

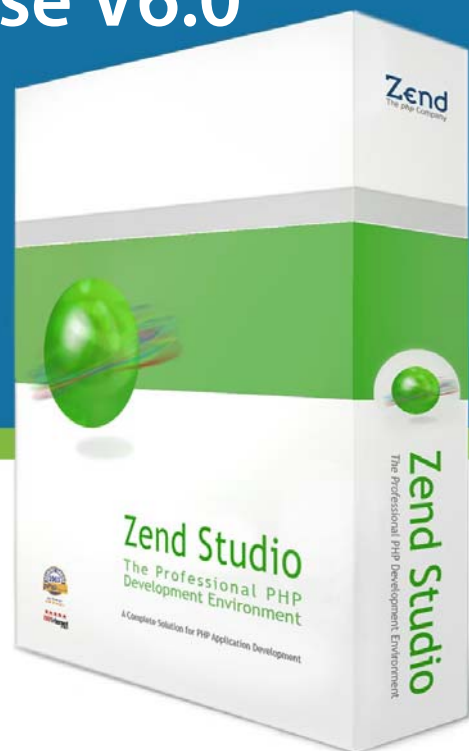


The PHP Company

User Guide

Zend Studio for Eclipse V6.0

By Zend Technologies, Inc.



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.

This document is a printed version of the Zend Studio for Eclipse Online Help. For full information, please see the Online Help contained within the Zend Studio for Eclipse application.

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

© 1999-2008 Zend Technologies Ltd. All rights reserved.

Zend Studio for Eclipse User Guide issued January 2008.

Product Version: Zend Studio for Eclipse

DN: ZSN-UG-010108-003

Table of Contents

Disclaimer.....	2
Zend Studio for Eclipse User Guide.....	8
What's New	8
Getting Started	10
Quick Start	10
Getting Started	10
Workbench	11
Creating a PHP Project	11
Creating a PHP File	11
PHP Debugging.....	12
PHPUnit	12
Source Control	13
Refactoring	13
Perspectives of Interest.....	13
Switching from Zend Studio to Zend Studio for Eclipse.....	14
Project/File Creation	14
Debugging	15
Profiling.....	17
Source Control - Subversion (SVN)	19
Source Control - CVS	20
FTP Connectivity.....	21
Database Connection	22
Tunneling	23
Basic Tutorials	24
Creating Projects and Files.....	24
Working with Code Assist	25
Working with the Debugger.....	30
Working with the Refactoring Feature.....	37
Working with the Profiler	40
Working with PHPUnit Testing	46
Working with CVS.....	53
Working with SVN.....	60
Concepts.....	66
PHP Support	66
Code Assist.....	68
Function Parameter Hints	69
Code Assist for Include Statements.....	69
Configuring Code Assist.....	70
Automatic Completion.....	70
Matching Brackets	70
Code Folding.....	70

Syntax Coloring.....	71
Bookmarks	72
Commenting Code	72
phpDoc Block Comments	72
PHPDocs.....	73
Hover Support	74
PHP Manual Integration.....	75
Refactoring.....	76
Code Galleries.....	76
Zend Framework Integration.....	77
PHP - Java Bridge Support.....	79
JavaScript Support	81
PHP/HTML WYSIWYG	82
Data Tools Platform	84
Real Time Error Detection.....	84
Code Analyzer	85
Zend Platform Integration	85
PHPUnit Testing	87
Profiling	88
Debugging.....	90
Zend Debugger Toolbar.....	91
Path Mapping.....	92
Include Paths.....	94
Tunneling (Communication Settings)	95
CVS.....	95
SVN.....	96
Local History.....	96
Zend Guard Integration.....	96
RSS Feeds.....	97
WSDL - Web Services Description Language	97
Update Manager.....	98
i5/OS Edition Extras.....	98
Tasks.....	102
Easy File Creation.....	102
Creating a PHP File within a Project.....	102
Creating a PHP File Outside of a Project.....	103
Opening an External File.....	103
Migrating From Zend Studio.....	105
Migrating Projects From Zend Studio.....	105
Migrating Keymaps from Zend Studio.....	107
Using Code Assist.....	108
Using Templates.....	108
Finding and Replacing	110
Searching for PHP Elements.....	111

Opening PHP Elements	112
Generating Getters and Setters	113
Formatting Code	116
Using Code Folding	117
Adding Comments	118
Adding PHP DocBlock Comments	119
Creating a PHPDoc	120
Using Smart Goto Source	123
Using Refactoring	124
Renaming Files	124
Renaming Elements	126
Moving Files	128
Organizing Includes	129
Using Code Galleries	131
Inserting Code Snippets into your Script	131
Creating and Editing Code Gallery Entries	132
Interacting with Code Gallery Sites	134
Creating Zend Framework Projects	138
Using Java Bridge	139
Creating Java Objects	139
Adding External Java Archives	142
Changing Projects' JREs	143
Using JavaScript	147
Using the PHP/HTML WYSIWYG Perspective	149
Creating and opening HTML files in the WYSIWYG Editor	149
Inserting HTML Objects	150
Configuring HTML Properties	151
CSS Editing	155
Using the Data Tools Platform	158
Creating a Database Connection Profile	158
Connecting to a Database	160
Viewing and Editing Table Content	162
Creating and Executing an SQL Query	163
Integrating with Zend Platform	164
Defining a Zend Platform Server	164
Accessing and Using the Platform Event List View	165
Using PHPUnit Testing	168
Creating a PHPUnit Test Case	168
Running a PHPUnit Test Case	169
Creating a PHPUnit Test Suite	171
Running a PHPUnit Test Suite	172
Reporting on PHPUnit Test Results	173
Using the Profiler	175
Profiling a PHP Script	175

Profiling a PHP Web Page	178
Profiling a URL	180
Profiling Using the Zend Debugger Toolbar	181
Using the Debugger	181
Debugging a PHP Script	181
Debugging a PHP Web Page	185
Debugging a URL	188
Running and Analyzing Debugger Results	188
Debugging Using the Zend Debugger Toolbar	190
Adding a Server Location Path Map	193
Adding Elements to a Project's Include Path	194
Setting Up Tunneling	196
Setting Up a Tunneling Server	196
Setting your Zend Studio for Eclipse to be an Allowed Host	198
Configuring Platform to Auto Detect Studio Settings	201
Activating Tunneling	202
Using CVS	202
Configuring a CVS Connection	203
Importing Projects from CVS	204
Uploading Projects to CVS	205
Using SVN	206
Configuring an SVN Connection	206
Importing Projects From SVN	208
Uploading Projects to SVN	209
Using Local History	210
Comparing Files	210
Replacing Files	211
Restoring Deleted Files	211
FTP and SFTP Support	212
Creating an FTP/SFTP Connection	212
Viewing and Editing Files on an FTP Server	213
Viewing and Editing FTP Files in your Workspace	215
Integrating with Zend Guard	218
Encoding Projects Using Zend Guard	218
Opening and Editing Zend Guard Projects in Zend Studio for Eclipse	221
Viewing RSS Feeds	221
Working with WSDL	222
Generating WSDL Files	222
Incorporating WSDL Files	226
Reference	229
PHP Perspectives and Views	229
PHP Perspective	230
PHP Debug Perspective	240
PHP Profile Perspective	250

PHP Perspective Menus	257
File	258
Edit.....	266
Source	267
Refactor	268
Navigate.....	269
Search	271
Project	272
Run	273
Window	275
Help.....	277
PHP Perspective Main Toolbar	278
PHP Preferences	281
PHP Preferences Page.....	282
Appearance Preferences	283
Code Analyzer Preferences.....	284
Code Coverage Preferences.....	286
Code Gallery Preferences.....	288
Debug Preferences.....	289
Editor Preferences	293
Formatter Preferences.....	301
Installed JREs Preferences	307
Path Variables Preferences.....	308
PHP Executables Preferences.....	309
PHP Interpreter Preferences.....	310
PHP Manual Preferences	312
PHP Servers Preferences.....	313
PHPUnit Preferences.....	315
Profiler Preferences.....	316
Templates Preferences	317
Zend Guard Preferences	319
Keymap	320
Useful Links	322
Index	323

Zend Studio for Eclipse User Guide

What's New

Zend Studio for Eclipse comprises a comprehensive package of features for full PHP development.

The following new features are available in Zend Studio for Eclipse:

- **Basic Features**
 - [PHP4 and PHP5 Support](#)
 - [Syntax Coloring](#)
 - [Code Assist](#) (elements, PHPDoc, parameter hints, include statements)
 - [Templates](#) (PHP, PHPDoc, New File)
 - [Code Folding](#) (Classes, functions and PHPDoc)
 - [Real time error detection](#)
 - [Bookmarks](#)
 - [Smart Goto Source](#)
 - [Hover Support](#)
 - [Automatic Insertion](#) (brackets, braces, PHPDoc)
 - [Matching Bracket](#)
 - [Comment / Uncomment PHP code](#)
 - [PHP \(Project\) Explorer View](#)
 - Open resource (File / function)
 - [PHP Manual Integration](#)
 - [Search PHP element](#)
 - [File / Project / PHP Inspectors \(Outlines\)](#)
 - [Code Formatting](#) (brackets and indentation)
 - [Find & Replace in Files](#)
 - [Tasks View](#)
 - [Project Include Path](#)
 - [Problems View](#)
 - [Easy Open File System File](#)
 - [New PHP Elements Wizards](#) (Class , Interface)
 - [Getter and Setter Generation](#)

- Source Control
 - [Local History](#)
 - [CVS](#)
 - [Subversion](#)

- **Refactoring**
 - [Move](#)
 - [Rename](#)
- **Organize Includes**
- **PHPUnit Testing**
- **Debug / Profile**
 - [Web Server Debugging](#)
 - Text Encoding Support
 - [Web Servers Management](#)
 - File Content Transfer (Use Local/Server Copy)
 - SSL Communication
 - Toolbar Support
 - [PHP Executable Profiler](#)
 - [Web Server Profiler](#)
 - Easy Debug
- **Code Coverage**
- **Miscellaneous**
 - [Advanced PHP code analyzer](#)
 - [RSS Reader](#)
 - Web Services Editor
 - [Web Services Support \(Wizard and Inspection\)](#)
 - [PHPDocumentor](#)
 - [HTML WYSIWYG](#)
 - [Check for Updates](#)
 - Java Code Assist in PHP Code
- **Platform Integration**
 - Basic Integration (Open Platform GUI)
 - Events List View
 - Debug / Profile Events
 - Allow Host/Tunneling
 - Platform API
- **Framework Integration**
 - Code Assist
 - Framework Project
 - MVC View
 - MVC Code Generation

- [Code Galleries Support](#)
- [Zend Guard Integration](#)
- [Zend Studio Migration](#)
 - [Zend Studio Keymaps](#)
 - [Import Zend Studio Projects](#)
- [FTP and SFTP Support](#)

Getting Started

The Getting Started section provides different levels of information according to the following descriptions:

[Quick Start](#) - This Quick Start describes how to easily locate and use commonly used features such as Creating projects and files, Debugging PHP code, using Source Control (CVS, SVN), Refactoring and more.

[Basic Tutorials](#) - These tutorials will demonstrate common processes along with examples and practical information on how to implement the information in order to improve your coding environment.

[Switching from Zend Studio to Zend Studio for Eclipse](#) - A quick guide for Zend Studio 5.5 users on how to perform commonly used functionality in Zend Studio for Eclipse 6.0.

Quick Start

Zend Technologies is happy to release an Eclipse based version of the traditional Zend Studio.

The following Quick Start page will help newcomers and (even) our veteran users familiarize themselves with this new version.

The Features covered in the Quick Start are:

[Getting Started](#)

[PHP Unit](#)

[Workbench](#)

[Source Control](#)

[Creating a PHP Project](#)

[Refactoring](#)

[Creating a PHP File](#)

[Perspectives of Interest](#)

[PHP Debugging](#)


Getting Started

Before launching, Zend Studio for Eclipse will ask you to select a Workspace where all projects will be created and stored.

You can use the default Workspace suggested or click Browse to select a different Workspace.

See the Workbench User Guide for more on Workspaces.

When Zend Studio for Eclipse is first launched, a Welcome Page will open containing links to actions and tutorials to help you get started with Zend Studio for Eclipse, as well as information on Zend Studio for Eclipse's features and functionality.

To start using Zend Studio for Eclipse, close the Welcome Page by clicking the X icon  in the Welcome tab in the top-left corner.

Workbench

The Workbench is a window displaying perspectives, views and menu bars through which different operations can be performed.

See the Workbench User Guide for more on how to customize your Workbench.

Creating a PHP Project

A project is a group of files and resources, which will be displayed in a tree in the Navigator and PHP Explorer views.



To create a new PHP project:

Go to the Menu Bar and select File | New | PHP Project.

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

New Project Wizard Note:

Leaving the 'Use Default' checkbox under the Project contents category selected will create a default library within the workspace, with the project name. Not using the default will allow you to add the contents to the directory of your choice, provided you have already created a directory before starting. (Recommended to prevent current project files from being stored with other project files).


Creating a PHP File



To create a new PHP file:

Go to the Menu Bar and select File | New | PHP File.

-Or- In PHP Explorer view, right-click the folder in which you would like to create your file and select New | PHP File

-Or- Click the New PHP File icon on the toolbar . This will create a file outside of a project.

PHP Debugging

The Zend Debugger detects and diagnoses errors in PHP code situated on local or remote servers:



To debug a PHP script situated on your workspace:

1. Set breakpoints at the relevant locations in your script by double-clicking the marker bar to the left of the relevant line. A blue ball will appear to indicate that a breakpoint has been set.
2. Go to the main menu and select Run | Open Debug Dialog -or- right-click the file in PHP Explorer view and select Debug As | Open Debug Dialog.
3. To create a new configuration, double-click the 'PHP Script' category.
4. Under the Debugger Location category, choose whether you want to debug locally using the internal debugger (PHP executable) or remotely using your server's debugger (PHP Web Server).
5. Enter all other information and click Apply and Debug.



To debug a PHP web page situated on a server:

1. Go to the main menu and select Run | Open Debug Dialog -or- right-click the file in PHP Explorer view and select Debug As | Open Debug Dialog.
2. To create a new configuration, double-click the 'PHP Web Page' category.
3. Enter the required information and click Apply and Debug.

Debugging Preferences can be configured from the [Debug Preferences page](#), which can be accessed from Window | Preferences | PHP | Debug.

See the [Debugging](#) topic for more on Debugging.

PHPUnit

A PHPUnit is a testing framework to write and run tests on PHP code. A test file can be created for each class, function and file. PHPUnits allow PHP developers to incrementally build test suites to constantly review progress and detect unintended side effects.



To create and run a PHPUnit Test Case:

1. In PHP Explorer view, right-click the file you want to test and select New | PHP Unit Test Case.
2. Fill in the required information in the New PHPUnit Test Case dialog.
3. Click Finish to create your Test Case file.
4. Edit the test functions in your new PHPUnit Test Case file by writing appropriate tests for the relevant functions.
5. Run the PHPUnit Test by going to Run | Run As | PHPUnit from the Menu Bar -or- right-clicking the file in PHP Explorer view and selecting Run As | PHPUnit.

Source Control

Zend Studio for Eclipse includes a built-in component for CVS (Concurrent Versions System) and SVN (Subversion).

Before accessing a repository, make sure that a CVS or SVN server is already configured on the host machine.



To configure access to a repository through Zend Studio for Eclipse:

1. On the main menu go to Window | Show View | Other.
2. Select either CVS or SVN repositories.
3. From the CVS/SVN view toolbar, select the Add CVS / SVN Repository button.
4. Fill in the location and authentication details and press Finish.

CVS / SVN functionality can then be accessed by right-clicking on or within the relevant file or project and selecting Team, Compare with, Replace with, or Source.

Refactoring

Refactoring is the process of renaming or moving selected resources in a 'smart' way while maintaining all the relevant links between files and elements. Refactoring automatically makes all relevant changes to your code.



To move / rename a resource:

1. Select the required file in PHP Explorer view -or- select the required element in the editor window.
2. Right-click and select Refactor | Move (files only) / Rename.
3. Select the resource's new location / name.

During the refactoring process, a preview screen will display showing the changes made.

Perspectives of Interest

To open a perspective go to Window | Open Perspective, select "Other" to view a full list of perspectives.

- **PHP (default)** - This is the perspective that will open by default in Zend Studio for Eclipse. It allows you to manage and create PHP projects and files.
- **PHP Debug** - Allows you to manage and track the debugging process.
- **CVS / SVN Repository Exploring** - Allows you to manage and view your source control.
- **PHP/HTML WYSIWYG** - Allows you to design, edit and preview your HTML/PHP files using drag and drop capabilities.
- **PHP Profile** - Allows you "profile" and analyze file running times.
- **Database Development** - Allows you to view and manage your database content. Zend Studio for Eclipse allows connection with several types of databases.

Switching from Zend Studio to Zend Studio for Eclipse

The following 'Switching from Zend Studio to Zend Studio for Eclipse' guide is intended for users who have worked with the traditional Zend Studio and want to learn how to perform Zend Studio 5.5 tasks in the new Zend Studio for Eclipse 6.0.

Contents:

[Project/File Creation](#)

[Source Control - CVS](#)

[Debugging](#)

[FTP Connectivity](#)

[Profiling](#)

[Database Connection](#)

[Source Control - Subversion \(SVN\)](#)



Project/File Creation

To create a new PHP project:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> From the main toolbar select Project New Project. The New Project Wizard dialog box will appear. Enter a name for the new project. The location is updated accordingly. Click Next to define specific properties for the new project. Click Finish. 	<ol style="list-style-type: none"> Go to File Menu and select New PHP Project -Or- In the PHP Explorer View, right-click and select New PHP Project. The New Project wizard will open. Enter a name for your new project into the Project Name field. Click Finish.

See the [Creating Projects and Files](#) tutorial in the Zend Studio for Eclipse Online Help for more information.

To create a new file:


Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> From the Menu Bar select File New File -Or- Click the New File icon  on the toolbar. A new PHP file will open in the editor. 	<p>To create a new PHP file not associated with a project:</p> <ol style="list-style-type: none"> Click the new Easy PHP File icon on the toolbar  -Or- In PHP Explorer view, right-click and select New Untitled PHP Document. A new PHP file, by default called PHPDocument1, will open in the editor. <p>To create a new PHP file within a project:</p> <ol style="list-style-type: none"> In PHP Explorer view, select the Project within which you would like to place the file. Right-click and select New PHP File -or- go to File on the Menu Bar and select New PHP

	<p>File.</p> <ol style="list-style-type: none"> 3. The PHP File creation dialog will be displayed. 4. Enter the name of the file and click Next. 5. Click Finish.
--	--



See [Easy File Creation](#) in the Zend Studio for Eclipse Online Help for more information.

Debugging

Note:



By creating a debug launch configuration in Zend Studio for Eclipse 6.0, you can easily rerun the debug session with the settings specified by clicking the arrow next to the debug button  on the toolbar and selecting your launch configuration.


To debug a PHP script using Zend Studio's internal debugger:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Open the Preferences window by selecting Tools Preferences from the main menu. 2. Select the Debug tab. 3. From the Debug Server Configuration area of the Debug tab, select 'internal' from the Debug Mode category. 4. Click Apply and OK. 5. In the main toolbar, click Go  to start the Debugger. -or- from the Menu Bar select Debug Go. 	<ol style="list-style-type: none"> 1. Click the arrow next to the debug button  on the toolbar and select Debug As PHP Script. -Or- In PHP Explorer view, right-click the required file and select Debug As PHP Script.

See [Locally Debugging a PHP Script](#) in the Zend Studio for Eclipse Online Help for more information.



To debug a PHP script using your server debugger:


Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Open the Preferences window by selecting Tools Preferences from the main menu. 2. Select the Debug tab. 3. From the Debug Server Configuration area of the Debug tab, select 'server' from the Debug Mode category. 4. Enter the URL of the server on which you want to Debug your files. 5. Click Apply and OK. 6. In Zend Studio's main toolbar, click Run  to start the Debugger. 	<ol style="list-style-type: none"> 1. Click the arrow next to the debug button  on the toolbar and select Open Debug Dialog –or- In PHP Explorer view, right-click and select Run Open Debug Dialog. A Debug dialog will open. 2. Double-click the PHP Script option to create a new debug configuration and enter a name for it. 3. Select the PHP Web Server option under the Debugger Location category and select your server from the list.

	<ol style="list-style-type: none"> 4. Under PHP File, click Browse and select your file. 5. Click Apply and then Debug. <p>Note: The next time you want to run this debug session, click the arrow next to the debug button  on the toolbar and select your launch configuration.</p>
--	---

See [Remotely Debugging a PHP Script](#) in the Zend Studio for Eclipse Online Help for more information.

To debug applications on a server:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. From the Menu Bar, select Debug Debug URL. The Debug URL dialog will appear. 2. Enter the URL of the file/application which you would like to debug. 3. Select whether to use the local copy or server copies of the files. 4. Click OK. 	<ol style="list-style-type: none"> 1. Click the Debug URL button  on the main toolbar -or- go to Run Debug URL. The Debug URL dialog will appear. 2. In the 'Open Browser at' field, enter the URL of the first page that should be debugged. 3. Click Debug.
	<p>Note: In Zend Studio for Eclipse 6.0, creating a Debug launch configuration gives you access to advanced Debugging options, and allows you to easily re-run past Debug sessions using the same settings.</p> <p>To create and execute a Debug Launch Configuration for debugging applications on a server:</p> <ol style="list-style-type: none"> 1. Click the arrow next to the debug button  on the toolbar and select Open Debug Dialog –or- select Run Open Debug Dialog. A Debug dialog will open. 2. Double-click the PHP Web Page option to create a new debug configuration. 3. Enter a name for the new configuration. 4. Select your server from the PHP Server drop-down list.


	<ol style="list-style-type: none"> 5. Under PHP File, click Browse and select your file. 6. Select the Advanced tab. 7. In the 'Source Location' category, select whether to use the local copy or server copies of the files. 8. Click Apply and then Debug. <div style="background-color: #cccccc; padding: 5px;"> <p>Note: The next time you want to run this debug session, click the arrow next to the debug button  on the toolbar and select your launch configuration.</p> </div>
--	---

See [Debugging a PHP Web Page](#) in the Zend Studio for Eclipse Online Help for more information.

Profiling

Note:
In addition to Profiling applications on a server, Zend Studio for Eclipse 6.0 includes the option to profile PHP Scripts situated on your workspace using the internal debugger or your server debugger. See [Profiling a PHP Script](#) in the Zend Studio for Eclipse Online Help for more information.

To profile applications on a server:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. From the Menu Bar, select Debug Profile URL. The Profile URL dialog will appear. 2. Enter the URL of the file/application which you would like to Profile. 3. Select whether to use the local copy or server copies of the files. 4. Click OK. 	<ol style="list-style-type: none"> 1. Click the profile URL button  on the main toolbar -or- from the Menu Bar, go to Run Profile URL. The Profile URL dialog will appear. 2. In the 'Open Browser at:' field, enter the URL of the page that should be profiled. 3. Click Profile. <div style="background-color: #cccccc; padding: 5px; margin-top: 10px;"> <p>Note: In Zend Studio for Eclipse 6.0, creating a Profiling launch configuration gives you access to advanced Profiling options, and allows you to easily re-run past profile sessions using the same settings.</p> </div>

To create and execute a Profile Launch Configuration for profiling applications on a server:

1. Click the arrow next to the Profile button



on the toolbar and select Open Profile Dialog –or- select Run | Open Profile Dialog.

A Profile dialog will open.

2. Double-click the PHP Web Page option to create a new Profile configuration.
3. Enter a name for the new configuration.
4. Select your server from the PHP Server drop-down list.
5. Under PHP File, click Browse and select your file.
6. Select the Advanced tab.
7. In the 'Source Location' category, select whether to use the local copy or server copies of the files.
8. Click Apply and then Profile.

Note:

The next time you want to run this profile session, click the arrow next to the profile button




on the toolbar and select your launch configuration.

See [Profiling a PHP Web Page](#) in the Zend Studio for Eclipse Online Help for more information.

Source Control - Subversion (SVN)

See the Subversive User Guide for more information on Subversion.

To define Subversion connectivity:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Open the Preferences window by selecting Tools Preferences from the Menu Bar. 2. Select the Source Control tab. 3. Select 'Subversion' in the Source Control drop-down list. This will cause all source control preferences and menu options to enable Subversion rather than CVS operations. 4. Configure any required Subversion settings. 5. Click Apply and OK. 	<ol style="list-style-type: none"> 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  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 your repository. 4. Click Finish. Your SVN repository will be added to the SVN Repository view. <p>Note: Once you have created your repository connection,</p>

See [Configuring an SVN Connection](#) in the Zend Studio for Eclipse Online Help for more information.

Checking out a Module from SVN

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Set Subversion to be your Source Control default by following the steps under 'To define Subversion connectivity', above. 2. Go to Tools Subversion Checkout. 3. Enter the repository details. 3. Set the Checkout Options. 4. Click OK. 	<ol style="list-style-type: none"> 1. Create an SVN repository connection by following the steps under 'To define Subversion connectivity', above. 2. Go to File Import Projects from SVN and click Next. 3. Select your repository and click Next. 4. Select the 'Use an existing module' option and select the required module/project to check out. 5. Click Finish.

See [Importing Projects From SVN](#) in the Zend Studio for Eclipse Online Help for more information.

To perform Subversion commands (e.g. update/commit) on an SVN module:


Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Set Subversion to be your Source Control default by following the instructions under 'To define Subversion connectivity', above. 2. Right-click the required file/project in the Project tab -or- open the file in an editor and right-click. 3. From the right-click menu, select Subversion and the required action. 	<ol style="list-style-type: none"> 1. Create an SVN repository configuration by following the steps under 'To define Subversion connectivity', above. 2. Right-click the required file/project in PHP Explorer -or- open the file in an editor and right-click. 3. From the right-click menu, select Team and the required action.

See the "[Making Changes, Comparing Changes, and Committing Changes](#)" section of the [Working with SVN Tutorial](#) in the Zend Studio for Eclipse Online Help for more information.

Source Control - CVS

See the 'Working with CVS' topic in the Workbench User Guide to learn more about working with CVS.

To define CVS connectivity:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Open the Preferences window by selecting Tools Preferences from the Menu Bar. 2. Select the Source Control tab. 3. Select 'CVS' in the Source Control drop-down list. This will cause all source control preferences and menu options to enable CVS rather than Subversion operations. 4. Configure any required CVS settings. 5. Click Apply and OK. 	<ol style="list-style-type: none"> 1. From the Menu Bar, open the CVS perspective by selecting Window Open Perspective Other CVS Repository Exploring from the Menu Bar. 2. Click the Add CVS Repository button  on the view's toolbar -or- right-click within the CVS 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. 4. Click Finish to create your connection.

See [Configuring a CVS Connection](#) in the Zend Studio for Eclipse Online Help for more information.

Checking out a Module from CVS

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Set CVS to be your Source Control default by following the instructions under 'To define CVS connectivity', above. 2. From the Menu Bar, go to Tools CVS Checkout. 3. Enter the details in the Checkout dialog. 4. Click OK. 	<ol style="list-style-type: none"> 1. Go to File Menu and select Import Projects from CVS and click Next. 2. Select your repository and click Next. A 'Select Resource' dialog will appear. 3. Select your project (if necessary, expand the nodes until you see it) and click Finish. A 'Check Out As' dialog will appear. 4. Click Finish.


See [Importing Projects from CVS](#) in the Zend Studio for Eclipse Online Help for more information.

To perform CVS commands (e.g. update/commit) on a CVS module:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Set CVS to be your Source Control default by following the instructions under 'To define CVS connectivity', above. 2. Right-click the required file/project in the Project tab -or- open the file in an editor and right-click. 3. From the right-click menu, select CVS and the required action. 	<ol style="list-style-type: none"> 1. Right-click the required file/project in PHP Explorer -or- open the file in an editor and right-click. 2. From the right-click menu, select Team and the required action.

See the "[Making Changes, Comparing Changes, and Committing Changes](#)" section of the [Working with CVS tutorial](#) in the Zend Studio for Eclipse Online Help for more information.

FTP Connectivity**To configure an FTP root:**

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> 1. Go to the main menu and select File Add FTP Server -Or- from the File Manager's File System tab, open the right-click menu and select Add FTP Server. 2. The Configure FTP Server dialog will appear. 3. Enter your FTP connection details and click OK. The new FTP Icon appears in the file system. 	<ol style="list-style-type: none"> 1. Switch to the Remote Systems view (tabbed with the PHP Explorer view) and click the 'Define a connection to remote system' button  on the view's toolbar. The New Connection dialog will appear. 2. Select 'FTP Only' and click Next. 3. Enter your Remote FTP connection details and click Finish. Your connection will be created and listed in the Remote Systems view.



See [Creating an FTP/SFTP Connection](#) in the Zend Studio for Eclipse Online Help for more information.

Database Connection

Note:


Zend Studio for Eclipse 6.0 allows connection to a variety of database types.

To create an SQL server connection:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> From Studio's File Manager, click the SQL tab . Right-click and select Add Server. The Add SQL Server dialog will open. Enter the required SQL Server Settings. Click OK. Your Server Tree will be added to the SQL tab. 	<ol style="list-style-type: none"> Click the Create New SQL Connection icon  on the toolbar. The New JDBC Connection Profile wizard opens. Select a driver from the drop-down list and enter the required information. Click Finish. Your new connection profile will be added to your databases list in the Data Source Explorer view.

See [Creating a Database Connection Profile](#) in the Zend Studio for Eclipse Online Help for more information.

To connect to your SQL database server:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> Create an SQL server connection by following the instructions under 'To create an SQL server connection', above. Select the SQL tab  from Studio's File Manager. Double-click the Server you want to connect to. The SQL Database tree will be displayed. 	<ol style="list-style-type: none"> Create an SQL server connection by following the instructions under 'To create an SQL server connection', above. Open the Database Development perspective by going to Window Open Perspective Other Database Development. In the Data Source Explorer view, expand the SQL Databases node, right-click your SQL server connection profile and select Connect. The SQL Database tree will be displayed.

See [Connecting to a Database](#) in the Zend Studio for Eclipse Online Help for more information.


To view the contents of a database table:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> Connect to your SQL database server by following the steps under 'To connect to your SQL database server', above. Expand the server tree until you see the required table. Double-click the table -or- right-click the 	<ol style="list-style-type: none"> Connect to your SQL database server by following the steps under 'To connect to your SQL database server', above. Expand the server tree until you see the required table. Double-click the required table or right-click


<p>table and select Show Table Data.</p> <p>4. In the Results in Page selector at the top of the Data Display, select the number of results you wish to display per screen. The Data Display will show the contents of the table according to the selected resolution.</p>	<p>it and select Data Edit. The table will open in a database editor displaying all the data within the table.</p>
--	--

See [Viewing and Editing Database Table Content](#) in the Zend Studio for Eclipse Online Help in the Zend Studio for Eclipse Online Help for more information.

Tunneling

Note:
 Zend Studio for Eclipse 6.0 allows you to define tunneling connections to a number of servers. To select a server to connect to using a tunneling connection, define the server following the instructions below, click the arrow next to the tunneling icon on the toolbar  and select the required server from the list.

To configure tunneling preferences in Zend Studio:

Zend Studio 5.5	Zend Studio for Eclipse 6.0
<ol style="list-style-type: none"> Go to Tools Tunneling Settings. Define the relevant tunneling settings. Click Connect. 	<p>To configure your Server for Tunneling with Zend Studio for Eclipse:</p> <ol style="list-style-type: none"> Open the PHP Server Preferences page by going to Window Preferences PHP PHP Servers from the Menu Bar. Click New to define a New Server (or Edit if the server has already been defined). Enter any relevant settings in the Server, Path Mapping and Platform Integration tabs. Click Next to go to the next tab. In the Tunneling Settings tab, check the "Enable Tunneling" option and enter all necessary information. Click Finish or OK. <p>You can now use this server configuration to connect to your server using a tunneling connection by clicking the Tunneling icon on the toolbar .</p>

See [Setting Up a Tunneling Server](#) in the Zend Studio for Eclipse Online Help for more information.

Note:
 To configure a tunneling connection between Zend Studio and your server, settings also need to be configured on your server.
 See [Setting Up Tunneling](#) in the Zend Studio for Eclipse Online Help for full instructions on setting up a Tunneling connection with your server.

Basic Tutorials

The Basic Tutorials section contains short tutorials on popular tasks that can be performed with Zend Studio for Eclipse. 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:

[Creating Projects and Files](#)

[Working with PHPUnit testing](#)

[Working with Code Assist](#)

[Working with CVS](#)

[Working with the Debugger](#)

[Working with SVN](#)

[Refactoring](#)

[Working with the Profiler](#)

Creating Projects and Files

The purpose of this tutorial is to guide you through the steps involved in creating PHP Projects and files.

Contents:

[Creating a PHP Project](#)

[Creating a PHP File](#)

Creating a PHP Project



To create a new PHP Project:

1. Go to File Menu and select New | PHP Project.
-Or- In PHP Explorer view, right-click and select New | PHP Project.
2. The New Project wizard will open.
Enter a name for your new project into the Project Name field.
The default file system location is displayed (grayed out) in the Directory field under Project contents. If you want to create the project and its contained resources in a different location, clear the Use default checkbox and specify the new location.
3. Click Finish to complete the creation of your project.

The new project will be listed in PHP Explorer view.

Creating a PHP File

Creating a PHP file within Zend Studio for Eclipse will automatically add PHP tags to the script, and allow you to fully utilize Zend Studio for Eclipse's PHP functionality.




To create a new PHP file within a project:

1. In PHP Explorer view, right-click your project and select New | PHP File -or- select your project and go to File Menu | New | PHP File.
2. Enter the File Name and click Finish.

An editor window will appear with the following basic PHP code:

```
1 <?php
2
3 ?>
```

3. Add your code to the new file.

4. Save the file by clicking the Save button  on the toolbar.
5. To complete your project, create more PHP files by repeating steps 1-7.




To create a new PHP file outside of a project:

1. Click the new Easy PHP File icon on the toolbar .

An editor window will appear with the following basic PHP code:

```
1 <?php
2
3 ?>
```

2. Add your code to the new file.
3. Save the file by clicking the Save button  on the toolbar.
4. To complete your project, create more PHP files by repeating steps 1-7.

Once you have created all your files, you can utilize all of Zend Studio for Eclipse's tools in order to debug, profile, test and monitor them.

Working with Code Assist

The purpose of this tutorial is to teach you how to use Zend Studio for Eclipse's Code Assist function in order to write code quickly, easily, and accurately.

Contents

[Purpose and Usage](#)

[Activating Code Assist](#)

[Making Code Assist Elements Available Within the Same](#)

[Scope](#)

[Function Parameter Hints](#)

[Class Type Hints](#)








[Configuring Code Assist](#)

Purpose and Usage

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

A list of possible code elements appears 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:

-  Reserved PHP Words
-  Functions
-  Templates
-  Classes
-  Interfaces
-  Constants
-  Variables (public)

Code Assist works with the following elements: PHP Classes, Functions, Variables, Constants, Keywords, Interfaces, Attributes, Values, Nested Functions, Names and Syntax, as well as all user defined Classes, Functions, and Constants.

Note:

Code Assist works with both PHP and HTML.

Activating Code Assist

By default, the Code Assist options will be automatically displayed once the first few characters of the code have been entered.



The following procedure demonstrates using Code Assist:

1. Create a new PHP File called 'File1'.
2. On the line beneath the opening PHP tag, type "def".
3. The Code Assist window will be displayed with a list of suitable code completion options:

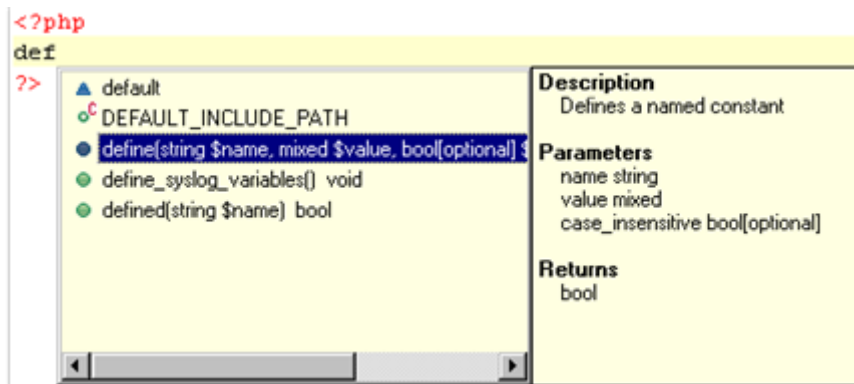


Figure 1 - Code Assist Box

4. Double-click the first define function from the Code Completion window -or- select it and press Enter.

"define()" appears on the edit line.

Note:

If the Code Assist window does not open automatically, place your cursor at the required location and press Ctrl+Space.

To enable the Code Assist window to open automatically, go to the [Code Assist Preferences page](#), accessed from Window | Preferences | PHP | Editor | Code Assist and mark the 'Enable auto-activation' checkbox.

Function Parameter Hints

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



The following procedure demonstrates using the Function Parameter Hint feature:

1. Place your cursor between the parentheses of the above function call: "define()"
2. Press Ctrl+Shift+Space.

A function parameter hint box will be displayed indicating the types of parameters that should be inserted between the parentheses.

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

Figure 2 - Parameter Hint

Making Code Assist Elements Available Within the Same Scope

Added Code - Available within the same function and file

Elements within the same scope (e.g. within the same project, file or function) will be available for use with Code Assist.

Variables belonging to different scopes are not available to each other via Code Assist.



The following procedure demonstrates using Code Assist for inserting elements within the same function and file:

1. Edit your PHP File ('File1') so that it contains the following code:

```
<?php
define('continent','africa');
$control = '';
$mail = 'int@eclipse.org';
function media() {
    $music = '';
    $messenger = '';
    $
        /*----- Location_1*/
    }
    $
        /* -----Location_2*/
?>
```

2. Place the cursor at the "\$" marked by "Location_1". This is within function "media".
3. Type the letter "m". The Code Assist window will be displayed with the variables "\$messenger" and "\$media", which were defined within the function.
Note that the variable \$mail (not within the scope of "media()") is not available.
4. Next, place the cursor at the "\$" marked by "Location_2".
5. Type the letter "m". The Code Assist window will be displayed with the variable \$mail, which is within the same file.
Note that media's variables - \$music and \$messenger - are not within the function 'media' and so are not displayed.
6. Select 'mail' from the Code Assist window to insert it into the script.

Added Code - Available Within the Same Project

Code elements defined in one file are also available for use by other files within the same project.



The following steps demonstrate using Code Assist for inserting elements within the same project:

1. Within the same project as "File1", create a new PHP file called "File2".
2. On the line beneath the opening PHP tag, type "def" and press Ctrl+Space to activate Code Assist. Double-click one of the define options.
3. Between the parentheses, type "cont" and press Ctrl+Space to activate Code Assist. The element 'continent', defined in "File1", will be available.
4. Double-click 'continent' to enter it into your code.

When the element is highlighted, Code Assist displays the original location of the code element, its value ('africa') and all other information available.

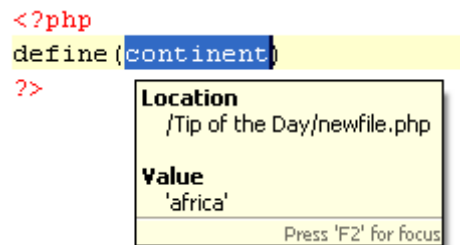


Figure 3 - Element Information

Class Type Hints

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



To see and trial this feature:

1. Create a new PHP file with the following code:

```
<?php
function getClass() {
    return new Test();
}
class Test {
    function printValues($a, $b) {
        echo "Values: $a, $b";
    }
}
$myVar = getClass();
/* @var $myVar Test */
$myVar->
?>
```

2. Place your cursor after '\$myVar->' (on the line above the closing PHP tag) and press Ctrl+Space to activate Code Assist. Code Assist will open with the function defined in 'Test' class (printValues(\$a, \$b)). Double click it to enter it into your code.

Note:

Without the comment, Code Assist will not be available for the function.

Configuring Code Assist

Code Assist options can be configured through the [Code Assist Preferences page](#), accessible from Window | Preferences | PHP | Editor | Code Assist.

Working with the Debugger

The purpose of this tutorial is to teach you how to debug files and applications both remotely and locally in order to gain maximum efficiency and accuracy from your files and projects.

Contents

[Purpose and Usage](#)

[Debugging PHP Files \(PHP Scripts\)](#)

[Debugging PHP Applications \(PHP Web Pages\)](#)

Purpose and Usage

Zend Studio for Eclipse's Debugging feature can detect and diagnose errors in PHP code situated locally or on remote servers. The debugger allows you to control the execution of your program by setting breakpoints, suspending launched programs, stepping through your code, and examining the contents of variables.

Zend Studio for Eclipse includes five types of debugging:

- Locally Debugging PHP Scripts - Debugging PHP files using Zend Studio for Eclipse's internal PHP Executable debugger.
- Remotely Debugging PHP Scripts - Debugging PHP files using your server's debugger.
- Debugging PHP Web Pages - Debugging files, applications and projects on your server, using the local or server copies of your files.
- Debugging URLs - Debug applications on your server by entering a URL.
- Toolbar Debugging - Debug files and applications directly from your browser.

Debugging PHP Files (PHP Scripts)

PHP files (PHP Scripts) on your workspace can be debugged using either Zend Studio for Eclipse's internal debugger or your server's debugger. Using your server's debugger, you can test the execution of the file on your server's environment. This is especially relevant if your server's loaded extensions are different to Zend Studio for Eclipse's internal server.


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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.



This procedure demonstrates how to debug a file using the internal debugger or your server's debugger:

1. Create a PHP file, called "debug", and copy-paste the example code into it.
(See the "Working with the Debugger" Tutorial in Zend Studio for Eclipse's Online Help for the example code.)
2. Set a breakpoint at line 103 by double-clicking the marker bar to the left of the editor window.
A blue ball will appear.
3. Save the file.
4. Click the arrow next to the debug button  on the toolbar and select Open Debug dialog -or- select Run | Open Debug dialog from the main menu -or- right-click the file in PHP Explorer view and select Open Debug dialog.
A Debug dialog will appear.
5. Double-click the PHP Script option to create a new debug configuration.

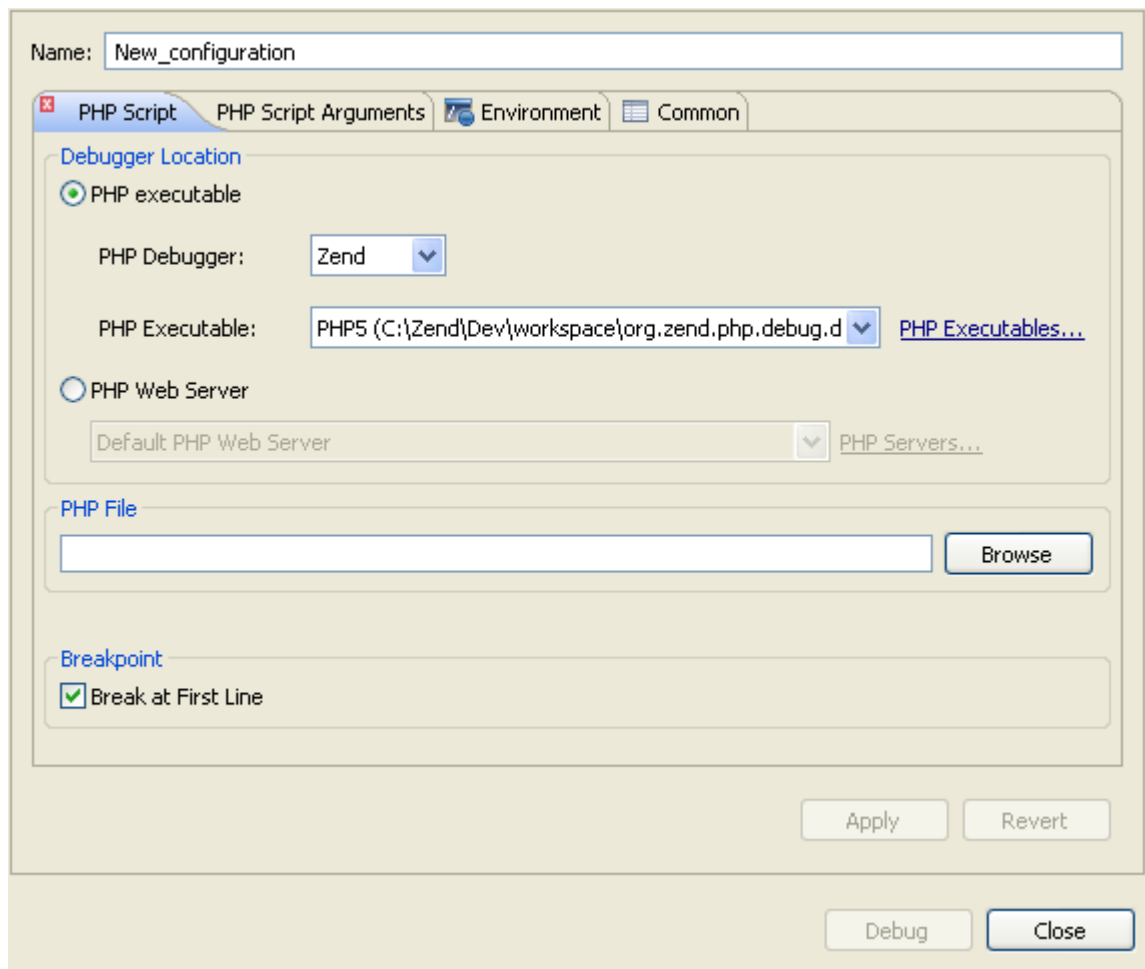


Figure 4 - New Debug Configuration

6. Enter a name for the new configuration.

7. To debug the file using Zend Studio for Eclipse's PHP Executable debugger, select the PHP Executable option in the Debugger Location category.
-Or- To debug the file using your server's Debugger, select the PHP Web Server option under the Debugger Location category and select your server from the drop-down list.
If you have not configured a server, click New and enter:
 - i. Your server's name.
 - ii. The URL of its document root.
8. Under PHP File, click Browse and select the "debug" file.
9. Ensure that the 'Break at First Line' Breakpoint checkbox is selected.
9. Click Apply and then Debug.
10. Click Yes if asked whether to open the PHP Debug Perspective.
11. A number of views will open with information about your script.
12. The Debug View is where the debug stack trace is displayed and the debugging process can be monitored and controlled.

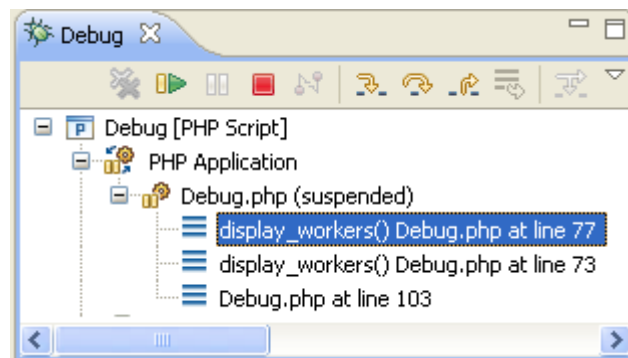




Figure 5 - Debug view

The debugging process will currently have stopped where your first `<?php` label appears.



13. Click the Resume icon  to continue to the breakpoint.
14. Click Step Into . The Debugger goes into the function defined in line 103 and advances to line 77.
15. The Variable view will now display various information about the relevant variables and parameters through which the function was reached.
16. In the editor window, place and hold the cursor over `$worker_name`, `$worker_address`, and `$worker_phone`. A tooltip appears displaying the variable values.

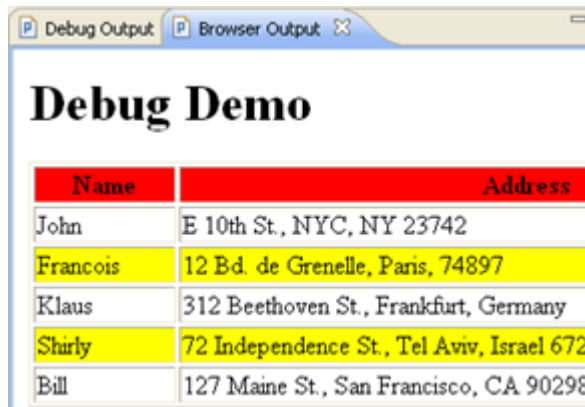

```

74
75 {
76
77 global $db;
78
79 for ($i=0, $n=count($db); $i<$n; $i++) {
80
81 $worker_data = $db[$i];
82 $worker_data = null;
83 $worker_name = $worker_data[0];
84
85 $worker_address = $worker_data[1];
86

```


Figure 6 - Editor view while debugging

17. Click Step Return.  The cursor returns to line 103.
The Debug Output view will display the HTML output created up until the breakpoint, while the Browser Output view will show the current output to a browser.
18. In the Debug view, click Resume  until the debugging process is terminated.
Notice that as the debugging process progresses, the Debug Output and Browser Output displays are updated.



Name	Address
John	E 10th St., NYC, NY 23742
Francois	12 Bd. de Grenelle, Paris, 74897
Klaus	312 Beethoven St., Frankfurt, Germany
Shirly	72 Independence St., Tel Aviv, Israel 672
Bill	127 Maine St., San Francisco, CA 90298

Browser Output

19. The console view will display any errors or warnings about your script. In this case, it will display a Notice about an undefined variable on line 105.
20. Click on the PHP Perspective icon to return to normal editing mode.
21. To run the debugging process again, click the arrow next to the debug icon  on the toolbar and select your configuration -or- select Open Debug Dialog and double-click your configuration from the Debug dialog.
Clicking the debug icon will debug the last executed launch configuration.

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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.

Debugging PHP Applications (PHP Web Pages)

Zend Studio for Eclipse also allows you to debug applications, projects or files that are already on the server. You can debug either the local (Workspace) copy of files or the server copy of files.

Note:

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

The Zend Debugger comes bundled with Zend Core and Zend Platform, but can also be downloaded as a separate component from <http://downloads.zend.com/pdt/server-debugger>.




This procedure demonstrates how to debug applications on a server:

1. Create a new PHP file, called "form1", with the following code:

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

2. Create a second PHP file, called "welcome", with the following code:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

3. Save both files.
4. Copy the files to your server.
5. Click the arrow next to the debug button  on the toolbar and select Open Debug dialog -or- right-click the file in PHP explorer or within the file's editor window and select Debug as | Open Debug dialog.
A Debug dialog will appear.
6. Double-click on the PHP Web Page option to create a new debug configuration.
7. Enter a name for the new configuration.
8. Select the Zend Debugger to from the Server Debugger drop-down list.

9. Select your server from the drop-down list.
If you have not configured a server, click New and enter:
 - i. Your server's name.
 - ii. The URL of its document root.
10. Under the File/Project category, click Browse and select the "form1" file. This will be the file from which the debugger will start debugging (the 'debug target'.)
11. Ensure that the URL pointing to the file location is correct.
If this is not correct, unmark the Auto Generate checkbox and manually change the URL.

Note:
You can choose whether the file content will be taken from the local copies of the files or from the files located on your server.
To select the file's Source Location, select the 'Advanced' tab and select the relevant option under the 'Source Location' category.

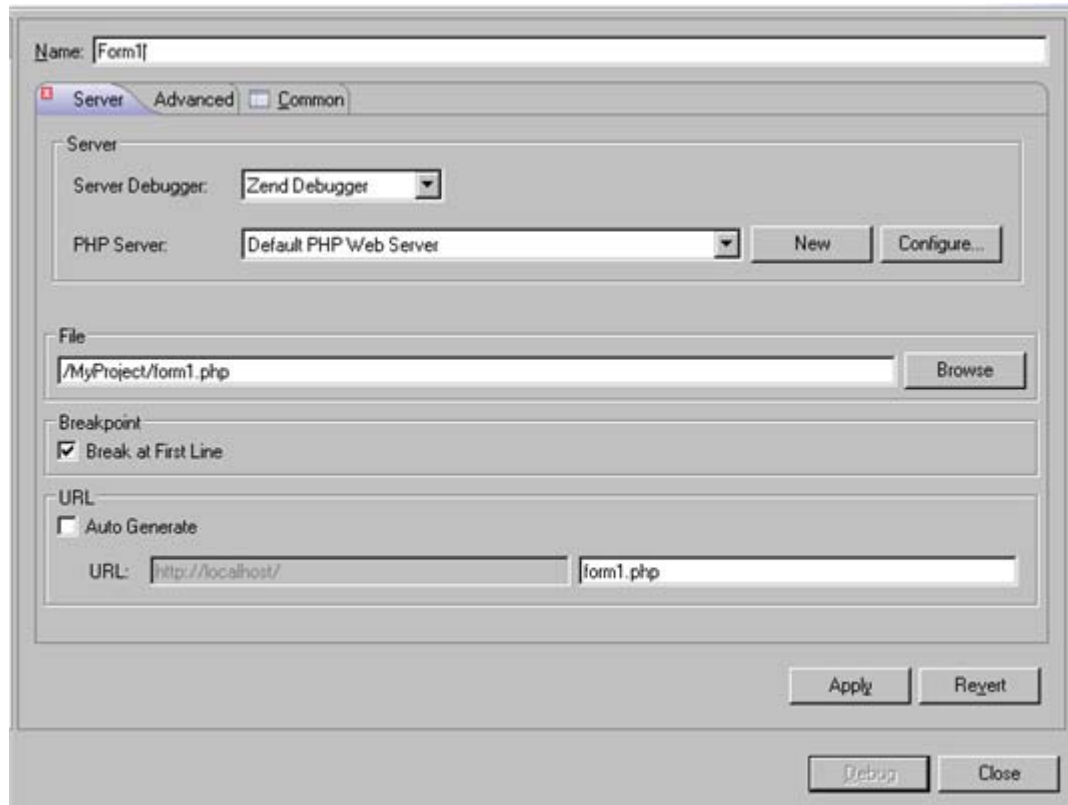


Figure 7 - New Debug Configuration

12. Click Apply and then Debug.
13. Click Yes when asked whether to open the PHP Debug Perspective.
14. The Debug Perspective will open with several views relevant to the debugging process (See ['PHP Debug Perspective'](#) for more information on the different views.)
15. In the editor view, you will see the code for the "form1" file.
16. In the Debug view, click Resume to resume the debugging process.

17. The browser output will display a form asking you to enter your Name and Age.
18. Select the browser view (tabbed with the editor window). This will display the output of your script in a browser in 'real time'.
Note that this is different from the Browser Output window.
19. In the browser view, enter your Name and Age and click Submit Query.

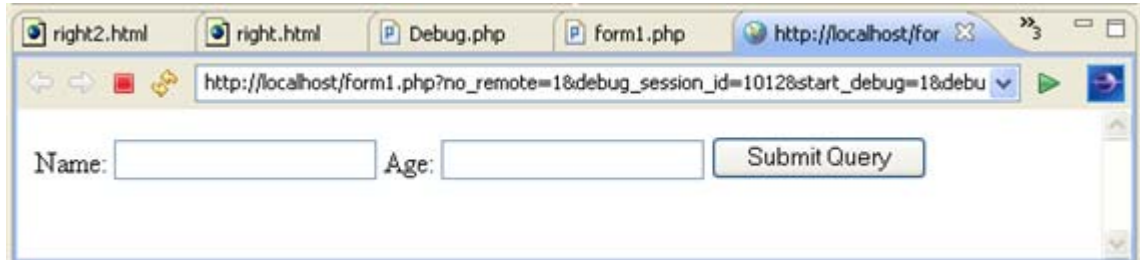
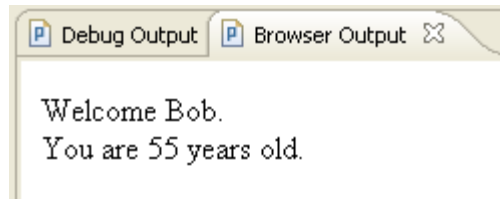



Figure 8 - Browser view

20. Another editor tab will open, with the script from the welcome.php file.
21. In the Debug view, click Resume to resume the debugging process.
22. The browser output and browser views will display the final result of your application:
"Welcome [Name].
You are [Age] years old."



Browser Output

23. The debugging process will have terminated.
24. Click on the PHP Perspective icon to return to normal editing mode.
25. To run the debugging process again, click the arrow next to the debug icon  on the toolbar and select your debugging configuration.

Working with the Refactoring Feature

The purpose of this tutorial is to teach you how to refactor your code to easily rename and move files and elements without damaging your projects or severing the link between referenced items, as well as creating the necessary links between PHP elements in separate files.

Contents

[Purpose and Usage](#)

[Renaming / Moving Files](#)

[Renaming Elements within a File](#)

[Organizing Imports](#)

Purpose and Usage

The refactoring feature 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.
- Organize Includes so that elements from one file can be referenced in another file.

Refactoring should be used when you are reorganizing your project and changing names and locations of files and elements.

Note:

Refactoring will only work from within PHP Explorer view and not from Navigator view.

Renaming / Moving Files

Renaming a file

Renaming a file will result in the automatic renaming of all instances where that file is referenced within the project.



This procedure demonstrates how to Rename a file:

1. Create a PHP file, called RenFile1, with the following code:

```
<?
$a = 5;
?>
```

2. In the same project, create a second PHP file, called RenFile2, with the following code:

```
<?
require( "RenFile1.php" );
$a = 8;
?>
```

3. Save both files.
4. In PHP Explorer view, right-click RenFile1 and select Refactor | Rename – or- select it and go to Refactor | Rename from the Main Menu.

5. A Rename File dialog will appear.
6. In the Rename File dialog box, rename RenFile1 to RenFile3.
7. Check the "Update references" box and click Preview.
8. In the Preview window, scroll through the changes and note that, as well as the name of the file itself being changed, the reference to the file in RenFile2 will also have been changed.
9. Press OK to apply changes.

The reference to RenFile1 in RenFile2 will have been updated to:

```
require( "RenFile3.php" );
```

to reflect the changes in the file name.

Moving a file

Moving a file will result in the automatic updating of all instances where that file is referenced within the project to reflect its change of location.



This procedure demonstrates how to Move a file:

1. Create RenFile1 and RenFile2 as described in steps 1 to 3 under ["Renaming a File"](#), above.
2. Within the same project, create an additional folder called RenFolder.
3. In PHP Explorer view, right-click RenFile1 and select Refactor | Move -or- select it and go to Refactor | Move from the Main Menu.
4. In the Move dialog, select RenFolder.
5. Click Preview.

The Preview windows will display the changes that the move will apply to your script. Note that RenFile1's new location will automatically be updated in the reference to it in RenFile2.

6. Click OK to execute the Move.

Renaming Elements within a File

All PHP Elements (e.g. classes, interfaces, functions, methods) can also be renamed and refactored from within the editor window.



This procedure demonstrates how to use the Rename Elements feature:

1. Create 2 files (RenFile1 and RenFile2) as described in steps 1 to 3 under ["Renaming a File"](#), above.
2. In the editor window of RenFile2, highlight the variable "a" on line 3.
3. Right-click and select Refactor | Rename -or- click Alt+Shift+R.
4. In the Rename Global Variable dialog box, rename the variable to b.
5. Check the "Update textual occurrences" box and click Preview.
6. In the Preview window, scroll through the changes and note that occurrences of variable "\$a" will be changed in RenFile1 as well as in RenFile2.
7. Press OK to apply changes.

Organizing Imports

Using the Organizing Imports feature will allow PHP objects created in one file to be called into other files within the project.



This procedure demonstrates using the Organizing Imports feature:

1. Create a new PHP file called OIFile1 with the following code:

```
<?php
class MyClass {}
?>
```

2. Create a second PHP file, called OIFile2, with the following code:

```
<?php
function myFunction() {
return 1;
}
?>
```

3. Create a third PHP file, called OIFile3, with the following code:

```
<?php
$z = new MyClass();
echo myFunction();
?>
```

Note that the code in OIFile3 attempts to call MyClass and myFunction created in OIFiles 1 and 2.

4. Save the files.
5. Right-click within OIFile3 and select Refactor | Organize Includes – or – go to Refactor Menu and select Organize Includes.
6. In the Organize Includes dialog, a display will show that instances of MyClass and myFunction have been found in the previous files.
7. Scroll down to preview the changes. Note that that include_once will now be added for OIFile1 and OIFile2 so that MyClass and myFunction can be called.
8. Click OK to apply the changes.

Note:

If two elements of the same name are recognized, a window will display so that the relevant one can be selected.

The Organize Includes feature can also:

- Delete references and "include" calls to files if the relevant items have been deleted from the code.
- Move "include" calls to the top of your script if they have been placed further down, in order for the elements to be referenced before the script is run.

Working with the Profiler

The purpose of this tutorial is to teach you how to profile files and applications in order to gain maximal efficiency in the execution of your script.

Contents

[Purpose and Usage](#)

[Profiling PHP Scripts](#)

[Profiling PHP Web Pages](#)

Purpose and Usage

Zend Profiler detects 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.

Zend Studio for Eclipse includes five types of profiling:

- Locally Profiling PHP Scripts - Profiling PHP files using Zend Studio for Eclipse's internal PHP Executable debugger.
- Remotely Profiling PHP Scripts - Profiling PHP files using your server's debugger.
- Profiling PHP Web Pages - Profiling files, applications and projects on your server, using the local or server copies of your files.
- Profiling URLs - Debug applications on your server by entering a URL.
- Toolbar Profiling - Debug files and applications directly from your browser.

Profiling PHP Scripts

Files can be profiled using either Zend Studio for Eclipse's internal debugger or your external (remote) server.

Use the remote profiling function if you want to test the execution of the file on your server's environment. This is especially relevant if your server's loaded extensions are different to Zend Studio for Eclipse's internal server.

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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.



The following procedure demonstrates how to profile a PHP Script, either locally or remotely:

1. Create a file, called Person, and copy-paste the example code into it.
(See the "Working with the Profiler" Tutorial in Zend Studio for Eclipse's Online Help for the example code.)

2. Create a second file, called tryPerson, and copy-paste the example code into it. See the "Working with the Profiler" Tutorial in Zend Studio for Eclipse's Online Help for the example code.
3. Save both files.
4. Click the arrow next to the Profile button  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.
5. A Profile dialog will appear.

