configure the display of elements in Outline views.
Code Coverage	Preview code in the Code Coverage view with the current color and font settings.
Code Gallery	Define and add code galleries.
Code Refactor	Configure code re-factoring preferences.
Code Style	Expand the list to access the Code Templates and Formatter Preferences.
Code Templates	Configure generated code and comments.
Formatter	Set preferences for the auto-formatter.
Debug	Configure your debug preferences.
Installed Debuggers	Configure the settings for your installed debuggers.
Step Filtering	Configure the Step Filtering options when debugging.
Workbench Options	Configure the workspace's behavior when a debug session is launched.
Editor	Configure Smart Caret positioning.
Content Assist	Configure Code Assist preferences.
Code Folding	Configure the elements which will be folded by default.
Hovers	Configure the settings and shortcuts for the hover functionality.
Mark Occurrences	Configured the Mark Occurences settings.
Save Actions	Configures the behavior of whitespaces after saving.
Syntax Coloring	Set the font color for different elements.
Task Tags	Add and edit tasks tags.
Templates	Define and create templates for content assist.
Typing	Configure which items should be automatically completed.
Zend Framework	Configure display settings for the Zend Tool Floating Window and Console.
New Project Layout	Configure the default project layout for new PHP projects.
PHP Executables	Add, remove or edit PHP executables definitions.
PHP Interpreter	Select the PHP version.
PHP Libraries	Add, remove, or edit PHP user libraries.
PHP Manual	Add PHP Manual sites.
PHP Servers	Add and edit PHP servers.

PHPUnit	Configure PHPUnit's Library path and port.
Semantic Analysis	Configure the severity for error and warning messages in different cases.
Zend Guard	Set Zend Guard's location.

PHP Preferences Page

The PHP Preferences page allows you to configure the hierarchy display in PHP Explorer view and set double-click behavior.

The PHP Preferences page is accessed from **Window | Preferences | PHP**.
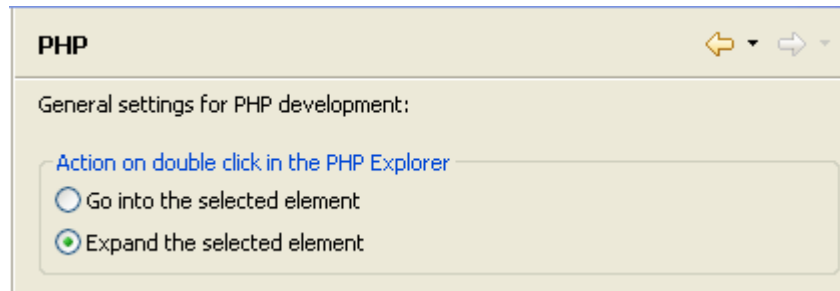
PHP Preferences page

Configuring PHP Preferences

To configure PHP Preferences:

1. Select the required option:
   - Go into the selected element - PHP Explorer view will display only a folder's contents once it is double-clicked.
   - Expand the selected element - PHP Explorer view will expand a folder once it is double-clicked, leaving the other projects and folders visible in a tree diagram.
2. Click Apply to apply your settings.

Appearance Preferences

The Appearance preferences page allows you to select whether to show PHP Elements' method return types in the Outline and PHP Project Outline views. These will be displayed in brackets next to the element.

PHP Project Outline view without the method return types display

PHP Project Outline view with the method return types display

The Appearance Preferences page is accessed from Window | Preferences | PHP | Appearance.
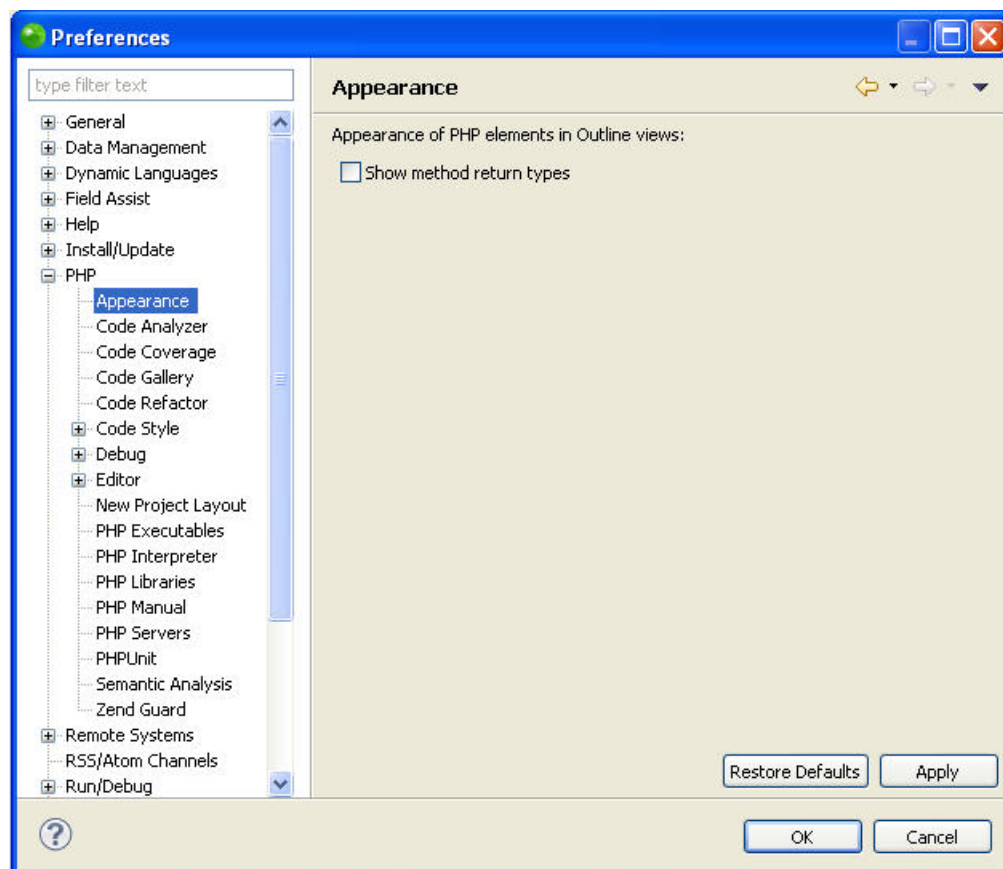
Appearance Preferences page

Displaying Element's Return Type Methods

To display element's return type methods:

1. Mark the 'Show methods return types checkbox'.

2. Click **Apply** to apply your settings.

Code Coverage Preferences

The Code Coverage Preferences page displays a preview of code in Code Coverage view, with the current color and font settings. Code coverage views are displayed when using various features such as profiling, unit testing and debugging to show which lines of code have been covered by this functionality.

The Code Coverage Preferences page is accessed from **Window | Preferences | PHP | Code Coverage**.
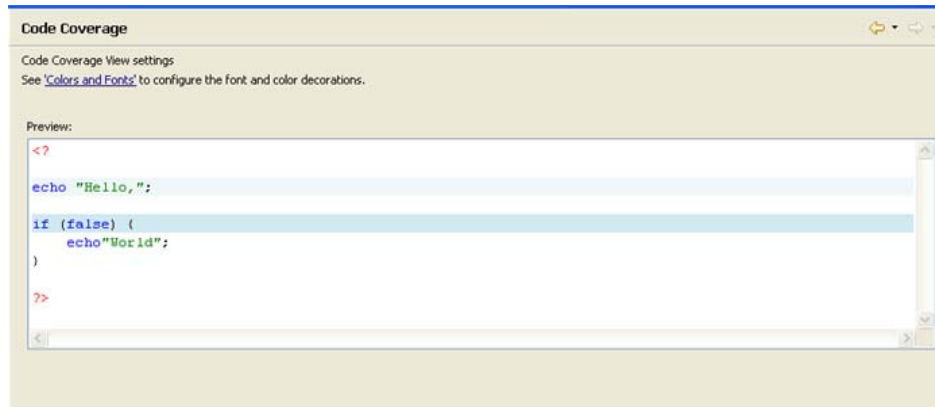
Configuring Code Coverage Colors and Fonts

To configure code coverage colors and fonts:

1. Click the 'Colors and Fonts' link.

    The Colors and Fonts preferences page will be displayed, with the PHP Debug category open.
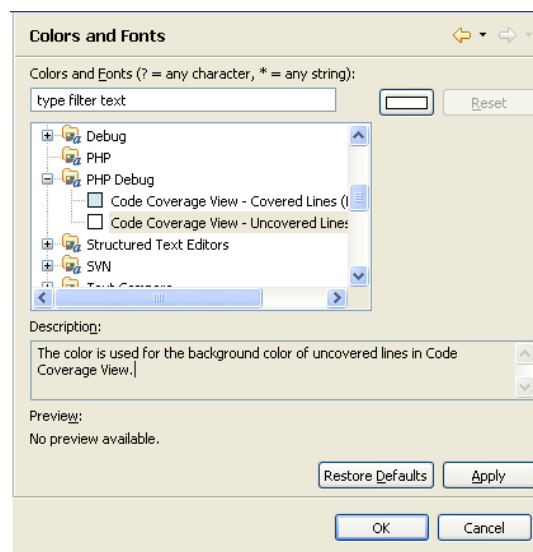
2. Select the required background color for covered lines and uncovered lines by selecting the relevant option and clicking the required color in the color selection box (top-right corner).

3. Click **Apply**.

4. The changes will be displayed in the Code Coverage preview page.

More color and font options can be configured by opening the preferences page (**Window | Preferences**) and selecting:

- General | Appearance | Colors and Fonts
- General | Editors | Text Editors | Annotation
- General | Editors | Text Editors | Quick Diff
- Run / Debug
- Run / Debug | Console
- Team | CVS | Console

Code Gallery Preferences

The Code Gallery preferences page allows you to add and edit code galleries.

Code Galleries are pre-defined code snippets sites. Code snippets can be used to easily insert pre-defined sections of code into your script.

The Code Gallery Preferences page is accessed from **Window | Preferences | PHP | Code Gallery** .
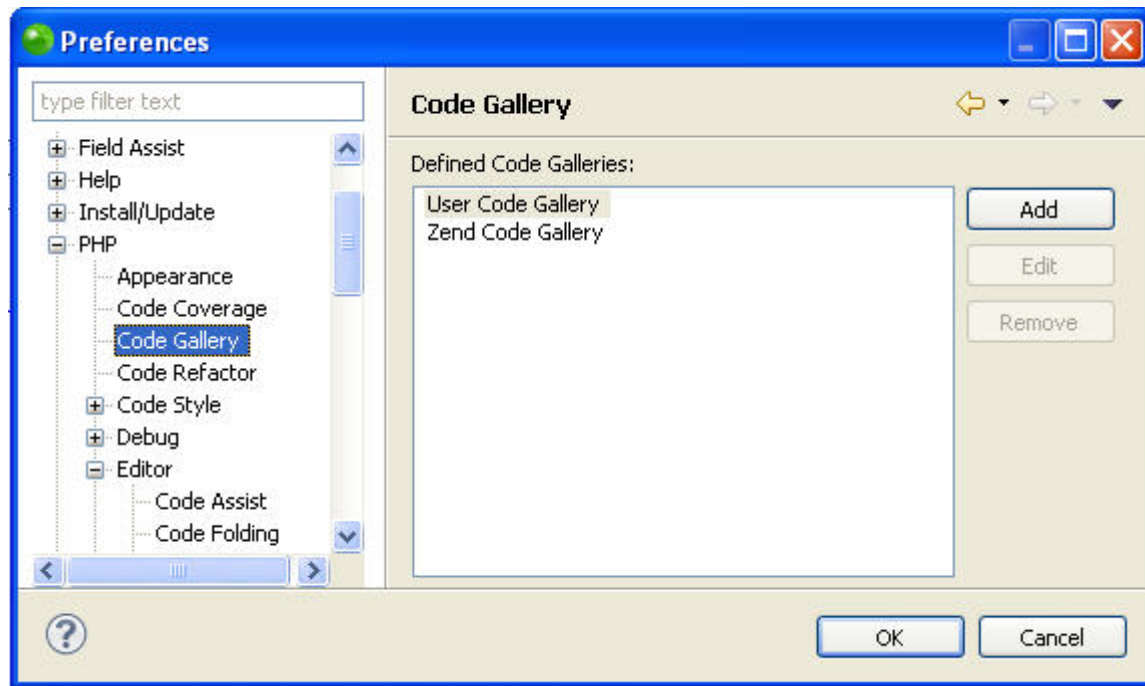
Code Gallery preferences page

Code snippets can be accessed from the Code Gallery view (**Window | Show View | Other | PHP Tools | Code Gallery**).

Adding a Code Gallery Site

To add a code gallery site:

1. Click **Add**.
2. Enter the URL of the required code gallery and click **OK**.

The new site will be added to the list and will be available from the Code Gallery view.

Editing a Code Gallery Site

To edit a code gallery site:

1. Select the required gallery from the list.
2. Click **Edit**.
3. Change the required information and click **OK**.

Removing a Code Gallery Site

To remove a code gallery site:

1. Select the required gallery from the list.
2. Click **Remove**.
3. Click **OK** to apply your settings.

The gallery will be removed from the list and will not longer be accessible from the Code Gallery view.

Note:
The User Code Gallery cannot be edited or removed.

Code Refactor Preferences

The Code Refactor Preferences page allows you to enable the in-place refactoring option. This option allows you to rename elements using the refactoring feature from within the editor rather than through the refactor dialog.

See the Refactoring topic for more information on refactoring or Renaming Elements

The Code Refactor Preferences page is accessed from **Window | Preferences | PHP | Code Refactor**.
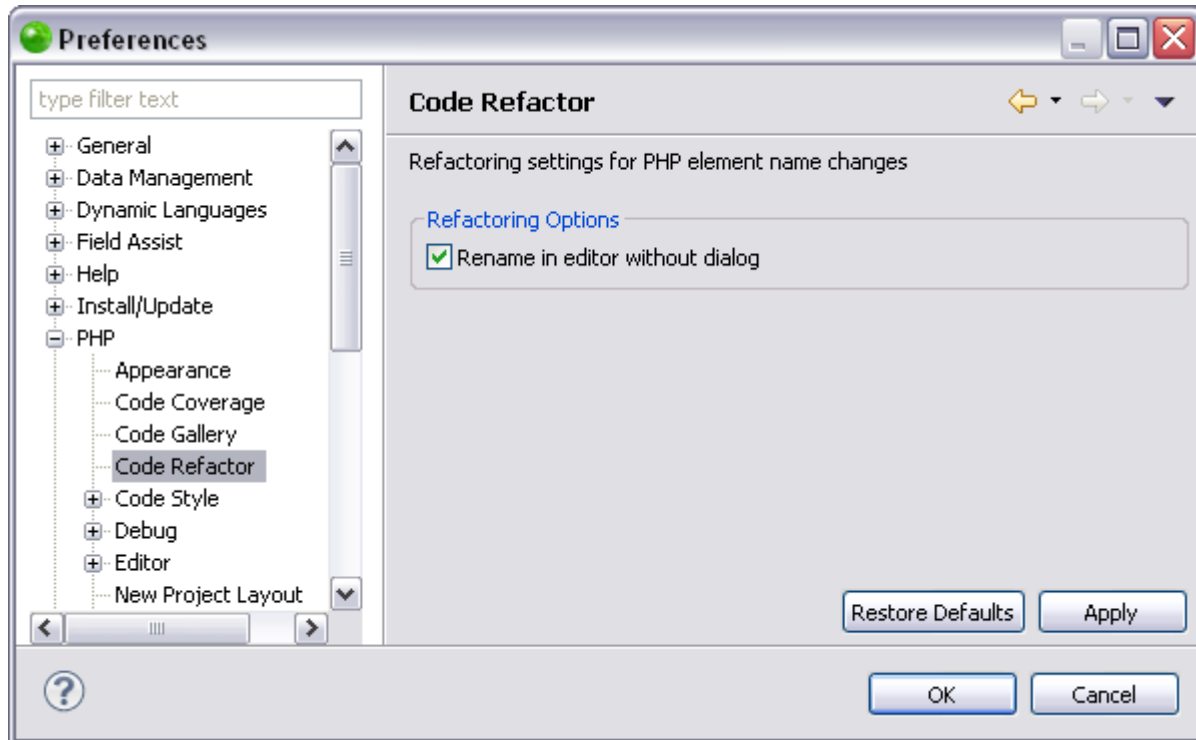
Configuring Your Code Refactor Settings

To configure your Code Refactor settings:
1. Mark the 'Rename in editor without dialog' checkbox to enable you to rename elements in the editor rather than through the refactor dialog.
2. Click **Apply** to apply your changes.

Code Style Preferences

Code Templates Preferences

The Code Templates Preferences page allows you to configure the code and comments that are automatically created for different types of elements.

The Code Templates Preferences page is accessed from **Window | Preferences | PHP | Code Style | Code Templates**.
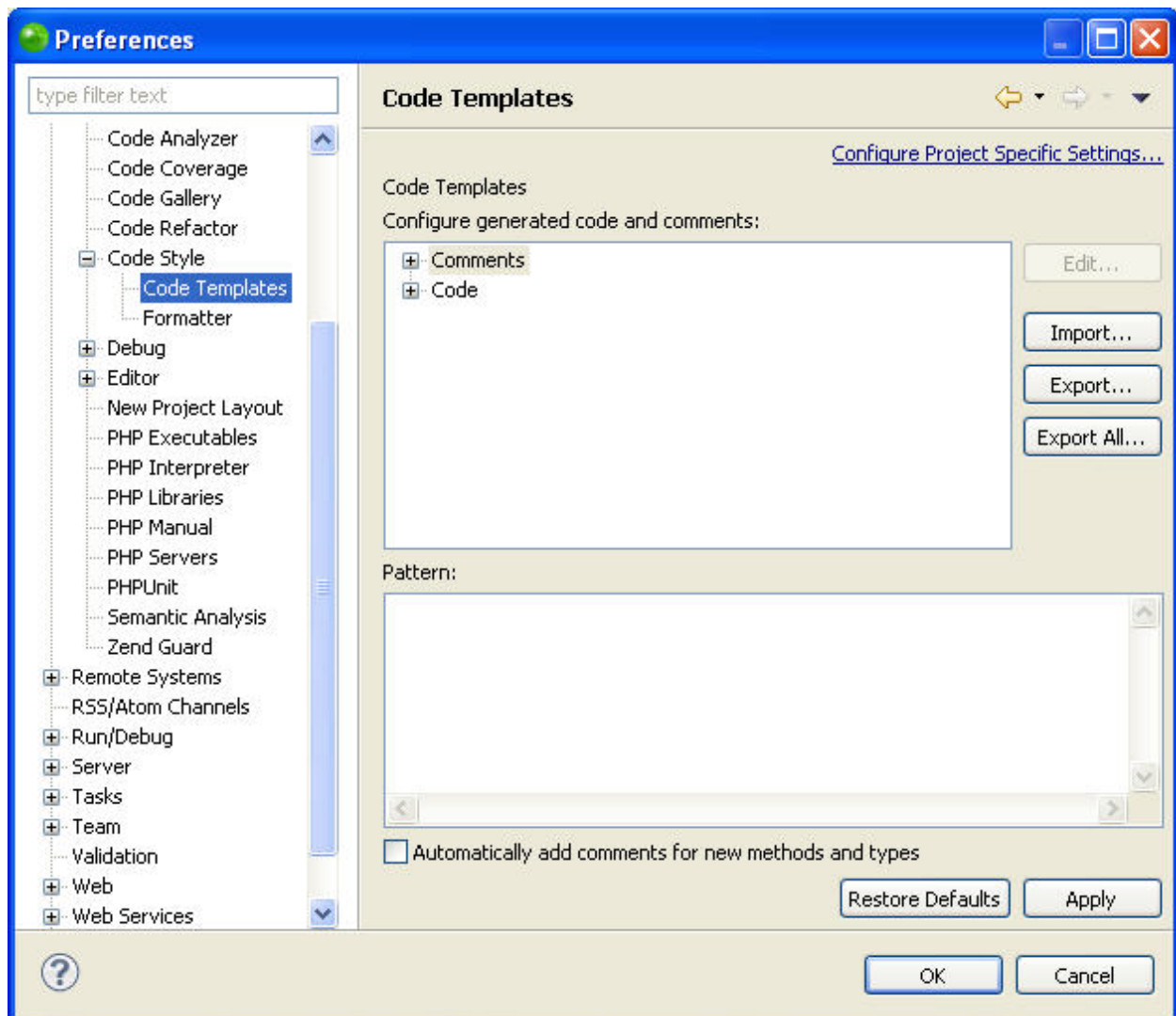
Code Templates Preferences page

Editing the Pattern for a Comment or Code Element

To edit the pattern for a comment or code element:

1. Expand the list and select the required element.

2. Click **Edit**.

   The "Edit Template" dialog is displayed.

3. Edit the pattern as required and click **OK**.

4. If required, mark the 'Automatically add comments for new methods and types' box for comments to be automatically generated when new methods and types are created.

