PHPeclipse

PHPeclipse User Manual

Robert Kraske

PHPeclipse User Manual

Robert Kraske
Different info about installation, setting up etc. : All the contributors to the **PHPeclipse** wiki

Published 2006-09-10

Table of Contents


PHPeclipse

List of Figures

Chapter 1. Installations

Here it goes.

1.1. Installing Java

The Java Runtime Environment (JRE) is used to run all programs written in the Java programming language. Java 2 is required to run both **eclipse** and **PHPeclipse**. Chances are that you will already have this installed. If not, or if you are not sure, visit Sun's Java Download Site [http://java.com/en/download], which will provide you with what you need (or inform you if you already have Java 2) for any operating system.

If you are using a platform with it's own software management system (such as many linux distributions), you may wish to use your package management system to install Java rather than the Sun download site.

Third party Java Runtime Environments (such as Blackdown [http://www.blackdown.org]) are also available, and may well run this software. Feedback on this is welcome. Our current experience is that Blackdown will not successfully run Eclipse (under Gentoo Linux, Nov 2004).

1.2. Installing eclipse

With Java installed, the Eclipse Platform can now be downloaded and installed. In order to use PHP Eclipse, Eclipse version 3.0 or greater is required. You can download eclipse from the Eclipse Download Page (http://download.eclipse.org/downloads/). You will want to install the Latest Release, which at the time of writing is 3.1.1. This will install the Eclipse SDK. Eclipse is also available as a Platform Runtime Binary, but this will not let you use the debugging features.

Although in general Java applications are platform independent, **eclipse** uses the SWT (Standard Widget Toolkit (http://www.eclipse.org/articles/Article-SWT-Design-1/SWT-Design-1.html)), which provides native look and feel on a range of platforms. This allows much better integration into your desktop than the normal generic Java widgets. This means, however, that you need to select the correct release of **eclipse** for your operating system. A range of different downloads are provided on the download page for your selected release version. Windows and Mac users can simply select the only version for their platform. Linux users need to select the appropriate download for their architecture and desktop environement.

Under Linux, **eclipse** is available for both the Motif and GTK graphical toolkits. GTK is ideal for making **eclipse** look like your other Gnome applications. No **eclipse** version is available for QT (the KDE graphical toolkit) due to the way in which QT is licenced. KDE users will probably want to grab the GTK version. Additionally, you may find it easier to install the Eclipse SDK using your distribution's package management system. For example, under Gentoo Linux, the package dev-util/eclipse-sdk works fine with **PHPeclipse**.

To install the **eclipse SDK** zip file downloaded from eclipse.org [http://www.eclipse.org], simple extract the file to the desired location (eg: C:\Program Files). The file contains a single directory called eclipse. To run eclipse, simple execute the eclipse file in this directory. For example, under MS Windows (if installed to the location above), you would execute "C:\Program Files\eclipse\eclipse.exe". A shortcut could be created for this location.

1.3. Installing PHPeclipse

There are two ways to install this plugin, automated or manual.

1.3.1. Automated

An automated installation of PHPEclipse is available via the Eclipse Update Manager.

- Click on Help->Software Updates->Find and Install... from the Eclipse menu bar. See Figure 1.1, “Find and Install...” [2]
- Select the radio button labeled, "search for new features to install". See Figure 1.2, “Feature Update” [3]
- Click on the **[New Remote Site]** button (see Figure 1.3, “Update sites to visit” [3]), and input the following:

- Name: PHPeclipse official releases
- URL: http://phpeclipse.sourceforge.net/update/releases

If you are interested in unofficial releases you can also add a second “Remote Site”:

- Name: PHPeclipse cvs releases
- URL: http://phpeclipse.sourceforge.net/update/cvs

- Click on **[Finish]** . **eclipse** will now look for new versions of **PHPeclipse**. If the **eclipse** update manager finds something new, you can select the new version within a dialog. See Figure 1.4, “Updates Search Results” [4].
- Click on **[Next]** , and the download of the selected versions begins. See Figure 1.5, “Update Manager Progress” [4].

Note

There may be a problem during automatic installation, if you use the **[Install All]** button during the Eclipse “Find/Install” feature. If the installation stops with an “connection timed out” message (when downloading additional files), close and restart Eclipse and use the **[Install]** button instead of the **[Install All]** button - this will require you to start the installation of the (currently) four packages manually but resolves the problem.

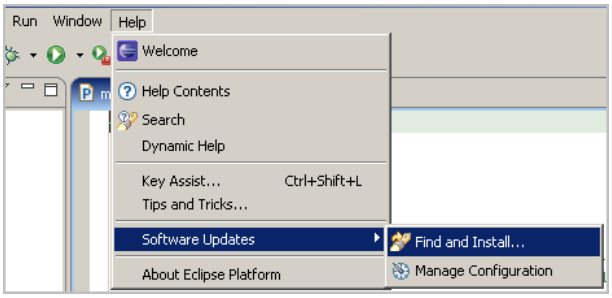

Figure 1.1. Find and Install...

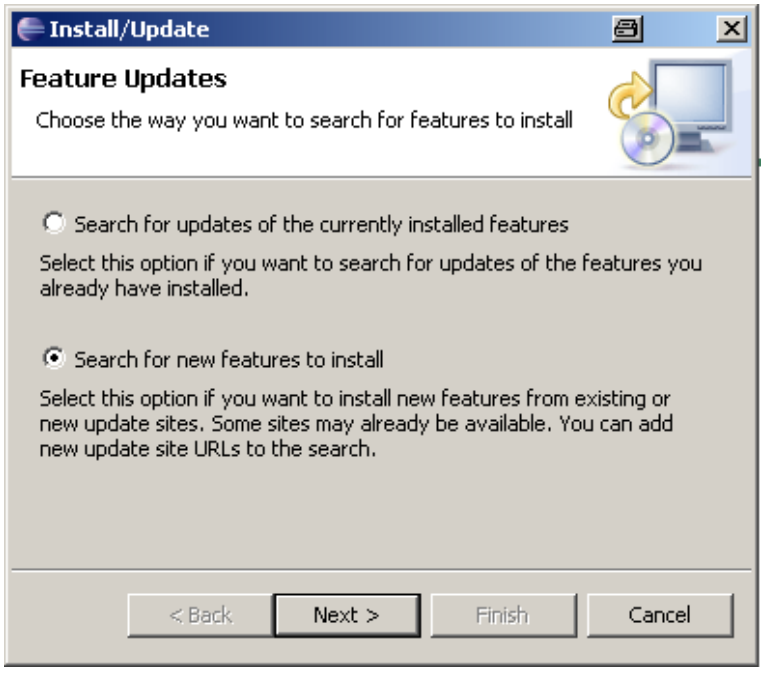

Figure 1.2. Feature Update

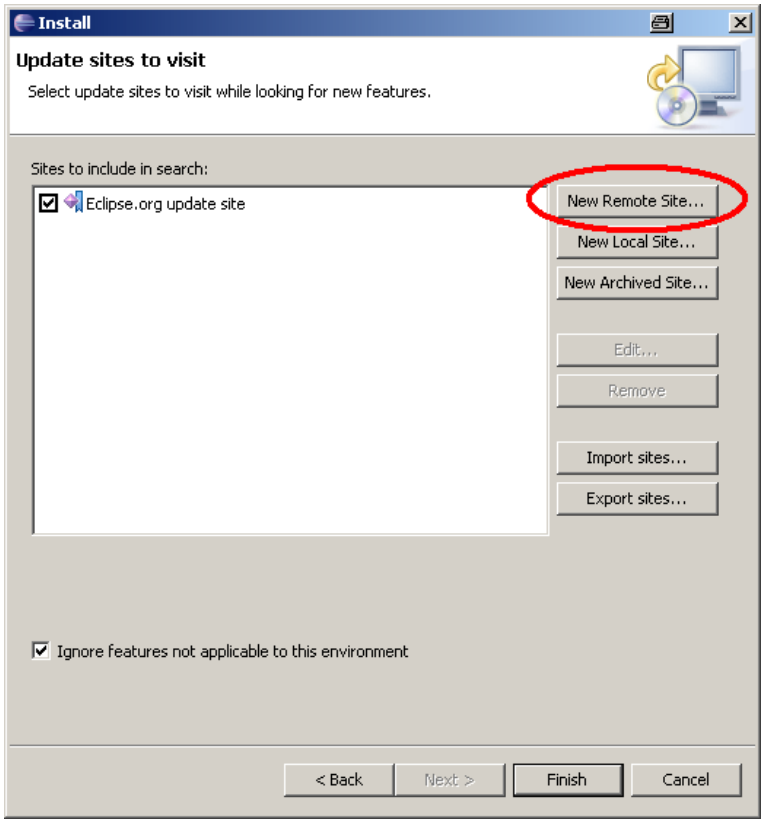

Figure 1.3. Update sites to visit

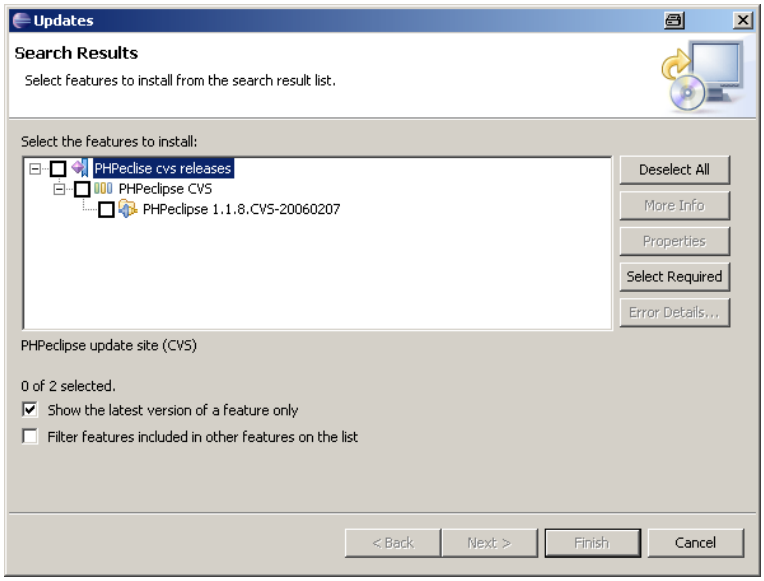

Figure 1.4. Updates Search Results

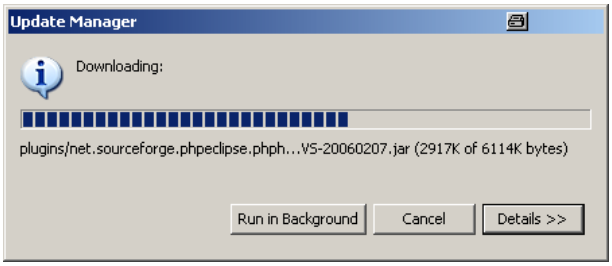

Figure 1.5. Update Manager Progress

Fedora Core 4 (and above) Users:

If clicking on Help->Software Updates->Find/Install commands produces an error message like "Error creating feature "file:/usr/share/eclipse/features/org.eclipse.rcp.source_3.1.1/". [/usr/share/eclipse/features/org.eclipse.rcp.source_3.1.1/feature.xml (No such file or directory)]", start Eclipse as root and try again. (Once you're over this hurdle everything goes smoothly -- even as a non-root user!)