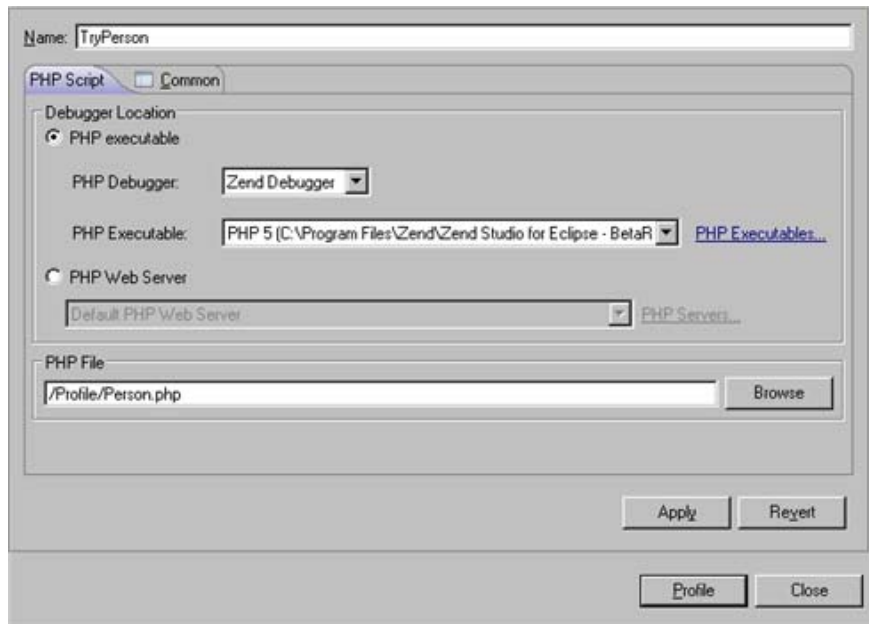


Figure 9 - Profile Configuration Dialog

6. Double-click the PHP Script option to create a new Profile configuration.
7. Enter a name for the new configuration.
8. To profile the file locally using Zend Studio for Eclipse's internal debugger, select the PHP Executable setting under the Debugger Location category and select the required PHP executable (PHP 4 or 5).
To profile the file remotely on your server using the Zend Debugger installed on your server, select the PHP Web Server option and select your server from the drop-down list. (If you have not yet configured a server, click the PHP Servers link and follow the instructions under [Adding PHP Servers.](#))
9. Under PHP File, click Browse and select the "TryPerson" file.
10. Click Apply and then Profile.
11. 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 the following information:

- **Profiler Information** - 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
- Query - The specific query parameters
- Path - The exact location of the first file called
- Total Request Time - Total process time of the entire page
- Number of Files - Number of files composing the page
- Date - Date and time that the profiling took place

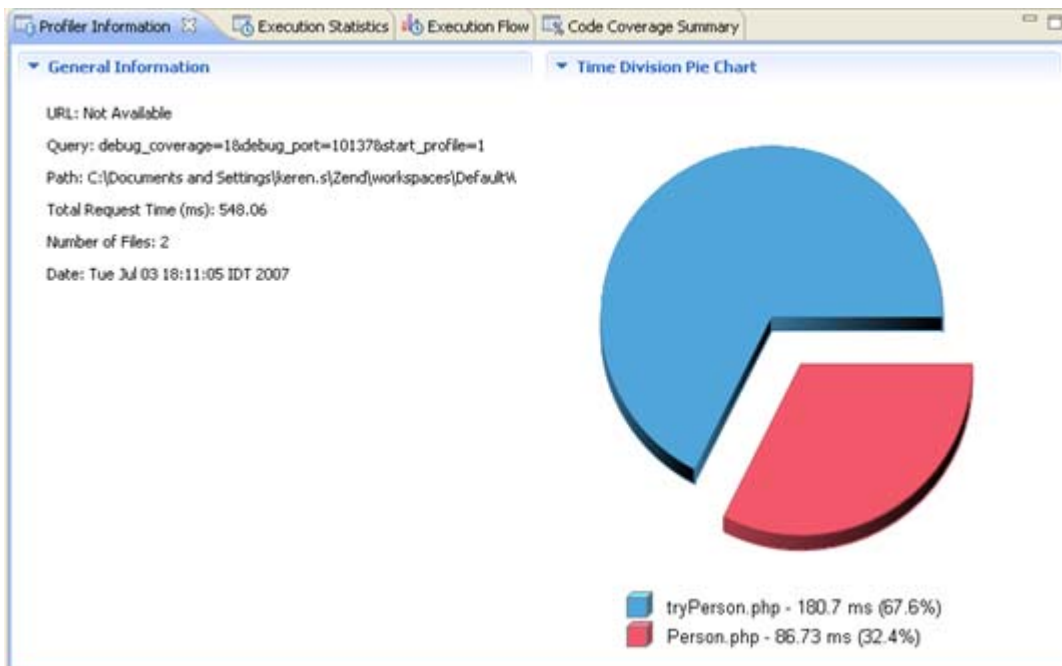


Figure 10 - Profiler Information

- **Execution Statistics** - Displays the list of files constructing the URL and detailed information on functions in the files. The window contains statistics relevant to each function:
 - 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 process duration including internal calls.

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
setGender	3	0.000004	0.000013	0.000000	0.000013

Figure 11 - Execution Statistics

Note:

Click the 'Show as percentage' button on the toolbar to see the statistics as percentages rather than times.

- **Execution Flow** - Shows the flow of the execution process and summarises 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.

Function	File	Total Execution Time	Duration Time (ms)
main	tryPerson.php	48.8%	267.43
main	Person.php	0.0%	0.01
Person::__construct	Person.php	15.73%	86.24
Person::setFirstName	Person.php	0.01%	0.03
Person::setLastName	Person.php	0.0%	0.01
Person::setAge	Person.php	0.0%	0.01
Person::setGender	Person.php	0.0%	0.02
Person::__construct	Person.php	0.01%	0.06
Person::__construct	Person.php	0.01%	0.06
Person::printData	Person.php	0.05%	0.28
Person::printData	Person.php	0.01%	0.04
Person::printData	Person.php	0.01%	0.05

Profiler Execution Flow

- **Code Coverage Summary** - Summary of how many lines of code were covered.
- Element - The file / folder that was called.
- Covered Lines (Visited / Significant / Total) - Percentage of lines covered within each file. (Visited = Number of lines covered / Significant = number of lines excluding comments and declarations / Total = Total number of lines in the file.)

Element	Covered Lines (Visited/Significant/Total)
Profile Project (2)	68% (39/57/72)
Person.php	63% (31/49/62)
tryPerson.php	100% (8/8/10)

Figure 12 - Code Coverage Summary

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

```

private $lastName ;
private $age ;
private $gender ;
private static $personId = 1 ;
public static $personCount = 0 ;
// constructor
function __construct ( $newFirstName , $newLastName , $newAge , $newGender ) {
    $this->id = Person::$personId ;
    $this->setFirstName ( $newFirstName ) ;
    $this->setLastName ( $newLastName ) ;
    $this->setAge ( $newAge ) ;
    $this->setGender ( $newGender ) ;
    Person::$personId ++ ;
    Person::$personCount ++ ;
}
    
```

Figure 13 - Covered Lines Editor

Profiling PHP Web Pages

Zend Studio for Eclipse also allows you to profile whole applications, projects or collections of files that are already on the server.


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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.



The following steps demonstrate how to profile a file on an external server:

1. Create a file, called Person, and copy-paste the example code into it.
See the "Working with the Profiler" Tutorial in Zend Studio for Eclipse's Online Help for the example code.
2. Create a second file, called tryPerson, and copy-paste the example code into it.
See the "Working with the Profiler" Tutorial in Zend Studio for Eclipse's Online Help for the example code.
3. Save both files.
4. Copy them to your server.
5. Click the arrow next to the Profile button  on the toolbar and select Open Profile dialog –or– right-click the file in PHP explorer or within the file's editor window and select Profile As | Open Debug dialog.
A Profile dialog will appear.
6. Double-click on the PHP Web Page option to create a new profile configuration.
7. Enter a name for the new configuration.
8. Select the Zend Debugger to from the Server Debugger drop-down list.
9. Ensure that your server is selected from the list.
If you have not configured a server, click New and enter:
 - i. Your server's name.
 - ii. The URL of its document root.
10. Under the File/Project category, click Browse and select the "tryPerson" file. This will be the file from which the profiling process will start.
11. Click Apply and then Profile.
A confirmation dialog will be displayed asking whether you want to open the Profiling Perspective.
12. 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 profiling information.

See [PHP Perspectives and Views](#) for details on the information displayed in the profiling perspective.

Working with PHPUnit Testing

The purpose of this tutorial is to teach you how to create and run PHPUnit tests on your code. You will learn how to create and run single unit test cases and test suites containing a number of test cases.

Contents:

[Purpose and Usage](#)

[Creating a PHPUnit Test Case](#)

[Running your PHPUnit Test Case](#)

[Analyzing Errors](#)

[Creating and Running a PHPUnit Test Suite](#)

[Generating a PHPUnit Test Report](#)

Purpose and Usage

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.

Creating a PHPUnit Test Case

Zend Studio for Eclipse will automatically create test case files which can be run in order to check the functionality of your code.



The following steps demonstrate how to create a PHPUnit Test Case:

1. Create a new PHP file, called "Calculator", and copy-paste the following code into it:

```
<?php
class Calculator {
    public function add($a, $b) {
        return $a + $b;
    }

    public function multiply($a, $b) {
        return $a * $b;
    }

    public function divide($a, $b) {
        if($b == null) {
            throw new Exception("Division by zero");
        }
        return $a / $b;
    }

    public function subtract($a, $b) {
        return $a - $b;
    }
}
?>
```

2. Save the file.
3. In PHP Explorer view, right-click the calculator file and select New | PHPUnit Test Case. The PHPUnit Test Case dialog will open.

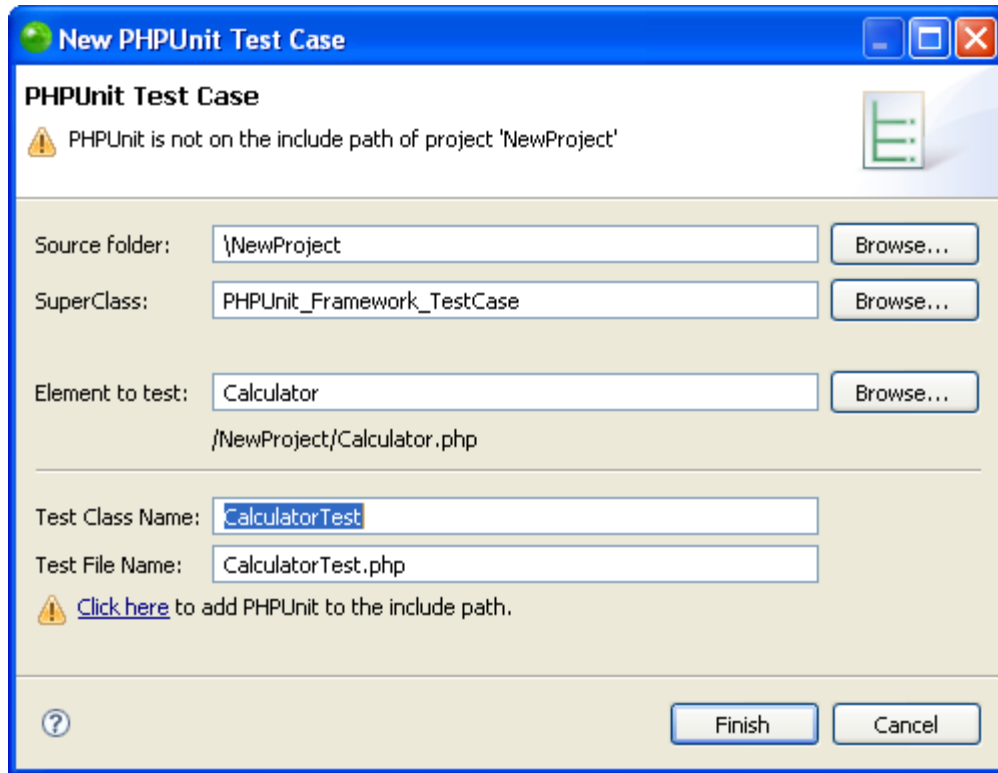


Figure 14 - New PHPUnit Test Case dialog

4. The relevant information will have already been entered. Note that a new file will be created called CalculatorTest.php
A warning will also appear stating that the PHPUnit is not the include class of your project.
5. To add it to the include path, click the underlined "Click here" link at the bottom of the dialog screen.
Once it has been clicked, the link and the warning message will disappear.
6. Click Finish to create your test case.

A CalculatorTest file will have been added to your project in PHP Explorer. This will contain tests for your original "calculator" file.

Note that all functions (add, multiply, divide and subtract) in the original "Calculator" file will have a corresponding test function in the "CalculatorTest" file:


```

53+    /**
56+    public function test_add()
64
65+    /**
68+    public function test_divide()
76
77+    /**
80+    public function test_multiply()
88
89+    /**
92+    public function test_subtract()
100

```

Figure 15 - Calculator Test Functions

Note:

Test functions will have been created, but parameters need to be entered in order for the test to run effectively. For more on this, see "[Running your Unit Test Case](#)", below.

Running your PHPUnit Test Case

Having created your PHPUnit Test Case, you will now need to customize it by entering relevant parameters to be checked before being able to run your test.

**To configure and run your test case:**

1. In the CalculatorTest file, expand public function test_add node.
2. Note that a function has been created but no parameters have been inserted. You will have to manually enter the relevant parameters to be tested and the predicted results.
3. Delete the following code:

```

// TODO Auto-generated CalculatorTest->test_add()
$this->markTestIncomplete("add test not implemented");
$this->Calculator->add(/* parameters */);

```



4. This is the default test which will return a "test not implemented" result if the test case is run.
5. Replace the above code with the following:

```




$this->assertEquals($this->Calculator->add(1, 2), 3);

```

6. The numbers 1 and 2 indicate that when the test case is run, the parameters 1 and 2 will be entered into the 'add' function in your Calculator file (i.e. the test will try to add 1 + 2). The last number (3) indicates that the expected result is 3. If the result is something other than 3, the test will report a failure for this function.
7. Save the file.


8. To run the unit test, click the arrow next to the Run button  on the toolbar and select Run As | PHPUnit Test  &endash; or- go to Run Menu and select Run As | PHPUnit Test



Or- to debug the PHPUnit Test Case, click the arrow next to the debug button  on the toolbar and select Debug As | PHPUnit Test  –or- from the Main Menu, go to Run and select Debug As | PHPUnit Test .

The unit test will be run and a PHP Unit view will open.

As the test is run, the parameters you have configured will be entered into the relevant functions in the Calculator file to test whether the correct result is outputted according to the expected results you specified.


- Four tests will be displayed - one for each calculator function - which should have passed successfully, as indicated by the green tick icon .

Note that the other three functions (divide, multiply and subtract), will have passed but will have a note indicating that they have not been implemented. This is because you have not yet specified the testing parameters.

- Repeat steps 1-6 above for the remaining functions, entering suitable parameters in the format:

```
$this->assertEquals($this->Calculator->subtract/divide/multiply(x, y), z);
```

Select each required operation (subtract, divide or multiply), and enter variables where x and y are the two parameters which will be entered into the calculator, and z is the expected result.

- Run the Unit Test again by clicking the Run Last Test button  in the PHPUnit view and check that all the test have passed successfully.

Notes:

- Tests can be written in alternate ways. So for example

```
$this->assertEquals($this->Calculator->add(1, 2), 3);
```

can also be written as:

```
$this->assertSame(3,$this->Calculator->add(1,2));
```

- You can create more than one test for each function, so that your function could appear as the following:

```
Tests Calculator->add()
*/
public function test_add()
{
$this->assertEquals($this->Calculator->add(1, 2), 3);
// $this->assertEquals($this->Calculator->add(1, 3), 4);
// $this->assertEquals($this->Calculator->add(-1, 2), 1);
}
```



Analyzing Errors

Once a PHPUnit test has been run, the results can be viewed and analyzed in order to diagnose and correct problematic sections of code.






The following steps demonstrate how to analyze and correct errors in your code:

1. To simulate a failed result, change the parameters under the add function so that the expected result is wrong. For example:


```
$this->assertEquals($this->Calculator->add(1, 2),4);
```
2. Save the file.
3. Run the Unit Test again by clicking the Run Last Test button  in the PHPUnit view.
4. The display in the PHPUnit view will now display that test_add has failed, indicated by the blue X icon .

The error message "Failed asserting that <integer:3> is identical to <integer:4>" indicates that the test failed as the actual result of the test was 3, but the expected result was 4.

5. To only view the failures, click the "Show failures only"  button on the view's toolbar.
6. Select a failed result to view it in the Failure Trace view. Click the Filter Stack Trace icon  to filter the results and view the relevant functions.
7. Double-click the failed result to go to the relevant section in the code.
8. Correct the code, save the file and run the test again by clicking the Run Last Test button  in the PHPUnit view.

The tests should be successful. If they are not, repeat steps 6-7.




Creating and Running a PHPUnit Test Suite

A number of different PHPUnit Test Cases can be unified into one UnitTest Suite file which will run all unit tests at once. This function is useful if you have a few tests within a project which you would like to run at once.



The following steps demonstrate how to create a PHPUnit Test Suite:

1. Create another Unit Test Case for your "Calculator" file by following steps 3 - 7 under ["Creating Unit Test Cases"](#), above.
2. Edit the test file to create different tests using your own tests, or copy-paste the example code into the file.
(See the "Working with PHPUnit Testing" Tutorial in Zend Studio for Eclipse's Online Help for the example code.)
3. Save the file.
4. From PHP Explorer View, select and right-click the Calculator project.
5. Select New | PHPUnit Test Suite.
A New PHPUnit Test Suite dialog will open.
6. Ensure that both your test cases are selected from the list.
7. Click Finish.
8. A new CalculatorSuite file will be created, integrating both tests into one file.

9. Run the CalculatorSuite by clicking the arrow next to the Run button  on the toolbar and select Run As | PHPUnit Test  –or- go to Run Menu and select Run As | PHPUnit Test .
10. Both tests will be run, with the results of both displayed in a tree in the PHPUnit view at the bottom of the screen.

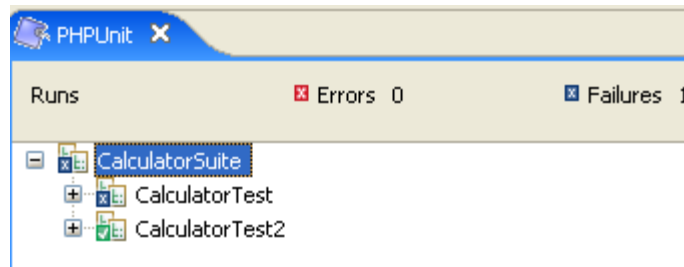



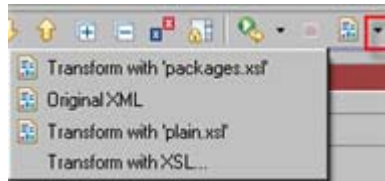
Figure 16 - Test Suite Results

Generating PHPUnit Test Reports



The following steps demonstrate how to run a report on your Unit Test results:

1. Once you have run the PHPUnit Test Suite, click the arrow next to the Report Generator icon  on the PHPUnit view's toolbar and select Transform with 'plain.xml' from the drop-down list.
See [PHPUnit Testing](#) for more on the different types of reports.



A report will be automatically generated and opened in a browser window.

Working with CVS

The purpose of this tutorial is to teach you how to work with the CVS source control system. You will learn how to configure your CVS repository, upload projects and files to it, check out (import) projects and files from it and commit changes which you have made to files.

Contents:

Purpose	Adding Files to Existing Projects
Adding a CVS Repository	Making Changes, Comparing Changes and Committing Changes
Sharing Projects	Replacing Files with Older Versions
Checking Out Projects from CVS	Deleting Files from CVS

Purpose

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 be able to revert file and project states back to previous versions.

Adding a CVS Repository

Before you can add projects to or export projects from CVS, you must define your CVS repository settings.

Note:

To access a repository, make sure that a CVS server is already configured.

This procedure describes how to create a CVS repository connection which you can access in order to be able to carry out CVS functionality.



To create a new CVS repository connection:


1. Open the CVS perspective by going to Window menu and selecting Open Perspective | Other | CVS Repository Exploring.
2. Click the Add CVS Repository button  on the view's toolbar -or- right-click within the CVS view and select New | Repository Location.
The Add CVS Repository dialog will open.



Figure 17 - New CVS Repository dialog

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.
- v. 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 repository will be displayed in the CVS Repositories view.



Sharing Projects

Through CVS, projects can be shared and worked on by numerous team members.



This procedure demonstrates how to upload a project you have created so that other users can work on it:

1. Create a new PHP project called "MyCVS Project".
2. Within the project, create a PHP file called "CVSFile1" with the following code:


```
<?php
//This is a new file
?>
```
3. In PHP Explorer View, right-click your project and select Team | Share Project. A Share Project dialog will open.
4. From the repository list, select CVS and click Next.
5. Select 'Use existing repository location', and select your CVS repository from the list.
6. Click Next.
7. In the Module Name dialog, select 'Use project name as module name'.
8. Click Next.
9. 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.)
10. The 'Share Project Resources' dialog will open. Your project will be displayed as a resource that is to be added to CVS. The purple plus icon  indicates that these are new files that have not previously been added to CVS.
11. Click Finish.
12. A Commit dialog will open. Enter the comment "I am uploading files to CVS." and click OK.
13. In PHP Explorer View, your project will now have a repository icon , indicating that it is in CVS.
14. Once you have committed your files, other team members will be able to access and change the files.

The instructions below explain how users can check out (import) projects from CVS, edit them and upload their changes.

Checking Out Projects from CVS


Once projects are placed on the CVS repository, they can be checked out (imported) by anyone with access to that repository.



This procedure demonstrates how to import (check out) projects from CVS into your workspace:

1. Delete the 'MyCVS Project' from your workspace in order to simulate being a new user who has not previously had access to this file.
Note: Deleting the project from your workspace will not delete it from CVS.
2. Go to File | Import | Projects from CVS.
3. Click Next.
4. Select your repository.
5. Click Next.
6. Select the 'Use an existing module' option to see all the available projects under that directory.
7. Select the 'MyCVS Project'.
8. Click Finish.

The project will now be added to your workspace and will be displayed in the PHP Explorer view.

Note the repository icon  next to the project in PHP Explorer view, indicating that they it is sitting in a CVS repository.

Adding Files to Existing Projects

You can add files to existing projects in the CVS repository and commit them so that other users can access them.




This procedure demonstrates how to add and commit a file into an existing project:

1. In your MyCVSProject, create a new PHP file, called "CVSFile2", with the following code:

```
<?php
//Another new file
?>
```

2. Save the file.
3. In PHP Explorer view, select the file, right-click and select Team | Commit.
4. A Commit dialog will open.
Enter a comment "Another new file added." and click OK.
5. The file will be committed to CVS and will be accessible by other users.

Note the  icon next to the file, indicating that it is sitting in a CVS repository.

Making Changes, Comparing Changes, and Committing Changes

Once files are stored on CVS, you and all other team members can make changes to the files and commit them. Before committing changes you have made to a file, you can compare the file stored locally in your workspace to the file stored on the CVS repository.

Making and Comparing Changes



This procedure demonstrates how to make changes to files and compare the newly edited local files to files in the repository:

1. Open CVSFile1 in your MyCVS Project.
2. After the text "This is a new file.", add "I have made a change".
3. Save the file.
In PHP Explorer view, the file and project will have a ">" suffix, indicating that a change has been made which has not yet been committed.
4. So far, the changes have only been saved in your workspace. In order to compare the local file to which you have made changes with the one sitting in the CVS repository, right-click the file in the PHP Explorer view and select Team | Synchronize with Repository.
5. A Text Compare dialog will open showing the local file you have just made changes to (in the left-hand window) as compared to the file in the repository (in the right-hand window).
The change you have made will be highlighted.

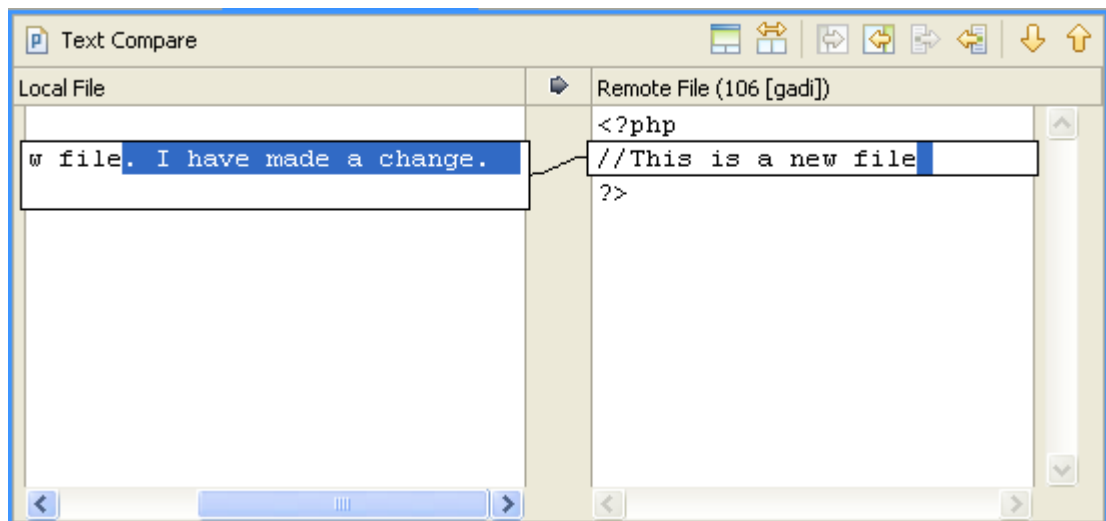


Figure 18 - Text Compare dialog

Committing Changes

Once you have edited your file and compared it to the one in the repository to ensure that the changes are correct, you can commit your changes.



To commit your changes to the repository:

1. In PHP Explorer View, right-click your file and select Team | Commit.
2. A Commit dialog will open.
Enter the comment "I have made changes to CVSFile1." and click Finish.
3. Your changes will now be committed to CVS and all users will be able to access the newly updated file.

Note:


If you had made changes to a number of files, you can use the Synchronize view, within the Team Synchronizing Perspective, to view and commit all your changes at once.

Replacing Files with Older Versions

Using CVS's version control system, users can revert back to older versions of files if incorrect or irrelevant changes have since been made.



This procedure demonstrates how to replace your file with an older version from CVS:

1. In PHP Explorer view, right-click your CVSFile1 and select Replace with | History.
2. A History view will be displayed, listing the times that the file has been committed. The changes made on CVS (as opposed to local workspace changes) will be shown with a repository icon.  As committed changes have been made twice (once during the original upload and once when you edited the file), 2 CVS revisions should be listed with their relevant comments.

Revision	Tags	Revision Time	Author	Comment
Today				
1.2		20/05/2007 15:02	keren.s	I have made changes to CVSFile1.
		20/05/2007 14:48		
		20/05/2007 14:25		
1.1		20/05/2007 14:25	keren.s	I am uploading files to CVS.

Figure 19 - CVS History view

3. To open the Text Compare view to see the previous version of the file, double-click the second CVS revision in the list (containing the comment "I am uploading files to CVS.") This is the original state the file was in when it was first uploaded.
4. To return the file to its previous state, select the second revision again, right-click and select Get Contents.
5. Your file will be replaced with the contents of the previous version of the file (without the line "I have made a change.")


Deleting Files from CVS

You can delete a file from the CVS repository by first deleting it from your workspace and then committing the deletion.



This procedure demonstrates how to delete a file from CVS:

1. In PHP Explorer view, right-click CVSFile2 and select Delete.
2. Click Yes when asked to confirm the deletion.
3. In PHP Explorer view, right-click MyCVS Project and select Team | Synchronize.
4. The Team Synchronizing Perspective will open.

The file you have deleted will be displayed with a purple arrow with a minus sign. 

5. Right-click the file and select Commit.
6. A Commit Deletion dialog will open.
Enter a comment "Deleting CVSFile 2."
7. Click OK.
8. The file will be deleted from CVS.

Note:

This action will delete the file from the CVS repository and not just from your workspace. This file will no longer be accessible by any users.

Working with SVN

The purpose of this tutorial is to teach you how to work with the SVN source control system. You will learn how to configure your SVN repository, upload projects and files to it, check out (import) projects and files from it and commit changes which you have made to files.

Contents:

Purpose	Adding Files to Existing Projects
Adding an SVN Repository	Making Changes, Comparing Changes and Committing Changes
Sharing Projects	Replacing Files with Older Versions
Checking Out Projects from SVN	Deleting Files from SVN

Purpose

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.

Adding an SVN Repository


Before you can add projects to or export projects from SVN, you must define your SVN repository settings.

Note:

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  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.
 - Authentication - The user name and password you use to connect to SVN.
4. Click Finish.
5. Your SVN repository will now be added to the SVN Repository view.

Sharing Projects


Through SVN, projects can be shared and worked on by numerous team members.



The following steps demonstrate how to upload a project to your SVN repository

location:

1. Create a new PHP project called "MySVN Project".
2. Within the project, create a PHP file called "SVNFile1" with the following code:


```
<?php
//This is a new file
?>
```
3. In PHP Explorer View, right-click your project and select Team | Share Project. A Share Project dialog will open.
4. From the repository list, select SVN and click Next.
5. Select 'Use existing repository location', and select your repository from the list.
6. Click Finish.
7. 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.)
8. A Commit dialog will open. Enter the comment "I am uploading files to SVN." and click OK.
9. In PHP Explorer View, your project will now have a repository icon , indicating that it is in SVN.
10. Once you have committed your files, other team members will be able to access and change the files.

The instructions below explain how users can check out (import) projects from SVN, edit them and upload their changes.

Checking Out Projects from SVN

Once projects are placed on the SVN repository, they can be checked out (imported) by anyone with access to that repository.



The following steps demonstrate how to check out (import) projects from SVN into your workspace:

1. Delete the 'MySVN' project from your workspace in order to simulate being a new user who has not previously had access to this file.

Note: Deleting the project from your workspace will not delete it from SVN.
2. Go to File Menu and select Import | Projects from SVN.
3. Click Next.
4. Select your repository.
5. Click Next.
6. A 'Select Resource' dialog will appear. Expand the nodes until you see your project (by default, this will have been placed under 'Trunk').

7. Select your project and click Finish.
8. A 'Check Out As' dialog will appear.
Leave the 'Check out as a project with the name specified' option marked and 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.

Now that you have imported a project from SVN into your workspace, you can add files, edit existing files and commit your changes to the SVN repository. (See below).

Adding Files to Existing Projects

You can add files to existing projects in the SVN repository and commit them so that other users can access them.



The following steps demonstrate how to add and commit a file into an existing project:

1. In your SVNProject, create a new PHP file, called "SVNFile2" with the following code:

```
<?php
//Another new file
?>
```

2. Save the file.
3. In PHP Explorer View, select the file, right-click and select Team | Commit.
4. A Commit dialog will open.
Enter a comment "Another new file added." and click OK.

The file will be committed to SVN and will be accessible by other users.

Making Changes, Comparing Changes, and Committing Changes

Once files are stored on SVN, you and all other team members can make changes to the files and commit them. Before committing changes you have made to a file, you can compare the file stored locally in your workspace to the file stored on the SVN repository.

Making and Comparing Changes



The following procedure demonstrates how to make changes to files and comparing local files to files in the repository:

1. Open SVNFile1 in your SVNProject.
2. After the text "This is a new file.", add "I have made a change".
3. Save the file.
In PHP Explorer view, the file and project will have a ">" suffix, indicating that a change has been made which has not yet been committed.
4. So far, the changes have only been saved in your workspace. In order to compare the local file to which you have made changes with the one sitting in the SVN repository, right-click the file in the PHP Explorer view and select Team | Synchronize with Repository.
5. Click Yes when asked whether to open the Team Synchronizing perspective. (Check the 'Remember my decision' box to prevent this prompt from appearing in the future.)

A Text Compare dialog will open showing the local file you have just made changes to (in the left-hand window) as compared to the file in the repository (in the right-hand window).
The change you have made will be highlighted.

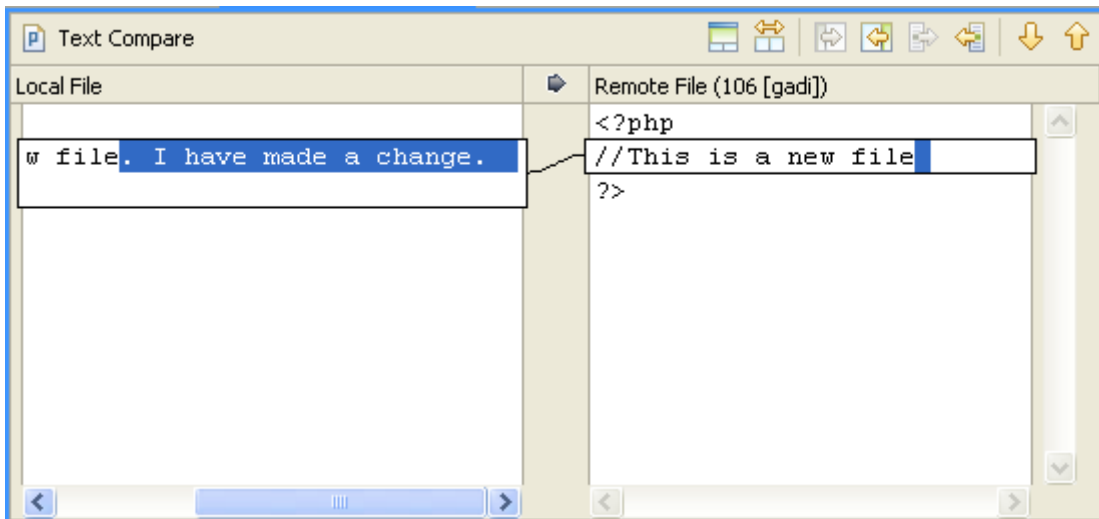



Figure 20 - SVN Text Compare View


In addition, the Synchronize view will have opened in the left hand-side, displaying the file to which you have made changes. The file will have a grey arrow icon  indicating that changes have been made which need to be synchronized with the repository.

Committing Changes

Once you have edited your file and compared it to the one in the repository to ensure that the changes are correct, you can commit your changes.



To commit your changes to the repository:

1. From the Synchronize view, click the 'Commit All Outgoing Changes'  button.
-Or- In PHP Explorer View, right-click your file and select Team | Commit.
2. A Commit dialog will open.
Enter the comment "I have made changes to SVNFile." and click Finish.

Your changes will now be committed to SVN and all users will be able to access the file.

Replacing Files with Older Versions

Using SVN's version control system, you can revert back to older versions of files if incorrect changes have since been made.



This procedure demonstrates how to replace your file with an older version:

1. In the PHP Explorer view, select your SVNFile1 and right-click.
2. Select Replace with | Revision.
3. A Replace with Revision dialog will open.
Select Revision and click 'Select'.
4. A Select Revision dialog will open, listing the various times the file has been committed (revisions).
As committed changes have been made twice (once during the original upload and once when you edited the file), 2 revisions should be listed with their relevant comments.

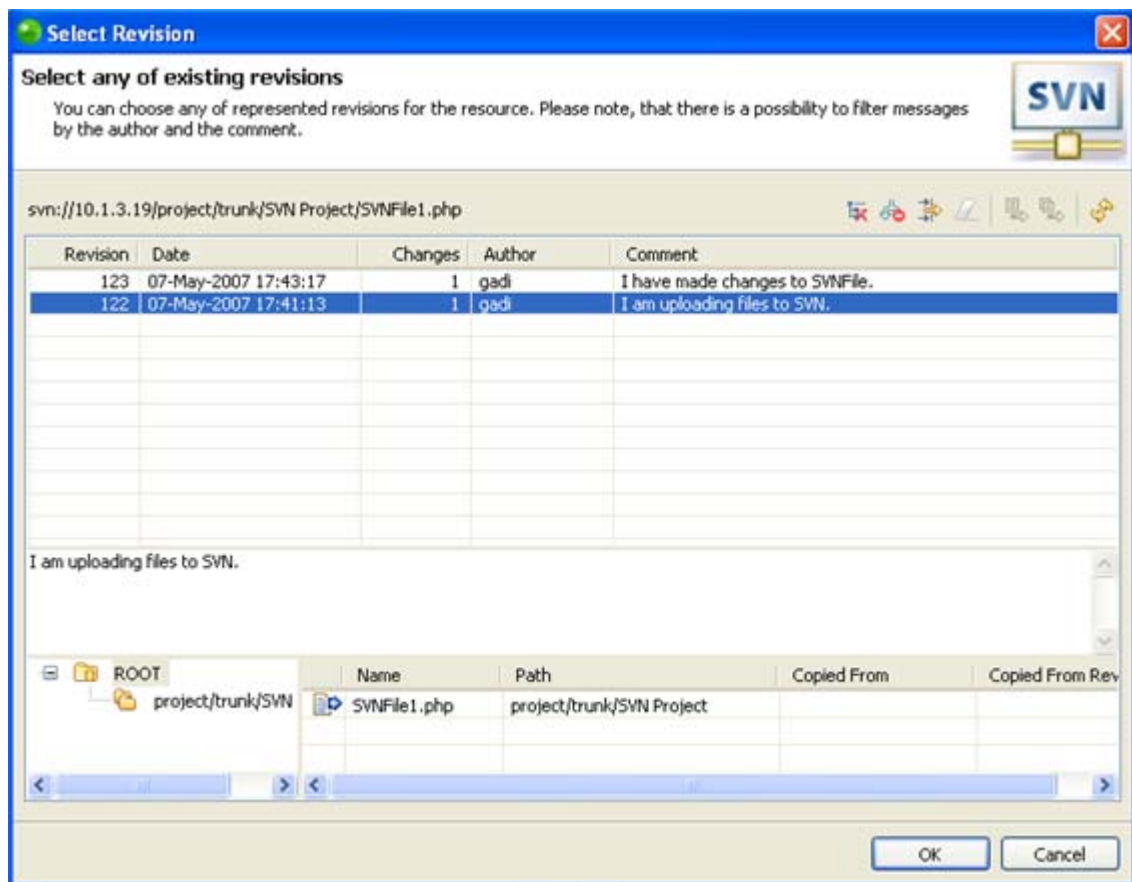


Figure 21 - Select Revision Dialog

5. Scroll between the two revisions.
To return the file to its previous state, select the second revision in the list (containing the comment "I am uploading files to SVN.") This is the original state the file was in when it was first uploaded.
6. Click OK.

7. You will be returned to the Replace with Revision dialog, which will now have a revision number in the Revision box.
8. Click OK.
9. You will be prompted to confirm that you would like to replace the revision.
Click yes.

Your project will be reverted to the old one and the line "I have made a change" will be removed from SVNFile1.

Deleting Files from SVN

You can delete a file from the SVN repository so that the file will no longer be available to any users.



This procedure demonstrates how to delete a file from SVN:

1. Open the SVN Repositories view.
2. Expand the nodes to find your project.
3. Right-click the file you would like to delete and select Delete.
4. A Commit Deletion dialog will open.
Enter a comment if required.
5. Click OK.

The file / project will be deleted from your SVN repository.

Note:

This action will delete the file from the SVN repository and not just from your workspace. This file will no longer be accessible by any users.

See the Subversive User Guide for more information.

Concepts

[PHP Support](#)

[Code Assist](#)

[Automatic Completion](#)

[Matching Brackets](#)

[Code Folding](#)

[Syntax Coloring](#)

[Bookmarks](#)

[Commenting Code](#)

[PHPDocs](#)

[Hover Support](#)

[PHP Manual Integration](#)

[Refactoring](#)

[Code Galleries](#)

[Zend Framework Integration](#)

[PHP - Java Bridge Support](#)

[JavaScript Support](#)

[PHP/HTML WYSIWYG](#)

[Data Tools Platform](#)

[Real Time Error Detection](#)

[Zend Platform Integration](#)

[PHPUnit Testing](#)

[Profiling](#)

[Debugging](#)

[Zend Debugger Toolbar](#)

[Path Mapping](#)

[Include Paths](#)

[Tunneling](#)

[CVS](#)

[SVN](#)

[Local History](#)

[Zend Guard Integration](#)

[RSS Feeds](#)

[WSDL - Web Services Description Language](#)

[Update Manager](#)

PHP Support

Zend Studio for Eclipse supports PHP versions 4 and 5.

PHP version settings affect:

- The elements displayed in the PHP Functions view.
- The options available in Code 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.
- 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.

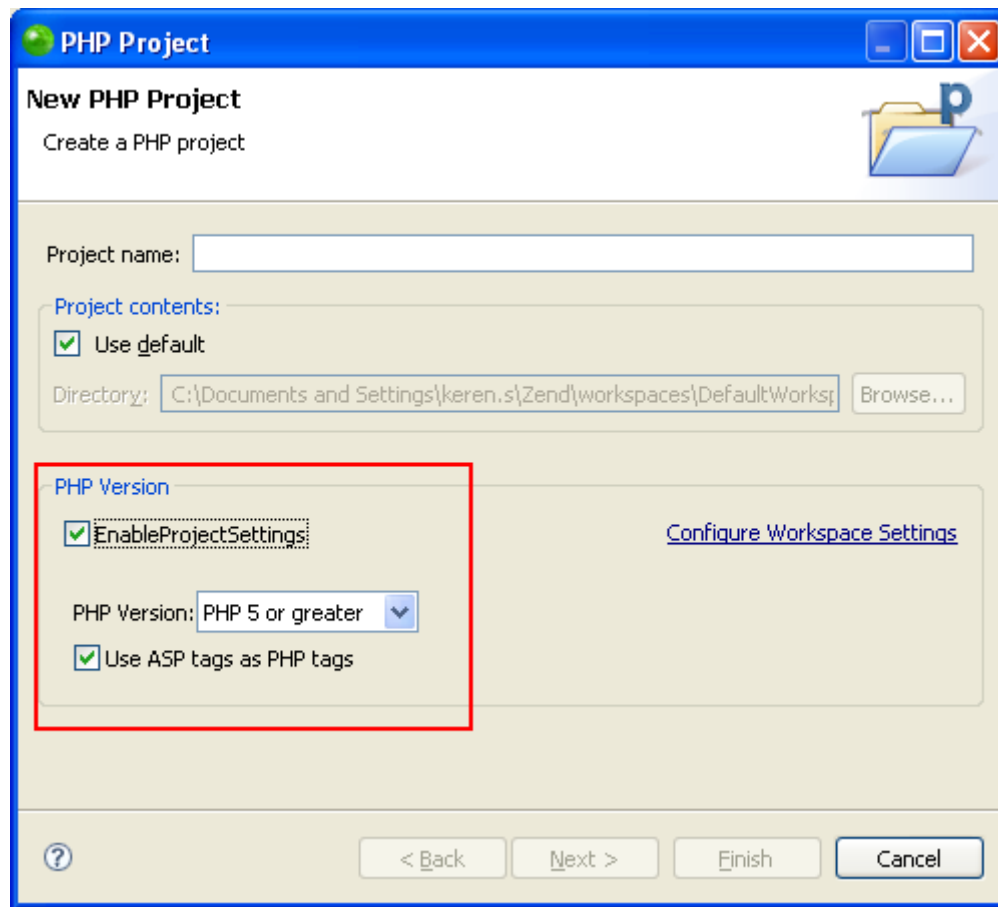


Figure 22 - New Project Settings

- The default PHP executable used with the debugger can be set in the [Debugging Preferences](#) 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.

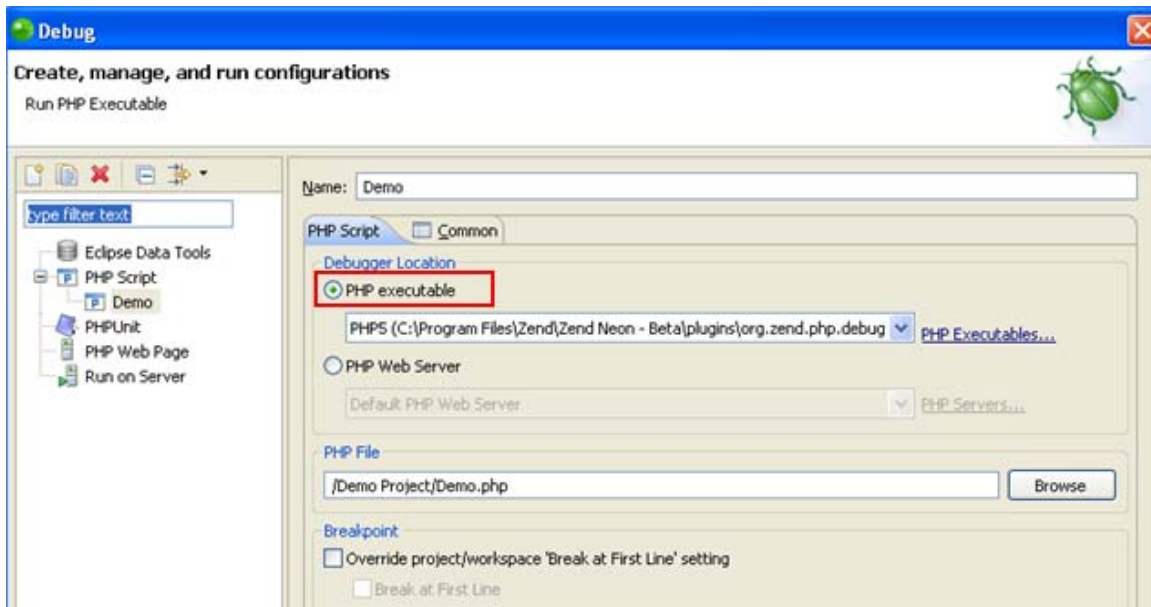


Figure 23 - New Debug Configuration

Note:









In order to minimize errors, the PHP Executable used for debugging/profiling should match the PHP version used for the project.

Code Assist

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

A list of possible code elements appears 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:

-  Reserved PHP Words
-  Functions
-  Templates
-  Classes
-  Interfaces
-  Constants
-  Variables (public)
-  PHP File Include Call

Code 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.

Using elements within the same scope

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



Examples:

- Variables within a function will be added to the Code Assist list when the cursor is within that function.
- Elements defined within one file will be available in the Code 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.

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

Figure 24 - Function Parameter Hint

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

Code Assist for Include Statements

Code 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 Code Assist window to display the files available for the function. Selecting a file will complete the include/require call.



Example:

```
include ("")
}
?>
```

<ul style="list-style-type: none"> <input type="checkbox"/> Debug.php <input type="checkbox"/> PHPDocument1.php <input type="checkbox"/> test/Debug.php 	Location: /Testing/Debug.php
--	--

```
16 include ("Debug.php")
17 }
18 }
```

Configuring Code Assist

To configure code assist options, go to the [Code Assist Preferences](#) page, accessed from Window Menu | Preferences | PHP | Editor | Code Assist.

Automatic Completion

Zend Studio for Eclipse 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.

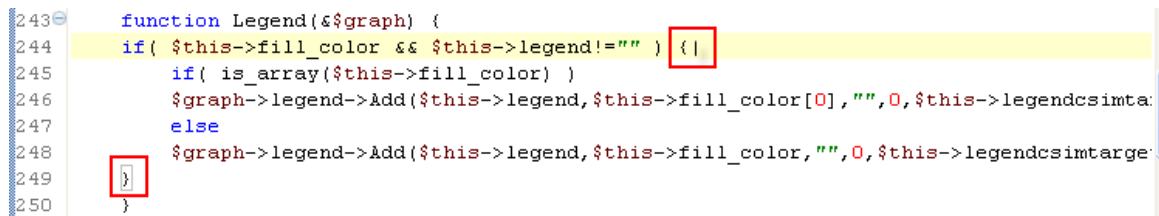
The types of patterns that can be auto-completed can be configured from the [Typing Preferences](#) page, accessible by going to Window | Preferences | PHP | Editor | Typing.

Matching Brackets

Zend Studio for Eclipse 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 }

```

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](#).

```

1  <?php
2  class Calculator {
3+   public function add($a, $b) {
6
7-   public function multiply($a, $b) {
8       return $a * $b;
9   }
10
11+  public function divide($a, $b) {
17
18+  public function subtract($a, $b) {
21 }
22 ?>

```

Figure 25 - Code Folding Example

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.

Syntax Coloring

Zend Studio for Eclipse 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.

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

The default color and font settings for PHP elements are:

- Boundry Maker
- Comment
- HereDoc
- Keyword
- Normal
- Number
- PHPDoc
- String
- Task Tag
- Variable

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

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.

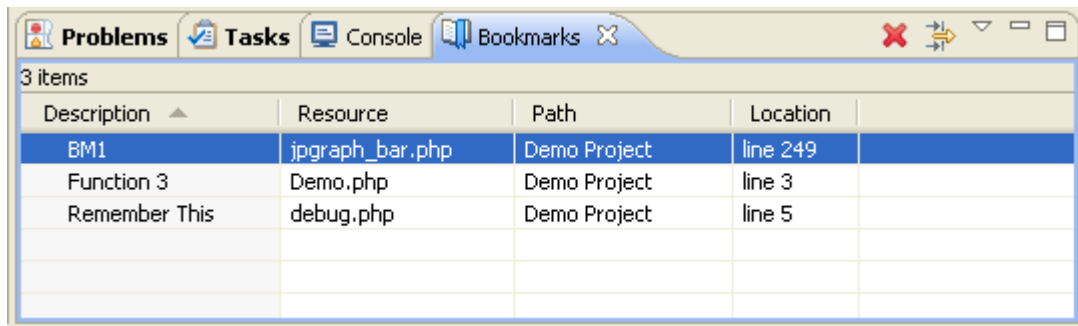


Figure 26 - Bookmarks View

For more on using bookmarks, see "Bookmarks, tasks and other markers" in the Workbench User Guide.

Commenting Code

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.

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
*/
```

phpDoc Block Comments

Zend Studio for Eclipse 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.

Descriptions that are added above a code element are also automatically added to the Code Assist bank so that the next time the code element is used it is readily available from the Code Assist list.

In addition, the element's descriptions will appear in the Outline view.

phpDoc blocks also serve as the input for creating a [PHPDoc](#).

See the [Commenting PHP DocBlocks](#) topic for more information.

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 for Eclipse's interface with phpDocumentor.



Figure 27 - PHPDoc Example

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:

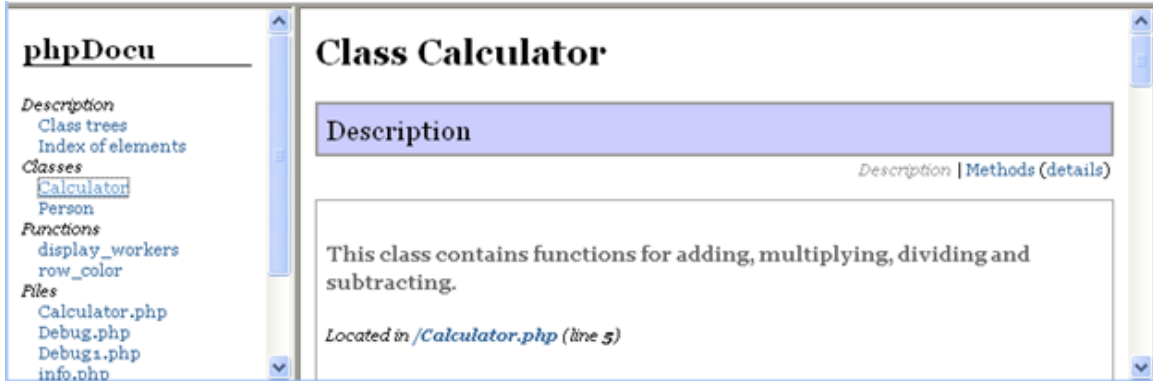
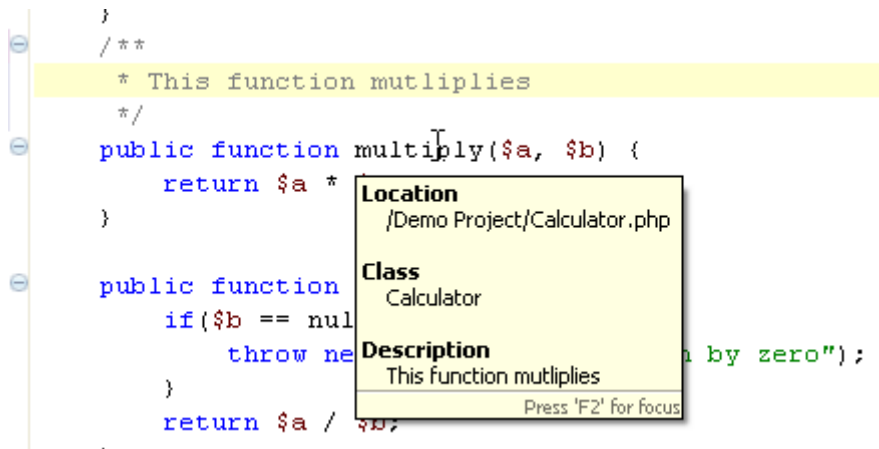


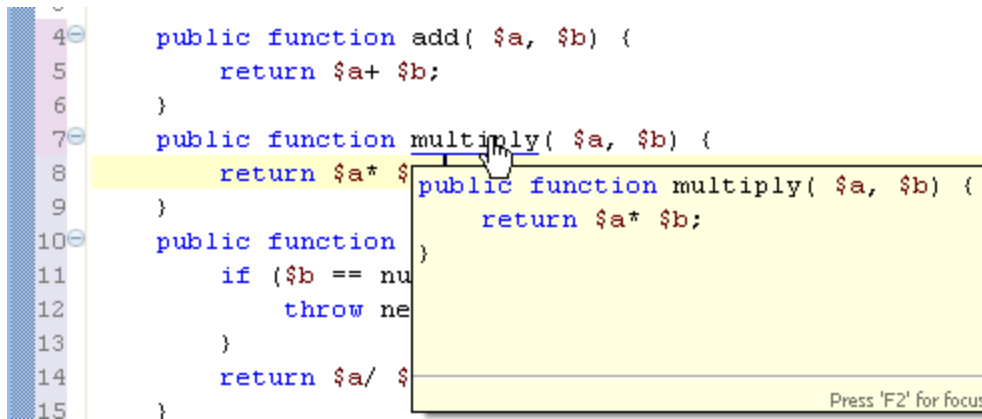
Figure 28 - PHPDoc Example - Classes

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](#):



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.

To configure your hover settings, go to the [Hover Preferences page](#), accessible by going to Window | Preferences | PHP | Editor | Hovers.

PHP Manual Integration

Zend Studio for Eclipse can integrate with PHP Manual sites in order to get the most up-to-date PHP information.

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.



Figure 29 - PHP Manual

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.

Refactoring

The Refactoring feature in Zend Studio for Eclipse 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.
- Organize Includes so that elements from one file can be referenced in another file.

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."

<http://www.refactoring.com>

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.

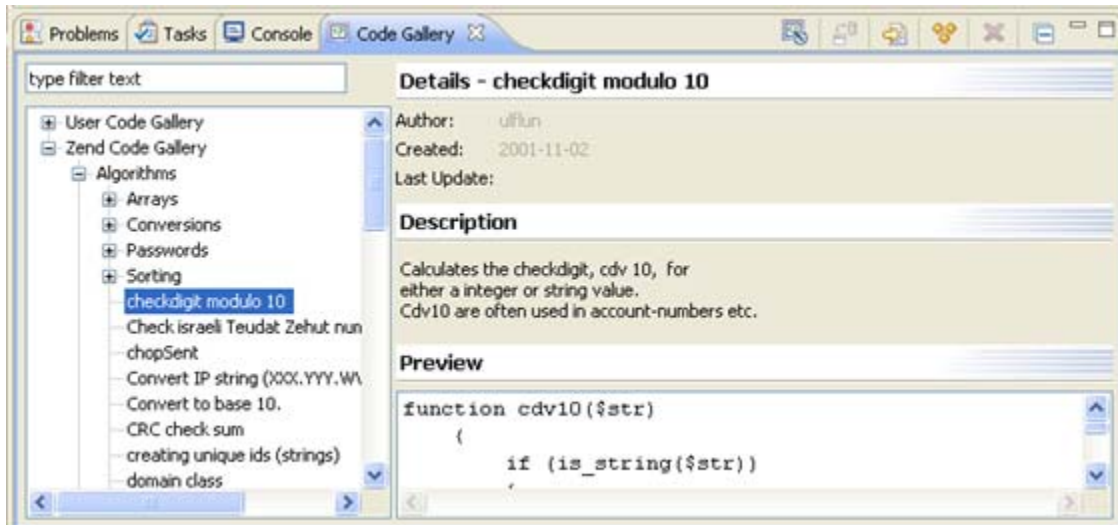


Figure 30 - Code Gallery

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.

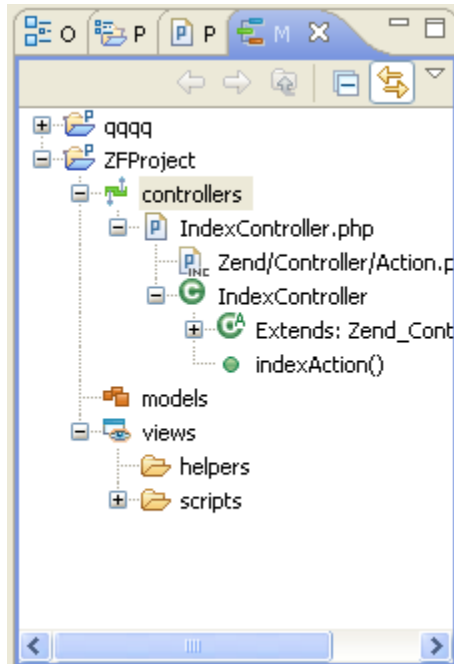
Zend Framework Integration

Creating a new Zend Framework Project will create a project that is divided into Framework's Controller, Model, View system.

In addition, 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 for Eclipse will contain basic files for a simple "Hello, World!" program.

When a Zend Framework Project is created, its outline will be displayed in the MVC Outline view.



MVC View

The MVC Outline view provides an outline of all controller, model and view classes, files, variables and related functions.

Note:

To open the MVC Outline view, go to Window | Show View | MVC Outline view.

Once Zend Framework's libraries are included in a project's include path, its classes, functions, iterators and variables will be available for use within the Code Assist window. Simply press Z followed by Ctrl+Space in the Editor to view the list of available Zend Framework elements:

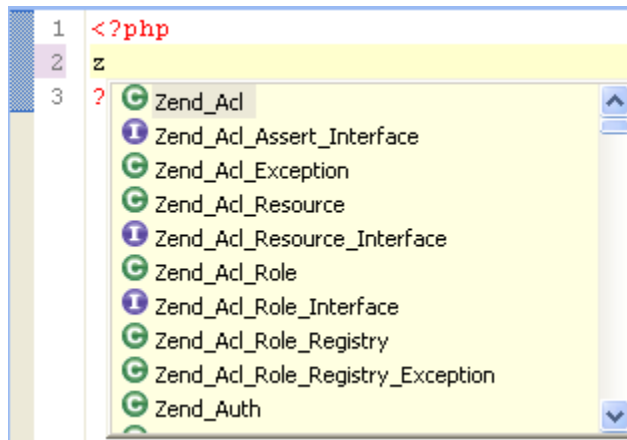


Figure 31 - Zend Framework Code Assist Options

Zend Framework Controller, Model and View classes can also be created using Zend Studio's New MVC wizards, accessible through the [File | New Submenu](#).

For more 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.

For more external resources, see "[Useful Links](#)".

PHP - Java Bridge Support

Zend Studio for Eclipse supports the Zend Platform Java Bridge which is the leading performance and reliability solution for businesses that seek to utilize both PHP and Java/J2EE.

Based on a unique design that allows for a single Java Virtual Machine (JVM) instantiation and direct calls from PHP, the Java Bridge delivers unprecedented performance and scalability that make true PHP and Java integration a reality.

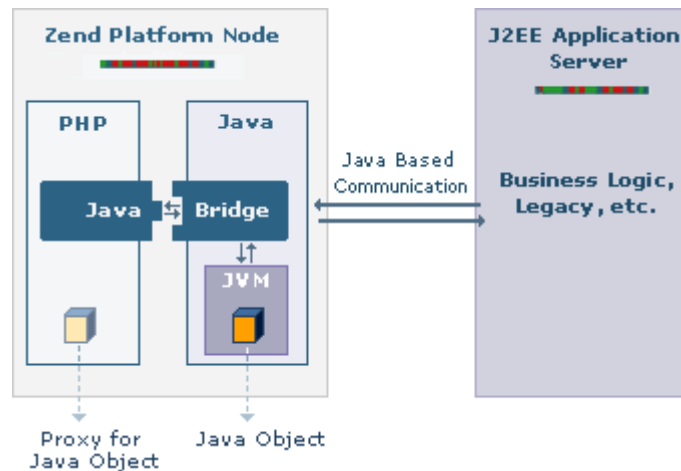


Figure 32 - Java Bridge Architecture

Java Bridge code can be run and debugged in the same way as normal PHP code.

Java Bridge Functionality:

Java Bridge provides four main functions:

1. Function invocation.

Java Bridge provides code completion for the following Java functions/methods:

- `java_get_statistics()` - Get Java server statistics
- `java_get_version_info()` - Get Java extension version information
- `java_last_exception_clear()` - Clear last Java exception
- `java_last_exception_get()` - Get last Java exception
- `java_set_encoding()` - Set encoding for PHP-Java data transfers
- `java_set_ignore_case()` - Set if search for Java classes and methods should be case-insensitive
- `java_throw_exceptions()` - Set if exceptions should be thrown on errors

2. Java instantiation / assignment - Creates new Java-PHP Objects.

Zend Studio for Eclipse supports Java Object instances. After instantiation of a PHP Java object, the PHP variable it was assigned to assumes the properties and methods of the Java class.

This allows the object to implement Java functionality, as well as allowing access to Java's Code Assist options.

Assignments attach the resulting Java object to the PHP variable. See "[Adding Java Objects](#)" for more.



Example:

```
/**
 * Java Bridge instantiation
 */
$firstService = new java("com.zend.FirstService");
$secondService = new java("com.zend.SecondService", date());
```

3. Java Exception handling - Catching Java exceptions.

Java exceptions can be caught by Zend Studio for Eclipse using Java Bridge.



Example:

```
/**
 * Java exception handling
 */
try {
    $a = new java("java.io.File");
    $a->doSomething();
    // java invocations
} catch (JavaException $e) {
    // catching exceptions that were thrown during the invocation
    echo "problem with the java execution ...";
    echo $e->getCause();
}
echo "everything was OK..."
```


4. Automatic Type Conversion

The Java Bridge will also perform the following conversions to PHP objects when Java objects are returned:

Java	PHP
lang.java.String	String
java.lang.Integer / int, Long, Byte, Char, Short	Int
java.lang.Double / double, Float	Float
java.lang.Boolean / Boolean	Boolean
Object[]	Array
Hashtable	Array

All other objects remain as Java objects.

JavaScript Support

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

The HTML `<script>` tag is used to insert a JavaScript into an HTML page.

```
<HTML>
<script type="text/javascript">
document.write("Hello World!");
</script>
</HTML>
```

Zend Studio for Eclipse provides Code Assist and Syntax Coloring for JavaScript code.

PHP/HTML WYSIWYG

Zend Studio for Eclipse comes with an advanced HTML WYSIWYG editor allowing for easy creation of HTML pages with advanced functionality.

These features are accessible through the PHP/HTML WYSIWYG perspective, (accessible from Window | Open Perspective | Other | PHP/HTML WYSIWYG).

To access this WYSIWYG functionality, files also need to be created with or opened in the PHP/HTML WYSIWYG editor.

The PHP/HTML WYSIWYG editor incorporates the following tabs:

- Design - Shows a visual outline of the text and HTML objects.
- Source - Shows the source code.
- Design/Source - Shows a split screen with both Design and Source displays
- Preview - Previews the file in a browser

In addition, the PHP/HTML WYSIWYG perspective incorporates the following additional views:

- PHP/HTML Toolbox - Enables drag-and-drop functionality for easy insertion of HTML tags and objects into your files.
- Properties - Enables you to view and configure the properties of various HTML objects.
- Image Preview - Allows for the quick selection of images for insertion into the file.
- CSS Preview - Displays a preview of code using your CSS style sheet.
- CSS Palette Editor - Allows the configuration of colours in your CSS style sheet.

