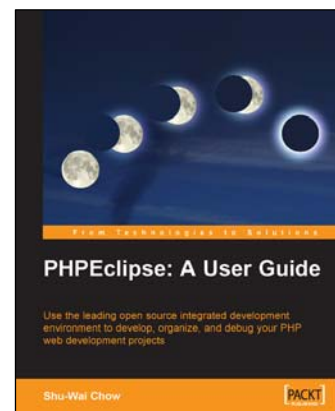




PHPEclipse: A User Guide

Shu-Wai Chow



Chapter 8 "Deploying Your Site"

In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter 8 "Deploying Your Site"

A synopsis of the book's content

Information on where to buy this book

About the Author

Shu-Wai Chow has worked in the field of computer programming and information technology for the past eight years. He started his career in Sacramento, California, spending four years as the webmaster for Educaid, a First Union company and another four years at Vision Service Plan as an application developer. Through the years, he has become proficient in Java, JSP, PHP, ColdFusion, ASP, LDAP, XSLT, and XSL-FO. Shu has also been the volunteer webmaster and a feline adoption counselor for several animal welfare organizations in Sacramento.

He is currently a software engineer at Antenna Software in Jersey City, New Jersey.

Born in the British Crown Colony of Hong Kong, Shu did most of his alleged growing up in Palo Alto, California. He studied Anthropology and Economics at California State University, Sacramento. He lives along the New Jersey coast with seven very demanding cats, three birds that are too smart for their own good, a cherished Fender Stratocaster, and a beloved, saint-like girlfriend.

For More Information: www.packtpub.com/phpeclipse/book

8

Deploying Your Site

We've walked through the complete process of web development using Eclipse, from development to code storage. The final part would be to deploy your site to a web server. Once again, Eclipse simplifies our work by including several tools that aid us in this process.

The key to deployment in Eclipse is the **export** function. Eclipse gives us many options in exporting our site. First, we will look at **FTP**, an old and common method of moving files. **WebDAV** is an interesting way to upload files using a web server. **Secure FTP (SFTP)**, a protocol similar to FTP, but encrypted, is enjoying immense popularity in this security-conscious age. FTP and WebDAV exports are provided through plug-ins as part of the PHPEclipse package. The **Klomp** plug-in gives us SFTP export capabilities, and comes bundled with PHPEclipse.

Finally, we will see how **Ant**, traditionally regarded as a Java tool, can help us in PHP deployment.

Setting Up a Test FTP Server

A server running FTP, SFTP, or WebDAV processes is a fairly common thing. However, if you do not have a server available for experimentation with Eclipse, you can easily set up these services. However, as of Eclipse 3.0/3.1, these clients do not work when connecting to a local server. You will have to set up a second machine on your local network to run these services. To help you, we will touch on how to quickly set up an FTP server.

Again, a word of caution—these instructions are designed to give us a crude but effective file server. It will neither be very secure nor optimized for performance. If you need a production FTP server, consult other resources on configuring your machine and security best practices.

To start off, decide on an upload area. This directory could be anywhere, but be sure you have *read* and *write* permissions to that directory.

Windows

On the Windows platform, you can place a copy of the XAMPP package on your server. XAMPP includes FileZilla, an easy-to-manage, open source FTP client and server package. First, make sure the FTP server is running by checking the XAMPP Control Panel. The status for FileZilla should be set to Running. Inside the main xampp directory there will be a FileZillaFTP directory.