1.3.2. Manual

To install **PHPeclipse** manually, first download the latest version from SourceForge [http://sourceforge.net/project/showfiles.php?group_id=57621]

The **PHPeclipse** zip file must be extracted in the same directory in which you installed **eclipse**. It contains two directories, `plugins` and `features`, and these must match to the plugins and features directory in your existing eclipse directory. See Figure 1.6, “Search for new features to install” [5]

Note

eclipse 3.x caches all plugin.xml files into a single repository for quicker loading. If you used **eclipse** before installing **PHPeclipse**, you should start **eclipse** once with the -clean option. This -clean forces **eclipse** to rebuild that repository. This applies to anything that is installed into **eclipse** by unzipping it into its plugins folder.

After extracting **PHPeclipse**, (re)start **eclipse**. **PHPeclipse** will be loaded automatically. If NOT, you should en-

able it manually. Open the Help->Software Updates->Manage Configuration menu and click the **[Show Disabled Features]** button from the toolbar if not enabled yet. See Figure 1.7, “Enable PHPeclipse in case it is disabled” [5]

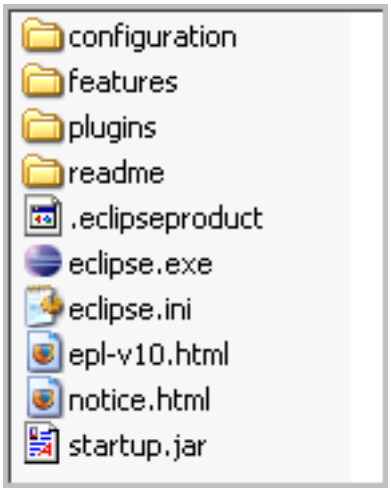

Figure 1.6. Search for new features to install

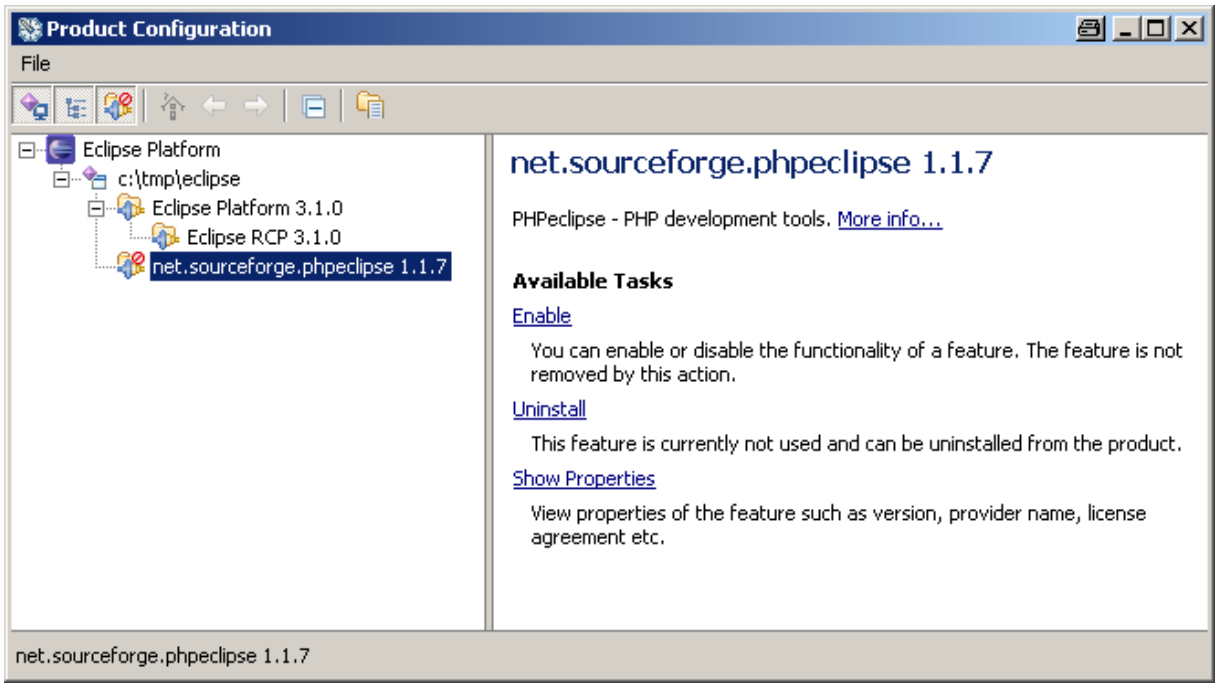

Figure 1.7. Enable PHPeclipse in case it is disabled

1.3.3. Switch between different PHPeclipse versions

If you install a new version (with a higher version number), this version will be activated by default. But sometimes it could be necessary to switch back to an older version (maybe of buggy “unofficial release”).

Switching back to any other version could be done in the following way:

- Click on Help->Software Updates->Manage Configuration from the Eclipse menu bar. See Figure 1.8, “Manage Configuration” [6]

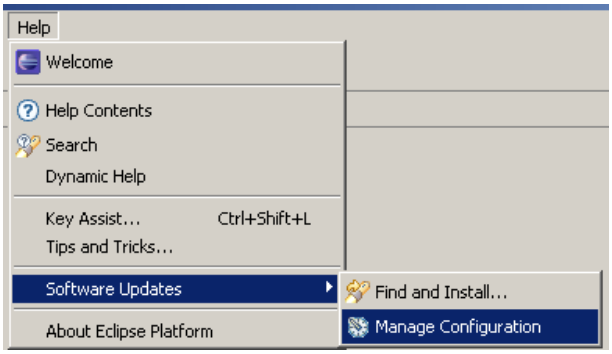

Figure 1.8. Manage Configuration

1.4. Installing Apache/MySQL/PHP,

Most PHP developers find it easiest to work with a running web server of their local machine. This allows you to test your work as you go, without having to upload it to a hosting environment. Therefore, it's suggested that at this point you install Apache, PHP, and (if you are using it), MySQL. This step is required if you plan to use the php debugger. Together this tools are often referred to as AMP (or AMPP if you include Perl as well as PHP).

1.4.1. Installing XAMPP

XAMPP is a software bundle which includes everything you need to install Apache/MySQL/PHP/Perl in one go. It's available for Windows, Linux and Solaris. Particularly on a Windows machine, this is definitely the fastest way to get up and running with these tools, as someone else has done the integration and configuration work for you. Under linux, it may be a quick solution to set up, but will not be as easy to automatically maintain (or integrate with other packages) as if you installed these tools separately with your distribution's package management system (such as: apt, YaST, emerge, or up2date). This is covered in the next section.

To install XAMPP, visit Apachefriends [http://www.apachefriends.org/en/xampp.html]. Select your operating system:

- Windows [http://www.apachefriends.org/en/xampp-windows.html]
- Linux [http://www.apachefriends.org/en/xampp-linux.html]
- Solaris [http://www.apachefriends.org/en/xampp-solaris.html]

and then follow the simple installation instructions found on that page.

There are two different methods of installing XAMPP for Windows:

- Method A: Installation with installer
- Method B: “Installation” without installer

The advantage of method B is, that it doesn't do anything with the Windows registry, and you can have different versions of XAMPP (and so different versions of PHP) in parallel. This can be important if you have to test your PHP files with different versions of PHP 4 (or different versions of PHP 5).

Tip

You can switch between PHP 4 and PHP 5 with `php-switch.bat`.

1.4.2. Changing the Document Root

Assuming that you have installed XAMPP for Windows under `C:\Program Files` the default path of the "document root" is `C:\Program Files\apachefriends\xampp\htdocs`.

Per default this folder contains help and demo files from the XAMPP maintainers. That's what you see if you have started XAMPP and type "http://localhost" within your web browser.

So, when you want to write your own files (That's what we suppose, when you use **PHPeclipse**), you can either delete everything what is within `C:\Program Files\apachefriends\xampp\htdocs`, or you can set the "document root" to any other path by opening `C:\Program Files\apachefriends\xampp\apache\conf\httpd.conf` with an editor, search the line which starts with **DocumentRoot** and change the path to what you need.

The next section shows an additional way to point the webserver to different locations without changing the "document root".

1.4.3. Adding Aliases

An additional way to let the web server know where to look for files is with the help of "Alias". Just open the file `C:\Program Files\apachefriends\xampp\apache\conf\httpd.conf` with an editor and append for example[1]:

```
Alias /my_php_workspace "C:/Documents and Settings/Robert/workspace"
```

Assuming we have PHP files within the folder `C:/Documents and Settings/Robert/workspace/MyFirstPHPProject` e.g. `index. php` the URL to access this file would be http://localhost/my_php_workspace/MyFirstPHPProject/index.php.

Note

The knowledge of where the "document root" is, or how we can change the "document root" or add aliases is a prerequisite when we want to successfully set up and work with **PHPeclipse**.

1.5. Installing Debugger

At the moment only DBG is supported by the standard **PHPeclipse** release. XDEBUG is only supported by loading the appropriate sources via CVS.

1.5.1. Installing DBG

DBG is a full-featured php debugger engine, an interactive tool that helps you debug PHP scripts. It works with a production and/or development web server and allows you to debug your scripts locally or remotely, from an IDE or the console. **PHPeclipse** has a built-in, pure Java debugging client that operates the DBG debugging engine.

To get the debugger to work you need to install the debugger engine to run in the PHP interpreter, and configure the debugger client to communicate with the engine.

Make sure you download the appropriate DBG binaries for your OS and your release of PHP (from 4.0.6 up to 4.3.10 and from 5.0.0 up to 5.0.3) (Note that the 5.0.3 debugger binary seems to work for the 5.0.4 and 5.05 interpreters.) If you are using Mac OS X you will have to build the debugger from source.

1.5.1.1. Getting the right php_dbg.dll

DBG can be downloaded from DBG Downloads [http://dd.cron.ru/dbg/downloads.php]. You need to download the dbg modules packet for the appropriate operating system. Unpack the zipped packet and take the dll (or so) file which matches the PHP version you have currently running. E.g. for PHP 5.1.2 the correct file is `php_dbg.dll-5.1.2`. Rename the appropriate file to `php_dbg.dll` and read Section 1.5.1.2, "Find the extension directory" [8] to find out the right destination location for the file.

[1]The example shows a eclipse default workspace path (in case your user name is Robert).

1.5.1.2. Find the extension directory

The best way to find out is via the “phpinfo()” command. Create a PHP-file called `phpinfo.php` with the following content:

```
<?php phpinfo(); ?>
```

and start it via your web browser. Then search for “extension_dir”. The right column shows the path to the directory where you have to place the `php_dbg.dll` (see Figure 1.9, “The information about the extension directory” [8]).

error_append_string	*no value*	*no value*
error_log	*no value*	*no value*
error_prepend_string	*no value*	*no value*
error_reporting	2039	2039
expose_php	On	On
extension_dir	C:\Program Files\apachefriends\xampp\php\extensions\	C:\Program Files\apa
file_uploads	On	On
gpc_order	GPC	GPC

Figure 1.9. The information about the extension directory

1.5.1.3. Setup `php.ini`

Where is the `php.ini` which needs to be set up?

The simplest way to find out the path of the `php.ini` is again with the help of the phpinfo function as described in Section 1.5.1.2, “Find the extension directory” [8]

Tip

Another possibility to find out where an application searches specific files is by using a tool called Filemon which you can get from Sysinternals [http://www.sysinternals.com].