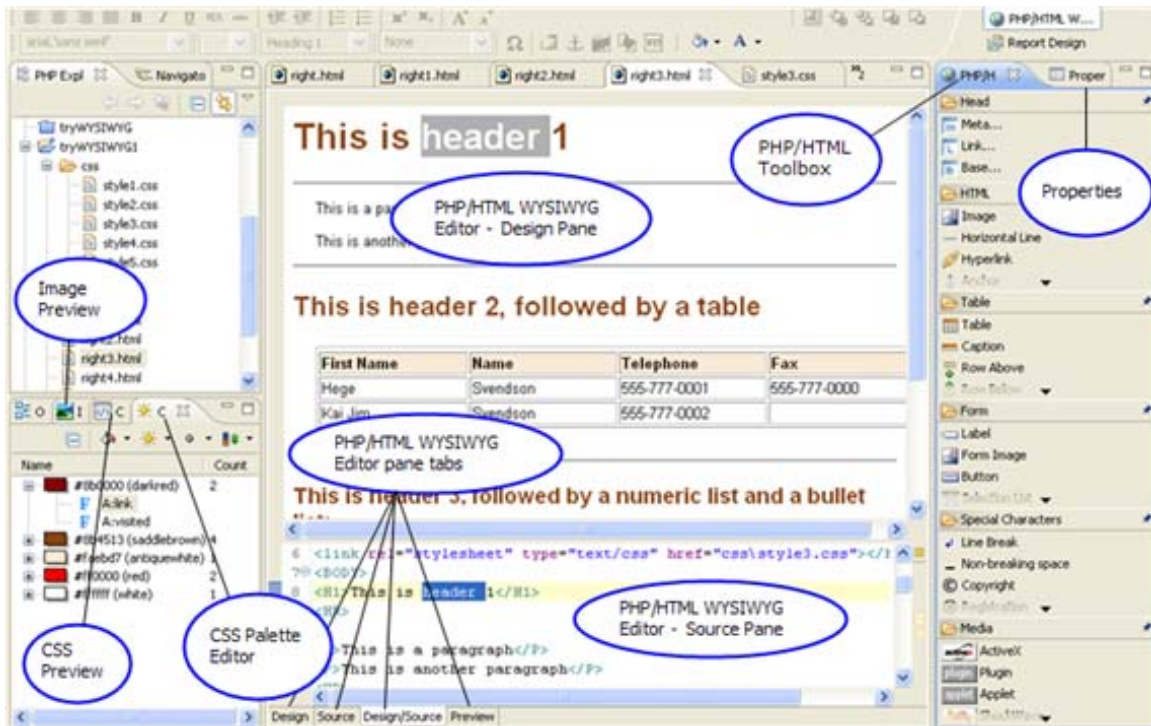


Figure 33 - PHP/HTML WYSIWYG Perspective

The PHP/HTML WYSIWYG Editor supports JavaScript and CSS. JavaScript actions can be set for any HTML object through the Advanced Tabbed Property View or Property View.

Opening a file with the PHP/HTML WYSIWYG Editor will enable various toolbar icons and Menu Bar menus:

- Modify Menu - Allows editing functionality
- Table Menu - Allows the insertion and editing of tables
- Insert Menu - Allows the insertion of various items and HTML objects.

More information about writing HTML can be found at: <http://www.w3schools.com/html>

Data Tools Platform

Zend Studio for Eclipse'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 Guide.

For instructions on connecting to and accessing your database, see [Using the Data Tools Platform](#).

Real Time Error Detection

Zend Studio for Eclipse 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.

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.

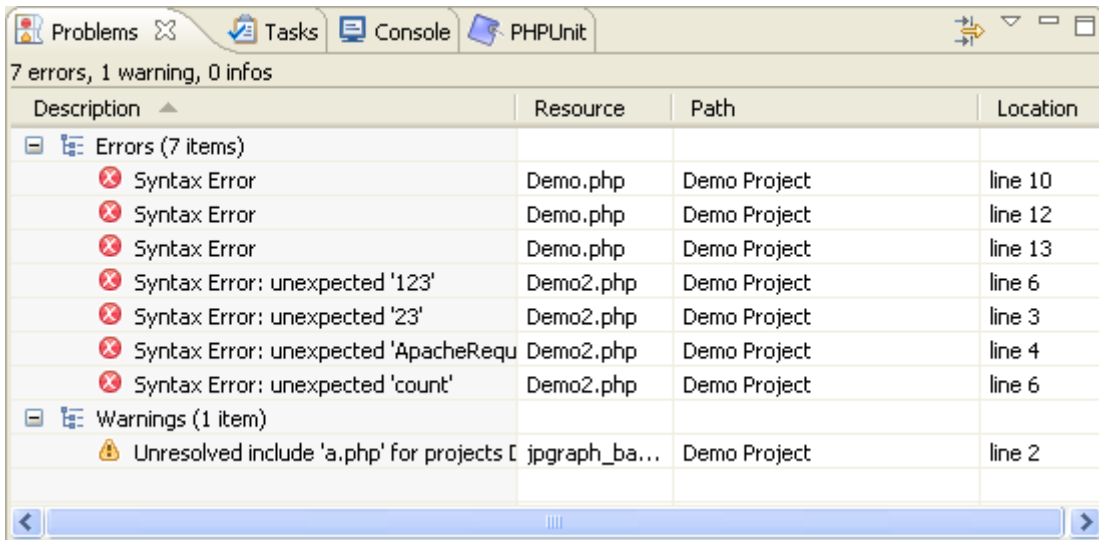


Figure 34 - Problems view

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.

Code Analyzer

Zend Studio for Eclipse's Code Analyzer can detect problems beyond the regular parsing warnings and errors. It helps developers to analyze static source code to enforce good coding practices and scan PHP code. The Code Analyzer 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). Code Analyzer 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.

Code Analyzer problems will also be displayed in the Problems view (see [Real Time Error Detection](#) for more information).

To enable / disable Code Analyzer and to configure which occurrences will trigger warning or error messages, go to the [Code Analyzer Preferences page](#), accessible by going to Window | Preferences | PHP | Code Analyzer.

Zend Platform Integration

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

Zend Platform 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 Platform as 'events.'

For more on events and how to configure what constitutes an event, see the PHP Intelligence chapter in the Platform User Guide.

Zend Platform's events can be viewed directly from Zend Studio for Eclipse using both the Platform Event view and Zend Studio for Eclipse's internal browser, which can display Platform's GUI.

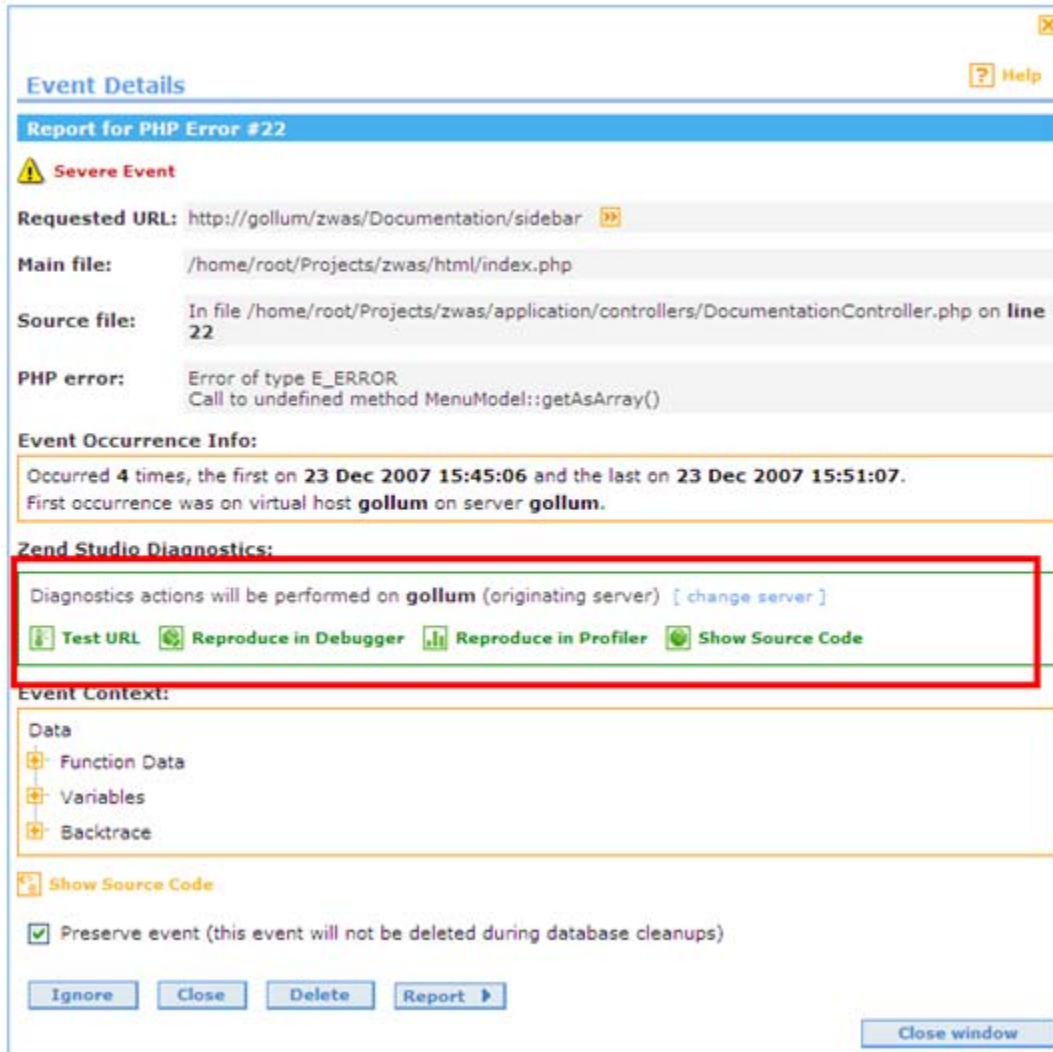


Figure 35 - Zend Platform Event Details - Zend Studio for Eclipse Internal Browser

Conversely, Platform's integration with Zend Studio for Eclipse means that problematic scripts found in Platform can be viewed, tested, debugged and profiled using Zend Studio for Eclipse's functionality.

For more on Zend Platform, go to <http://www.zend.com/en/products/platform>.

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 for Eclipse can generate a range of reports to easily view and analyze your tests.

The types of reports available are:

- Original XML - Generates an XML output of your test results. This can be used to create your own reports.
- Transform with '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.
- Transform with '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 'Trasnform 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.

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.

Zend Studio for Eclipse includes five different profiling methods:

PHP Script Local Profiling

Allows you to profile files on your workspace using Zend Studio for Eclipse'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 for Eclipse.

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

PHP Script Remote Profiling

Allows you to profile files on your workspace using your server's Zend Debugger.

This allows you to profile local files using the Zend Debugger situated on your server. This allows you to test your files in the production environment, and allows you to utilize the extensions installed on your server.

See [Remotely Profiling a PHP Script](#) 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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.

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 in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Core and Zend Platform, 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.

Debugging

Zend Studio for Eclipse's 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.

Zend Studio for Eclipse includes five different debugging methods:

PHP Script Local Debugging

Allows you to debug files on your workspace using Zend Studio for Eclipse'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 for Eclipse Internal Debugger uses its own PHP executable that was installed together with Zend Studio for Eclipse.

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

PHP Script Remote Debugging

Allows you to debug files on your workspace using your server's Zend Debugger.

This allows you to debug local files using the Zend Debugger situated on your server. This allows you to test your files with the production environment, and allows you to utilize the extensions installed on your server.

See [Remotely Debugging a PHP Script](#) 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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.

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 in order for remote debugging and profiling capabilities to function.

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

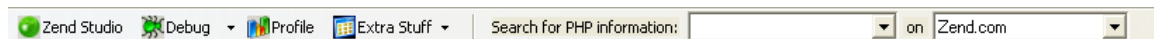
Toolbar Debugging

Debug files and applications directly from your browser.

See [Debugging Using the Zend Debugger Toolbar](#) for more information.

Zend Debugger Toolbar

The Zend Debugger Toolbar provides an easier way to debug or profile your sites and applications by initiating debug and profile sessions directly from your browser.



The Zend Debugger Toolbar contains the following options:

- **Zend Studio** - Opens Zend Studio for Eclipse. Ensure the Zend Studio for Eclipse .exe file is configured in the Zend Toolbar Settings.
- **Debug** - Launches a Debugging session in Zend Studio for Eclipse.
- **Profile** - Launches a Profiling session in Zend Studio for Eclipse.
- **Extra Stuff** - Provides access to Zend Debugger 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 Debugger Toolbar](#) for information on how to get started with the Zend Debugger Toolbar.

Path Mapping

Zend Studio for Eclipse 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:

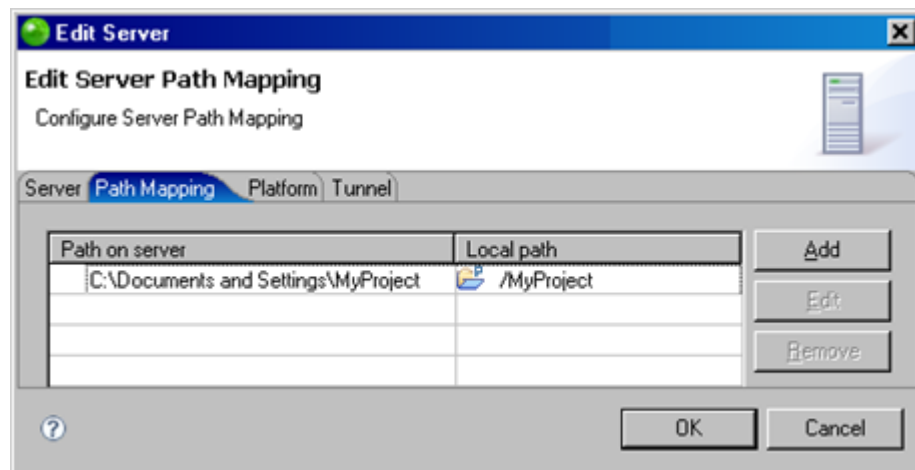


Figure 36 - Path Mapping Server Settings

During Remote PHP Script Debugging, a file is called from location 'C:\Documents and Settings\MyProject\a.php':

```
<?php
echo "b.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](#). See '[Adding a Server Location Path Map](#)' 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 for Eclipse 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.

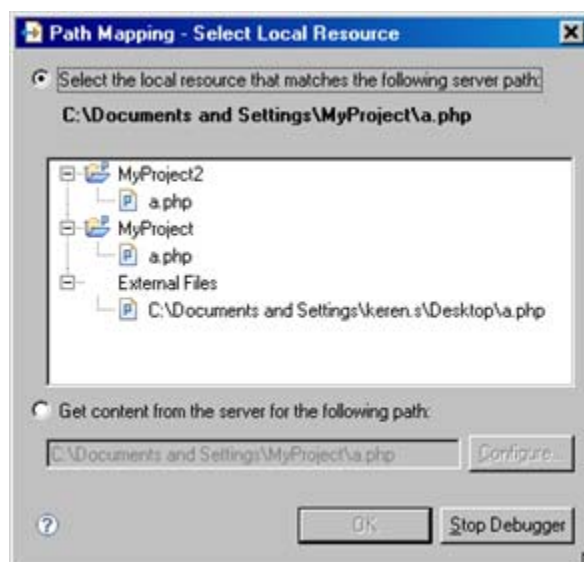


Figure 37 -

Figure 38 - Path Mapping file options dialog

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, it's 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, during the debugging/profiling process, whenever this path is called, it will be searched for on the server only, using PHP's search mechanism. (See <http://il2.php.net/manual/en/function.include.php> for more on PHP's search mechanism.)

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.

Include Paths

An Include Path is a location which connects elements created in external projects/libraries to the active project.

Projects/libraries added to a project's Include Path affect the following:

- [Debugging/Profiling](#) - If a 'require'/'include' call calls a file that exists outside of the active project and its location is specified in a 'Relative' way (see below), you must add its location to the project's Include Path in order for the correct file to be called during debugging.
- [Organizing Includes](#) - If a 'require'/'include' call calls a file that exists outside of the active project and its location is specified in a 'Relative' way (see below), you must add its location to the project's Include Path in order for the correct file to be included.
- [Code Assist](#) - Adding projects/libraries to a project's Include Path will make elements defined within the included projects/libraries available as Code Assist options to the project.

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

- Absolute Path**- The exact file location is specified (e.g. C:\Documents and Settings\MyProject\myfolder\a.php). 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, the [Path Mapping](#) mechanism will be activated.
- 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()".
- Relative Path** - Only the file name or partial path is specified (e.g. /myfolder/a.php). In this case, Zend Studio for Eclipse will search for the file's path in the project's Include Path and then in the project itself, as explained below.

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 for Eclipse encounters during the search process.

The order in which Zend Studio for Eclipse 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.
See [Adding Elements to a Project's Include Path](#) for more on defining the projects / libraries in a project's Include Path.
- ii. The "debug target" file's project.

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

Tunneling (Communication Settings)

Tunneling provides a means of creating a connection between Zend Studio for Eclipse and a remote server situated behind a security device such as a Firewall or NAT. Tunneling creates a secure communication tunnel that keeps a persistent connection with the designated communication port.

Note:

A communication tunnel cannot currently be created to a Windows 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 in order for remote debugging and profiling capabilities to function.

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

2. When communicating between Zend Studio for Eclipse and Zend Platform, when Zend Platform is situated on a remote server which is behind a firewall or other security device. The communication between Zend Studio for Eclipse and Zend Platform facilitates the integration that combines Zend Platform's event reporting capabilities with Zend Studio's editing, debugging and profiling features. This enables the viewing and debugging/profiling of Platform events in Zend Studio for Eclipse. See [Zend Platform Integration](#) for more information.

To set up a tunneling connection, several configuration settings need to be defined both in Zend Studio for Eclipse and on your server's debugger. See [Setting Up Tunneling](#) 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 the 'Working in a Team Environment with CVS' in the Workbench User Guide for more information.

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 Working with SVN topic for more on using SVN.
For more information, see the Subversive User Guide.

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 topic for more information.

Zend Guard Integration

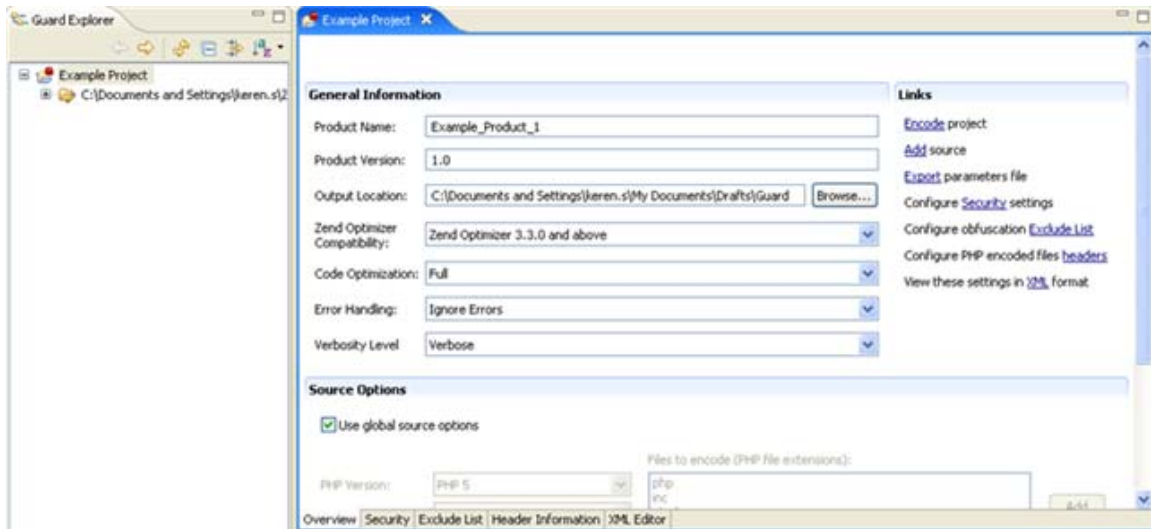
Zend Guard is the first Electronic Licensing solution for the PHP marketplace. It includes the Encoding solution that pioneered PHP intellectual property protection.

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.

Zend Studio for Eclipse's integration with Zend Guard allows you to apply Zend Guard's encoding functionality to projects and applications created and stored in Zend Studio for Eclipse by allowing you to open them in Zend Guard.

Conversely, files stored in Zend Guard can be opened and edited in Zend Studio for Eclipse.



Zend Guard

RSS Feeds

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

Zend Studio for Eclipse 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 for Eclipse's internal browser.

Each new item will be listed according to the RSS channel or time.

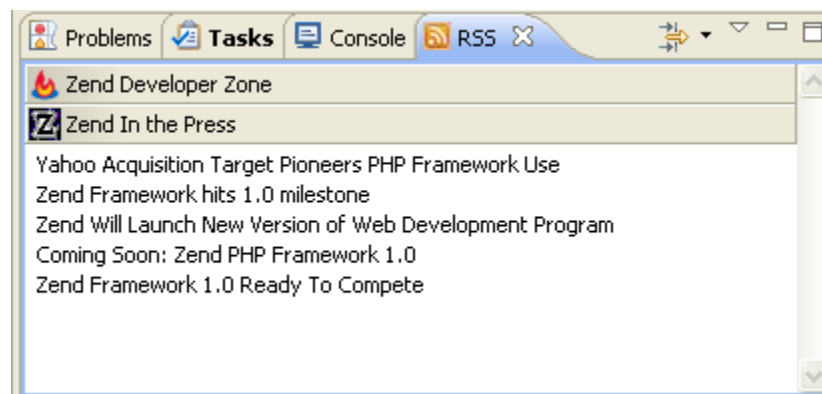


Figure 39 - RSS view

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 for Eclipse provides an integrated means for incorporating and inspecting WSDL files and a wizard for generating your own WSDL files.

Update Manager

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

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

i5/OS Edition Extras

Note:

The features listed below are only available in the i5/OS Edition of Zend Studio for Eclipse.

The i5/OS Edition of Zend Studio for Eclipse contains additional extras to help you connect to and use i5/OS functionality.

Zend Studio for Eclipse - i5/OS Edition contains the following extras:

1. [The ability to register for a free Zend Studio for Eclipse - i5/OS Edition license](#)
2. [Templates for i5/OS PHP API Toolkit functions](#)
3. [Code Assist](#) for:
 - The Zend 5250 Bridge
 - i5/OS PHP API Toolkit functions

Free Registration

System i users who have downloaded Zend Studio for Eclipse - i5/OS edition are entitled to a free license. To receive your free license, go to [Help | Register](#) and click on the link to the Zend registration website. Simply enter your System i server serial number to receive your free license.

i5/OS PHP API Toolkit functions Templates

For information on Templates and how to use them, see the ['Using Templates'](#) topic.

Zend Studio for Eclipse i5/OS Edition contains templates for the following i5/OS PHP API Toolkit functions:

i5/OS Template	Explanation
i5ActiveJobs	Enables retrieving the system's active jobs, it: <ol style="list-style-type: none"> 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, it: <ol style="list-style-type: none"> 1. Connects to i5 server 2. Closes connection to i5 server
i5DataAreaCreate	Creates the data area, it: <ol style="list-style-type: none"> 1. Connects to i5 server 2. Creates data area of given size 3. Closes connection to i5 server
i5DataAreaDelete	Enables deleting the data area, it: <ol style="list-style-type: none"> 1. Connects to i5 server 2. Deletes data area 3. Closes connection to i5 server
i5DataAreaRead	Enables reading from a data area, it: <ol style="list-style-type: none"> 1. Connects to i5 server 2. Reads from data area 3. Closes connection to i5 server
i5DataAreaWrite	Enables reading from a data area, it: <ol style="list-style-type: none"> 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, it: <ol style="list-style-type: none"> 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, it: <ol style="list-style-type: none"> 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, it: <ol style="list-style-type: none"> 1. Connects to i5 server 2. Puts data to the data queue without key

	<ol style="list-style-type: none"> 3. Closes connection to i5 server
i5DtaqSendKey	<p>Enables putting data into the data queue without a key, it</p> <ol style="list-style-type: none"> 1. Connects to i5 server 2. Puts data to the data queue without key 3. Closes connection to i5 server
i5JobLogs	<p>Enables retrieving job log entries, it:</p> <ol style="list-style-type: none"> 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	<p>Enables getting an array with the message element for an object list entry, it:</p> <ol style="list-style-type: none"> 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	<p>Enables calling a program and accept results from it, it:</p> <ol style="list-style-type: none"> 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	<p>Creates Web Services class enabling invoking an RPG program, it:</p> <ol style="list-style-type: none"> 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	<p>Enables getting spool file data from the queue and getting the data from the spool file, it:</p> <ol style="list-style-type: none"> 1. Connects to i5 server 2. Creates an pool file lists, of certain output

	<p>queue or for all queues</p> <ol style="list-style-type: none"> 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	<p>Creates a new user space object, it:</p> <ol style="list-style-type: none"> 1. Connects to i5 server 2. Creates new user space object 3. Closes connection to i5 server
i5UserSpaceDelete	<p>Enables deleting a user space object, it:</p> <ol style="list-style-type: none"> 1. Connects to i5 server 2. Deletes user space object 3. Closes connection to i5 server
i5UserSpaceGet	<p>Retrieves user space data, it:</p> <ol style="list-style-type: none"> 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	<p>Enables to add user space data, it:</p> <ol style="list-style-type: none"> 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

Code Assist

Zend Studio for Eclipse - i5/OS edition, contains Code Assist for commonly used i5/OS functionality. Code Assist is available for the i5/OS PHP Toolkit functions (listed above), as well as for connectivity to the Zend 5250 Bridge.

For more information on Code Assist and how to use it, see the [Code Assist](#) topic.

Tasks

[Easy File Creation](#)

[Migrating From Zend Studio](#)

[Using Code Assist](#)

[Finding and Replacing](#)

[Searching for PHP Elements](#)

[Opening PHP Elements](#)

[Generating Getters and Setters](#)

[Formatting Code](#)

[Using Code Folding](#)

[Using Templates](#)

[Adding Comments](#)

[Creating a PHPDoc](#)

[Using Smart Goto Source](#)

[Using Refactoring](#)

[Using Code Galleries](#)

[Creating Zend Framework Projects](#)

[Using Java Bridge](#)

[Using JavaScript](#)

[Using the PHP/HTML WYSIWYG Perspective](#)

[Using the Data Tools Platform](#)

[Integrating with Zend Platform](#)

[Using PHPUnit Testing](#)

[Using the Profiler](#)

[Using the Debugger](#)

[Adding a Server Location Path Map](#)

[Adding Elements to a Project's Include Path](#)

[Setting Up Tunneling](#)

[Using CVS](#)

[Using SVN](#)

[Using Local History](#)

[FTP and SFTP Support](#)

[Integrating with Zend Guard](#)

[Viewing RSS Feeds](#)

[Working with WSDL](#)

Easy File Creation

PHP files can be created and opened in Zend Studio for Eclipse 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 folder 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.




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

1. 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.



To save the file to a project:

1. Click save. 
 - 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 for Eclipse.

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

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

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



To open a file by dragging-and-dropping:

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

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



To open a file by double-clicking:

1. If the file type you are trying to open was associated with Zend Studio for Eclipse 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 for Eclipse you can:
 - Right-click the file and select Open With | Choose Program | Zend Studio for Eclipse.
 - -Or- Add the file type to the list of file types which will automatically be opened in Zend Studio by doing the following:
 - a. Open your Windows Explorer.
 - b. Go to Tools | Folder Options | File Types.
 - c. From the File Types list, select PHP File.
 - d. In the Opens with category click Change, browse to your Zend Studio for Eclipse .exe location and click OK.
 - e. Click Apply.

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

The file will be displayed in an editor.



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

1. In Zend Studio for Eclipse, 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.

Migrating From Zend Studio

If you have previously used Zend Studio, you can simply and easily migrate existing Studio projects and keymaps into Zend Studio for Eclipse.

[Migrating Projects From Zend Studio](#)

[Migrating Keymaps from Zend Studio](#)

Migrating Projects From Zend Studio

This procedure describes how to quickly and easily import projects from Zend Studio 5.x into Zend Studio for Eclipse 6.0. Any links to CVS and SVN repositories will also automatically be created and maintained.



To import a project from Zend Studio:

1. Right-click in PHP Explorer view and select Import | Zend Imports | Import from Zend Studio 5.x -or- go to File | Import | Zend Imports | Import from Zend Studio 5.x. The Import from Zend Studio Wizard will open.

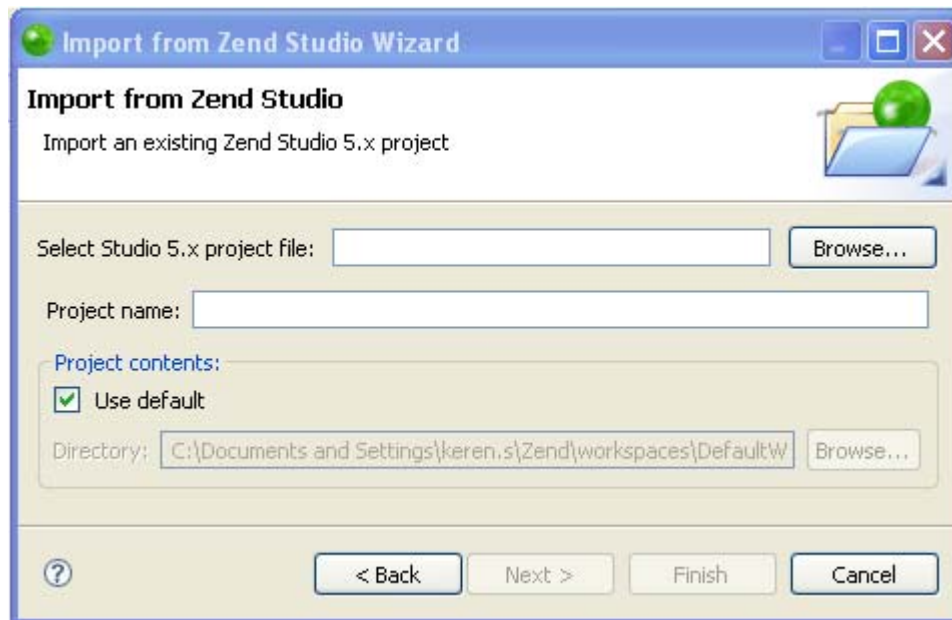


Figure 40 - Zend Studio Import Wizard

2. Click Browse to find your Zend Studio project stored on your file system. The project will automatically have been stored with a .zpj file extension.
3. Click Open
4. The Project name will by default be given the same name as your .zpj file. Edit this name if required.
If the project root contains folders linked to source control, the name given will be the project name specified in this screen with the name of the project root in parentheses following it. E.g. if the project name given was MyProject, and the imported project root was MyProjectRoot, the Project will be displayed as MyProject (MyProjectRoot).
5. If you want the project to be imported into somewhere other than your current workspace,

unmark the 'Use default' checkbox under the Project contents category and browse to a different location.

6. Click Next.

The Import Summary dialog will open.

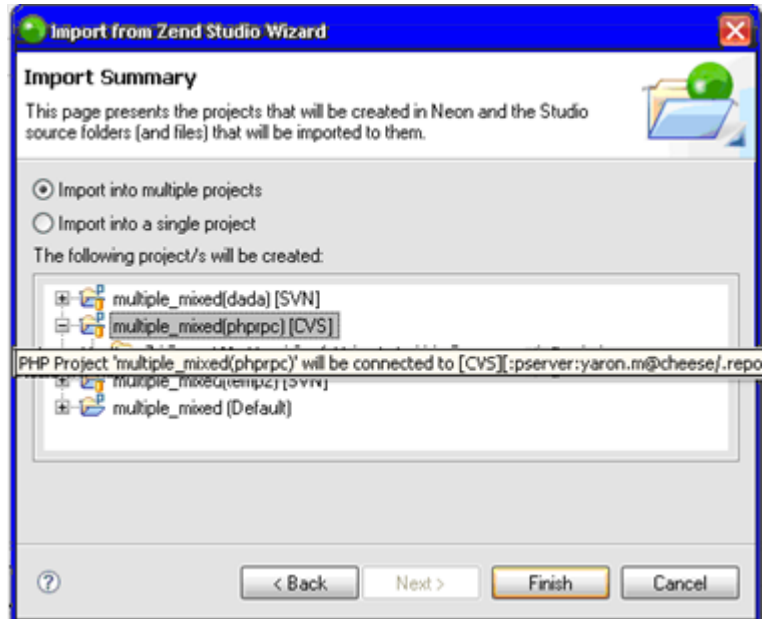


Figure 41 - Zend Studio Import Wizard

7. If the project roots (folders and files) contained within your folder were not mapped to a Version Control System (CVS or SVN), they will all be contained within a single project in Zend Studio for Eclipse. This includes folders and files linked to FTP.
If the project roots (folders and files) contained within your folder are mapped to a Version Control System, each project root will be imported into a separate project.
8. If you would like all projects root to be combined into one Zend Studio for Eclipse project, select the 'Import into a single project' option. Note that in this case the Version Control System mappings will not be created.
9. Click Finish.

Your project(s) will be imported into Zend Studio for Eclipse and will be available in PHP Explorer view. Selected Version Control links will be maintained.

Any required FTP connections will be created in the Remote Systems View. See the [FTP and SFTP Support](#) topic for more on FTP/SFTP connectivity.

Note:

The project root itself needs to be mapped to a Version Control System so that its subfolders can be mapped.

Migrating Keymaps from Zend Studio

Zend Studio keymaps, or collection of command shortcuts, can be easily migrated and applied to Zend Studio for Eclipse for a seamless transition from Studio to Zend Studio for Eclipse.

This procedure describes how to apply a Zend Studio keymap so that Zend Studio's shortcuts will be available in Zend Studio for Eclipse.



To apply a Zend Studio keymap:

1. Go to Window | Preferences | General | Keys.
The Keys preferences page will be displayed.

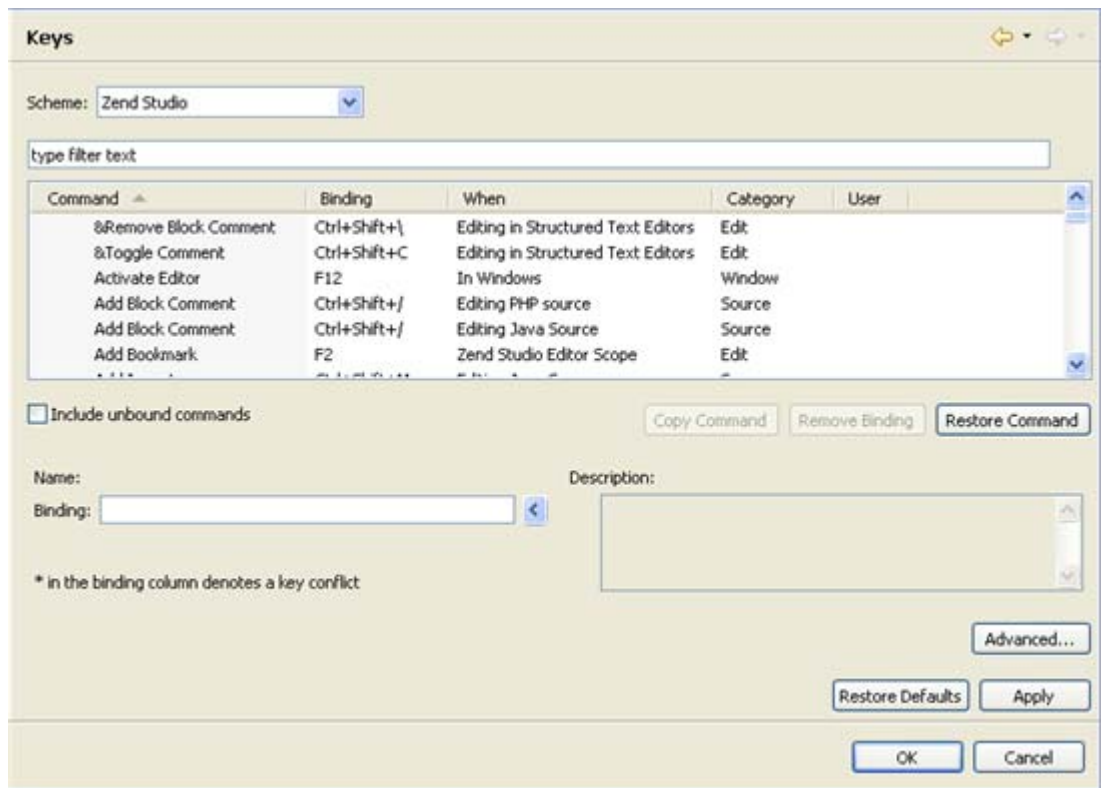


Figure 42 - Zend Studio Keymap

2. Select Zend Studio from the 'Scheme' drop-down list.
3. Click Apply.

Zend Studio's key shortcuts will be applied and available in Zend Studio for Eclipse.

Using Code Assist

This procedure describes how to use Code Assist in order to quickly and easily insert code elements into your script:



To use Code Assist:

1. Enter the first few characters of the required code element into the editor.
2. The Code Assist window should be automatically displayed.
If the Code Assist window does not pop up automatically, press Ctrl+Space.

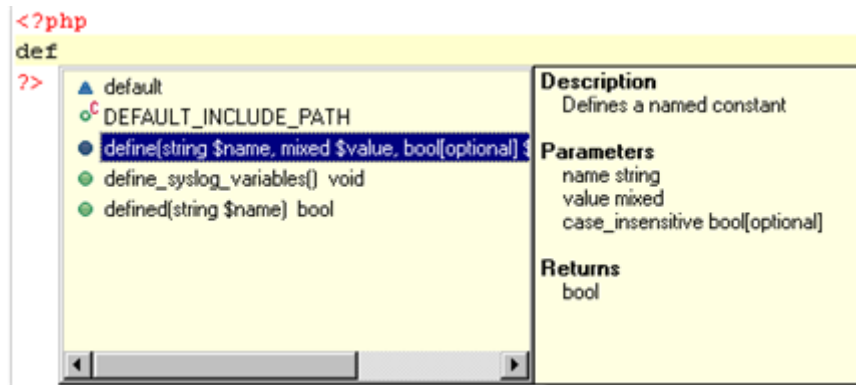


Figure 43 - Code Assist Window

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 Code Assist window to open automatically, go to the [Code Assist Preferences page](#), accessed from Window | Preferences | PHP | Editor | Code 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 | 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 for Eclipse 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 [Code Assist](#) box will appear, listing all available templates and completion options that begin with that combination of keys.

Templates are marked in the code 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 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 | Templates.

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 Ctrl+F or from the Menu Bar go to Edit | Find/Replace.
2. A find and replace dialog will appear.

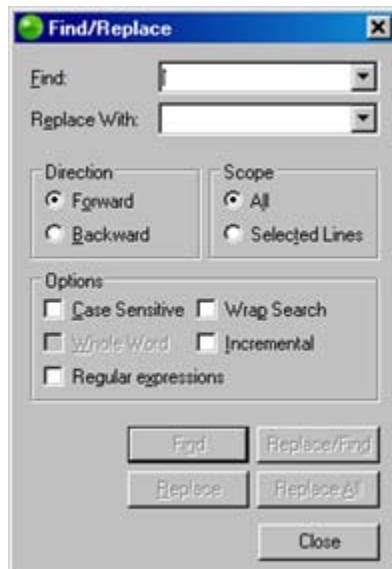


Figure 44 - Find/Replace dialog

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.

Searching for PHP Elements

This procedure describes how to search for PHP elements (classes, functions and constants) within your files and projects.



To search for a PHP element:

1. From the Menu Bar, select Search | Search -or- press Ctrl+H.
A PHP Search dialog will open.

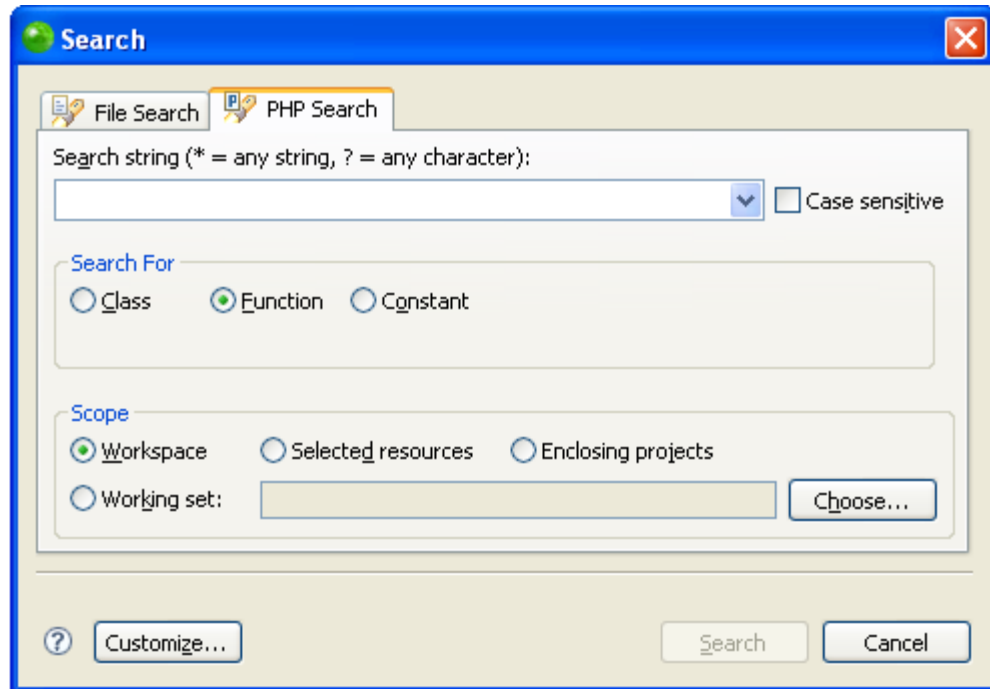


Figure 45 - PHP Search dialog

2. Enter a search string.
3. Select whether to search for a class, function or constant.
4. 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.
5. 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.

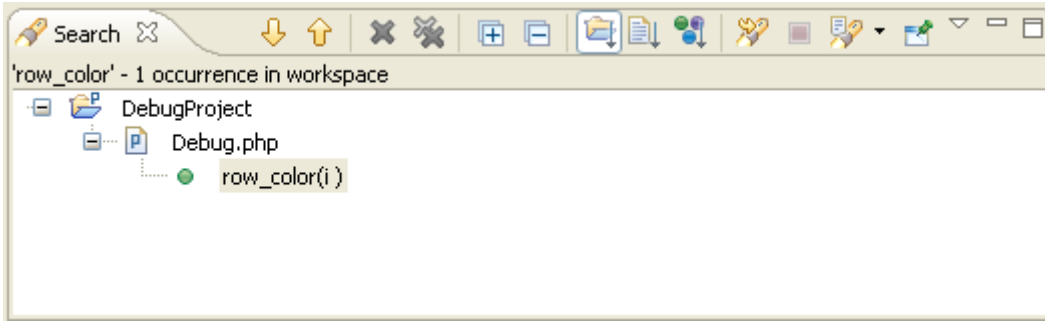


Figure 46 - Search Results

Note:


By default, the File Search dialog will be tabbed with the PHP Search dialog. To make the File Search dialog unavailable, click Customize within the PHP Search dialog and unmark the File Search dialog option.

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.

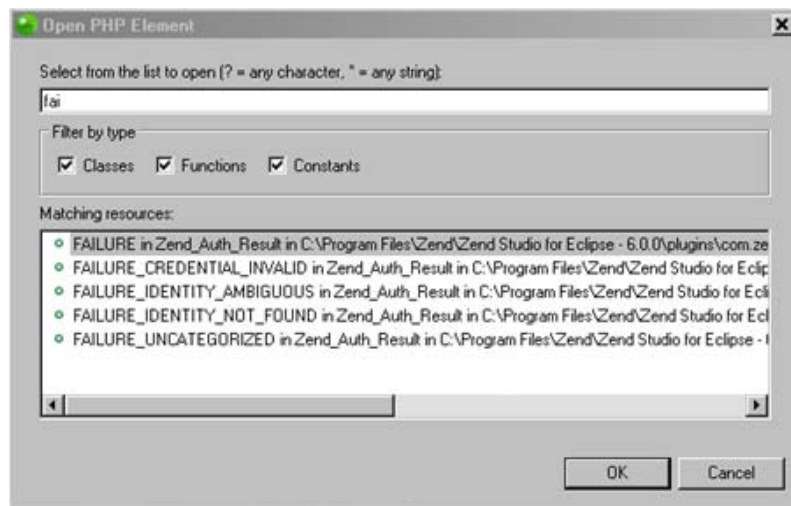


Figure 47 - Open PHP Element dialog

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.

Generating Getters and Setters

Zend Studio for Eclipse 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. 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

```

1  <?php
2  class Person{
3      private $age;
4      private $id;
5      public $name;
6      protected $eyes;
7  }
8
9  ?>
10

```

3. The 'Generate Getters and Setters' dialog will open, displaying all variables that might require a getter/setter.

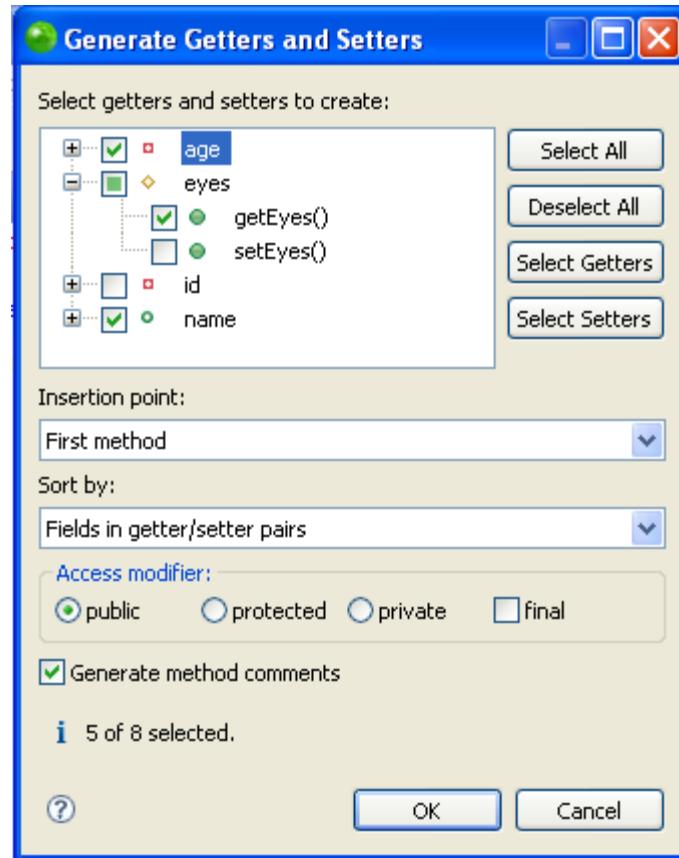


Figure 48 - Generate getters and Setters dialog

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
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 ;
    }
}
```

Figure 49 - Generated getter and Setter

Formatting Code

Zend Studio for Eclipse can auto-format your code according to set standards in order to make it easily navigable and readable.

This procedure demonstrates how to format your scripts.



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 according to the settings defined in the [Formatter Preferences](#) page, accessed from Window | Preferences | Formatter.

<pre> 1 <?php 2 class Calculator { 3 public function add(\$a, \$b) { return \$a + \$b; } 4 public function multiply(\$a, \$b) { return \$a * \$b; } 5 public function divide(\$a, \$b) { 6 if(\$b == null) { throw new Exception("Division by zero"); 7 } 8 return \$a / \$b; } 9 public function subtract(\$a, \$b) { return \$a - \$b; 10 } 11 }> </pre>	<pre> 1 <?php 2 class Calculator { 3 public function add (\$a , \$b) { 4 return \$a + \$b ; 5 } 6 public function multiply (\$a , \$b) { 7 return \$a * \$b ; 8 } 9 public function divide (\$a , \$b) { 10 if (\$b == null) { 11 throw new Exception ("Division by zero") 12 } 13 return \$a / \$b ; 14 } 15 public function subtract (\$a , \$b) { 16 return \$a - \$b ; 17 } 18 } 19 > </pre>
<p><i>Figure 50 - Unformatted Code</i></p>	<p><i>Figure 51 - Formatted Code</i></p>



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.

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 | Folding Preferences.

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.



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.

```

10
11+ public function divide($a, $b) {
17     if($b == null) {
18         throw new Exception("Division by zero");
19     }
20     return $a / $b;
21 }

```

Figure 52 - Folded code hover



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.

```

10
11- public function divide($a, $b) {
12     if($b == null) {
13         throw new Exception("Division by zero");
14     }
15     return $a / $b;
16 }
17

```

Figure 53 - An unfolded function

```

10
11+ public function divide($a, $b) {
17

```

Figure 54 - A folded function



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.

```

1  <?php
2  function abc () {
3      function def () {
4          function ghi () {
5              }
6          }
7      }
8  ?>

```

Figure 55 - Nested Function Folding

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.

Adding Comments

Zend Studio for Eclipse 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.



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.

```

1  <?php
2  //This is a comment
3
4

```

**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.

```

1 <?php
2 //This is a comment
3 //This is another comment
4 //And another one
5

```

**To uncomment a line / lines:**

1. Select the required line(s).
2. Press Ctrl + /
The commenting formatting will be removed from the code.

**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.



Example:

When creating a PHPDoc Comment 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.

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.

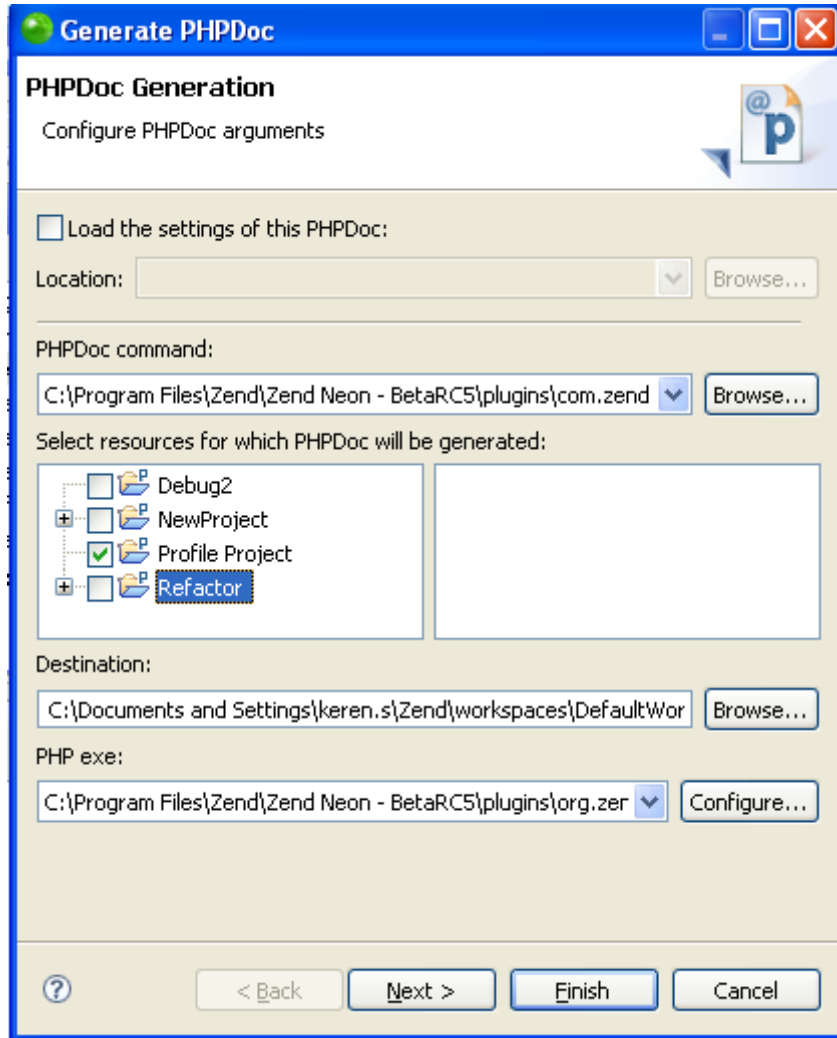


Figure 56 - PHPDoc Generation dialog 1

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.

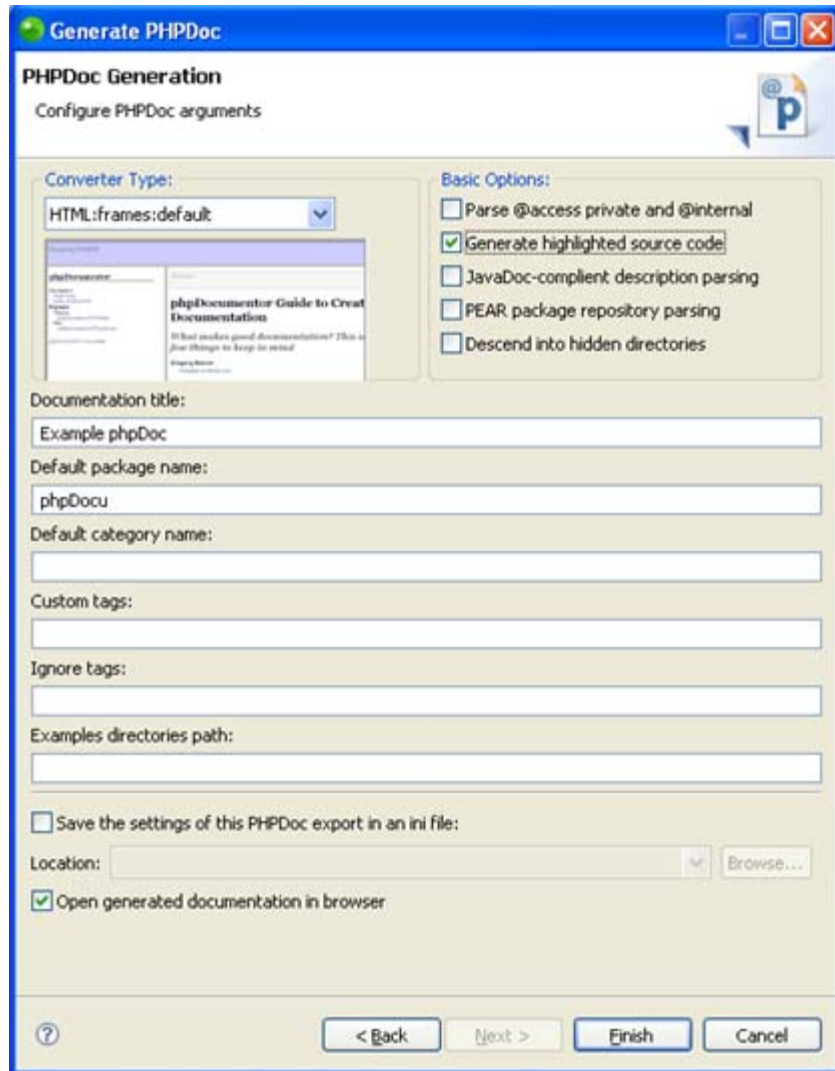


Figure 57 - PHPDoc Generation dialog 2

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 @access 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).

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.

```

1  <?php
2  multiply();
3  Require Calculator::
4
5  >>

```

Location
/Demo Project/Calculator.php
Press 'F2' for focus

Figure 58 - Element with its source 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.

Using Refactoring

The Refactoring feature allows you to:

- Rename and move files and elements.
- Organize Includes so that elements from one file can be referenced in another file.

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.

Note:

Ensure that you save any changes to the file before applying the refactoring feature to it.



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 &endash; or- select it and go to Refactor | Rename from the Menu Bar. A Rename File dialog will appear.

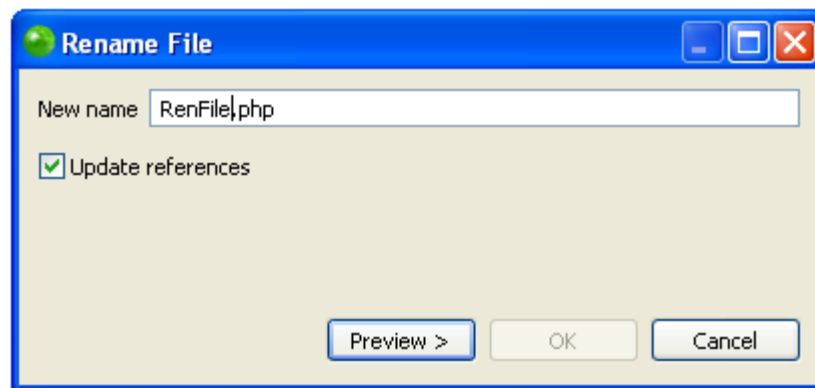


Figure 59 - Rename File dialog

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.

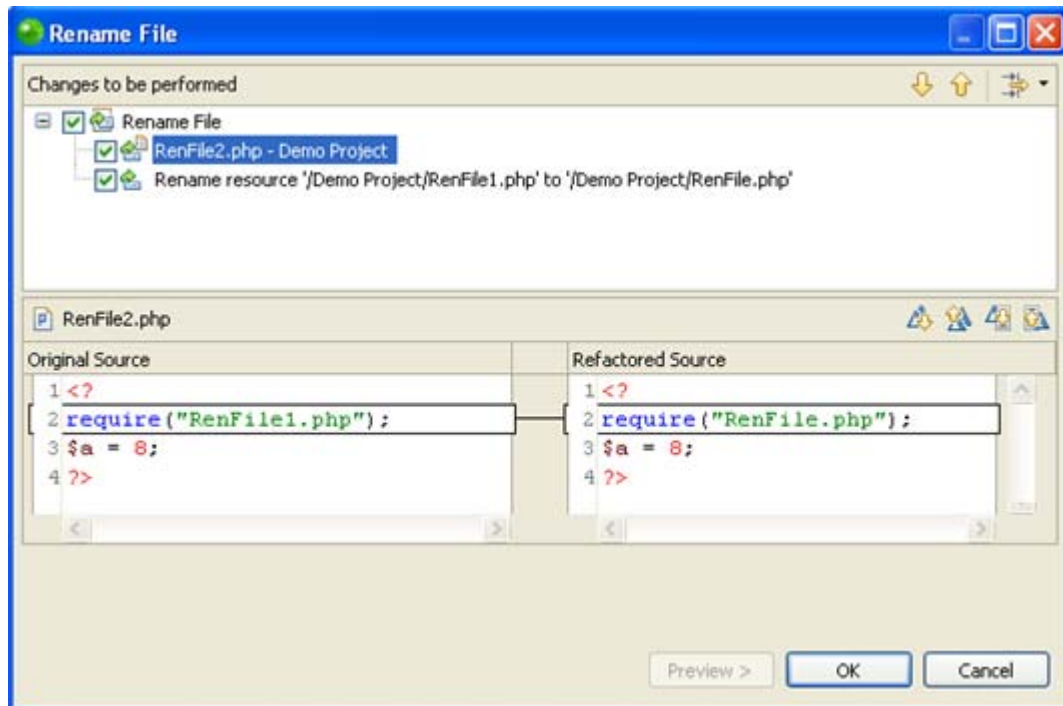




Figure 60 - Rename File change tree

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 within the editor window. The following is a list of 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.

**To rename an element using the Refactoring feature:**

1. In the editor window, right-click the required element and select Refactor | Rename -or- press Alt-Shift-R - or- if applicable, right-click the element in PHP Explorer view and select Refactor | Rename.

The Rename dialog box will be displayed. The name of the dialog will be dependent on the element type.

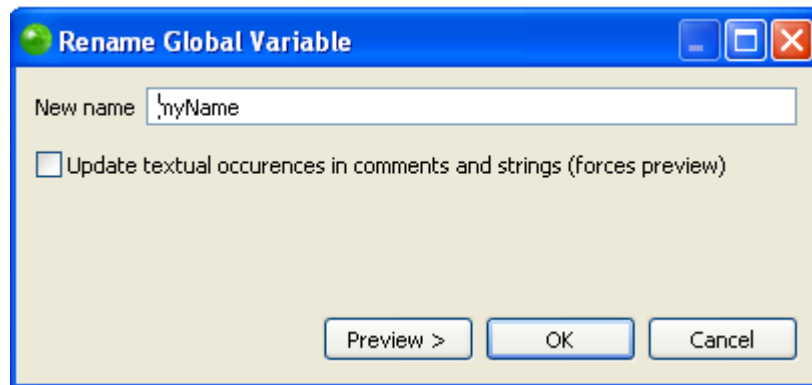



Figure 61 - Rename Global Variable

2. 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.
3. 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.

4. Click OK to apply your changes or click Preview if you want to see a preview of the changes that this refactoring will create.
5. 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.
6. 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.

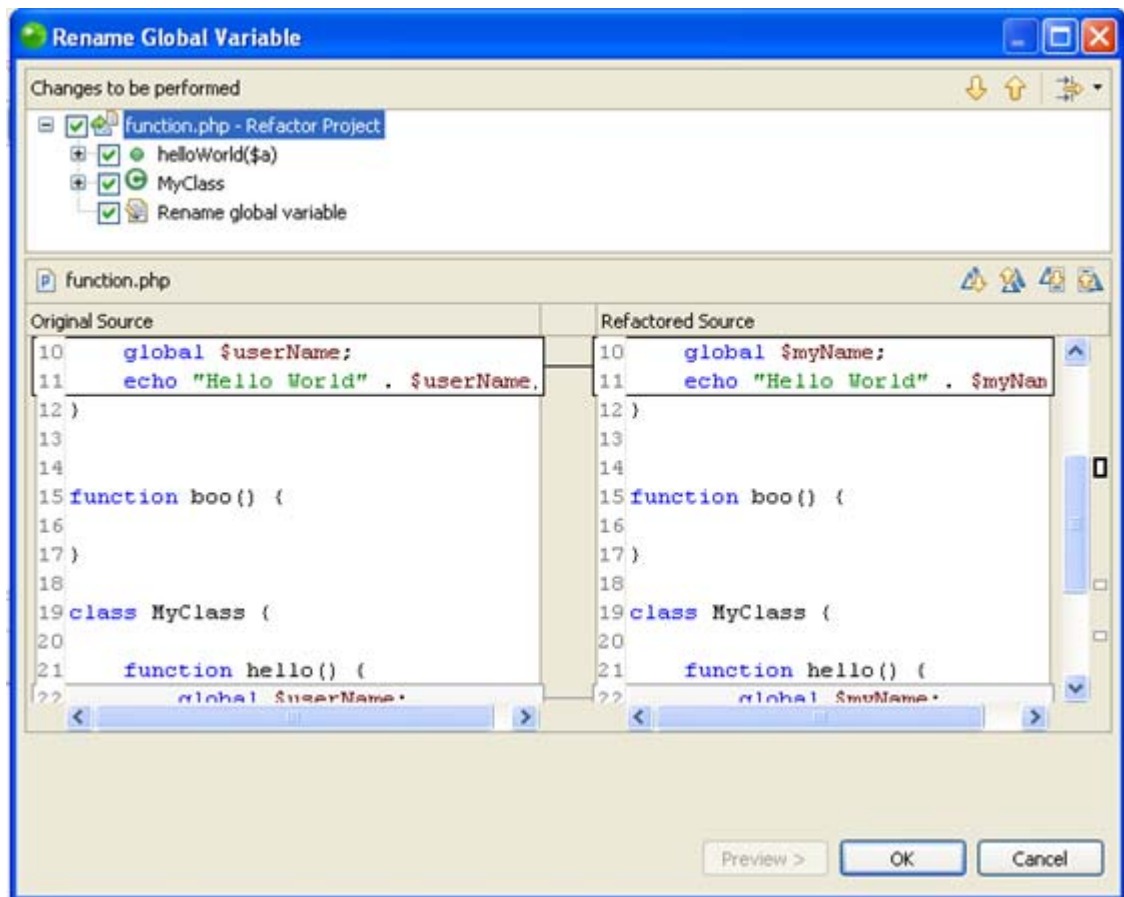




Figure 62 - Rename Global Variable changes tree

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.

7. 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.
8. 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.

Note:

Ensure that you save any changes to the file before applying the refactoring feature.



To move a file using the Refactoring feature:

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.