For More Information: www.packtpub.com/phpeclipse/book

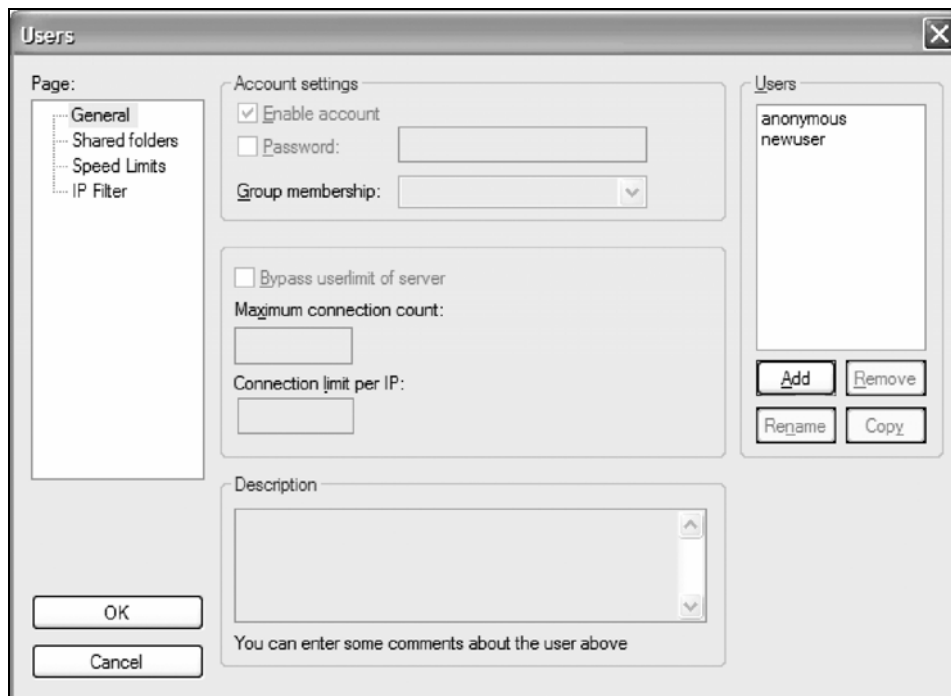
Deploying Your Site

In here, you'll find an executable named FileZilla Server Interface. This program provides a GUI front-end to any FileZilla FTP server, local, or remote. Launch this application. You'll be presented with a dialog box to choose your FTP server:

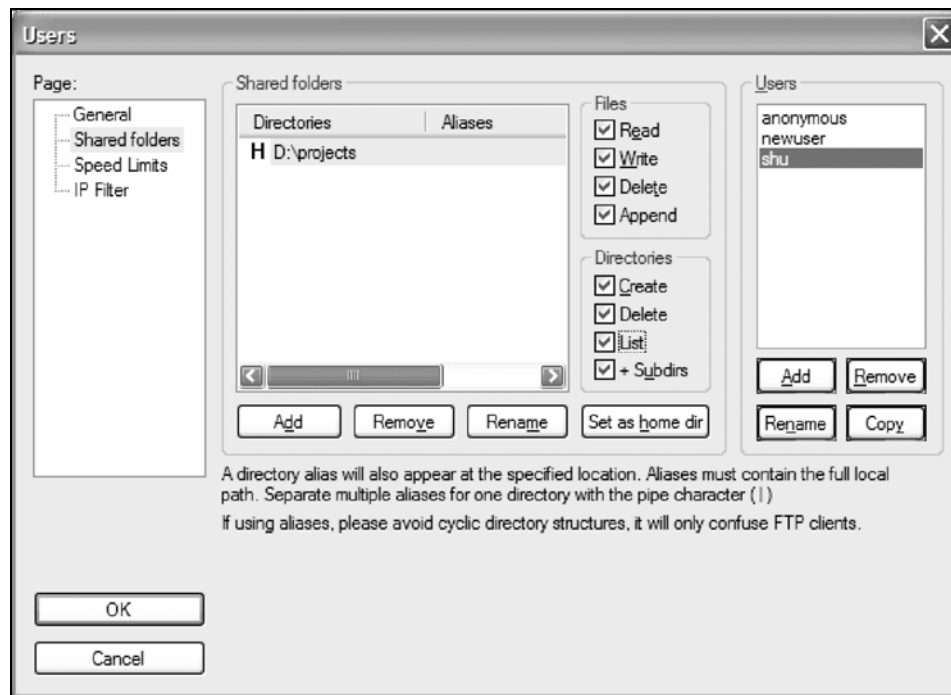


This application automatically lets you administer your local instance of FileZilla. Click on the OK button to accept the default options.

You'll be taken to the main administration screen, which is essentially a console for the FTP server. We need to add ourselves as an FTP user to this system, so click on Edit | Users to pull up the main Users administration screen.