5. Click **Apply** to apply your changes.

Exporting and Importing Code Templates

Zend Studio enables you to export and import code templates, which are created within XML files in the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<templates>
<template
autoinsert="true" context="php" deleted="false" description="description"
enabled="true"
id="org.eclipse.php.ui.editor.templates.php.codetemplates.xxxcomment"
name="name">
/** * ${tags} */
</template>
</templates>
```

Importing a Code Template

To import a code template:

1. Click Import to open the "Import Templates" browser.

2. Select the relevant XML file containing the template information.

3. Click **Open**.

The templates contained in the template.xml file will be imported into the list of Templates.

Exporting a Code Template

To export a code template:

1. Select the template(s) for export from the Template list.

2. Click **Export** to open the "Export Templates" dialog.

3. Select the location to save the XML file to.

4. Click **Save**.

An XML file will be created with the template information.

Exporting All Code Templates

To export all code templates:

1. Click **Export All**.

2. Select the location to save the XML file to.

3. Click **Save**.

An XML file will be created with the template information.

Note:
If you selected more than one template to export, all of them will be present in the exported XML file.
Each of the original Templates is bounded by: < template > </template>

Formatter Preferences

Zend Studio can auto-format scripts to organize them into an easily readable format.
The Formatter Preferences page allows you to customize the way it is formatted.

The Formatter Preferences Preferences page is accessed from **Window | Preferences | PHP | Code Style | Formatter**.
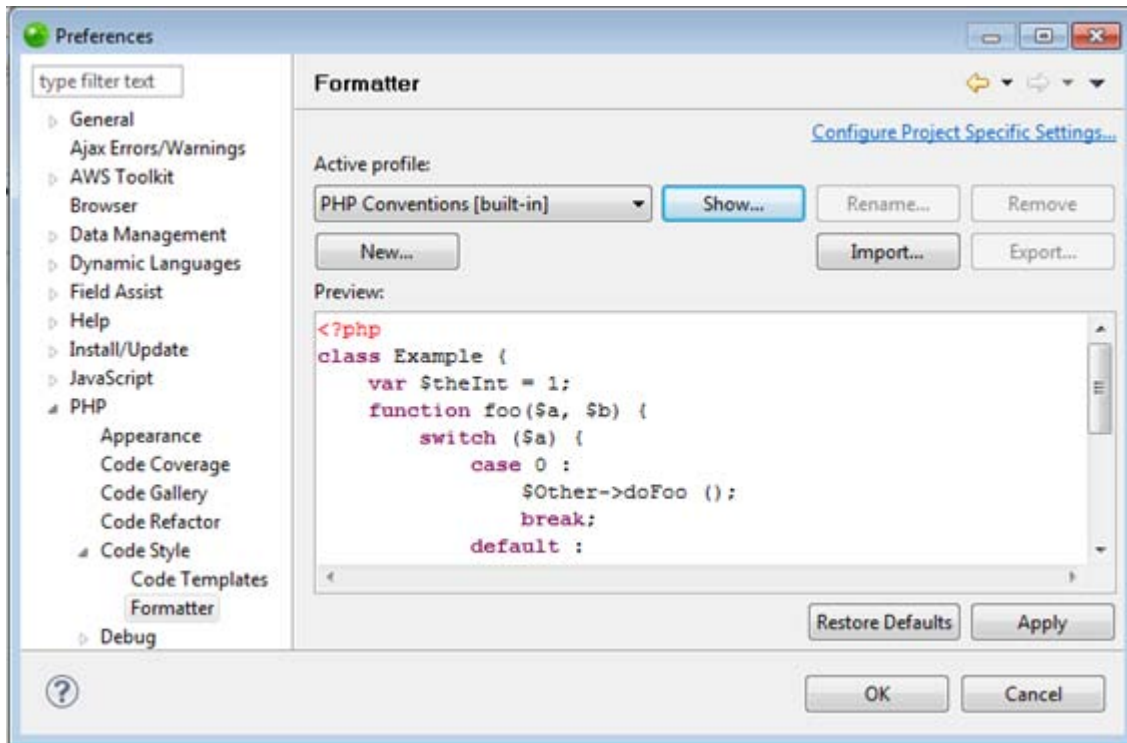
Creating Your Own Set of Configuration Settings

The default formatting settings are based on the PHP Conventions settings.
Click Show to see these settings.

> **To create your own set of configuration settings:**
> 1. Click **New**.
> 2. Enter a name for you profile.
> 3. Select which profile to base your new profile on. (This will duplicate these settings and allow you to edit them).
> 4. Ensure that the 'Open the edit dialog now' checkbox is marked and click **OK**.

5. The Edit dialog will open with the following tabbed option screens:

- Indentation
- Braces
- White Space
- Blank Lines
- New Lines
- Control Statements
- Line Wrapping

These are described in more detail below.

6. Once you have set the required options, click **OK**.

The new configuration will be added to the list.

The Options Available in the Formatter Tabs

These options are available by clicking **Show** in the Formatter Preferences page.

Indentation

The Indentation tab allows you to configure the indentation size and select which elements should be indented.
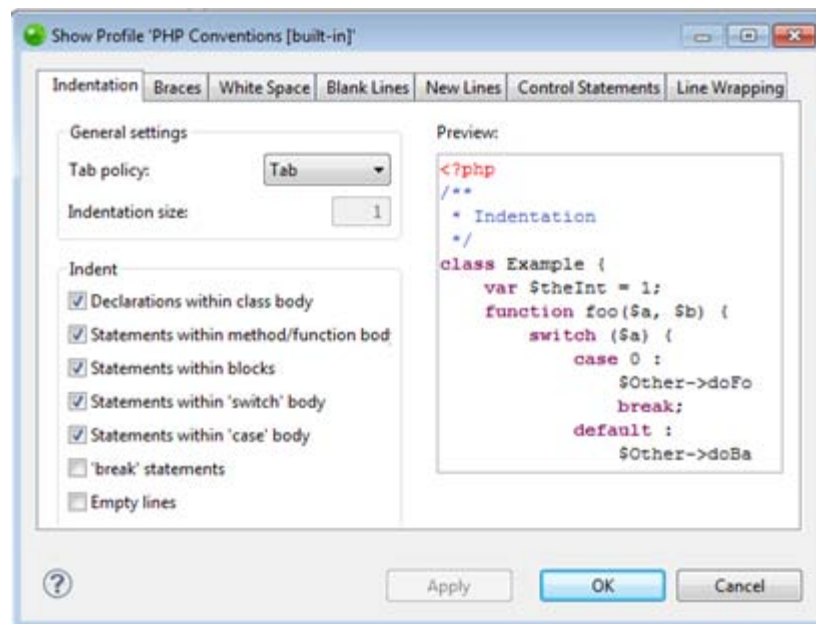
The Indentation tab contains the following options:

General Settings

- Tab Policy - Select Tab or Spaces from the drop-down list to set the indentation size for a tab.
- Indentation size - If you select spaces, enter the number of spaces to be created.

Indent

Select the elements to be indented by marking the relevant checkbox from the following options:

- Declarations within class body
- Statements within method/function body
- Statements within blocks
- Statements within 'switch' body
- Statements within 'case' body
- 'break statements
- Empty lines

Braces

The Braces tab allows you to configure brace positions.
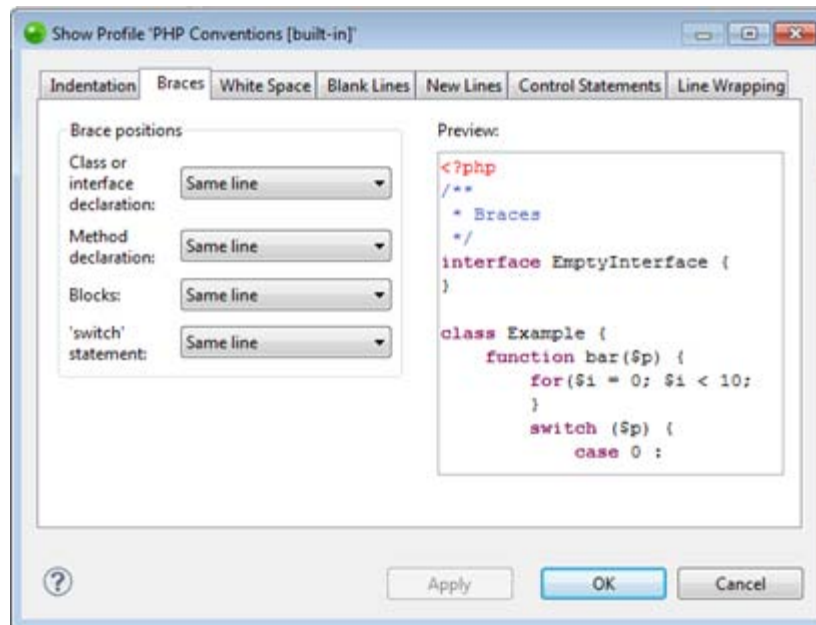
Choose the brace positions (Same line, Next line or Next line indented) for the following:

- Class or interface declaration
- Method declaration
- Blocks
- 'switch' statement

White Space

The White Space tab allows you to configure where spaces should be entered for declarations, control statements, expressions and arrays.
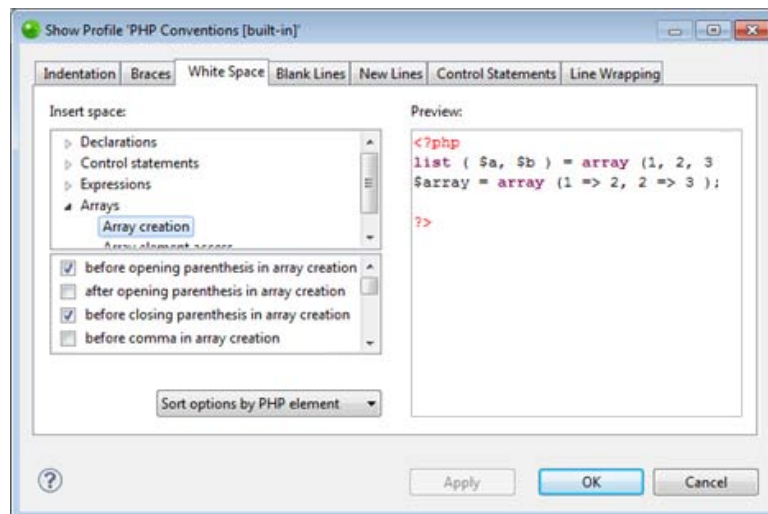
Expand each category by clicking on the + sign to configure which items are applied to each instruction.

Standing on an item will display a list of possible syntax conditions in which white spaces will be inserted. Select the required conditions by marking the relevant checkboxes.

Choosing 'sort options by Syntax element' from the drop down list will sort the list of by syntax conditions rather than by element.

Blank Lines

The Blank Lines tab allows you to set the number of blank lines to be created in various conditions.
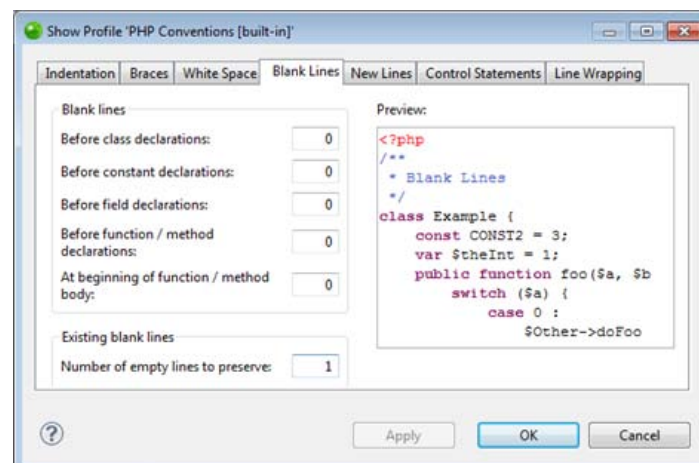
Enter the number of blank lines (between 0-32) to be created in the following conditions:

- Before class declarations
- Before constant declarations
- Before field declarations
- Before function/method declarations
- At beginning of function / method / body

Existing Blank Lines

- Number of empty lines to preserve

New Lines

The New Lines tab allows you to select when new lines will be created.
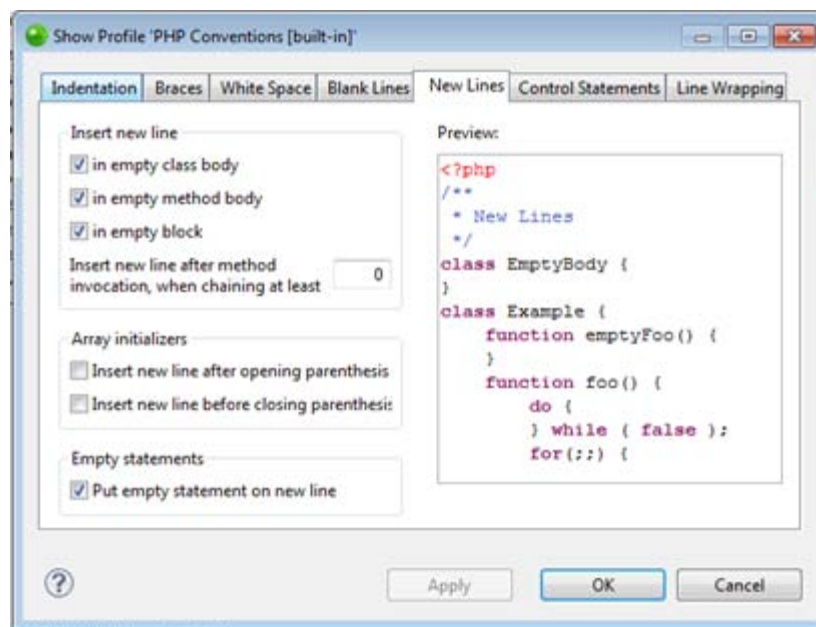
The Options in the New Lines tab

Insert New Line

Select whether to insert a new line in the following conditions:

- In empty class body
- In empty method body
- In empty block
- Insert new line after method invocation, when chaining at least - When chaining at least the number of methods you specify here, a new line will be inserted after the method invocation.

Array Initializers

Select whether to:

- Insert new line after opening parenthesis of array initializer
- Insert new line before closing parenthesis of array initializer

Empty statements

Select whether to put empty statements on a new line.

Control Statements

The Control Statements tab allows you to configure the line formatting for Control Statements.
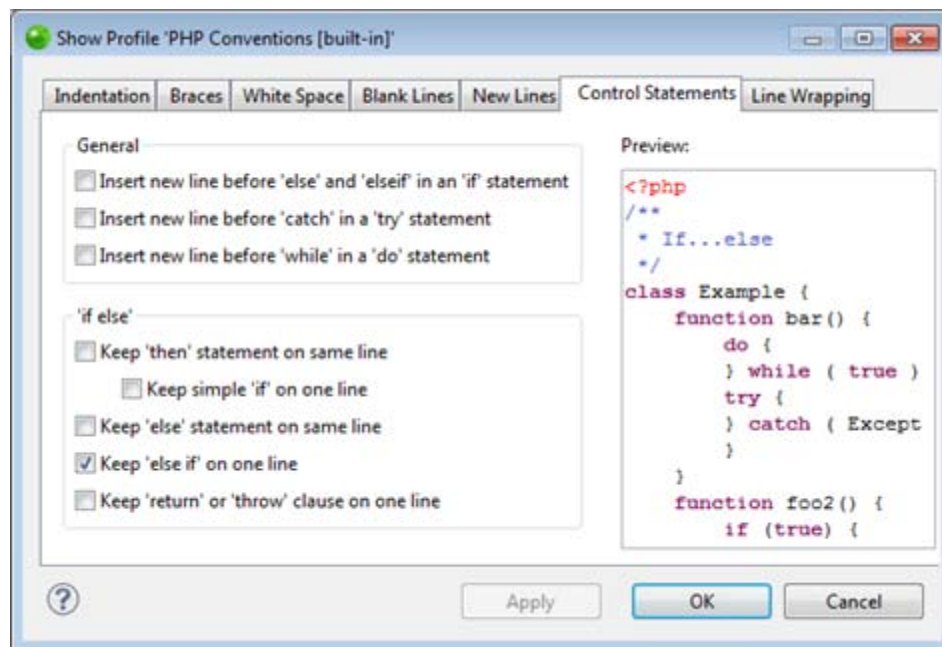
The Control Statements tab contains the following options:

General

- Insert new line before 'else' and 'elseif' in an 'if' statement
- Insert new line before 'catch' in a 'try' statement
- Insert new line before 'while' in a 'do' statement

'if else'

- Keep 'then' statement on same line
- Keep simple 'if' on one line
- Keep 'else' statement on same line
- Keep 'else if' on one line
- Keep 'return' or 'throw' clause on one line

Line Wrapping

The Line Wrapping tab allows you to configure Line Wrapping properties.
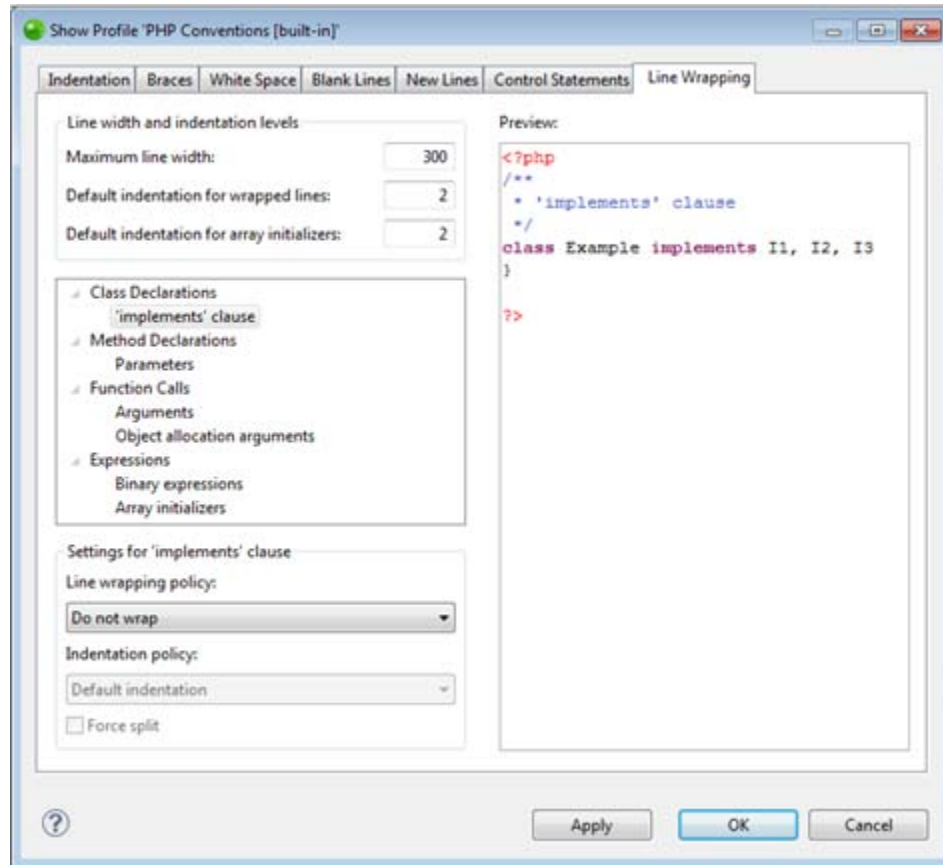
The Line Wrapping tab contains the following options:

Line Width and Indentation Levels

- Maximum Line Width - Enter the maximum line width (in characters).
- Default indentation for wrapped lines - Enter the default indentation for wrapped lines (in characters).
- Default indentation for array initializers - Enter the default indentation for array initializers (in characters).

Line Wrapping and Indentation Policies

To set line wrapping and indentation wrapping policies for a specific element, select an element from the collapsible list and select an option from the 'Line wrapping policy' and 'Indentation policy' drop-down lists. The elements for which the line and indentation policies can be applied are :

- Class Declarations
  - 'implements' clause'
- Method Declarations
  - Parameters

715

- Function Calls
    - Arguments
    - Object allocation arguments
- Expressions
    - Binary expressions
    - Array Initializers

Importing an Existing Formatting Configuration

To import an existing formatting configuration:

1. Click **Import**.
2. Select an XML file with the required configuration settings.
3. Click **OK**.

The new configuration will be added to the list.

Exporting a Configuration File to an XML File

To export a configuration file to an XML file:

1. Select the required configuration from the drop-down list.
2. Enter a name and location for the file.
3. Click **OK**.
4. An XML file will be created with the required settings.

Applying Formatter Preferences Settings to a Specific Project

To apply Formatter Preferences settings to a specific project only:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.

   A Formatter Preferences Properties dialog will appear.
3. Select the required settings and click **Apply**.

   A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
4. Click **Yes** to rebuild the project.

   -Or- Click **No** for a rebuild to be performed only when Zend Studio is restarted.

   -Or- Click **Cancel** to cancel the operation.

Debug Preferences

The Debug preferences page allows you to configure default settings for the debugging process.

The Debug Preferences page is accessed from **Window | Preferences | PHP | Debug Preferences**.

Debug Preferences page

The settings that can be configured from the debug preferences page are:

Default Settings

- PHP Debugger - The default debugger is the Zend Debugger. Go to the <u>Installed Debuggers</u> <u>preferences page</u> to configure Zend Debugger settings.
- Server - Choose which server the debugger will use by default. Click the "PHP Servers" category to be taken to the PHP Servers management page. For more on this, see <u>PHP Servers.</u>
- PHP Executable - Choose the required default PHP version. Click the "PHP Executables" category to be taken to the PHP Executables management page. For more on this, see <u>PHP</u> <u>Executables.</u>

Encoding Settings

- Debug Transfer Encoding - Select the required debug transfer encoding from the drop-down list.
- Debug Output Encoding - Select the required debug output encoding from the drop-down list.
- Break at First Line - Mark this checkbox to force the debugging process to stop at the first line of code by default.

Note:

Further PHP encoding options can be accessed from the preferences menu under **General | Content Types | Text | PHP Content Type**.

To apply Debug Preferences settings to a specific project only:

1. Select the link labeled "Configure Project Specific Settings".
2. Select the required project from the list.
   A Debug Preferences Properties dialog will appear.
3. Select the required settings and click **Apply**.
4. A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
5. Click **Yes** to rebuild the project. Click **No** for a rebuild to be performed only when Zend Studio is restarted. Click **Cancel** to cancel the operation.

Installed Debuggers Preferences

The Installed Debuggers Preferences page allows you to configure your Debugger settings.

The Installed Debuggers Preferences page is accessed from **Window | Preferences | PHP | Debug | Installed Debuggers**.
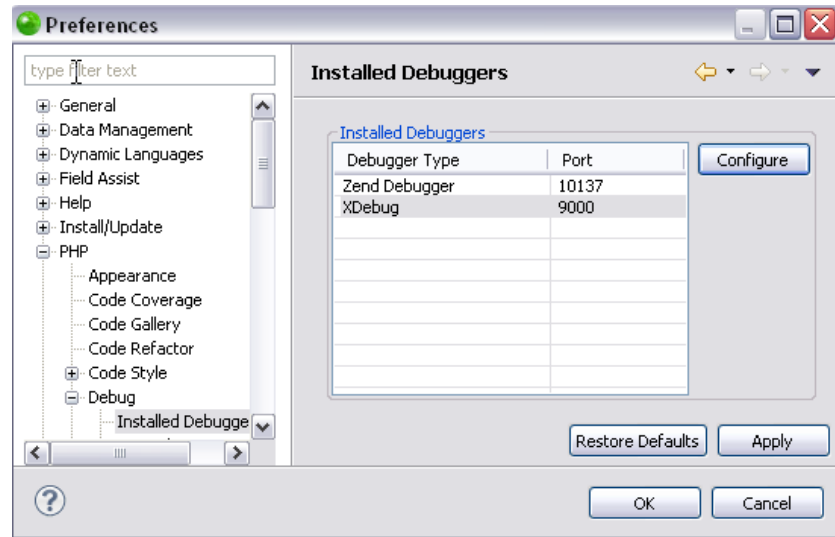
Configuring Your Zend Debugger Settings

To configure your Zend Debugger settings:

1. Select the Zend Debugger.

2. Click **Configure**.

    The Zend Debugger Settings dialog will open.
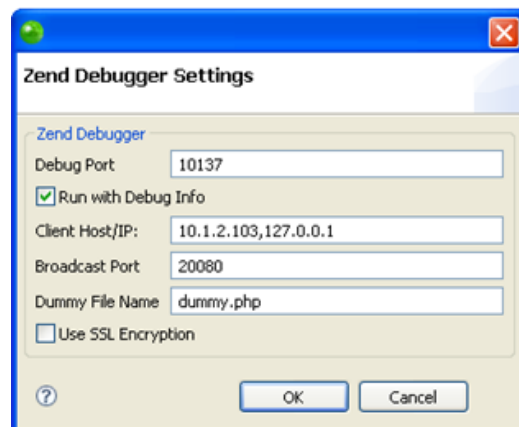
3. Configure the following:
    - Debug Port - The port which the Zend Debugger will use. The default port is 10137.
    - Run with Debug info - Mark the checkbox for Debug info, such as the Console view and the Browser Output, to be displayed when a Run configuration is executed.
    - Client Host/IP - Enter the Client Host/IP to which debugging results will be returned. Zend Studio will automatically search for and recognize the Client Host/IP, but entering a specific Host/IP will speed up the debugging process and decrease the likelihood of session timeouts.

Note:

If the field contains too many host/IPs, the session could timeout before a debugging connection is established. Entering a non-existent host/IP could cause the session to terminate.

    - Broadcast Port - The Broadcast Port allows your [Zend Browser Toolbar](#) or your [Zend Server](#) to detect your debugging preferences. The Broadcast Port number entered here must match the Broadcast Port entered in your Zend Browser Toolbar.
      The default port is 20080.
    - Dummy File - This is the file which the PHP Script debugger uses in order to start a PHP script debugging session on a specified server. The name should be left as the default dummy.php. However, if this is changed, ensure the change has also been made on your server.
      See [Ensuring the Placement of dummy.php](#) for more information.
    - Use SSL Encryption - Mark this checkbox to Encrypt Communication using SSL. Your server must support this option in order for it to be applicable.
4. Click **OK** to return to the Installed Debuggers Preferences page.
5. Click **Apply** to apply your settings.

Step Filtering Preferences

The Step Filtering Preferences page allows you to select certain resources/file patterns which will not be 'stepped into'/displayed during debugging.

This feature should be used when there are files which you do not want to inspect during debugging. This is especially useful if you have included large external resources (such as a framework or library).

The Step Filtering Preferences page is accessed from **Window | Preferences | PHP | Debug | Step Filtering**.
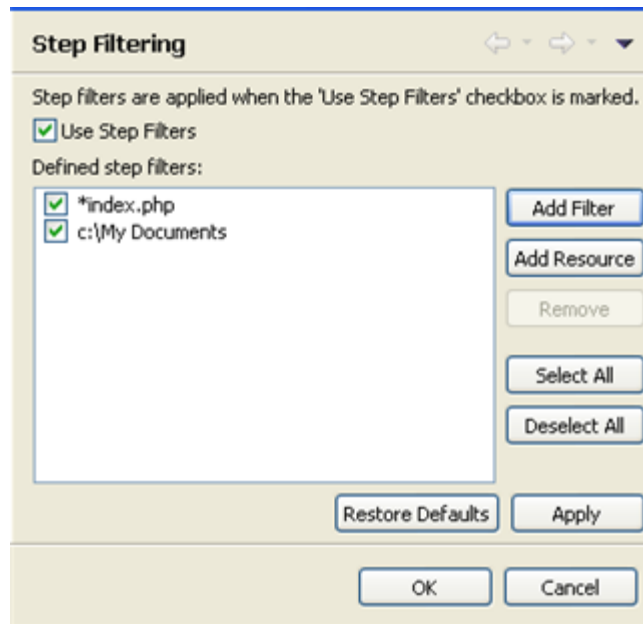
Step Filtering Preferences page

To enable the step filtering feature, mark the 'Use Step Filters' checkbox and the add the required resources to the 'Defined step filters' list.

Note:

The Step Filters functionality can be toggled on/off during debugging by clicking the Use Step Filters button on the Debug view toolbar.

Adding a File Name Pattern to Exclude

To add a file name pattern to exclude:

1. Click **Add Filter**.

   The Add Step Filter dialog is displayed.

2. Enter the pattern to filter.

> **Note:**
> Wild cards are enabled, so entering *index.php will exclude all resource paths ending in
> index.php.

3. Click **OK**.

The file exclusion pattern will be added to the step filters list.

Adding a Resource to Exclude

To add a resource to exclude:

1. Click **Add Resource**.

   The Select Resource dialog is displayed.

2. Select the resources (projects/folders/files) which you want to exclude and click **OK**.

The selected resources will be added to the step filters list.

Workbench Options Preferences

The Workbench Options preferences dialog allows you to configure the default behavior of the workspace during the debugging process.

The Workbench Options Preferences page is accessed from **Window | Preferences | PHP | Debug | Workbench Options**.
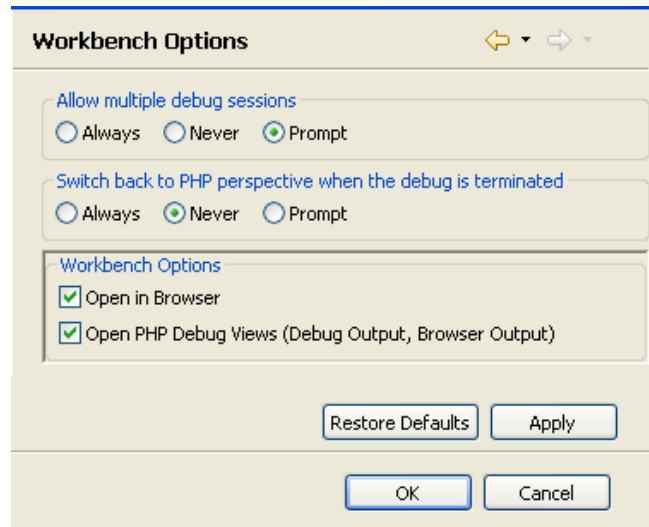
Workbench Options Preferences page

The Workbench Options configuration options are:

- Allow multiple debug sessions - Select whether to Allow multiple debug sessions to run simultaneously (Always, Never or Prompt).
- Switch back to PHP perspective when the debug is terminated - Select whether the PHP Persepctive will open when the debug is terminated (Always, Never or Prompt).

Workbench Options

- Open in Browser - Mark the checkbox for the debugged files to be displayed in a browser during debugging.
- Open PHP Debug Views - Mark the 7checkbox for PHP Debug Views to be displayed when a debug session is launched.
  By default, a dialog will appear when a debug session is launched asking whether you want to open the Debug Perspective when a debugging session is run. To change this behavior, open the Perspectives Preferences dialog by going to Window | Preferences | Run/Debug | Perspectives and select Always, Never or Prompt in the 'Open the associated perspective when launching' category.

Editor Preferences

The Editor Preferences page allows you to configure smart caret positioning behavior in the editor. Smart caret positioning determines where the cursor will jump to when certain positioning keys (e.g. Home / End) are pressed.

The Editor Preferences page is accessed from **Window | Preferences | PHP | Editor**.
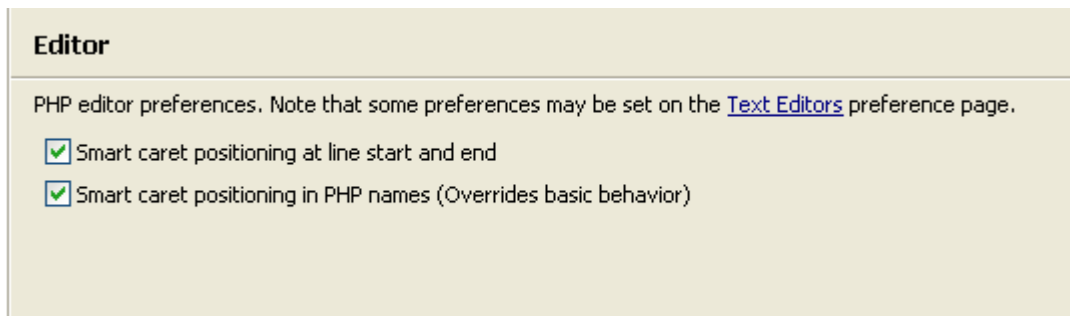
Editor preferences page

To configure Smart Caret Positioning Preferences

1. Mark the required checkboxes to configure the following options:
   - Smart caret positioning at line start and end - If this checkbox is unmarked, the cursor will jump to the beginning/end of a line when Home/End are pressed. If it is marked, the cursor will jump to the beginning/end of the *typed* line (i.e. ignoring the tabs at the beginning/end of a line).
   - Smart caret positioning in PHP names - If this checkbox is marked, the cursor will jump to the beginning/end of a 'word' within a PHP element (class / function / variable) when Ctrl + left arrow/right arrow are pressed. When unmarked, it will jump to the beginning/end of the whole element.

2. Click **Apply** to apply your settings.

Note:
More editor settings can be accessed by clicking the "Text Editors" link.

Content Assist Preferences

The Content Assist feature enables the selection and insertion of existing code elements to complete partially entered code.

The Content Assist preferences page allows you to configure your Content Assist preferences.

The Content Assist Preferences Preferences page is accessed from **Window | Preferences | PHP | Editor | Content Assist**.
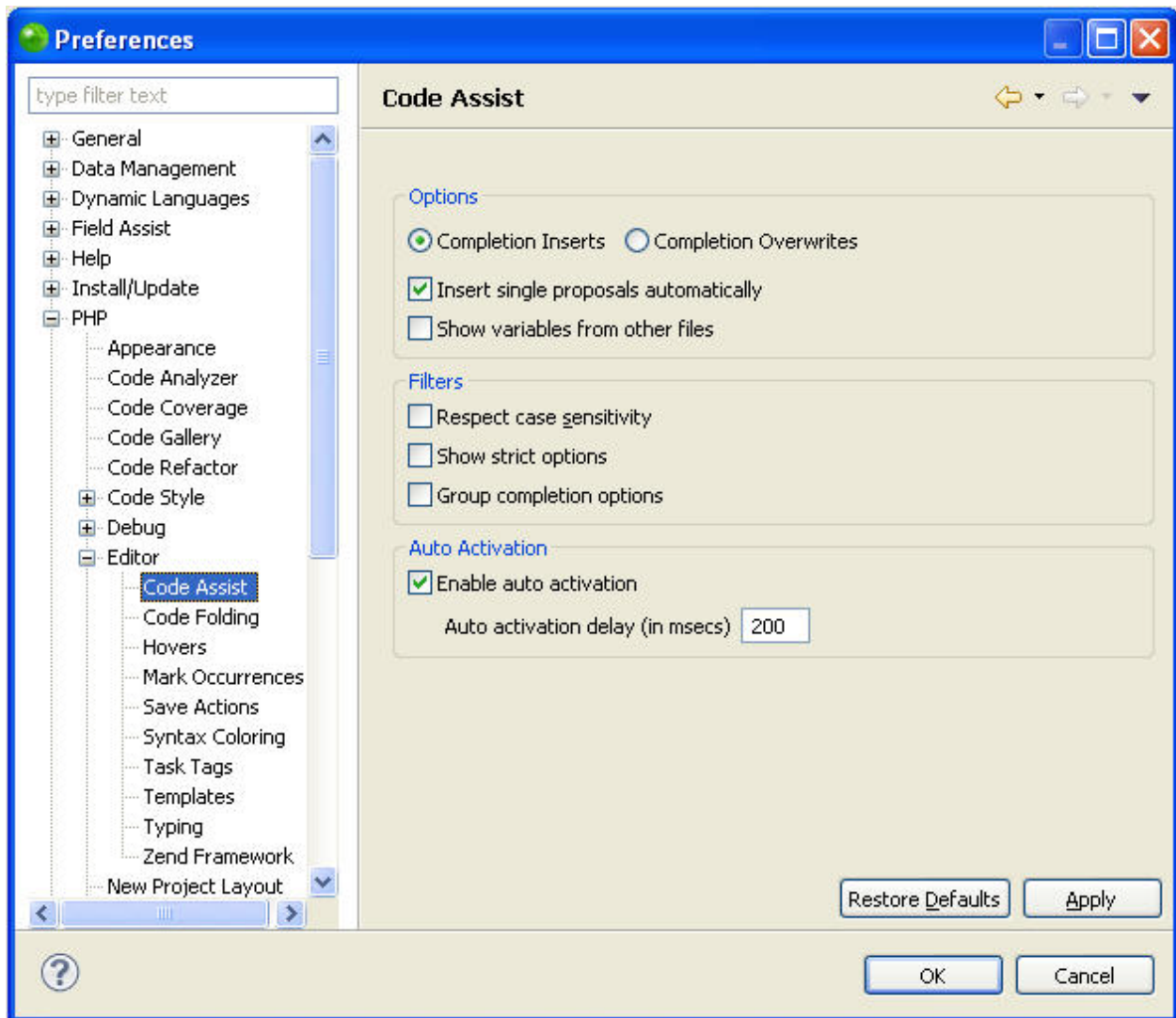
Configuring Content Assist Preferences

To configure Content Assist preferences:

1. Mark the required checkboxes to configure the following options:

Options

- Completion Inserts / Completion Overwrites - Select whether selecting an item from the Content Assist list will cause new code to be entered or existing code to be overwritten.
- Insert single proposals automatically - When only one content assist suggestion exists, the content assist suggestion will be inserted automatically.
- Show variables from other files - Shows variables which are in other files in the project.

Filters

- Respect case sensitivity – Define statements are used to create global constants. One can "r;define('r;MY_CONST', 5)", and then use "r;MY_CONST" all over the application, this property is used to determine if "r;my_const" will also be suggested as usage of that constant.
- Show strict options – PHP alerts users if they access static methods and fields from within an instance. Users can enable/disable this in the php.ini file for their php copy and hence can disable the content assist for these cases.

Auto-activation

- Enable auto activation - The content assist list will automatically be displayed.

Note:
If this is unmarked, you can still display the content assist list by pressing Ctrl+Space.

- Auto activation delay: Determines the delay before the Content Assist box is automatically displayed (in msecs).

2. Click **Apply** to apply your settings.

Code Folding Preferences

Code Folding enables you to 'collapse', or hide, certain sections of code while you are not working on them. This enables you to manage larger amounts of code within one window.
The Code Folding preferences page allows you enable / disable code folding and to select which elements should be folded by default.

The Code Folding Preferences page is accessed from **Window | Preferences | PHP | Editor | Code Folding**.
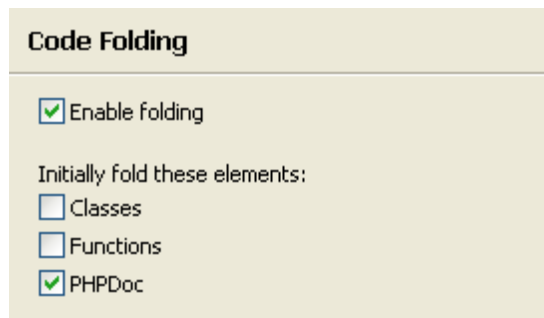
Folding preferences page

Configuring Your Code Folding Preferences

To configure your code folding preferences:

1. Mark the Enable folding checkbox to enable code to be folded.
2. Select which off the following elements should be folded by default by marking the relevant checkboxes:
   - Classes
   - Functions
   - PHPDocs
3. Click **Apply** to apply your settings.

Hovers Preferences

The Hover functionality will display information about an item when the mouse is placed on it. The Hovers preferences page allows you to configure the settings and shortcuts for the Hover functionality.

The Hovers Preferences page is accessed from **Window | Preferences | PHP | Editor | Hovers**.