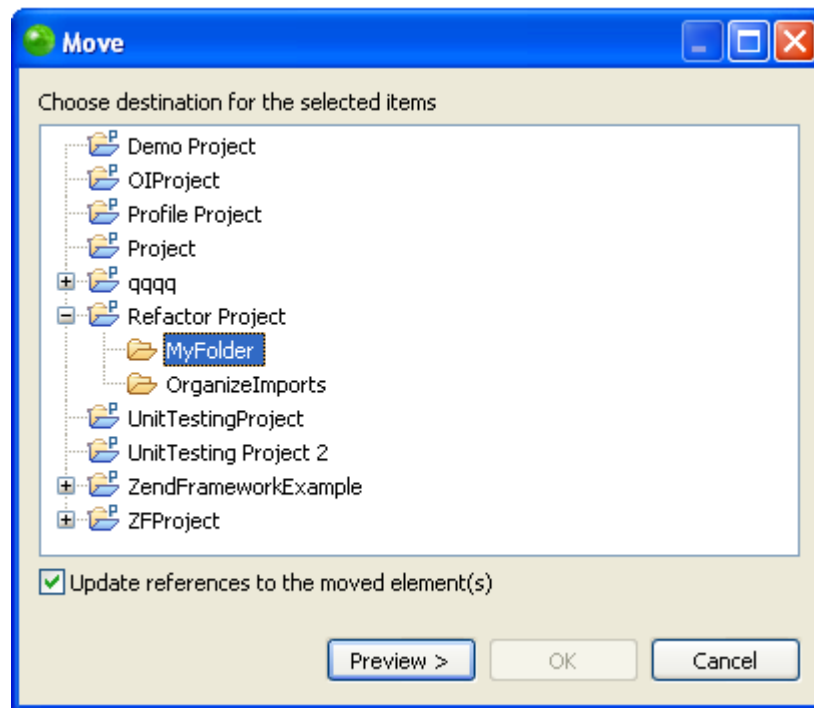


Figure 63 - Move location

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.

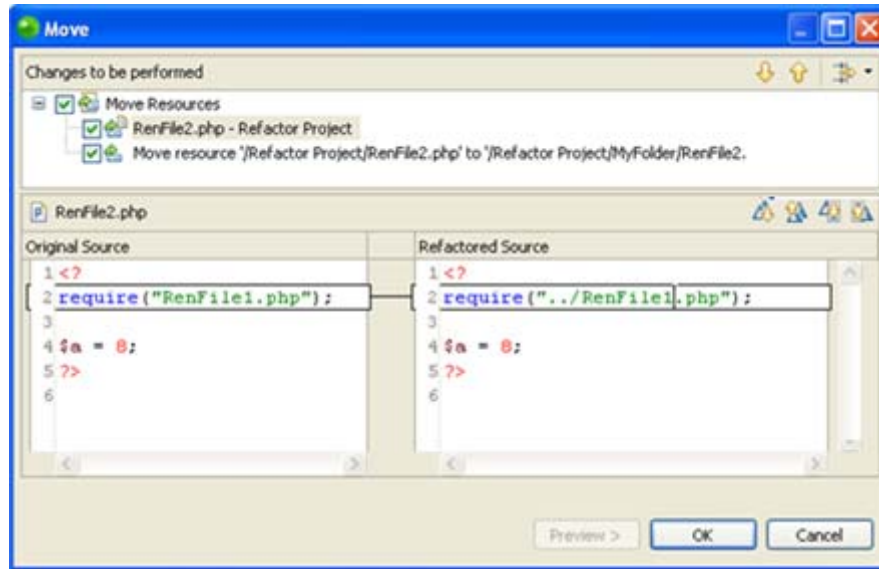


Figure 64 - Move changes tree

- 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.
- Unmark the checkbox of changes which you do not want applied.
- 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.

Organizing Includes

This procedure describes how to use the Organize Includes feature in order for PHP objects created in one file to be called into other files within the project.

Before calling objects into one file, you must first create them in another file.

Note:

Ensure that you save any changes to the file before applying the refactoring feature.



To use the Organize Includes feature:

- In PHP Explorer view, right-click the file and select Refactor | Organize Includes -or- open the file and press Ctrl+Shift+O.

An Organize Includes preview dialog will be displayed with a changes tree showing the changes that will be made when the object(s) is 'included' into the new file.

If two objects of the same name are recognized within a project, a window will display so that the relevant object can be selected.

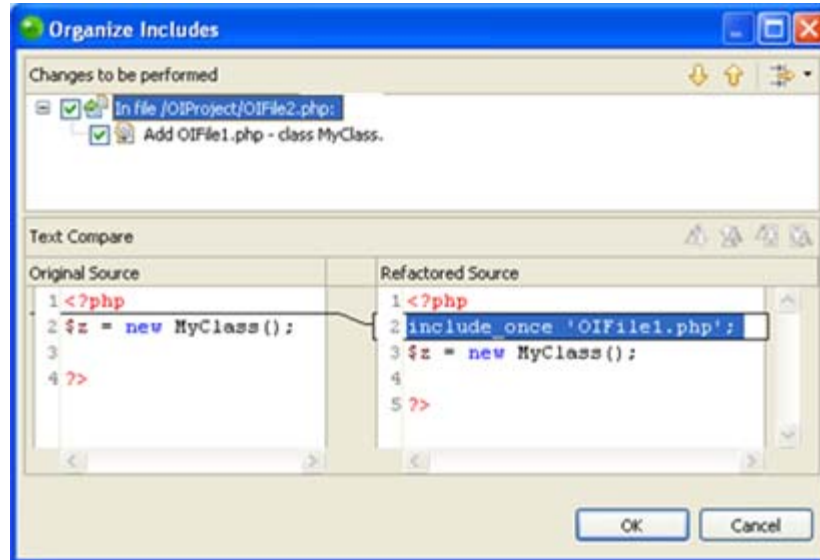



Figure 65 - Organize Includes changes tree

2. You can scroll through the different changes using the scrolling arrows . Note that that include_once will now be added for each object that you are attempting to call, with the name and location of the file in which it was created.
3. Unmark the checkbox of changes which you do not want applied.
4. If you are satisfied with the changes, press OK.
5. Click OK to apply the changes.

All objects that were called into the file from other files will now have an 'include_once' command.

Note:

Only static include calls will be refactored.

The Organize Includes feature can also:

- Delete references and "include" calls to files if the relevant items have been deleted from the code.
- Move "include" calls to the top of your script if they have been placed further down, in order for the elements to be referenced before the script is run.

Note:

If the included files are not contained within the same project as the active file, their location must be added to the project's include path in order for them to be 'Organize Includes' feature to work fully. See the '[Include Paths](#)' and '[Adding Elements to a Project's Include Path](#)' topics for more information.

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.

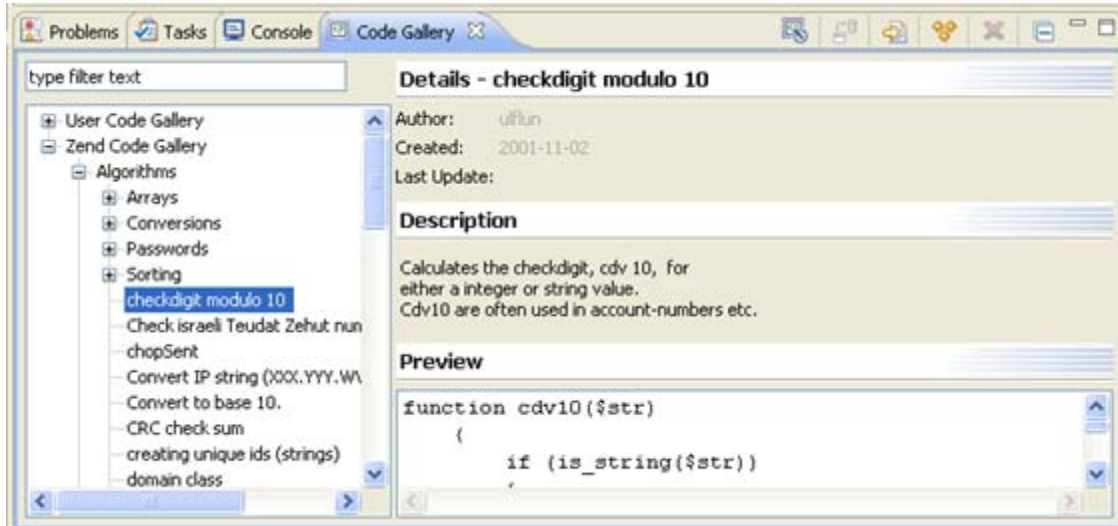



Figure 66 - Code Gallery view

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

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 on the Code Gallery view's toolbar.
A New code gallery entry dialog will open.

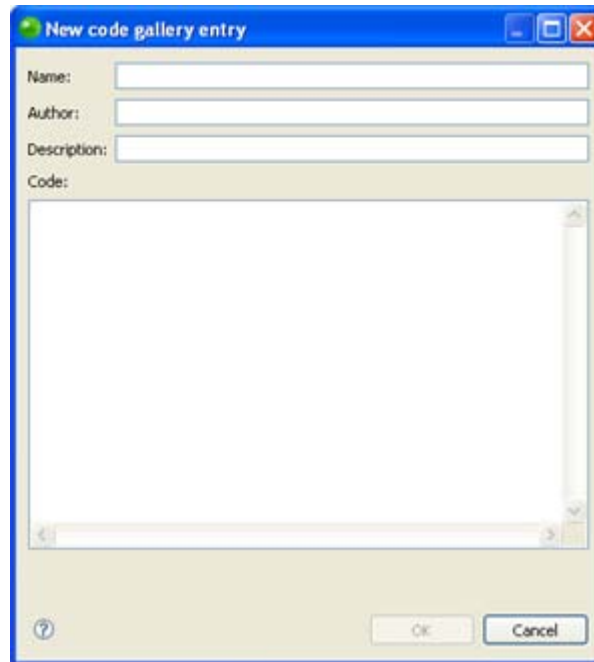


Figure 67 - New Code Gallery Entry

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.

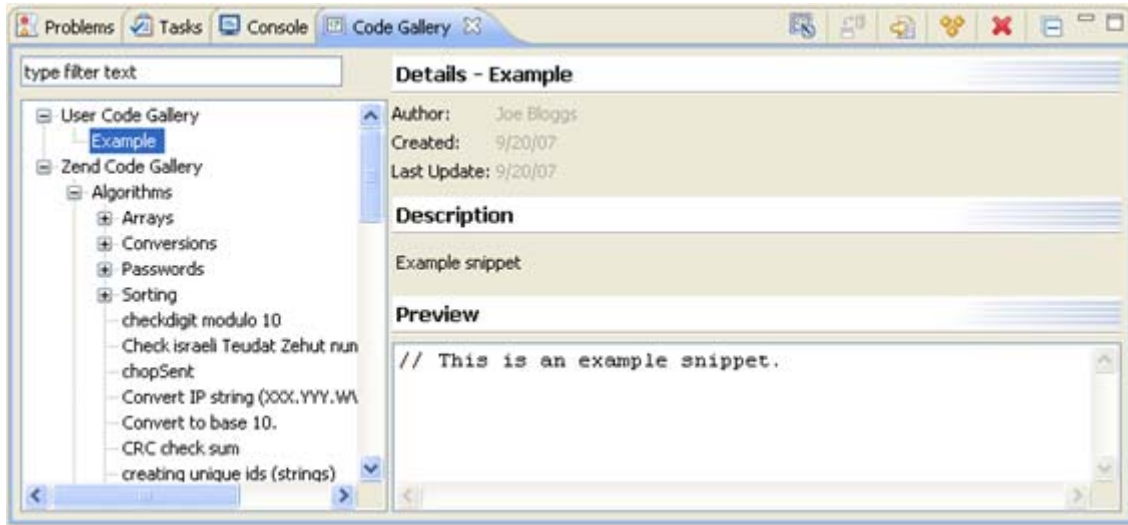


Figure 68 - Code Gallery view



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.

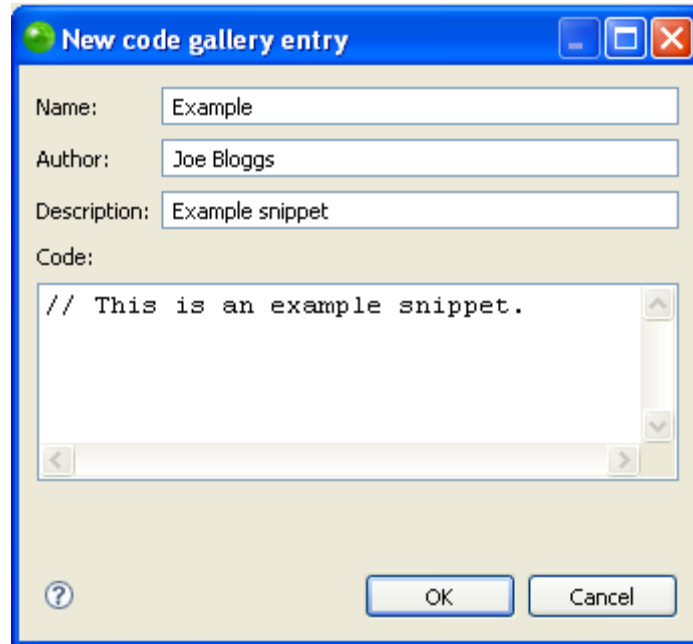


Figure 69 - Code Gallery Edit

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.



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.

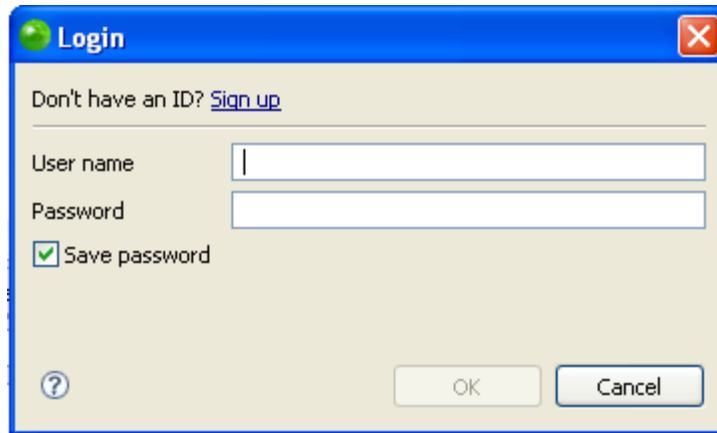


Figure 70 - Zend Network Login

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:

<https://devzone.zend.com/member/register>.

Your Zend Code Gallery list will be updated with all the code snippets from the Zend Code Gallery site, divided into categories.

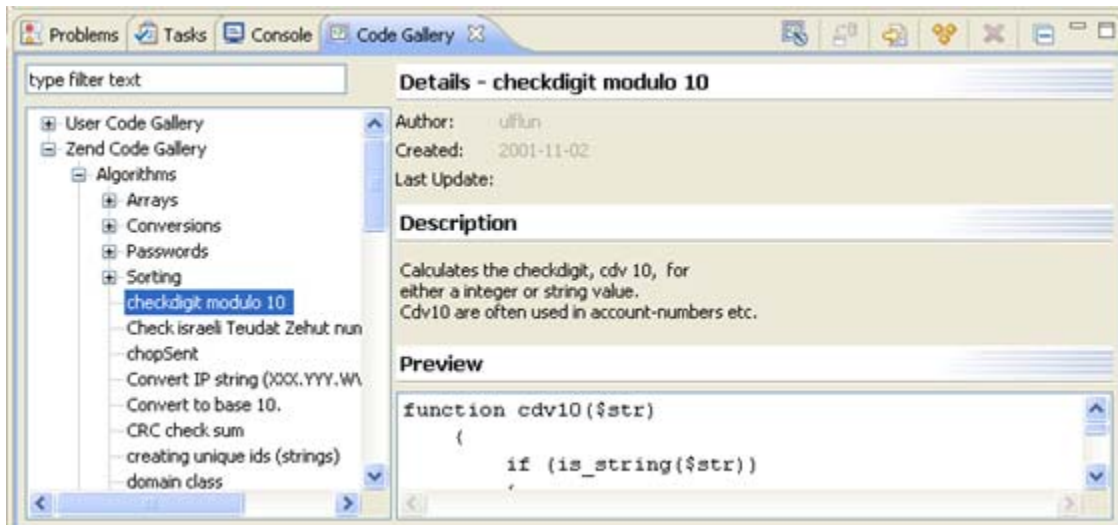



Figure 71 - Code Gallery view




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.



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.



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.



Figure 72 - 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.



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.

The 'Send your rating' dialog will appear.



Figure 73 - Code Gallery Rating

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.

Creating Zend Framework Projects

Zend Framework Projects allow you to access Zend Framework's libraries and formats your project in Framework's Controller, Model, View format.

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:



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 and click Finish.
Your Zend Framework Project will be added to your list in PHP Explorer view.
3. Open the MVC Outline view by going to Window | Show View | PHP Tools | MVC Outline.

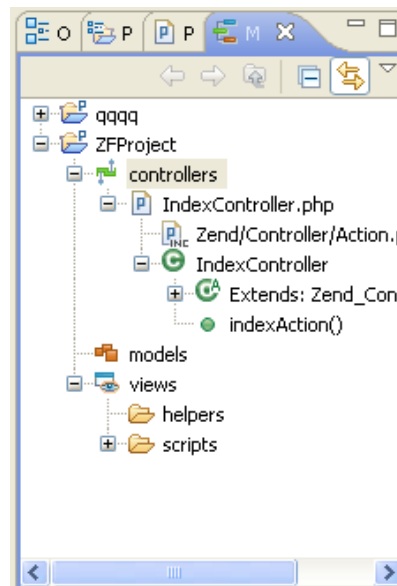


Figure 74 - MVC view

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 by doing the following:

- For a new Controller file, right-click the controllers folder and select New | Zend Controller.
This will create a new file based on the Zend Controller template.
- For a new Model file, right-click the models folder and select New | Zend Model.
This will create a new file based on the Zend Model template.
- For a new View file, right-click the controllers folder and select New | Zend View.
This will create a new file based on the Zend View template.

Note:

To trial using Zend Framework, create the Zend Framework Example Project by going to File | New | Example | Zend Framework | Zend Framework Example project.

For a tutorial on using the Zend Framework example project, see the readme.html file contained within the Zend Framework Example project's readme folder.

See "[Zend Framework Integration](#)" for more information.

Using Java Bridge

Creating Java Objects

This procedure describes how to create a new Java object within a PHP script, which will adopt Java object properties and code assist options.



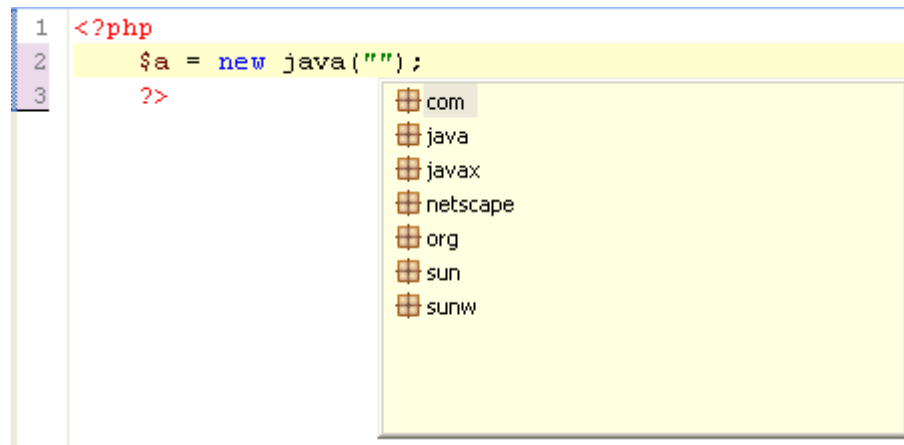
To create a new Java Object:

1. Create a new Java instance by typing the following code:

```
$a= new Java( "" );
```

2. Put the cursor between the quotation marks.

A Code Assist window will appear with available Java packages. You can manually open the Code Assist box by pressing Ctrl+Space.



3. Select the required java package and press enter.
4. Type a period "." after the selected java package to activate the code assist options within that package.

```

1 <?php
2 $a = new java("java.");
3 ?>

```

- Continue with the Code Completion procedure until the appropriate class has been selected.

```

1 <?php
2 $a = new java("java.lang.");
3 ?>

```

```

1 <?php
2 $a = new java("java.lang.System");
3 ?>

```

The PHP variables that the Java objects have been assigned to will assume the properties and methods of the Java class. When creating new instances of the java class, java code assist options will therefore be available.

**Example:**

1. After the line above, create a new instance of the created PHP object (in this case, \$a).

```
$a->
```

2. Place your cursor after the -> and activate Code Assist by clicking Ctrl+Space.

The Code Assist list will be displayed with java code completion:

```

1 <?php
2 $a = new java("java.lang.StringBuffer");
3 $a->
4

```

Class
java

Description
Create Java object

Parameters
classname string

Returns
java

- java(string \$classname) java
- append(bool \$arg0) java.lang.AbstractStringB...
- append(bool \$arg0) java.lang.StringBuffer
- append(float \$arg0) java.lang.AbstractString
- append(float \$arg0) java.lang.AbstractString
- append(float \$arg0) java.lang.StringBuffer
- append(float \$arg0) java.lang.StringBuffer
- append(float \$arg0) java.lang.StringBuffer
- append(int \$arg0) java.lang.AbstractStringBu...
- append(int \$arg0) java.lang.AbstractStringBu...

Zend Studio for Eclipse stores Java variables in the same way as PHP variables, allowing for code completion functionality for nested method invocation:

```

25 $system = new java("java.lang.System");
26 $properties = $system->getProperties();
27 $javahomeFolder = $properties->getProperty("java.home");
28 $javahomeFolder = $properties->g
29
30
31
32
33
34
35
36
37
38
39

```

Location
java.util.Properties

- getProperty(string \$arg0) string
- getProperty(string \$arg0, string \$arg1) string

Adding External Java Archives

These procedure describe how to add external Java Archives (JARs) to your new or existing PHP projects' Java Bridge classpaths so that additional Java objects can be created.



To add JARs to a new project:

1. Create a new PHP project by going to File | New | PHP Project.
The New PHP Project dialog will open.
2. Enter the project name and change the project contents or PHP version settings if required.
3. Click Next.
4. Add variables to the project's PHP Include Path if required.
5. Click Next.
The Java Bridge Support dialog will open containing the list of available JAR libraries.

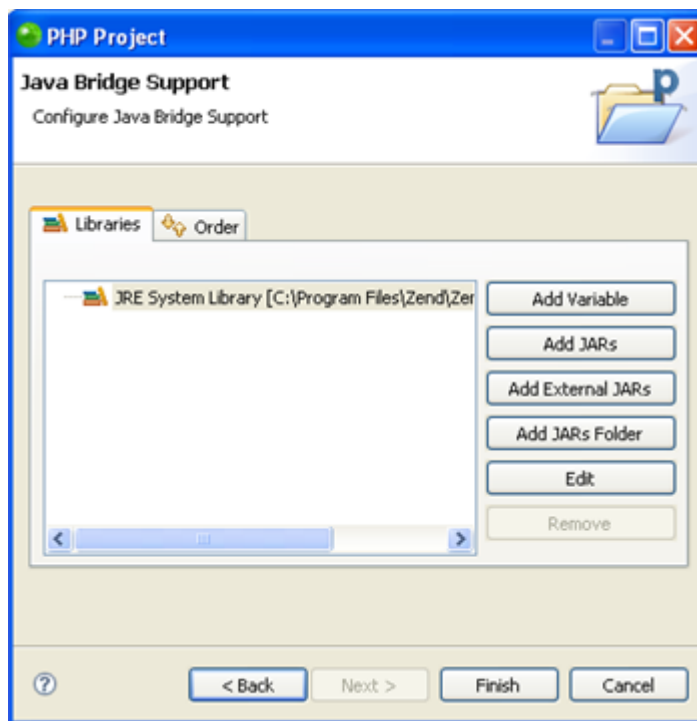


Figure 75 - New PHP Project - Java Bridge options

6. Click Add External JARs.
7. Browse for the JARs on your file system and click Open.
Your JAR will be added to the list.
8. Click Finish.

Your new PHP project will be created including the new JAR archive. Elements within the JAR will be available for use within Java objects for that project.



To add JARs to an existing project:

1. Right-click your project in PHP Explorer view and select Properties -or- click Alt-Enter.
2. Select PHP Java Bridge from the properties list.

The list of available JAR libraries will be displayed.

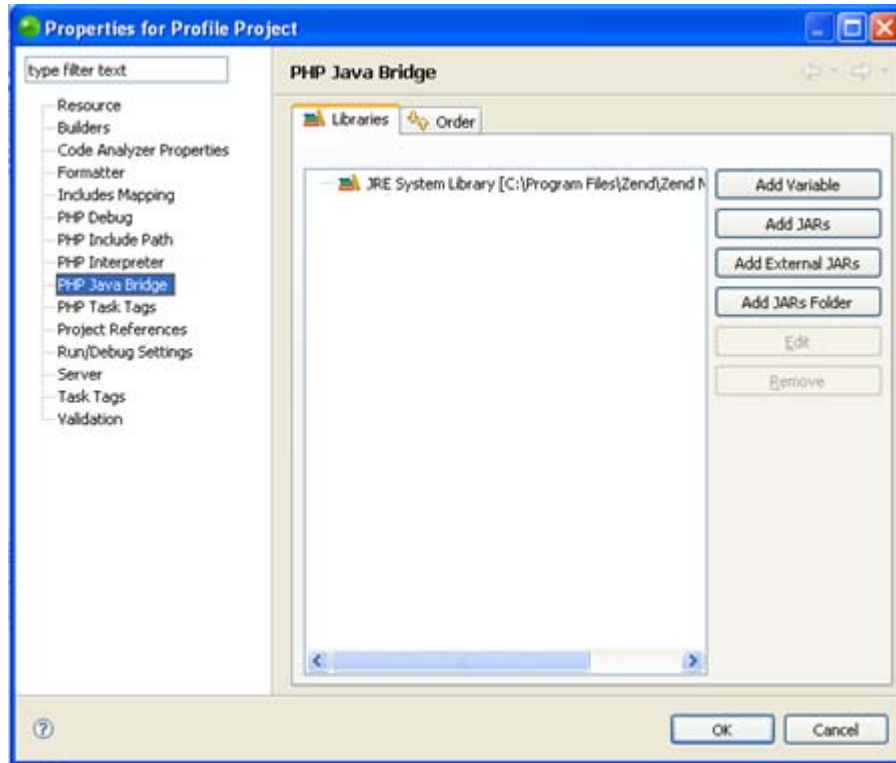


Figure 76 - PHP Java Bridge Project Properties

3. Click Add External JARs.
4. Browse for the JARs on your file system and click Open.

Your JAR will be added to the list and will be available for use within Java objects for that project.

Note:

The JAR will have been added for the specified project only.

Changing Projects' JREs

Different Java Runtime Environments can be applied to project by default or can be applied or changed for specific projects.

These procedures describe how to apply different JREs to a specific project. JRE settings can be set while creating a new project or changed once a project has been created.

Before JREs can be applied to a project they need to be configured in the [Installed JREs Preferences](#) page, accessible from Window | Preferences | PHP | Installed JREs.



To apply a JRE to a new project:

1. Create a new PHP project by going to File | New | PHP Project.
The New PHP Project dialog will open.
2. Enter the project name and change the project contents or PHP version settings if required.
3. Click Next until you reach the Java Bridge Support screen.

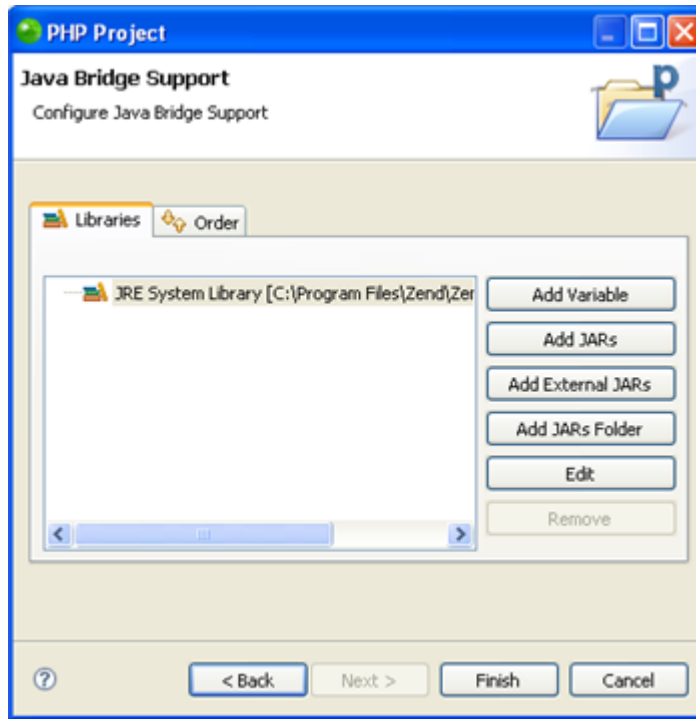


Figure 77 - New PHP Project - Java Bridge options

4. Select the JRE System Library and click Edit.
The Edit Project JRE dialog will appear.

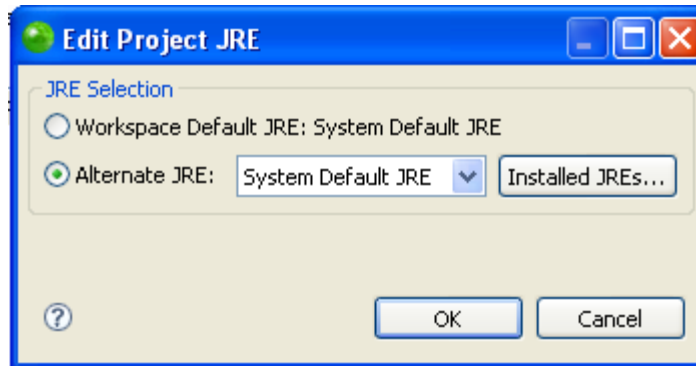


Figure 78 - Edit Project JRE

5. Select 'Alternate JRE' and choose the required JRE from the drop-down list.
6. If the required JRE does not appear in the list, click Installed JREs to be taken to the Installed JREs Preferences page.

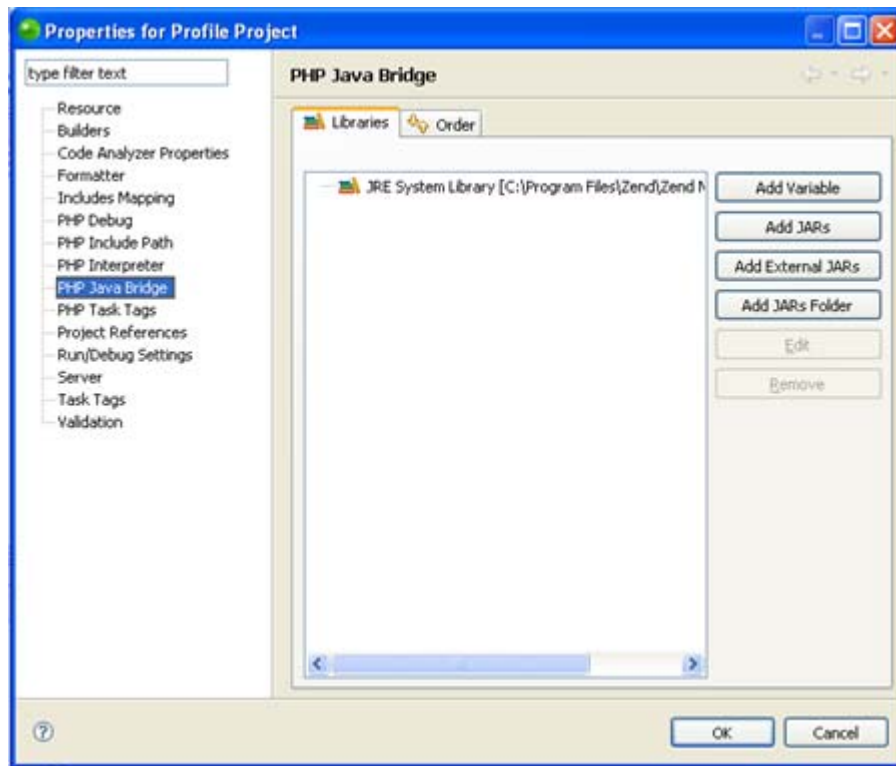
7. Click OK.
8. Click Finish to create the project.

Your new PHP project will be created with the required JRE applied to the project.



To apply a JRE to an existing project:

1. Right-click your project in PHP Explorer view and select Properties -or- click Alt-Enter.
2. Select PHP Java Bridge from the properties list.



PHP Java Bridge Project Properties

3. Select the JRE System Library and click Edit.
4. The Edit Project JRE dialog will appear.

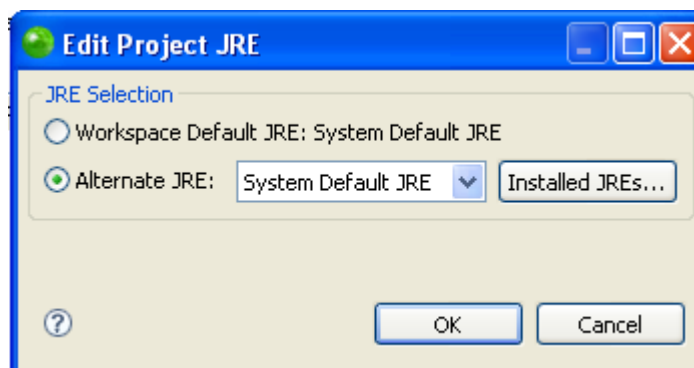


Figure 79 - Edit Project JRE dialog

5. Select 'Alternate JRE' and choose the required JRE from the drop-down list.
6. If the required JRE does not appear in the list, click Installed JREs to be taken to the [Installed JREs Preferences page](#).
7. Click OK.
8. Click Finish to create the project.

Your project properties will be altered to include the alternate JRE.

Note:

The JRE will have been changed for the specified project only.

Using JavaScript

These procedures describe how to access JavaScript Code Assist options from within your PHP files, and how to view your JavaScript objects.



To access JavaScript Code Assist options:

1. Type the relevant HTML and JavaScript tags:

```
<HTML>
<script type="text/javascript">
|
</script>
</HTML>
```

2. JavaScript Code Assist and relevant syntax coloring will now be available. Type the first few letters of a JavaScript string and press Ctrl+Space to activate Code Assist. JavaScript Code Assist options will be preceded by either a blue triangle ▲, indicating a reserved JavaScript word, object or variable, or a green circle ●, indicating a function.
 3. Select the relevant option from the Code Assist window by double-clicking or pressing Enter.
 4. If you selected a JavaScript class, type a period "." after the name of the class to display a Code Assist window with the classes' relevant functions and methods.

```
<HTML>
<script type="text/javascript">
window.
</scrip
</HTML>
```


- alert()
- atob()
- attachEvent()
- back()
- blur()
- btoa()
- captureEvents()
- clearInterval()
- clearTimeout()
- ▲ clientInformation

5. Select the required option to complete your JavaScript code.

JavaScript objects and elements can be viewed within the HTML section of the Outline view.



To view your JavaScript objects in the Outline view:

1. Go to the Outline view.
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.

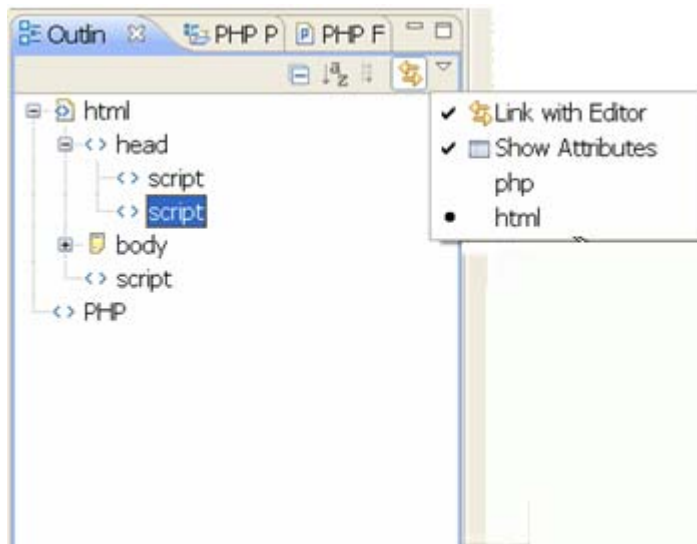


Figure 80 - Outline view - HTML

3. Double-Clicking the <Script> node in the outline view will select the entire <script> element in the Editor.

Using the PHP/HTML WYSIWYG Perspective

The PHP/HTML WYSIWYG Perspective allows you to create and edit HTML pages by allowing you to carry out the following actions:

[Create and open HTML files in the WYSIWYG Editor](#)

[Insert HTML Objects](#)

[Configure HTML Properties Using the Properties View](#)

[Edit CSS](#)

Note:

More information about writing HTML can be found at: <http://www.w3schools.com/html>

Creating and opening HTML files in the WYSIWYG Editor

These procedures describe how to create new HTML files and how to open them in the PHP/HTML WYSIWYG Editor, allowing implementation of Zend Studio for Eclipse'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 -or- right-click the folder and select File | New | HTML.
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.



To open an HTML file with the PHP/HTML WYSIWYG Editor:

1. Right-click the file in PHP Explorer view.
2. Select Open With | PHP/HTML WYSIWYG Editor.
3. Click Yes when asked whether to open the PHP/HTML WYSIWYG Perspective.

Your file will be opened in a PHP/HTML WYSIWYG Editor, containing the following tabs:

- Design - Shows a visual outline of the text and HTML objects.
- Source - Shows the source code.
- Design/Source - Shows a split screen with both Design and Source displays.
- Preview - Shows a preview of what the page would look like in a browser.

Note:

More information about writing HTML can be found at: <http://www.w3schools.com/html>

Inserting HTML Objects

The PHP/HTML Toolbox view contains a list of HTML object which can be easily drag-and-dropped into your code to insert HTML objects. In addition, the Insert menu available in the PHP/HTML WYSIWYG Perspective can also be used for this function.

This procedure describes how to insert HTML objects into your code, using both the PHP/HTML Toolbox view and the



To insert HTML objects into your script:

1. Ensure your HTML file is opened in the PHP/HTML WYSIWYG Editor by following the steps under "[Creating and opening HTML files in the WYSIWYG Editor](#)".
2. Open the PHP/HTML WYSIWYG Perspective by going to Window | Open Perspective | PHP/HTML WYSIWYG.
3. Select the Design/Source pane in the PHP/HTML WYSIWYG Editor.

```
11  
12 <br>  
13 <a href="#">anchor</a>  
14 <input type="button" value="button">&nbsp;  
15 </body>  
16 </html>
```

The screenshot shows the PHP/HTML WYSIWYG Editor interface. The code editor displays HTML code with line numbers 11 through 16. Line 14, containing the code `<input type="button" value="button"> `, is highlighted in yellow. Below the code editor, there are four tabs: Design, Source, Design/Source, and Preview. The Design/Source tab is selected and highlighted with a red box.

Figure 81 - PHP/HTML WYSIWYG Editor

4. The PHP/HTML Toolbox view contains a variety of different HTML objects, divided into the following categories:
 - Head
 - HTML
 - Table
 - Form
 - Special Characters
 - Media

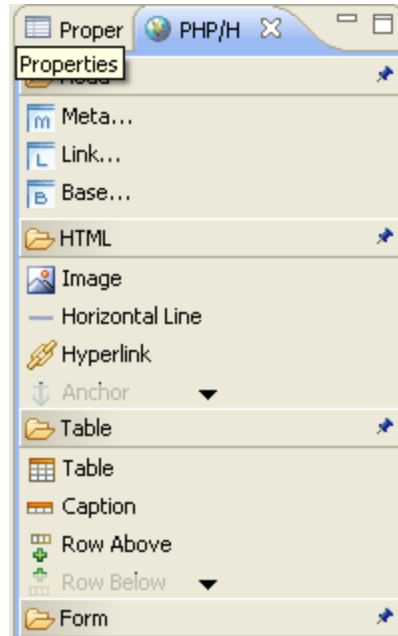


Figure 82 - PHP/HTML WYSIWYG Properties view

5. Select an HTML object from the PHP/HTML Toolbox view and drag-and-drop it into the required location in the editor's Design pane
 - Or- place your cursor in the required location in your editor and double-click the element
 - Or- go to the Insert Menu and select the required element.

The selected object will be displayed in the Design pane and the relevant code will be added to the Source pane.

Note:

More information about writing HTML can be found at: <http://www.w3schools.com/html>

Configuring HTML Properties

The Properties view that is incorporated into the PHP/HTML WYSIWYG Perspective shows a list of editable properties for various HTML elements.

The Properties view will display different tabs and options according to the context of the cursor in the PHP/HTML WYSIWYG Editor.

In an HTML file, the HTML tab will always be available to configure general HTML properties.

In addition, extra property tabs will be added according to the context of the cursor within the HTML script.

Most property tabs will have General and Advances tab options. The General section allows you to edit the element's properties using simple field inputs, dropdown lists and checkboxes.

The Advanced section contains all available properties of the current element as a list which can be easily edited by selecting the value column to the right of the property and inputting or selecting the relevant options. Editing empty lines allows adding a new property to the existing properties list.

The Properties view can be opened by going to Window | Show View | Other | General | Properties.

These procedures describe how to configure various HTML properties.



To configure general HTML Properties:

The HTML tab in the properties view will be available whenever an HTML file is open.

1. Select the HTML tab in the Properties view.

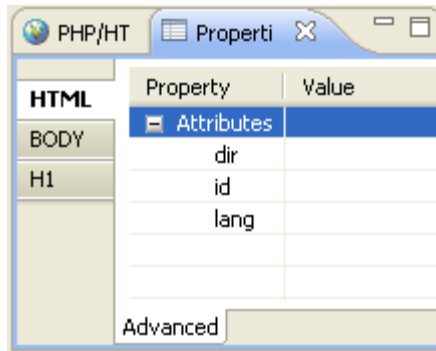


Figure 83 - PHP/HTML WYSIWYG Properties view

2. Click on the Value column to the right of the required property to edit the property values.

The relevant changes will be made to the source code and, if applicable, will also be visible in the Design section of the PHP/HTML WYSIWYG Editor.



To configure the Head/Body properties:

1. Select a location within the Head / Body of the code in the PHP/HTML WYSIWYG Editor.
A Head/Body tab will appear in the Properties view.

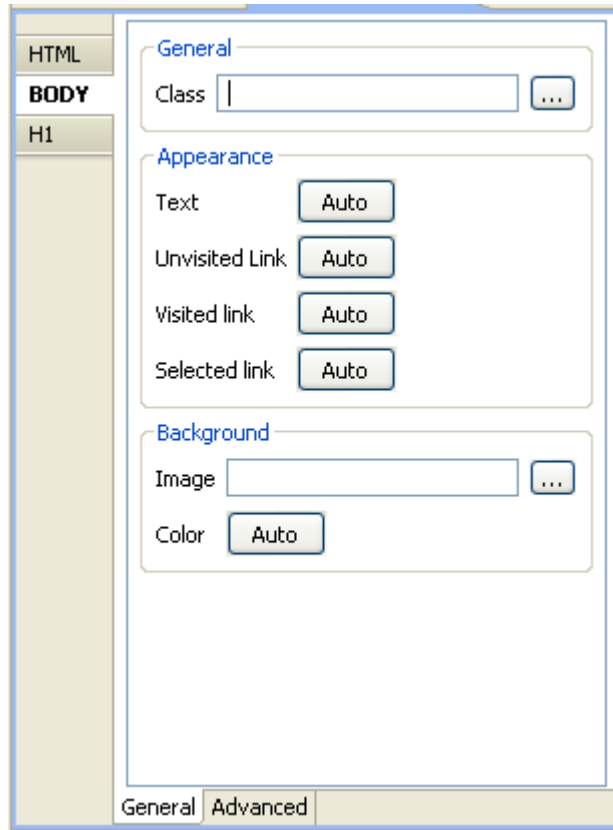


Figure 84 - PHP/HTML WYSIWYG Properties view - Body Tab

2. Select the Head/Body tab to see the properties that can be configured.
3. If required, enter a class name. Multiple class names can be entered.
4. In the Appearance category, choose the colour for the text, unvisited link, visited link and selected link by clicking on the Auto button and choosing Select color.
5. If required, select a background image by clicking the browse button next to the Image field in the Background category and select a background color by clicking on the Auto button and choosing Select color.
6. Select the Advanced tab to configure additional options by selecting the value column to the right of the property and inputting or selecting the relevant options. Editing empty lines allows adding a new property to the existing properties list.

The relevant changes will be made to the source code and, if applicable, will also be visible in the Design section of the PHP/HTML WYSIWYG Editor.



To configure an HTML object's properties:

1. Place your cursor on the object in either the Design or Source PHP/HTML WYSIWYG Editor panes.
The relevant properties will be displayed in the Properties view.

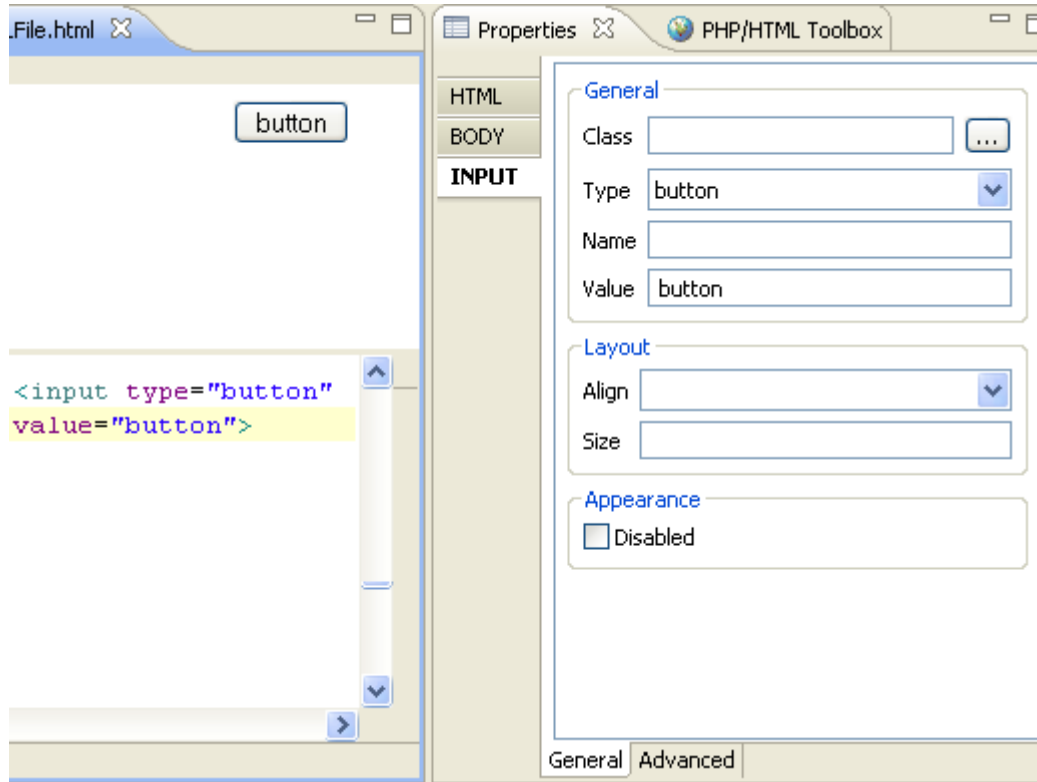


Figure 85 - PHP/HTML WYSIWYG Properties view - Input Tab

2. Each type of object will have different fields and configuration options.
Select the required property options.

The relevant changes will be made to the source code and, if applicable, will also be visible in the Design section of the PHP/HTML WYSIWYG Editor.

Note:

More information about writing HTML can be found at: <http://www.w3schools.com/html>


CSS Editing

CSS Styles can be implemented either within individual HTML files or separate HTML files within the project.

These procedures describe how to create new CSS selectors and how to edit various CSS characteristics.



To create a CSS Selector:

1. To create a new CSS file, go to File | New | Other | Web | CSS.
2. In PHP Explorer view, right-click the HTML or CSS file into which you want to create a CSS creator and select Open With | PHP/HTML WYSIWYG Editor.
3. Place your cursor at the location into which you want to insert the CSS selector.
4. Click the Create CSS Selector button  -or- from the Menu Bar go to Modify | Create CSS Selector.

The Create CSS Selector dialog will open.

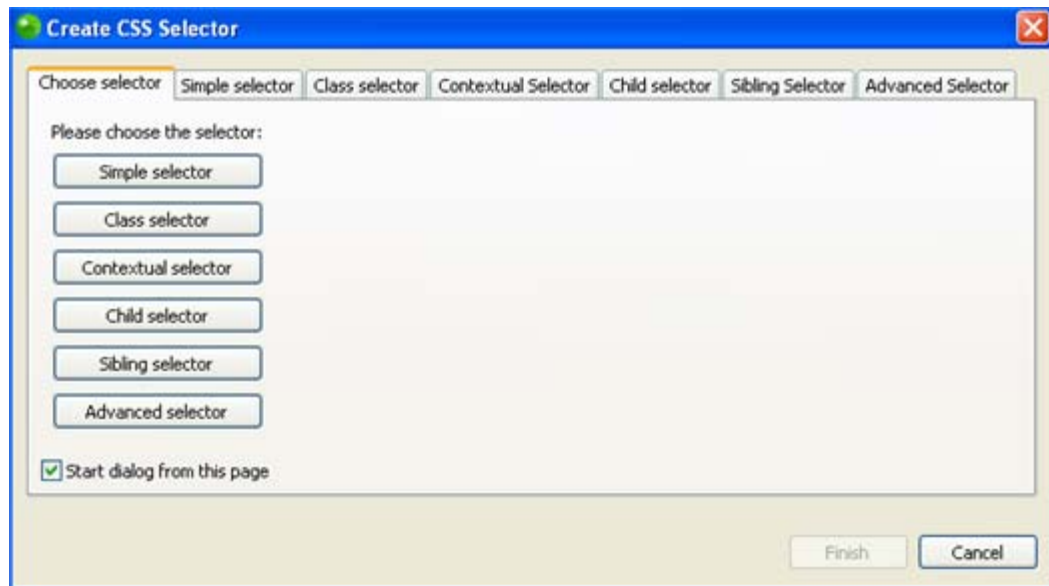


Figure 86 - CSS Selector creation dialog

4. Select the type of selector you would like to create by clicking on the relevant button or tab and choose the required element options:
 - Simple selector - A simple selector is one that applies to a specific HTML element. Double-click the required element (s) to add it.
 - Class selector - A class selector defines a style that applies to elements of a specific class. Double-click the required element(s) and enter a class name.
 - Contextual selector - A contextual selector applies to elements with a specific ancestor. Select the required Ancestor and Second elements and click Add to add the element combination.


- Child selector - A child selector defines a style that applies to elements with a specific parent.
Select the required Ancestor and Second elements and click Add to add the element combination.
 - Sibling selector - A sibling selector defines a style that applies to elements that are preceded by a specific element with the same parent.
Select the required Ancestor and Second elements and click Add to add the element combination.
 - Advanced selector - Select an HTML element and select its relevant Pseudo element and Pseudo class. Click Add.
5. Each tab contains an 'Insert this style into' drop-down list in the bottom-right corner. This enables you to choose whether to enter the CSS selector into a new css document (this will be created), the existing document, or, if applicable, an existing CSS file in the project.

Click Finish to create the CSS selector in the required location.

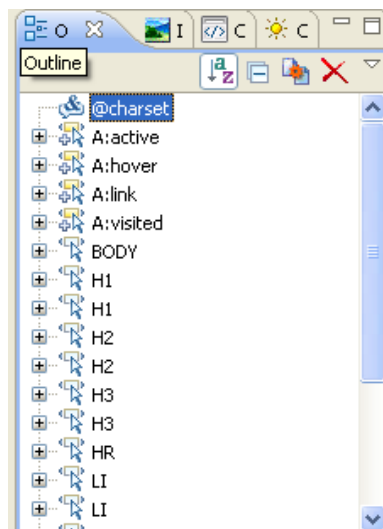


To configure CSS characteristics:

1. In PHP Explorer view, select the HTML or CSS file whose CSS characteristics you would like to edit.
2. Select the CSS Preview view, tabbed with the Outline view, to see what the HTML file will look like with the current CSS settings.
The preview will change according to real-time changes made to the file.

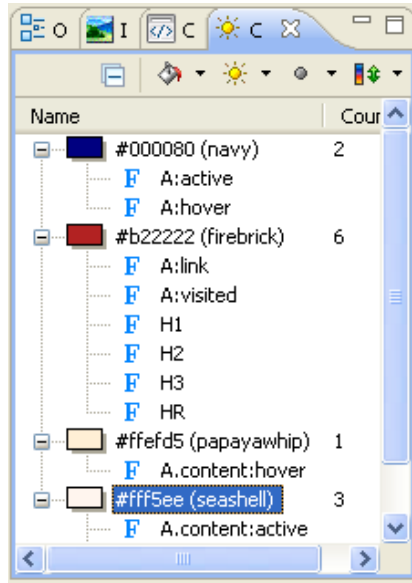
If the opened file was a CSS file, click the Use Template icon  to preview what the CSS settings would look like using a draft template.





3. If the opened file is a CSS file, select the Outline view to see all the styles defined in the current document
Double-click on a selector to be taken to the relevant place in the file.



4. Select the CSS Palette Editor, tabbed with the CSS Preview view, to see and edit all the

colors defined by CSS selectors within the project.



5. Expand the list under each color by clicking the plus "+" next to each color. A list of places where the color is applied will be displayed.
6. To edit the colors, select the required color from the list and select to Replace , Lighten  or Darken  the color by clicking the relevant icon from the view's menu and clicking the arrow next to each icon to choose whether to apply the change to foreground, background, border or unknown properties or colors.
7. You can also choose to convert the selected color to the nearest named color, nearest browser safe color or grayscale by clicking the arrow next to the Convert icon  on the view's toolbar.

Changes will be made in the relevant places within your script to reflect the options you have chosen.

Using the Data Tools Platform

The following tasks describe how to connect to and interact with your database:

- [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 Guide.

Creating a Database Connection Profile

This procedure describes how to create a connection profile to your database, allowing you to connect, view and edit it through the Database Development perspective.



To create a JDBC connection profile:

1. Open the Database Development perspective by going to Window | Open Perspective | Other | Database Development.
2. Click the Create New SQL Connection icon  in the Data Source Explorer view or on the main Toolbar -or- right-click SQL Databases and select New.
3. The New JDBC Connection Profile wizard opens.

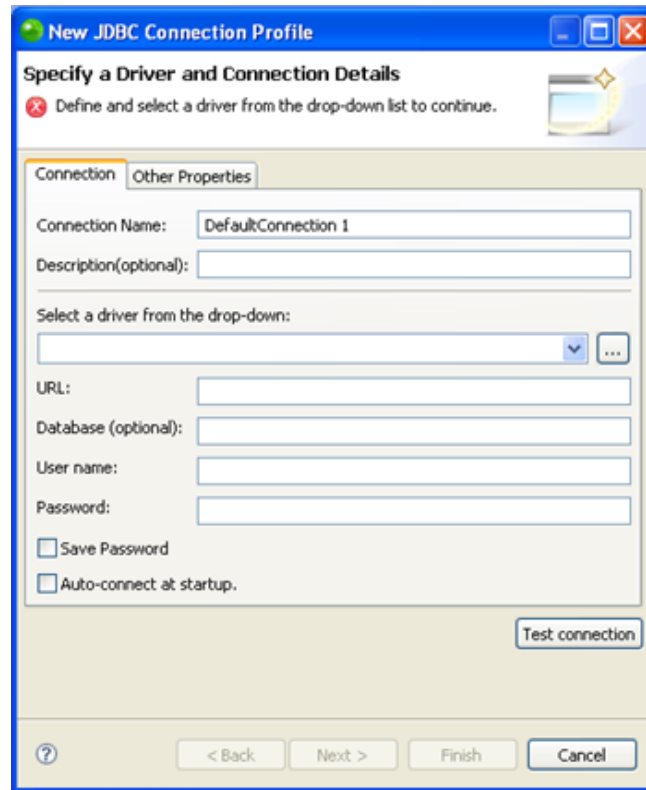



Figure 87 - New Connection Profile dialog

4. Enter the following information:
 - Connection Name - Give your connection a name
 - Description - Enter a description for the connection (optional).
 - Driver - Select a pre-defined driver from the drop-down list. The following drivers are pre-configured in Zend Studio for Eclipse:
 - DB2
 - SB2 i-series
 - Derby
 - MySQL
 - Oracle
 - Postre SQL
 - SQL Server
5. Click the Preferences button  to the right of the drop-down list for further driver options.
 - The URL that corresponds to the selected driver will be automatically entered in the URL field.
Edit the URL according to your database's details.

- Enter the database name, if required.
 - Enter the User Name and Password for your Database connection.
 - Select whether the password should be saved by un/marketing the corresponding checkbox.
 - Select whether the database connection will be re-established when Zend Studio for Eclipse is started by un/marketing the corresponding checkbox.
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.

Note:

To change the properties of your connection profile, right-click it in the Data Source Explorer view and select 'Properties'.

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

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.

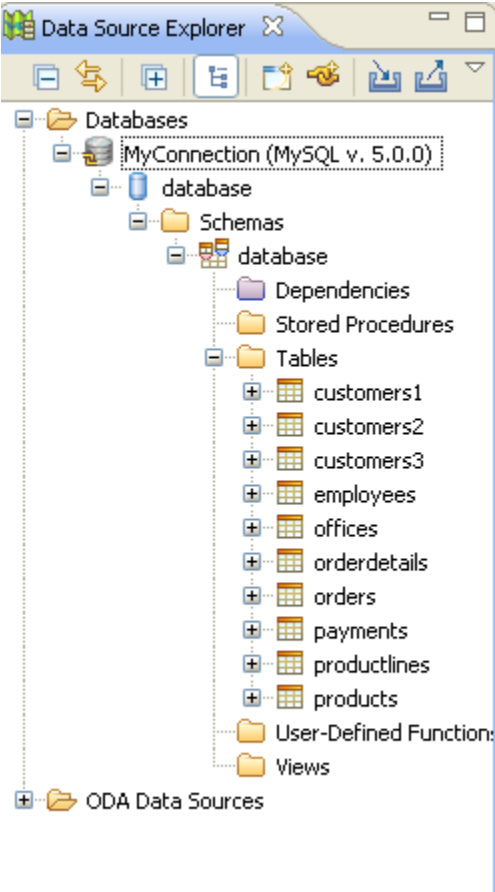


Figure 88 - Data Source Explorer

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.

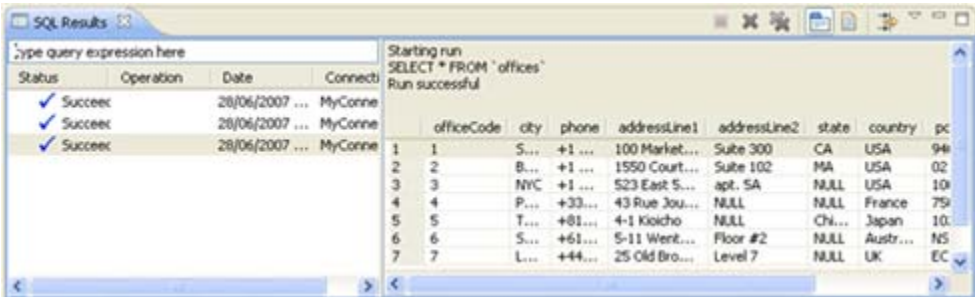


Figure 89 - SQL Results view

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

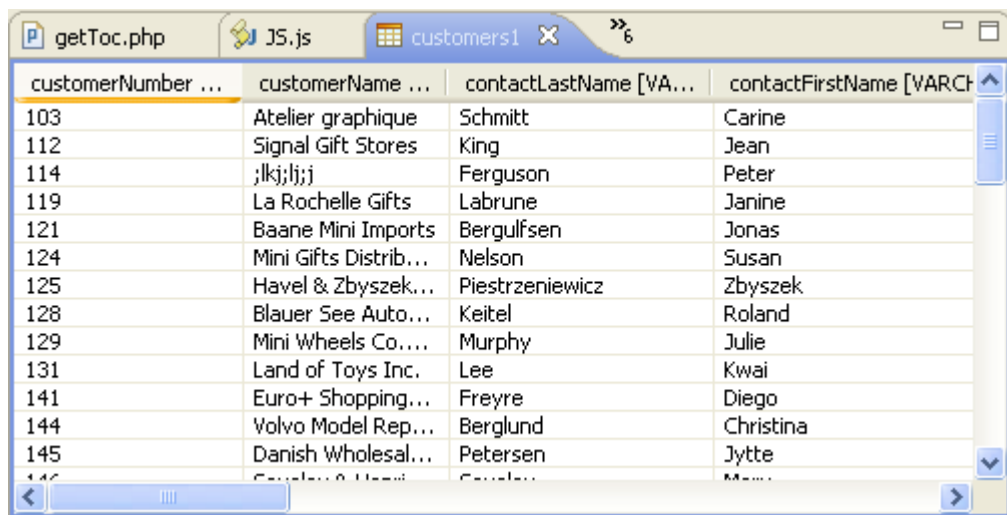
Viewing and Editing Table Content

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 [VARC...
103	Atelier graphique	Schmitt	Carine
112	Signal Gift Stores	King	Jean
114	;lkj;l;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
146	Canadian Music	Carson	Mary

Figure 90 - 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, please see the Data Tools Platform User Guide.

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.
4. From the Connection Profile settings at the top of the scrapbook, select your Database server type, Connection Profile name and Database name (if available) from the drop-down lists.

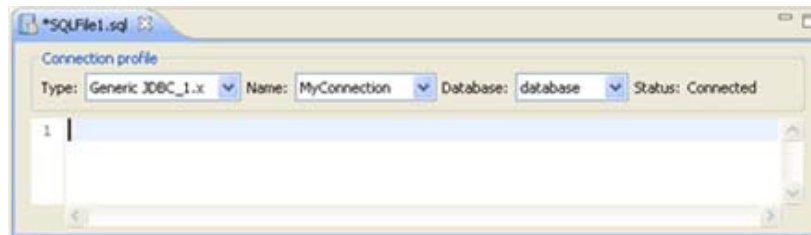


Figure 91 - SQL File editor

5. Write an SQL query to be run on your database (e.g. select * from mytablename;).
6. 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.

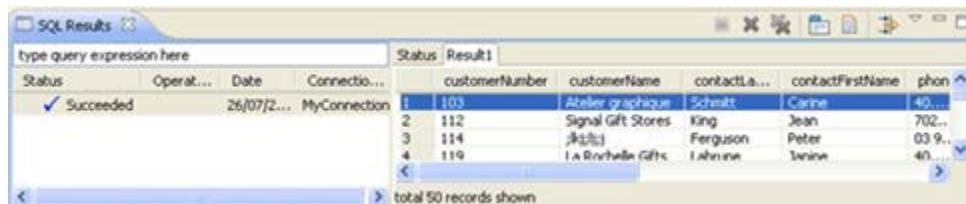


Figure 92 - SQL Results view

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

Integrating with Zend Platform

Zend Studio for Eclipse allows integration with Zend Platform so that Zend Platform events can be viewed in order to Debug, Profile, test and see the source code of problematic URLs through Zend Studio for Eclipse.

For more on Zend Platform Events, see the Zend Platform User Guide.