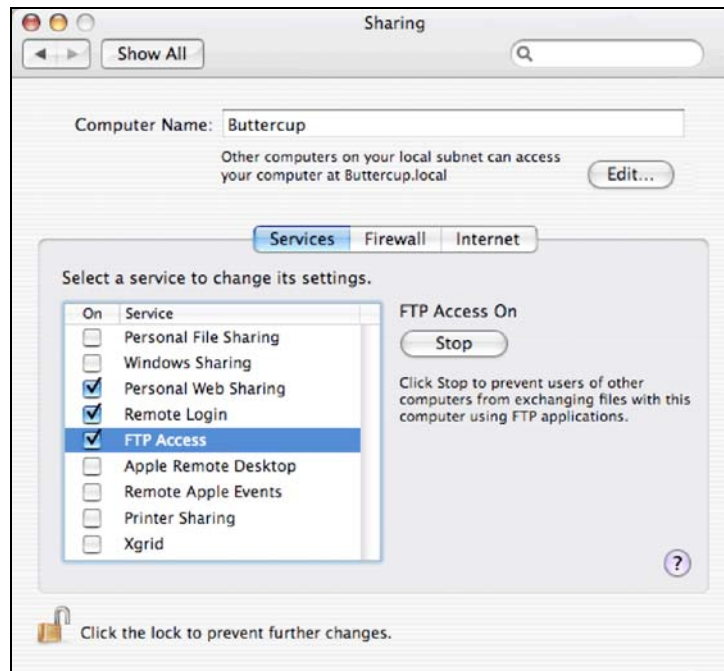
Under the Users listing, click on the Add button. Specify a login name. When we approve the addition, we'll be taken back to the Users administration screen. At this time, it would be prudent to add a password to our account in the Account Settings section. Next, we need to specify the home directory of our account. Make sure the newly added account is selected on the Users listing, and then select the Shared folders listing on the right, under Page.



In the Shared folders area, click on the Add button. Browse to a directory that you selected as your publishing directory. This should be the same directory as your Apache document root. If this is the first directory that you added through the FileZilla Server Interface, it will automatically receive an **H** icon next to the name in the Shared folders box. This designates that the directory is the home directory for the user, and at every logon, the user will automatically be re-routed to that directory after login. Before you click the OK button, make sure all the permissions checkboxes are checked for the publishing directory. FTP services run on TCP port 21, and FileZilla will open that port for use. If you are using Windows XP, a system dialog box might appear, asking whether you are sure you wish to unblock the port. Click the OK button to allow the unblocking.

Macintosh

Mac OS X includes an FTP/SFTP server built in. The server is also preconfigured for our needs, using the users and groups of the local machine. We won't have to do any extra setup. However, it is turned off by default. To turn on the server, go to System Preferences | Sharing. Make sure the checkbox next to FTP Access is checked.



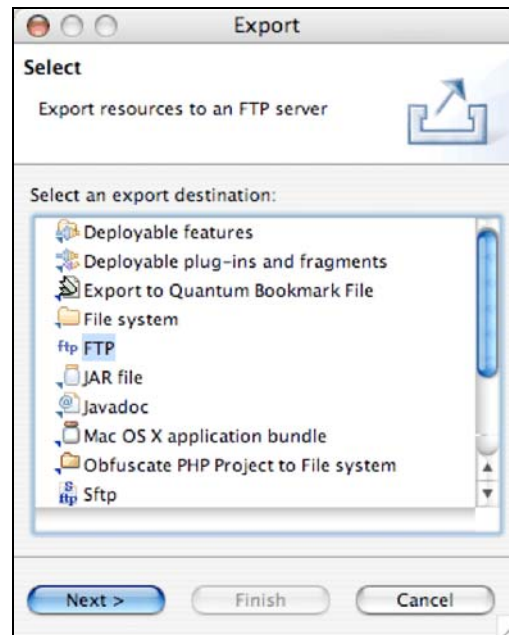
Linux

An FTP daemon is a basic component of a Linux server, and your distribution should have one installed or available to install. Unfortunately, it may not always be started and configured on your Linux distribution. Also, how you start and configure differs greatly by distribution. The GUI clients will differ if you are using Gnome or KDE desktops. Some distributions include a GUI client to administer FTP services, while others need to be administered by manually editing configuration text files and stopped and started via the command line. Consult your distribution's documentation for instructions.

FTP, SFTP, and WebDAV Export

FTP, SFTP, and WebDAV exports are very similar. They follow the same flow and use almost the same screens. To trigger an export, select the project in the Navigator view and click on the File | Export... menu option. This will give you a list of export options to select from.

The Klomp plug-in gives us the ability to export via SFTP. Unfortunately, as of this writing, it does not work with the latest version of Eclipse, version 3.1. If you absolutely need a secure way to transfer files and Klomp does not work with your version of Eclipse, you might consider using WebDAV over the `https://` protocol, or Ant using SFTP.

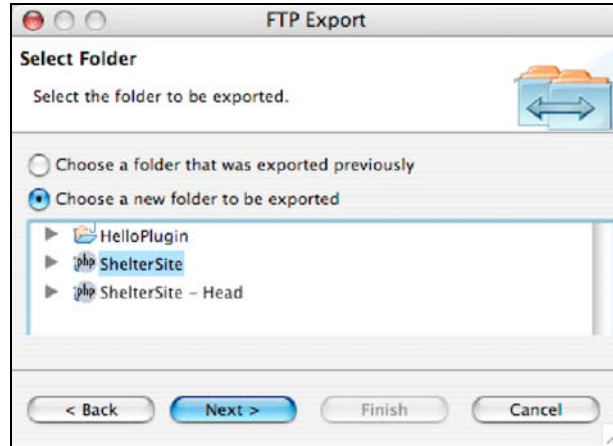


Most of the options in this list are installed by the JDT and are Java related. Some allow you to export settings to be shared between team members. A handful, like FTP, Sftp, and WebDAV, are actually related to moving source files to another area. Select FTP and click on the Next button to continue.