The resulting output shows the path of the `php.ini` file within the topmost header. See Figure 1.10, “Get the php.ini path” [9]

Now open `php.ini` and search for “implicit_flush”, and set it to “On”.

```
; this is to see output while debugging
implicit_flush = On
```

copy the following lines into `php.ini` (they can go at the end of the file). For linux it is a php_dbg.so instead of php.dbg.dll.

```
[debugger]
extension=php_dbg.dll
debugger.enabled=on
debugger.profiler_enabled=on
debugger.hosts_allow=localhost
debugger.hosts_deny=ALL
debugger.ports=7869, 10000/16
```

Disable eAccelerator [http://eaccelerator.net] if it is installed and enabled:

```
eaccelerator.enable="0"
```

Disable the Xdebug [http://xdebug.org/] extension if installed (usually via pear) by commenting out

```
;zend_extension=/usr/lib/php4/20020429/xdebug.so
```

Also if you have Zend Extension Manager installed (You should find a [Zend] section) make sure you add this line before any other zend_extension_ts (windows) or zend_extension (linux) lines: For Windows:

```
zend_extension_ts = "C:\path\to\php_dbg.dll"
```

For linux:

```
zend_extension=/var/lib/php4/dbg.so
```

Restart web server

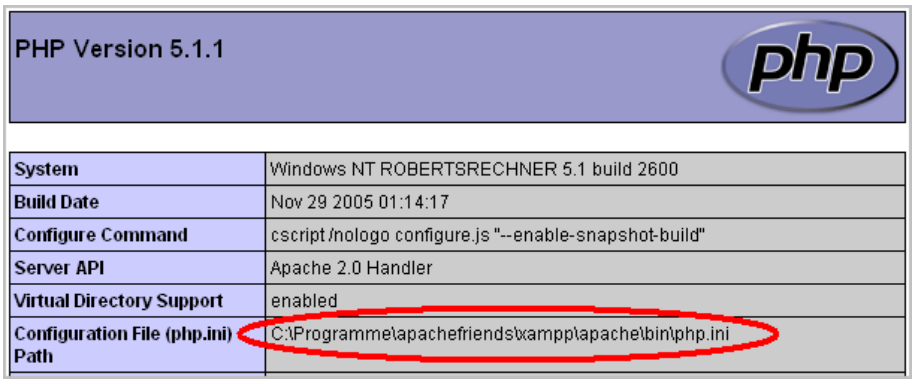

Figure 1.10. Get the `php.ini` path

1.5.1.4. Test the DBG Installation

When you have done all steps, restart your web server and again submit the `phpinfo` function as described in Section 1.5.1.2, “Find the extension directory” [8]. Search the browser output for “dbg” and you should find a section like you can see in Figure 1.11, “phpinfo showing the DBG section” [9]

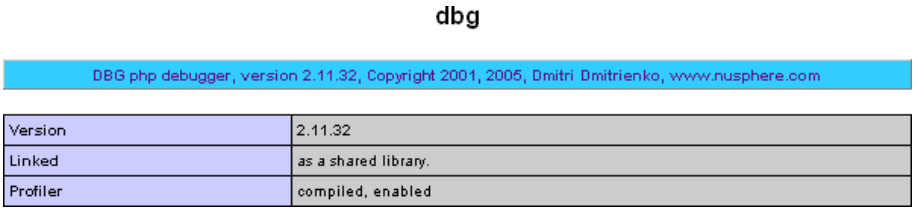

Figure 1.11. phpinfo showing the DBG section

In addition you can submit the following URL within your browser: http://localhost/index.php?DBGSESSID=1@localhost:10001. Now, as there is no client running listening to dbg, you should see the following error message

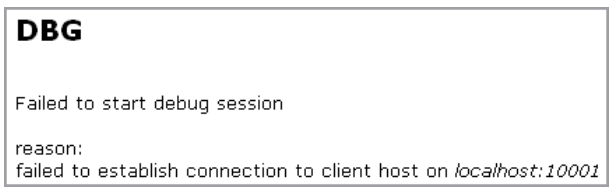

Figure 1.12. DBG Error Message

Note

For setting up a debug configuration and doing real debugging with **PHPeclipse**, see Section 4.1, “PHP Source Level Debugging” [25].

1.5.2. Installing XDEBUG

The XDebug support is only available through direct checkout of the **PHPeclipse** CVS repository.

More to come.

Chapter 2. Workspace

Once you have installed **eclipse** and start it the first time **eclipse** asks you about selecting a workspace. The (see Figure 2.1, “Select a workspace” [11]. The default path for the workspace on a Windows system is`C:\Documents and Settings\your_username\workspace`. Even if you don't know what the workspace is good for, and even if you don't know whether this is a good place for a workspace you can safely accept this path with clicking “OK”.

It is also no problem to activate the **[Use this as default and do not ask again]** . You can always change your workspace and/or can activate to be asked again for a workspace when **eclipse** is starting.

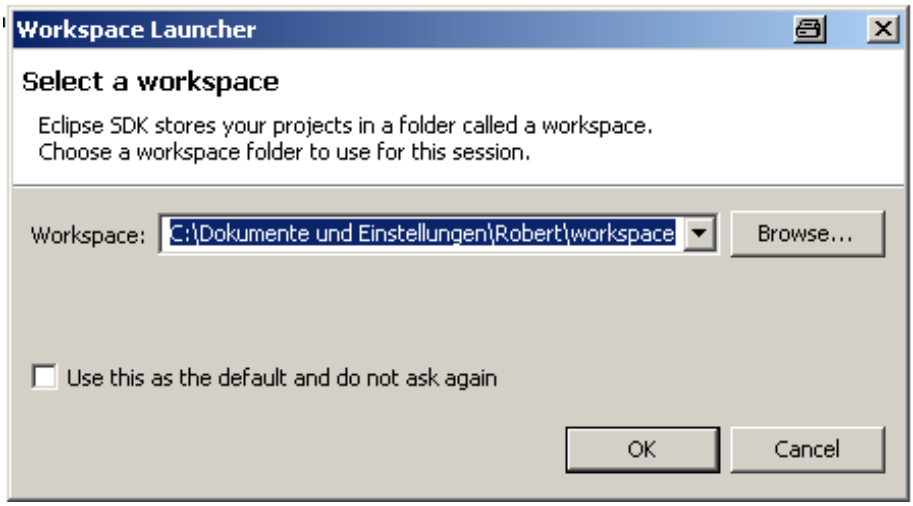

Figure 2.1. Select a workspace

2.1. Changing the workspace

You can change the workspace after **eclipse** has started by clicking on File->Switch Workspace... (see Figure 2.2, “Switch a workspace” [11] This opens a dialog (see Figure 2.3, “Select a workspace” [12] where you can select an already existing workspace, or a new folder which should be used as a workspace.

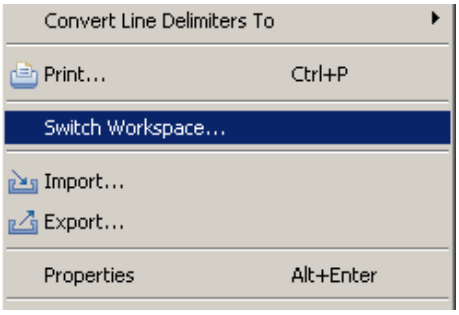

Figure 2.2. Switch a workspace

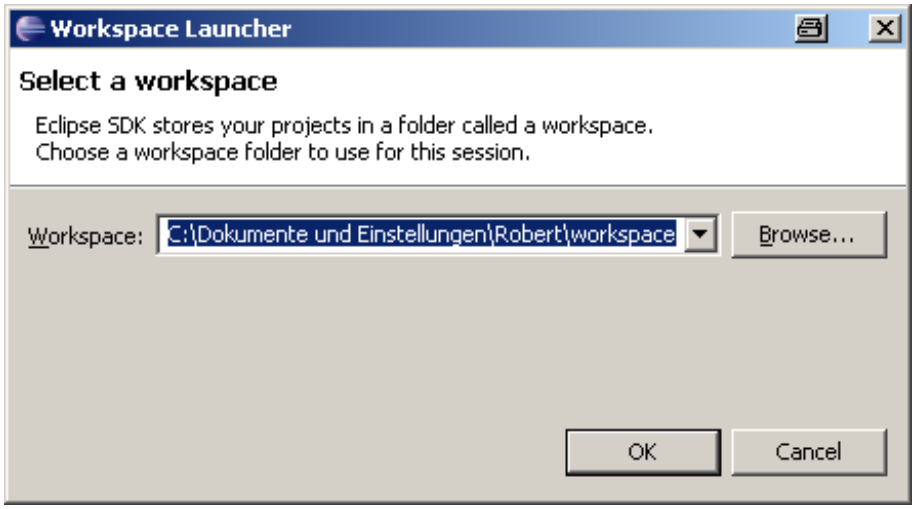

Figure 2.3. Select a workspace

2.2. Activate the “Select a workspace” dialog

In case you have deactivated the prompting for a workspace setting, you can activate it again by clicking on Window->Preferences->General->Startup and Shutdown and then activating **[Prompt for workspace on startup]** (see Figure 2.4, “Activate prompt for workspace” [12])

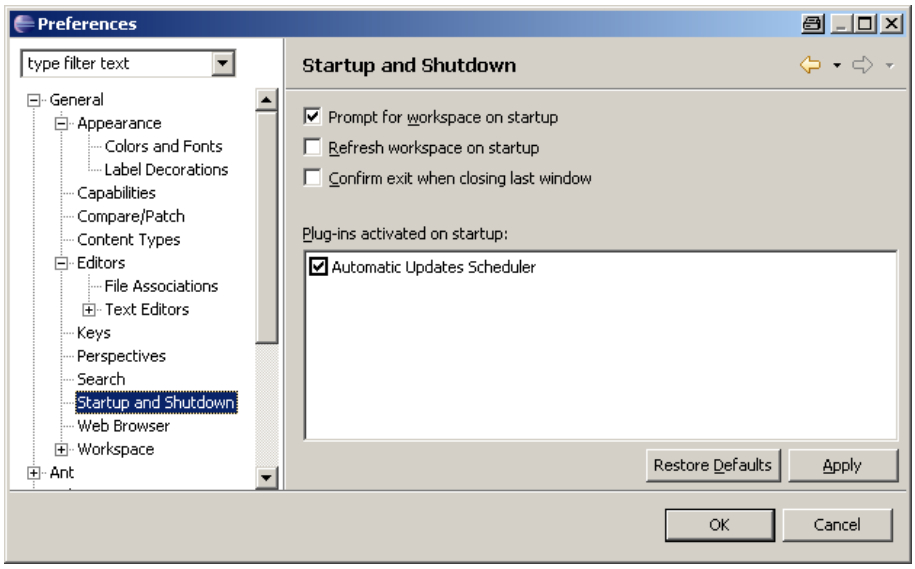

Figure 2.4. Activate prompt for workspace

2.3. What workspace to choose for PHP

So now I know how to select a workspace, but what is a good place for a workspace. I have different projects and/or have already existing files, have a already running webserver. So how does this all match?

So, as all roads lead to Rome there are several different methods which will be discussed in the following sections. But first, let us explain how the workspace directory will look like, as the structure of a workspace will always be the same (independent of where it is located).

Within the workspace folder there is a folder named `.metadata`, which we will, for simplicity, not discuss

here. Besides this folder, **eclipse** will create additional folders for every project, independently whether the project is a PHP, Java or anything else project.

2.3.1. Workspace Method 1

The workspace is set to be in the `Documents and Settings` folder and also the PHP-files are located within the workspace.

In this case we would have the following path: `C:\Documents and Settings\Robert\workspace`. When we have created a project (e.g. MyFirstPHPProject) through **eclipse** and create a PHP-file (e.g. index.php) we would see the file as: `C:\Documents and Settings\Robert\workspace\MyFirstPHPProject\index.php`.