Before being able to Debug and Profile URLs, ensure that the following conditions are met:

- Zend Platform is installed on your server.
- The Zend Debugger must be loaded on the server before you can debug / profile.
- Zend Platform must be configured to allow your host to initiate debug/profile sessions. Initiating debug/profile sessions is restricted based on IP access lists. You can configure the list of allowed/denied hosts through Zend Platform by going to Configuration | Studio. Check that Zend Studio for Eclipse's IP is in the Allowed Hosts List and is not in the Denied Hosts list.
- If you are using a proxy, ensure that Zend Studio for Eclipse's correct IP is entered in the Zend Platform Category of Platform's Platform | Preferences settings. If it is incorrect, disable the 'Auto Detect Zend Studio Client settings' option and manually enter the correct IP.
- If you have a firewall installed, ensure the correct tunneling settings are configured both in Zend Studio for Eclipse (Window | Preferences | PHP | PHP Servers | Add/Edit | Tunneling), and in Platform (Configuration | Allowed Hosts for Tunneling.)

Once all conditions have been met you can:

- [Define a Zend Platform Server.](#)
- [Access the Platform Event List and utilize Platform / Zend Studio for Eclipse integration functionality.](#)

Defining a Zend Platform Server

When configuring a server in Zend Studio for Eclipse, Zend Platform integration can be configured in order to allow the appliance of Zend Studio for Eclipse functionality (Profiling, Debugging etc.) to Platform events and URLs, as well as allowing access to Zend Platform's event list.

Before configuring a server which can connect to Zend Platform, Zend Platform needs to be installed and be running on a server.



To define a Platform 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 Platform 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 document root's URL.)
4. Click Next.

5. If necessary, define Path Mapping. See [Adding a Server Location Path Map](#) for more information.
6. Click Next.
A Platform Integration dialog will appear.
7. Mark the Enable Platform Integration checkbox to enable Platform Integration features.
8. Select the desired Platform GUI URL suffix.
Leaving the Use default checkbox marked will create a URL in the format <server's document root>/ZendPlatform. Unmark the checkbox to edit the suffix to point to your Zend Platform GUI URL.
9. Enter your Platform User Name and Password.
10. Click Next to configure Tunneling Settings or Finish to create your server.

Your Server will be added to the Server list and will allow you to use Zend Platform integration features. Your Server will also now be available from the arrow next to the Zend Platform icon on the toolbar.

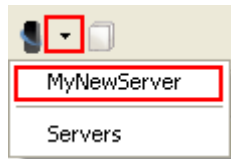


Figure 93 - Zend Platform Server list

Accessing and Using the Platform Event List View

Once Integration with Zend Platform has been configured on your server (see '[Defining a Zend Platform Server](#)'), you can connect to your Zend Platform through Zend Studio for Eclipse to see your Event List and debug, profile, test or view source code of problematic URLs.



To access and use the Zend Platform Event List:

1. Open the Platform Event List view by going to Window | Show View | Platform Events.
The Platform Events view will open.
2. Click the arrow next to the Zend Platform icon in the Platform Events view and select your server.

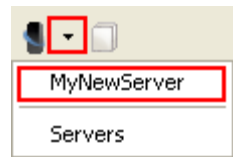


Figure 94 - Zend Platform Server list

A list of Platform events will appear.

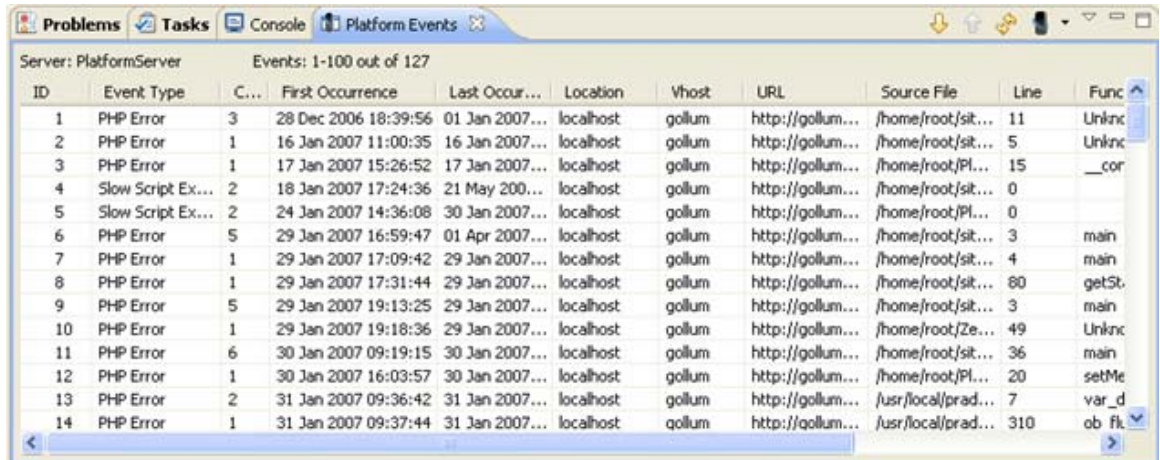


Figure 95 - Platform Events view

3. Events can be debugged and profiled through the Platform Events view or from the Zend Platform GUI.

To perform actions on Platform Events through the Platform Events view:

- i. Right-click an event in the Platform Events view and select one of the following options:

- Debug Event
- Profile Event
- Show Source Code

To perform actions on Platform Events through the Platform GUI:

- i. Double-click on an event to open it in Zend Platform using Zend Studio for Eclipse's internal browser -or- Open Zend Platform in an external browser and go to PHP Intelligence | Event List and double-click an event.

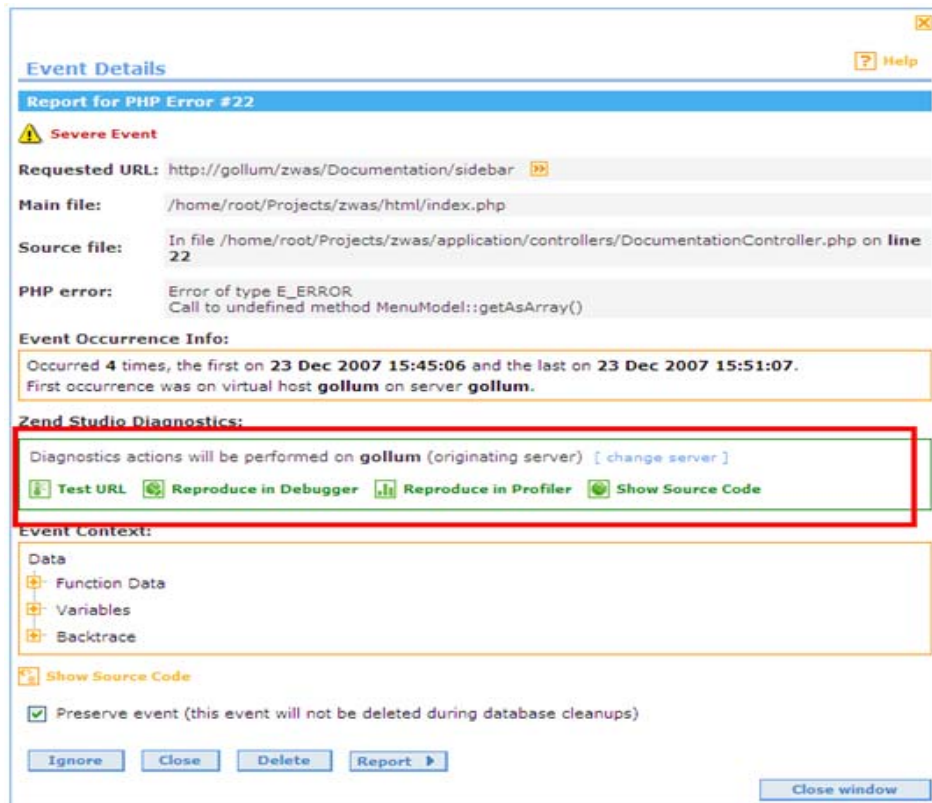


Figure 96 - Zend Platform Event Details - Zend Studio for Eclipse internal browser

- ii. Under the Zend Studio Diagnostics category, you are presented with the following options:
 - Test URL
 - Debug URL
 - Profile URL
 - Show Source Code

Selecting an option will result in the relevant action being carried out in Zend Studio for Eclipse. If you selected to debug or profile a URL, the relevant debug/profile session will be triggered in Zend Studio for Eclipse.

See the [Running and Analyzing Debugger Results](#) topics for more on the information displayed once a debug session has been executed.

See the [PHP Profile Perspective](#) topic for more on the information displayed once a profiling session has been executed.

Using PHPUnit Testing

The following tasks will guide you through the process of creating and running 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](#)

Creating a PHPUnit Test Case

This procedure describes how to create a PHP Unit test case.

Zend Studio for Eclipse 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 project, folder containing the classes you would like to test and select New | PHP Unit 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

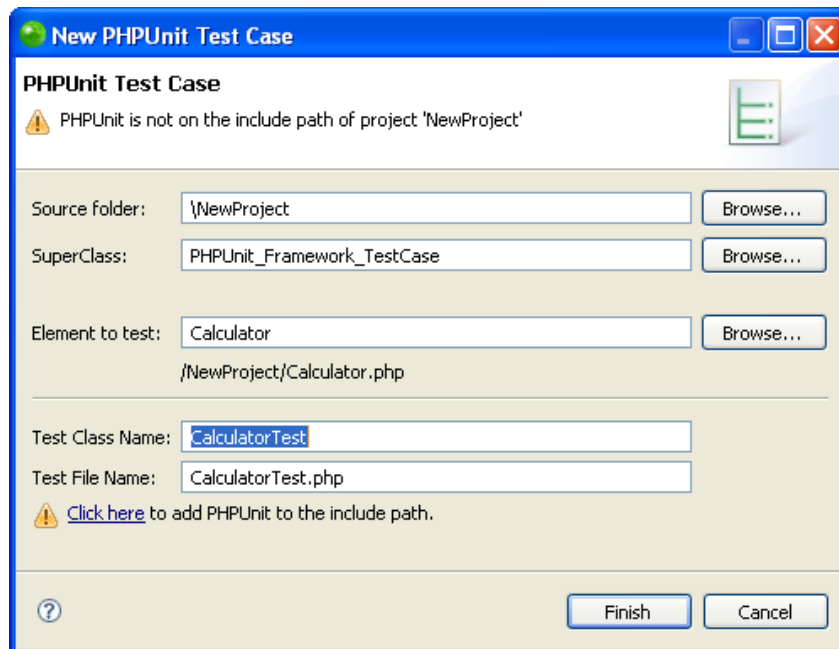


Figure 97 - New PHPUnit Test Case dialog

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 Code 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 for Eclipse 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 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](#)'.



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.

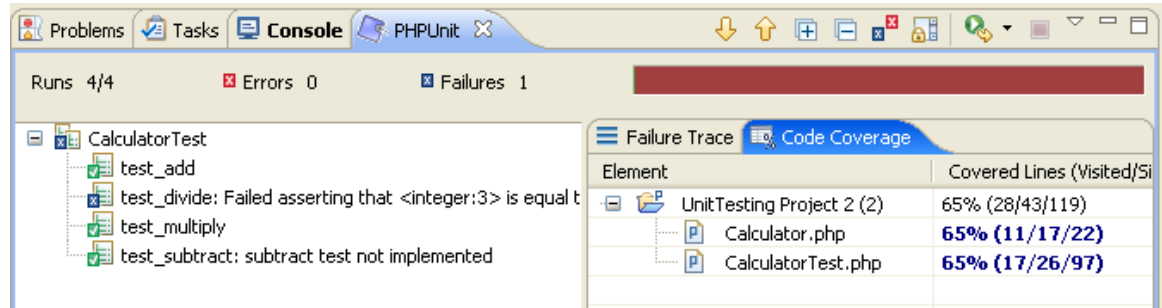








Figure 98 - PHPUnit view

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.

You can create reports based on your results 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 | PHPUnit Test Suite.

The New PHPUnit Test Suite dialog appears.

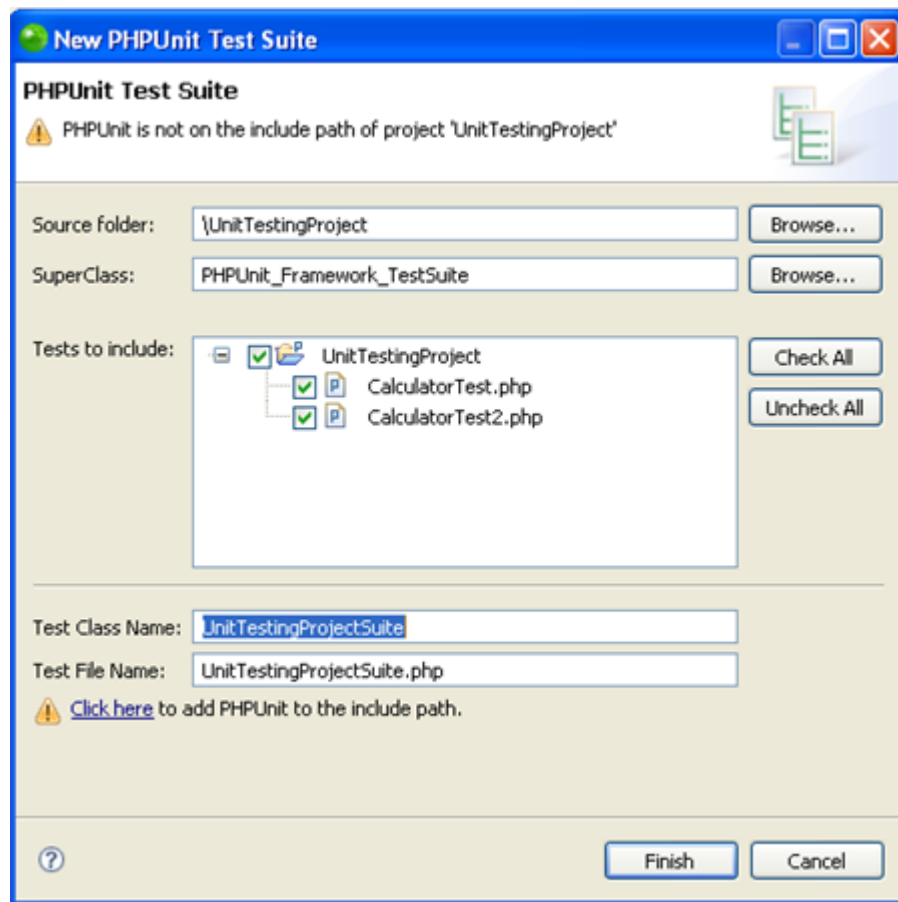


Figure 99 - New PHPUnit Test Suite dialog

2. The 'Tests to include' category will show the available test cases within the project. Mark the checkboxes of the test cases which 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  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.
All the PHPUnit Test Cases contained inside the PHPUnit Test Suite will be run.

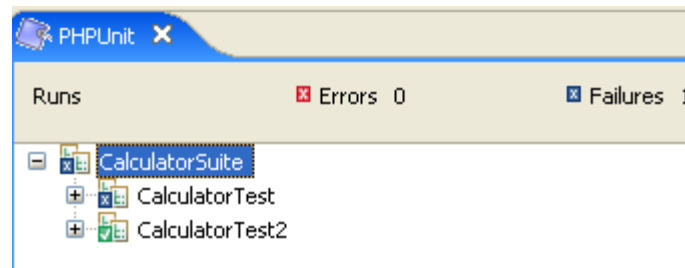





Figure 100 - PHPUnit view


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. 
Tests that have failed will be displayed with a 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  in the PHPUnit view until all tests pass successfully.

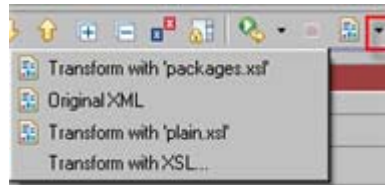
Reporting on PHPUnit Test Results

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

Designed for use with [PHPUnit](#) and [Zend Studio](#).

Summary

Tests	Failures	Errors	Success rate	Time
4	1	0	75.00%	0.066

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Overview

Name	Tests	Failures	Errors	Time(s)
CalculatorTest	4	1	0	0.066

TestCase CalculatorTest

Name	Status	Details	Time(s)
testAdd	Success		0.012
testDivide	Failure	PHPUnit_Framework_ExpectationFailedException: Failed asserting that <integer:3> is equal to <double:0.5>. Trace: PHPUnit_Framework_Assert::assertEquals() C:\Documents and Settings\shachar\workspace\Calculator 2\CalculatorTest.php:69	0.031
testMultiply	Error	PHPUnit_Framework_IncompleteTestError: multiply test not implemented Trace: PHPUnit_Framework_Assert::markTestIncomplete() C:\Documents and Settings\shachar\workspace\Calculator 2\CalculatorTest.php:79	0.013
testSubtract	Success		0.011

[Back to top](#)

Report generated at 2007-11-13T10:58:14+02:00

Figure 101 - Unit Test Results Report

Note:

Reports will be generated in the location defined in the PHPUnit Preferences page.

- Clicking the link beneath a failed test result will take you to the relevant test.

Using the Profiler

The following Profile functionality is available in Zend Studio for Eclipse:

- [Profiling a PHP Script](#) - Profile PHP Scripts on your Workspace
- [Locally Profiling a PHP Script](#) - Profile a PHP Script using Zend Studio for Eclipse's internal debugger
- [Remotely Profiling a PHP Script](#) - Profile a PHP Script using your remote server's 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.

Profiling a PHP Script

Files located in your workspace can be profiled in two ways:

[Locally Profiling a PHP Script](#) - Using Zend Studio for Eclipse's internal debugger


[Remotely Profiling a PHP Script](#) - Using your server's Zend Debugger.

Locally Profiling a PHP Script

This procedure describes how to Profile a PHP Script from your workspace using Zend Studio for Eclipse's internal debugger:



To locally Profile a PHP Script:

1. Click the arrow next to the Profile button  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.

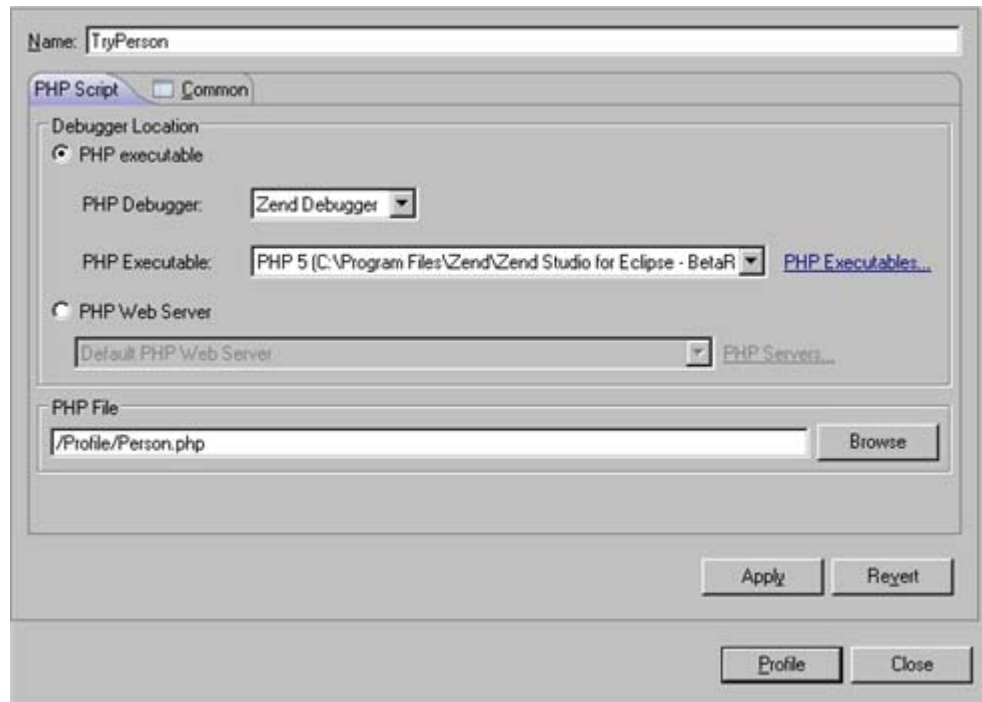


Figure 102 - Profile Configuration Dialog

3. Double-click the PHP Script option to create a new Profile configuration.
4. Enter a name for the new configuration.
5. Ensure that the PHP Executable setting is selected under the Debugger Location category.
6. Select the required PHP executable.
7. Under PHP File, click Browse and select the required file.
8. Click Apply and then Profile.
9. 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 the PHP Profile Perspective topic 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.

Remotely Profiling a PHP Script

This procedure describes how to profile a PHP Script from your workspace using the Zend Debugger installed on your remote server.


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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.



To remotely profile a PHP script:

1. Click the arrow next to the Profile button  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.

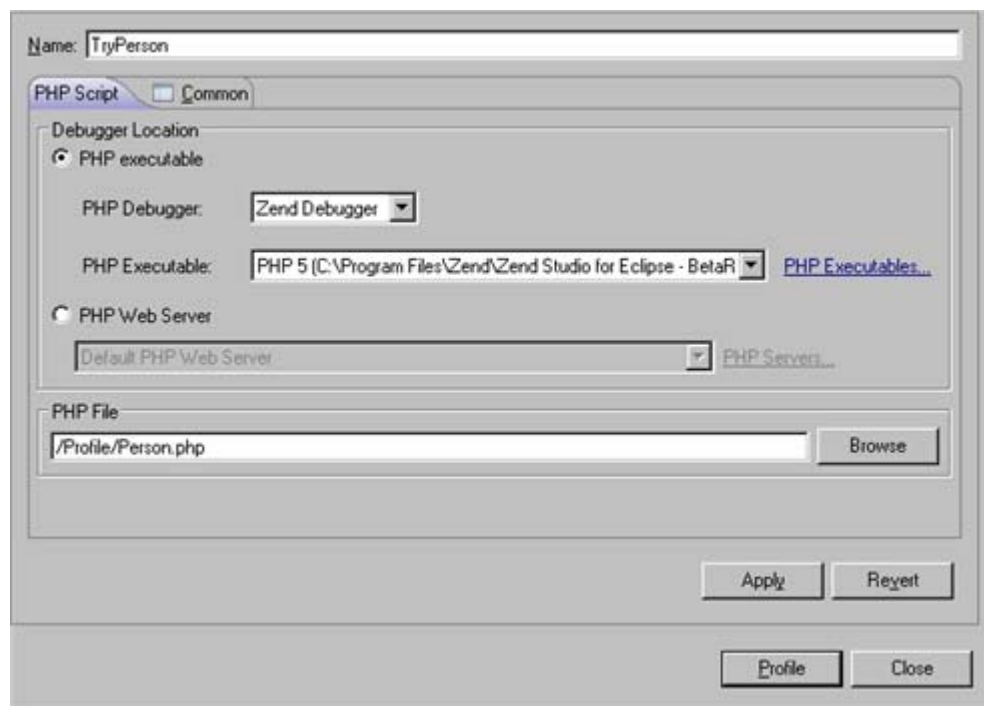


Figure 103 - Profile Configuration Dialog

3. Double-click the PHP Script option to create a new Profile configuration.
4. Enter a name for the new configuration.
5. Select the PHP Web Server option and select your server from the drop-down list. (If you have not yet configured a server, click the PHP Servers link and follow the instructions under [Adding PHP Servers.](#))
6. Under PHP File, click Browse and select the required file.

7. Click Apply and then Profile.
8. 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 the [PHP Profile Perspective](#) for more on the information that will be 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.

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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.



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.

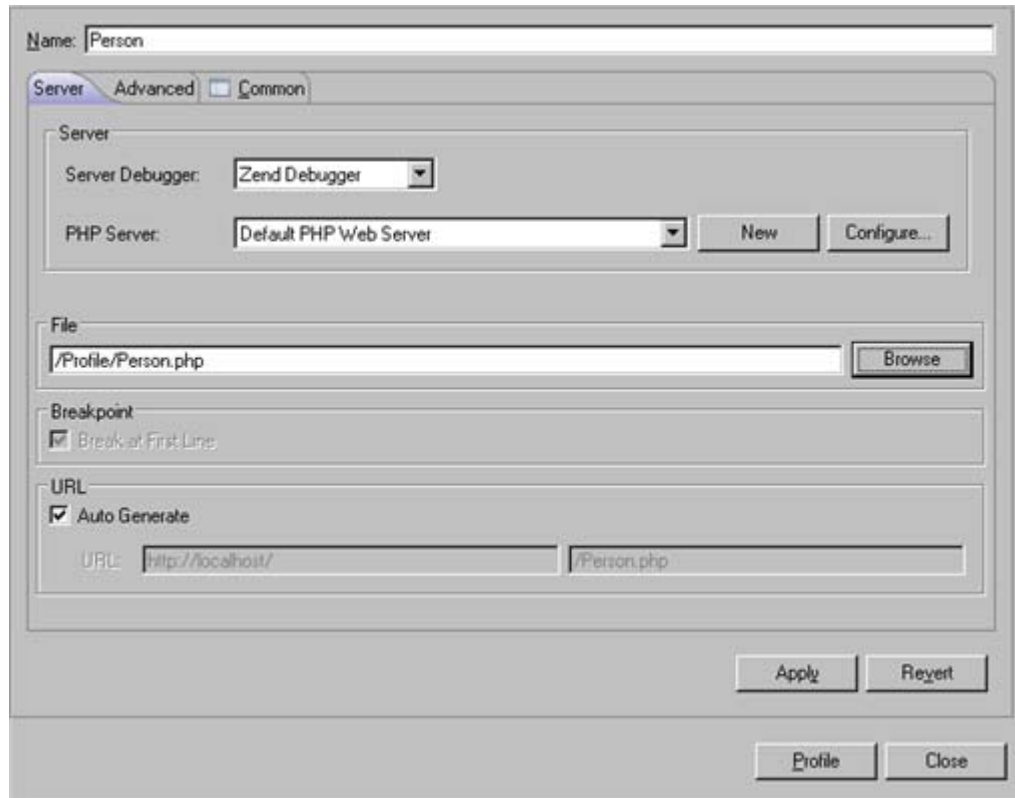


Figure 104 - Profile PHP Web Page Configuration

4. Enter a name for the new configuration.
5. Select your server from the list.
If you have not yet configured your server, click the underlined 'PHP Servers' shortcut. The Servers preferences page will open. Configure your server by following the instructions on ['adding a new server'](#) under the PHP Servers Preferences page.
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 for Eclipse'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 [Path Mapping](#) 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 [PHP Profile Perspective](#) for more on the information that will be displayed once a profile session has been run.

Profiling a URL

This procedure describes how to debug a URL.


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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.



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.

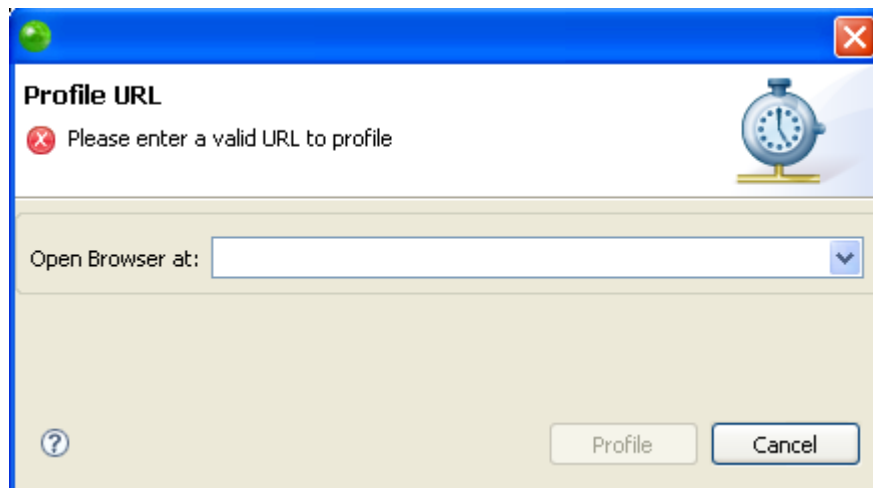


Figure 105 - Profile URL dialog

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 Debugger Toolbar

This procedure describes how to Profile using the Zend Debugger Toolbar.



To Profile using the Zend Debugger Toolbar:

1. Ensure the Zend Debugger toolbar is installed on your system.
If you have not yet installed the Zend Debugger Toolbar, see [Installing and Configuring the Zend Debugger Toolbar](#) for more information.
2. Open your browser and browse to the page you would like to profile.
3. Click the  button on the toolbar to profile the page currently displayed in the Explorer.

The relevant Profile session will be launched in Zend Studio for Eclipse.

If Zend Studio for Eclipse is not open, you will be prompted to open it before the profiling session is launched.

Using the Debugger

The following Debug functionality is available in Zend Studio for Eclipse:

- [PHP Script Debugging](#)
- [Local PHP Script Debugging](#)
- [Remote PHP Script Debugging](#)
- [PHP Web Page Debugging](#)
- [URL Debugging](#)
- [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.

Debugging a PHP Script

Files located in your workspace can be debugged in two ways:

[Local Debugging](#) - Using Zend Studio for Eclipse's internal debugger


[Remote Debugging](#) - Using your remote server's debugger

Locally Debugging a PHP Script

This procedure describes how to debug a PHP Script from your workspace using an internal debugger.



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  on the toolbar and select Open Debug Dialog -or- select Run | Open Debug Dialog.
A Debug dialog will open.
4. Double-click the PHP Script option to create a new debug configuration.

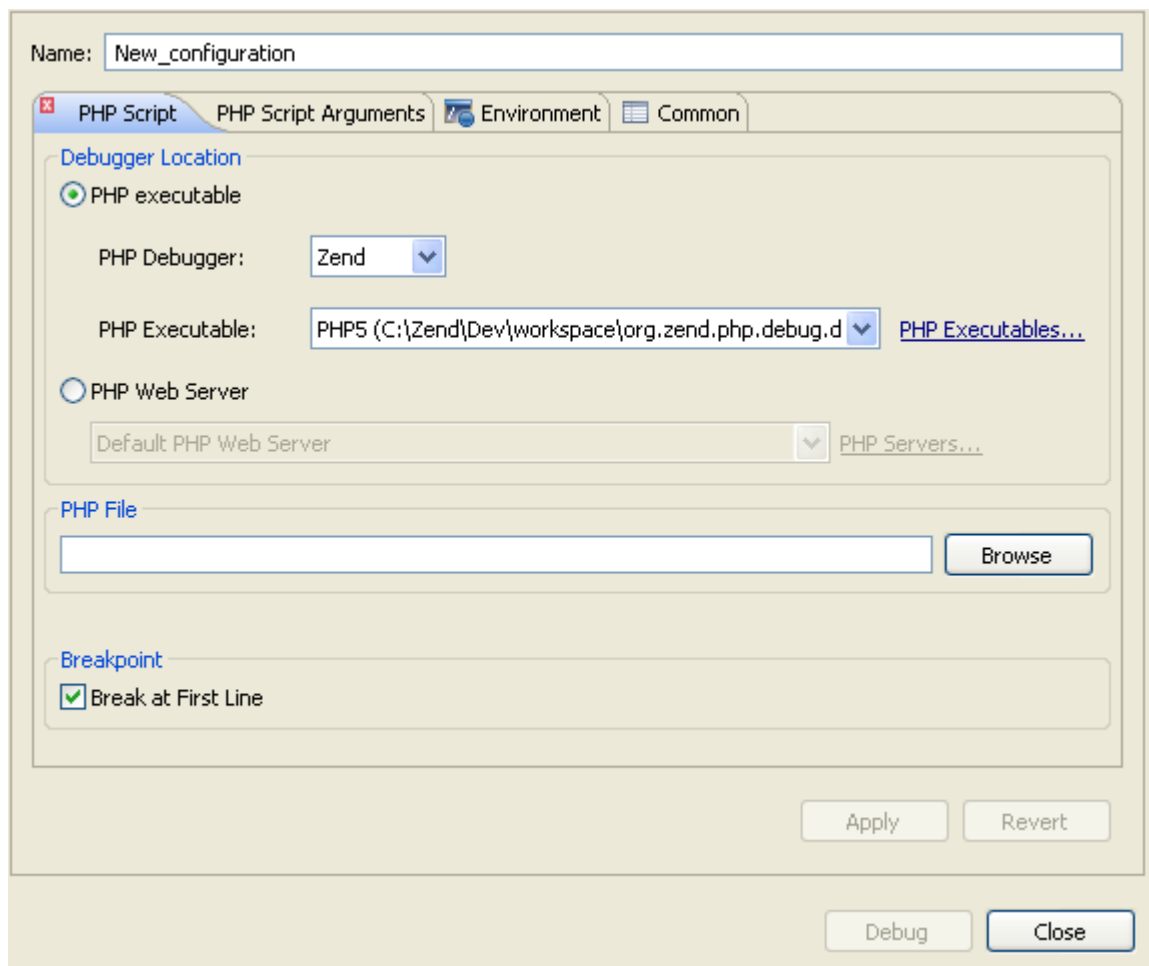


Figure 106 - New Debug Configuration

5. Enter a name for the new configuration.
6. Ensure that the PHP Executable option is selected under the Debugger Location category and select

the required PHP executable.

7. Select the PHP Debugger to be used.
8. Under PHP File, click Browse and select your file
9. Marking the Breakpoint checkbox will result in the debugging process pausing at the first line of code.
10. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.
11. Click Apply and then Debug.
12. 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.

Remotely Debugging a PHP script

This procedure describes how to debug files in your workspace remotely using your server's Zend debugger. Use this function if you want to test the execution of the file in 'real time' on the production server. This is especially relevant if your server has loaded extensions.


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 Core and Zend Platform, but can also be downloaded as a separate component from <http://www.zend.com/en/products/studio/downloads>.



To remotely debug a PHP Script:

1. Set breakpoints at the relevant places in the file that you would like to debug.
2. Save the file.
3. Click the arrow next to the debug button  on the toolbar and select Open Debug Dialog -or- select Run | Open Debug Dialog.
A Debug dialog will open.
4. Double-click the PHP Script option to create a new debug configuration.

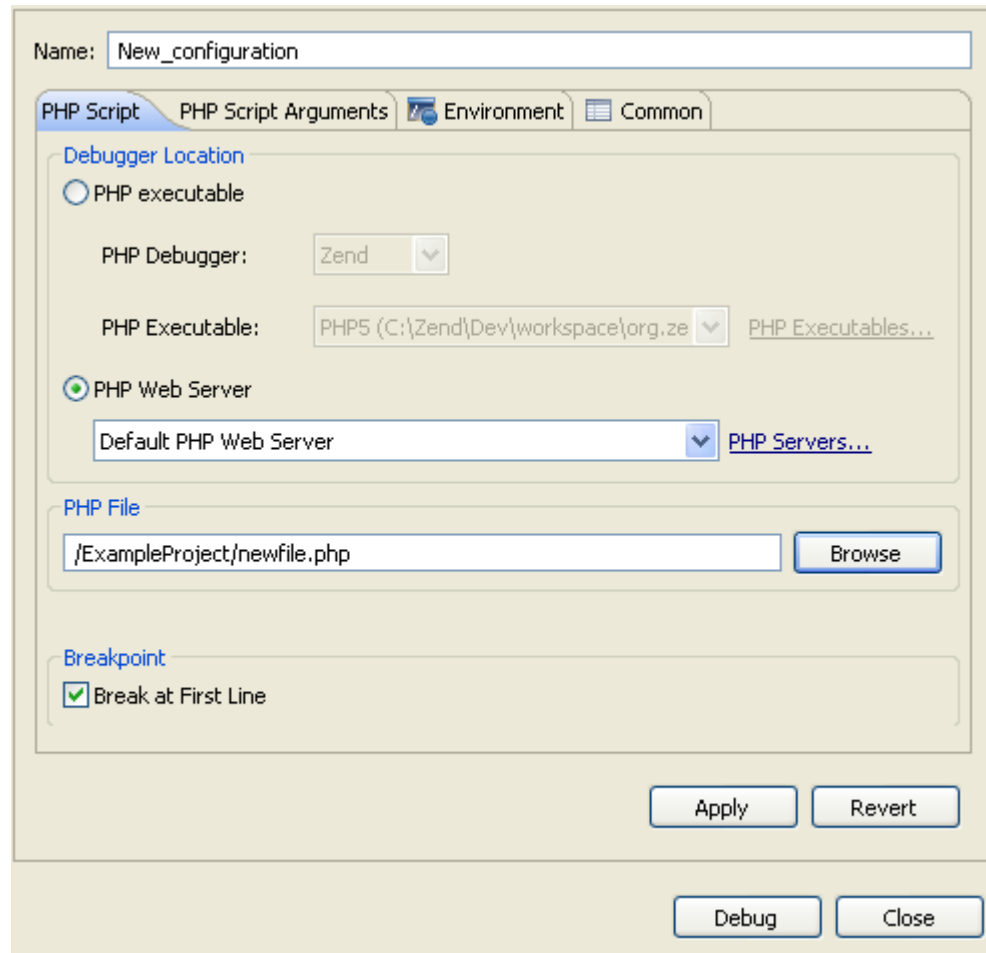


Figure 107 - New Debug Configuration

5. Enter a name for the new configuration.
6. Select the PHP Web Server option under the Debugger Location category.
7. Select your server from the list.
If you have not yet configured your server, click the underlined 'PHP Servers' shortcut. The Servers preferences page will open. Configure your server by following the instructions on ['adding a new server'](#) under the PHP Servers Preferences page.
8. Under the PHP File category, click Browse and select your file.
9. Marking the Breakpoint checkbox will result in the debugging process pausing at the first line of code.
10. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.
11. Click Apply and then Debug.
12. 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 remote debugging session is unsuccessful, check that your server's root directory contains a Dummy File. This file should match the name of the Dummy File as defined in the Advanced Options section of the [PHP Debug preferences page](#) (accessible from Window | Preferences | PHP | Debug). By default, the file will be called "dummy.php".

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 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 in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Core and Zend Platform, 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. Set breakpoints at the relevant places in the file that you would like to debug and save the file.
2. Save the file.
3. Click the arrow next to the debug button  on the toolbar and select Open Debug Dialog - or- select Run | Open Debug Dialog.
A Debug dialog will open.
4. Double-click the PHP Web Page option to create a new debug configuration.

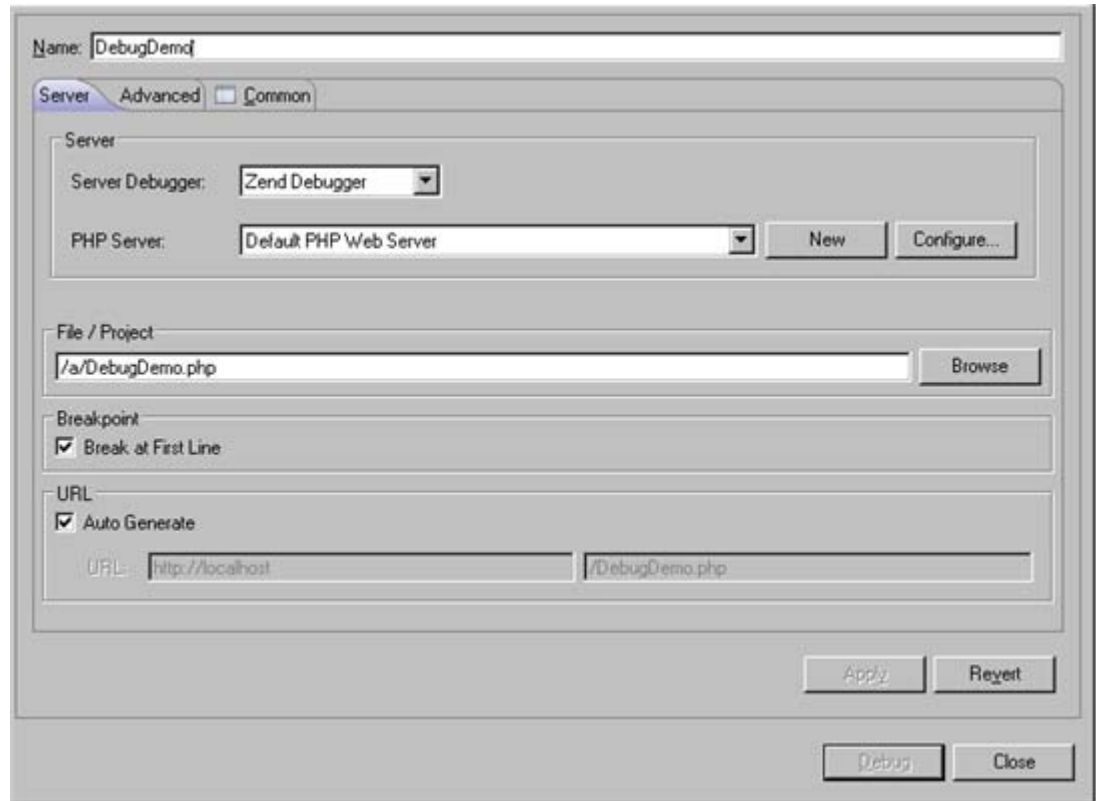


Figure 108 - New Debug Configuration

5. Enter a name for the new configuration.
6. Select the Server Debugger to be used (by default this will be the Zend Debugger).
7. Select your server from the list.
If you have not yet configured your server, click the underlined 'PHP Servers' shortcut. The PHP Servers preferences page will open. Configure your server by following the instructions on ['adding a new server'](#) under the [PHP Servers Preferences](#) page topic.
8. Under PHP File, click Browse and select your 'debug target' file (the file from which the debugging process will start.)
9. Select whether the Debugger should stop at the first line of code by marking/unmarking the 'Break at First Line' checkbox.
10. 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.
11. For further Debug options, select the Advanced tab, which has the following options:

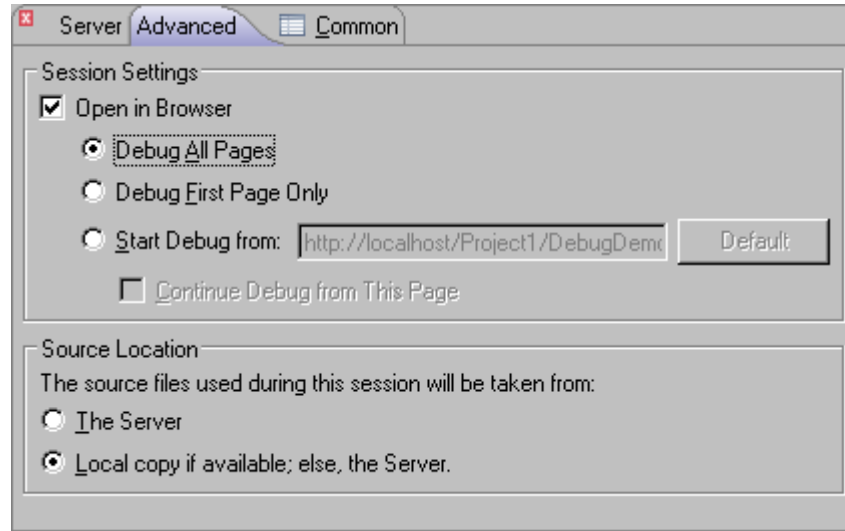


Figure 109 - New Debug Configuration - Advanced

- Open in Browser - Mark if you would like the application to be displayed in Zend Studio for Eclipse's internal browser while it is run.
- 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.
- 12. Click Apply and then Debug.
- 13. 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 in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Core and Zend Platform, 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  on the main toolbar -or- go to Run | Debug URL.
2. The Debug URL dialog will appear.

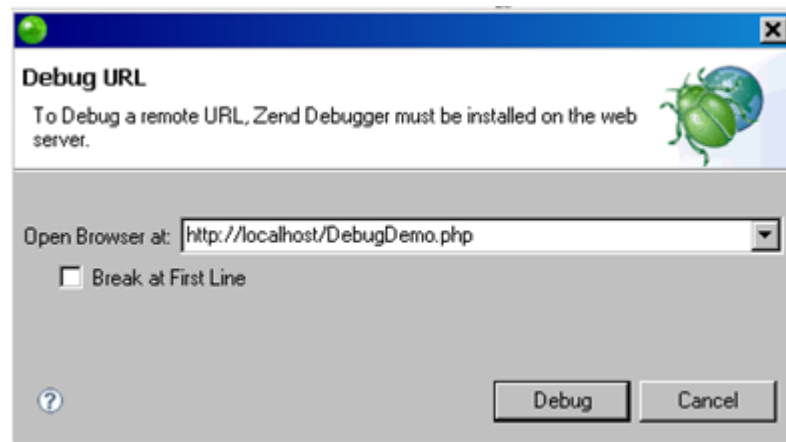


Figure 110 - Debug URL dialog

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.

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.

During the debugging process, 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 - Will display any error and warning messages.
- Tasks - If you had added any tasks to your script, these would be displayed here.

Note:

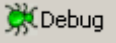
See '[PHP Debug Perspective](#)' for more on the views that will be displayed during Debugging.

Debugging Using the Zend Debugger Toolbar

This procedure describes how to Debug using the Zend Debugger Toolbar.



To Debug using the Zend Debugger Toolbar:

1. Ensure the Zend Debugger toolbar is installed on your system.
If you have not yet installed the Zend Debugger Toolbar, see [Installing and Configuring the Zend Debugger Toolbar](#) for more information.
2. Open your browser and browse to the page from which you would like to start debugging.
3. Click the  button on the toolbar to debug the page currently displayed in the Explorer.
-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 or form is posted. This will automatically start executing the next page consulted in debug mode.
 - All forms (POST) on this site - The debugging session will be launched every time a link is clicked or a form is posted. 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.

Note:

If the file you would like to debug exists in your workspace, you can choose to debug the workspace copy of your file by going to Extra Stuff | Settings on the Toolbar and selecting the 'Debug Local Copy' option. 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 the [Path Mapping](#) topic for more information.

The relevant Debug session will be launched in Zend Studio for Eclipse.

If Zend Studio for Eclipse is not open, you will be prompted to open it before the debugging process is launched.

Installing and Configuring the Zend Debugger Toolbar

Installing the Zend Debugger Toolbar

The Zend Debugger Toolbar can be installed together with Zend Studio for Eclipse.

If you did not install the Zend Debugger Toolbar during Zend Studio for Eclipse installation, you can download and install it manually.

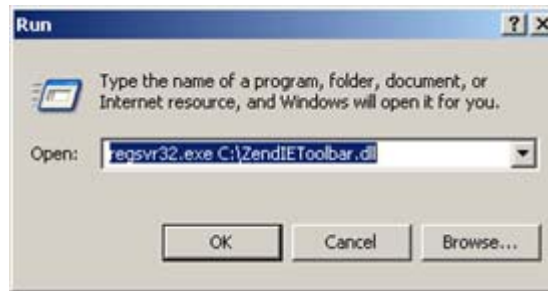
This procedure describes how to download and install the Zend Debugger Toolbar.



To manually install the Zend Debugger Toolbar:

For Internet Explorer

1. Download the Windows Zend Studio Browser Plugin from <http://www.zend.com/en/products/studio/downloads>.
2. Save the file.
3. Run the command "regsvr32.exe [full path to]\ZendIEToolbar.dll" by going to Start | Run in the Windows Start menu.
4. Restart Internet Explorer.
5. If the Toolbar is not automatically displayed, go to View | Toolbars and select Zend Studio.



For Firefox

1. Download the cross-platform Firefox plug-in from <http://www.zend.com/en/products/studio/downloads>.
2. Run the installation.
3. Restart Firefox.
4. If the Toolbar is not automatically displayed, go to View | Toolbars and select Zend Studio.

Configuring the Zend Debugger Toolbar

Before you can debug/profile through the toolbar, you must first configure the Zend Debugger settings to match those configured in your Zend Studio for Eclipse.

This procedure describes how to configure the Zend Debugger Toolbar to be able to communicate with Zend Studio for Eclipse.



To configure the Zend Debugger Toolbar:

1. From the Toolbar, go to Extra Stuff | Settings.
The Zend Toolbar Settings will be displayed.
2. In the Debug Session Settings category, mark the Detect Zend Debugger checkbox to disable the Debug and Profile buttons when a server is browsed to which does not have a Zend Debugger.

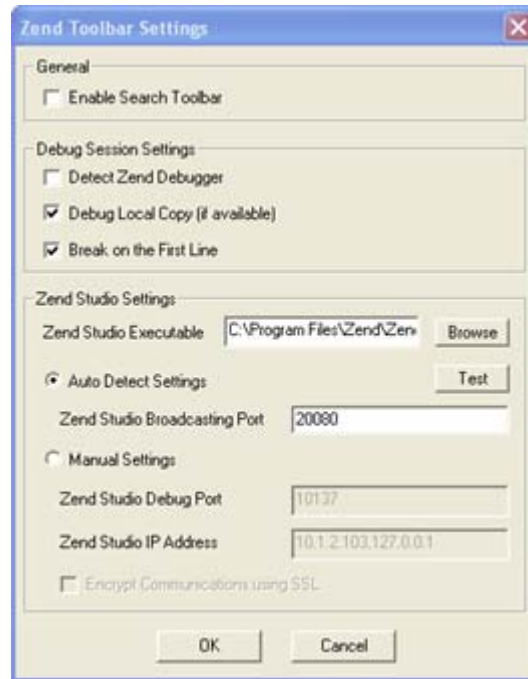


Figure 111 - Zend Toolbar Settings

3. In the Zend Studio Settings category, configure the following:
 - Zend Studio Executable - Click Browse and select the location of your Zend Studio for Eclipse .exe file.
 - The Debug Port and IP settings in your Zend Toolbar must match the settings defined in your Debug preferences in Zend Studio for Eclipse.

To automatically synchronize these settings:

 - i. Select 'Auto Detect Settings'.
 - ii. Enter the Zend Studio Broadcasting Port. This must match the Broadcasting Port defined for your Zend Debugger in your [Installed Debuggers Preferences](#) in Zend Studio for Eclipse. (In Zend Studio for Eclipse, this is accessible from Window | Preferences | PHP | Debug | Installed Debuggers | Configure).
The default port is 20080.
 - iii. Click Test.
The Debug Port and Zend Studio IP settings defined in Zend Studio for Eclipse will be automatically updated in your Zend Debugger Toolbar.

To manually enter your settings:

 - i. Select 'Manual Settings'.
 - ii. Enter the Zend Studio Debug Port and Zend Studio IP Address defined for your for your Zend Debugger in your [Installed Debuggers Preferences](#) in Zend Studio for Eclipse. (In Zend Studio for Eclipse, this is accessible from Window | Preferences | PHP | Debug | Installed Debuggers | Configure).
4. Click OK to save your settings.

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 remote PHP Script debugging/profiling and Web Page debugging/profiling when the 'use local copy' option is selected.



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. An Add new Path Mapping dialog appears.
6. Enter the Server Path from which you would like to create the Path Map.
7. Select either the 'Path in Workspace' or 'Path in File System' option and click Browse to specify the location.

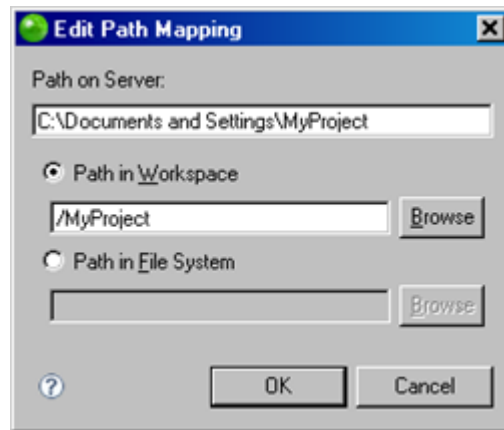
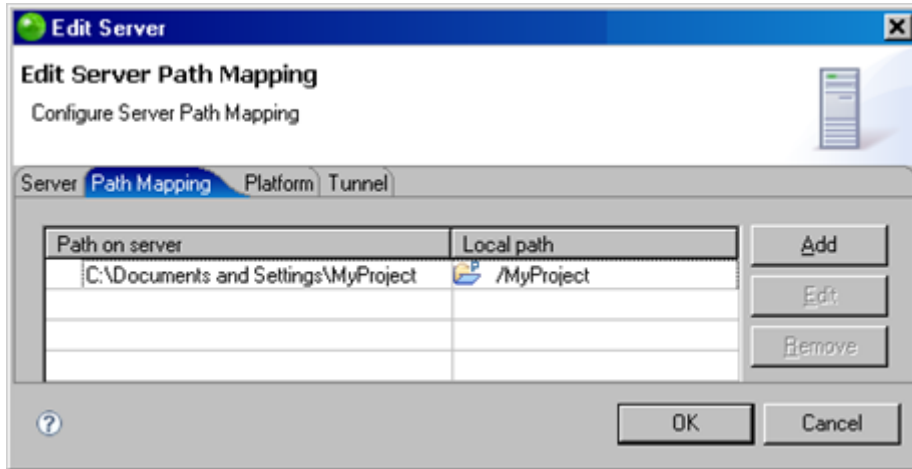


Figure 112 - Edit Path Mapping

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.

Note:

Path Mapping can also be set automatically during Debugging / Profiling. See the [Path Mapping](#) topic for more details.

Adding Elements to a Project's Include Path

This procedure describes how to add a project/variable/folder to a project's include path.



To add a project/library to your project's include path:

1. In PHP Explorer view, right-click the project's Include Paths.
2. Select 'Configure Include Paths'.

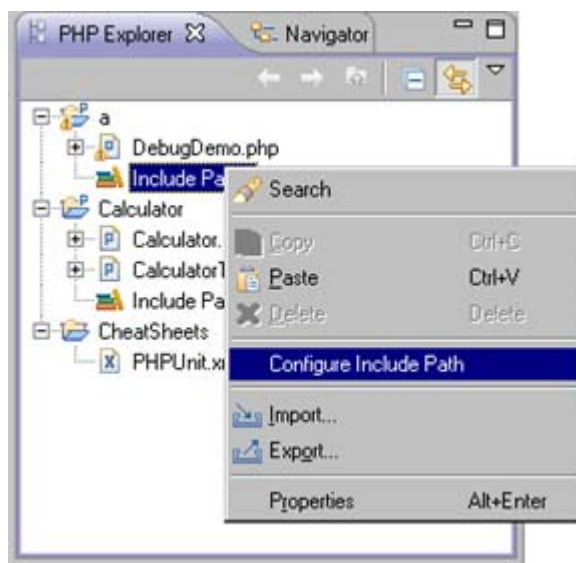


Figure 113 - Configure Include Path

3. The project's PHP Include Path properties page will appear.

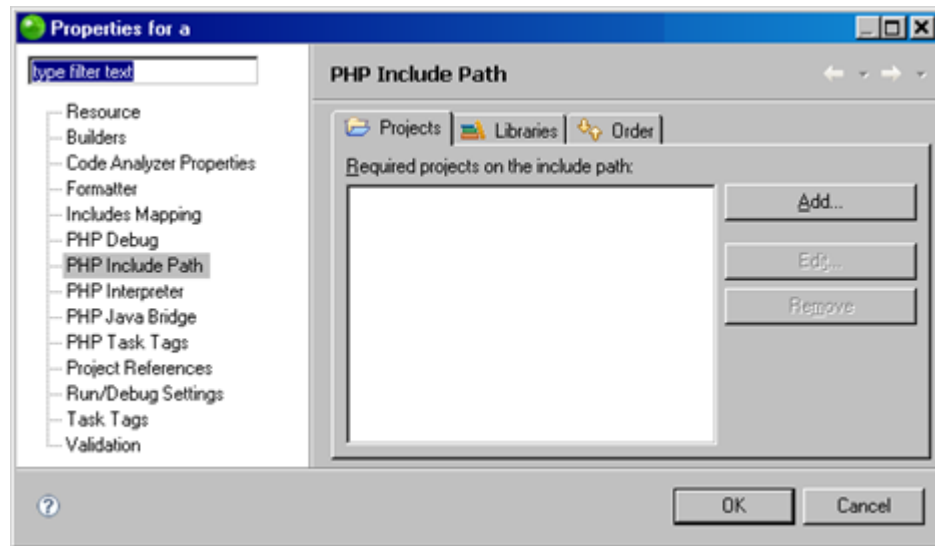


Figure 114 - Include Path Properties

4. To add a project 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.
5. To add Path Variables to your Include Path:
 - i. Select the Libraries tab.
 - ii. Click Add Variable.
The New Variable Entry dialog appears.
 - iii. Select the required Path Variable.

Note:

The Variable points to a folder. To choose an archive inside the folder, click Extend.
To add a Variable to the Path Variables list, click Configure Variables to be taken to the [Path Variables Preferences page](#).

- iv. Click OK.
The Variable will be added to your project's Include Path.

Note:

The variable is a read-only file and will not be available for editing.

6. 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 will be a read-only file and will not be available for editing.

7. Select the Order tab.
8. Ensure the included elements appear in the same order as they do in your php.ini.
9. If you want to rearrange the order, select an element and click Up or Down to move it in the list.
10. 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.

Setting Up Tunneling

To establish a tunneling connection, the following tasks need to be performed:

1. [Setup a server allowing tunneling in Zend Studio for Eclipse](#). This can be done from the PHP Servers Preferences page (Window | Preferences | PHP | PHP Servers). If you require Zend Platform integration, this will also be defined here.
2. [Ensure your Zend Studio for Eclipse is an allowed host](#) for your server debugger. This can be done through Zend Core or Zend Platform (if installed), or through your php.ini file.
3. If you established Zend Platform integration, set the Zend Platform Settings to [Auto detect the Zend Studio Client settings](#) (in Zend Platform's Platform | Preferences tab).
4. [Activate the tunnel](#) by selecting your Tunneling server from the list next to the Tunneling icon on the

toolbar .

Setting Up a Tunneling Server

This procedure describes how to configure a server to allow Tunneling.

Note:

You can configure several servers which allow tunneling.



To configure your Server for Tunneling with Zend Studio for Eclipse:

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 Platform integration:

- Check the "**Enable Platform Integration**" option.
- In the Platform GUI section, ensure the **Use Default option** checkbox is marked and the **"/ZendPlatform"** suffix is entered. This information is used to locate Platform
- In the Authentication section enter your Platform **User Name** and **Password**.

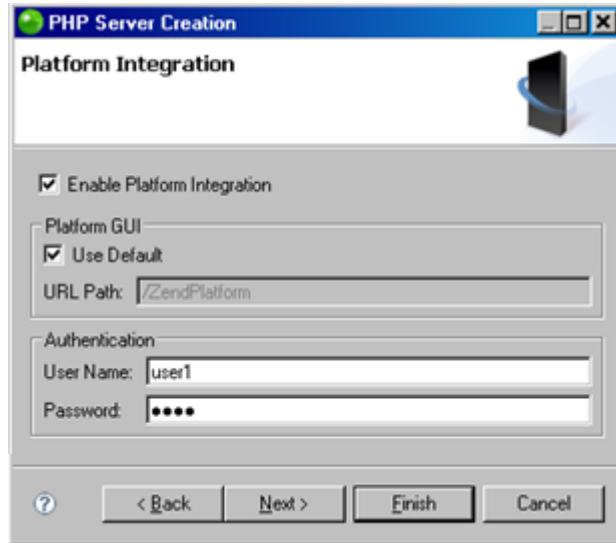


Figure 115 - Platform Integration Preferences

6. Click **Next** to continue or go to the next Tab.

7. In the Tunneling Settings section, check the "**Enable Tunneling**" option.

8. Specify your "**Return Host**". This should be the IP address of the machine on which Zend Studio for Eclipse is installed.

9. If your Web server requires HTTP authentication, enter your User Name and Password in the **Authentication** category.

Zend Studio for Eclipse sends the authentication information in the header.

Note:

This assumes the user account is set up on the Web server

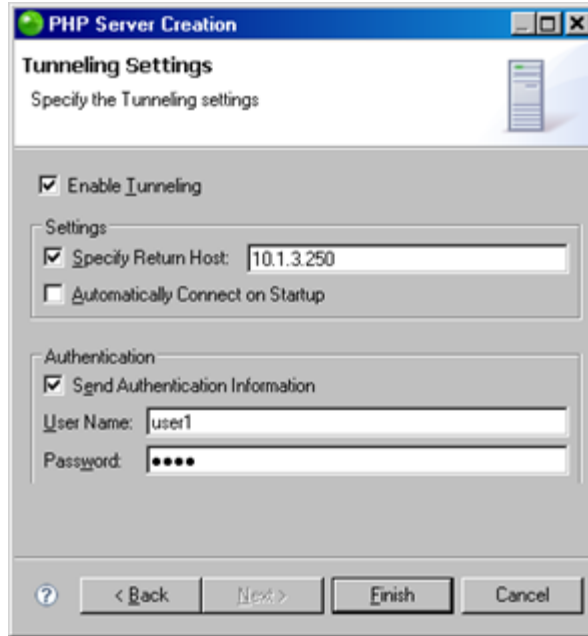


Figure 116 - Tunneling Settings

10. Click Finish or OK.

Your server is now configured to allow tunneling in Zend Studio for Eclipse.

The next step in creating a tunneling connection is to [Set your Zend Studio for Eclipse to be an Allowed Host](#).

Setting your Zend Studio for Eclipse to be an Allowed Host

This procedure describes how to ensure the machine on which your Zend Studio for Eclipse is installed will be an Allowed Host for initiating your Debug session on the remote server.

The steps you need to follow will depend on whether you have [Zend Platform](#), [Zend Core](#) or the [standalone Zend Debugger](#) installed on your server:

Note:

If you have both Zend Platform and Zend Core installed on your remote server, you only need to configure Zend Studio for Eclipse as an Allowed Host in one of them.

If Zend Platform is installed on your server:



1. Open your Zend Platform GUI. This can be done from within Zend Studio for Eclipse by selecting the server on which you have configured your Zend Platform integration from the drop-down list

next to the Zend Platform icon on the toolbar .

2. Go to the Configuration | Studio tab.

Allowed Hosts for Tunneling

The following hosts will be allowed to use the Zend Studio Tunnel, for debugging across a firewall

Server Host	
10.1.1.222/32	Edit Remove
10.1.3.186/32	Edit Remove
127.0.0.1/32	Edit Remove
127.0.0.2/32	Edit Remove

[Add](#)

Other Settings

This setting determines whether the Debug Server will expose itself to remote clients. (Selective, exposes allowed hosts only).

Expose Remotely: ▼

Figure 117 - Allowed Hosts for Tunneling

3. In the Allowed Hosts for Tunneling section, click Add.
The Add New Allowed Host dialog will open.
4. Enter the IP address and Net mask of the machine on which your Zend Studio for Eclipse is installed and click Save.
Your Zend Studio for Eclipse's machine's address will be added to the Allowed Hosts for Tunneling list.
5. Ensure the 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.
6. Under the 'Other Settings' category, set the Debug Server to expose itself to remote clients by selecting 'Always' from the Expose Remotely drop-down list.
7. Click to save your settings.
8. Restart your web server for the settings to take effect.




If Zend Core is installed on your server:



1. Open your Zend Core GUI.
2. Go to the Configuration | Zend Debugger tab.



Figure 118 - Allowed Hosts for Tunneling

3. In the Allowed Hosts for Tunneling section, click Add .
A new Allowed Host for tunneling line will appear with the IP 0.0.0.0.
4. Enter the IP address and Net mask of the machine on which your Zend Studio for Eclipse is installed.
5. Ensure the 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.
6. Under the 'Expose Remotely' category, set the Debug Server to expose itself to remote clients by selecting 'Always' from the drop-down list.
7. Click  to save your settings.
8. Restart your web server for the settings to take effect.

If only the standalone Zend Debugger is installed on your server:



1. Open your zend.ini file.
2. Edit the zend_debugger.allow_tunnel parameter to include the IP address of the machine on which your Zend Studio for Eclipse is installed.
(e.g. 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.