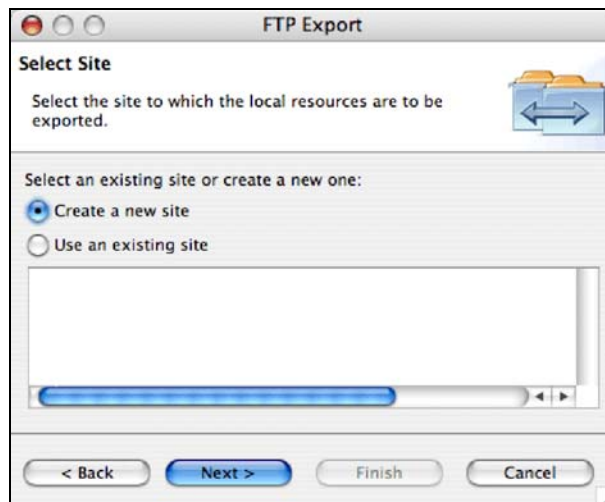
If you get an error message saying 'unable to load class' you will have to download drops containing FTP and WebDAV target management support plug-ins from the Eclipse site. Go to the Eclipse download site <http://download.eclipse.org/eclipse/downloads/> and click on your version; you will find the drops for FTP and WebDAV Support on that page. Download and copy all the contents of the features and plugins directories into the features and plugins directories of your Eclipse installation.

Deploying Your Site

The next screen will ask if you wish to export a project that you previously exported via FTP:

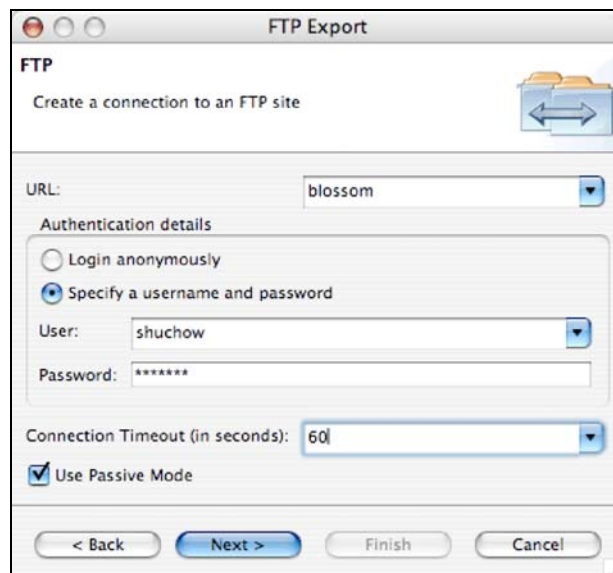


When you are first presented with this screen and if you have not done this before, the text area will be empty. Click on the Choose a new project to be exported radio button and you will be presented with a list of all the projects in your workspace. Select the ShelterSite project and click the Next button to continue. Eclipse will create a deployment mapping for this project. When you export again in the future, Eclipse reads this mapping to the server settings associated with a project, saving you a few of the following steps to choose a server.



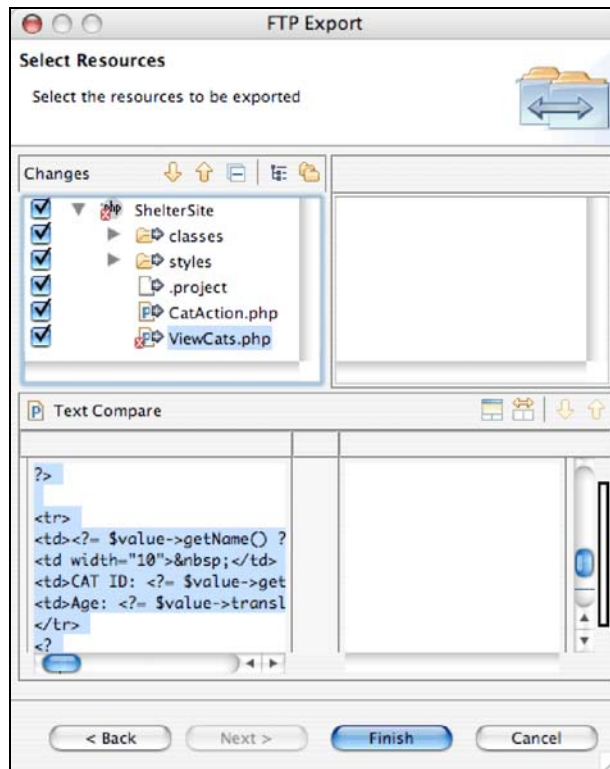
Similar to the last screen, this screen will ask if you want to export the project to a location to which you previously exported, or if you want to export to a new location. Again, if you have never exported via FTP before, the list area will be blank. Choose Create a new site and click on Next, this will launch the Create a connection to an FTP site dialog box.