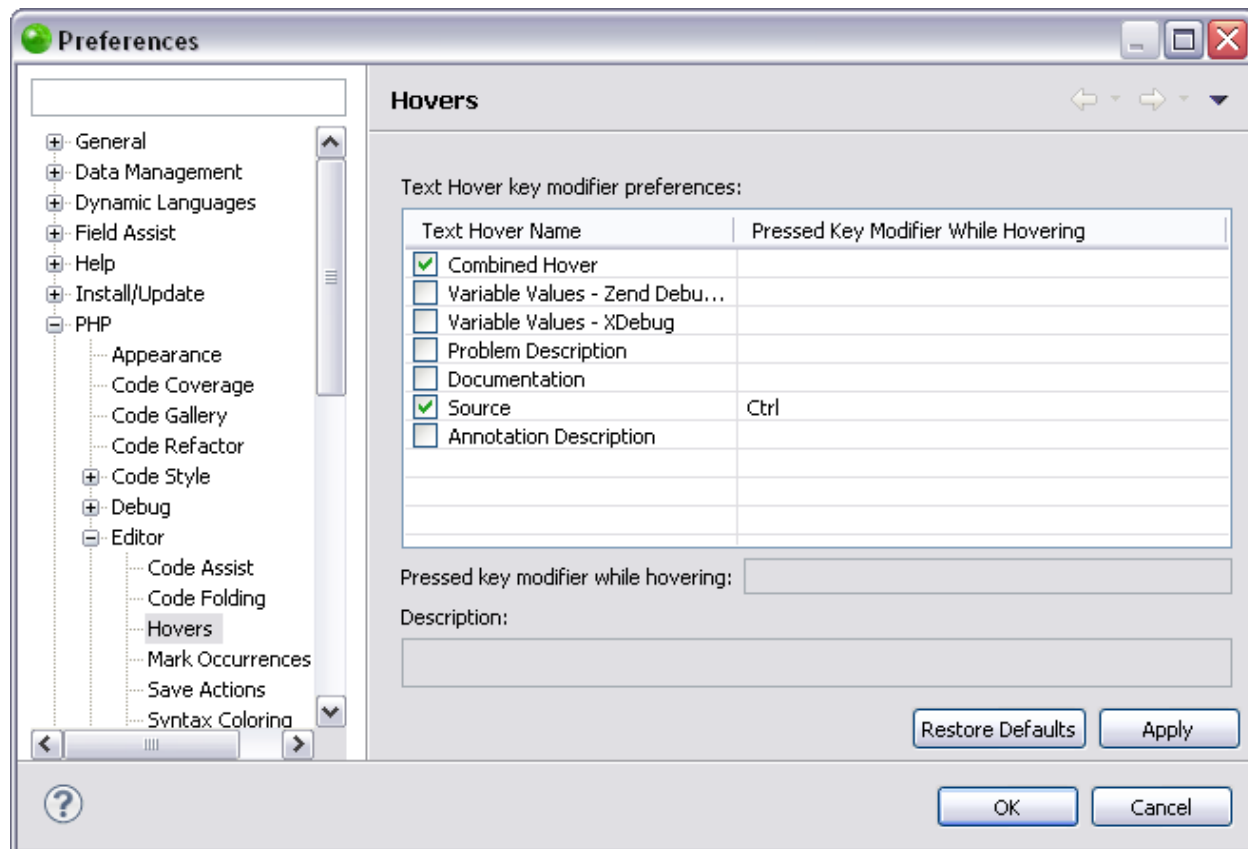
Hovers preferences page

The Text Hover key modifier preferences table allows you to modify hover key preferences for certain elements. Pressing the configured key while hovering over the element in the editor will display the relevant information or take the relevant action.

For example, applying the settings displayed in the screenshot above (Source key preference = Ctrl) and pressing Ctrl while hovering over an element in the editor will take you to that element's source.

You can configure key preferences for the following elements:

- Combined Hover - Tries the hover in the sequence listed in the table and uses the one which fits best for the selected element and the current context.

- Variable Values - Zend Debugger - Shows the value of the selected variable while debugging with the Zend Debugger.

- Variable Values - XDebug - Shows the value of the selected variable while debugging with XDebug.

- Documentation - Shows the documentation of the selected element.

- Problem Description - Shows the description of the selected problem.

- Source - Shows the source of the selected element.

- Annotation Description - Shows the description of the selected annotation.

Configuring the Key Preferences

To configure the key preferences:

1. Mark the checkbox next to the required preference.
2. Enter the required key in the 'pressed key modifier while hovering' box.
3. Click **Apply** to apply your settings.

Mark Occurrences Preferences

The Mark Occurrences Preferences page allows you to configure for which types of elements the Mark Occurrences feature will be enabled.

The Mark Occurrences Preferences page is accessed from **Window | Preferences | PHP | Editor | Mark Occurrences**.
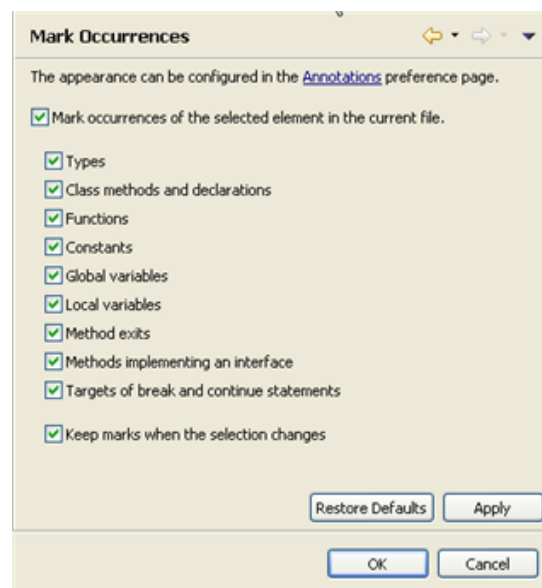
Mark Occurrences Preferences

To configure Mark Occurrences preferences:

1. Mark the 'Mark occurrences of the selected element in the current file' checkbox to enable the Mark Occurrences functionality.

2. Configure the elements for which the Mark Occurrences feature will be enabled by marking the relevant checkboxes. The options are:

   - Types
   - Class Methods and declarations
   - Functions
   - Constants
   - Global variables
   - Local variables
   - Expressions throwing a declared exception
   - Method exits - Marks the exit points (throws / return / end of flow) of a method
   - Methods implementing an interface
   - Targets of break and continue statements - Marks the scope (for, foreach, while, do-while or switch structure) of a break / continue statements
   - HTML Tags

3. Mark the 'Keep marks when the selection changes' checkbox for marks to continue to be displayed once the cursor has been moved from the selected element.

4. Click Apply to apply your changes.

The appearance of the Mark Occurrences annotations which appear in the vertical ruler (to the left of the editor) and the annotation bar (to the right of the editor) can be configured in the Annotations preferences page (**Window | Preferences | General | Editors | Text Editors | Annotations**) by selecting PHP elements 'read' or 'write' occurrences.
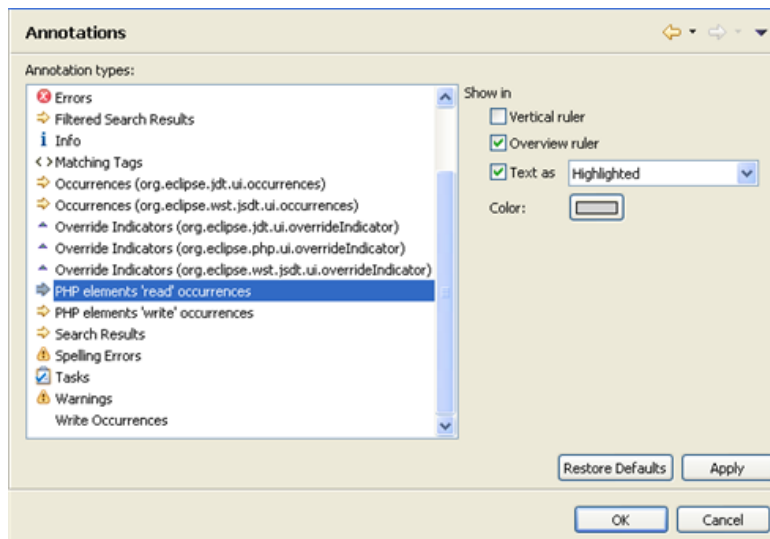
Save Actions Preferences

The Save Actions Preferences page lets you remove trailing whitespace from a file each time you save it.

The Save Actions Preferences page is accessed from **Window | Preferences | PHP | Editor | Save Actions**.
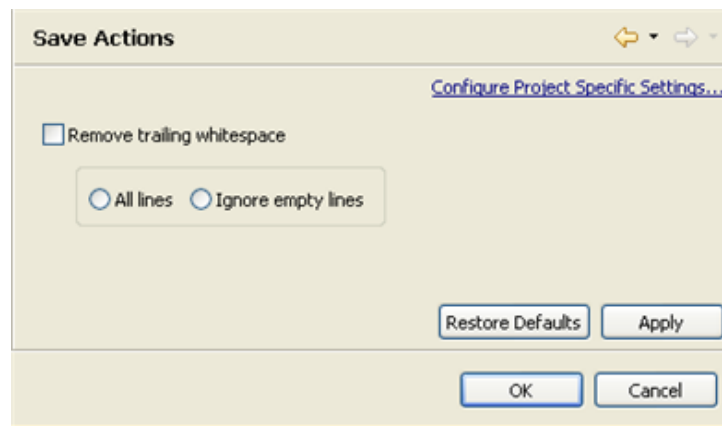
Configuring Save Actions

To configure Save Actions:

1. Mark the 'remove trailing whitespace' checkbox to enable the removal of whitespace after every save.
2. Select whether All whitespace lines will be removed (by marking 'All lines'), or whether empty lines will be ignored (by marking 'Ignore empty lines).
3. Click **Apply** to Apply your settings.

Every time you save a file, whitespace will be removed according to the configured settings.

Applying Save Actions Settings to a Specific Project

To apply Save Actions settings to a specific project only:

1. Click the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.

    A Save Actions Properties page will appear.
3. Select the required settings and click **Apply**.

Note:

Save Action settings can also be configured for an existing project by right-clicking the project in PHP Explorer view and selecting Properties | Save Actions.

Syntax Coloring Preferences

The Syntax Coloring preferences page allows you to set the foreground color, background color and font type for different icons, in order to make your script manageable and easier to read.

The Syntax Coloring Preferences Preferences page is accessed from **Window | Preferences | PHP | Editor | Syntax Coloring**.
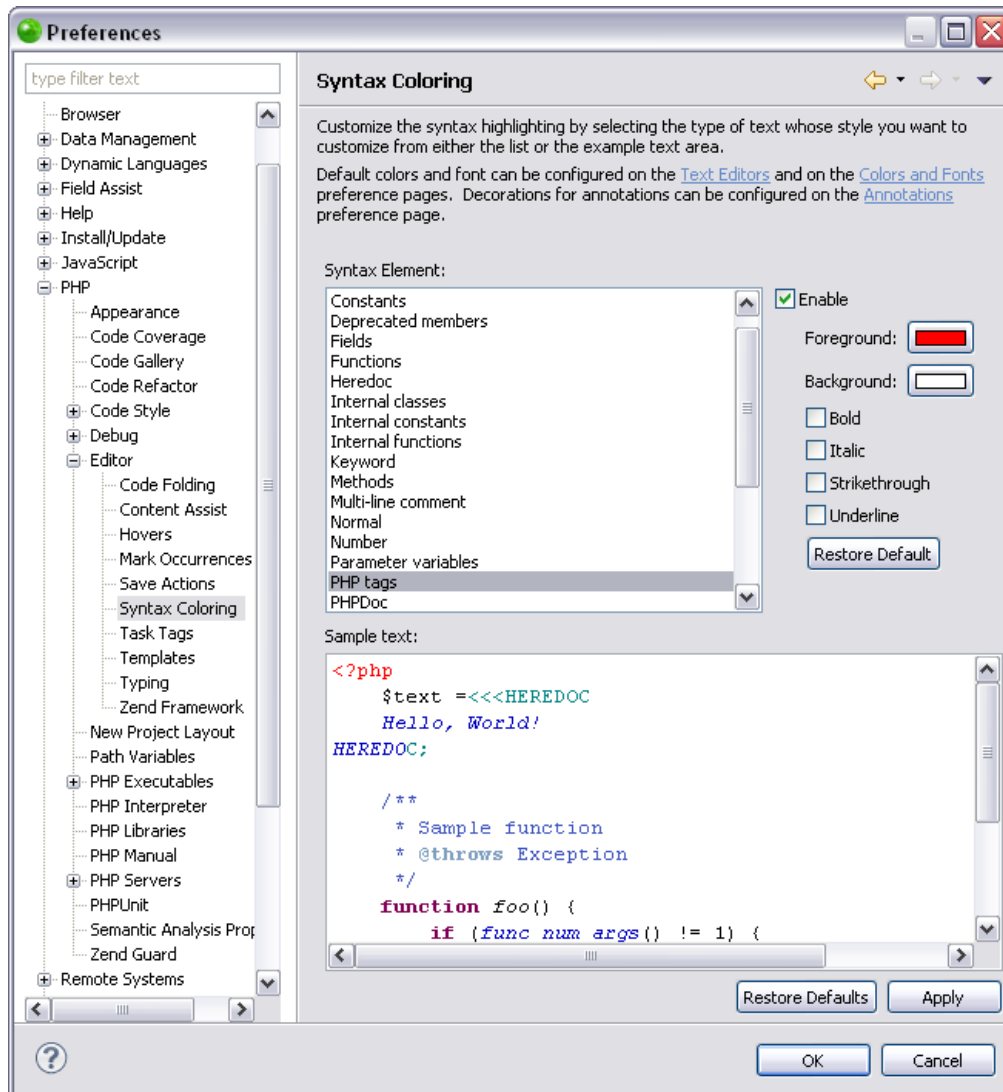
Configuring the Colors and Fonts for an Item

To configure the colors and fonts for an item:

1. Select the required item from the Syntax element list.

2. Click on **Foreground** or **Background** to select a color.

3. Select what formatting, if any, you would like to apply to the text (Bold, Italic, Strikethrough, Underline)

4. Click **Apply** and **OK** to apply and save your settings.

The Sample text box displays a preview of the different elements.

More color and font options can be configured by opening the preferences page, accessed from **Window | Preferences**, and selecting:

- General | Appearance | Colors and Fonts
- General | Editors | Text Editors | Annotation
- General | Editors | Text Editors | Quick Diff
- Run / Debug
- Run / Debug | Console
- Team | CVS | Console

Task Tags Preferences

The Task Tags preferences page allows you to add new task tags and edit existing ones. Tasks are used as reminders of actions, work to do or any other action required by the programmer. Task tags are strings used by Zend Studio to recognize when tasks are added in your script. Anything after these strings inside a comment will be recognized as a task.

See the Tasks view topic in the Workbench User Guide for more information on using tasks.

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

The Task Tags Preferences page is accessed from **Window | Preferences | PHP | Editor | Task Tags**.
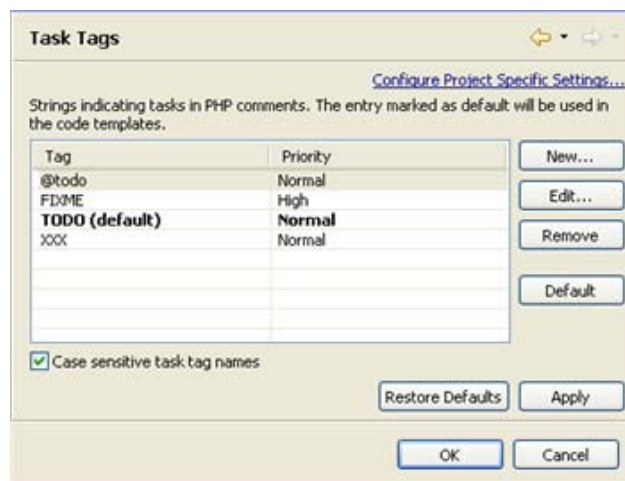
Adding a New Task Tag

To add a new Task Tag:

1. Click **New**.

2. Enter a tag name and priority (High/Normal/Low). Tags may contain any character string.

3. Click **OK**.

4. Click **Apply** to apply your settings.

The new tag will be added to the list and will trigger a task when inserted in the editor.

Editing a Tag

To edit a tag:

1. Double click the tag -or- select it and click **Edit**.

2. Edit the tag name or priority.

3. Click **OK**.

4. Click **Apply** to apply your settings.

Selecting a tag and clicking Default will set the task tag as the default one to be used in Code templates.

See the [Templates Preferences](#) topic for more on template preferences.

Note:

If the 'case sensitive task tag names' checkbox is marked, task tag names will be case sensitive.

Applying Task Tags Settings to a Specific Project

To apply Task Tags settings to a specific project only:

1. Select the link labelled "Configure Project Specific Settings".

2. Select the required project from the list.

    A Task Tags Properties dialog will appear.

3. Select the required settings and click **Apply**.

    A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.

4. Click **Yes** to rebuild the project

    -Or- Click No for a rebuild to be performed only when Zend Studio is restarted.

    -Or - Click Cancel to cancel the operation.

Templates Preferences

The Templates Preferences page allows you to create, edit, delete, import and export templates. Templates are shortcuts used to insert a pre-defined framework of code. The purpose is to save time and reduce the potential for errors in standard, repetitive code units. Once a template is inserted, you can complete the code quickly using manual and automated code entry methods.

To insert a template into your code, enter the first few letters of its name and press Ctrl+Space to activate the Content Assist function and open the content assist list. Select the required template from the list. Templates are marked with a ⊟ icon. For more on using the content assist function, see Working with Content Assist.

The Templates Preferences page is accessed from **Window | Preferences | PHP | Editor | Templates**.
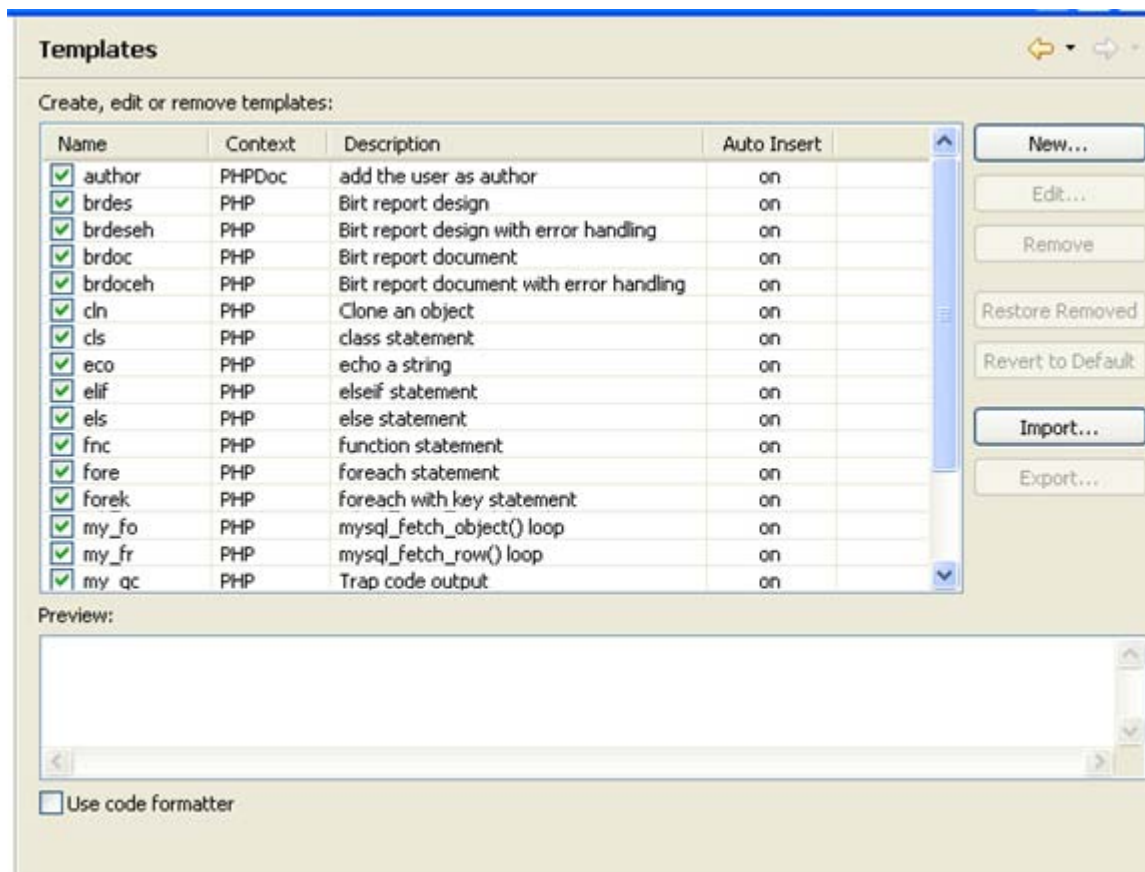
To remove a template from the list of available options, unmark its checkbox in the list.
To edit an existing template, select it from the list and click **Edit**.

This procedure describes how to create a new template to be added to the template list.

To create a new template:

1. Click **New**.

2. The New Template dialog will open.
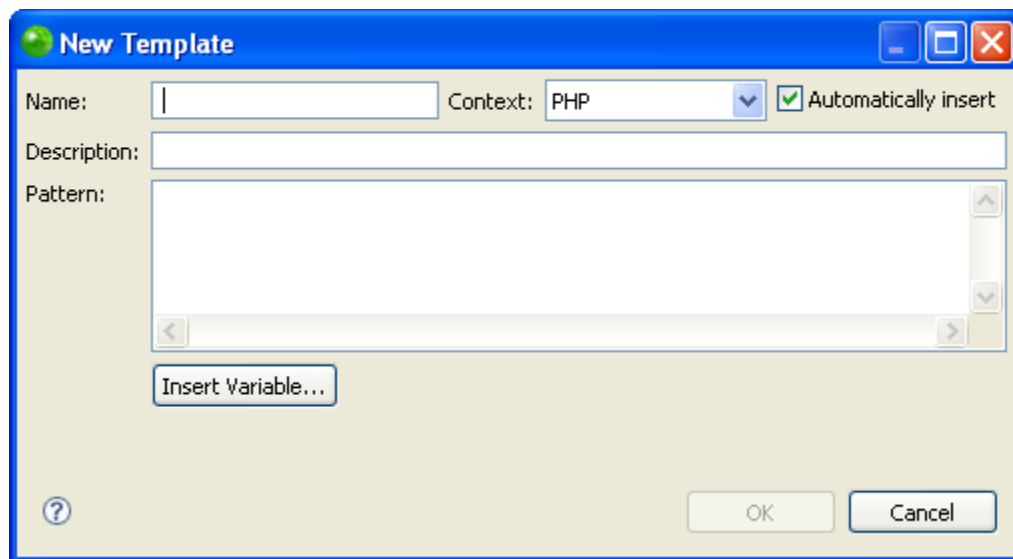
3. Enter the template's details:

   ▪ Name - A short name to identify the template (e.g. 'while' for a template for a while loop).

   ▪ Context - The code context when the template will be available (PHP, PHPDoc, New PHP, BIRT or Zend Framework). E.g. PHP templates will only be available for use when writing PHP code.

   ▪ Description - A short description of the template's code.

   ▪ Pattern - The pattern is the actual code string that will be inserted into the editor whenever this template is selected.

     Use the Insert Variable button to select from a list of common variables.

4. Click **OK**.

Your template will be added to the list and will be available from the Content Assist in the relevant context.

Exporting and Importing Templates

Zend Studio enables you to export and import Templates, created within XML files in the following format:

```
<?xml version="1.0" encoding="UTF-8" ?>
<templates>
<template autoinsert="true" context="php" deleted="false"
description="description" enabled="true" name="for">
for($$i = 0; $$i < 1; $$i++){ }
</template>
```

To import a template:

1. Click **Import** to open the Import Template's browser.
2. Select the location to import the relevant XML file containing the template information.
3. Click **Open**.

The templates contained in the template.xml file will be imported into the list of Templates.

To export a template:

1. Select the template(s) for export from the Template list.
2. Click **Export** to open the Export Template's dialog.
3. Select the location to save the XML file to.
4. Click **Save**.

An XML file will be created with the template information.

Note:

If you selected more than one template to export, all of them will be present in the exported XML file.

Each of the original Templates is bounded by: < template > </template>

Typing Preferences

The Typing preferences page allows you to configure the code and language patterns that Zend Studio will automatically complete, and whether the tab key will indent the current line.

The Typing Preferences page is accessed from **Window | Preferences | PHP | Editor | Typing**.
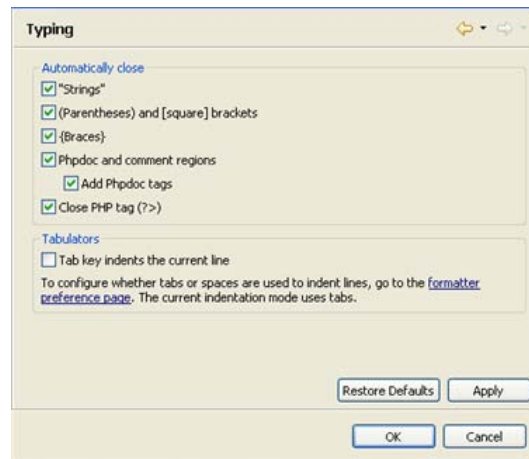
Automatically Close

Zend Studio can be set to automatically complete the following types of patterns:

- "Strings" - A pair of double quotes ("") will be inserted when a single quotation mark (") is entered.
- (Parentheses) and [Square] brackets - A pair of brackets will be inserted when the opening bracket is entered.
- {Braces} - A pair of braces will be inserted when the opening brace is entered.
- PhpDoc and comment regions - Automatically creates phpDoc Blocks
    - Add phpDoc tags - Adds phpDoc tags within the phpDoc Block comment.
- Close PHP tag - A closing PHP tag (?>) will be inserted when the opening PHP tag (<?php) is entered.

Configuring Your 'Automatic Close' Options

To configure your 'automatic close' options:
1. Mark the checkboxes of the patterns you would like Zend Studio to auto-complete.
2. Click **Apply** to apply your settings.

To use the auto-complete function, type the opening character in the editor. The matching character will be automatically inserted.

Tabulators

Indenting a Selected Line in the Editor Using the Tab Key

To be able to indent a selected line in the editor using the Tab Key:

1. Mark the 'tab key indents the current line' checkbox.

2. Click **Apply** to apply your settings.

For more on indentation preferences, see the Formatter Preferences page.

Zend Framework Preferences

The Zend Framework preferences page allows you to configure the display of the Zend Tool Console. The Zend Tool console displays messages and warnings when executing Zend Framework commands which utilize the Zend_Tool, such as create new Zend Framework Project or create new Zend Controller. See Using the Zend Tool Floating Window for more information.
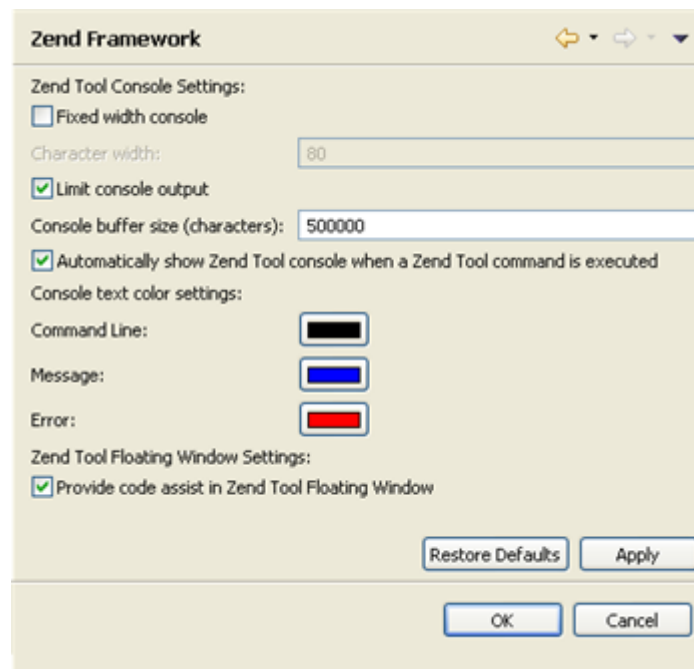
Zend Framework Preferences page

The Zend Framework Preferences Preferences page is accessed from **Window | Preferences | PHP | Editor | Zend Framework**.

Configuring the Zend Tool Console Display Settings

To configure the Zend Tool Console display settings:

1. Configure the following options:

   Zend Tool Consolse Settings:

   - Fixed width console - Mark the checcbox to limit the width of messages displayed in the Zend Tool console.

     Enter the required width in characters. The value must be between 1000 and 2,147,483,646.

   - Limit console output - Mark the checkbox to limit the amount of characters stored and displayed in the Zend Tool Console.

     Enter the required size in characters. The value must be between 80 and 2,147,483,646.

   - Automatically show Zend Tool console when a Zend_Tool command is executed - Mark this checkbox for the Zend Tool Console to be automatically displayed whenever a command which utilizes the Zend_Tool (e.g. create new Zend Framework project) is executed.

   - Console text color settings - Edit the colors for text displayed in the Zend Tool Console for the following:

     - Command line
     - Messages
     - Errors displayed

   Zend Tool Floating Window Settings:

   - Provide content assist in Zend Tool Floating Window - Provides content assist options for Zend_Tool commands in the Zend Tool Floating Window.

     See Using the Zend Tool Floating Window for more information.

2. Click **Apply** to apply your settings.

New Project Layout Preferences

The New Project Layout Preferences page allows you to configure the default layout for new PHP
projects. The default layout will configure whether all folders under the project root will be considered as
'source' folders for the Build process, or whether separate folders will be created for resources which are
included or excluded from the Build process.

The preferences configured in the New Project Layout Preferences page will affect the default option
selected in the Project Layout category of the New PHP Project wizard. This setting can be modified per
project during creation.
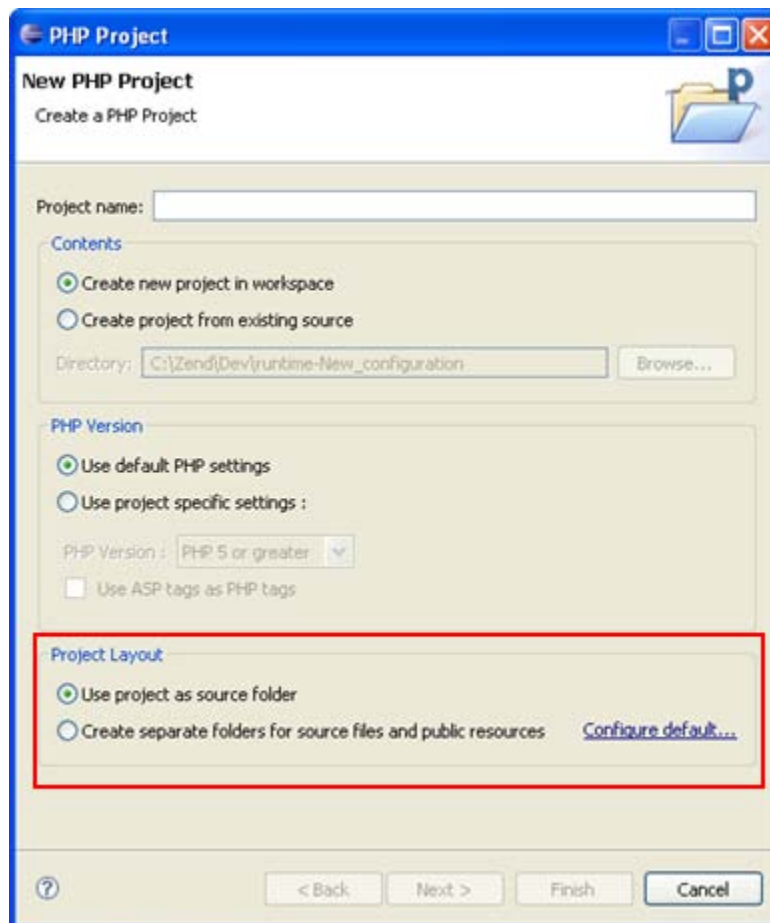
New PHP Project wizard - Project layout category

The New Project Layout Preferences page is accessed from **Window | Preferences | PHP | New Project Layout.**
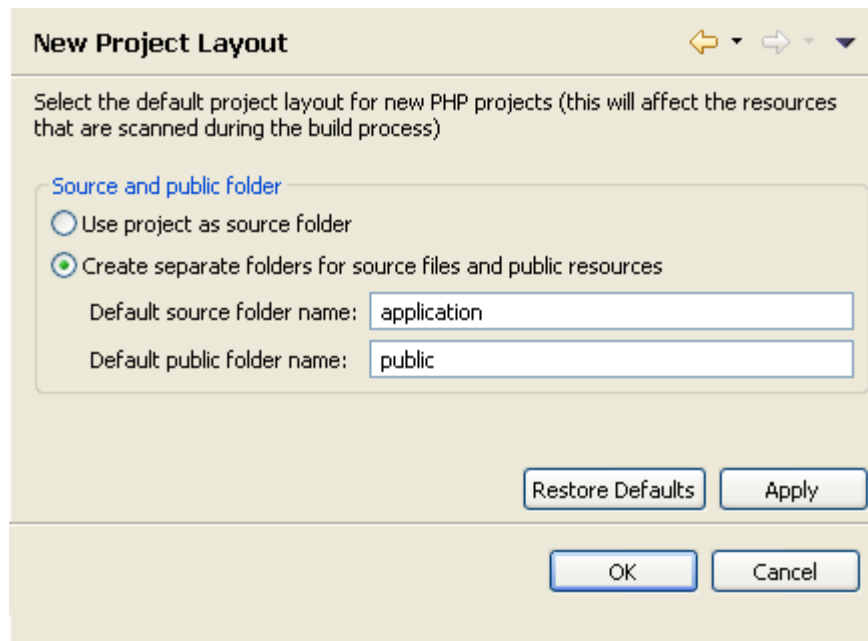
New Project Layout Preferences page

Configuring the Default New PHP Project Layout

To configure the default New PHP Project Layout:

Select the required option:

- Use project as source folder - By default, all folders created under the project root will be considered 'source folders' and will be scanned during the Build process.
  See <u>Configuring a Project's PHP Build Path</u> to configure the project's PHP Build Path.
- Create separate folders for source files and public resources - Two folders will be created under your project root - one 'source' folder for resources which will be scanned during the Build process and one 'public' folder which will be skipped during the Build process. You can change the default names for these folders by entering the required name in the 'Default source/public folder name' fields.

PHP Executables Preferences

The PHP Executables Preferences page allows you to add, edit, remove and find PHP Executables.

The internal debugger in the defined PHP Executable is used for <u>local PHP Script debugging</u>. Zend
Studio comes with a bundled pre-configured PHP Executable and local debugger.

The PHP Executables Preferences Preferences page is accessed from **Window | Preferences | PHP |
PHP Executables Preferences**.
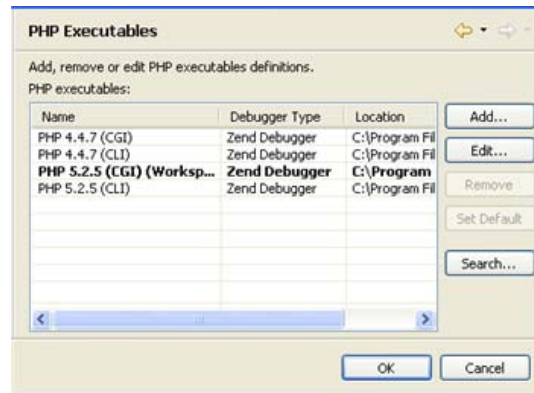
PHP Executables

Adding a PHP Executable to the List

To add a PHP executable to the list:

1. Click **Add**.

   The Add PHP Executable dialog will appear.
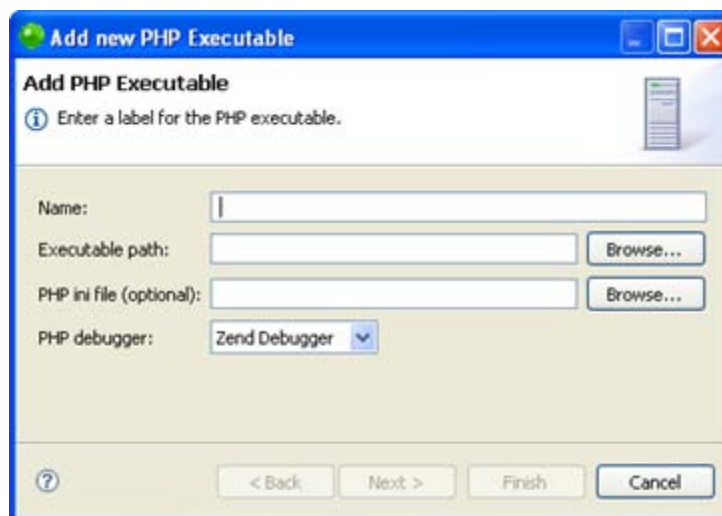
Add PHP executable dialog

2. Enter a name for the PHP Executable.

3. In the Executable path selection, enter the location of the PHP executable on your file system.

4. Select the PHP ini file to be associated with the PHP Executable by clicking Browse (optional).

5. Select the PHP Debugger to be used with the executable. By default this will be the Zend Debugger.

6. Click **Finish**.

The PHP executable will be added to your list.

Searching for a PHP Executable on Your Local File System

To search for a PHP executable on your local file system:

1. Click Search.

2. In the Directory Selection dialog, select the folder to search.

3. Click **OK**. Zend Studio will search for PHP executables in the location specified.

Any found PHP executables will be added to the list.

To change the name of the new PHP executable, select it from the list and click **Edit**.

If a PHP executable is not found in the specified directory, select a different location or add the PHP Executable manually by following the instructions under '<u>To add a PHP executable to the list</u>', above.

Execution Environments Preferences

The Execution Environments Preferences page displays the PHP Interpreters that are compatible with each PHP version available in Zend Studio. Selecting a compatible Interpreter with the PHP version you are using in your project allows Zend Studio to execute your project in an execution environment that contains Interpreters that are compatible to the PHP version being used, which avoids errors due to incompatible Interpreters.

The Execution Environments Preferences Preferences page is accessed from **Window | Preferences | PHP | PHP Executables | Execution Environments**.
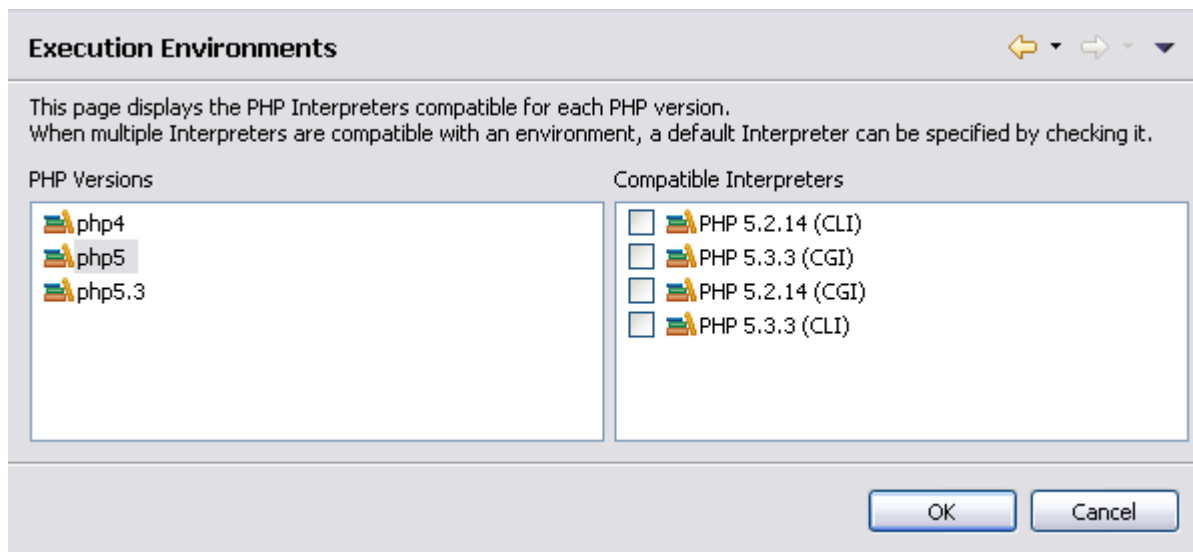
The Execution Environment preferences describe all of the available execution environments in Zend Studio, allowing default PHP Interpreters to be specified for any given PHP version. This allows you to create a compatible execution environment for PHP projects.

The following components make up the page:

- PHP Versions - The available PHP versions in Zend Studio. You can set which PHP version you would like to use in the PHP Interpreter Preferences page. You can also set a PHP version for a specific project when creating PHP Projects.

- Compatible Interpreters - The compatible Interpreters for the selected PHP version.
  When multiple Interpreters are compatible with a PHP version, a default Interpreter can be specified by checking it.

To manage PHP Executables see PHP Executables Preferences.

PHP Interpreter Preferences

The PHP Interpreter preferences page allows you to set which PHP version to use for the project. This will affect the internal debugger, code analyzer and content assist.

The PHP Interpreter Preferences page is accessed from **Window | Preferences | PHP | PHP Interpreter**.
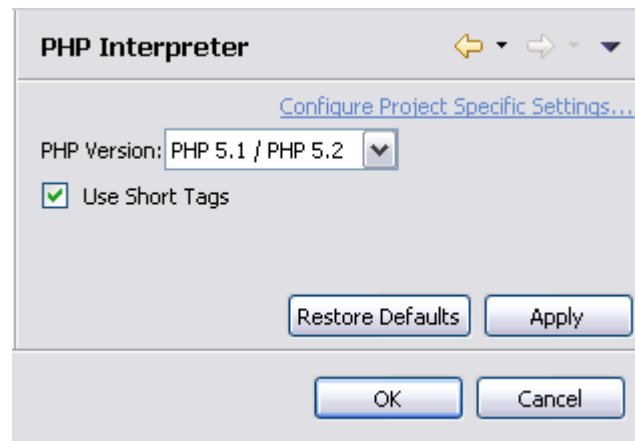
Configuring Your PHP Version

To configure your PHP version:

1. Select the PHP Version to use.
   See PHP Support for more on effects the PHP version settings have.
2. Mark the "Use Short Tags" check-box in order for Code Completion to respond to Short tags in the same way as it responds to PHP tags. A short tag is "<?".
3. Click **OK** to apply your settings.

Using a Different PHP Interpreter for a Specific Project

To use a different PHP Interpreter for a specific project:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the specific project from the list.
3. Another PHP Interpreter preferences page will open.
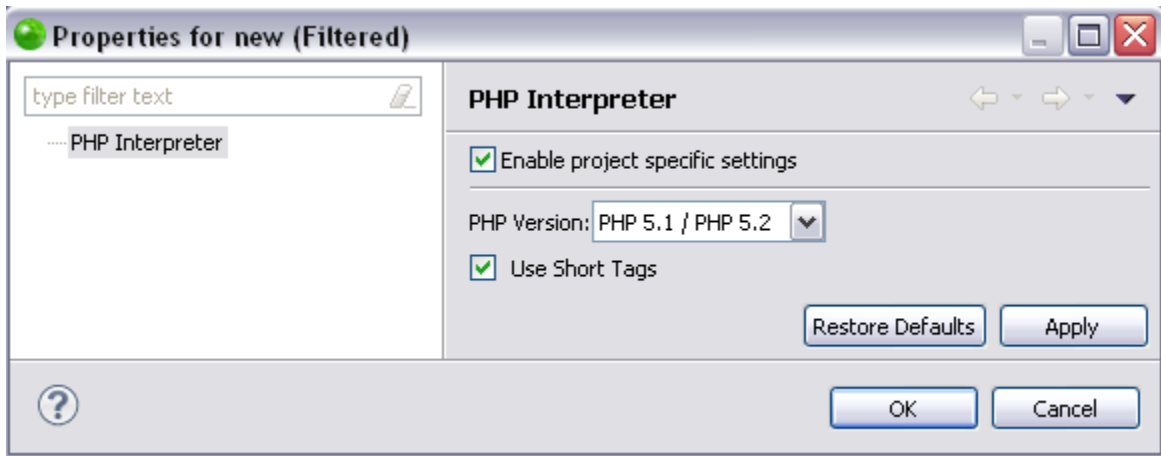
4. Mark the "Enable project specific settings" check-box.
5. Choose your PHP version.
6. Click **Apply**.
7. A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
8. Click **Yes** to rebuild the project. Error parsing will be performed according to the PHP version chosen.
   If you click **No**, the rebuild will be performed the next time Zend Studio is restarted.

PHP Libraries Preferences

The PHP Libraries Preferences page allows you to create and maintain an external code library.

Enabling PHP Libraries in your project allows libraries to be referenced by the project and makes the

elements within these resources available for operations such as Content Assist and Refactoring.

Note:
You may change the order your user libraries are in by using the **Up** and **Down** buttons. The order in which the libraries are arranged in this page defines the order they are available in Zend Studio's functionality, such as content assist.

The PHP Libraries Preferences page allows you to do the following:

- Add PHP Libraries
- Add External Folders to PHP Libraries
- Export PHP User Libraries
- Import PHP User Libraries
- Edit PHP User Libraries
- Edit PHP Library Components or Folders
- Remove a PHP Library or Library Folder

The PHP Libraries Preferences page is accessed from **Window** | **Preferences** | **PHP** | **PHP Libraries**.

Note:

Once you have added a user library in the PHP preferences page, you must also add it to your PHP

Include Path of the project in which you would like to have it available. For more information see

Configuring a Project's PHP Include Path.

PHP Manual Preferences

The PHP Manual preferences page sets the location of sites which allow you to access and view the PHP Manual and allows you to add, edit or remove sites.

The PHP Manual contains an explanation of PHP functions. PHP Manuals can be accessed online or locally from within Zend Studio in order to provide an immediate explanation of the functionality and proper use of all PHP functions.

The PHP Manual Preferences Preferences page is accessed from **Window | Preferences | PHP | PHP Manual Preferences**.
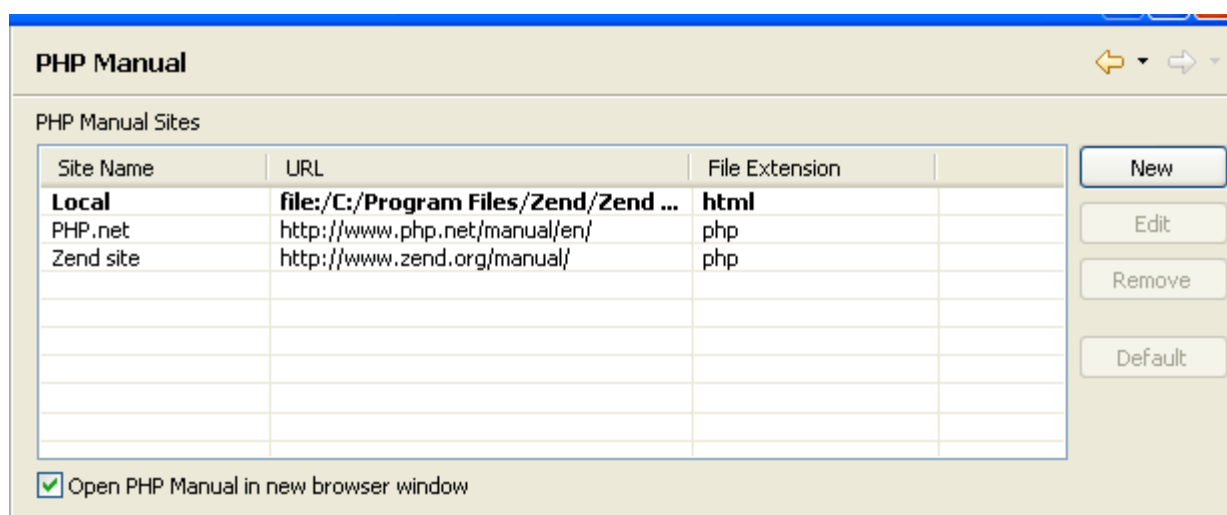
Adding Additional Sites to the List

To add additional sites to the list:

1. Click **New**.
2. Enter the Name of a site allowing access to the PHP Manual and its URL, Local Directory location or Windows CHM File location.
3. Choose whether its file extension is php, htm or html.
4. Click **OK**. Your new site will be added to the list.

See the PHP Manual Integration topic for more information.