If you want to create integration with Zend Platform, the next step in creating a tunneling connection is to [Configure Platform to Auto detect Studio Settings](#).

If you don't want to create integration with Zend Platform, the next step in creating a tunneling connection is to [Activate your Tunnel](#).

Configuring Platform to Auto Detect Studio Settings

This procedure describes how to configure Zend Platform so that Zend Studio for Eclipse's settings are automatically detected during the Debugging / Profiling of Zend Platform events.



To establish a communication tunnel between Zend Platform and Zend Studio:

1. Access your Zend Platform GUI.
2. Go to the Platform | Preferences tab.

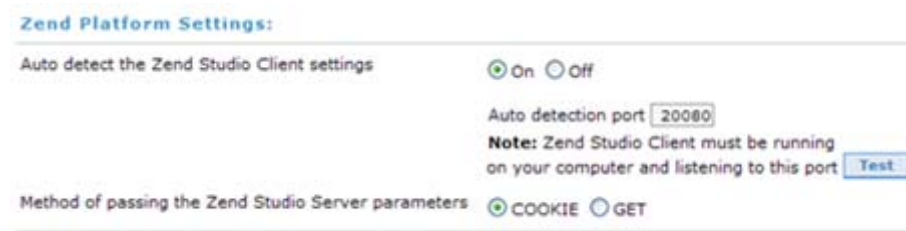


Figure 119 - Communication Tunnel Method

3. In the Zend Platform Settings section, configure the following:
 - **Auto detect the Zend Studio Client Settings** - Set to 'On' to inform Zend Platform of the method of connection to Zend Studio.
 - **Auto Detection Port** - Ensure the Auto detection port number is set to the Broadcast Port number in Zend Studio for Eclipse. This is configured in Zend Studio for Eclipse's [Installed Debuggers preferences page](#) (Window | Preferences | PHP | Debug | Installed Debuggers) by clicking 'Configure'. The default port number is 20080. Click Test to verify that Zend Studio's Broadcast Port is set to the same port number as Platform's Auto Detection Port. This test should only be run when Zend Studio for Eclipse is running.
 - **Method of passing the Zend Studio Server parameters** - Defines the means for passing communication parameters from the Zend Platform to Zend Studio for Eclipse. Choose COOKIE or GET method.

Note:

The default method is Cookie, and it is recommended that you use the default. Platform supports a Get method as well that can be used if experiencing problems with Cookies.

4. Click Save.

The next step in creating a tunneling connection is to [Activate your Tunnel](#).


Activating Tunneling

Once you have configured all the necessary settings in Zend Studio for Eclipse and on your server, you can activate your Tunnel connection.

This procedure describes how to open a tunnel between Zend Studio for Eclipse and your remote server.



To activate Tunneling:

1. Click the arrow next to the Tunneling icon on the toolbar  and select the server which you configured for tunneling.

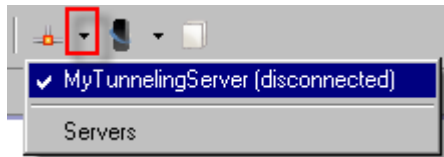



Figure 120 - Tunneling server list


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.

If you set up Platform Integration, you can also now view, debug and profile Platform events. See [Integrating with Zend Platform](#) for more information.

Note:

Several Tunneling sessions can be configured. 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.

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 for Eclipse's CVS functionality:

- [Configuring a CVS connection](#)
- [Importing a project from CVS](#)
- [Uploading projects to CVS](#)

See the 'Working in a Team Environment with CVS' in the Workbench User Guide for more information.

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  on the view's toolbar -or- right-click within the view and select New | Repository Location. The Add CVS Repository dialog will open.

Figure 121 - New Repository Location dialog

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.

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 an CVS repository location to your desktop.

Prerequisites: You should have a CVS repository configured before you follow this procedure. See [Configuring a CVS Connection](#) for instructions.



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, select 'Create a new repository location'. Click Next and 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.

Note:

To enable all Zend Studio for Eclipse'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 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.

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 an 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 already configured an CVS repository, select to "create a new repository location" and click Next. 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 an 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 for Eclipse's SVN functionality:

- [Configuring an SVN connection](#)
- [Importing a project from SVN](#)
- [Uploading projects to SVN](#)

See the Subversive User Guide for more information.

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  on the view's toolbar - or- right-click within the SVN view and select New | Repository Location.
The Add SVN Repository dialog will open.

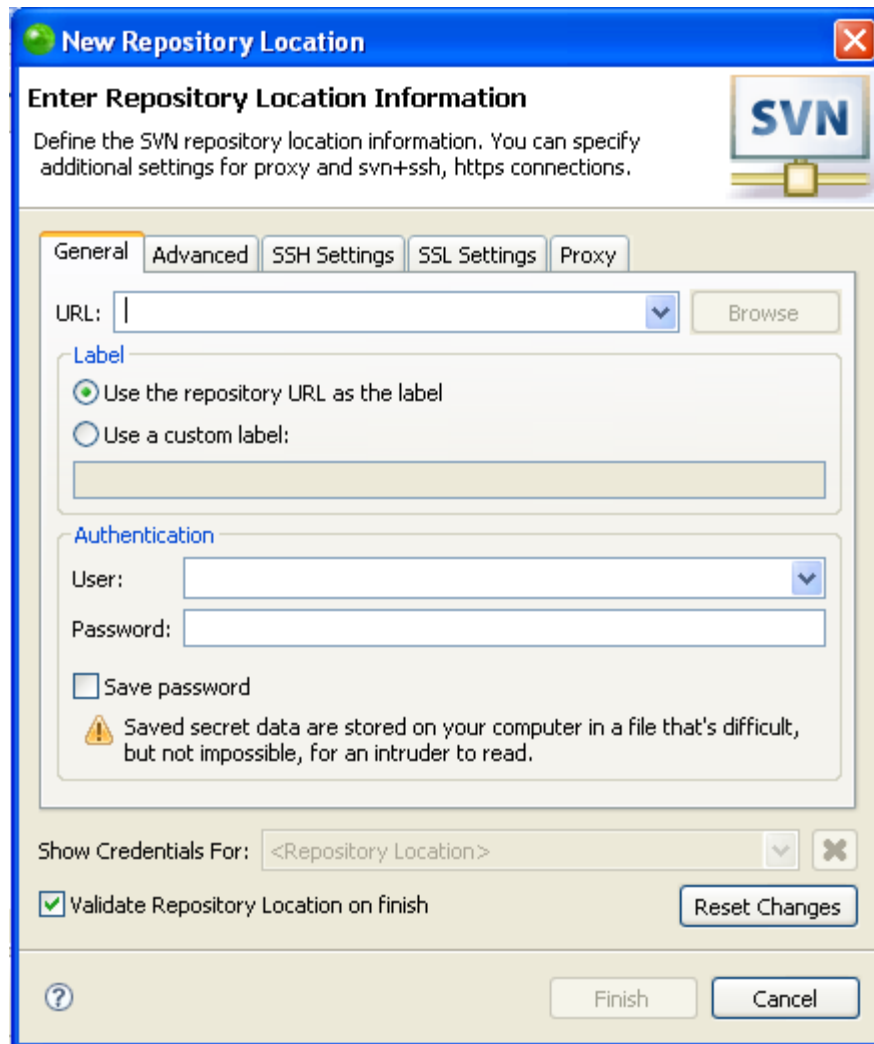


Figure 122 - New Repository Location dialog

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.

Note:

See the Subversive User Guide for more information on using SVN.

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.

Prerequisites: You should have an SVN repository configured before you follow this procedure. See [Configuring an SVN Connection](#) for instructions.



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, select 'Create a new repository location'. Click Next and 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.

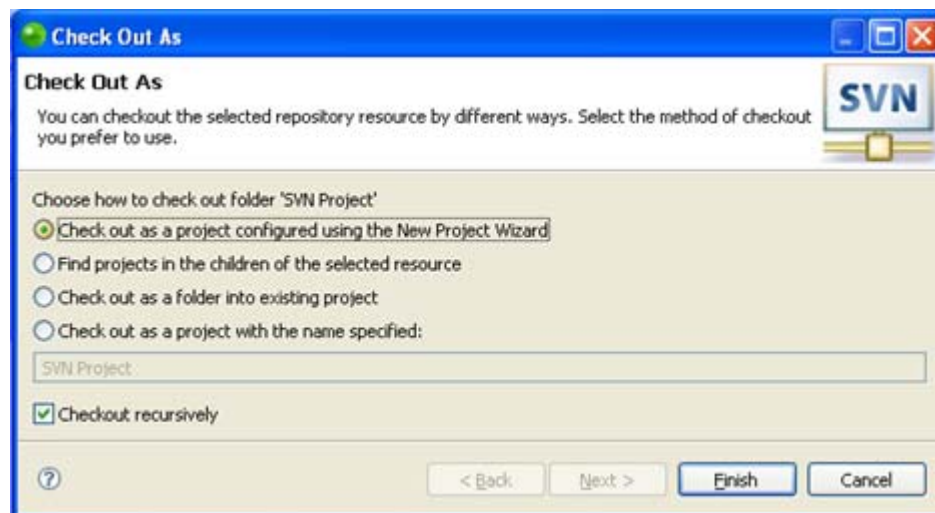


Figure 123 - Check Out As dialog

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 for Eclipse'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 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 using SVN.

Uploading Projects to SVN

Using SVN, you can upload projects and files which other team members can work on.

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. Select 'Use existing repository location', and select your repository from the list. (If you have not already configured an SVN repository, select to "create a new repository location" and click Next. See [Configuring an SVN 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 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.

Note:

See the Subversive User Guide for more information on using SVN.

Using Local History

The following actions can be done 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.

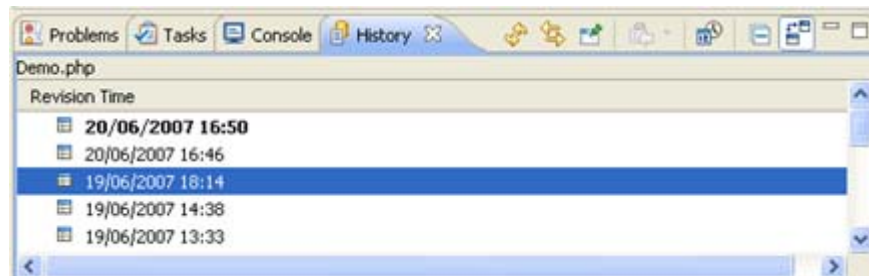


Figure 124 - History view

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.

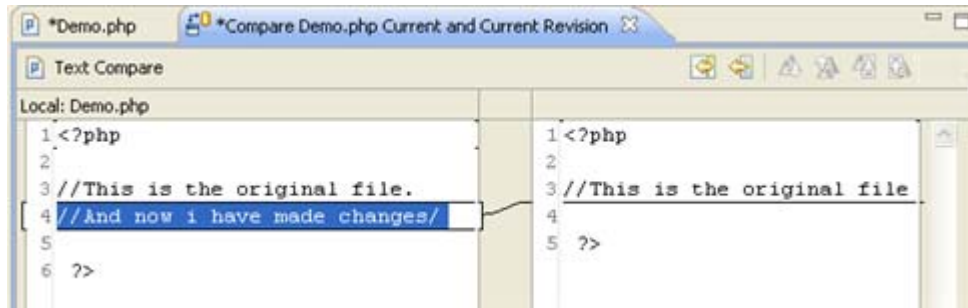


Figure 125 - Local History changes tree

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.

FTP and SFTP Support

Zend Studio for Eclipse's FTP and SFTP support is given through the RSE (Remote System Explorer) plugin.

Files from FTP/SFTP servers opened in Zend Studio for Eclipse can be edited using Zend Studio for Eclipse's PHP editing functionalities such as Debugging, Profiling, Code Assist, Outline views, Formatting and Getter and Setter generation.

To view and edit a file from an FTP/SFTP server in Zend Studio for Eclipse, you must first [Create an FTP/SFTP connection](#).

There are then two ways of working with files linked to FTP/SFTP:

1. [Viewing and editing files that are located on a remote FTP/SFTP server only](#).
2. [Viewing and editing files in your workspace with a link to the FTP/SFTP server](#).


See the RSE User Guide for more on FTP/SFTP connectivity.

Creating an FTP/SFTP Connection

This procedure describes how you can configure an FTP or SFTP connection in Zend Studio for Eclipse.



To create an FTP/SFTP Connection:

1. Open the Remote Systems view by going to Window | Show View | Other | Remote Systems | Remote Systems.
2. Click the 'Define a connection to remote system' button  on the view's toolbar.
3. The New Connection dialog will appear.

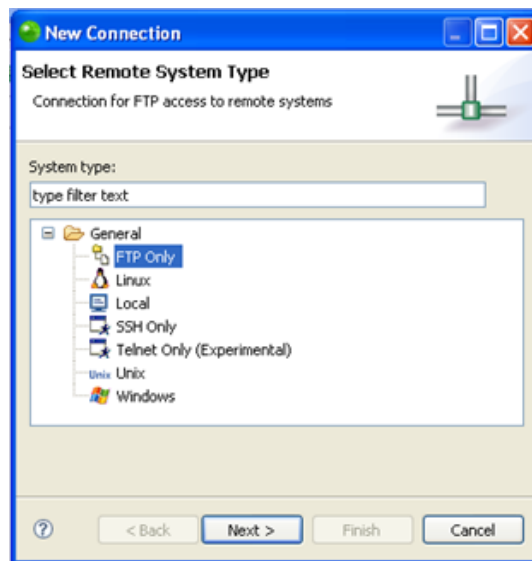


Figure 126 - New Connection dialog

4. Select 'FTP Only' for an FTP connection or 'SSH Only' for an SFTP connection and click Next.

Figure 127 - New FTP Connection dialog

5. Enter the following Remote FTP/SFTP connection details:
 - Parent Profile - Select a Profile from the drop-down list.
 - Host Name - Enter a valid Host name or IP address for the FTP/SFTP Server.
 - Connection name - Enter a Connection Name. A Connection Name is an alias that identifies the server in the Zend Studio for Eclipse file system.
 - Description - Enter a description to help identify your connection (optional.)
 - Verify host name - Mark the checkbox for Zend Studio for Eclipse to verify the existence of the host name before the connection is created.
6. Click Finish.

Your connection will be created and listed in the Remote Systems view.

See the RSE User Guide for more on FTP/SFTP connectivity.

Viewing and Editing Files on an FTP Server

Files located on an FTP/SFTP server can be opened and edited in Zend Studio for Eclipse without having to save them to your Workspace.

This procedure describes how to view and edit files which are situated on your FTP/SFTP server.



To view and edit files located on an FTP/SFTP server:

1. Create an FTP/SFTP connection by following the steps under the "[Creating an FTP/SFTP Connection](#)" topic.
2. Open the Remote Systems view by going to Window | Show View | Remote Systems | Remote Systems.
3. Expand the tree under your FTP/SFTP connection.

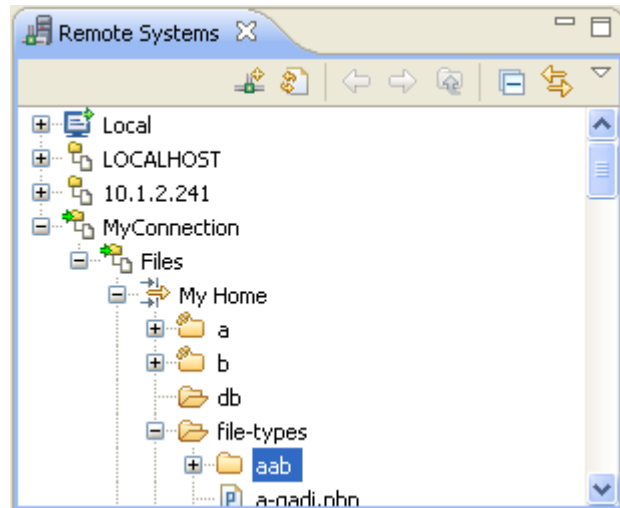


Figure 128 - Remote Systems View

4. Some files and folders might require you to enter your FTP/SFTP login User ID and Password:



5. Double-click a PHP file to open it in a Zend Studio for Eclipse PHP editor. Most of Zend Studio for Eclipse's functionality, such as Code Assist, Formatting, Outline Models and Getter and Setter Generation will be available to the file.
6. Edit the file as required.
7. Click Save.

The edits will be saved on the FTP version of the file.

Viewing and Editing FTP Files in your Workspace

Folders situated on FTP/SFTP servers can be saved onto your Workspace while maintaining the link with the FTP/SFTP server.

Any changes made in the files and folders located on either the FTP/SFTP servers or the Workspace will be updated in both of the file locations.

This procedure describes how to simultaneously edit files and folders on your FTP server and your Workspace.



To save, view and edit files in your Workspace which are located on an FTP/SFTP server:

1. In PHP Explorer view, right-click and select New | Remote Folder -or- go to File on the Menu Bar and select New | Remote Folder.
2. The New Remote Folder creation wizard will open.

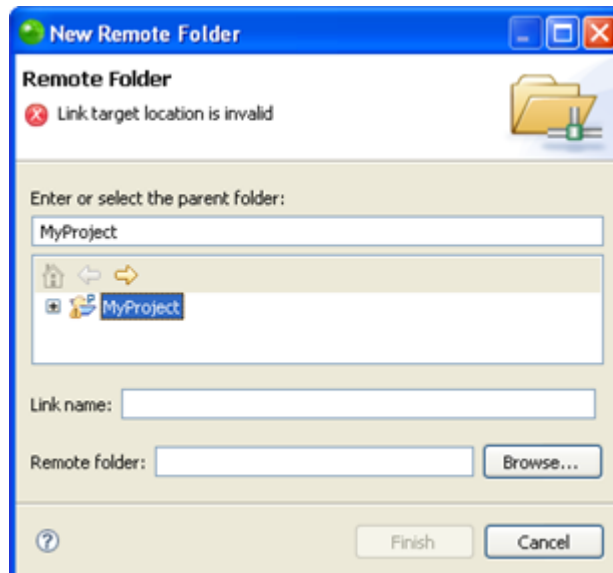


Figure 129 - New Remote Folder creation wizard

3. Select the project in which you want to place your FTP/SFTP folder.
4. In the Remote folder selection, click Browse.
The Select Folder dialog will open.

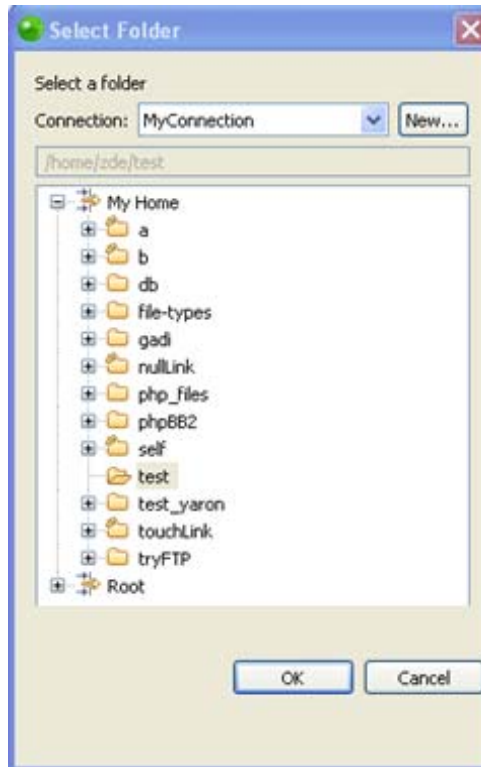


Figure 130 - Select FTP Folder dialog

3. In the 'Connection' drop-down list, select your FTP/SFTP connection.
If you have not yet established a connection, click New and follow the instructions under [Creating an FTP/SFTP Connection](#).
4. Your FTP/SFTP file system will be displayed.
5. Select the folder you would like to link to your Workspace and click OK.
6. You will be returned to the New Remote Folder dialog.

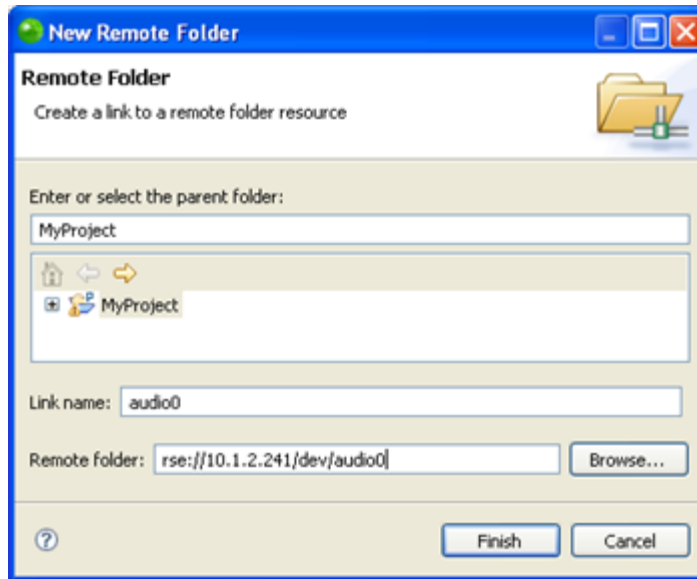


Figure 131 - New Remote Folder creation wizard 2

7. The link name will be automatically be updated with the name of your folder. Edit this if required.
8. Click Finish.

Your FTP/SFTP folder and its contents will be added to your Workspace.

Most of Zend Studio for Eclipse's functionality, such as Code Assist, Formatting, Outline Models and Getter and Setter Generation will be available to help you edit the file.

Any changes made to the files and folders in your Workspace will be updated in the FTP copy, and conversely any changes made on the FTP copies will be updated in your Workspace.

Folders that are linked to FTP will be displayed with a linked connection icon .

Note:

Deleting an FTP/SFTP folder from your Workspace will only delete it from your Workspace.

However, deleting a file contained within a folder will also delete the file on the FTP/SFTP server.

Adding new files to the linked folder in your workspace will upload the new files to your FTP/SFTP server.

Integrating with Zend Guard

Zend Guard's Integration with Zend Studio for Eclipse allows you to:

- [Encode your Zend Studio for Eclipse projects using Zend Guard.](#)
- [Open and edit Zend Guard projects in Zend Studio for Eclipse.](#)

Encoding Projects Using Zend Guard

This procedure describes how to open your Zend Studio for Eclipse Projects in Zend Guard in order to encode them.

Zend Guard must be configured in Zend Studio for Eclipse 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 for Eclipse, 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.

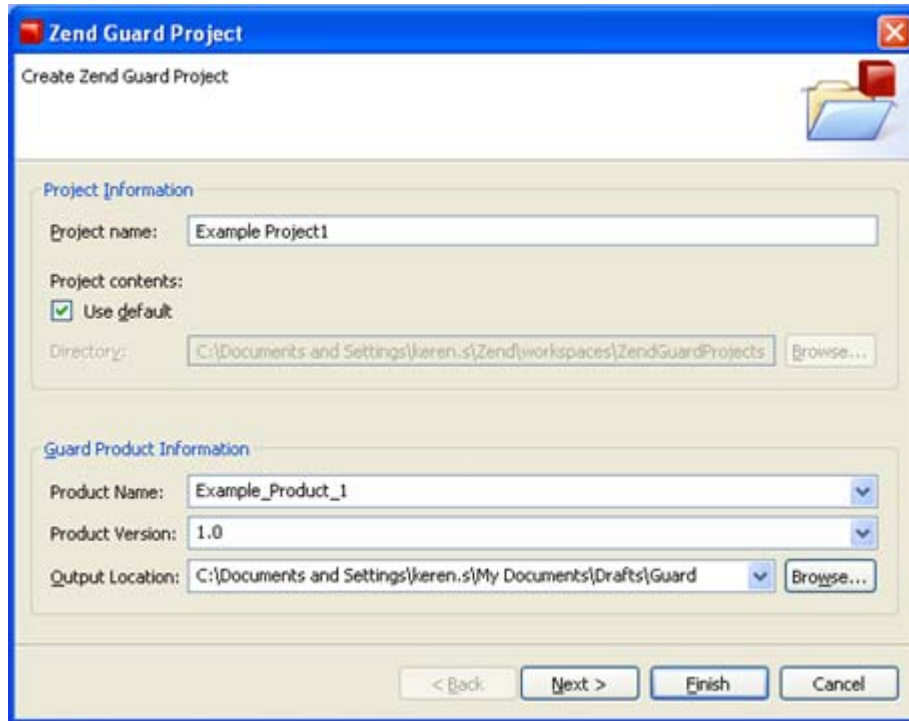


Figure 132 - Zend Guard Project creation dialog

2. Enter the following information in the relevant fields:
 - Project Name - This will automatically be taken from the name of your Zend Studio for Eclipse 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.

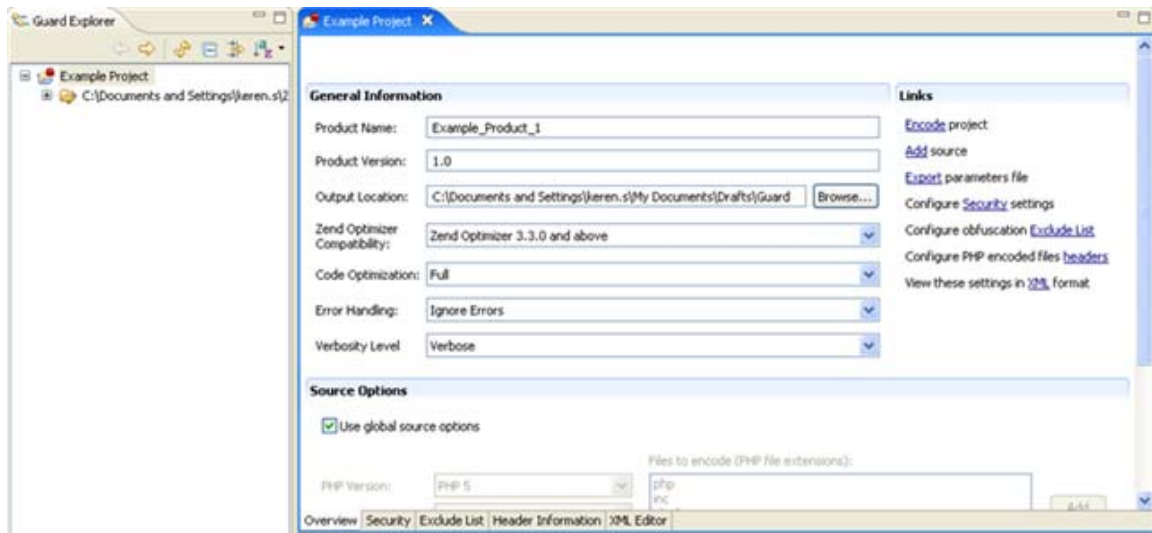


Figure 133 - Zend Guard

See the Zend Guard User Guide 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 Core](#) or can be downloaded from the [Zend Guard product page](http://www.zend.com/en/products/guard/optimizer) (<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 for Eclipse

This procedure describes how to open Zend Guard's file in Zend Studio for Eclipse, enabling users to extend Zend Guard with the rich editing features included in Zend Studio for Eclipse. This provides users with a seamless workflow for fixing and modifying code through Zend Guard.



To open a Zend Guard file in Zend Studio for Eclipse:

1. In Zend Guard, go to Edit | Preferences | Zend IDE.
2. Enter the path to Zend Studio for Eclipse 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 for Eclipse integration has been configured as described in steps 1 and 2.
4. Zend Studio for Eclipse will open and the file will be displayed in an editor. All of Zend Studio for Eclipse's editing capabilities will now be available to the file.
5. Once you have made the required edits, click Save  on Zend Studio for Eclipse's Menu Bar to save the edits made to the file. Changes will be updated in the file accessible through Zend Guard.

Viewing RSS Feeds

These procedure describe how to view RSS feeds within Zend Studio for Eclipse'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.

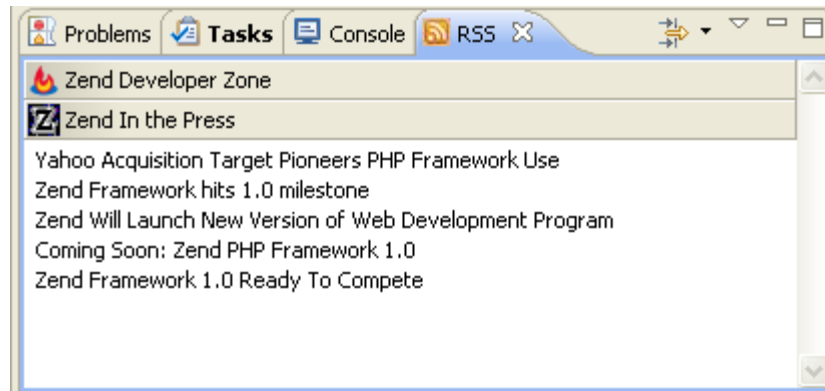



Figure 134 - RSS view


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 for Eclipse's internal browser.



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 Add Channel.
3. The Add New Channel dialog will be displayed.

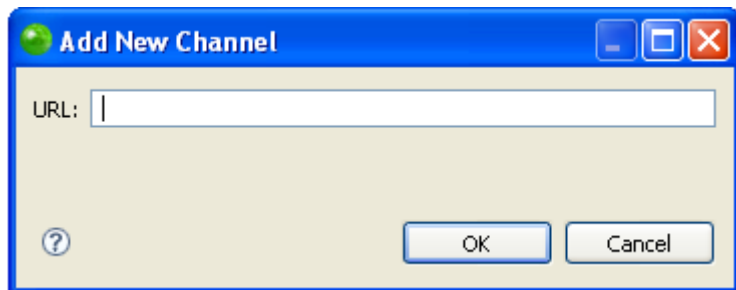


Figure 135 - Add New RSS Channel dialog

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 WSDL

Using Zend Studio for Eclipse, you can either reference and view existing WSDL files in your PHP Code or create your own WSDL files using the WSDL file generator wizard.

See [Generating WSDL Files](#) for more on creating your own WSDL files.

See [Incorporating WSDL Files](#) for more on referencing WSDL files in your code.

Generating WSDL Files

Generating a WSDL File is the process of selecting classes and non-class functions from your PHP files to be included in a WSDL file.

The WSDL Generator creates WSDL files based on user defined Configuration sets that are created in the wizard. This means the wizard remembers configuration settings for each WSDL file.

Note:

Generated WSDL files contain the public functions of a class only.

This procedure describes how to generate a WSDL file from a PHP file.



To create a WSDL file:

1. Create the PHP file which includes all the services you want to include in your WSDL file.

Note:

The functions must be documented in order for the output file to include element's descriptions.

See "[Adding PHP DocBlock Comments](#)" for more on documenting PHP code.

```
<?php
class service{
    /**
     * Enter description here...
     *
     * @param service3| $a
     * @param string $b
     * @return string
     */
    function op1($a, $b){
        return 1;
    }
}

class service2{
    /**
     * Enter description here...
     *
     * @param integer $a
     * @param integer $b
     * @return integer
     */
    function op2($a, $b){
        return 1;
    }
}
?>
```

2. From the Menu Bar, go to File | Export | PHP | WSDL File
-Or- In PHP Explorer view, right-click and select File | Export | PHP | WSDL File.
The Generate WSDL File dialog will appear.

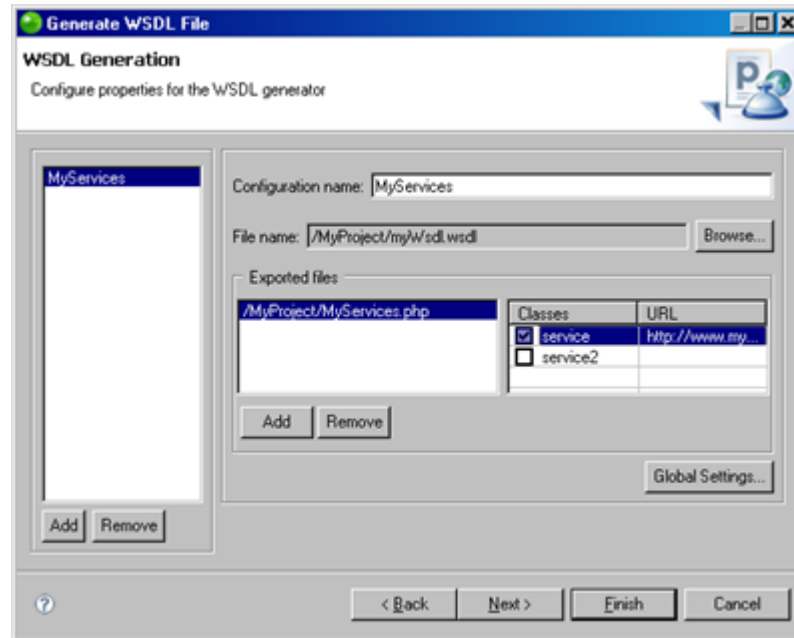


Figure 136 - New WSDL File dialog

3. Enter a name for the WSDL file creation configuration.
4. Click Browse to select a location where the new WSDL file will be saved. If you are regenerating an existing WSDL file, use the browse button to navigate to the WSDL file's location. The old WSDL file will be overwritten with the new file.
5. Click Add and select a file from which you want the classes to be taken. Available classes and non-class functions will appear in the pane on the right.

Note:

All Non-Class functions in a file are considered a single class and are mapped to a single port. However, non-class functions from different files in the same WSDL configuration are given a different port URL.

6. Mark the classes which you want to include.
7. If you want the classes to be taken from a different URL, enter it in the URL column.
8. Click Global Settings for additional advanced settings for creating and transferring information:

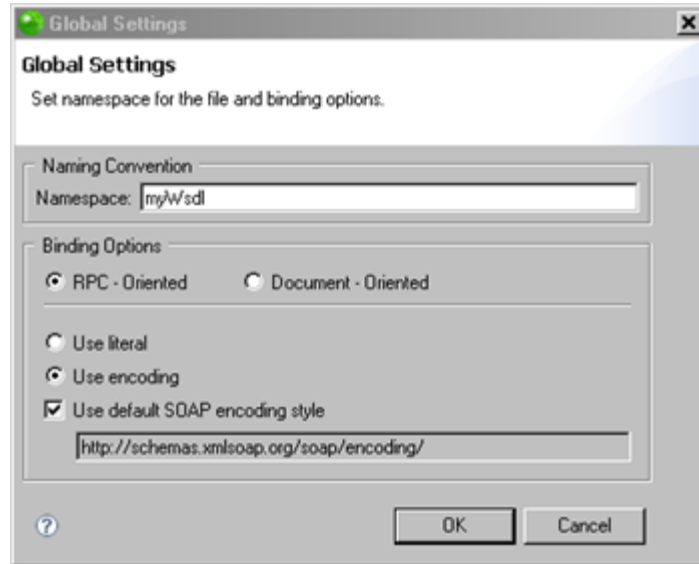


Figure 137 - Global Settings dialog

- Namespace - Defining a Namespace provides a method of avoiding element name conflicts.
 - Binding Options - Choose the format for transferring information. The binding style can be RPC oriented or Document oriented.
 - Encoding - If you choose to use encoding, make sure you specify the URL of the proper encoding file to determine the encoding type applied to the WSDL file.
9. Click OK to return to the WSDL Generation dialog.
 10. Once all the added files have been defined, click Next to view the configuration summary dialog.
 11. If some of the parameters or return values of a function are of types not found in your project, an additional screen will appear instructing to enter the URL to the XML file containing the class schema.

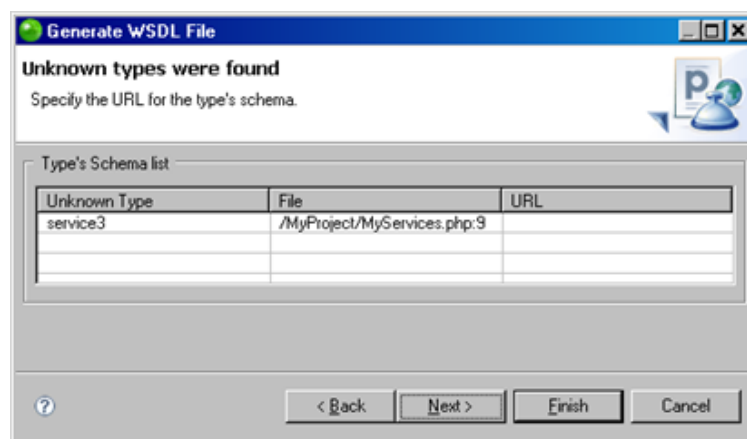


Figure 138 - Unknown types dialog

- Click Finish to approve the configuration settings and generate the WSDL file.

A WSDL file will be generated with the functions defined in your PHP file:

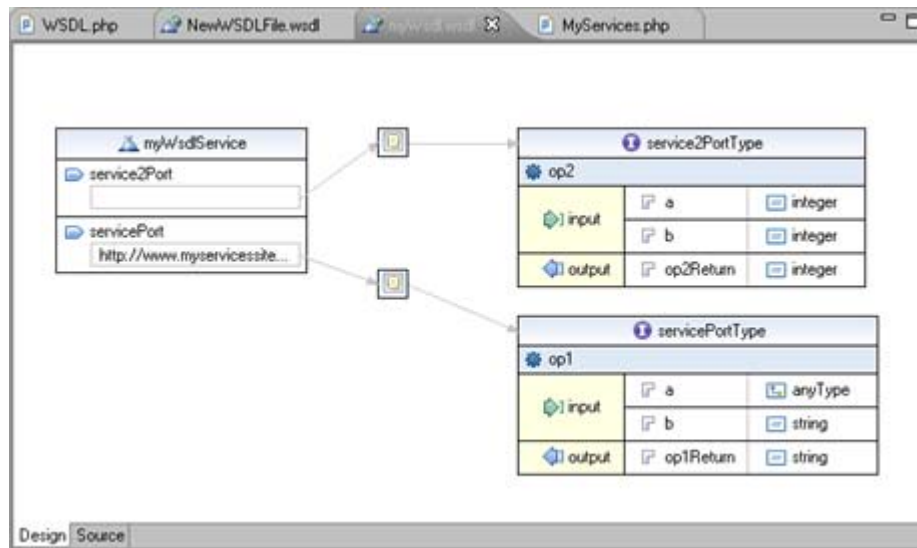


Figure 139 - Generated WSDL File - Design view

Once you have generated a WSDL file, you can use your configuration to easily regenerate it.



To generate a WSDL file after creating a configuration set:

- From the Menu Bar, go to File | Export | PHP | WSDL File
-Or- In PHP Explorer view, right-click and select File | Export | PHP | WSDL File.
The Generate WSDL File dialog will appear.
- Select a configuration set from the list.
- Click Finish.

Your WSDL file will be generated.

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 for Eclipse (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).



To create a SOAP Client:

Method 1

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.



To create a SOAP Client:

Method 2

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 WSDL 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.

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.

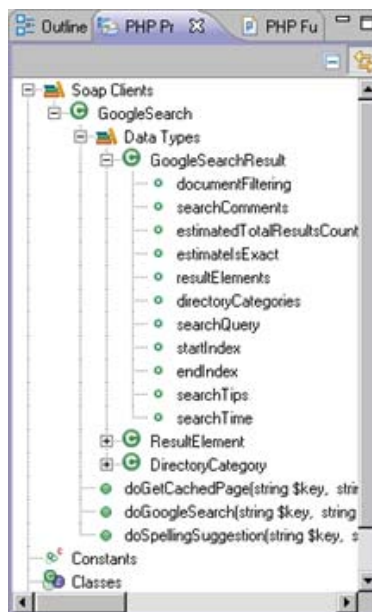
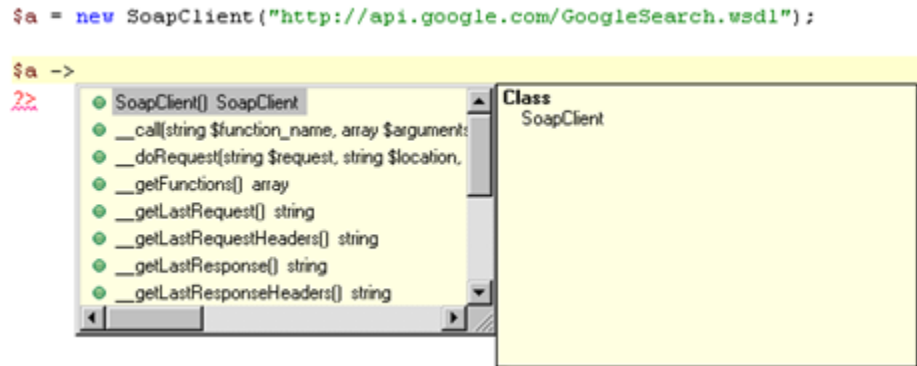


Figure 140 - PHP Project Outline view - with the GoogleSearch classes

Code 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.

Reference

[PHP Perspectives and Views](#)

[PHP Perspective Menus](#)

[PHP Perspective Main Toolbar](#)

[PHP Preferences](#)

[Useful Links](#)

[Keymap](#)

PHP Perspectives and Views

Zend Studio for Eclipse comes with 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](#)

See the Workbench User Guide for more on Perspectives and Views.

PHP Perspective

The PHP Perspective is Zend Studio for Eclipse's default perspective. It incorporates all Zend Studio for Eclipse's PHP project/file creation, inspection and editing functionality.

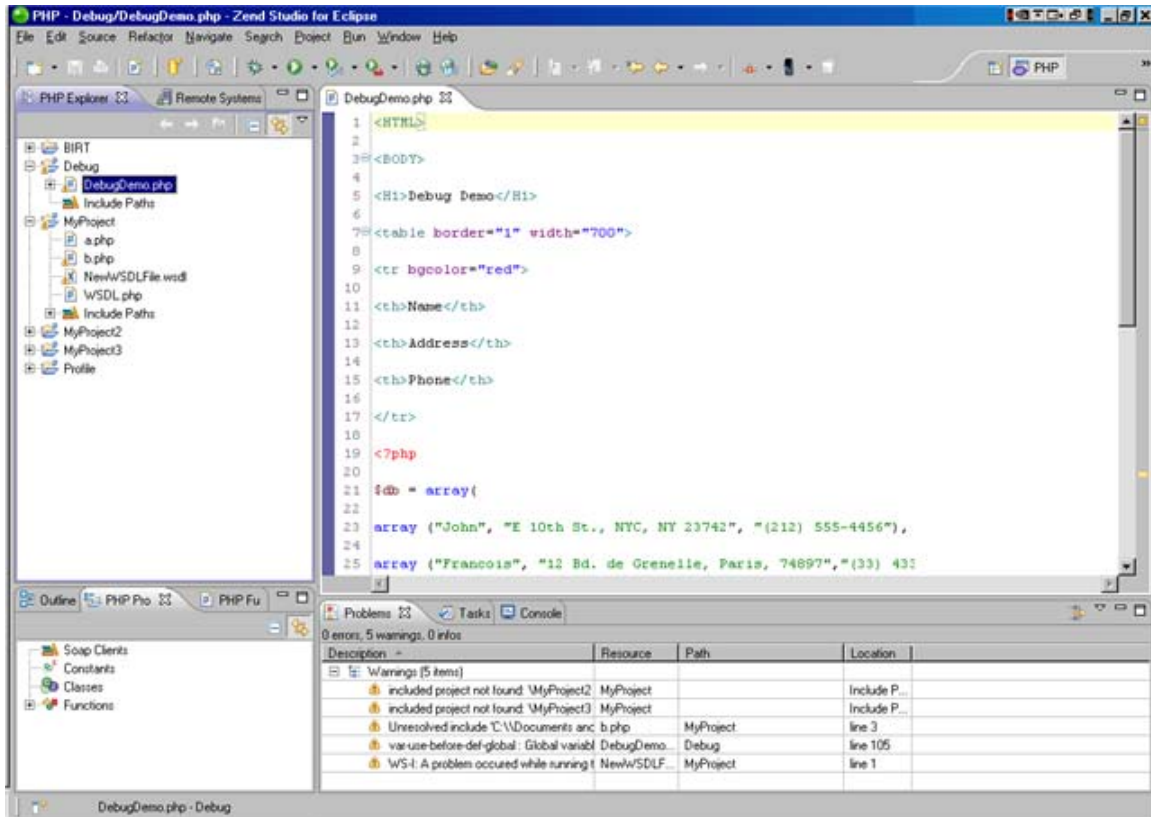


Figure 141 - PHP Perspective

The PHP Perspective contains the following views:

- [PHP Explorer view](#)
- [PHP Functions View](#)
- [PHP Project Outline View](#)
- [Outline View](#)
- [Remote Systems View](#)

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.

The PHP Explorer view shows the PHP element hierarchy of the PHP projects in the Workbench. It provides you with a PHP-specific view of the available resources in the Navigator. The element hierarchy is derived from the project's build paths. 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 appear in the PHP editor.

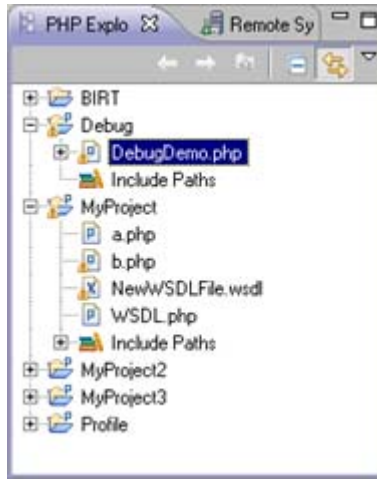









Figure 142 - PHP Explorer view

PHP Explorer view Toolbar Commands

	<p>Back/Forward</p>	<p>Scrolls backwards and forwards through your recently navigated resources.</p> <p>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).</p>
	<p>Up</p>	<p>Navigates up one level.</p> <p>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).</p>
	<p>Collapse All</p>	<p>Collapses the list of elements</p>
	<p>Link with Editor</p>	<p>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.</p>

PHP Explorer view Menu Commands

The view's menu can be accessed through the view menu icon .

	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 the Working Sets topic in the Workbench User Guide for more informations.
	Deselect Working Set	Deselects the Working Set (if selected).
	Edit Active Working Set	Allows you to edit the selected Working Set. See the Working Sets topic in the Workbench User Guide for more informations.
	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.
	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 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.

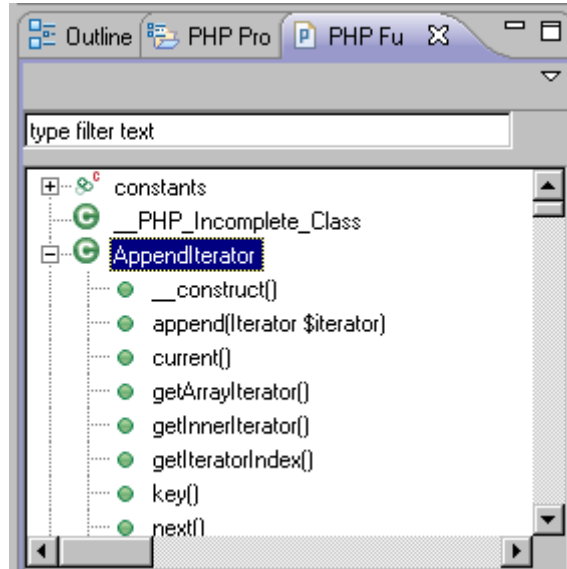


Figure 143 - PHP Functions view

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.




Figure 144 - PHP Manual

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.

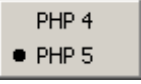
PHP Manual sites can be added and edited from the [PHP Manual Preferences](#) page.

PHP Functions view Toolbar commands

	Filter Text box	Allows you to find a particular function. Start typing the function name. Relevant results will be displayed below it.
---	-----------------	--

PHP Functions view Menu Commands

The view's menu can be accessed through the view menu icon .

	PHP Version selection	Allows you to choose between PHP4 and PHP5 functions.
---	-----------------------	---

PHP Project Outline View

The PHP Project Outline view displays a list of Soap Clients, 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.

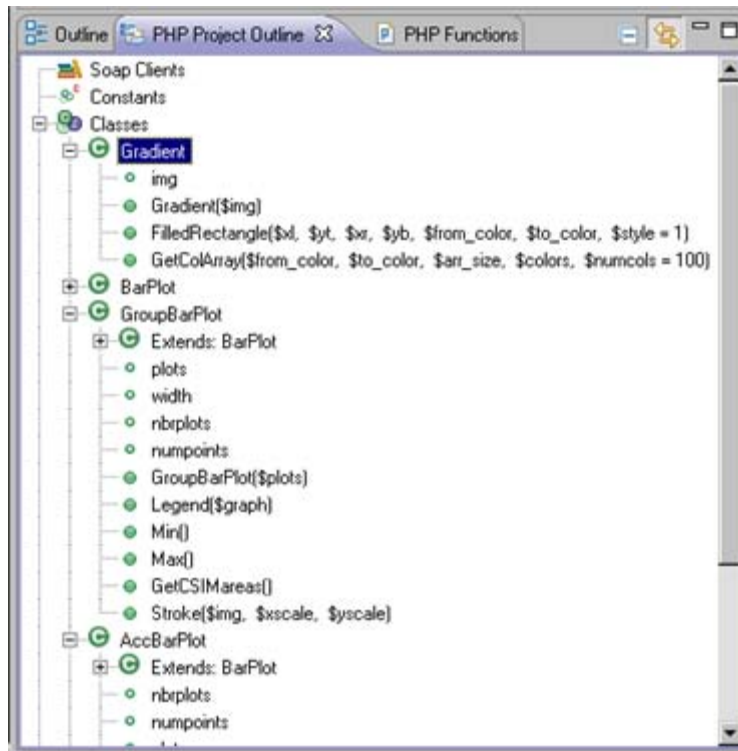




Figure 145 - PHP Project Outline view

PHP Project Outline view Toolbar commands

	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.

Outline View

The PHP Project 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.

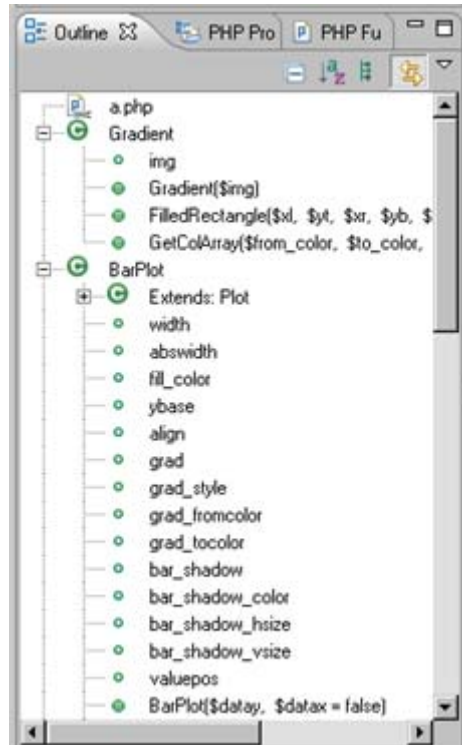










Figure 146 - PHP Outline view

Features

- The Outline View is updated interactively according to changes made in the files.
- Each type of PHP element has a unique icon to represent it:
 -  Reserved PHP Words
 -  Functions
 -  Templates
 -  Classes
 -  Interfaces
 -  Constants
 -  Variables (public)
- 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:

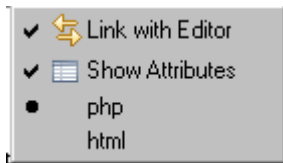
1. Right-click the element in Outline view.
2. Select Source | Add PHP Doc -or- Generate Getters and Setters.

Outline View Toolbar Commands

	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 selected. If unselected, elements will be displayed in the editor when they are double-clicked.

Outline View Menu Commands

The view's menu can be accessed through the view menu icon



	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.
	Show Attributes	If selected, element attributes will be displayed. These are defined by the element's PHP Doc Block .
	PHP/HTML selection	Toggles the view to display PHP or HTML elements.

Remote Systems View

The Remote Systems view helps you create, view and manage your Remote Systems connections such as FTP and SFTP connections.

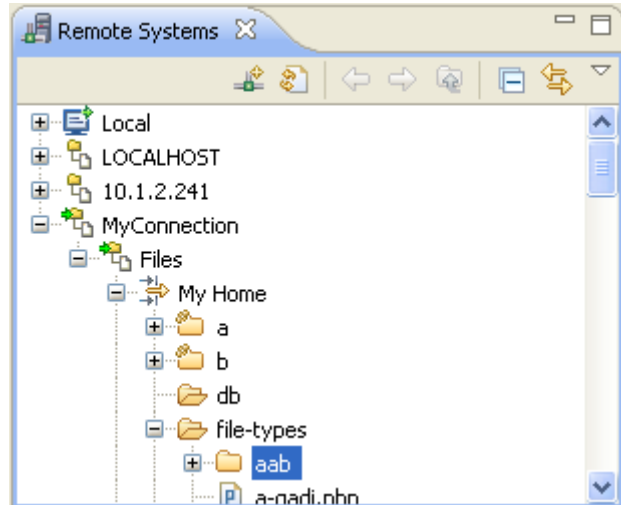








Figure 147 - Remote Systems view

Remote Systems view Toolbar Commands

	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.

PHP Functions View Menu Commands

The view's menu can be accessed through the view menu icon .

New Connection...	Opens the 'New Connection' dialog.
Work with Profiles	Opens the Team profile view. See 'Remote System Explorer Profiles' in the RSE User Guide for more information.
Refresh All	Refreshed 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.
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.

See the RSE User Guide for more on FTP/SFTP connectivity.

PHP Debug Perspective

The PHP Debug Perspective can be launched automatically when a Debug session is run. It contains views which allows you to control and monitor the debugging process.

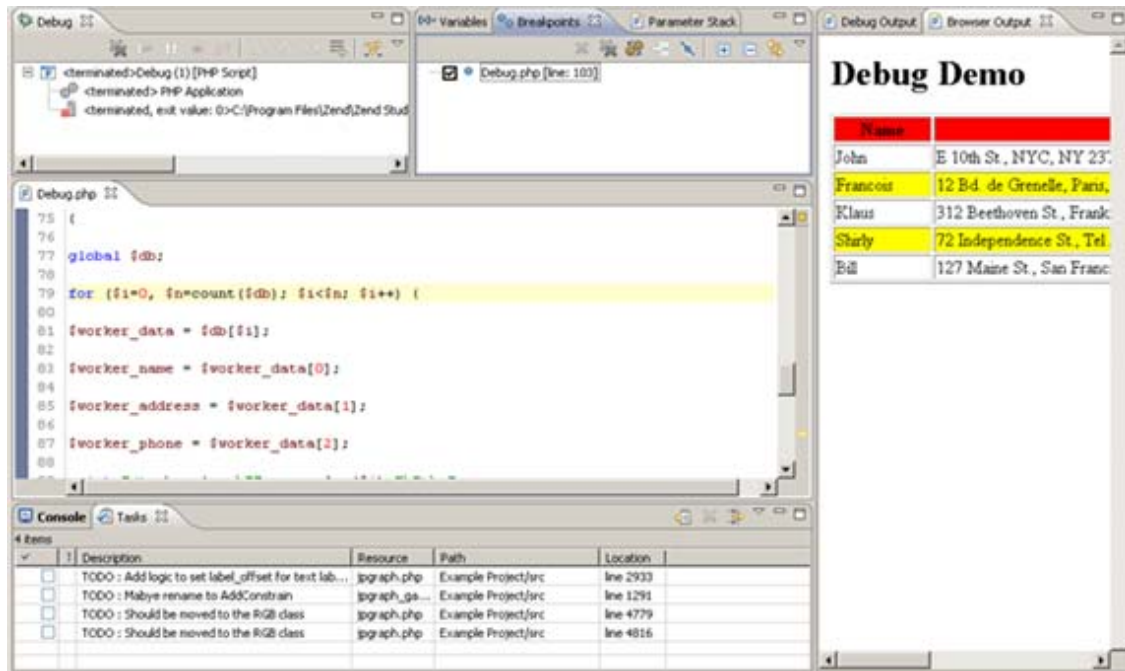


Figure 148 - PHP Debug perspective

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 off) certain functions.
- [Variables View](#) - Will display the various variables in your script.
- [Breakpoints View](#) - Will display the breakpoints you have entered
- [Parameter Stack View](#) - Will display the parameters through which functions are reached.
- [Debug Output View](#) - Will show the textual output of the script. This will be updated as the debugging process continues.
- [Browser Output View](#) - Will show 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.
- Editor - Will display the code at the relevant sections, according to which line is selected in the Debug View window.
- Console View - Will display any error and warning messages.
- Tasks - If you had added any tasks to your script, these would be displayed here.

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

The Debug view displays the stack trace and allows you to monitor control the Debugging process.

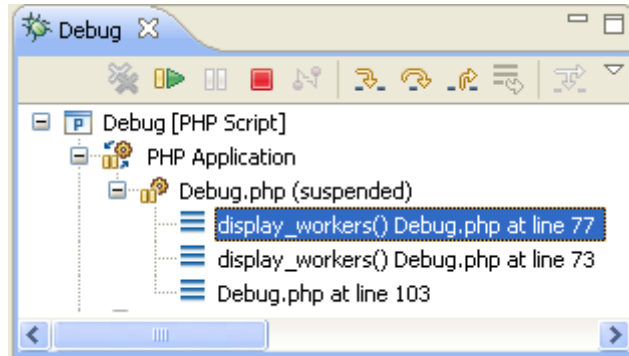









Figure 149 - Debug view

Debug View Toolbar Commands:

	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	Change whether step filters should be used in the current Debug View.

Variables View

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.

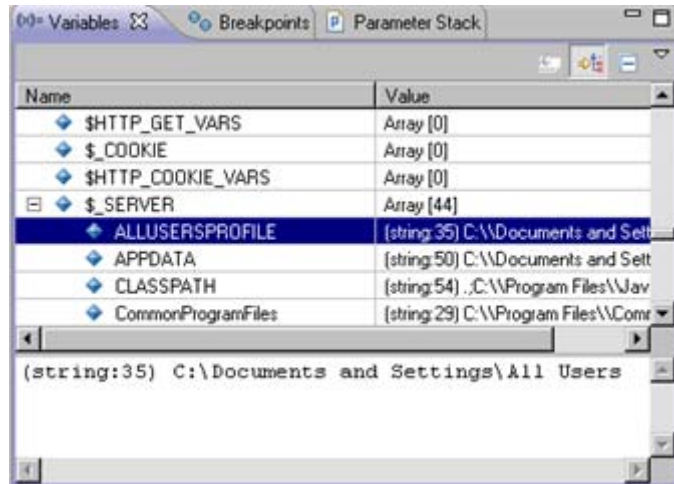





Figure 150 - Variables view

Note:

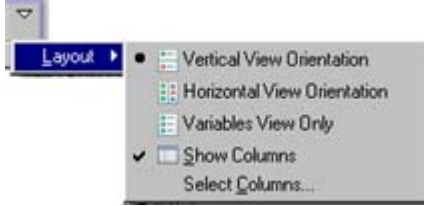
Right-click a variable and select Watch or Create Watch Expression to add the variable to the [Expressions view](#).

Variables View Toolbar Commands

	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 .



<p>Layout</p>	<p>Defines the view's layout:</p> <ul style="list-style-type: none"> ▪ 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 - Divided the view into columns. ▪ Set Columns - Only available if Show columns is selected. Allows you to choose which of the following columns to display: <ul style="list-style-type: none"> • Name • Declared Type • Value • Actual Type
---------------	--

Breakpoints View

The Breakpoints view displays and allows you to monitor and control the breakpoints set in the files being debugged.

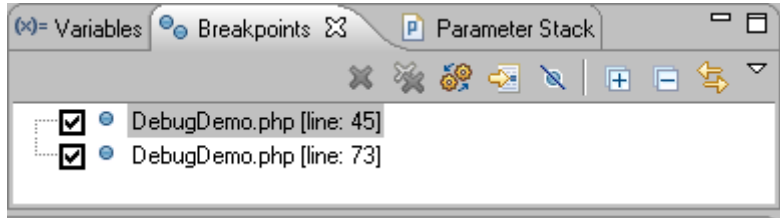










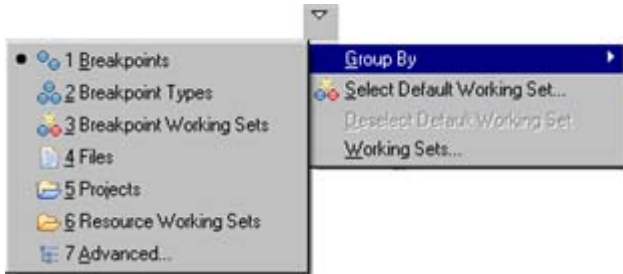
Figure 151 - Breakpoints view

Breakpoints View Toolbar Commands

	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, if 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.

PHP Functions View Menu Commands

The view's menu can be accessed through the view menu icon .



	Group By	<ul style="list-style-type: none"> ▪ 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.

Parameter Stack view

The Parameter Stack view displays the parameters executed when stepping into a function during the debugging process.

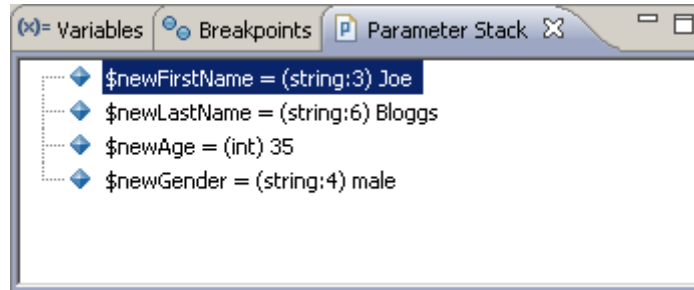


Figure 152 - Parameter Stack view

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.

Debug Output View

The Debug Output view shows the textual output of the script. This will be updated as the debugging process continues.



```
X-Powered-By: PHP/5.2.2
Set-Cookie: ZendDebuggerCookie=127.0.0.1%3
Content-type: text/html

<HTML>
<BODY>
<H1>Debug Demo</H1>
<table border="1" width="700">
<tr bgcolor="red">
<th>Name</th>
<th>Address</th>
<th>Phone</th>
<td>(44) 332-8065</td>
</tr>
<tr bgcolor="yellow">
<td>Shirly</td>
<td>72 Independence St., Tel Aviv, Israel 6728:
<td>(972) 156-7777</td>
</tr>
<tr bgcolor="white">
<td>Bill</td>
<td>127 Maine St., San Francisco, CA 90298</td>
<td>(415) 555-6565</td>
```

Figure 153 - Debug Output view

Browser Output View

The Browser Output view will show the output of the script to a browser. This will be updated as the debugging process continues.

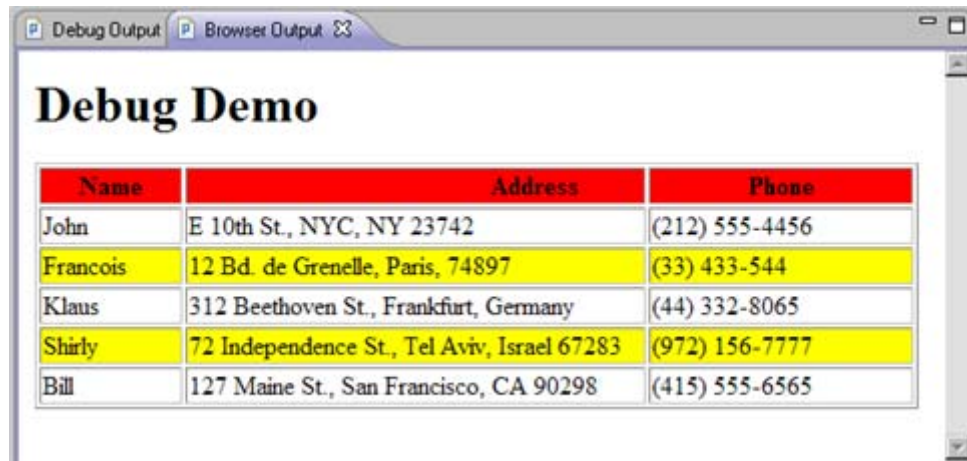


Figure 154 - Browser Output view

Expressions View

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.