All of your FTP, SFTP, and WebDAV locations on this screen can be edited and deleted from the Site Explorer view. Access this view by going to the Window | Show View... | Other... | Target Management | Site Explorer menu option.



Enter your server connection settings here. Click the Next button to continue. If you are using WebDAV or SFTP, this screen will be different. However, all three will essentially ask you for the same details, like server name and authentication.

On the final screen in the export wizard, you will have a chance to decide which files you want to upload. You can select individual files, or upload the entire project into the remote directory by checking the correlating checkbox next to the item. If the site has been previously uploaded, you can also do a line-by-line code comparison of the project.



After you are done analyzing the code, click on the Finish button. The upload process will begin at this point. After this step, your site will be uploaded to the remote server, and you will be taken back to the development perspective.

Using Ant for Deployment

You may have heard of **Ant** in the Java development world. Being a pure Java tool, Ant is often called 'the Java version of make'. With Ant, you create scripts called **build files** that are interpreted by the **Ant parser**. These scripts aid you in Java development by compiling your code and deploying it in your build directory. For both small home-grown projects and large Java enterprise environments, Ant has become an absolutely critical tool for Java developers. In the PHP world, though, we do not have any code to compile, and often, FTP is adequate for moving files into production. Why, then, as PHP developers, do we care about Ant?

In some business environments, production web servers are tightly controlled. Developers are not allowed anywhere near the servers, let alone pushing out code at their whim. The sheer act of deploying new code often occurs only after a long ritual of meetings and approvals at various levels. In these environments, deployment occurs with a script that automates the delivery of an application from one area to the production site. Ant scripts can do this job for you for applications in any language.

Even if you are not working in such an environment, why would you use Ant? A key benefit is Ant's integration with CVS and Subversion. Using Ant, you can write a script to grab all files in the repository with a certain tag and FTP them to a production server. Imagine the time saved and reduction in human error with this. In a large application with PHP class files in one directory, HTML view files in another, JavaScript files in their own separate directory, and stylesheets in yet another, a deployment involves a huge effort and it is difficult to keep track of what has changed.

At deployment time, you have to check out all the correct files from CVS or Subversion and then FTP all of them again to the production web server. If all you have to do is tag a file at the end of testing, you have drastically reduced the chance that you'll miss something when retrieving from the code repository or FTPing.

Finally, Ant scripts are easy to write. Being XML-based, these scripts are fairly simple and intuitive. If we were not concerned with code cleanliness, it would take only one line of code to check out a project from the code repository, and another line to FTP files. The reference documentation is excellent. If you require any more support, Ant is supported by a large user base and online community. Being an Apache Foundation project doesn't hurt its popularity either.

Eclipse integrates Ant smoothly into our development environment. Included with the JDT, Eclipse has a nice Ant editor, templates, and tag help. You can execute them directly from Eclipse. Let's walk through an example using CVS.

The official Ant release does not support Subversion. Luckily, Ant employs an extensible architecture like Eclipse. Tigris.org, the developer of Subversion, has created an Ant task called **SvnAnt** that allows Ant to interface with Subversion repositories. More information about this project is available at <http://subclipse.tigris.org/svnant.html>.

Setting up Ant for FTP

Before we get started, we need to install some additional files for Ant. FTP ability is an optional **Ant task** that is not included with the default installation of Eclipse. We need to download these files and add them to our Ant classpaths in Eclipse.

The steps required are:

1. Download and unzip the latest binaries of `commons-net.jar` and `commons-oro.jar`.
2. Install the `commons-net` and `commons-oro`.
3. Add these new files to the Ant classpath.

Downloading

You can find the `commons-net.jar` and `commons-oro.jar` files at http://jakarta.apache.org/site/downloads/downloads_commons-net.cgi and http://jakarta.apache.org/site/downloads/downloads_oro.cgi respectively.