2.3.1.1. Configure Web Server by changing Document Root

Given that our webserver is installed in a different directory and also the “document root” points to a different directory, we have to change the “document root” path to our `workspace` folder (see Section 1.4.2, “Changing the Document Root” [6]), which would be the following in our example:`C:\Documents and Settings\Robert\workspace`

As the project's default “document root” is set to the current workspace, we do not need to change anything.

In this case the file `myFirstFile.php` can be called by the web browser with the following URL: http://localhost/MyFirstPHPProject/myFirstFile.php

2.3.1.2. Configure Web Server by adding Alias

As an alternative to changing the “document root” we can add an alias (see Section 1.4.3, “Adding Aliases” [7]) to the web server's configuration file. In this case we also would have to change the localhost setting within our project properties.

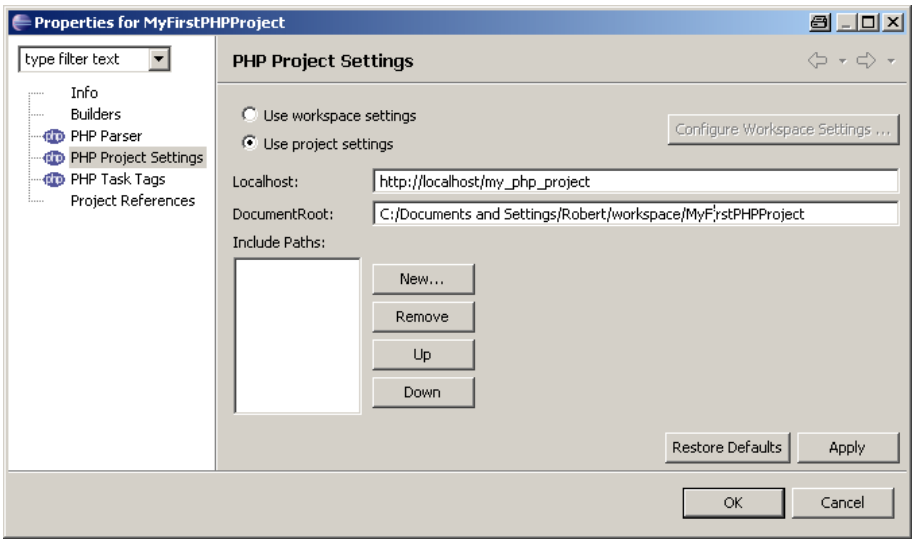

Figure 2.5. Set localhost when using Alias

2.3.2. Workspace Method 2

The workspace path is the same as the “document root” path.

In some cases (or maybe in many?) people which tries to work with **PHPeclipse** have already set up a working Apache and PHP environment. Which means, they have already a htdocs directory anywhere on their local machine, and they have already some PHP-files within the existing htdocs folder. And they don't want to move all their files into the **eclipse** workspace folder. One way to accomplish this would be to set the **eclipse** workspace to the already existing htdocs folder.

For example you have set up XAMPP and your “document root” path is the XAMPP default: `C:\Program Files\apachefriends\xampp\htdocs`, then your appropriate workspace path should be one level less: `C:\Program Files\apachefriends\xampp`. Now you create a new PHP project and name it “htdocs”. **PHPeclipse** will automatically add all files within the `htdocs` folder to the project. Just adjust the project's “DocumentRoot” setting as described in Section 2.3.1.1, “Configure Web Server by changing Document Root” [13].

2.3.3. Workspace Method 3

The workspace is set to be in the `Documents and Settings` folder and the PHP-files are in a different folder.

As we have seen how we can work with method 1 and method 2, there is another method of setting up your workspace and project which also allows you to hold the **eclipse** workspace and your PHP-files separated.

For example your workspace path is `C:\Documents and Settings\Robert\workspace` and your current “document root” is `C:\Program Files\apachefriends\xampp\htdocs`. See Section 3.2.3.2, “Link Folders” [22] how to set up a project for this case.

Chapter 3. Project

So, when finished with setting up a workspace, we can go for our first PHP project. This is a necessary step, because a workspace alone will not help. We also need to set up a project within a workspace. As we want to work with PHP we will create a PHP project.

3.1. Create a project

For creating a project klick on File->New->Project... (see Figure 3.1, “Create a new projec” [15]). This will open a project creation wizard dialog (see Figure 3.2, “Select a wizard” [16]). Now click on “PHP Project” under the PHP tree entry as you can see in Figure 3.3, “Name your PHP Project” [16]. When you click on**[Next >]** the PHP Project dialog will be shown, where you can input a name for your new PHP Project. In our example we choose the name “MyFirstPHPProject”. Now, just click on **[Finish]** , and your project will be created. Now you should see your project within the Navigator pane (see x

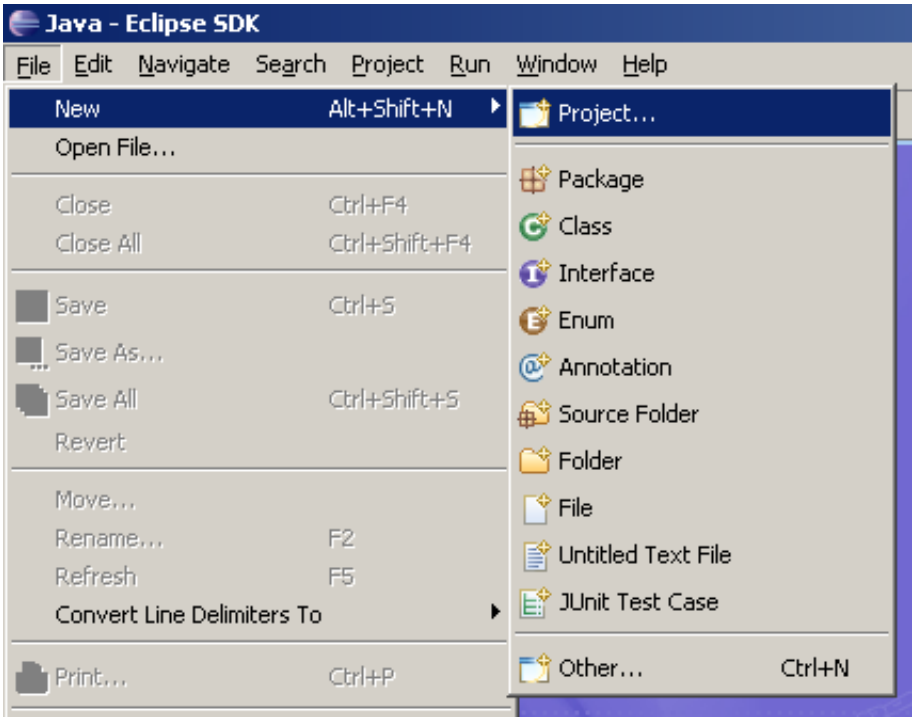

Figure 3.1. Create a new projec

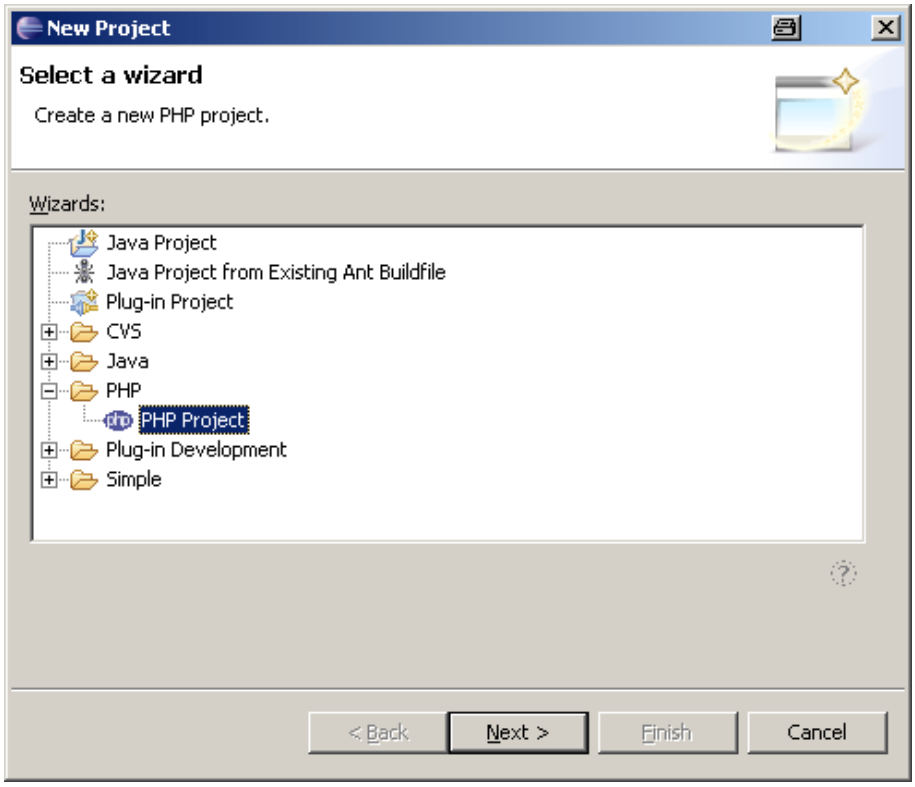

Figure 3.2. Select a wizard

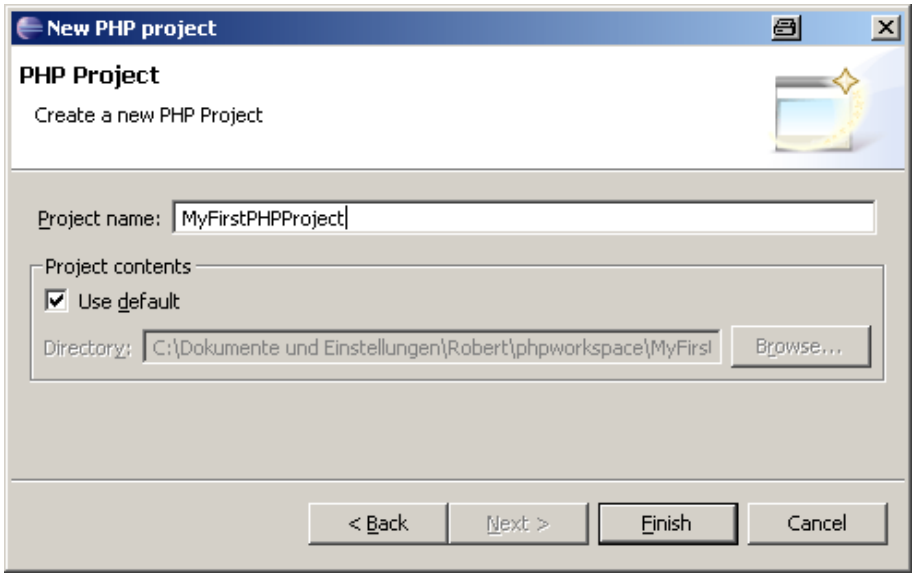

Figure 3.3. Name your PHP Project

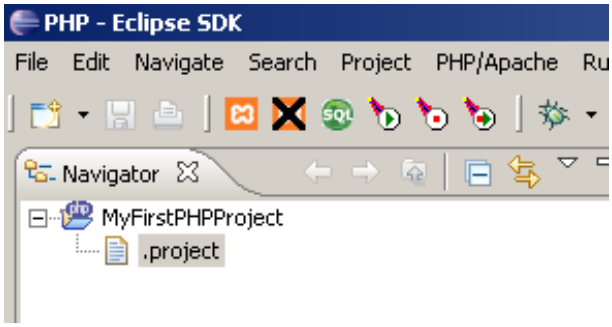

Figure 3.4. The new PHP Project within the Navigator pane

3.2. Adding files to the project

Now, as we have created a PHP project we need some PHP files in it. The following sections will explain the different aspects about adding a file to your project.