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 | Expressions.

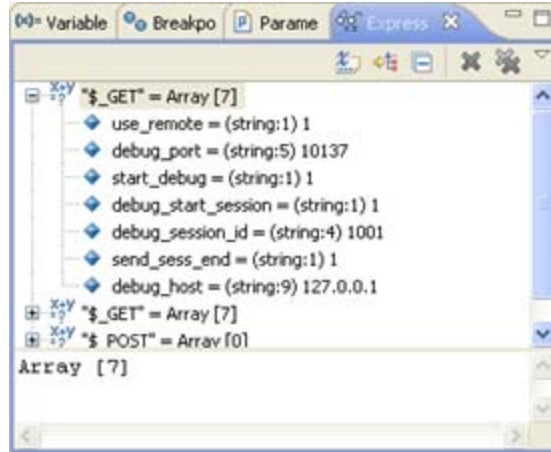






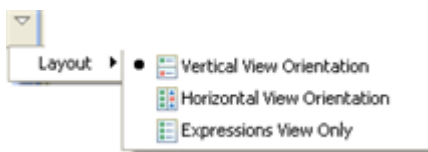
Figure 155 - Expressions View

Expressions View Toolbar Commands

	Show Type Names	If selected, type names will be displayed.
	Show Logical Structure	Shows the logical structure.
	Collapse All	Collapses the list.

Expressions View Menu Commands

The view's menu can be accessed through the view menu icon .



Layout	<p>Defines the view's layout:</p> <ul style="list-style-type: none"> ▪ 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.
--------	---

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.

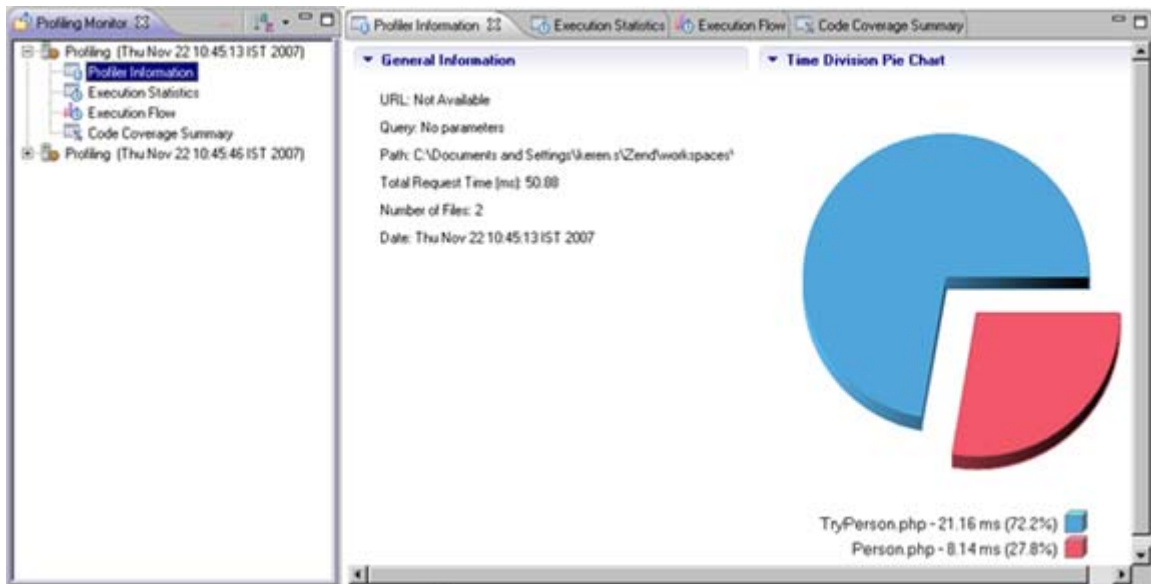


Figure 156 - Profiling Perspective

The PHP Profile Perspective contains the following views:

- [Profiling Monitor](#)
- [Profiler Information](#)
- [Execution Statistics](#)
- [Execution Flow](#)
- [Code Coverage Summary](#)

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.

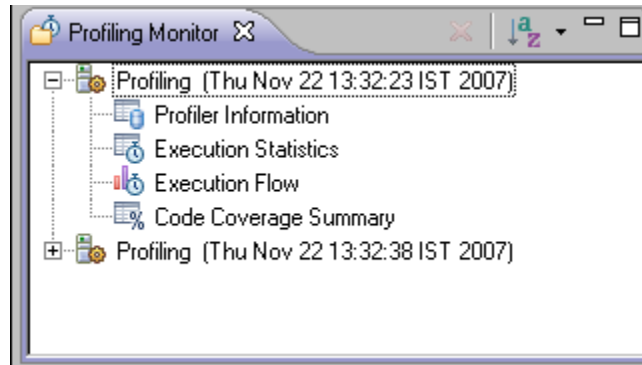




Figure 157 - Profiling Monitor view

Profiling Monitor View Toolbar Commands:

	Delete Session	Delete a Profiling session from the list. This will only be enabled if a profiling session is selected.
	Sort Profile Sessions	Click the arrow next to the Profile Session to sort the Profile Session list by date or time.

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.

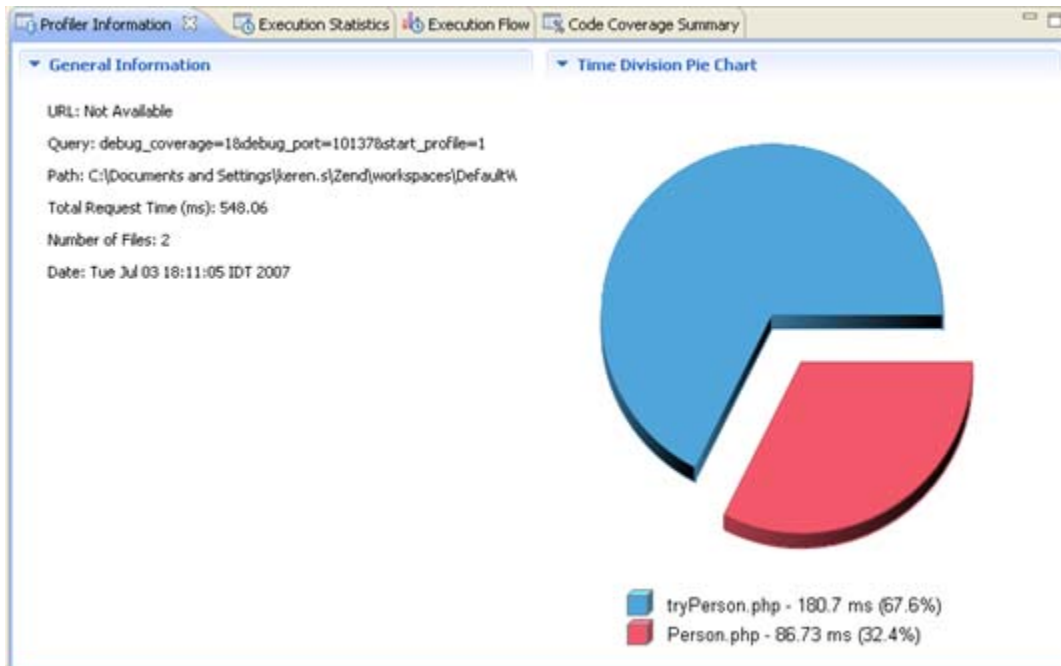


Figure 158 - Profiler Information view

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.

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
setTime	3	0.000004	0.000013	0.000000	0.000013

Figure 159 - 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.

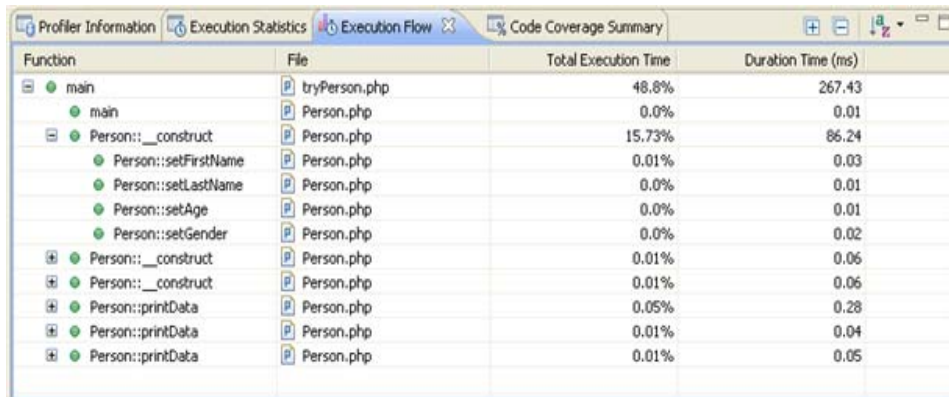
Execution Statistics View Toolbar Commands:

	Filters...	Click the arrow next to the icon to select to display only the results with: <ul style="list-style-type: none"> • 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.

Execution Flow View

The 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.



Function	File	Total Execution Time	Duration Time (ms)
main	tryPerson.php	48.8%	267.43
main	Person.php	0.0%	0.01
Person::__construct	Person.php	15.73%	86.24
Person::setFirstName	Person.php	0.01%	0.03
Person::setLastName	Person.php	0.0%	0.01
Person::setAge	Person.php	0.0%	0.01
Person::setGender	Person.php	0.0%	0.02
Person::__construct	Person.php	0.01%	0.06
Person::__construct	Person.php	0.01%	0.06
Person::printData	Person.php	0.05%	0.28
Person::printData	Person.php	0.01%	0.04
Person::printData	Person.php	0.01%	0.05

Figure 160 - Profiler Execution Flow

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.

Execution Flow view Toolbar Commands

	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.

Code Coverage Summary View

The Code Coverage Summary view displays a summary of the lines of code that were covered during the Profiling process.

It contains the following columns:

- 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.)

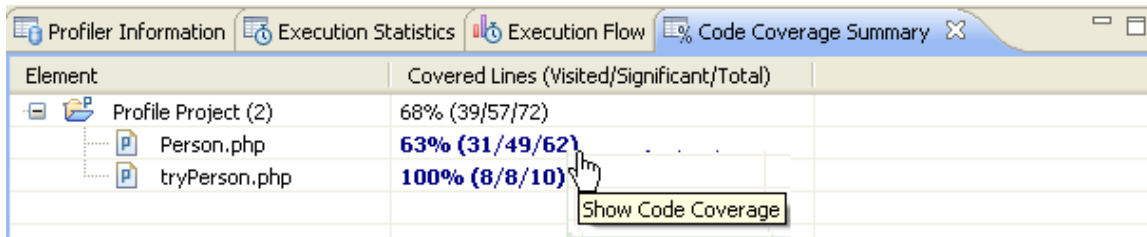
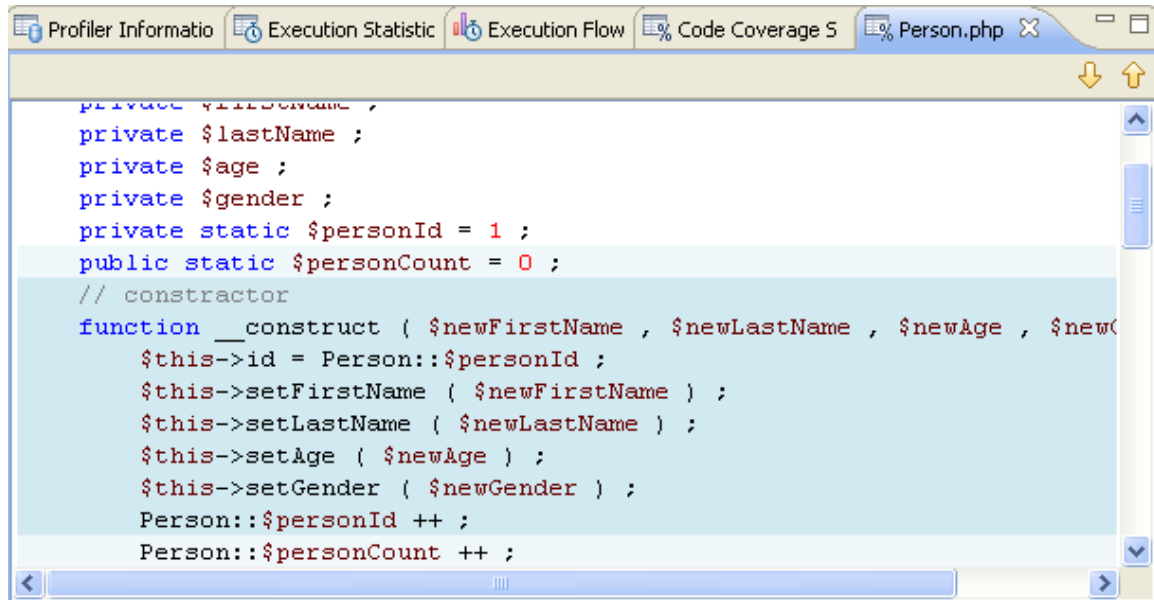


Figure 161 - Code Coverage Summary

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



The screenshot shows the Zend Studio interface with the 'Code Coverage S' view selected. The editor displays the following PHP code, with lines 10 through 20 highlighted in light blue to indicate they are covered:

```
private $firstName ;  
private $lastName ;  
private $age ;  
private $gender ;  
private static $personId = 1 ;  
public static $personCount = 0 ;  
// constructor  
function __construct ( $newFirstName , $newLastName , $newAge , $newGender )  
{  
    $this->id = Person::$personId ;  
    $this->setFirstName ( $newFirstName ) ;  
    $this->setLastName ( $newLastName ) ;  
    $this->setAge ( $newAge ) ;  
    $this->setGender ( $newGender ) ;  
    Person::$personId ++ ;  
    Person::$personCount ++ ;  
}
```

Figure 162 - Covered Lines

PHP Perspective Menus

Zend Studio for Eclipse'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 following descriptions are relevant to the default settings for the PHP perspective:

Menu Option	Description
File	<p>Allows you to carry out various functions on active files and folders, as well as organizing current files and creating new items.</p> <p>The File menu options are: New, Open File, Close, Close All, Save, Save As, Save All, Revert, Move, Rename, Refresh, Convert Line Delimiters To, Print, Switch Workspace, Import, Export, Properties, Last Viewed Files and Exit.</p>
Edit	<p>Normal text editing functionality, as well as features such as tasks and bookmarks which are specifically related to editing code.</p> <p>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.</p>
Source	<p>Allows you to organize your scripts by adding or removing comments and formatting the script to make it more easily viewable.</p> <p>The Source menu options are: Toggle Comment, Add Block Comment, Remove Block Comment, Format Document and Format Active Elements.</p>
Refactor	<p>Allows you to edit names and locations of files and resources while maintaining the links between the files.</p> <p>The Refactor menu options are: Organize Imports, Rename and Move.</p>
Navigate	<p>Allows you to navigate through your scripts in order to find relevant resources, information and text.</p> <p>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.</p>
Search	<p>Allows you to Search for text or PHP elements in your workspace.</p> <p>The Search menu options are: Search, File and Text.</p>

Project	<p>Allows you to carry out different functions on your projects.</p> <p>The Project menu options are:</p> <p>Open Project, Close Project, Build All, Build Project, Build Working Set, Clean, Build Automatically, Generate PHPDoc..., Encode Project and Properties.</p>
Run	<p>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 functionality.</p> <p>The Run menu options are:</p> <p>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.</p>
Window	<p>The Window menu allows you to customize your workspace display.</p> <p>The Window menu options are:</p> <p>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.</p>
Help	<p>Allows access to the most updated information on all aspects of Zend Studio for Eclipse, as well as allowing access to software updates and registration locations so that you can get the most out of the product.</p> <p>The Help menu options are:</p> <p>Welcome, Help Contents, Search Dynamic Help, Key Assist, Tips and Tricks, Software Updates, Register, Tip of the Day and About Zend Studio for Eclipse.</p>

File

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	<p>Creates various items and types of files.</p> <p>To see the list of new items that can be created through this menu, see the "New" subtopic.</p> <p>Note: The options available in the New Menu for the current perspective can be configured through the Customize Perspectives option in the Window Menu.</p>
Open File		<p>Opens a previously created file in an editor view. If the file is stored in active projects in your workspace they will be displayed in PHP Explorer / Navigator views.</p>

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 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 updates 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. For more on renaming files using the rename feature, see " Renaming a file ".
Refresh	F5	Refreshes the Navigator views when external changes have been made.
Convert Line Delimiters To		Selects the preferred line ending style. Choices of Windows, Unix and Mac styles.
Print	Ctrl+P	Prints the active file.

<p>Switch Workspace</p>		<p>Allows you to open an alternate workspace. Using this feature will restart Zend Studio for Eclipse 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.</p>
<p>Import</p>		<p>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".</p>
<p>Export</p>		<p>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".</p>
<p>Properties</p>	<p>Alt+Enter</p>	<p>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.</p>
<p>Last viewed files</p>		<p>Lists the last viewed files for easy access.</p>
<p>Exit</p>		<p>Shuts down Zend Studio for Eclipse. You will be prompted to save any files to which unsaved changes have been made.</p>

New

The New submenu is available under File | New from the Menu Bar.

Note:

The options available in the New Menu for the current perspective can be configured through the [Customize Perspectives](#) option in the Window Menu.

The options available under the New submenu are:

Name	Description
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 for Eclipse, see " Zend Framework Integration ".
PHP Project	Creates a new project within the workspace, with PHP configuration settings allowing full PHP functionality.
Project	Adds a new Project to Navigator / PHP Explorer views in which files can be created.
Zend Controller	Creates a new file calling the Zend Controller class. This should be used within a Zend Framework Project. For more on Zend Framework, visit the Zend Framework site at http://framework.zend.com For more on using Zend Framework with Zend Studio for Eclipse, see " Zend Framework Integration ".
Zend Model	Creates a new Zend Model file. This should be used within a Zend Framework Project. For more on Zend Framework, visit the Zend Framework site at http://framework.zend.com For more on using Zend Framework with Zend Studio for Eclipse, see " Zend Framework Integration ".
Zend View	Creates a new Zend View file. This should be used within a Zend Framework Project. For more on Zend Framework, visit the Zend Framework site at http://framework.zend.com For more on using Zend Framework with Zend Studio for Eclipse, see " Zend Framework Integration ".
PHPUnit Test Case	Creates a new PHP Unit Test Case. Unit tests are a way of running tests on your code to ensure that the right output is being generated. Running unit tests can ensure that your code is stable and functioning correctly, and can help you to diagnose errors. For more on PHP Unit Testing, see " Using PHPUnit Testing ".

PHPUnit Test Suite	Creates a new PHP Unit Test Suite, containing several PHP Unit Test Cases (see above. For more on PHP Unit Test Suites, see " Creating and Running a PHPUnit Test Suite ").
PHP Class	Inserts a new PHP Class within existing or new files, including the required modifiers, Superclasses, interfaces, method stubs, comments etc.
PHP 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.
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.
CSS	Inserts a cascading style sheet into a project.
HTML	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.
Example	Creates the following example projects in your workspace: <ul style="list-style-type: none"> ▪ PHP-BIRT - For more on BIRT, see the BIRT Report Developer Guide. ▪ 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.

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:

- **Folder**
- **General** - File, Folder, Project, Untitled Text File
- **Business Intelligence and Reporting Tools** - Library, ODA Designer Plug-in Project, ODA Runtime Driver Plug-in Project, Report, Report Project, Template
- **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
- **Java** - Annotation, Class, Enum, Interface, Java Project, Java Project from Existing Ant Buildfile, Package, Source Folder, Scrapbook Page, JUnit Test Case, JUnit Test Suite
- **Java Emitter Templates - Convert Projects to JET Projects**
- **PHP** - PHP Class, PHP File, PHP Interface, PHP Project, Untitled PHP Document, Zend Controller, Zend Framework Project, Zend Model, Zend View, PHPUnit Test Case, PHPUnit Test Suite
- **Plug-in Development** - Extension Point Schema, Feature Patch, Feature Project, Fragment Project, Plug-in from existing JAR archives, Plug-in Project, Product Configuration, Target Definition, Update Site Project
- **Server** - Server
- **SQL Development** - SQL File
- **SVN** - Projects from SVN, Repository Location
- **User Assistance** - Cheat Sheet
- **Web** - CSS, HTML, JavaScript, Static Web Project
- **Web Services - WSDL**
- **XML** - DTD, XML, XML Schema
- **Examples** - PHP-BIRT Example Project, Editing and Validating XML Files, Zend Framework Example Project

Import

The Import submenu is available under [File | Import](#) from the Menu Bar.

Name	Description
General	<p>Archive File - Extracts files from the archive file into the workbench.</p> <p>Breakpoints - Creates a breakpoint working set.</p> <p>Existing Projects into Workspace -Projects from the local file system.</p> <p>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.</p> <p>Preferences - Import preferences from a preferences file on the local file system into the workbench.</p>
CVS	<p>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.</p> <p>A CVS repository connection needs to be configured before using this function. For more information, see "Working with CVS".</p>
PHP Profiler	<p>Profile Session - Imports a profile session file into the workspace.</p>
Plug-in Development	<p>Features - Imports features from your file system into your workspace.</p> <p>For more see "Feature Import".</p> <p>Plug-ins and Fragments - Imports plugins and fragments from your file system into your workspace.</p> <p>For more see "Plug-in Import".</p>
SVN	<p>Projects from SVN - Imports projects by connecting to an SVN repository.</p> <p>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.</p> <p>An SVN repository connection needs to be configured before using this function. For more information, see "Working with SVN".</p>
Team	<p>Team Project Set - Imports a description of the repository and version control information for a set of projects.</p> <p>For more on Team Project Sets, see 'Sharing your workspace setup using Project Sets'.</p>
Zend Imports	<p>Import from Zend Studio 5.x - Imports projects from Zend Studio 5.x</p> <p>Use this function if you have projects in your Zend Studio which you would like to move to Zend Studio for Eclipse.</p> <p>Do not keep Zend Studio open while importing Zend Studio files.</p>

Export


The Export submenu is available under [File | Export](#) from the Menu Bar.


Name	Description
General	<p>Ant Buildfiles - Generates Ant buildfiles based on the configuration of the Java projects.</p> <p>Archive File - Exports files from the Workbench to an archive file in the local file system.</p> <p>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.</p> <p>Note: The Breakpoint (.bkpt) file will not appear in Navigator / PHP Explorer views.</p> <p>File System - Export files from your workspace to your local file system.</p> <p>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.</p>
Java	<p>JAR File - Allows you to create a JAR file. See "JAR file exporter" for more.</p> <p>Javadoc - Generates a javadoc from your source code. See "Javadoc generation" for more.</p>
PHP	<p>PHP Doc - Creates a PHPDoc from your projects or files. See PHPDocs for more.</p> <p>WSDL File - Creates a WSDL File from a selection of classes and non-class functions in your workspace. See WSDL - Web Services Description Language for more information.</p>
PHP Profiler	<p>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 Tutorial.</p> <p>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 Tutorial.</p>
Plug-in Development	<p>Deployable features - Exports the selected features in a form suitable for deploying in an Eclipse product.</p> <p>Deployable plug-ins and fragments - Exports the selected plug-ins and/or fragments in a form suitable for deploying in an Eclipse product.</p> <p>Eclipse product - Exports an Eclipse product.</p>
Team	<p>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 Project Sets'.</p>

Edit

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	Ctrl+J	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.
Incremental Find Previous	Ctrl+Shift+J	Finds character strings before the cursor within the active file. To use this feature, press Ctrl+Shift+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...		<p>Inserts a task into your script.</p> <p>Tasks are used as reminders of work still needing to be done.</p> <p>For maximum effectiveness, tasks should be placed next to the section of code on which the action will be implemented.</p> <p>Open the tasks view ( Window Show View Tasks) to navigate between existing tasks.</p>
Show Tooltip Description		<p>Shows the value of a hover that would appear at the current cursor location. The dialog shown is scrollable and does not shorten descriptions.</p>
Word Completion	Alt+/ 	<p>Completes a word being typed. Enter the first few letters of the word and press Alt+/ to complete the word.</p> <p>Completes a prefix to a word occurring in all currently open editors or buffers.</p>
Quick Fix	Ctrl+1	<p>Displays possible quick fix options for problems in the Problems view.</p> <p>To use this option, first select a problem in the Problems view.</p> <p>Note: This option will not always be available.</p>
Set Encoding		<p>Changes the file encoding used to read and write the file in the active editor.</p>

Source

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
Toggle Comment	Ctrl+/ 	<p>Comments or uncomments a line by adding or removing "//" characters.</p> <p>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.</p> <p>To use this feature, select the line and press Ctrl+/ .</p> <p>For more, see "Commenting Code."</p>
Add Block Comment	Ctrl+Shift+/ 	<p>Comments a block by adding "/*" and "*/" characters to either side of the code.</p> <p>To use this feature, select the block and press Ctrl+Shift+/ .</p> <p>For more, see "Commenting Code."</p>
Remove Block Comment	Ctrl+Shift+\ 	<p>Removes a block comment.</p> <p>To use this feature, place the cursor anywhere within the comment and click Ctrl+Shift+\ .</p> <p>For more, see "Commenting Code."</p>
Format	Ctrl+Shift+F	<p>Auto formats a script to organize it into an easily readable</p>

Document		<p>format.</p> <p>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.</p> <p>You can configure your auto-formatting options through the Formatter Preferences page, accessible from Window Preferences PHP Formatter. For more, see Formatting Code.</p>
Format Active Elements	Ctrl+I	<p>Only formats selected code.</p> <p>To format active elements, select the required code to format and press Ctrl+I.</p> <p>Appropriate line breaks and indents will be added to the active elements.</p> <p>You can configure your auto-formatting options through the Formatter Preferences page, accessible from Window Preferences PHP Formatter. For more, see Formatting Code.</p>

Refactor

The Refactor menu allows you to edit names and locations of files and resources while maintaining the links between the files.

For more on refactoring, see "[Using Refactoring](#)".

The options available from the Refactor menu are:

Name	Shortcut	Description
Organize Imports	Ctrl+Shift+O	<p>The Organizing Imports feature will allow PHP objects created in one file to be called into other files within the project.</p> <p>Use this feature when referencing items in one file which have been created in another file. When referencing an item, click Ctrl+Shift+O. A dialog will be displayed showing that instances of the item have been found in another file, and will display the include calls which will be added in order to call this item into the active file.</p> <p>Press OK to add an include call for the item into your file.</p> <p>The Organize Imports feature can also delete references and include calls to files if the relevant items have been deleted, and move "include" calls to the correct location at the top of your script.</p>
Rename	Alt+Shift+R	<p>Renames a file or element.</p> <p>To rename a file, select it from the PHP Explorer view and press Alt+Shift+R.</p> <p>To rename an element within a file, highlight it in the editor view and press Alt+Shift+R.</p> <p>In the rename dialog, enter the new name and click Preview to see all changes that will be made as a result of the rename.</p>

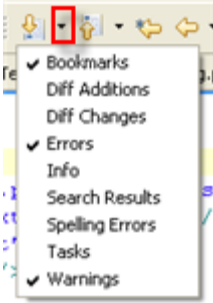
		Click OK to implement your changes. All references to the file / element will be automatically updated to reflect the change.
Move	Alt+Shift+V	<p>Moves a file to a different folder.</p> <p>To move a file, select it from the PHP Explorer view and press Alt+Shift+V.</p> <p>In the Move dialog, select the required folder and click Preview to see all changes that will be made as a result of the Move.</p> <p>Click OK to implement your changes and move the file.</p> <p>All references to the file will be automatically updated to reflect its new location.</p>

Navigate

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		<p>Goes into a selected folder so that only that folder's contents will be displayed in the Navigator view.</p> <p>Note: This will not work in PHP Explorer view.</p>
Go To		<p>Back</p> <p>Displays the previously displayed hierarchy in the Navigator view.</p>
		<p>Forward</p> <p>Returns to the display from which the back button was pressed in the Navigator view..</p>
		<p>Up one level</p> <p>Will go up one level in the hierarchy in the Navigator view..</p>
		<p>Resource</p> <p>Goes to a resource within the files and folders displayed in the Navigator view.</p> <p>Enter the first few letters of the a resource in the Go To Resource dialog, and select the required one from the list.</p> <p>Note: This functionality will not work in the PHP Explorer view.</p>
	Ctrl+Shift+P	<p>Matching Bracket</p> <p>Jumps to a bracket's pair.</p> <p>Clicking to the right of a bracket will highlight its matching pair. To jump to the matching bracket, press Ctrl+Shift+P.</p>
Open Declaration	F3	Goes to the declaration of an item selected in the editor.
Open PHP Element...	Alt+Shift+G	Navigates to declarations of classes, functions and constants in other files (including the Zend Framework library).

		<p>Choosing this option will cause an Open PHP Element dialog to be displayed. Enter the first few letters of your PHP Resource (classes, functions or constants) to see a list of possible elements.</p> <p>Select the required element and click OK.</p> <p>The file containing the element will be opened, and the element will be highlighted.</p>
Open Resource...	Ctrl+Shift+R	<p>Opens files within the same project as the active file.</p> <p>An Open Resource dialog will appear. Enter the first few letters of the required file to see a list of matching files.</p> <p>Select the required file and click OK to open it in an editor window.</p>
Show In	Alt+Shift+W	Navigator - Displays the current active file in the Navigator view.
		Outline - Displays an element in the Outline view.
Next / Previous Annotation	Ctrl+. Ctrl+,	<p>Goes to the next / previous annotation in the script.</p> <p>Possible annotations are: Bookmarks, Diff additions, Diff changes, Errors, Info, Search Results, Spelling Errors, Tasks and Warnings.</p> <p>Click the arrow next to the next / previous annotation icon on the toolbar to configure which types of annotations should be included.</p> 
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

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	<p>Opens the PHP Search dialog.</p> <p>PHP Search enables you to locate declarations of PHP Classes, Functions and Constants.</p> <p>For more, see "Searching for PHP Elements".</p>
File		<p>Opens the File Search dialog.</p> <p>File Search enables you to locate text in all files in your workspace.</p> <p>To run a File Search or Replace:</p> <ol style="list-style-type: none"> 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: <ul style="list-style-type: none"> • 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. <p>To replace the string, click Replace. The Replace dialog will open.</p> <p>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.</p>
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

The Project menu allows you to carry out different functions on your projects, including open, close, build and encode.

The options available from the Refactor menu are:

Name	Shortcut	Description
Open Project		Opens the previously closed currently selected project.
Close Project		<p>Closes the currently selected project.</p> <p>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.</p> <p>Closing projects takes up less memory and speeds up the build process.</p>
Build All	Ctrl+B	<p>This command manually invokes an incremental build on all projects in the Workbench.</p> <p>This is only available if automatic build is not selected (see below). For more information, see the "Builds" topic in the Workspace User Guide.</p> <p>Note: The build function can be configured by selecting General Workspace from the preferences dialog (Windows Preferences).</p>
Build Project		<p>This command manually invokes an incremental build on any resources in the currently selected project that have been affected since the last build.</p> <p>This is only available if automatic build is not selected (see below). For more information, see the "Builds" topic in the Workspace User Guide.</p> <p>Note: The build function can be configured by selecting General Workspace from the preferences dialog (Windows Preferences).</p>
Build Working Set		<p>This command manually invokes an incremental build on any resources in a working set that have been affected since the last build.</p> <p>This is only available if automatic build is not selected (see below). For more information, see the "Builds" topic in the Workspace User Guide.</p> <p>Note: The build function can be configured by selecting General Workspace from the preferences dialog (Windows Preferences).</p>
Clean...		<p>Invokes a clean build. This will discard all previous build results.</p> <p>For more information, see the "Builds" topic in the Workspace User Guide.</p> <p>Note: The build function can be configured by selecting General Workspace from the preferences dialog (Windows Preferences).</p>

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).
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. The PHPDoc can be created in your workspace, but will not be available from PHP Explorer or Navigator views.
Encode Project		Encodes the selected project using Zend Guard. This option is only available if Zend Guard is installed and configured in Zend Studio for Eclipse. Once you have installed Zend Guards, integrate it with Zend Studio for Eclipse by settings its location 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 project info, builders, code analyzer properties, formatter, includes mapping, PHP debug, PHP include path, PHP interpreter, PHP Java Bridge, PHP task tags, project references, task tags and validation.

Run

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 functionality.

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 on Debugging, see [Using the Debugger](#).

Profiling a file or application allows you to detect bottlenecks in scripts by locating problematic sections of code.

For more on Profiling, see [Using the Profiler](#).

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		Allows you to temporarily remove all breakpoints from your script so that the debugging process will not stop at them. Select this function 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.
Open Run Dialog...		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 ".
Open Debug Dialog...		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...		Launches the Profile dialog to create / edit Profiling

		configurations.
External Tools		<p>Run As If applicable, allows you to run the file using External tools.</p> <p>Open External Tools Dialog Opens the configuration dialog for running a file using external tools.</p> <p>Organize Favorites Opens a dialog allowing you to organize your external tools. See the Workbench User Guide for more on External Tools.</p>
Debug URL		Launches a Debug session for a specified URL.
Profile URL		Launches a Profile session for a specified URL.

Window

The Window menu allows you to customize your workspace display.

The options available from the Window menu are:

Name	Shortcut	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 Perspectives.
Show View		Displays a selected view. See the Workbench User Guide for more on Views.
Customize Perspective		Configures settings for the active perspective, including settings for: <ul style="list-style-type: none"> - 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 the Workbench User Guide for more on Working Sets.
Web Browser		Choose which web browser to use in Zend Studio for Eclipse. Options are: - Internal Web Browser (default) - Default system Web browser - Firefox - Internet Explorer
Preferences		Opens the preferences dialog to configure all aspects of the workspace. See the PHP Preferences section for on PHP Preferences .

Navigation

The Navigation submenu is available under Window | Navigation from the Menu Bar.

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, Close.)
Show View Menu	Ctrl+F10	Shows the active view's menus, containing functionality for each view.
Maximize / Minimize Active View or Editor	Ctrl+M	Toggles between full screen and minimal views.
Activate Editor	F12	Switches to the editor view.
Next Editor	Ctrl+F6	Switches to the next open editor. Hold down the Ctrl key and press F6 to scroll between the editors. This command is similar to the Alt+Tab functionality on Windows.
Previous Editor	Ctrl+Shift+F6	Switches to the previous open editor. Hold down the Ctrl+Shift keys and press F6 to scroll between the editors. This command is similar to the Alt+Tab functionality on 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.
Quick Switch Editor	Ctrl+E	Displays a list of open editors and allows you to select the required one from the list.

		Enter the first few letters of the name of the file open in the editor window to filter.
Next View	Ctrl+F7	Switches to the next open view. Hold down the Ctrl key and press F7 to scroll between the views. This command is similar to the Alt+Tab functionality on Windows.
Previous View	Ctrl+Shift+F7	Switches to the previous open view. Hold down the Ctrl key and press F7 to scroll between the views. This command is similar to the Alt+Tab functionality on Windows.

Help

The Help menu allows access to the most updated information on all aspects of Zend Studio for Eclipse, 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 for Eclipse 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 for Eclipse 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.
Software Updates		Find and install - Searches for available software updates. Manage Configuration - Opens a dialog allowing you to manage Zend Studio for Eclipse's configurations.
Register		Allows you to enter your Zend Studio for Eclipse license key to enable Zend Studio for Eclipse functionality. For more on purchasing a Zend Studio for Eclipse 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 for Eclipse. See i5 Edition Extras for more information.
Send Feedback		Opens the Zend feedback site to allow you to send feedback and comments on the product directly to Zend.
Tip of the Day		Opens the Tip of the Day window.
Protect your PHP Code!		Opens the Zend Guard site.
Speed up your PHP Site!		Opens the Zend Platform site.
About Zend Studio for Eclipse		Opens the About dialog, displaying information about the current version of Zend Studio for Eclipse.








PHP Perspective Main Toolbar





The PHP Perspective's Main Toolbar offers shortcuts to frequently used functionality:



Shortcut Icon	Shortcut Keys	Name	Description
		New	Clicking on the icon 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+B	Build All	Builds all projects.
	Ctrl+Alt+N	New Untitled PHP Document	Creates a new untitled PHP Document.
		Create New SQL Connection	Opens the New SQL Connection dialog.

		<p>Generate PHPDoc</p>	<p>Opens the PHPDoc Generation wizard.</p>
		<p>Debug</p>	<p>Clicking the Debug Button executes the last run configuration.</p> <p>Clicking the arrow next to the icon gives access to the following options:</p> <ul style="list-style-type: none"> ▪ 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. ▪ Open Debug dialog - 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.
		<p>Run</p>	<p>Clicking the Run Button executes the last run configuration.</p> <p>Clicking the arrow next to the icon gives access to the following options:</p> <ul style="list-style-type: none"> ▪ 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. ▪ Open Run dialog - 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.
		<p>Profile</p>	<p>Clicking the Profile Button executes the last run configuration.</p> <p>Clicking the arrow next to the icon gives access to the following options:</p> <ul style="list-style-type: none"> ▪ Profile a previously executed launch configuration.

			<ul style="list-style-type: none"> ▪ Profile As... - Profiles the active file as a PHP Script, PHP Web Page, or, when applicable, a PHPUnit Test. ▪ Open Profile dialog - 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	<p>Clicking the External Tools Button opens the External Tools Configuration dialog.</p> <p>Clicking the arrow next to the icon gives access to the following options:</p> <ul style="list-style-type: none"> ▪ 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. <p>See the Workbench User Guide for more on External Tools.</p>
		Debug URL	Launches the Debug URL dialog.
		Profile URL	Launches the Profile URL dialog.
	Alt+Shift+G	Open PHP Element	Launches the Open PHP Element dialog.
	Ctrl+H	Search	Launches the Search dialog.
	Ctrl+. Ctrl+,	Next/Previous Annotation	<p>Navigates to the next / previous annotation in the script.</p> <p>Possible annotations are: Bookmarks, Diff additions, Diff changes, Errors, Info, Search Results, Spelling Errors, Tasks and Warnings.</p> <p>Click the arrow next to the next / previous annotation icon on the toolbar to configure which types of annotations should be included.</p>
	Ctrl+Q	Last Edit Location	Jumps to the last location that was edited.

	Alt+Left Alt+Right	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 to enable a Tunneling connection for.
		Launch Platform Integration	Launches Platform Integration. Click the arrow next to the Platform Integration icon to select the server to launch Platform Integration for.
		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 set and configured.

To access this menu, go to Window menu and select Preferences | PHP.

Preference	Description
PHP	Configure the hierarchy display in PHP Explorer view.
Appearance	Configure the display of elements in Outline views.
Code Analyzer	Select the severity for error messages in different cases.
Code Coverage	Preview of code in Code Coverage view with the current color and font settings.
Code Gallery	Define and add code galleries.
Debug	Configure your debug preferences.
Installed Debuggers	
Workbench Options	Configure the workspace's behaviour when a debug session is launched.
Editor	Configure Smart Caret positioning.
Code Assist	Configure Code Assist preferences.
Folding	Configure the elements which will be folded by default.
Hovers	Configure the settings and shortcuts for the hover functionality.
Syntax Coloring	Set the font color for different elements.
Task Tags	Add and edit tasks tags.
Typing	Configure which items should be automatically completed.

Formatter	Set preferences for the auto-formatter.
Installed JREs	Add, remove or edit JRE definitions.
Path Variables	Add path variables.
PHP Executables	Add, remove or edit PHP executables definitions.
PHP Interpreter	Select the PHP version.
PHP Manual	Add PHP Manual sites.
PHP Servers	Add and edit PHP servers.
PHPUnit	Configure PHPUnit's Library path and port.
Profiler	Configure the workspace's behaviour when a profile session is launched.
Templates	Create, edit or remove templates.
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 behaviour.

The PHP Preferences page is accessed from Window | Preferences | PHP.

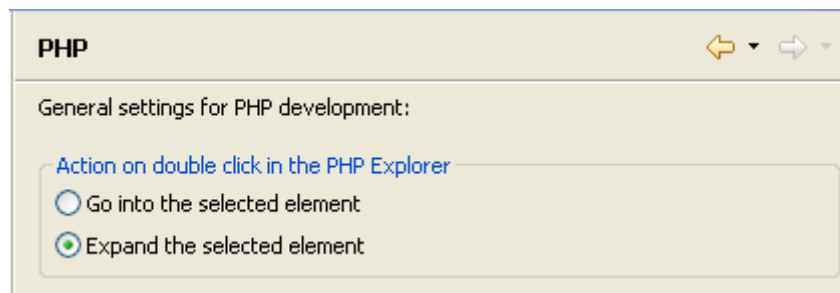


Figure 163 - PHP Preferences page

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.

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 views.

The Appearance Preferences page is accessed from Window | Preferences | PHP | Appearance.

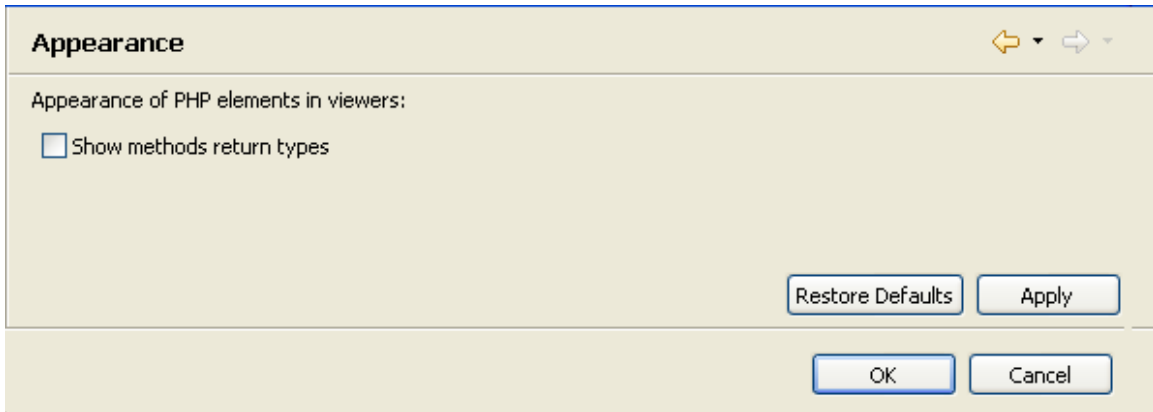


Figure 164 - Appearance Preferences page

To display the return type method for elements shown in the Outline and PHP Project Outline views (displayed in brackets next to the element), mark the 'Show methods return types checkbox.



Figure 165 - PHP Project Outline view without the method return types display

PHP Project Outline view with the method return types display

Click Apply to apply your settings.

Code Analyzer Preferences

The Code Analyzer enables warning and error messages to be displayed when Zend Studio for Eclipse detects possible errors or problems in your script.

See the [Code Analyzer topic](#) for more on the Code Analyzer.

The Code Analyzer Preferences page is accessed from Window | Preferences | PHP | Code Analyzer.

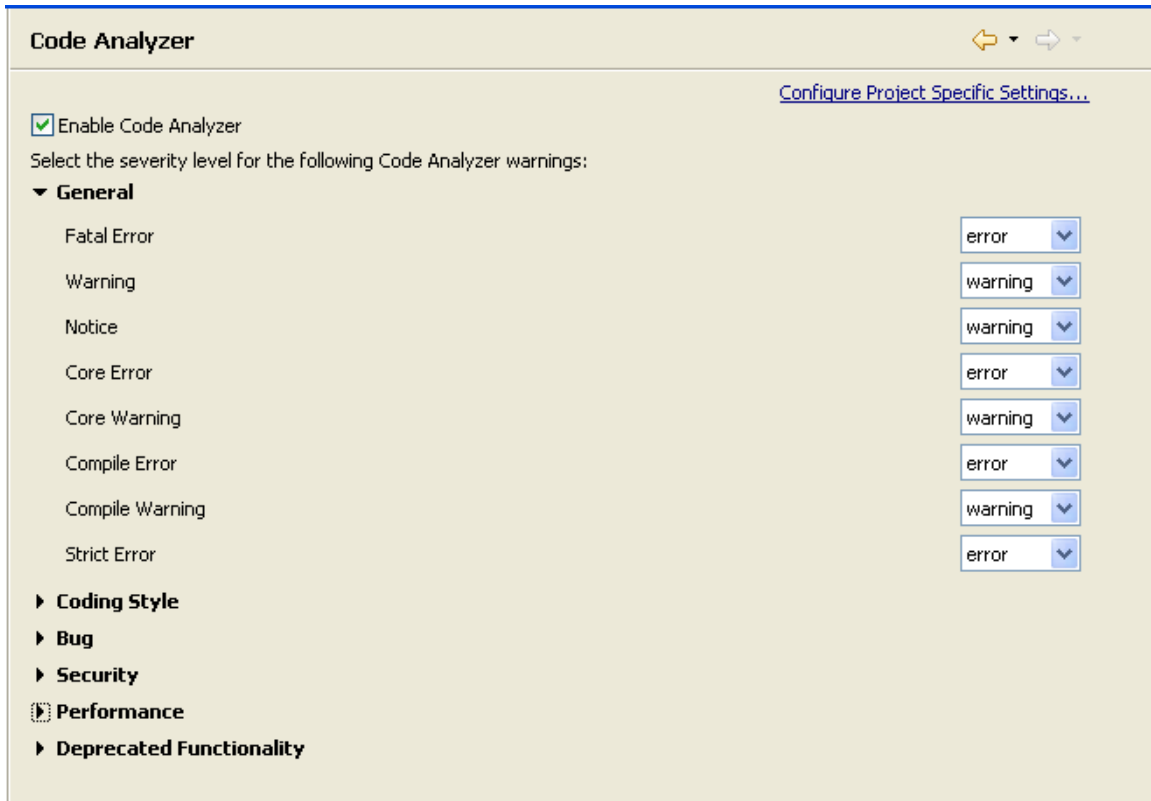


Figure 166 - Code Analyzer Preferences page

To enable Code Analyzer, mark the 'Enable Code Analyzer' checkbox.

The Code Analyzer preferences page allows you to select the severity level displayed in error messages (warning, ignore or error) for a variety of occurrences, divided into the following categories:


- General
- Coding Style
- Bug
- Security
- Performance
- Deprecated Functionality

Note:

The Enable Code Analyzer checkbox must be marked in order for the code analyzer configuration settings to be accessible.



To select a severity level for an event:

1. Click the  arrow next to each header.
2. Select the severity level from the drop-down list next to each option to apply it.
3. Click Apply to apply your settings.

Note:

Once Code Analyzer settings have been changed, a full rebuild will be required before the changes can take effect.



To apply Code Analyzer settings to a specific project only:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.
A Code Analyzer 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 for Eclipse is restarted.
-Or- click Cancel to cancel the operation.

Note:

Code Analyzer settings can also be configured for an existing project by right-clicking the project in PHP Explorer view and selecting Properties | Code Analyzer Properties.

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.

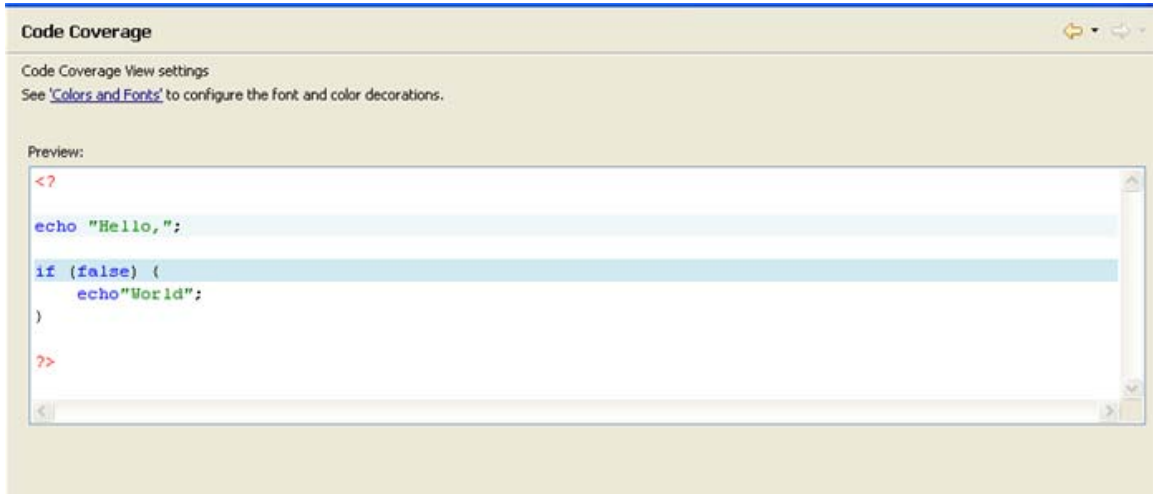


Figure 167 - Code Coverage Preferences page



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.

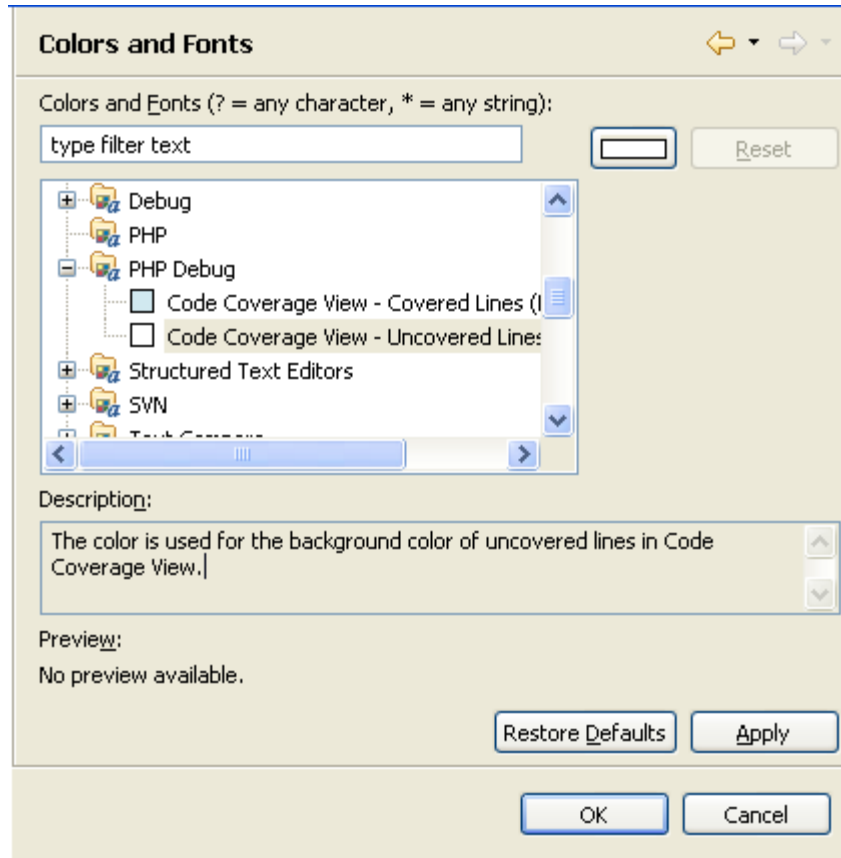


Figure 168 - Colors and Fonts Preferences page - PHP Debug

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.

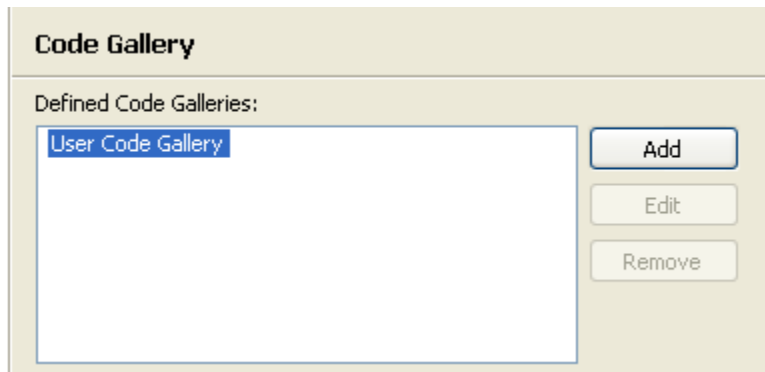


Figure 169 - Code Gallery preferences page

Code snippets can be accessed from the Code Gallery view (Window | Show View | Other | PHP Tools | Code Gallery).



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.



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.



To remove a code gallery site:

1. Select the required gallery from the list.
2. Click Remove.

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.

Click OK to apply your settings.

Debug Preferences

The Debug preferences page allows you to configure various preferences related to the debugging process.

The Debug Preferences page is accessed from Window | Preferences | PHP | Debug.

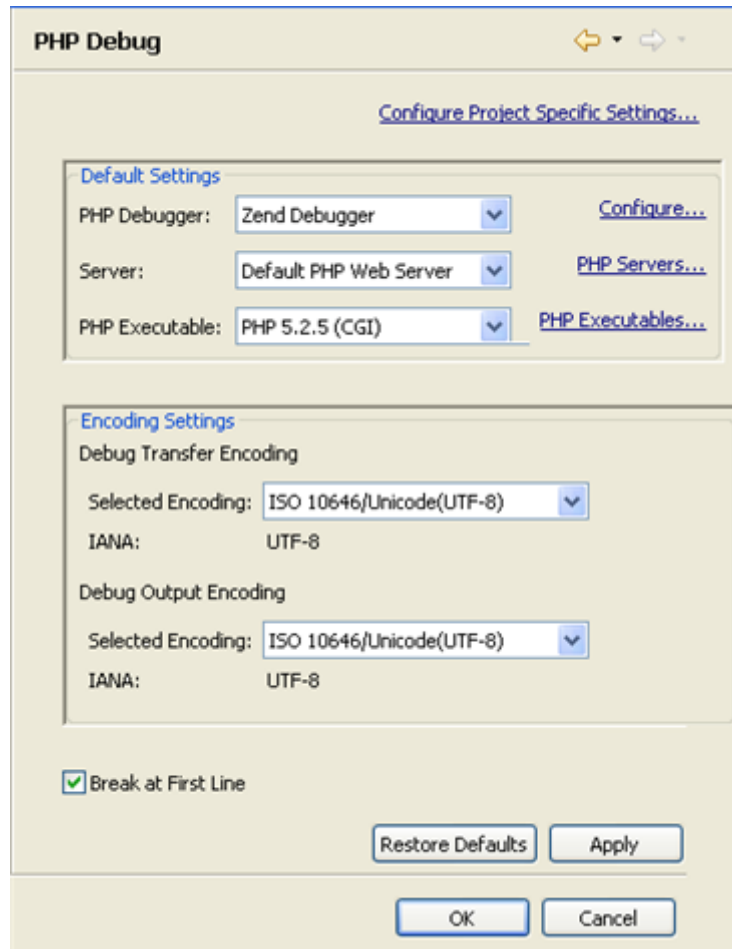


Figure 170 - Debug Preferences page

The settings that can be configured from the debug preferences page are:

Default Settings

- PHP Debugger - The default debugger is the Zend Debugger. Open the Installed Debuggers preferences page 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 [PHP Servers](#).
- 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 [PHP Executables](#).

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.

Click Apply to apply your settings.



To apply Debug settings to a specific project only:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.
A Debug 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 for Eclipse is restarted. Click Cancel to cancel the operation.

Installed Debuggers

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.

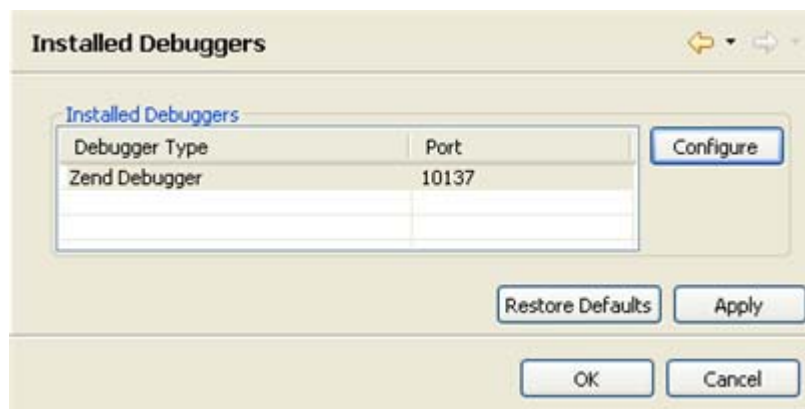


Figure 171 - Installed Debugger Preferences page



To configure your Zend Debugger settings:

1. Select the Zend Debugger.
2. Click Configure.
The Zend Debugger Settings dialog will open.

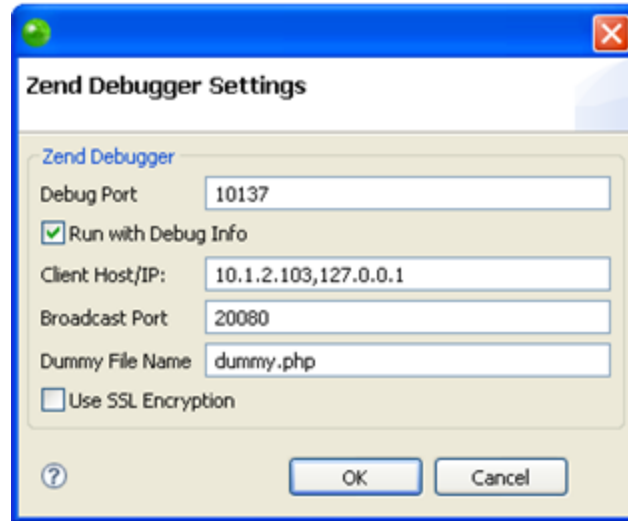


Figure 172 - Zend Debugger Settings dialog

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 for Eclipse 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.
 - Broadcast Port - The Broadcast Port allows your Zend Toolbar debugger or your Zend Platform to detect your debugging preferences. The Broadcast Port number entered here must match the Broadcast Port entered in your Zend Toolbar/Zend Platform. 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.
 - 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.

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.

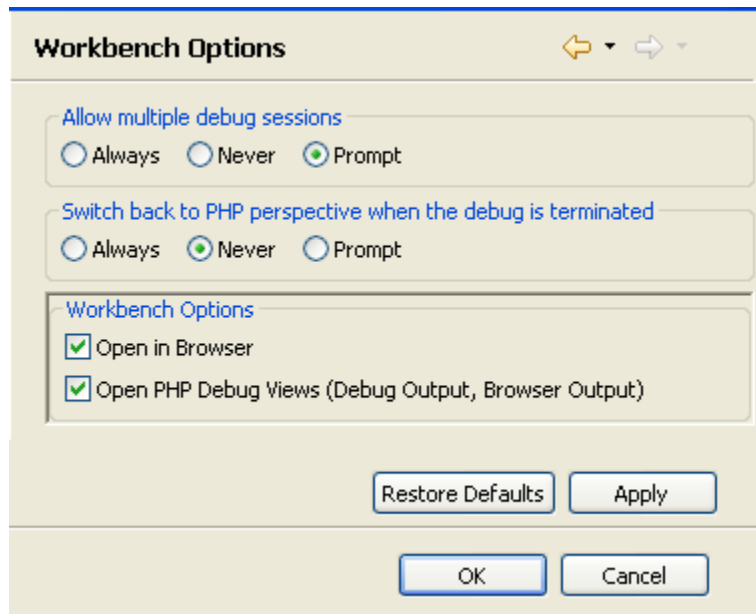


Figure 173 - Workbench Options Preferences page

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 Perspective 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 this checkbox 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 behaviour in the editor. Smart caret positioning determines where the cursor will jump to when the Home and End keys are pressed.

The Editor Preferences page is accessed from Window | Preferences | PHP | Editor.

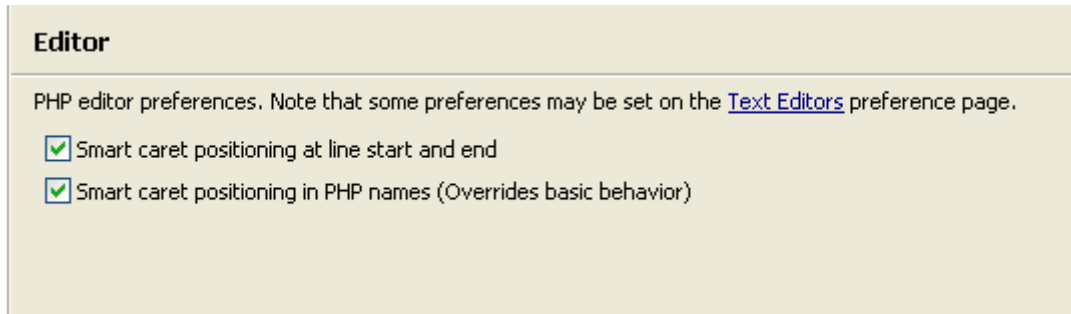


Figure 174 - Editor preferences page

Choose whether to enable:

- Smart caret positioning at line start and end - If this option is unchecked the cursor will jump to the beginning/end of a line, when Home/End are pressed. If it is checked, checked, 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 options is checked, pressing Home and End will jump to the beginning/end of a PHP name.

Click Apply to apply your settings.

Note:

More editor settings can be accessed by clicking the "Text Editors" link.

Code Assist Preferences

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

The Code Assist preferences page allows you to configure your Code Assist preferences.

The Code Assist Preferences page is accessed from Window | Preferences | PHP | Editor | Code Assist.

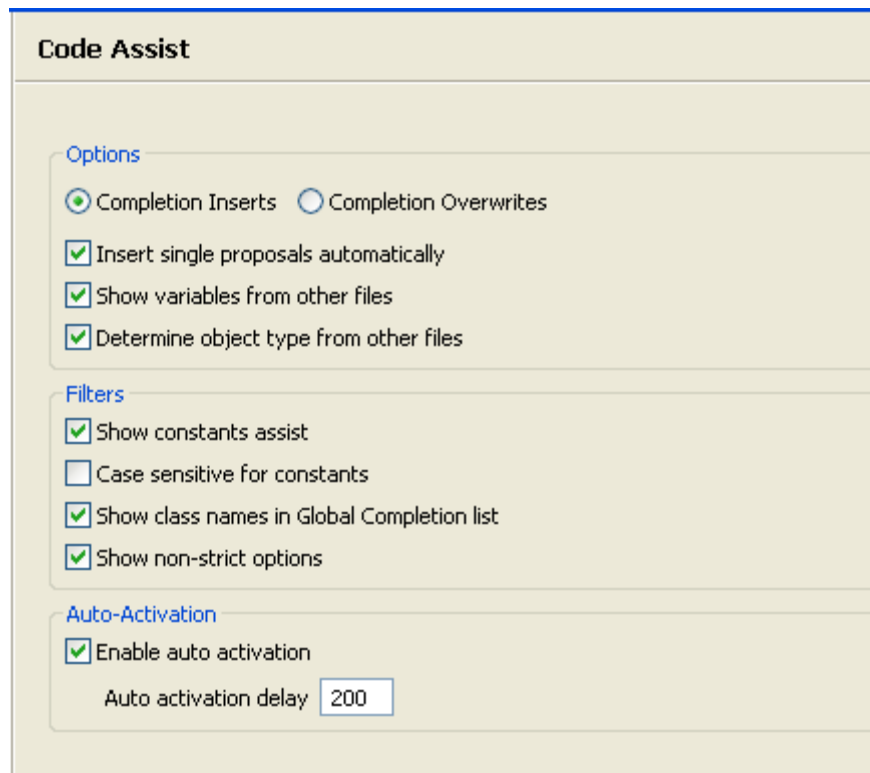


Figure 175 - Code Assist preferences page

Mark the required checkboxes to configure the following options:

Options

- Completion Inserts / Completion Overwrites: Select whether to insert new code or overwrite existing code.
- Insert single proposals automatically: When only one code assist suggestion exists, the code assist suggestion will be inserted automatically.
- Show variables from other files: Shows variables which are in other files in the project.
- Determine object type from other files: Shows objects from other files in the project.