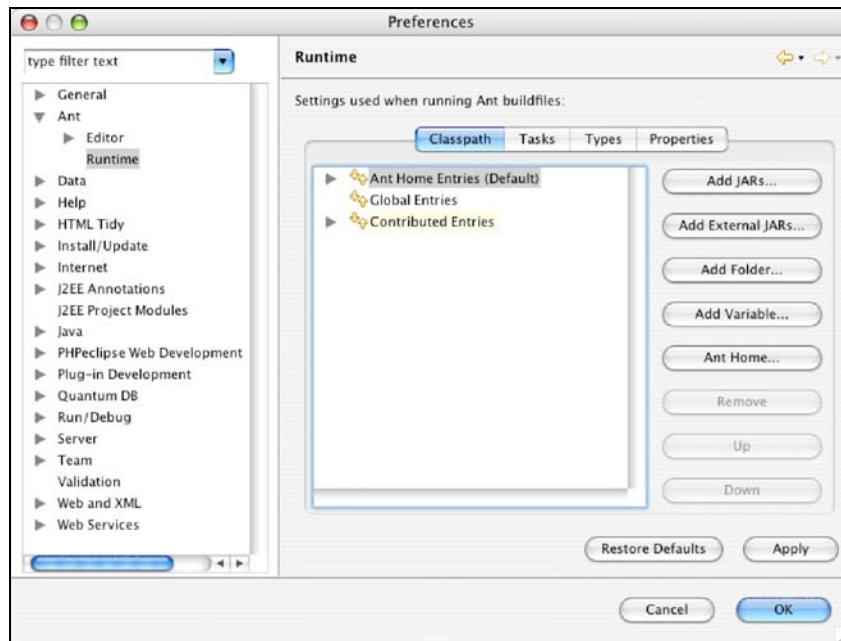
You will need at least version 1.4.0 for `commons-net.jar` and at least version 2.0.8 for `commons-oro.jar`. When you expand the downloaded files, the JARs should have the version numbers as part of the file names, like `commons-net-1.4.0.jar`. This is perfectly acceptable and there is no need to change the file names.

Installing

Place these files in a clear location. It is recommended to create a new directory under the Eclipse installation folder called `antjars`. If you explore the Eclipse installation directory, you will find that the Ant plug-in is stored in its own folder under the `plugins` directory. This folder contains the original `commons-net.jar` and `commons-oro.jar`. These are the JARs that do not have our necessary FTP classes. You could replace these files, but it is generally not a good idea to fiddle with the default files. Therefore, we're going to place our replacement JARs outside the area having the default Eclipse-installed files. This way, Eclipse can still manage its own installed files, and if we upgrade, we don't lose any functionality because the user settings remember our classpath changes.

Adding Files to the Ant Classpath

By naming these files in the Ant classpath, we are telling Eclipse of their existence, and that critical JARs may be found in them. To add them to the classpath, go to the `Window | Preferences | Ant | Runtime` menu option. Click on `Ant Home Entries (Default)` in the list under `Classpath`. Click on the `Add External Jars...` button.



Select the two new JARs that you just downloaded. Click on the OK button. Our installation of Ant is now FTP enabled.

Creating Our Sample Ant Build File

In this demonstration, we will make a script to actually do what we have described. Our script will log into CVS, pull files from the Head of our project, put the files locally in a temporary directory, FTP the files into a web server, and finally do some cleaning up by deleting our temporary directory.

Generally, Ant scripts are composed of targets. Targets are basically groupings of commands called tasks. In order to be functional, each target contains at least one Ant task. These tasks are listed and explained in the official Ant documentation. A target might contain the task of compiling a Java program. At the beginning of the Ant file, we ordain one target as the special default target. Normally, when you run an Ant file, you pass it the target that you want to execute. If it doesn't have one, it will execute the default target.

We are going to create an Ant file with four targets. The first will check out the project from CVS into our local machine. The next will upload the files to a web server. The third target will delete the copy of the application that we downloaded from CVS. Our default target sits above these three. Its job will be to call the previous three targets in the correct order.

First, create a file in our project called `build.xml`. This is the standard name for Ant build files. This script has already been included in the sample code, so you do not have to type everything in manually. However, we will walk through the build file in its entirety and look at it section by section. You may want to pull up the complete file to follow along.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Deployment Script"
        default="startPublish"
        basedir=".">
```

This is the beginning of the Ant file. We include in a proper XML declaration here for good form. Ant files begin and end with a `<project>` tag. The important part of this tag is the `default` attribute. The `default` attribute tells the Ant engine which target to run if one is not specified when the build file is executed from the command line. You must declare this target later on in the file. While not officially required, Eclipse needs it to run build files because it automatically fires build files as-is. In other words, Eclipse does not prompt you for a target. Further, it is good coding practice to include a default.