Mark the 'Open PHP Manual in new browser window' checkbox to select that each request to open the manual will appear in a new browser tab of the Editor.

Note:
The initial, default site cannot be removed or edited.

Click **OK** to apply your settings.

PHP Servers Preferences

The PHP Servers Preferences page will display a list of your currently configured servers and allow you to add servers or edit settings for existing servers. The server settings will be used for debugging/profiling files on a remote server.

The PHP Servers Preferences page is accessed from **Window | Preferences | PHP | PHP Servers**.
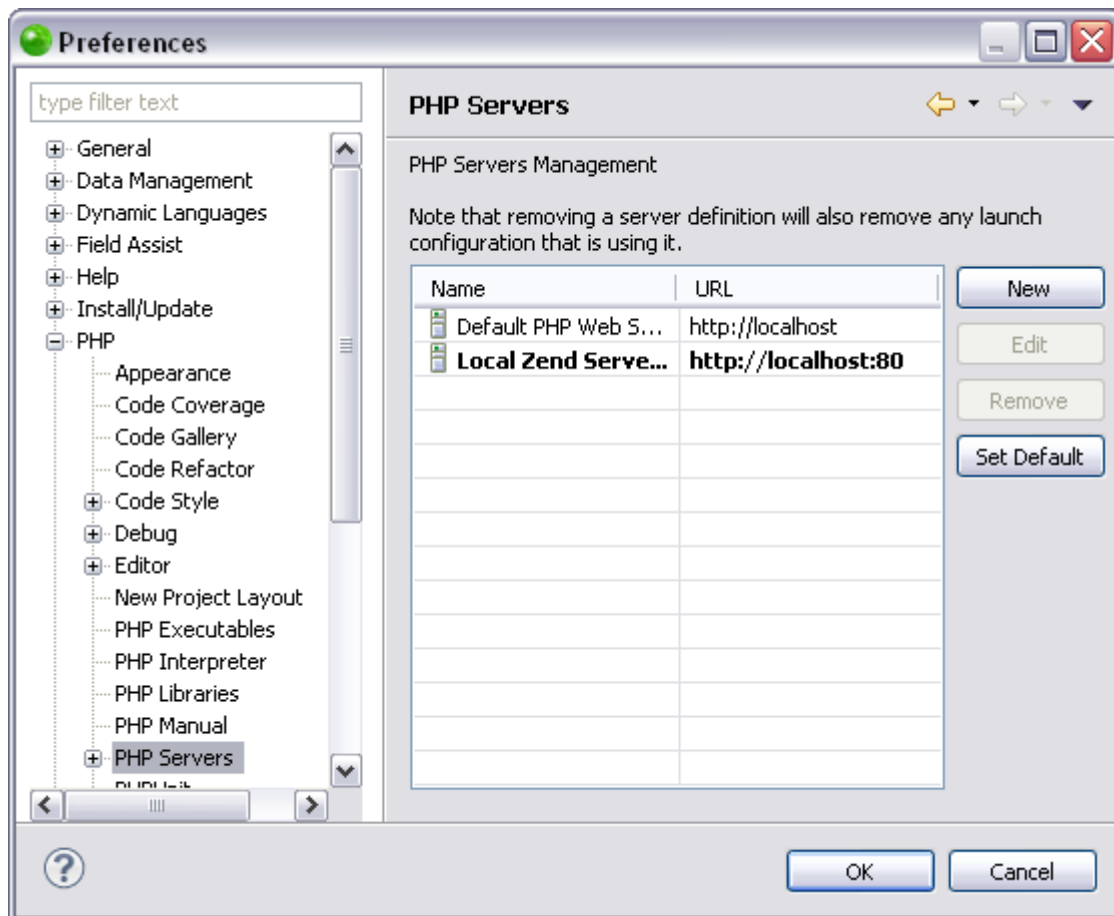
One server configuration setting is configured by default and will point to the URL http://localhost.

Note:

You can benefit from Zend's expertise by using the Zend Server , which includes the Zend Debugger, in order to perform optimal debugging and profiling functionalities. See Zend Server Integration for more information.

If you have a Zend Server installed on your local machine it is automatically detected and configured as the default server in the PHP Servers list. See Configuring Zend Server Settings in Zend Studio for more information.

Adding a New Server to the List or Editing an Existing Server Configuration

To add a new server to the list or edit an existing server configuration:

1. Click **New**

   -Or- select an existing server and click **Edit**.

2. Enter the name of your server.
3. Enter the URL that points to its document root.
4. Click **Next**.

   The Server Path Mapping dialog appears.
5. If you would like to map a path on your server to a local path, click Add and enter:
   - The path on your server.
   - The path you would like to map it to in your Workspace or on your File System.

     See Managing Path Maps for more information.
6. Click OK and Next.
7. If you would like to enable integration with Zend Server , mark the 'Enable Zend Server Integration' checkbox and enter the relevant information:
   - URL Suffix - The suffix which should be added to the URL of your document root in order to browse to your Zend Server GUI.
   - Port number - Enter the port number you defined during Zend Server installation.

     See the Zend Server FAQ site for default port number settings.
   - Password - The password used to access your Zend Server GUI.
8. Click **Next**.

9. If you would like to enable Tunneling, mark the 'Enable Tunneling' checkbox and enter the relevant information:

   - Specify Return Host - If the debug session occurs on a node other than the central server, unmark the checkbox and specify the IP address of the central server to which the tunnel will be opened.
   - Automatically Connect on Startup
   - Send Authentication Information - User Name and Password.
     See [Setting Up Tunneling](#) for more information.

10. Click **Finish**.

Your new server configuration will be added to the server list and will be available for actions such as debugging, profiling and

Tracer Preferences

The Tracer Preferences page displays the ports that the Code Tracing functionality will use. The ports that Zend Studio chooses by default should not be changed unless they are specifically blocked or in use by another application.

The Tracer Preferences page is accessed from **Window | Preferences | PHP | PHP Servers | Tracer**.
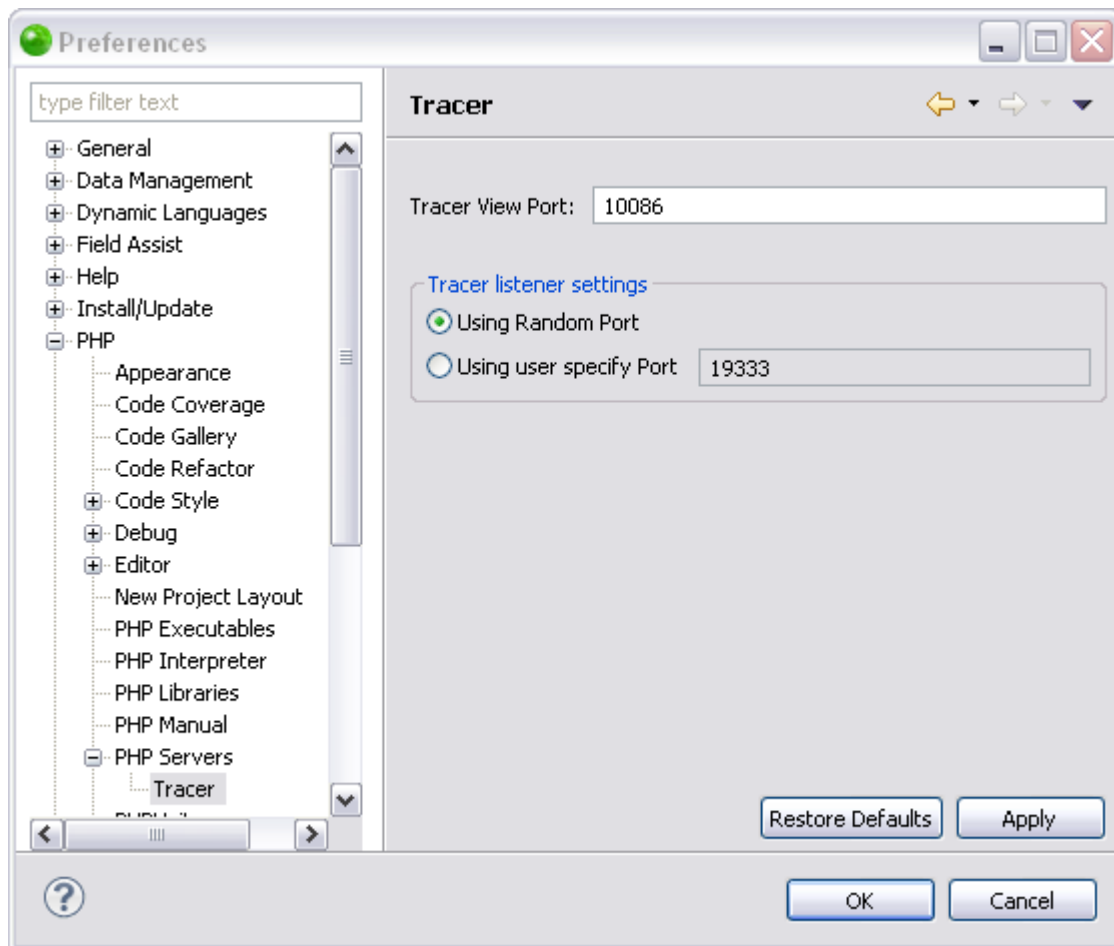
Editing the Tracer Listener Settings

This procedure describes how to change the port the Code Tracing functionality will use. By default Zend Studio will find the first available port.

Important Note:
The ports in the Tracer preferences should only be changed if the default port is blocked or in use by another application.

To edit the tracer listening port:

1. Go to **Window | Preferences | PHP | PHP Servers | Tracer.**

   The Tracer preferences page opens with the "Use a random port" option selected by default. The random port that has been selected is expressed in the "Use a user specified port" text field.

2. Select the "Use a user specified port" option and enter the port number in the text field.

3. Click **OK** to save the changes.

   The Tracer listening port has been changed.

If you would like Zend Studio to revert back to a default port select the "Use a random port" option. To revert back to the original port that was used click **Restore Defaults**.

PHPUnit Preferences

The PHPUnit Preferences page allows you to see your PHPUnit Library Path and set the PHPUnit's communication port.

 PHP Unit tests are a way of constantly testing your code to ensure the right output is being generated each time.

The PHPUnit Preferences page is accessed from **Window | Preferences | PHP | PHPUnit**.
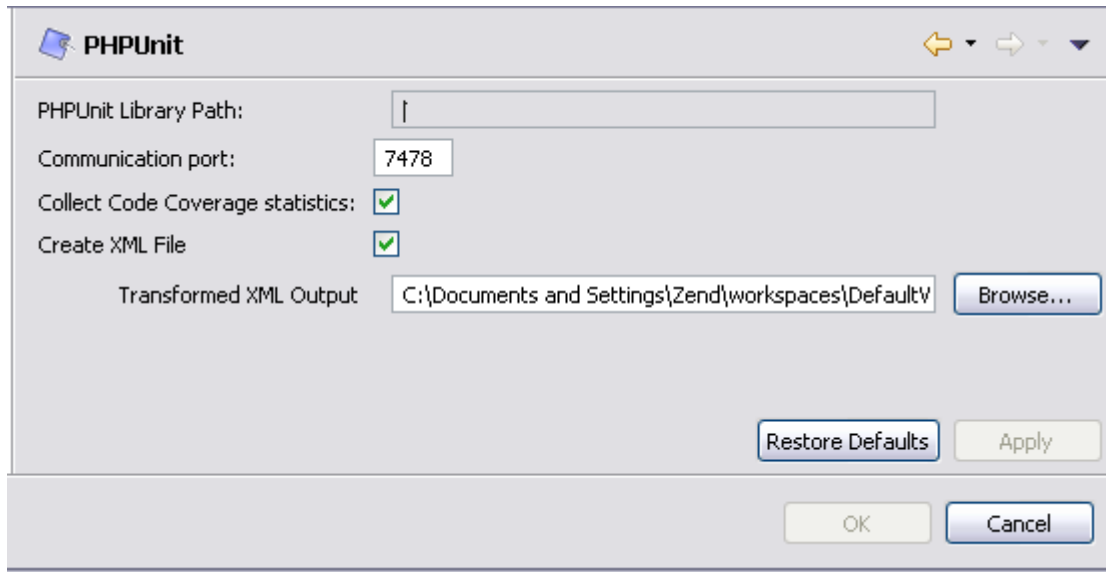
PHPUnit Preferences page

The PHPUnit Library Path displays the location of your PHPUnit Library. These settings cannot be changed.

Configuring Your PHPUnit Settings

To configure your PHPUnit settings:

1. Set the communication port which will be used when PHPUnit Tests are run. The default port number is 7478. Ensure that the selected port is not already in use.

2. Mark the 'Collect Code Coverage statistics' checkbox to enable code coverage while running unit tests.

3. Mark the 'Create XML File' checkbox to enable XML file generation and click Browse in the

Report Location option to select the location in which they will be created. XML files can later

be generated into reports from the PHPUnit Test results view by clicking . See Reporting on PHPUnit Test Results for more information.

4. Click **OK** to apply your settings.

Semantic Analysis Preferences

The Semantic Analysis feature enables warning and error messages to be displayed when Zend Studio detects possible errors or problems in your script.

The Semantic Analysis Preferences page is accessed from **Window | Preferences | PHP | Editor | Semantic Analysis Properties**.
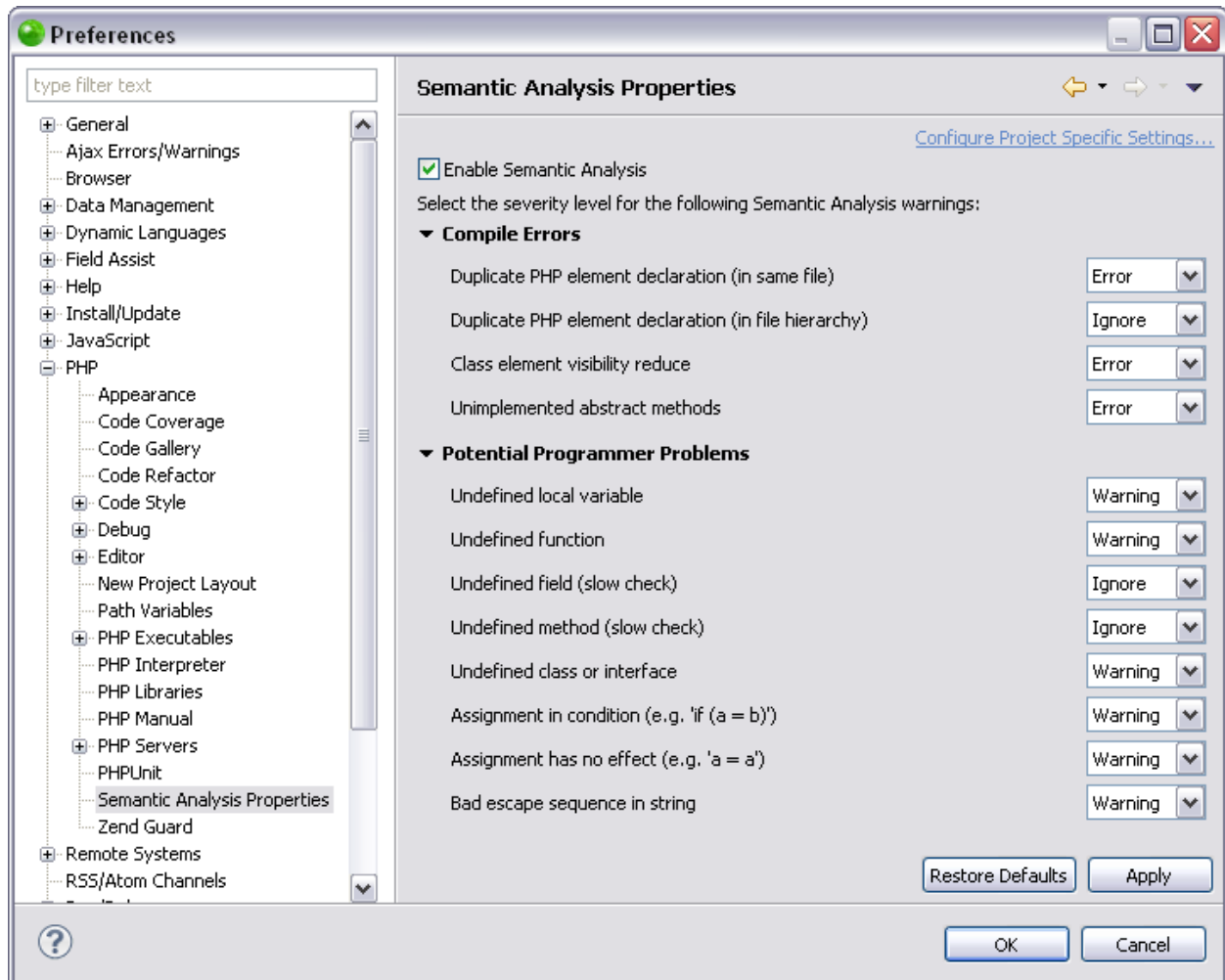
Semantic Analysis Preferences page

Enabling and Configuring Semantic Analysis for All Projects

To enable and configure Semantic Analysis for all projects:

1. Mark the 'Enable Semantic Analysis' checkbox.

2. Select the severity level displayed in error messages (warning, ignore or error) for a variety of occurrences, divided into the following categories:

   - Compile Errors
   - Potential Programmer Problems

To select a severity level for an event, click the ▶ arrow next to each header to display the possible options (if not displayed) and select the severity level from the drop-down list next to each option.

3. Click **Apply** to apply your settings.

   A prompt dialog is displayed indicating that a rebuild of the project must occur for the settings to take effect.

4. Click **Yes** to rebuild the project. Click **No** for a rebuild to be performed only when Zend Studio is restarted. Click **Cancel** to cancel the operation.

Compile Errors

Allows you to select the severity level (error, warning or ignore) for the following:

- Duplicate PHP element declaration (in same file)
- Duplicate PHP element declaration (in file hierarchy)
- Class element visibility reduce
- Unimplemented abstract methods

Potential Programmer Problems

Allows you to select the severity level (error, warning or ignore) for the following:

- Undefined local variable
- Undefined function
- Undefined field (slow check)
- Undefined method (slow check)
- Undefined class or interface
- Assignment in condition (e.g. 'if (a = b)')
- Assignment has no effect (e.g. 'a=a')
- Bad escape sequence in string

Applying Semantic Analysis Preferences Setting to a Specific Project

To apply Semantic Analysis Preferences settings to a specific project only:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.

   A Semantic Analysis Properties page will appear.

3. Select the required settings and click **Apply**.

    A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.

4. Click **Yes** to rebuild the project.

    -Or- click **No** for a rebuild to be performed only when Zend Studio is restarted.

    -Or- click **Cancel** to cancel the operation.

Note:
Semantic Analysis settings can also be configured for an existing project by right-clicking the project in PHP Explorer view and selecting **Properties | Semantic Analysis Properties**.

Configuring Tunneling Debug Preferences

Before initiating a debug session using tunneling, you must ensure the correct port numbers are configured in the Debug Preferences page.

This procedure describes how to configure your Debug preferences to enable tunneling.

To configure Debug Settings for Zend Studio:

1. Open the PHP Debug Preferences page by going to **Window | Preferences | PHP | Debug**.
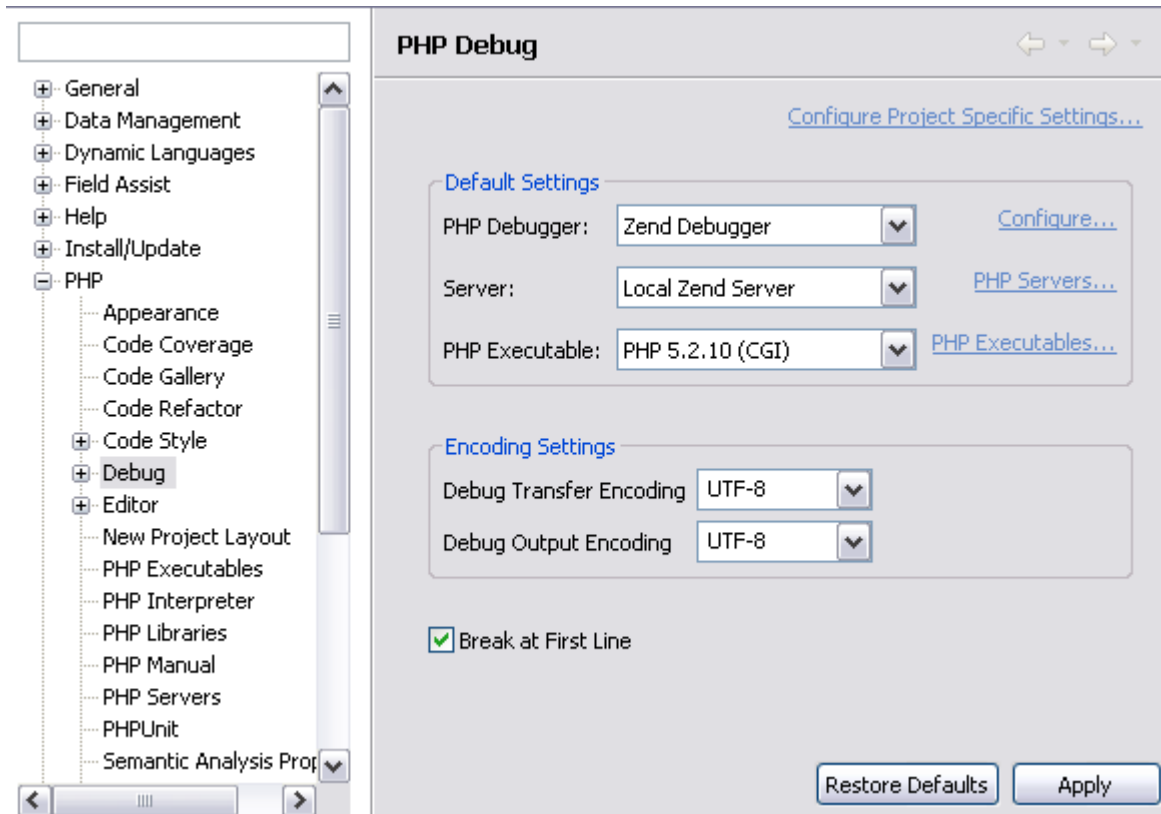
2. In the "Debug Options" section, ensure the following default settings are configured:
   - The Zend Debugger port number must be 10137.
   - In the "Advanced Zend Debugger Options" category, the Broadcast Port must be set to 20080.
   - In the Client Host/IP category, enter the Client Host/IP to which debugging results will be returned. Zend Studio will automatically search for and recognize the Client Host/IP, but entering a specific Host/IP will speed up the debugging process and decrease the likelihood of session time-outs.

The next step in creating a tunneling connection is to Set your Environment to be an Allowed Host.

Zend Guard Preferences

Zend Guard protects your commercial PHP 5 and PHP 5.3 applications from reverse engineering, unauthorized customization, unlicensed use and redistribution.

The Zend Guard Preferences page allows you to set Zend Guard's location so that it can be accessed by Zend Studio.

The Zend Guard Preferences page is accessed from **Window | Preferences | PHP | Zend Guard**.
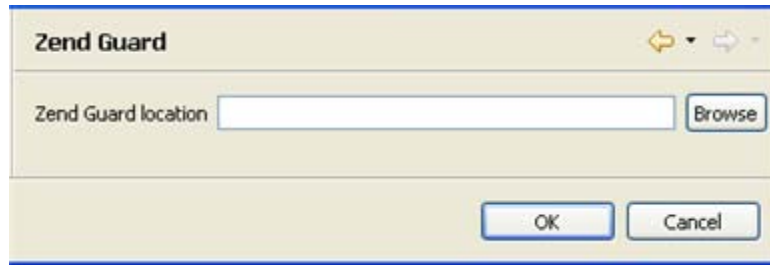
Zend Guard preferences page

To configure your Zend Guard's location:

1. Click **Browse**.
2. Find the location of your Zend Guard installation and click **Open**.
3. Click **OK** to apply your settings.

The connection will be made between Zend Studio and Zend Guard.

Zend Guard's functionality can now be accessed from Zend Studio by clicking the Encode Project icon

on the main toolbar -or- going to Project menu and selecting Encode Project.

See the Zend Guard Integration topic for more information.

For more information on Zend Guard and to download a trial version, please visit:

http://www.zend.com/en/products/guard.

Ajax Errors/Warnings Preferences

Ajax Tools is a feature that allows you to incorporate a Mozilla browser into Zend Studio. The Ajax Errors/Warning preferences will only appear when in the Web Browser Tools Perspective. To open it go to **Window | Open Perspective | Web Browser Tools**.

The Ajax Errors/Warnings preferences page is accessed from **Window | Preferences | Ajax Errors/Warnings**.
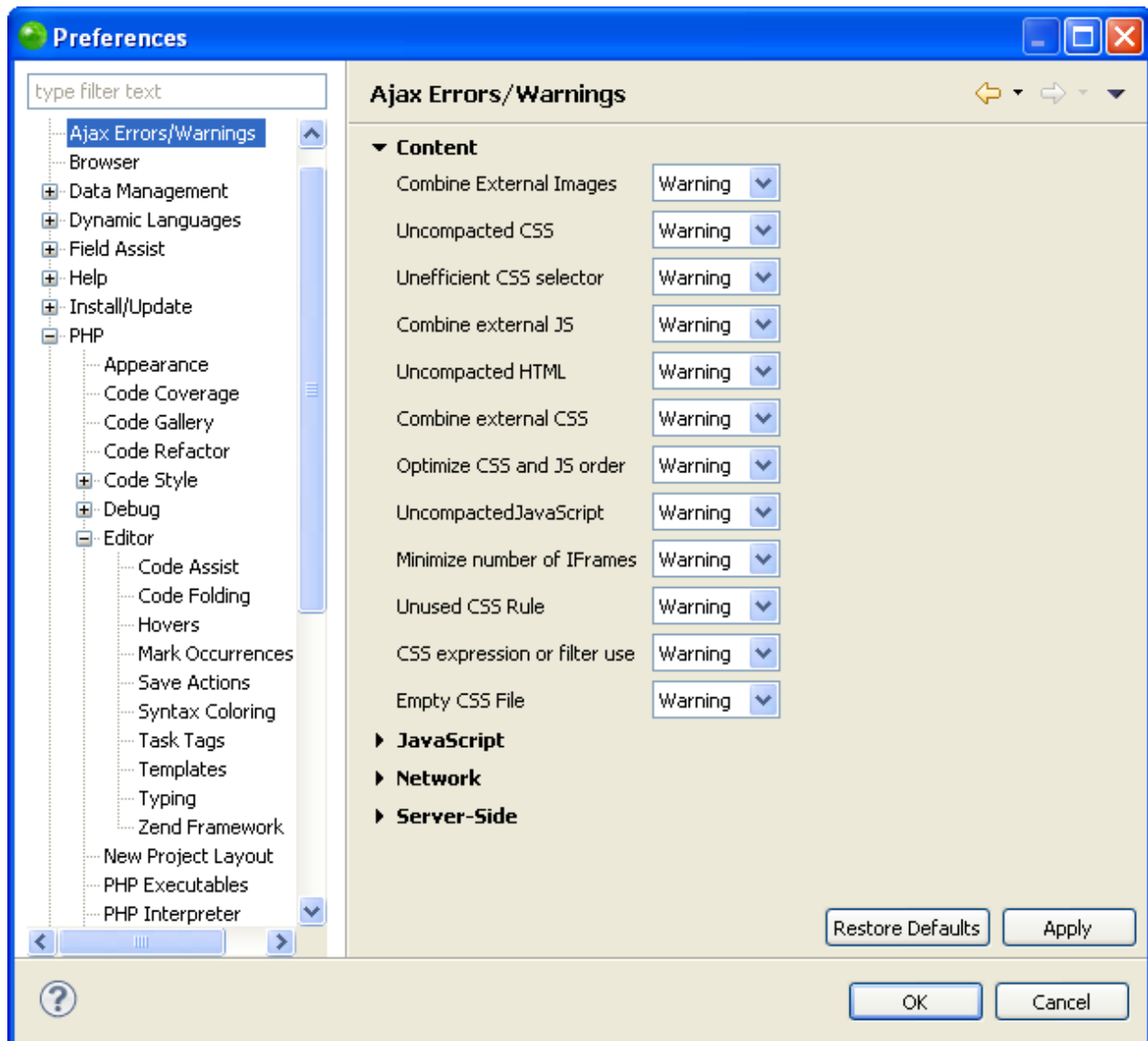
The Ajax preferences allow you to choose what kind of notification you would like for different events relating to:

- Content - Problems possibly affecting web site performance that can be found in HTML documents as well as resources such as CSS and images.

- JavaScript - Problems possibly affecting web site performance that are caused by JavaScript code.

- Network - Problems possibly affecting web site performance that are caused by the network's environment such as DNS, HTTP servers availability, and optimization.

For more information see the [Request Monitor View Rules](#).

The notification options are:

- Ignore - Problems will not be marked in the Browser Monitor view activity diagram.

- Info - Problems will be marked in the Browser Monitor view with a blue information icon with a question mark.

- Warning - Problems will be marked in the Browser Monitor view with a yellow warning icon with an exclamation point.

- Error - Problems will be marked in the Browser Monitor view with a red error icon.

Browser Preferences

The Internal Web Browser allows you to open a browser inside of your environment. The Internal Web

Browser is only available in the Web Browser Tools Perspective. To open it go to **Window | Open**

Perspective | Web Browser Tools. To open a web page in the browser click 🌐 from the main toolbar.

The Browser Preferences page is accessed from **Window | Preferences | Browser**.
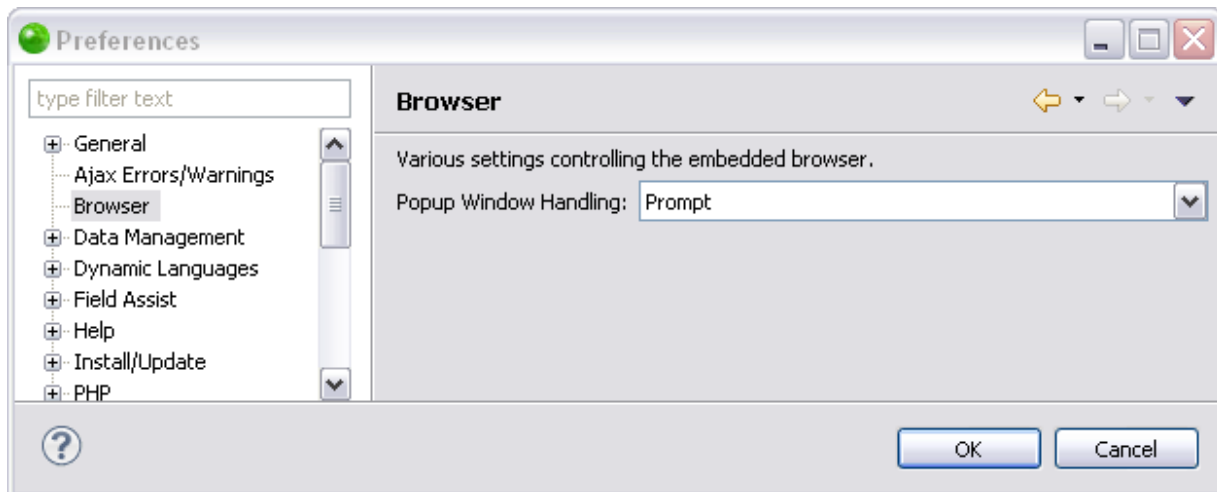
The Browser Preferences page allows you to control how popup Windows in your Internal Web Browser

will be handled.

The options for handling popup windows are:

- Open in new browser editor - Opens the popup window in a new browser editor allowing the use of the browser tools.
- Open as a dialog - Opens the popup window in a separate window.
- Ignore - Does not open the popup window.
- Prompt - Allows you to choose whether to Open as Editor, Open as Dialog, or Ignore for each popup window the browser detects.

PHP Project Properties

A project's properties pages allows you to configure various settings for a specific project.