Filters

- Show constants assist: Displays constants in Code Assist.
- Case sensitive for constants: Constants will be case sensitive.

- Show class name in Global Completion list: Enables the addition of class names to the code assist list.
- Show non-strict options : Will show options which go against PHP strict practice.
- Group completion options: Groups options according to element type.

Auto-activation

- Enable auto activation: Enable code assist to be automatically activated.
- Auto activation delay: Determines the delay before the Code Assist box is automatically displayed.

Note:

A maximum of 4 characters can be entered.

Click Apply to apply your settings.

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 without getting lost in a mass of complexity.

The Code Folding preferences page allows you enable / disable code folding and to select which elements should be folded by default.

The Folding Preferences page is accessed from Window | Preferences | PHP | Editor | Folding.

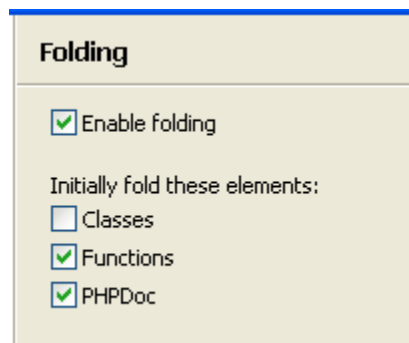


Figure 176 - Folding preferences page



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

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.

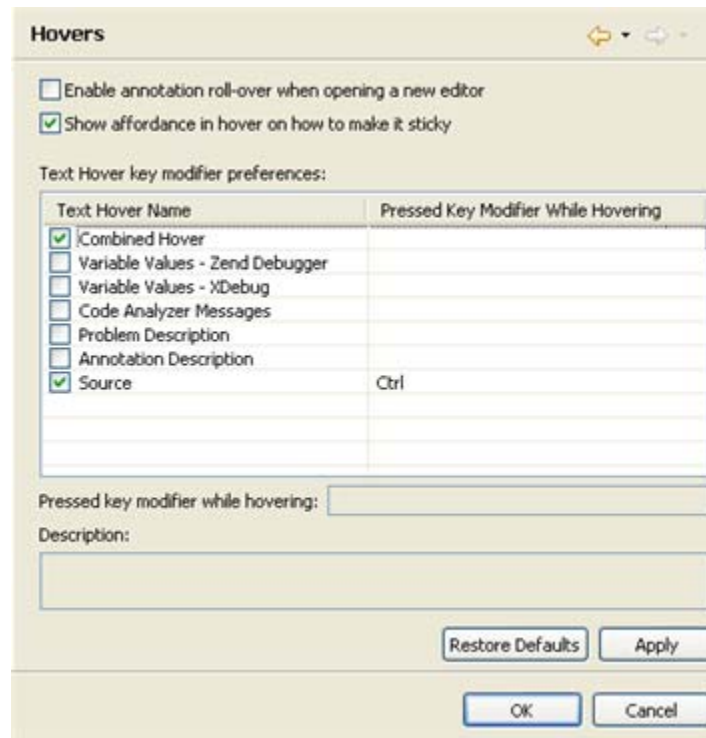


Figure 177 - Hovers preferences page

Select which of the following elements to enable by marking the relevant checkboxes:

- Enable annotation roll-over when opening a new editor
- Show affordance in hover on how to make it sticky

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 using the Zend Debugger.

- Variable Values - XDebug - Shows the value of the selected variable while debugging using XDebug.
- Code Analyzer Messages - Decorates problem hover with messages that come from [PHP Code Analyzer](#).
- Problem Description - Shows the description of the selected problem.
- Annotation Description - Shows the description of the selected annotation.
- Source - Shows the source of the selected element.



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.

Click Apply to apply your settings.

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 page is accessed from Window | Preferences | PHP | Editor | Syntax Coloring.

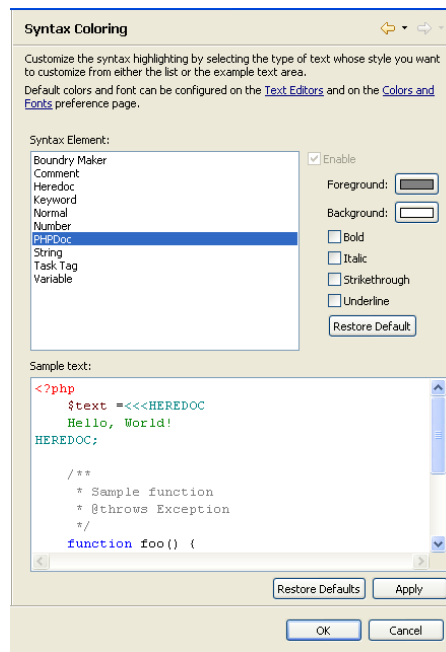


Figure 178 - Syntax Coloring preferences page



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 colour.
3. Select what formatting, if any, you would like to apply to the text (Bold, Italic, Strikethrough, Underline)
4. Click Apply to apply 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. See the Workbench User Guide for more on using tasks.

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.

The Task Tags Preferences page is accessed from Window | Preferences | PHP | Editor | Task Tags.

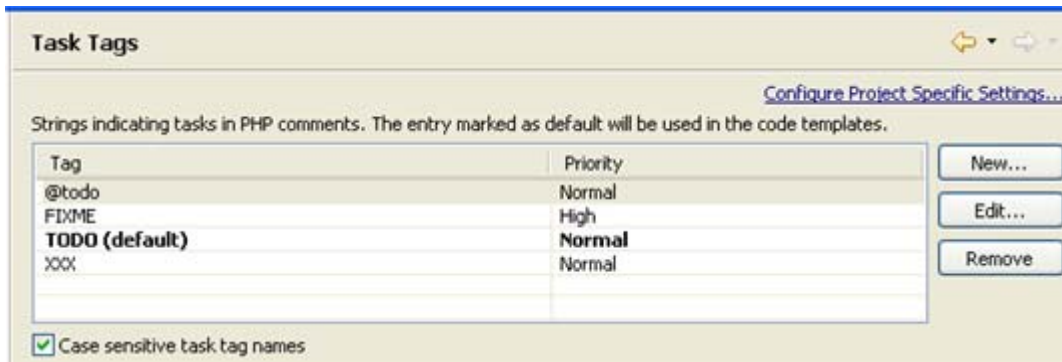


Figure 179 - Task Tags preferences page

Four common strings are included in the list by default.



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.

The new tag will be added to the list and will trigger a task when inserted in the editor.



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.

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 checkbox is marked, task tag names will be case sensitive.

Click Apply to apply your settings.



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.

Typing Preferences

The Typing preferences page allows you to configure which code and language patterns that Zend Studio for Eclipse 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.

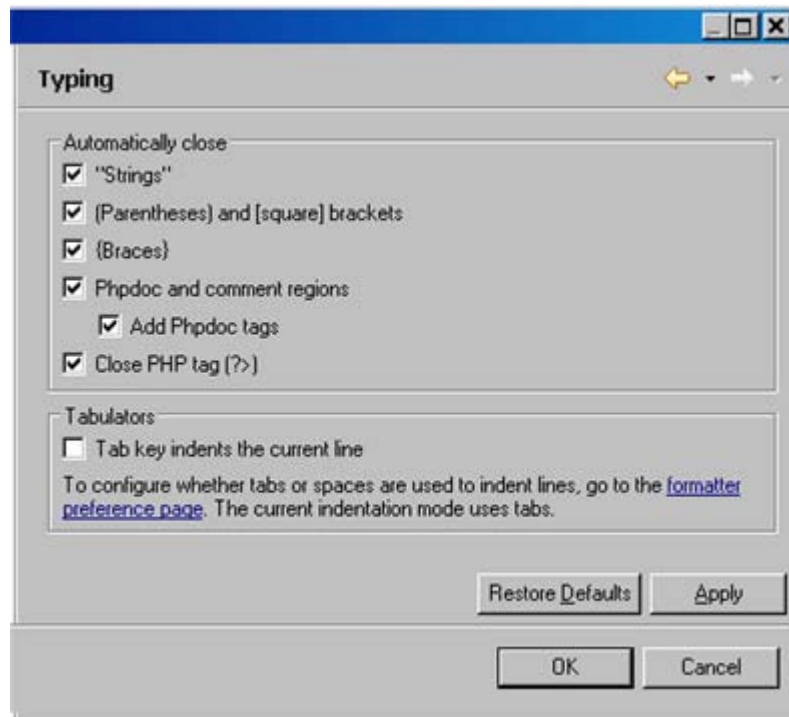


Figure 180 - Typing preferences page

Automatically Close

Zend Studio for Eclipse can be set to automatically complete the following types of patterns:

- "Strings" - A pair of double quotes ("") will be inserted when a single quotation markl (") 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 completes [PHPDoc](#) symbols for phpDoc Block comments.
- - Add Phpdoc tags - Adds Phpdoc tags
- Close PHP tag - A closing PHP tag (?>) will be inserted when the opening PHP tag (<?) is entered.

Mark the checkboxes of the patterns you would like Zend Studio for Eclipse to auto-complete.

To use the auto-complete function type the opening character in the editor. The matching character will be automatically inserted.

Tabulators

Mark the checkbox to be able to indent the selected line in the editor using the Tab Key.
For more on indentation preferences, see the [Formatter](#) Preferences page.

Click Apply to apply your settings.

Formatter Preferences

Zend Studio for Eclipse can auto format your scripts to organize it into an easily readable format.
The page allows you to customize the way it is formatted.

The Formatter Preferences page is accessed from Window | Preferences | PHP | Formatter.

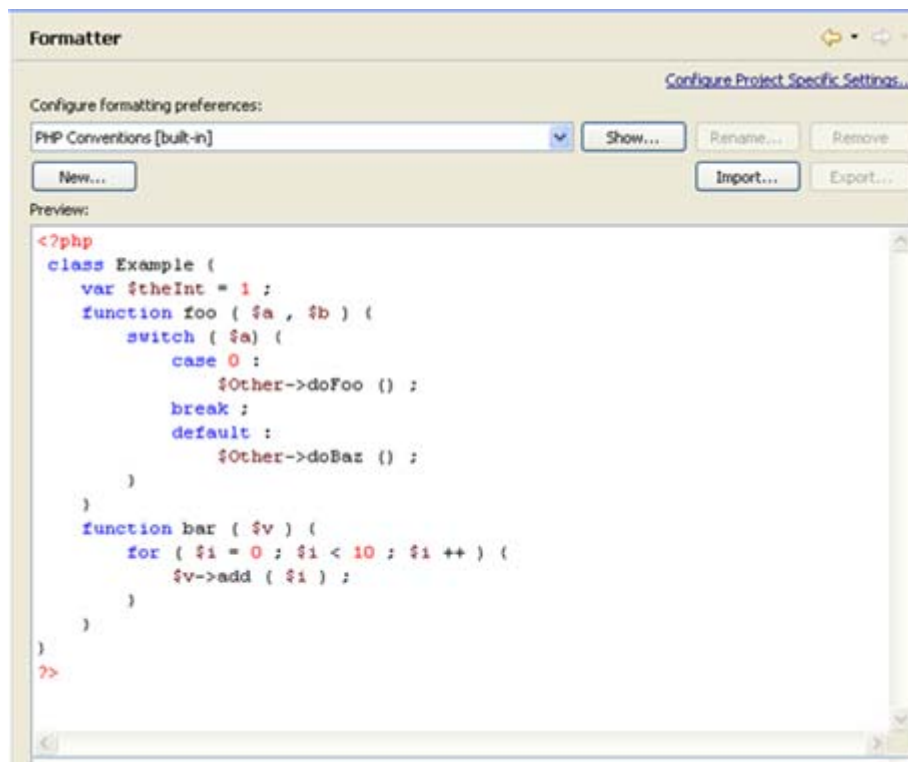


Figure 181 - Formatter preferences page

The default formatting settings are 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

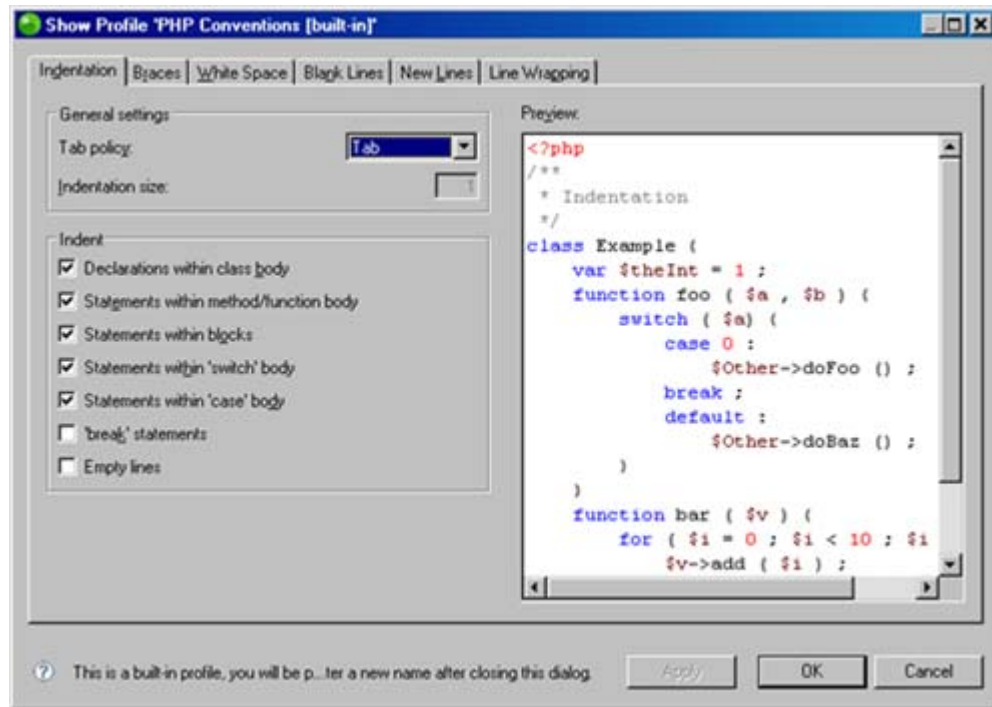


Figure 182 - Formatter - Indentation Tab

General Settings

Tab Policy - Select Tab or Spaces

If you select spaces, select the indentation size.

Indent

Select the elements you want indented.

Braces

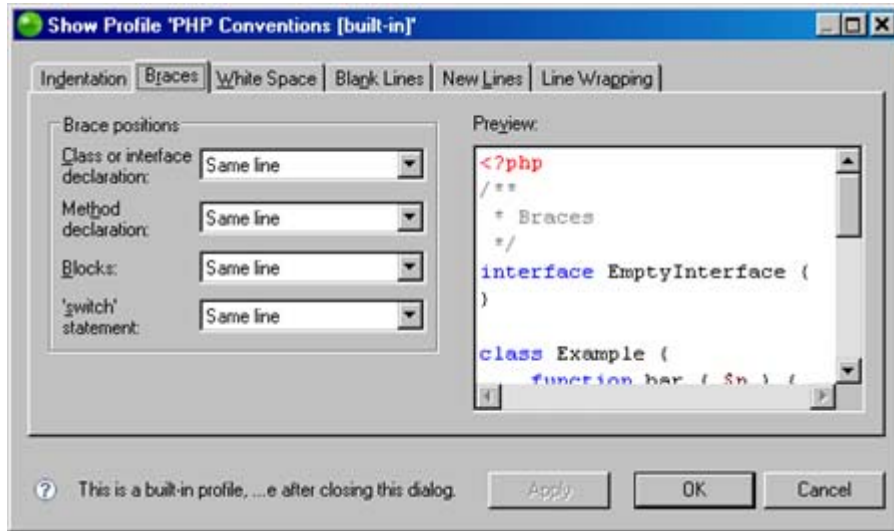


Figure 183 - Formatter - Braces Tab

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

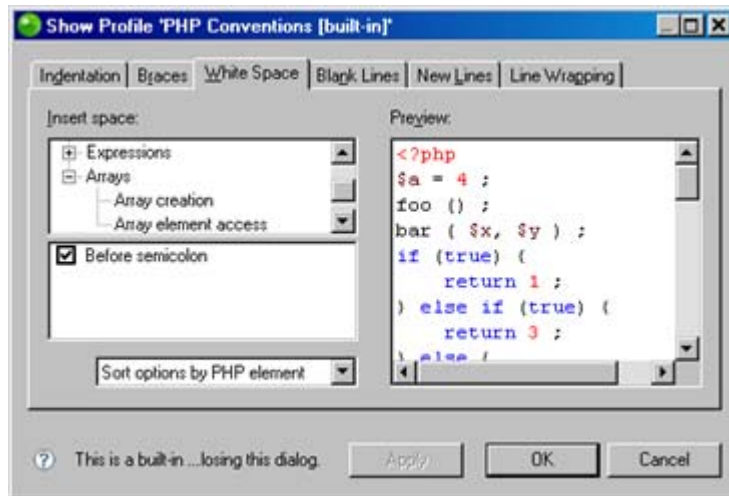


Figure 184 - Formatter - White Space tab

Configure where spaces should be entered for declarations, control statements, expressions and arrays.

Stand on each item to see the options for which syntax white spaces should be inserted.

Choosing 'sort options by Syntax element' from the drop down list will sort the list by syntax

options rather than by item.

Expand each category by clicking on the + sign to configure which items are applied to each instruction.

Blank Lines

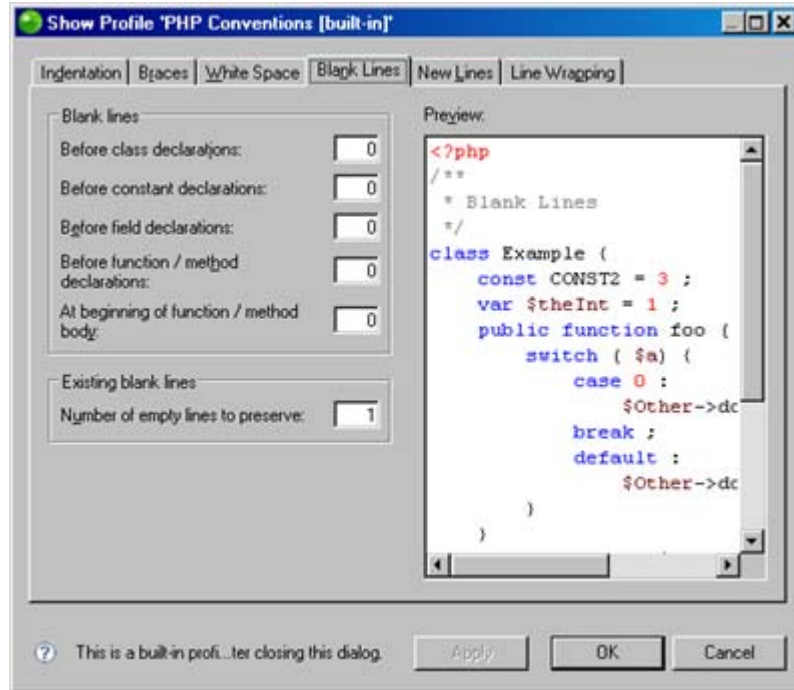


Figure 185 - Formatter - Blank Lines tab

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
- Number of existing blank lines to preserve

New Lines

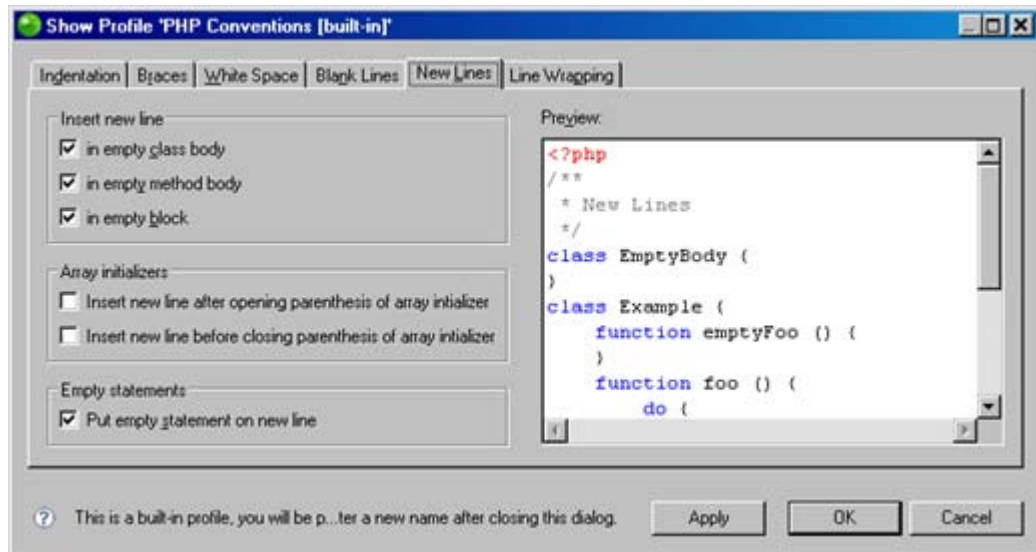


Figure 186 - Formatter - New Lines

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

Array Initializers

Select whether to:

- Insert new line after opening parenthesis of array initializer
- Insert new line before opening parenthesis of array initializer

Empty statements

Select whether to put empty statements on new line.

Line Wrapping

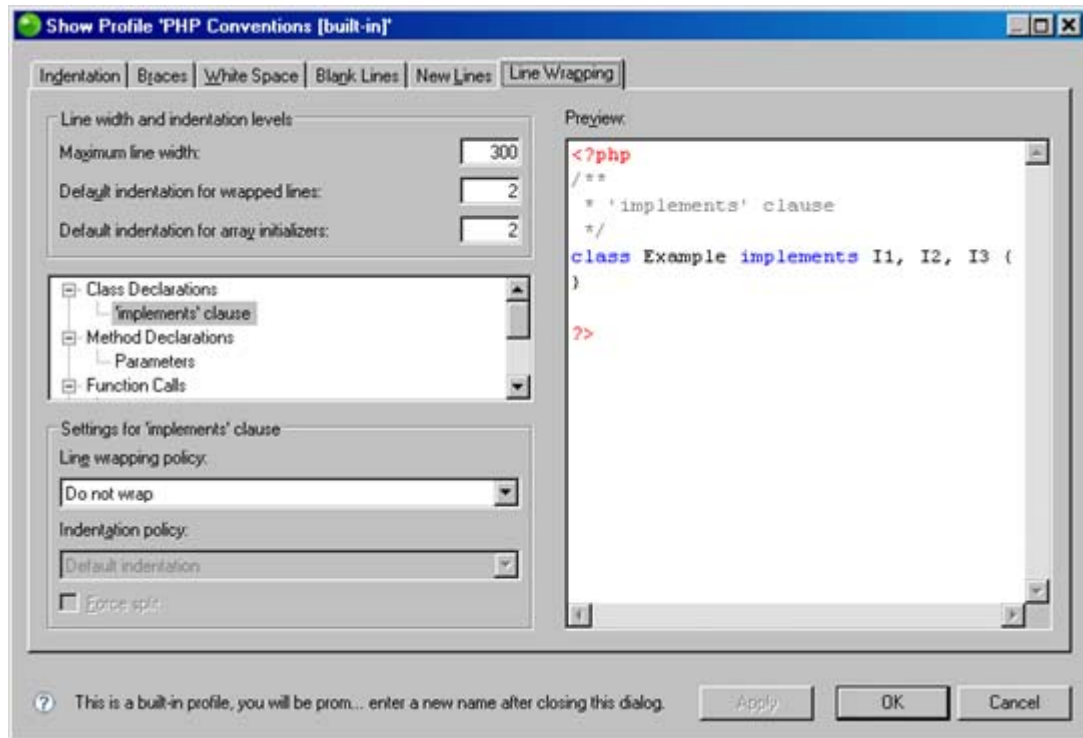


Figure 187 - Formatter - Line Wrapping

Line Width and Indentation Levels

Enter the:

- Maximum Line Width
- Default indentation for wrapped lines
- Default indentation for array initializers

Select the line wrapping and indentation policies for 'Class Declarations' ('implements' clause'), Method Declarations (parameters), Function Calls (Arguments, Object allocation arguments) and Expressions (Binary expressions, Array Initializers) by selecting the relevant option from the collapsible list and selecting an option from the 'Line wrapping policy' and 'Indentation policy' drop-down lists.

Once you have set the required options, click OK. The new configuration will be added to the list.



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.



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.



To apply Formatter 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 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 for Eclipse is restarted.
-Or- Click Cancel to cancel the operation.

Click OK to apply your settings.

Installed JREs Preferences

The Installed JREs Preferences page allows you to configure and add installed Java Runtime Environments.

The Installed JREs Preferences page is accessed from Window | Preferences | PHP | Installed JREs.

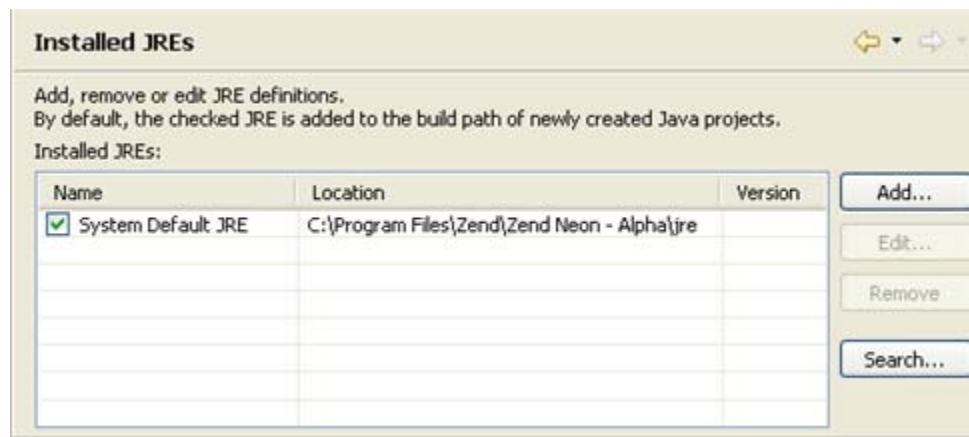


Figure 188 - Installed JREs Preferences page



To add a new JRE:

1. Click Add.
2. Enter the JRE Label and directory path from your local file network.
3. Click OK.

The new JRE will be added to the list.



To search for a JRE on your local file system:

1. Click Search.
2. Choose the directory in which it will search.
3. The Search process will detect any available JREs.

Click OK to apply your settings.

Path Variables Preferences

The Path Variables preferences page displays a list of your configured path variables and allows you to add or edit them.

A path variable is a name that is mapped to a specific location on the machine.

By using a path variable, you can share projects containing linked resources with team members without requiring the same directory structure as on your file system.

Include path variables can also be added to a project's include path.

The Path Variables Preferences page is accessed from Window | Preferences | PHP | Path Variables.

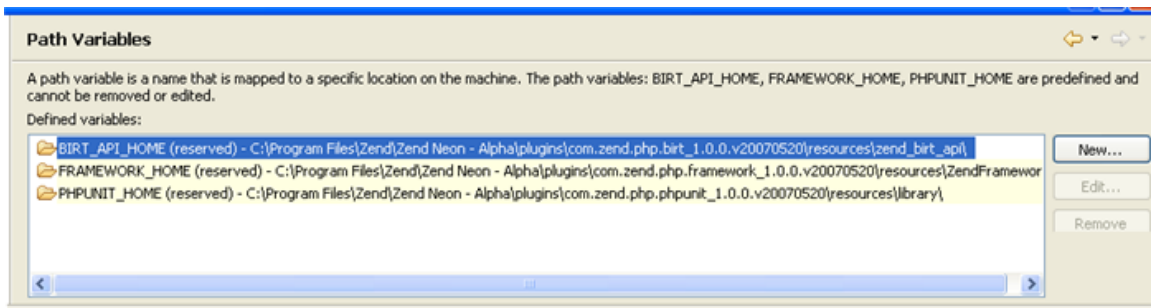


Figure 189 - Path Variables Preferences dialog



To add a new path variable:

1. Click New.
2. Enter the name of the path.
3. Enter the location to which the path should point.
4. Click OK.

The Path Variable will be added to the list.

Click OK to apply your settings.

PHP Executables Preferences

The PHP Executables Preferences page allows you to add, edit, remove and find PHP Executables.

The PHP Executables Preferences page is accessed from Window | Preferences | PHP | PHP Executables.

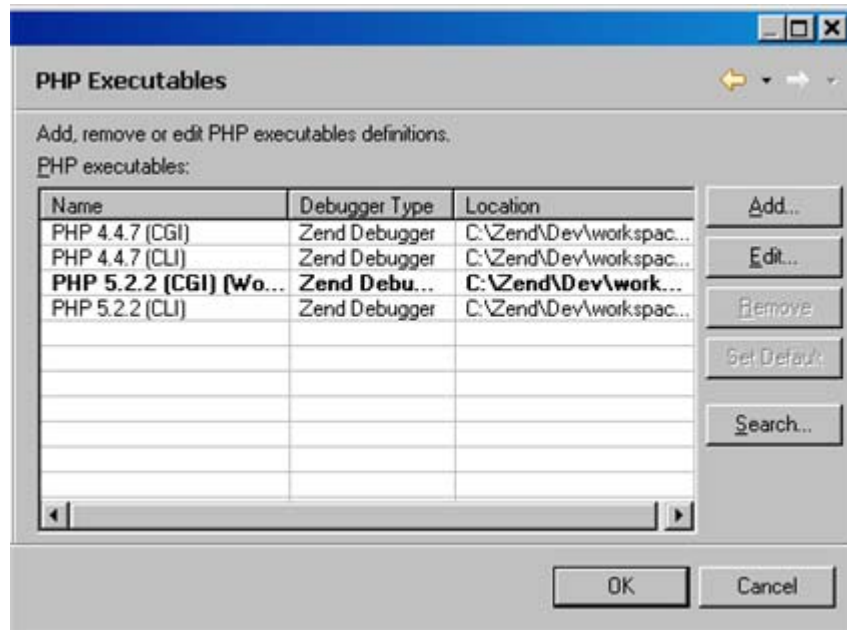


Figure 190 - PHP Executables



To add a PHP executable to the list:

1. Click Add.
An Add PHP executable dialog will appear.

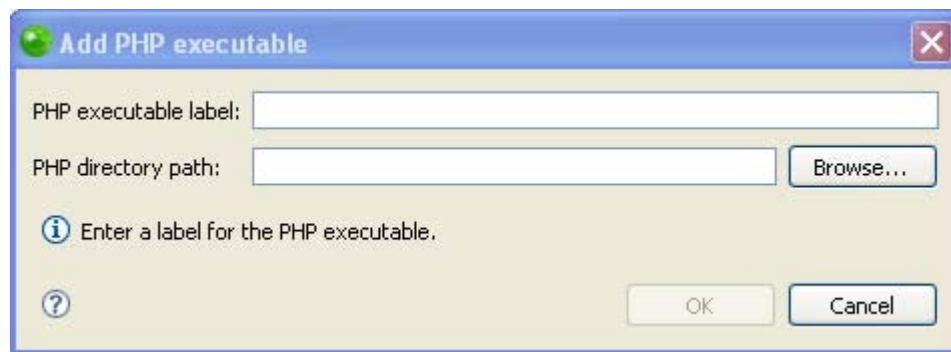


Figure 191 - Add PHP executable dialog

2. In the PHP executable label selection, enter the name of the executable.
3. In the PHP directory path selection, enter the location of the PHP executable on your file system.
4. Select the PHP Debugger.

5. Click OK.

The PHP executable will be added to your list.



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.
4. Zend Studio for Eclipse will search for PHP executables in the location specified.

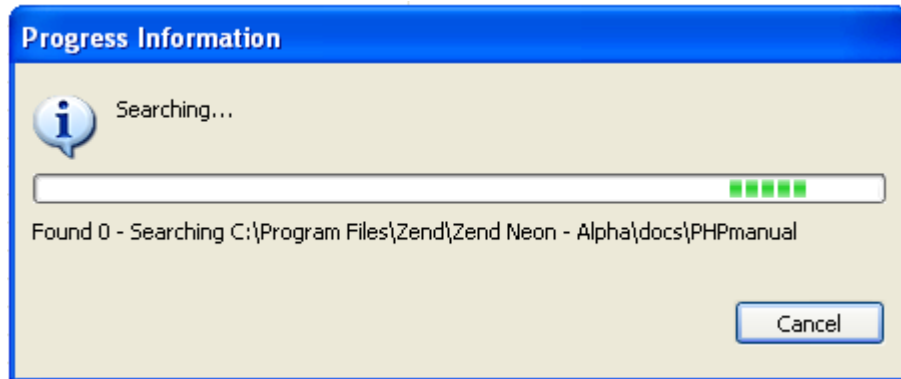


Figure 192 - Search Progress Information

5. If a PHP executable is found, it will be added to the list.
6. To change the name of the new PHP executable, select it from the list and click Edit.

Click OK to apply your settings.

PHP Interpreter Preferences

The PHP Interpreter preferences page allows you to set which PHP version to use.

The PHP Interpreter Preferences page is accessed from Window | Preferences | PHP | PHP Interpreter.

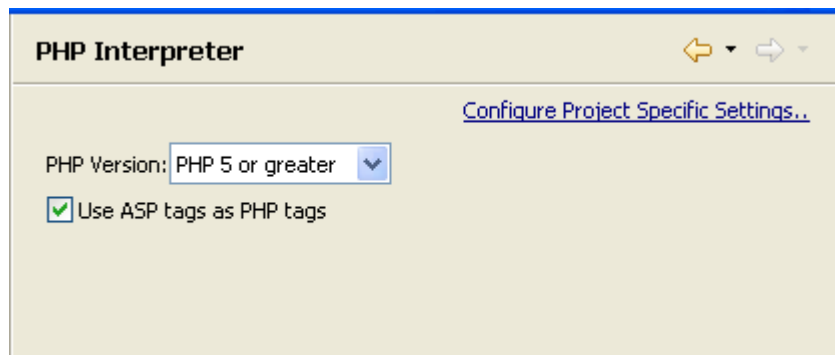


Figure 193 - PHP Interpreter Preferences page



To configure your PHP version:

1. Select the PHP Version to use. The default options are PHP 4 or PHP 5 or greater. The PHP version chosen will affect the internal debugger, code analyzer and code completion. See [PHP Support](#) for more on what PHP version settings affect.
2. Mark the "Use ASP tags as PHP tags" checkbox in order for Code Completion to respond to ASP tags in the same way as it responds to PHP tags.



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.



Figure 194 - PHP Interpreter Project Specific settings

4. Mark the Enable project specific settings checkbox.
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 done when Zend Studio for Eclipse is restarted.

Click OK to apply your settings.

PHP Manual Preferences

The PHP Manual preferences page sets the location of PHP Manuals and allows you to add, edit or remove manuals.

PHP Manuals contain an explanation of PHP functions. They can be accessed online or locally from within Zend Studio for Eclipse in order to provide an immediate explanation of the functionality and proper use of all PHP functions.

The PHP Manual Preferences page is accessed from Window | Preferences | PHP | PHP Manual.

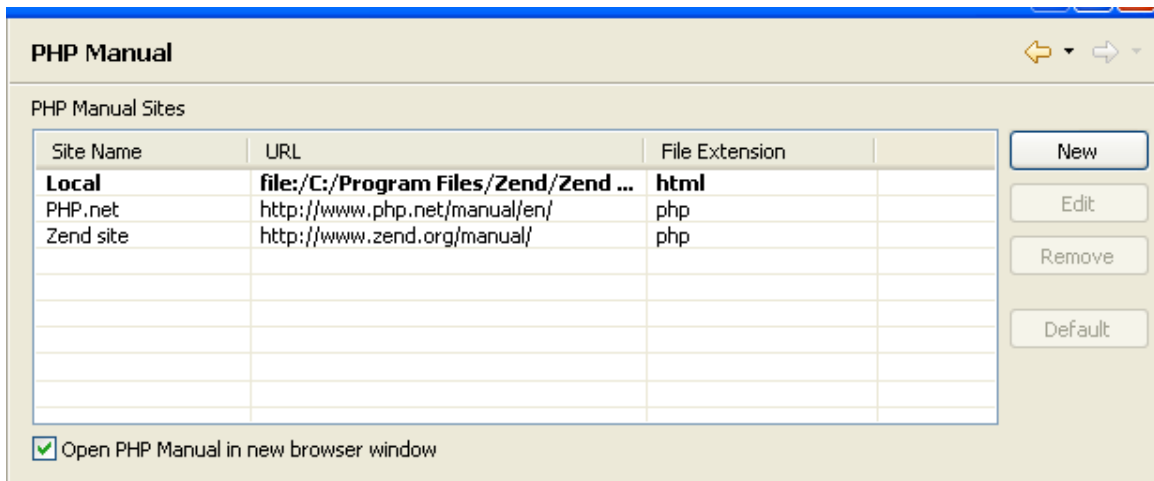


Figure 195 - PHP Manual preferences page



To add additional manuals to the list:

1. Click New.
2. Enter the Name of the site 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.

Zend Studio for Eclipse comes with its internal Default PHP Web Server, which will be used by default.

The PHP Servers Preferences page is accessed from Window | Preferences | PHP | PHP Servers.

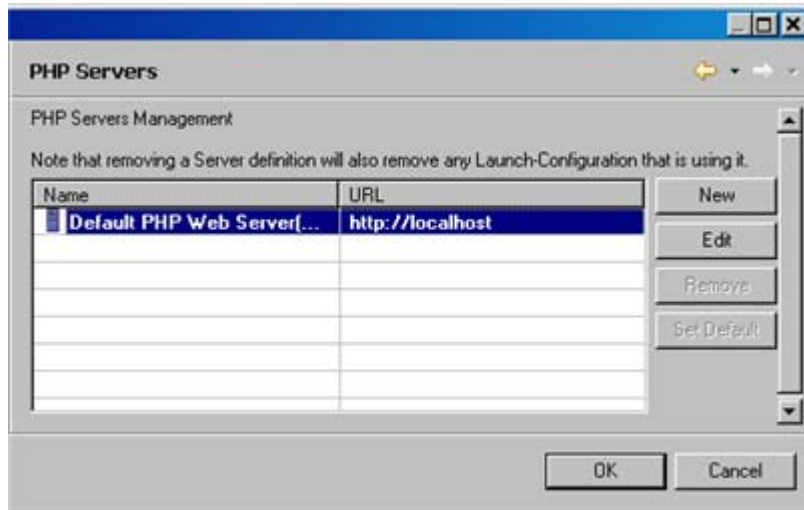


Figure 196 - PHP Servers



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.

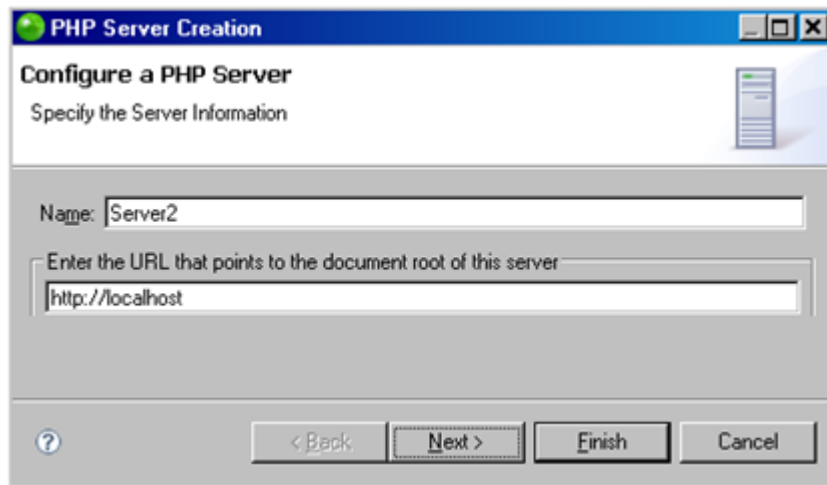


Figure 197 - PHP Server Creation page

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 [Adding a Server Location Path Map](#) for more information.
6. Click OK and Next.
7. If you would like to enable integration with Zend Platform, mark the 'Enable Platform Integration' checkbox and enter the relevant information:
 - Platform GUI
 - Authentication - Your Platform User Name and PasswordFor more, see [Defining a Platform Server](#).
8. Click Next.
9. If you would like to enable Tunneling, mark the 'Enable Tunneling' checkbox and enter the relevant information:
 - Specify Return Host
 - Automatically Connect on Startup
 - Send Authentication Information - User Name and Password.
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 Zend Platform integration.

Click OK to apply your settings.

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 that the right output is being generated.

For more on PHP Unit Testing, see the [PHPUnit Testing Tutorial](#).

The PHPUnit Preferences page is accessed from Window | Preferences | PHP | PHPUnit.

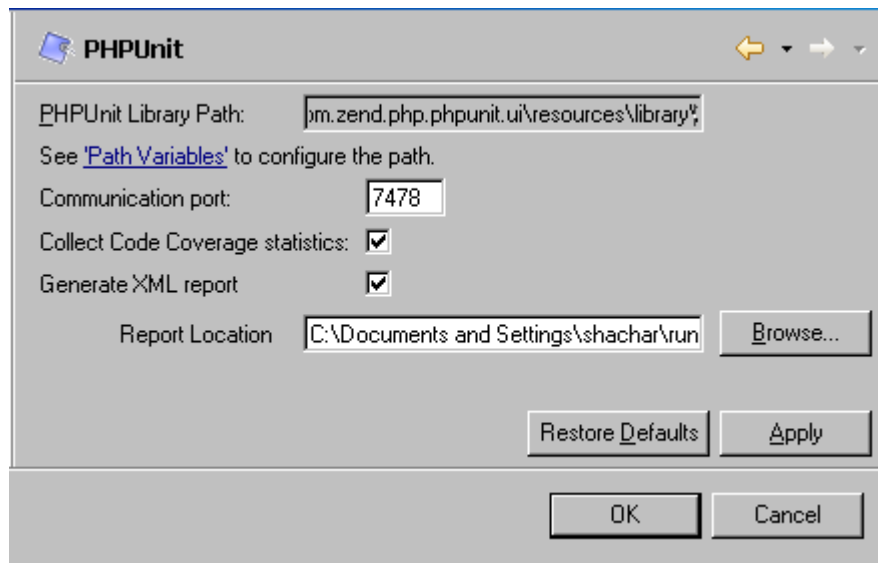


Figure 198 - PHPUnit Preferences page

The PHPUnit Library Path displays the location of your PHPUnit Library. This cannot be changed.



To configure your PHPUnit settings:

1. Change the communication port number, if required. Ensure the 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 'Generate XML Report' checkbox to enable XML Report generation.
4. XML Reports will be created in the location specified in the 'Report Location' option. Click Browse to change the location.

Click OK to apply your settings.

Profiler Preferences

The Profiler Preferences page allows you to set when the PHP Profile perspective will open. For more on running Profile sessions, see the Profiling tutorial.

The Profiler Preferences page is accessed from Window | Preferences | PHP | Profiler.

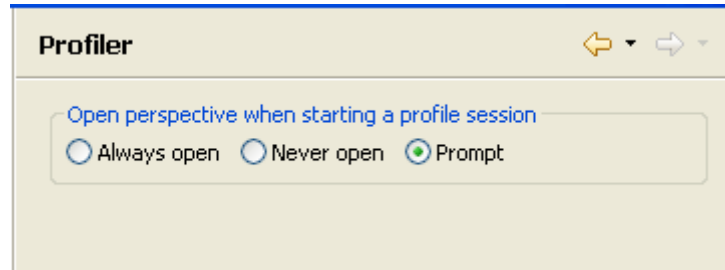


Figure 199 - Profiler preferences page

Mark the relevant checkboxes to configure the following options:

- Always Open - Opens the PHP Profile perspective whenever a profile session is run.
- Never open - The PHP Profile perspective will not open automatically when a profiling session is run. You can manually open the PHP Profile perspective by going to Window menu and selecting Open Perspective | PHP Profile.
- Prompt - The following prompt will appear when a profiling session is run:



Figure 200 - Profiler Perspective Prompt Dialog


From the Confirm Open Perspective dialog, you can click Yes to open the PHP Profile perspective or No to keep your current perspective active. You can also mark the 'Remember my decision' checkbox so that your decision will always be implemented.

Click OK to apply your settings.

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 Code Assist function, and select the template from the list. Templates are marked by a  icon. For more on using the code assist function, see '[Working with Code Assist](#)'.

The Templates Preferences page is accessed from Window | Preferences | PHP | Templates.

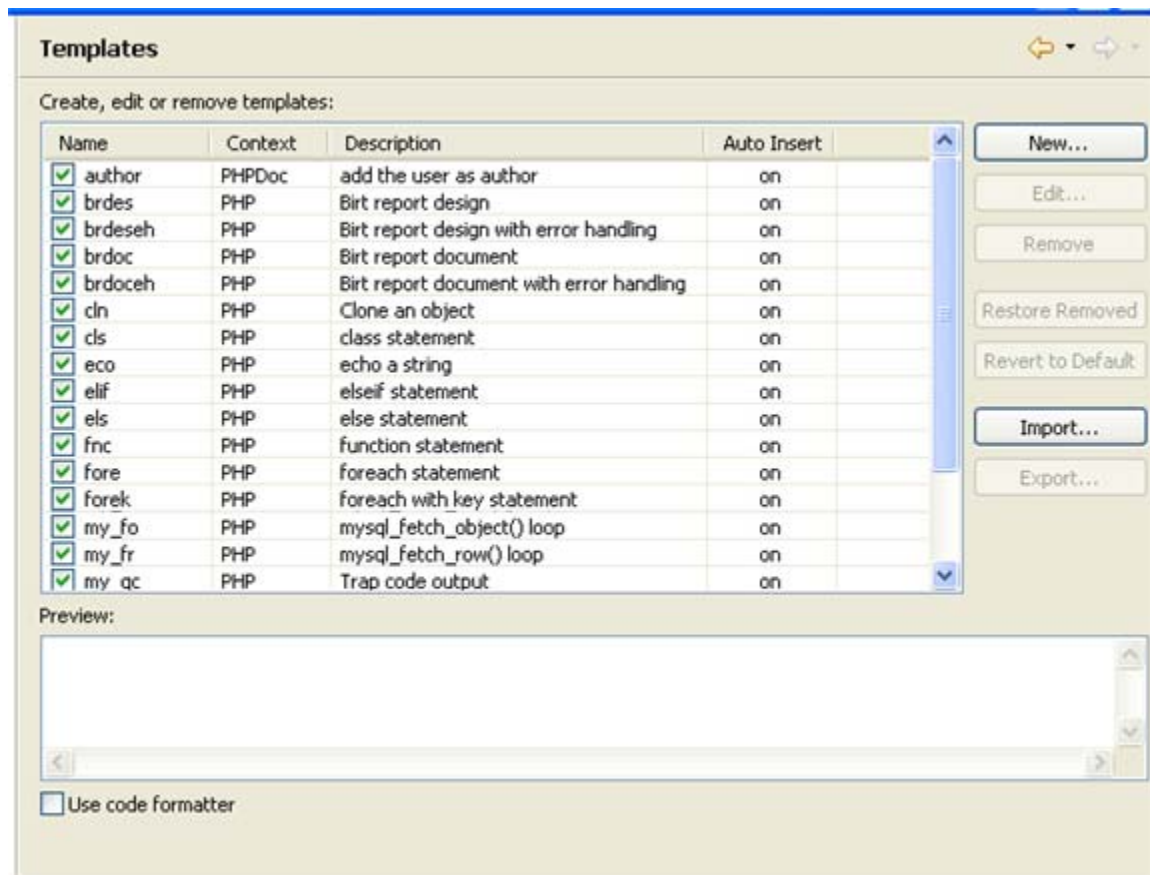


Figure 201 - Template preferences page

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.

Creating a new Template

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.

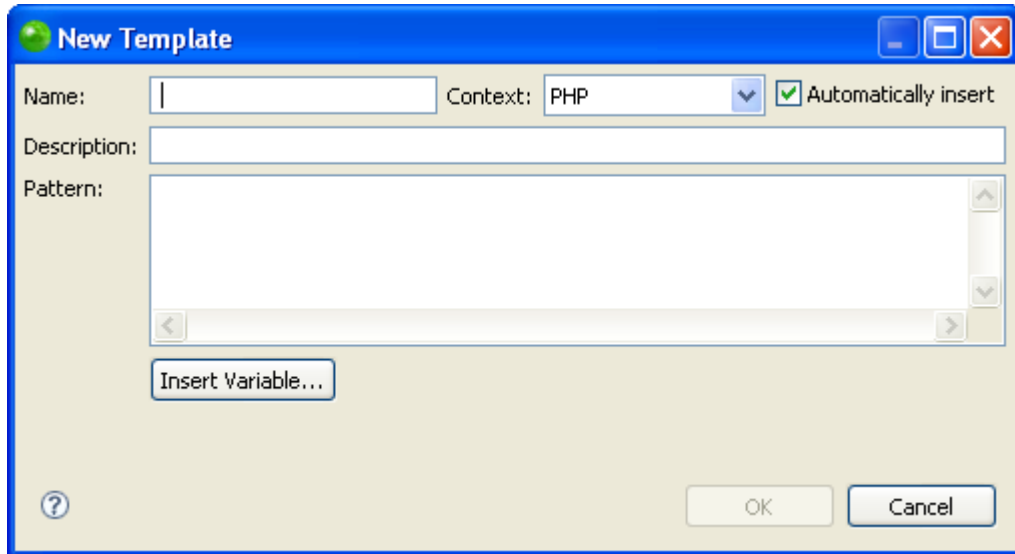


Figure 202 - New Template Dialog

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 code assist in the relevant context.

Exporting and Importing Templates

Zend Studio for Eclipse 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 and click Open.

The templates contained in the template.xml file will be imported into the list of Templates.

Note:

An imported template with the same name as a pre-existing template will overwrite (replace) the pre-existing one. In order to prevent a Template from being overwritten, you must rename the existing Template or edit the Template XML file to change the import name of the template.

**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>`

Click OK to apply your settings.

Zend Guard Preferences

Zend Guard protects your commercial PHP 4 and PHP 5 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 for Eclipse.

The Zend Guard Preferences page is accessed from Window | Preferences | PHP | Zend Guard.

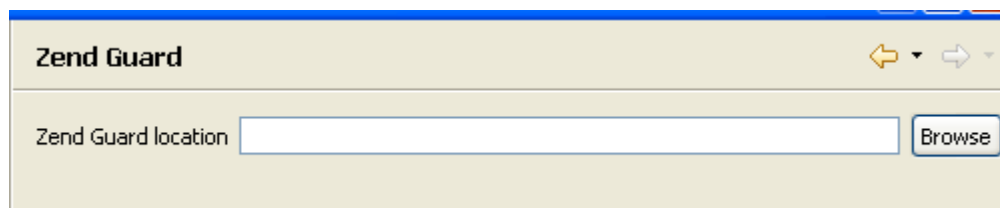


Figure 203 - Zend Guard preferences page




To configure your Zend Guard's location:

1. Click Browse.
2. Find the location of your Zend Guard installation and click Open.

The connection will be made between Zend Studio for Eclipse and Zend Guard.

Click OK to apply your settings.

Zend Guard's functionality can now be accessed from Zend Studio for Eclipse 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>.

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.

File		Refactor		Source	
Ctrl+S	Save	Alt+Shift+V	Move	Ctrl+Shift+/ Ctrl+Shift+\	Add/Remove Block Comment
Ctrl+F4 / Ctrl+W	Close	Alt+Shift+R	Rename	Ctrl+Shift +F	Format Script
Ctrl+P	Print	Alt+Shift+O	Organize Imports	Ctrl+I	Format Active Elements
Alt+Shift+N / Ctrl+N	New (Menu)	Run/Debug		Navigate	
Alt+Enter	Properties	F11	Debug	Alt+Left Alt+Right	Backward / Forward Editor History
F5	Refresh	Alt+Shift+D, H	Debug PHP Script	Ctrl+PgUp/ Ctrl+PgDown	Navigate between open editors
Ctrl+Shift+S	Save All	Alt+Shift+D, U	Debug PHPUnit Test	Ctrl+Shift+P	Go to Matching Bracket
Edit		Alt+Shift+D, W	Debug PHP Web Page	Ctrl+L	Go to Line
Ctrl+Shift+Q	Quick Diff Toggle	Ctrl+U	Execute	Alt+Shift+G	Open PHP Element
Ctrl+Shift+C	Toggle Comment	Ctrl+F11	Run	Ctrl+Shift+R	Open Resource
Ctrl+Shift+A	Find in file	Alt+Shift+X, H	Run PHP Script	Window	
Ctrl+Space	Open Code Assist List	Alt+Shift+X, U	Run PHPUnit Test	Ctrl+M	Maximize Active View or Editor
Ctrl+Shift+Space	Context Information	Alt+Shift+X, W	Run PHP Web Page	Ctrl+F10	Show View Menu
Ctrl+J/ Ctrl+Shift+J	Incremental Find Next / Previous	Ctrl+Shift+B	Toggle Breakpoint	WYSIWYG Editor	
Alt-/	Word Completion	Views		F4	Show / Hide Tabbed Properties View
F2	Show Tooltip	Alt+Shift+Q, Q	Show View	F6	Switch Source / Design

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/forums>
- PHP Certification - <http://www.zend.com/en/services/certification>
- PHP Yellow Pages - <http://www.zend.com/store/education/certification/yellow-pages.php>
- Support Center - <http://www.zend.com/en/support-center>
- Knowledge Base Search - <http://www.zend.com/support/knowledgebase.php>
- Refactoring Information - <http://www.refactoring.com>
- HTML Tutorials - <http://www.w3schools.com/html>

Index

(
(check out.....	204
5	
5250 Bridge	98
A	
Absolute	94
Activating Tunneling	202
Add CVS Repository	53
Adding an SVN Repository.....	60
Allowed Host	198
Analyzer	85
Appearance preferences.....	283
Auto Detection Port	201
Automatic Type Conversion	79
Average Own Time	40
B	
Body	151
Bookmarks.....	72
Bookmarks view	72
bottlenecks	40
Boundry Maker	71
bracket.....	70
brackets	70
breakpoints.....	30, 90, 244
Breakpoints view	244
Browser Output view	248
C	
called	129
Called Parameters.....	246
Calls Count	40
Check Out.....	204
Checking Out Projects	53, 60
Class Type Hints	25
Code Analyzer	85
Code Analyzer Preferences.....	284
Code Assist	25, 68, 108
Code Assist preferences.....	294
Code Coverage Preferences.....	286
Code Coverage Summary.....	40
Code Coverage Summary View	255
code elements	108
Code Folding	70, 117
Code Gallery	76, 131, 132, 134
Code Gallery preferences	288
Code snippet	76, 131, 132, 134
colors.....	71
comment	118
Commenting	72
Commenting Code	72
Comments	72
communication port	95
Communication Settings.....	95
communication tunnel.....	201
compare	210
Concurrent Versions System	53, 95
Configure Include Paths.....	194
Connecting to a Database	162
Connection Profile.....	160
Controller	77, 138
Cookie.....	201
Covered Lines	40
CSS Editing.....	155
CSS Palette Editor.....	82
CSS Preview	82
CSS selector	155
CSS Styles.....	155
Current Working Directory	94
CVS	53, 95, 203
CVS connection	204, 205
CVS perspective	203
CVS Repository	53, 203, 204, 205
CVS Repository Exploring	203
CVS Repository Exploring Perspective.....	95, 204
D	
Data Source Explorer	162
Data Source Explorer view	158
Data Tools Platform	84, 160
database	84
Database Development Perspective.....	84, 158, 162, 163
Debug Output view	247
Debug view.....	241
debugger	90
Debugger Toolbar	91

Debugging	30, 90, 241	Function invocation	79
Debugging PHP Scripts	30	Function Parameter Hint	25
Debugging PHP Web Page	30	G	
Debugging URLs	30	Get content	92
Design	82	getters and setters	113
DocBlock	119	Goto Source	123
Duration Time	40	Guard	96, 221
E		Guard Preferences	319
Easy File Creation	102	H	
Easy PHP File	103	Head	150, 151
Electronic Licensing solution	96	HereDoc	71
Enable Project Settings	66	Hover Support	74
encode	218	Hovering	74
encoding	96	Hovers Preferences	296
errors	84	HTML	82
event monitoring	85	HTML editing	149
Events	164	HTML file	149
Execution Flow view	254	HTML objects	150
Execution Statistics	40	HTML properties	151
Execution Statistics View	253	I	
Execution Time	40	i5 Edition	98
external file	103	i5 Edition license	98
external Java Archives	142	i5 PHP API Toolkit	98
external projects	94	Image Preview	82
F		Import	105
Failure Trace	46	Include Path	94, 194
file	102, 103	Include Paths	92
Filter Stack Trace	46	Include Statements	68
Finding and Replacing	110	included	128
Firefox	91	Includes	76
Firewall	95	inserting elements	25
fold	70	Installed Debuggers preferences	290
Folding	117	Installed JREs Preferences	307
Folding Preferences	295	Internal Debugger	90
font	71	Internet Explorer	91
Form	150	J	
Format Active Elements	116	JARs	142
Format Document	116	Java Archives	142
Formatter	116	Java Bridge	79, 142
Formatter preferences	301	Java Exception handling	79
Framework	77, 138	Java instantiation	79
Framework Project	77, 138	Java integration	79
Free Registration	98	Java object	139
FTP	212, 213, 215, 238	Java Virtual Machine	79

Java/J2EE	79	Path Mapping	92
Java-PHP Object	79	pattern	70
JavaScript	81, 147	PHP API Toolkit	98
JavaScript Code Assist	147	PHP element	111
JDBC connection profile	158	PHP Executable	30, 66
JRE	143	PHP Executables Preferences	309
JREs	307	PHP Explorer view	231
K		PHP File	24, 102, 103
keymaps	107	PHP Functions view	233
L		PHP Intelligence	85
libraries	94	PHP Interpreter	66
loading-time	40	PHP Interpreter Preferences	310
Local Copy	90	PHP Manual	75
Local Debugging	90	PHP Manual preferences	312
Local History	96, 210	PHP Preferences	282
Locally Debugging PHP Scripts	30	PHP Project	24
M		PHP Project Outline view	235, 236
Manual	75	PHP Script	40, 88, 90
Matching Brackets	70	PHP Script Local Debugging	90
matching pair	70	PHP Script Local Profiling	88
Media	150	PHP Script Remote Debugging	90
minus sign	117	PHP Script Remote Profiling	88
Model	77, 138	PHP Scripts	30
Move	37, 76, 124, 128	PHP Servers Preferences	313
Moving Files	128	PHP version	66
MVC	77	PHP Web Page	30, 90
MVC Outline view	77	PHP Web Page Debugging	90
N		PHP Web Page Profiling	88
NAT	95	PHP/HTML Toolbox	82, 150
New PHP file	24, 102, 103	PHP/HTML WYSIWYG	82, 149
New PHP Project	24	PHP/HTML WYSIWYG Perspective	151
news feed	97	PHPDoc	120
Number of Files	40	phpDoc Block Comments	72
O		PHPDoc Comment	119
Open Debug dialog	30	phpDoc comments	72
Open PHP Element	112	phpDoc tags	72
Organize Includes	37, 76, 124	PHPDocBlock	118
Organizing Includes	129	PHPDocBlock comment	118
Others Time	40	PHPDocs	73
Own Time	40	PHPDocument1	103
P		PhpDocumentor	73
parameter	246	PHPUnit Preferences	315
Parameter Stack view	246	PHPUnit Reporting	87
Parentheses	70	PHPUnit Test	87
Path	40		

PHPUnit Test Case.....	46, 87	required	128
PHPUnit Test Reports.....	46	Restoring Deleted Files	211
PHPUnit Test Suite	46, 87	RSE	238
PHPUnit view	46	RSS	97, 221
Platform.....	85, 164	RSS feed	97, 221
Platform Events	164	RSS view	97
Platform Integration	196	S	
Platform Java Bridge	79	Sample Contents	160
Platform Server	164	Scrapbook	163
plug-ins	98	Searching for PHP Elements	111
port.....	95	security device	95
Preview	82	server	90
problems	84, 85	Server Debugger	30
Problems view	84	Server Location	92
profile.....	88	Server Path Maps	92
Profiler	40	service	226
Profiler Information	40	SFTP	212, 213, 215, 238
Profiler Information View	252	Share Project	205, 209
Profiler Preference.....	316	Sharing Projects	53, 60
Profiling	88	Smart Goto Source	123
Profiling Monitor view	251	SOAP Client	226
Profiling Perspective	40	Source	82
Profiling PHP files	40	source control	53, 60, 95
Profiling PHP Scripts	40	Special Characters	150
Properties	82	SQL Connection.....	158
Properties view.....	151	SQL Databases.....	158, 160
Q		SQL query	163
Query.....	40, 163	SQL Results view	160
R		SQL scrapbook	163
Refactor.....	129	Square	70
Refactoring	37, 76, 124, 128	stack trace.....	241
Relative Path	94	Strings	70
Remote Debugging.....	90	Subversion	60, 96
Remote Profiling	40, 88	Subversive.....	206
remote server.....	95, 202	SVN	60, 96
Remote Systems.....	212	SVN connection	206
Remote Systems view	238	SVN repository	206, 208, 209
Remotely Debugging PHP Scripts	30	SVN Repository Exploring	206
Rename	37, 76, 124	SVN Repository Exploring Perspective	96
Renaming Elements.....	126	SVN server	206
Renaming Files	124	Syntax Coloring.....	71
Replacing Files.....	211	Syntax Coloring preferences	297
repository	53, 95, 96	syntax elements	71
Repository Location.....	53		

- T**
- Table..... 150, 160
 - table content 162
 - Task Tags preferences 298
 - Team..... 205, 209
 - Team Environment 95
 - Templates 108
 - Templates Preferences..... 317
 - Test Case..... 46, 87
 - Test Reports..... 46
 - Test Suite 46, 87
 - Toolbox 82
 - tooltip 74
 - Total Execution Time 40
 - Total Request Time 40
 - Total Time..... 40
 - Tunneling..... 95, 196, 202
 - tunneling icon..... 202
 - Typing preferences..... 300
- U**
- uncomment..... 118
 - Unit testing 46, 87
 - Update Manager 98
 - URL Debugging..... 90
 - URL Profiling 88
- V**
- Variable 194
 - Variables view 242
 - View..... 77, 138
- W**
- warning 84, 85
 - Web Page Debugging 90
 - Web Page Profiling 88
 - Web Services Description Language 97
 - weblogs..... 97
 - Workbench 10
 - Workbench Options preferences 292
 - WSDL..... 97, 222, 226
 - WYSIWYG 82, 149
- X**
- XML 97
- Z**
- Zend 5250 Bridge 98
 - Zend Code Gallery 134
 - Zend Core..... 198
 - Zend Debugger 90, 198
 - Zend Debugger Toolbar 91
 - Zend Framework 77, 138
 - Zend Framework Project..... 77
 - Zend Guard 96, 218, 221
 - Zend Guard Preferences 319
 - Zend Imports 105
 - Zend Network 76
 - Zend Platform 85, 95, 164, 198, 201
 - Zend Platform Events 164
 - Zend Platform GUI 201
 - Zend Platform Java Bridge 79
 - Zend Studio 107
 - Zend Studio 5 105
 - Zend Studio Browser Plugin 91
 - Zend Studio Client Settings..... 201
 - Zend Studio keymap 107
 - Zend Toolbar..... 91
 - ZendIEToolbar.dll 91
 - ZendPlatform 164