```
<target name="startPublish">
  <antcall target="getFilesFromCVS" />
  <antcall target="startFTP" />
  <antcall target="cleanUp" />
</target>
```

The code snippet shown above is our first target, named `startPublish`. We separate our deployment process into three separate tasks—check the files out from CVS into our temporary area, FTP them to our server, and finally clean up our temporary area. The `startPublish` target merely calls the other tasks in the correct order. Calling other targets is done by the `antcall` tag, which includes a mandatory target to call defined by the `target` attribute.

```
<target name="getFilesFromCVS">
  <cvs cvsRoot=":local:/var/lib/cvsroot"
      package="ShelterSite"
      dest="/tmp" />
</target>
```

Deploying Your Site

This is our second target, `getFilesFromCVS`. There is only one task, `cvs`. This task pulls files from a CVS repository. The `cvsRoot` attribute defines the repository based on the CVS connection string. We specify a package to check out in the `package` attribute, and the `dest` attribute specifies the local directory where we want to place the checked out files.

We briefly talked about CVS connection strings in the previous chapter. Here, we use the local connection method to connect to our local CVS server. This method is simple and easy, but only works on a CVS repository that is on our machine. Consult your CVS administrator or CVS documentation on how to construct a proper connection string for your server setup.

```
<target name="startFTP">
  <ftp server="127.0.0.1"
        user="shuchow"
        password="TopSecretPassword!"
        remoteDir="/Library/WebServer/Documents/test/"
        action="send">
```

The interaction with the FTP server is contained in the third target. First, we enter our connection settings in the appropriate attributes within the `ftp` tag. The attributes `server` and `user` are your account credentials on the FTP server. While the FTP and SFTP export plug-ins in Eclipse do not work with local machines, Ant uses its own FTP client, and thus, can send files to a local FTP server. `remoteDir` is the attribute that tells Ant the location of the remote directory on which we will be performing our actions. The `action` attribute tells Ant what to do on that directory. All our standard FTP commands, for example, `get`/`put`, are available via the `action` attribute.

```
<fileset dir="/tmp/ShellSite" id="id">
```

Nested in the `ftp` element is the `fileset` tag. The `dir` attribute specifies the local root directory we wish to use. Combined with the previous `ftp` element's `action` and `remoteDir` attributes, Ant will send this local directory to the remote directory.

You may have created this build file underneath the project directory and checked it into CVS. This is not a bad thing because you now have a history of a file that is a critical piece of the project. However, it is a bad thing if it gets FTPed into the production web server, especially if you have FTP server passwords! We should also prevent Eclipse from uploading its own `.project` file to the web server.

```
<exclude name="**/build.xml" />
<exclude name="**/*.project" />
```

To exclude the build and `.project` files from the upload, we use `exclude` elements for these two files nested within `fileset`. There is a corresponding `include` element to specify the inclusion of files. For both elements we name our file using the `name` attribute. Ant includes a powerful pattern-matching engine for use in `include` and `exclude`. In our example, the first `exclude` tells Ant to exclude anything named `build.xml` from the upload process. The second tells Ant to exclude anything ending with `.project` from the upload process. Be aware that the order of `include` and `exclude` tags is important in a `fileset` element, with the lower `include/exclude` taking precedence over previous ones in case of conflict. For example, if you tell Ant to include a directory, you can name specific files to exclude within that directory by placing the `exclude` tags after the directory `include` tag.

```

    </fileset>
  </ftp>
</target>

```

Finally, we close out the `fileset`, `ftp`, and `target` elements in this block.

```

    <target name="cleanUp">
      <delete dir="/tmp/ShellterSite"></delete>
    </target>
  </project>

```

At the end of the file, we clean up our temporary directory. We do this with our fourth target. The only task in here is a `delete` task. It identifies which local directory we want to delete with a `dir` attribute. Lastly, we close our `project` element. Our build file is complete. It's time to run our file.

Running an Ant Script

To run an Ant Script, select the `build.xml` file in the Navigator view. Select the `Run | External Tools | Run As | Ant Build` menu option. Eclipse will automatically trigger the build file. The results of our execution will be output in the Console view.