3.2.1. Adding a simple file

Per se, there are two different methods of adding a file to your project. Both are available through the “File” menu.

The first method is to add a “simple” file to your project by clicking on File->New->File (see Figure 3.5, “Adding a simple file to the project” [17]). This will open the appropriate “New File” dialog. Input the name for that file (with the extension “php”), and click on **[Finish]** . This will add the new file to the “Navigator” pane, and opens an editor view.

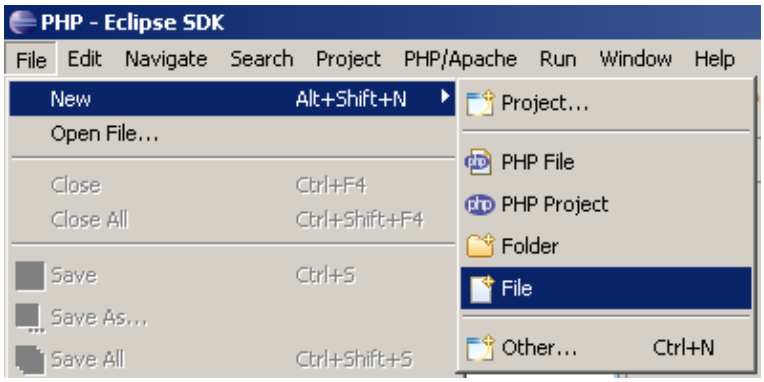

Figure 3.5. Adding a simple file to the project

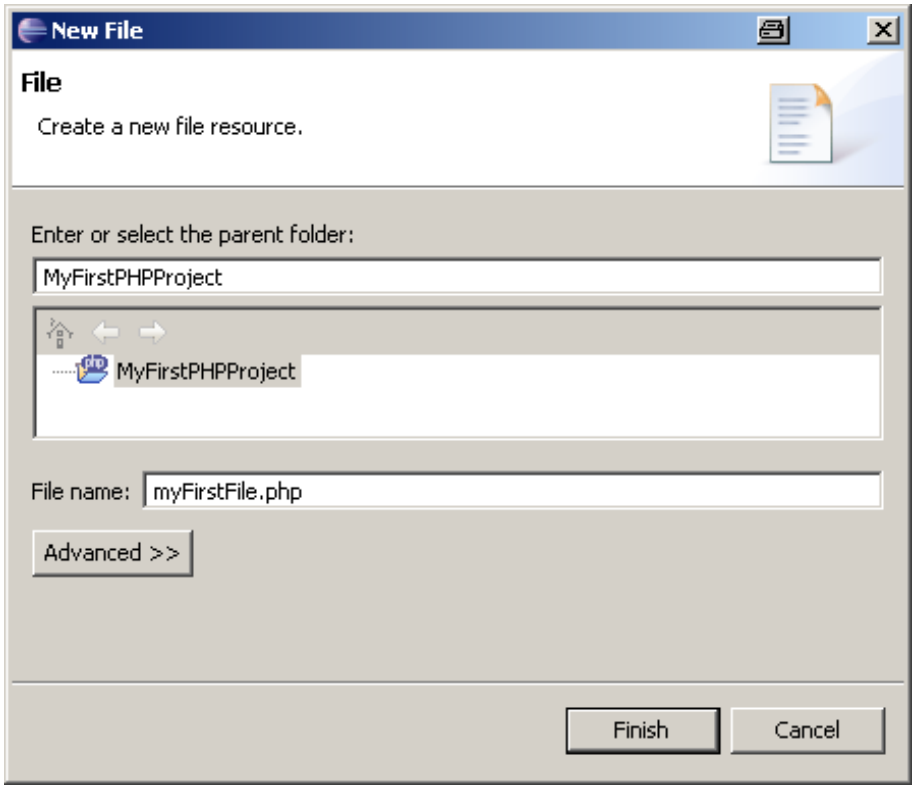

Figure 3.6. The *New File* Dialog

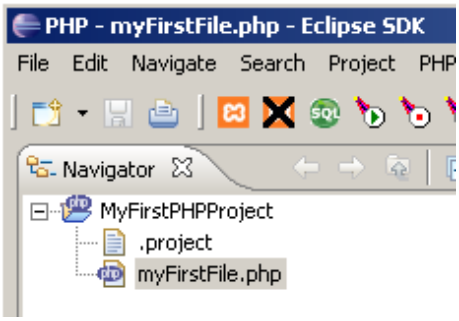

Figure 3.7. The new file in the *Navigator* view

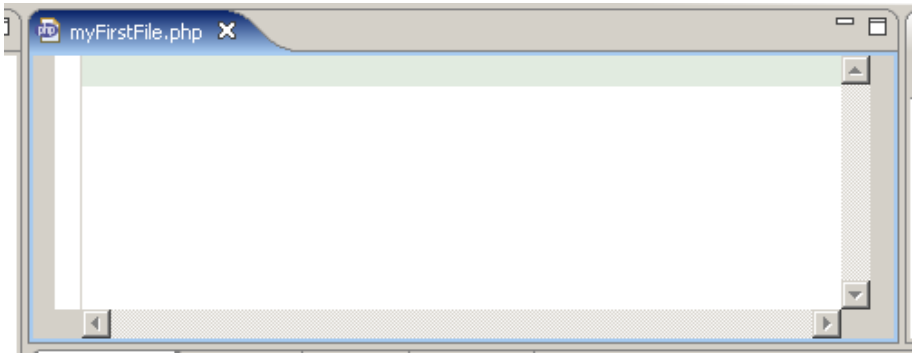

Figure 3.8. The new file opened in PHP editor

3.2.2. Add a PHP file to the project

The second method of adding a PHP file to your project is through File->New->PHP File (see Figure 3.9, “Adding a PHP file to the project” [19]). This will open a “New PHP file” dialog with a default File name entry of “file.php” (see Figure 3.10, “The New PHP filedialog” [20]). We change the file name to`mySecondFile.php` (see Figure 3.11, “Change the name to what you like” [20]) and click on**[Finish]** .

This also will add a PHP file to your project as you can see in the *Navigator* view (see Figure 3.12, “The new PHP file in the Navigator View” [20]), and it also opens this new file within the PHP editor (see Figure 3.12, “The new PHP file in the Navigator View” [20]) as it was done in Section 3.2.1, “Adding a simple file” [17].

But there is obviously a difference. In opposite to `myFirstFile.php` which is a real empty file, `mySecondFile.php` has some text in it (see Figure 3.13, “The new PHP file opened in PHP editor” [21]). The file we just created is filled with a “template”.

The difference is only at creation time. The files themselfs, when created, do not differ in any sense.

When we look at file system level, the absolute path of the files we just created are in respect of the current workspace, in general: `workspace_path/project/file.php`. For example if you have accepted the default workspace when you started eclipse (and your username is Robert):

- *myFirstFile.php*: `C:/Dokuments and Settings/Robert/workspace/MyFirstPHPProject/myFirstFile.php`
- *mySecondFile.php*: `C:/Dokuments and Settings/Robert/workspace/MyFirstPHPProject/mySecondFile.php`

Note

The importance of knowing the absolute path is when the web server comes into account. As you want to test the program you have written, the web server needs access to your PHP files. This could be accomplished either when you set your web server's document root to point at your current workspace, or set your workspace to the currect web server's document root.

But there is a third way, which will make things easier, as you can leave the workspace wherever you want, and also do not need to change the web server's document root.

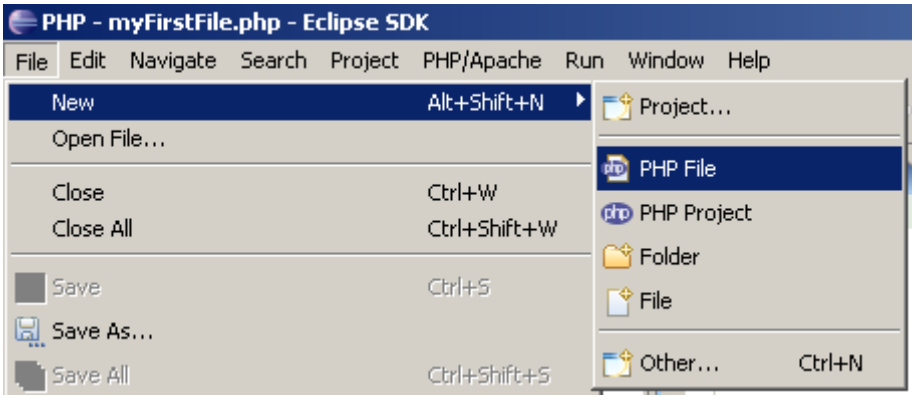

Figure 3.9. Adding a PHP file to the project

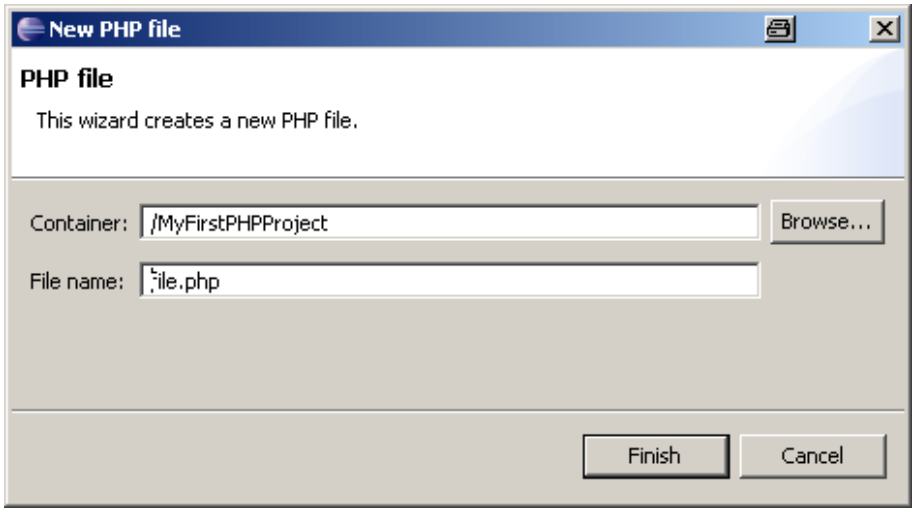

Figure 3.10. The *New PHP file*dialog

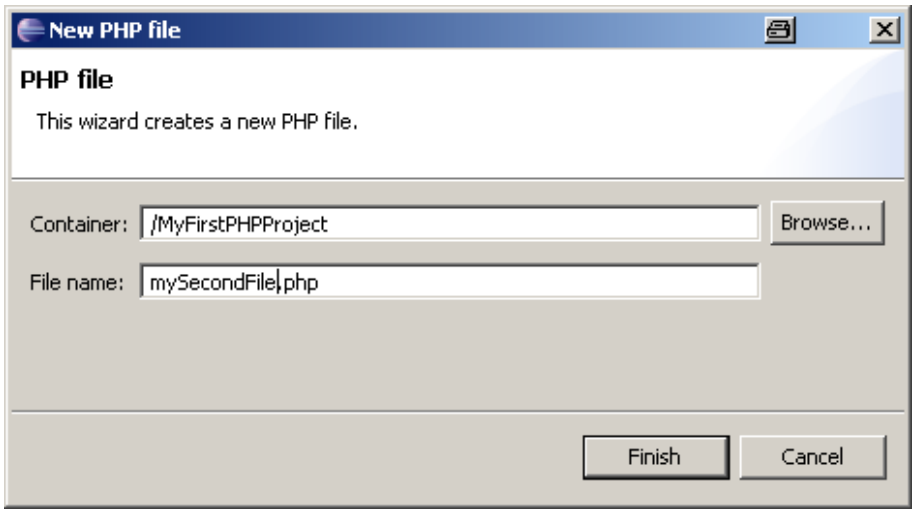

Figure 3.11. Change the name to what you like

Figure 3.12. The new PHP file in the “Navigator View”