Project's properties can be accessed by right-clicking the required project and selecting Properties -or- selecting the project and from the menu bar going to Project | Properties.

The properties available will depend on the resource selected. The following properties are available for PHP projects:

- Resource Properties
- Builders Properties
- Code Style Properties
    - Code Templates Properties
    - Formatter Properties
- PHP Build Path Properties
- PHP Debug Properties
- PHP Include Path Properties
- PHP Interpreter Properties
- PHP Task Tags Properties
- Project References Properties
- Run/Debug Settings Properties
- Save Actions Properties
- Semantic Analysis
- Task Repository - For more information see the Mylyn User Guide.
    - Commit Template
- Task Tags Properties
- Validation Properties
    - HTML Syntax
    - XSLT Validation
- WikiText

Note:

Properties pages are configured for a specific project only. To configure settings for all PHP projects, go to the PHP preferences pages (Window | Preferences | PHP).

Resource Properties

The Resource Properties page displays information about your project and allows you to set the project's text file encoding and line delimeters.

To access the Resource Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Resource**.
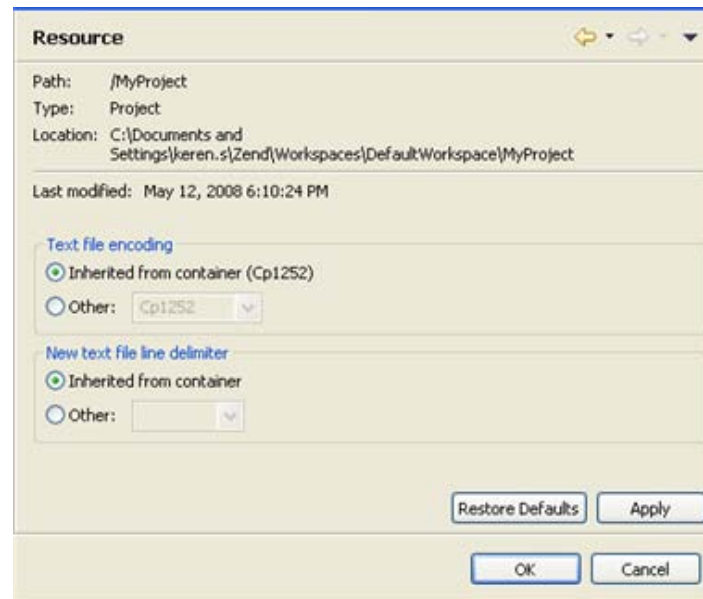
Resource Properties

The following information is displayed in the Resource Properties page:

- **Path**: For PHP Projects, this will be the project's name. If a file inside a project was selected, this would display the file's location within the project.
- **Type**: For PHP Projects, this will be 'Project'.
- **Location**: The project's location on the file system.

Configuring PHP Project Resource Properties

To configure PHP project resource properties:

1. Select options for:
   - Text File encoding:

     By default, this will be inherited from the container (this will be determined according to your local settings).

     If your text files use a different encoding scheme (because, for example, they originated from an external source), you can change the encoding by selecting the 'Other' option

and choosing the required encoding from the drop-down list.

- New text file line delimiter:

    This will define the line delimiter for new text files. Select to inherit from container or select 'Other' and choose the required option from the drop-down list.

2. Click **Apply** to apply your settings.

Builders Properties

The Builders Properties page allows you to configure the External Tool builders that are run when this project is built and add, remove, or reorder external tools in the build order. The Eclipse build process scans external resources which are referenced in a project so that their contents can be made available for operations such as Content Assist and Refactoring.

To access the Builders Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Builders Properties** .
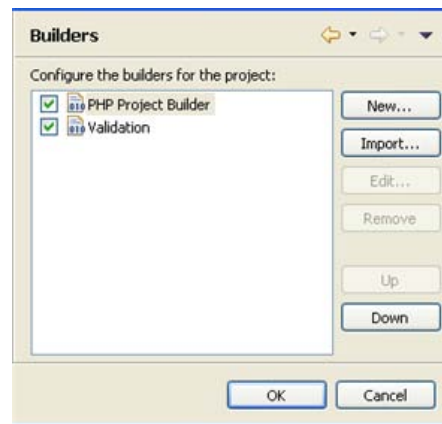
By default, Builders will be added according to the type of resources in your projects (e.g. if you add JavaScript libraries the JavaScript builder will be added.).

However, you can also configure your own external builders, if required.

Adding a New Builder

To add a new Builder:

1. Click **New**.

   The Choose Configuration Type wizard is displayed.
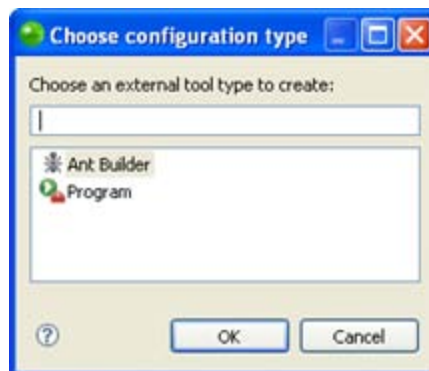
2. Select the required Builder type and click **OK**.

3. A wizard will be displayed allowing you to configure your Builder.

   See the [External Tools](#) topic in the Workbench User Guide for information on configuring your Builder.

Note:
Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/helios/index.jsp).

4. Click OK to apply your settings.

Code Style Properties

Code Templates Properties

The Code Templates Properties page allows you to configure the code and comments that are automatically created for different types of elements for the selected project.
See <u>Code Templates Preferences</u> for more information.

To access the Code Templates Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to **Project | Properties | Code Style | Code Templates**.

To configure Code Templates Properties for the project:
1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
   See <u>Task Tags Preferences</u> for more information on the settings available.
3. Click **OK** to apply your settings.

Default Code Templates Properties for all projects can be set in the Code Templates Preferences page (accessed by going to **Window | Preferences | PHP | Code Style | Code Templates** -or- by clicking the Configure Workspace Settings link on the properties page.)

Formatter Properties

Zend Studio can auto-format scripts to organize them into an easily readable format. The Formatter Properties page allows you to customize the way they are formatted for the project.

To access the Formatter Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Code Style | Formatter**.

To configure Formatter Properties for the project:
1. Mark the Enable project specific settings checkbox.
2. Configure the settings according to your preferences.
   See <u>Formatter Preferences</u> for more information on the settings available.

772

3.   Click Apply.

Default Formatter Properties for all projects can be set in the Formatter Preferences page, (accessed by going to **Window | Preferences | PHP | File Network** -or- by clicking the Configure Workspace Settings link on the properties page.

Semantic Analysis Properties

The Semantic Analysis Properties page allows you to configure Semantic Analysis Properties for your project. The Semantic Analysis feature enables warning and error messages to be displayed when Zend Studio detects possible errors or problems in your script.

To access the Semantic Analysis Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Semantic Analysis** -or- select the project and from the menu bar go to **Project | Properties | Semantic Analysis.**

Configuring Semantic Analysis Properties

To configure Semantic Analysis Properties for the project:

1. Mark the Enable project specific settings checkbox.
2. Configure the settings according to your preferences.

   See Semantic Analysis Preferences for more information on the settings available.

774

3. Click **Apply**.

A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.

4. Click **Yes** to rebuild the project.

-Or- click **No** for a rebuild to be performed only when Zend Studio is restarted.

-Or- click **Cancel** to cancel the operation.

Default Semantic Analysis Properties for all projects can be set in the Semantic Analysis Preferences page (accessed by going to (**Window | Preferences | PHP | Semantic Analysis Properties** -or- by clicking the Configure Workspace Settings link on the properties page).

PHP Build Path Properties

The PHP Build Path Properties page allows you to configure the project's PHP Build Path.

To access the PHP Build Path Properties page, right-click a PHP project in PHP Explorer view and select **Properties | PHP Build Path Properties** -or- select the project and from the menu bar go to **Project | Properties | PHP Build Path Properties** .

See Configuring a Project's PHP Build Path for more information.

Formatter Properties

Zend Studio can auto-format scripts to organize them into an easily readable format. The Formatter Properties page allows you to customize the way they are formatted for the project.

To access the Formatter Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Code Style | Formatter**.

To configure Formatter Properties for the project:

1. Mark the Enable project specific settings checkbox.
2. Configure the settings according to your preferences.
   See [Formatter Preferences](#) for more information on the settings available.
3. Click **Apply**..

Default Formatter Properties for all projects can be set in the Formatter Preferences page, (accessed by going to **Window | Preferences | PHP | File Network** -or- by clicking the Configure Workspace Settings link on the properties page.

PHP Debug Properties

The PHP Debug Properties page allows you to configure default settings used when debugging files in the project.

To access the PHP Debug Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | PHP Debug Properties .

To configure PHP Debug Properties for the project:

1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
    See Debug Preferences for more information on the settings available.
3. Click **OK** to apply your settings.

Default PHP Debug Properties for all projects can be set in the Debug Preferences page (accessed by going to **Window | Preferences | PHP | Debug** -or- by clicking the Configure Workspace Settings link on the properties page.)

PHP Include Path Properties

The PHP Include Path Properties page allows you to configure the project's <u>Include Path.</u>

To access the PHP Include Path Properties page, right-click a PHP project in PHP Explorer view and select **Properties | PHP Include Path** -or- select the project and from the menu bar go to **Project | Properties | PHP Include Path Properties** .

See <u>Configuring a Project's PHP Include Path</u> for more information.

PHP Interpreter Properties

The PHP Interpreter Properties page allows you to set the PHP version used for the project. This will affect the internal debugger, code analyzer and content assist options.

To access the PHP Interpreter Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | PHP Interpreter Properties** .

To configure PHP Interpreter Properties for the project:

1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
   See PHP Interpreter Preferences for more information on the settings available.
3. Click **Apply**.
   A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
4. Click **Yes** to rebuild the project.
   -Or- click **No** for a rebuild to be performed only when Zend Studio is restarted.
   -Or- click **Cancel** to cancel the operation.

Default PHP Interpreter Properties for all projects can be set in the Debug Preferences page (accessed by going to **Window | Preferences | PHP | Debug** -or- by clicking the Configure Workspace Settings link on the properties page.)

PHP Task Tags Properties

The PHP Task Tags Properties page allows you to add new task tags and edit existing ones. Tasks are used as reminders of actions, work to do or any other action required by the programmer. Task tags are strings used by Zend Studio to recognize when tasks are added in your script. Anything after these strings inside a comment will be recognized as a task.

To access the PHP Task Tags Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | PHP Task Tags Properties** .

To configure PHP Task Tags Properties for the project:

1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.

   See Task Tags Preferences for more information on the settings available.
3. Click **OK** to apply your settings.

Default PHP Task Tags Properties for all projects can be set in the Task Tags Preferences page (accessed by going to **Window | Preferences | PHP | Editor | Task Tags** -or- by clicking the Configure Workspace Settings link on the properties page.)

Project References Properties

The Project References Properties page allows you to set the projects which are referenced by your project. This affects actions such as opening and closing of projects. (i.e. if you close a project and re-open it, you will be prompted to also open the projects which it references.)

To access the Project References Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Project References Properties** .
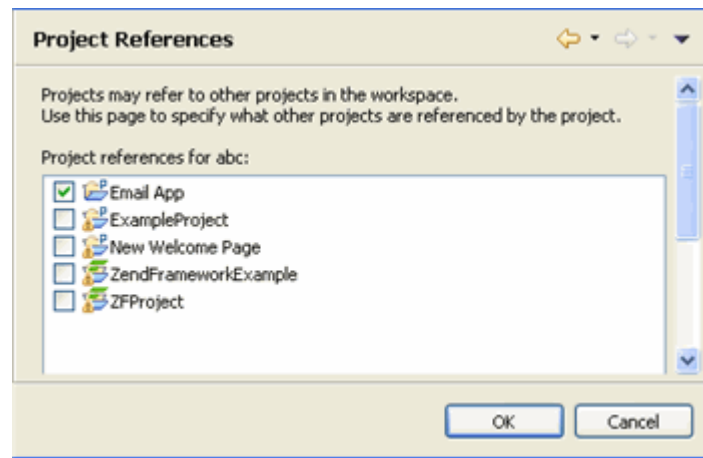
Project References

	To set the projects referenced by your project:
	1. Mark the checkbox of the projects that are referenced within your project.
	2. Click **OK**.

The selected projects are now specified as referenced by your project.

Remote Server Support Properties

The Remote Server Support Properties page displays information about your remote project, your Remote Connection Profile and allows you to enable a PHP project as a remote project.

To access the Remote Server Support Properties page go to **Project | Properties | Remote Server Support** - Or - **Select Properties | Remote Server Support** from the Right Click Menu in your project directory.
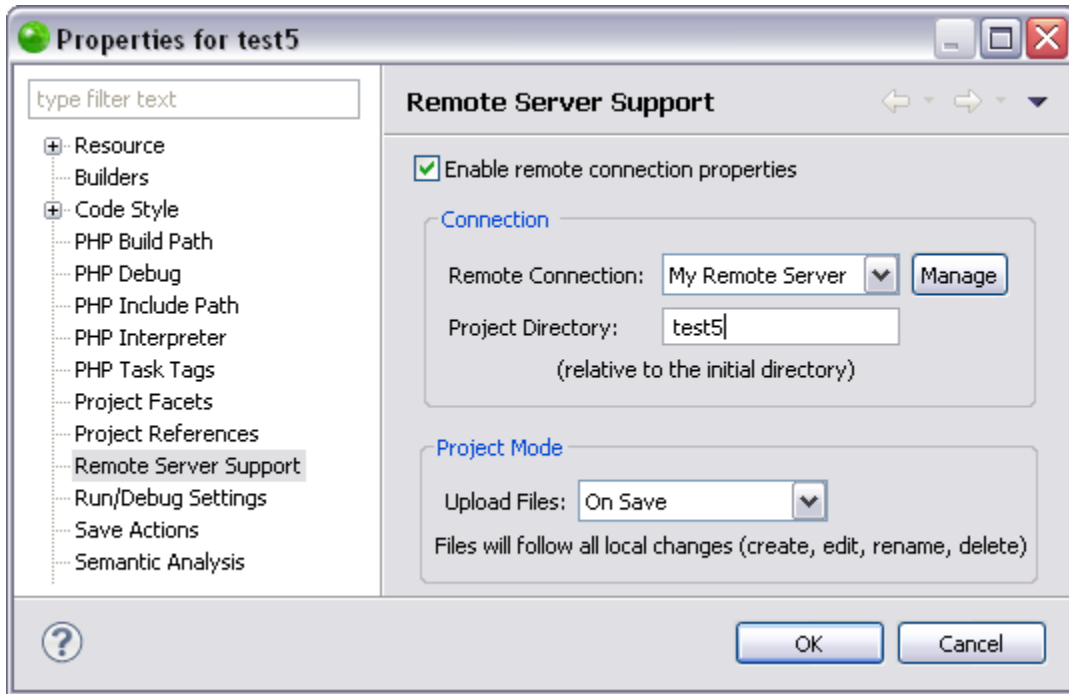
The Remote Server Support Properties page allows you to:

- Enable Remote Connection properties.
- Select a Remote Connection from the dropdown menu.
- Manage Remote Connection Profiles by clicking **Manage.**
- Select a Project Directory - A Project Directory is a folder within the Initial Directory that you would like to work with. This is the folder with which you will be transferring data when Uploading and Downloading files and folders to/from the remote server.
  The Project Directory is often the same as the project name.
- Select an Upload Files mode - The Upload Files dropdown menu allows you to select to transfer files to the remote server in three ways:
  - Manually - Data is only transferred to/from the remote server when you manually perform the procedure. This option is available for uploading and downloading data to/from the

> remote server. See <u>Uploading Manually</u> or <u>Downloading Manually</u> for more information.
>
> - On Save - Data is transferred to the remote server every time you save your project or perform a change event. This option is only available for uploading data to the remote server. See <u>Uploading On Save</u> for more information.
> - On Run - Data is transferred to the remote server before you run or debug your project. This option is only available for uploading data to the remote server. See <u>Uploading On Run</u> for more information.

All tasks you perform while working with Remote Server Support will appear in the Console view. Checking the Console view is the most efficient way to ensure that the action has been completed as you have requested. It will also show any problems that occurred during the task.

Refactoring History

The Refactoring History Properties page allows you to view all refactoring operations that have been performed on the project.

To access the Resource Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Refactoring History** -or- select the project and from the menu bar go to **Project | Properties | Refactoring History**.
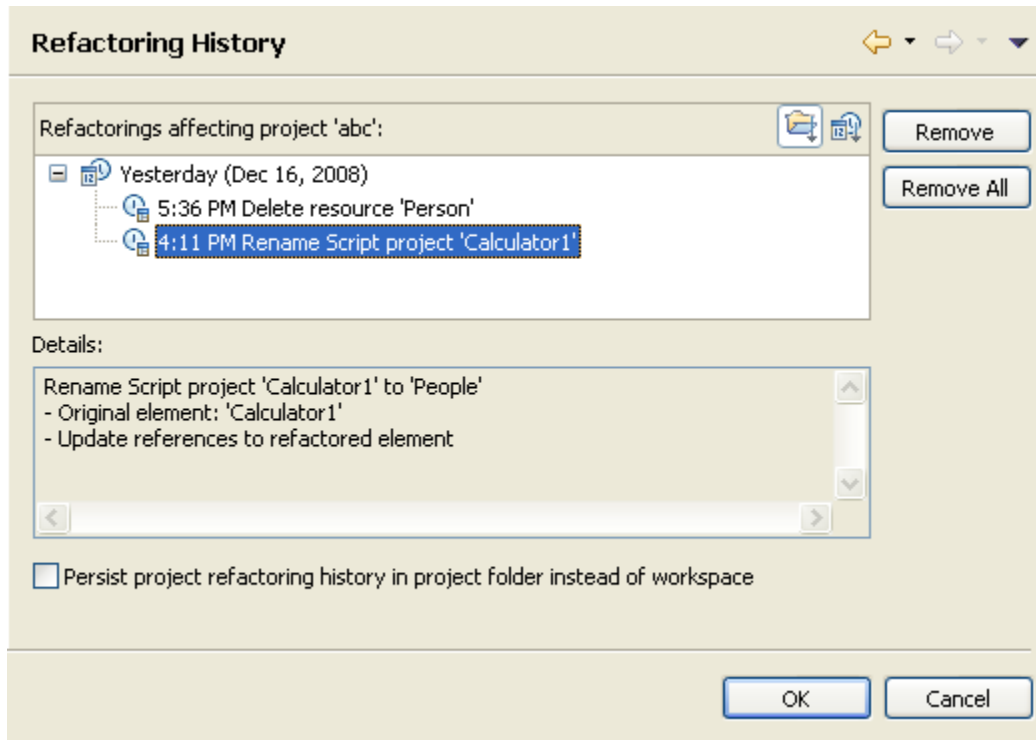
Refactoring History Properties page

Run/Debug Settings Properties

Note:

The Run/Debug Settings Properties page is not applicable for PHP projects.

Save Actions Properties

The Save Actions Properties page allows you to remove trailing whitespace from a file each time you save it.

To access the Save Actions Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Save Actions Properties** .

To configure Save Actions Properties for the project:

1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
   See [Save Actions Preferences](#) for more information on the settings available.
3. Click **OK** to apply your settings.

Default Save Actions Properties for all projects can be set in the Save Actions Preferences page (accessed by going to **Window | Preferences | PHP | Editor | Save Actions** -or- by clicking the Configure Workspace Settings link on the properties page.)

Task Tags Properties

Note:

The Task Tags Properties page is not applicable for PHP projects.

The Task Tags Properties page allows you to configure Task Tags for non-PHP files.

To access the Task Tags Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Task Tags Properties** .

Task Tag properties for PHP projects should be configured from the [PHP Task Tags Properties](#) page.

Validation Properties

The Validation Properties page allows you to specify the validators for your project.

To access the Validation Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Validation Properties** .

Amazon AWS Toolkit

AWS Toolkit Preferences

The AWS Toolkit Preferences page allows you to configure your Amazon EC2 credentials in Zend Studio.
Inserting these credentials in your Zend Studio will define which Amazon AWS Account your Amazon
AWS Toolkit for Zend Studio will work with.

The AWS Toolkit Preferences is accessed from **Window | Preferences | AWS Toolkit**.
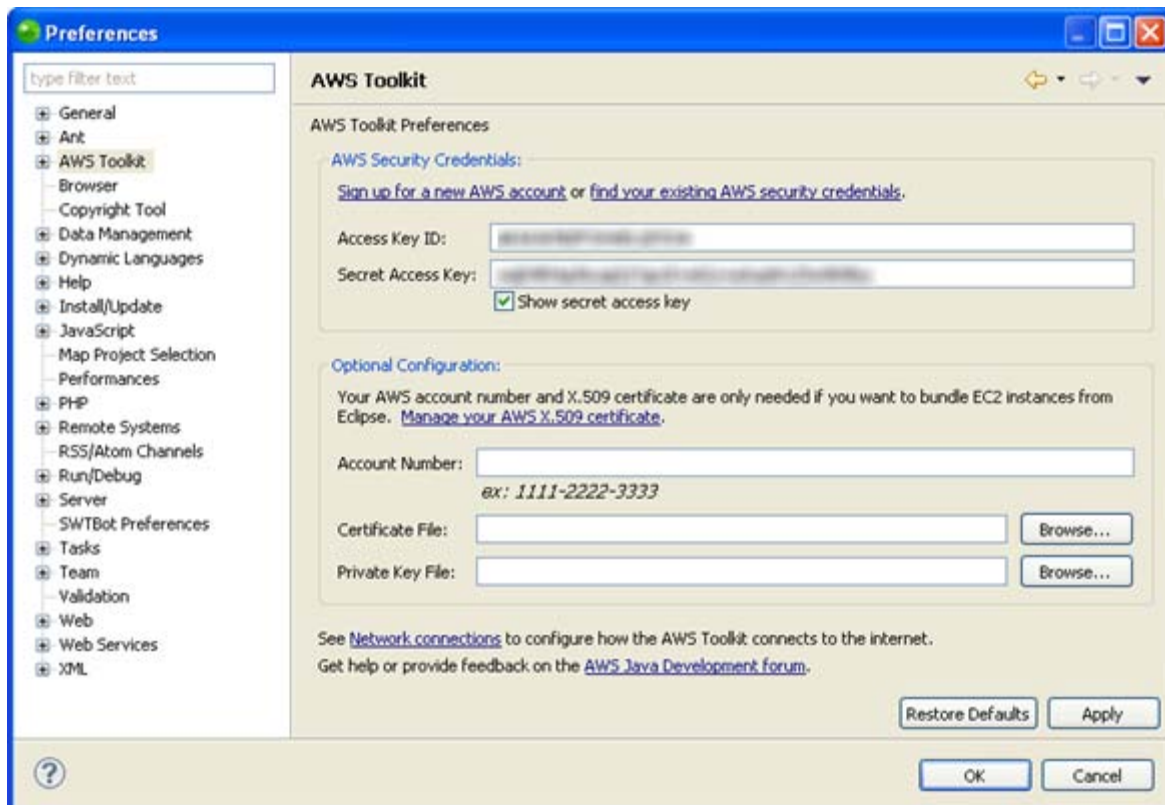
For information on working with the AWS Toolkit Preferences see Setting Up Your Amazon EC2 Instance
in Zend Studio.
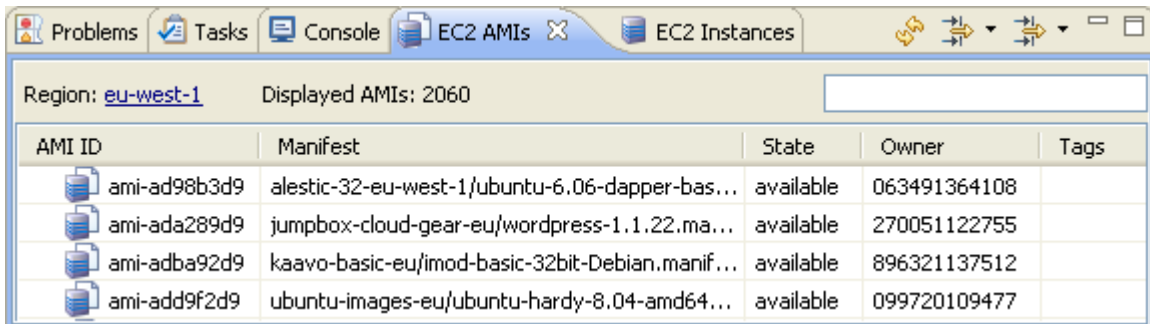
Amazon AWS Toolkit Views

The Amazon AWS Toolkit for Zend Studio includes views to allow you to see information about your AMIs, Instances and Security Groups from inside your Zend Studio. These views allow you to see the information you need to work with Amazon AWS Toolkit for Zend Studio.

EC2 AMIs View

The EC2 AMIs view allows you to see all the available AMIs in your region.
The EC2 AMIs view is accessible from **Window | Show View | Other | AWS Toolkit | EC2 AMIs.**
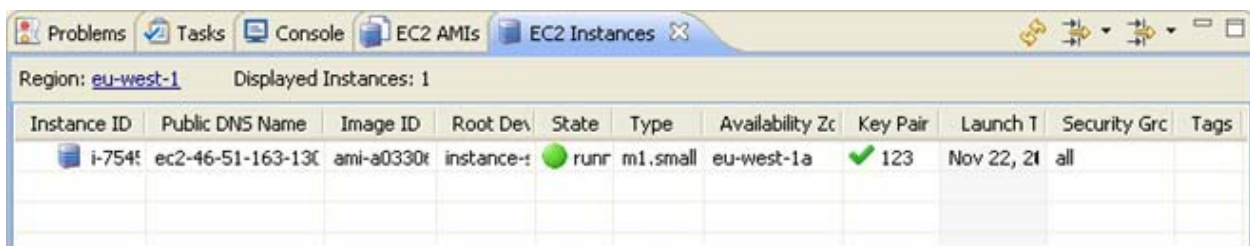
Find your AMI using the AMI ID column or the information from the provider in the Manifest column. You can also use the search text field to search for a specific AMI.
From the EC2 AMIs view you can launch an Amazon EC2 Instance in Zend Studio or create a Private Key. You can also see what your current region is.

EC2 Instances View

The EC2 Instances view allows you to see your current Instances. This includes running Instances, and Instances that were terminated during your most recent session.
The EC2 Instances view is accessible from **Window | Show View | Other | AWS Toolkit | EC2 Instances**.
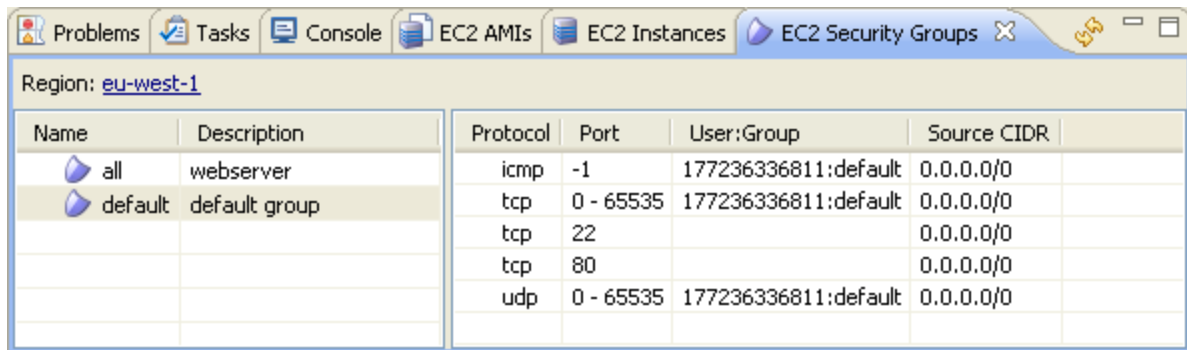
From the EC2 Instances view you can copy your Instance's Public DNS Name in order to open your Zend Server in a browser, and terminate your Instance.

EC2 Security Groups View

The EC2 Security Groups view allows you to see all the information about the Security Groups you have configured. For more information on Security Groups see Amazon's Getting Started Guide.
The EC2 Security Groups view is accessible from **Window | Show View | Other | AWS Toolkit | EC2 Security Groups**.

From the EC2 Security Groups view you can add a new Security Group in Zend Studio.

For more information on Amazon AWS Toolkit for Zend Studio see the AWS Toolkit for Eclipse User Guide by going to **Help | Help Contents | AWS Toolkit for Eclipse User Guide** or see the AWS Forums.

PHP Icons

The following is a list of icons representing different PHP elements which are displayed in the PHP Explorer view and in Content Assist lists:

PHP Elements

Icon	PHP Element
	PHP Project
	PHP File (included in the Build Path)
	PHP File (excluded from the Build Path)
	PHP File (containing warnings)
	PHP File (containing errors)
	Library
	Source Folder (included in the Build Path)
	Folder (excluded from the Build Path)
	PHP Class (public)
	PHP Class (default) (Only displayed in Content Assist list)
	PHP Interface
	PHP Method / PHP keyword (only displayed in Content Assist list)
	PHP Function / Method (public)
	PHP Function (private)
	PHP Function (protected)
	PHP Variable (public)
	PHP Variable (private)
	PHP Variable (protected)
	Static Constructor
	PHP Constructor / Constant
	Abstract Member
	Final Member
	Template (only displayed in Content Assist list)
	Unknown Type (only displayed in Content Assist list)
	Namespace

Zend Framework Elements

Icon   **Zend Framework Element**

Zend Framework project

Zend Controller (displayed in MVC Outline
view)

Zend Model (displayed in MVC Outline
view)

Zend View (displayed in MVC Outline view)

Other Icons

Icon   **PHP Element**

PHP project connected to a repository (e.g.
CVS / SVN)

Keymap

The following table displays a list of commonly used keyboard shortcuts which can be printed for quick access.

To see a full list of shortcuts and to configure the keymap, go to Window | Preferences | General | Keys.

To bring up a list of commonly used commands, press Ctrl+Shift+L in the Editor.

Show Key Assist	Ctrl + Shift + L			
	Action	**Shortcut**	**Action**	**Shortcut**
Source Editing	Add Block Comment	Ctrl+Shift+/	Copy Lines	Ctrl + Alt + Down
	Toggle Comment	Ctrl + /	Insert Line Below Current Line	Shift+Enter
	Generate Element Comment	Ctrl + Shift + J	Delete Line	Ctrl + D
	Toggle Mark Occurences	Alt + Shift + O	Quick Fix	Ctrl + 1
	Code Assist	Ctrl + Space	Format	Ctrl + Shift + F
	Context Information	Ctrl + Shift + Space	Format Active Elements	Ctrl + I
Navigation	Open Resource	Ctrl + Shift + R	Open Views List	Ctrl + 3
	Open Type	Ctrl + Shift + T	Show View	Alt + Shift + Q, Q
	Open Type Hierarchy	F4	Previous/Next View	Ctrl + F7 / Ctrl + Shift + F7
	Open Type in Hierarchy	Ctrl + Shift + H	Previous/Next Editor	Ctrl + F6 / Ctrl + Shift + F6
	Open Call Hierarchy	Ctrl + Alt + M	Quick Hierarchy	Ctrl + T
	Open Task	Ctrl + F12	Quick Outline	Ctrl + O
	Open Method	Ctrl + Shift + M	Find Text in Workspace	Ctrl + Alt + G
	Open Selection	F3	Go to Matching Bracket	Ctrl + Shift + P
Refactor	Move	Alt + Shift + V	Extract Variable	Alt + Shift + L
	Rename	Alt + Shift + R	Extract Method	Alt + Shift + M
Execution	Debug as PHP Script	Alt + Shift + D, H	Run as PHP Unit Test	Alt + Shift + X, U
	Debug as PHP Web Page	Alt + Shift + D, W	Terminate	Ctrl + F2
	Run as PHP Script	Alt + Shift + X, H	Use Step Filters	Shift+F5
	Run as PHP Web Page	Alt + Shift + X, W	Toggle Breakpoint	Ctrl + Shift + B

Useful Links

This page includes links to commonly used online reference information. This page can be used as a bookmarks area for web links. If you have a resource that you would like to see in this area send a mail to documentation@zend.com and the link will be added for the next version.

- PHP Manual (English)- **http://www.php.net**
- Zend Framework - **http://framework.zend.com**
- Zend Dev Zone - **http://devzone.zend.com/public/view**
- Zend Forums - **http://www.zend.com/en/forums**
- PHP Certification - **http://www.zend.com/en/services/certification**
- PHP Yellow Pages - **http://www.zend.com/en/store/education/certification/yellow-pages.php**
- Support Center - **http://www.zend.com/en/support-center**
- Knowledge Base Search - **http://kb.zend.com/**
- Refactoring Information - **http://refactoring.com/sources.html**
- HTML Tutorials - **http://www.w3schools.com/html**

Contribute to the Documentation

Your feedback is important to us. Therefore, at the bottom of each page is a link for sending e-mails directly to the Zend documentation team.

You can also let us know your thoughts and suggestions on Zend documentation by participating in our Zend Documentation Satisfaction survey:

http://www.zoomerang.com/survey.zgi?p=WEB226L47RWR8P

Support

Zend Technologies provides a wide range of resources for obtaining additional information and support, such as the Zend Support Center, the Zend Newsletter, and the Zend Developer Zone.

Zend Support Center

The Zend Support Center is a portal for information on all Zend Product related issues.

From the Zend Support Center (http://www.zend.com/en/support-center) you can access:

Zend Forums

Hosted user forums for the Zend product user community. See what other users have to say and get answers directly from the Zend Support team.

Visit: http://www.zend.com/en/forums

Zend Support Knowledge Base

The Zend Knowledge Base contains an extensive range of articles on commonly encountered problems, troubleshooting, tips and workarounds.

Search the Knowledge Base for any Zend product related issue at http://kb.zend.com/.

Online Documentation

The Zend Product Online Documentation Center is an easily browseable and searchable resource for accessing the most updated information on using all Zend Products.

Visit: http://www.zend.com/en/resources/zend-documentation/

Open a Support Ticket

If you did not find the answer to your question in any of the Support resources, you can open a ticket with the Zend Support team, who will answer each ticket on an individual basis.

This can be done through https://www.zend.com/en/helpdesk/newticket.php.

Note:
More information on specific Zend Studio component errors is sometimes recorded in Zend Studio's error log. This is located in a '.log' file, created in the .metadata folder of your workspace. (e.g. C:\Documents and Settings\Bob\Zend\Workspaces\DefaultWorkspace\.metadata\.log.) The information in this log can help the Zend Support team to diagnose a specific problem.

Zend Newsletter

Sign up for Zend PHP email updates for the hottest updates, special promotions and useful developer information.

To sign up, log in to your Zend account at https://www.zend.com/en/user/myzend, enter your email address and click Subscribe.

Zend Developer Zone

The Zend Developer Zone is the leading resource center for PHP developers, where you can learn about PHP and meet the experts.

The Zend Developer Zone features the following:

- The PHP 5 Info Center
- Articles and Tutorials
- PHP and PEAR News Weeklies
- Worldwide Job Classifieds

Visit: http://devzone.zend.com

Feedback

Send feedback, questions and comments on the Online Help and Documentation to:

documentation@zend.com

Registering Your License

Once Zend Studio is installed, all its features will be available for a 30 day trial. At the end of the 30 day trial, the professional features will be disabled. These include refactoring, getter and setter generation, Zend Framework integration, profiling, PHPUnit testing, Zend Server Integration , debugging through tunneling, advanced code formatting, code galleries, and more.

Additional new features for the most recent version of Zend Studio are listed in What's New in Zend Studio 9.x

To enjoy the full range of Zend Studio functionality, you should register a valid license.

To register a valid license and activate all Zend Studio features, go to Help | Register and enter your username or order #, and license key.

You can purchase a Zend Studio license from the Zend Store at http://www.zend.com/store/software/studio or from the menu bar go to Help | Purchase a License.

Index