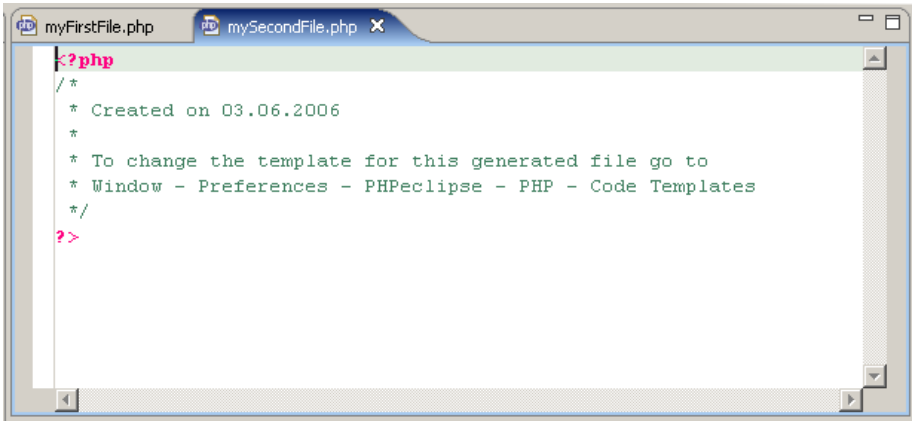

Figure 3.13. The new PHP file opened in PHP editor

3.2.3. Add existing files to the project

There are generally two ways to add files, which already exists anywhere on your file system, to a project[2].

3.2.3.1. Import Files

You can import files from anywhere on your local machine by eclipse itself via File->Import... (see Figure 3.14, “Import Files into Project” [22]) or by manually copying the files via the file system into your project folder (see Section 3.2.2, “Add a PHP file to the project” [19]). In fact the way via *Import...* is more convenient.

But this method only makes a copy of existing files, and doesn't disburden you from changing the web server's “document root”. Especially if the files you have currently imported to your workspace are located on your current web server's “document root”.

In this case it would be easier to set the workspace to the current web server's “document root”, or do as described in the following section Section 3.2.3.2, “Link Folders” [22].

[2]Every file which you want to open and handle with **PHPeclipse** needs to be within a project.

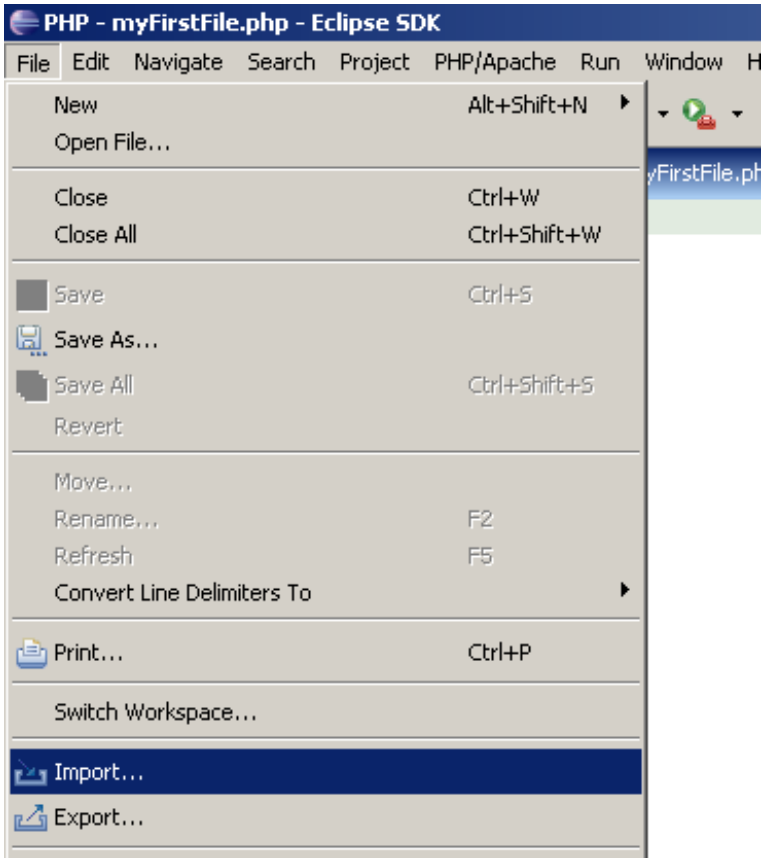

Figure 3.14. Import Files into Project

3.2.3.2. Link Folders

Indeed, this should be the best option if you want to have independence between a eclipse workspace and the web server's “document root”.

- Create a new project (e.g. “MyLinkedPHPProject”).
- Right click on this project within the “Navigator View” to open the context menu and click on. New->Folder (see Figure 3.15, “Add a Folder to Project” [23]).
- Click on **[Advanced >>]** and enable **[Link to folder in the file system]** (see Figure 3.16, “Create a Link Folder” [23]).
- Now you can either manually enter the path or browse to the folder which is to link. In the example we input `C:\Program Files\apachefriends\xampp\htdocs`, and click on **[Finish]** . The “Navigator View” should now look like (see Figure 3.17, “The Navigator View with the new Project” [24]).
- At last we have to set the correct “document root” (which is `C:\Program Files\apachefriends\xampp\htdocs`) within the project properties as shown in Figure 3.18, “Change the Project's Document-Root Setting” [24]

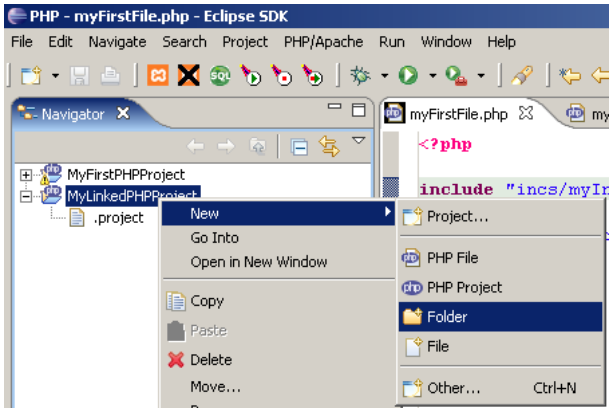

Figure 3.15. Add a Folder to Project

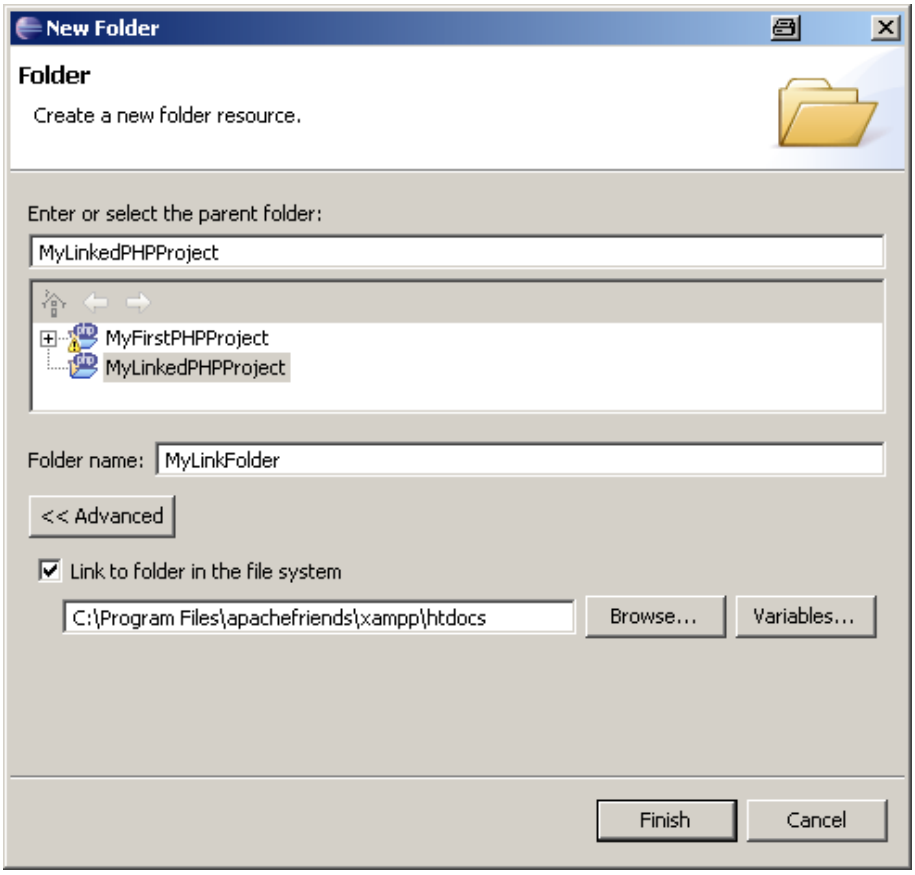

Figure 3.16. Create a Link Folder

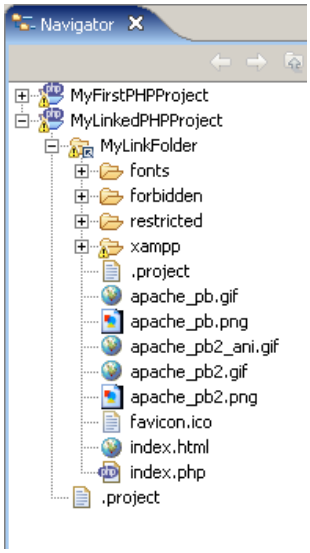

Figure 3.17. The “Navigator View” with the new Project

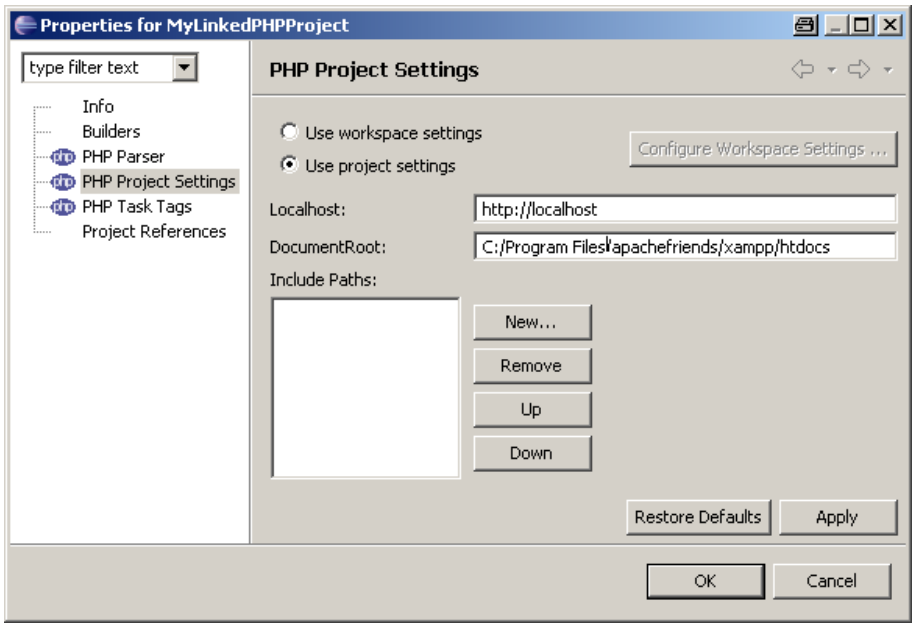

Figure 3.18. Change the Project's DocumentRoot Setting

Chapter 4. Debugging

The following section (Section 4.1, “PHP Source Level Debugging” [25]) describes one of the most interesting aspects when developing software: Debugging.

The second section (Section 4.1, “PHP Source Level Debugging” [25]) within this chapter will describe how you can debug **PHPeclipse** itself. In case you want to find out how things work, or you found a bug in **PHPeclipse** and want to help the **PHPeclipse** developers.

4.1. PHP Source Level Debugging

Besides the method of outputting runtime information by adding `echo` or `var_dump` functions to your code, the more sophisticated and convenient method is using a so called “Source Level Debugger”.

With the current release of **PHPeclipse** only DBG is supported. XDebug is currently supported only by directly checking out from the **PHPeclipse** CVS repository and setting it up appropriately (see Section 1.5.2, “Installing XDEBUG” [10]).

In case you havn't already installed DBG, see Section 1.5.1, “Installing DBG” [7]. When you are sure the DBG module is correctly installed, you can go forward.

To do debugging in **PHPeclipse** you need to set up a so called “Debug Configuration”. The following sections describe how to set up such a “Debug Configuration” and how to start a debug session.

4.1.1. Setting up a Debug Configuration

To set up a “Debug Configuration” click on Run->Debug... (see Figure 4.1, “Open the Debug Configuration dialog” [25]. Within the opened dialog click on “PHP DBG Script” and click on **[New]** (see Figure 4.2, “Debug Configuration with the Perspectives view” [26]). This opens the “Create, manage, and run configuration” dialog.

Note

The first time you open the configuration dialog, you will see an error message within this dialog, complaining about a missing interpreter setting. For entering the interpreter see

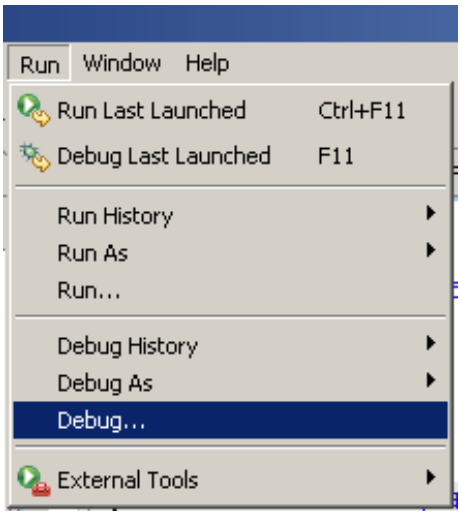

Figure 4.1. Open the Debug Configuration dialog

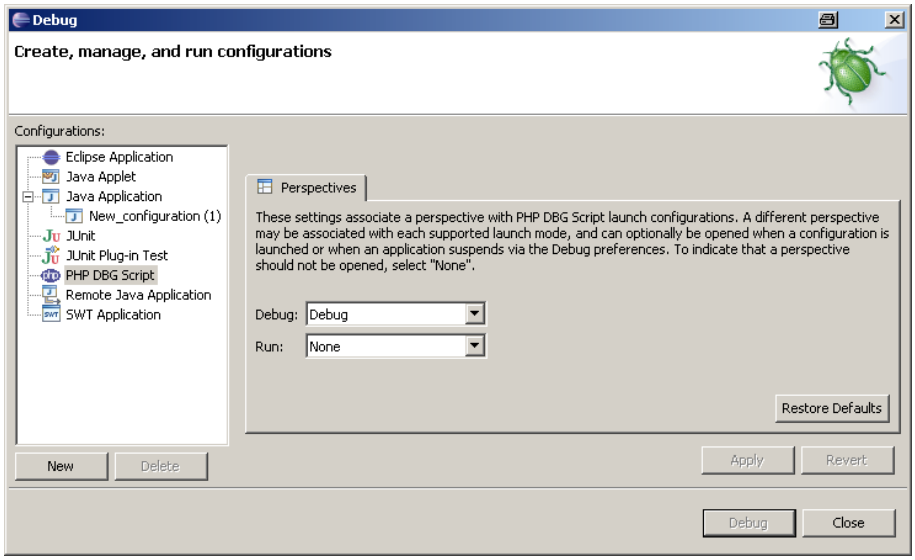

Figure 4.2. Debug Configuration with the “Perspectives” view

4.1.1.1. Name the Debug Configuration

Within the “Name:” text field you should change the name (which is “New_configuration” per default) to a more reasonable one (see Figure 4.3, “Set up Debug Configuration - File” [27]).

4.1.1.2. Configure File

Within the “File” tab you find two entries:

- **Project:** This is the project this debug configuration belongs to.
- **File:** This is used for “Non Remote Debugging”. The given file is directly given the interpreter which is entered within the Interpreter tab (see Section 4.1.1.4.1, “Configure Environment Interpreter” [27]). And/or is it called when configured for using the internal browser.

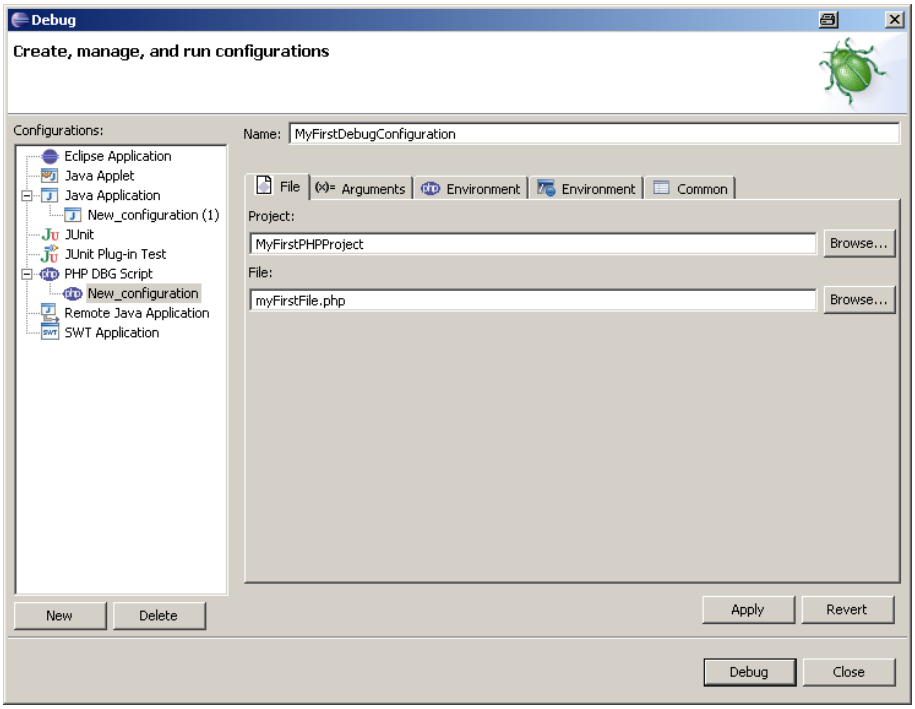

Figure 4.3. Set up Debug Configuration - File

4.1.1.3. Configure Arguments

These settings are used for “Non Remote Debugging”. If the given Interpreter is directly called by **PHPeclipse** (see Section 4.1.3, “Debugging CLI” [28]).

- **Working Directory**:
- **Use default working directory**:
- **Interpreter Arguments**: Here you can specify the arguments which will passed to the interpreter (see Using PHP from the command line [http://de.php.net/features.commandline]).
- **Program Arguments**: Here you can specify the arguments which will be passed to your script, and can be accessed by the `$argv[]` - array.

4.1.1.4. Configure Environment

Within this tab you have to enter the most important settings for a successful debugging session.

4.1.1.4.1. Configure Environment Interpreter

Although the setting of the interpreter isn't really necessary for “Remote Debugging”, you have to set it up. For a default XAMPP installation this would be `C:\Program Files\apachefriends\xampp\php\php.exe`.

The interpreter is used when you set a “Debug Configuration” with “Remote Debug” deactivated. This is true if you want to debug a script which is started per php cli.

4.1.1.4.2. Configure Environment Remote Debug

Here can you have up to five options to enter:

- **Remote Debug**: You activate this if you want to debug a script which is started from an external browser (this is in opposite of the usage of the browser which is build into **PHPeclipse**). When you disable this checkbox the appropriate script is forwarded directly to the given interpreter (php.exe)
- **Cross Platform debugging**: Originally used for converting the path seperator (which are different on *nix and Windows systems). As far as I can remember this isn't necessary anymore.
- **Open with DBGSession URL in internal Browser**: Activate this if you don't want to use the an external browser but start a debug session in internal browser.
- **Remote Sourcepath**: The incorrect setting of this path information is responsible for the most annoyances in respect of debugging problems. The “Remote Sourcepath” is important because it is used for translating the path informations which are exchanged between **PHPeclipse** and DBG. So, if this translation is incorrect, **PHPeclipse** will not find the correct file which is currently running, nor will DBG know for which file it should set a breakpoint. More different and detailed examples are to come:
- **Mapped Path**: Although in most cases a single path mapping is sufficient, but there are situations where this isn't enough. This is when you use PHP files (e.g. phplib) which are located on a total different path. To use this option, it is necessary to leave the “Remote Sourcepath” field empty. More different and detailed examples are to come:

4.1.1.5. Configure Environment Variables

Used within “Non Remote Debugging”. You can specify variable - value pairs which are passed to the script within the `$_ENV` and the `$HTTP_ENV_VARS` array.

4.1.1.6. Configure Common

Leave all as it is per default.

4.1.1.7. Save Configuration

When you have done all the necessary settings click on **[Apply]** to save the configuration. You can recall this configuration by clicking on the name of the desired configuration in the “Configurations pane” on the left side.

4.1.1.8. Start a Debug Configuration

4.1.2. Remote Debugging

It is called “Remote Debugging” if you want to debug php scripts which are started by web server. This is the case if you are doing so called “Web Devolepment”. In opposite you also can write php scripts and start them from a shell or command line (see Section 4.1.3, “Debugging CLI” [28].

4.1.3. Debugging CLI

In contrary to “Remote Debugging” where your php scripts are started by a web server, you could also write scripts and start them from command line[3]. For that task you have to deactivate the “Remote Debug” checkbox (see Section 4.1.1.4.2, “Configure Environment Remote Debug” [27]), and set the appropriate php cli application. According the PHP version you use, this is:

- phpcli.exe for PHP 4
- php-win.exe for PHP 5

The prerequiste for debugging is also that you have installed dbg as it is described in Section 1.5.1, “Installing DBG” [7]. In addition, for the breakpoints to work, you have to add the following line to the “[debugger]” section of your `php.ini`:

[3]cli stands for “Command Line Interface”

```
debugger.JIT_enabled=on
```

And also you need the following code at the top of your php file you want to debug. At least this code snippet should be located before the first breakpoint you want to set.

```
if (function_exists ("DebugBreak")) {
  DebugBreak ();
}
```

Note

For debugging a php-cli script you need the appropriate **PHPeclipse** version, which is > phpeclipse-1.1.9-cvs-20060424. This is due to the fact that older **PHPeclipse** versions listen on port 10001, but dbg uses the standard port 7869, and there is no way to submit a different port neither by cli nor through the `php.ini`[4].

Tip

If you don't want to change the `php.ini` which you also do use for your web scripts, you can "importy" the appropriate `php.ini` to your project, and modify this project related local copy of the `php.ini` file. To use this local `php.ini` set the "Interpreter Argument" (see Section 4.1.1.3, "Configure Arguments" [27]) as follows: "-c your_project_name/php.ini".

4.1.4. Debug Scenarios

4.1.4.1. Same Machine, Workspace is "document root"

4.1.4.2. Same Machine, Workspace and "document root" is Different

4.1.4.3. Different Machine, Workspace and "document root" is Different

4.1.5. Running a Debug Session

Open Debug perspective. Select a existing debug configuration. Or create a new one. Should see following in Debug View. Internal Browser or external browser. Submit URL in external browser.

4.1.5.1. Resume

4.1.5.2. Suspend

4.1.5.3. Terminate

4.1.5.4. Step Into

4.1.5.5. Step Into

[4]This is true for the free dbg version.

4.1.5.6. Step Return

4.1.5.7. Breakpoints

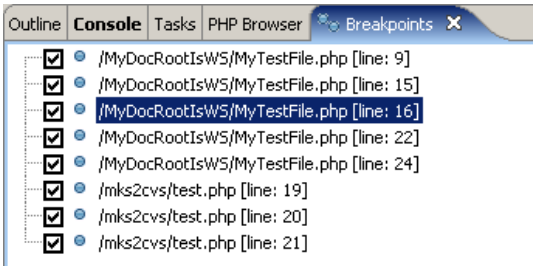

Figure 4.4. The “Breakpoints View”

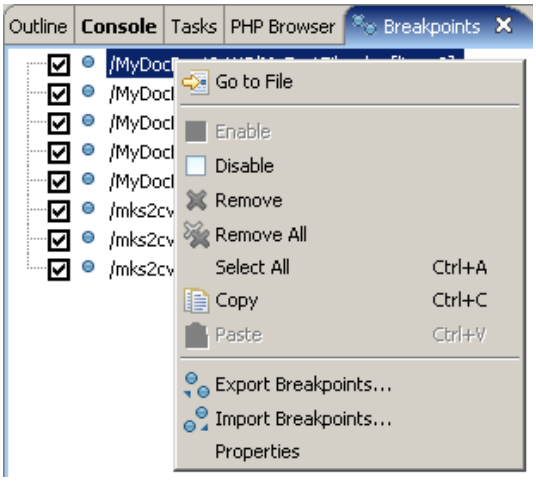

Figure 4.5. The “Breakpoints View” context menu

Figure 4.6. The “Breakpoints View” icon bar

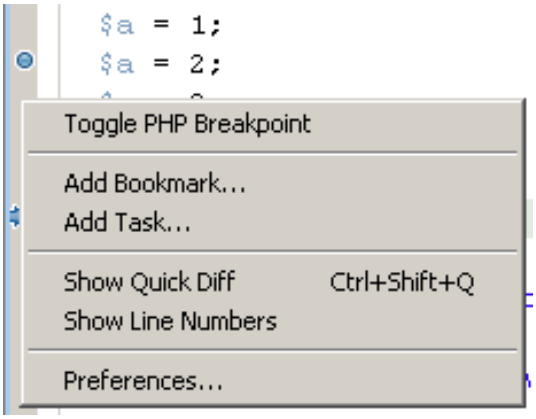

Figure 4.7. The “Editor View” left ruler context menu

4.1.5.7.1. Set Breakpoint

A breakpoint can be set by the following methods:

- By double clicking within the left vertical ruler of the editor window
- Via the left vertical ruler context menu item Toggle PHP Breakpoint (see Figure 4.7, “The Editor View left ruler context menu” [30])

4.1.5.7.2. Remove Breakpoint

A breakpoint can be removed by the following methods:

- By double clicking within the left vertical ruler of the editor window
- Via the left vertical ruler context menu item Toggle PHP Breakpoint (see Figure 4.7, “The Editor View left ruler context menu” [30]).
- Via the “Breakpoints View”, by simple typing **Del**
- Via the “Breakpoints View” context menu item Remove (see Figure 4.5, “The Breakpoints View context menu” [30])
- Via the “Breakpoints View” icon bar “Remove” icon (see Figure 4.6, “The Breakpoints View icon bar” [30]).

4.1.5.7.3. Enable/Disable Breakpoint

A breakpoint can temporary disabled (and enabled) by the following methods:

- By clicking into the check box of the appropriate breakpoint within the “Breakpoints View” (see Figure 4.4, “The Breakpoints View” [30]
- Via the “Breakpoints View” context menu item Disable (and enabled via Enable) (see Figure 4.5, “The Breakpoints View context menu” [30]). The context menu commands are related to all selected breakpoints.

4.1.5.7.4. Conditional Breakpoints

You can assign a condition to a breakpoint via the “Breakpoints View” context menu item Properties (see Figure 4.5, “The Breakpoints View context menu” [30]). This opens a dialog (see Figure 4.8, “The breakpoint Properties dialog” [31]). Within the text field “Break Condition” you can set a condition. Don't forget to activate the conditional break via the check box “Enable Condition” at the bottom of the dialog.

For example, if you type “$a == 12” as condition, the program will break if the variable “$a” has the value “12”.

Figure 4.8. The breakpoint “Properties” dialog

4.1.5.7.5. Breakpoint Skipcounts

An additional feature to set for breakpoints is the “Skipcounts” condition. Normally the skipcounts is set to 0, which means the program breaks immediately if it scores a breakpoint. If the “Skiptcounts” is set to 1 it means, that the first score of a breakpoint is ignored (skipped) and the second score will lead to a break of the program.

The “Skipcounts” can be set via the same dialog box as the conditional breakpoints (see Section 4.1.5.7.4, “Conditional Breakpoints” [31]). For the “Skipcounts” the checkbox need not to be activated.

Note

The “Break Condition” has precedence over the “Skipcounts”.

4.1.5.8. Inspect Variables

Within the “Variables View” you can see all the variables with their values.

To open the “Variables View” within the “Debug Perspective” go to Window->Show View->Variables (see Figure 4.9, “Open the Variables View” [32]).

Note

Be sure that your current perspective is the “Debug Perspective”. Although you can open the “Variable View” within a different perspective, this doesn't make sense.

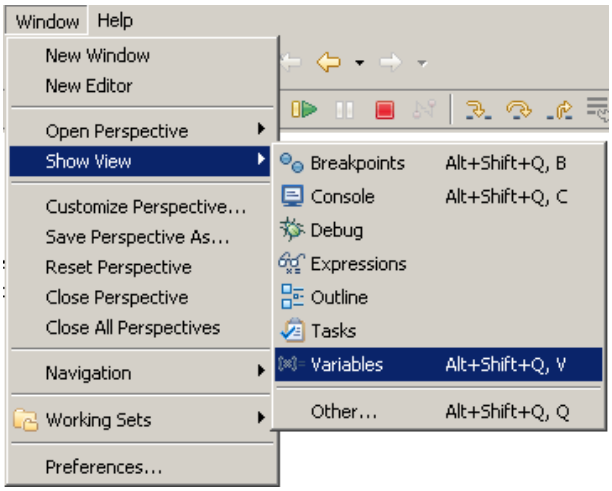

Figure 4.9. Open the “Variables View”

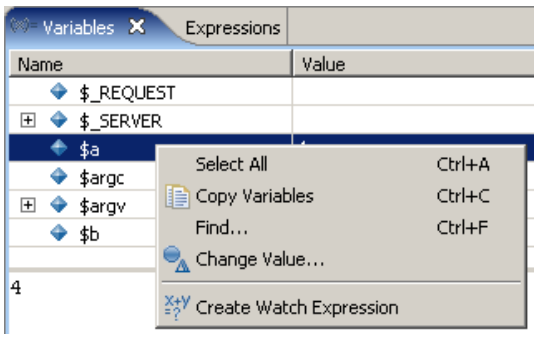

Figure 4.10. The “Variables View” context menu

4.1.5.9. Show Variable Value by Hovering

Besides inspecting the value of a variable by looking into the “Variables View” or the “Expressions View” a simple and quick method is to position the mouse pointer over the variable within the source code. Within a second the value and type of the variable is shown (see Figure 4.11, “Show variable value by hovering” [33]).

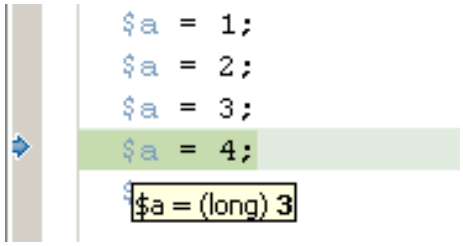

Figure 4.11. Show variable value by hovering

4.1.5.10. Watch Expressions

Within the “Expressions View” you can see the result of an expression. In the simplest form, this is just a variable and the corresponding value. But of course an expression can also be more complex, e.g. a mathematical formula.

There are different ways to open the “Expressions View” and to add an expression.

4.1.5.10.1. Open “Expressions View” via Main Menu

To open the “Expressions View” within the “Debug Perspective” go to Window->Show View->Variables (see ???).

Note

Be sure that your current perspective is the “Debug Perspective”. Although you can open the “Expressions View” within a different perspective, this doesn't make sense.

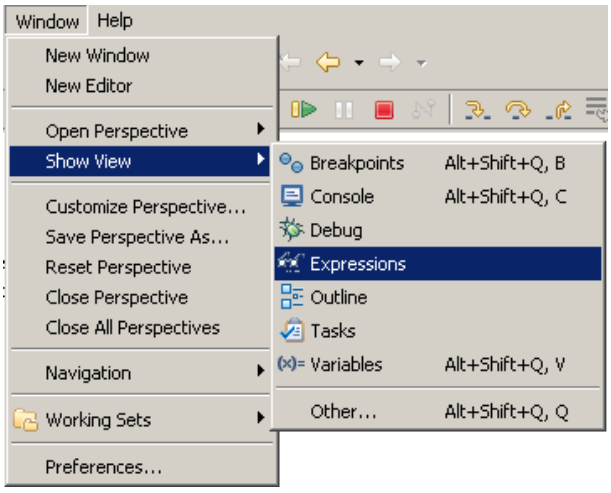

Figure 4.12. Open the “Expressions View” via the main menu

4.1.5.10.2. Open “Expressions View” via the “Variables View”

Right clicking on a variable within the “Variables View” opens the context menu. Then click on Create Watch Expression. This adds the selected variable to the “Expressions View” and opens/activates the “Expressions View”. Window->Show View->Variables (see Figure 4.9, “Open the Variables View” [32])

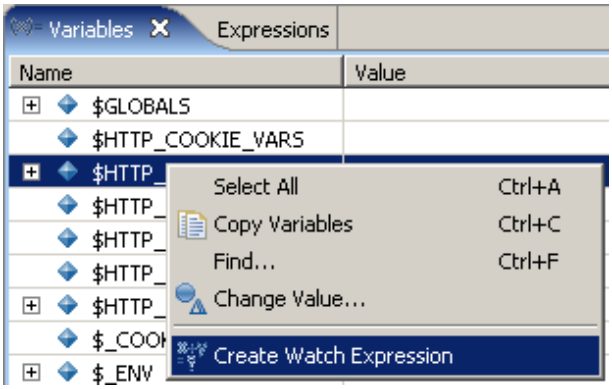

Figure 4.13. Open the “Expressions View” via the “Variables View” context menu

4.1.5.10.3. Open “Expressions View” via context menu

Unfortunately, yet this isn't implemented.

4.1.5.10.4. Remove a watch expression

An expression can be removed from the “Expressions View” by activating the appropriate watch expression and typing **Del** or via the context menu Remove.

4.2. Debugging PHPeclipse

Index