```

Buildfile: /Library/WebServer/Documents/ShellterSite/build.xml
startPublish:
getFilesFromCVS:
[cvs] cvs checkout: Updating ShellterSite
[cvs] U ShellterSite/.project
[cvs] U ShellterSite/CatAction.php
[cvs] U ShellterSite/ViewCats.php
[cvs] cvs checkout: Updating ShellterSite/classes
[cvs] U ShellterSite/classes/clsCat.php
[cvs] U ShellterSite/classes/clsCatView.php
[cvs] U ShellterSite/classes/clsDatabase.php
[cvs] U ShellterSite/classes/clsPet.php
[cvs] cvs checkout: Updating ShellterSite/styles
[cvs] U ShellterSite/styles/shellter.css
startFTP:
[ftp] sending files
[ftp] 7 files sent
cleanUp:
[delete] Deleting directory /tmp/ShellterSite
BUILD SUCCESSFUL
Total time: 8 seconds

```

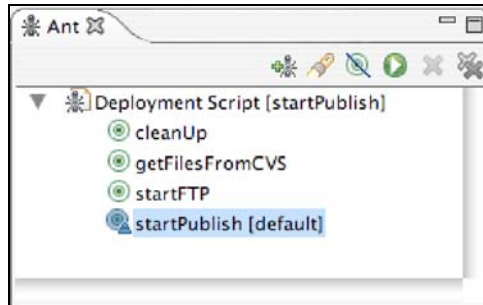
If there are any errors during the execution, they will also output here to help you troubleshoot.

Avoid Putting FTP Passwords in Build Files

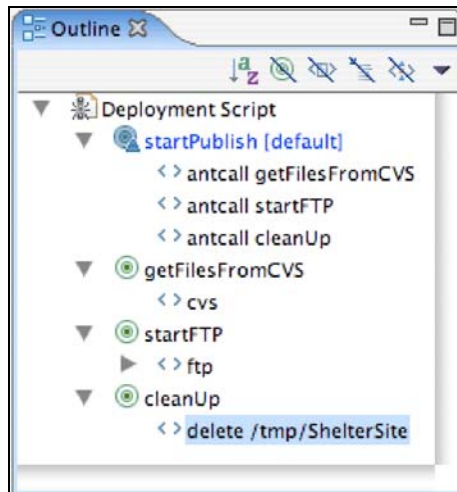
You may want to avoid putting passwords in build files, since they are simple text files. To do this, reference them with a dollar sign and bracket: `${ftpPassword}`. When you execute the file, pass in an argument. To pass Ant arguments in Eclipse, go to the `Run | External Tools | External Tools...` menu option. Ant files have configuration profiles much like debugging configurations. This window will pull up the configuration for a particular Ant file. Highlight the build file you wish to work with and in the Main tab's Arguments area, type in `-DftpPassword="YourSecretPassword"` where `ftpPassword` is the name of the variable you specified in the build file and the enclosing quotes hold your password. Note the dash at the beginning of the argument and that there is no space after the `D` flag and the variable name.

Ant Tools

While Ant does not have its own perspective in Eclipse, it does have its own view and leverages the existing Outline view. Both are very helpful in developing Ant files for your projects.



The Ant view allows you to manage Ant build files in your workspace. It offers an overview of the build file and all its targets. You can add other build files to this view, execute them, and delete them via the icons in the toolbar.



When you are editing an Ant build file, the Outline view will give you structural information about your file. Organized by targets, the view gives you information on tasks and important parameters in the build file.

Summary

In the final step of our development, Eclipse also helps by providing tools to help with the deployment of our site. Using the FTP and WebDAV export plug-ins of the JDT, and the Klomp SFTP plug-in, we can directly push a site to a web server. In more controlled environments, we can automate this process by creating Ant build files. While writing Ant build files may involve more work initially, we save time in the long run because Ant can automate the tedious movement of deploying files, automatically grab the source files for us in CVS, and reduce human error in the process. Eclipse also helps us in creating Ant files with a built-in Ant editor and tools to execute and manage Ant build files. Initially built for Java, the use of Ant is just another example of the flexibility of Eclipse for all development, including for PHP-driven sites.

PHPEclipse: A User Guide

The PHP language has come a long way from its humble roots as a set of Perl scripts written by Rasmus Lerdorf. Today, PHP enjoys enormous market share and the latest release, PHP 5, sports a robust object-oriented programming model. Naturally, development practices have also matured. Those of us who taught ourselves PHP in the late nineties have become more sophisticated in our coding techniques. PHP has also made significant headway into corporate environments. Both changes have led to a demand for tools that make development easier, faster, and more integrated with other systems such as databases and version-control tools.

Our tool selections, however, have historically been one of two extremes. On one hand are the editors. Fundamentally, these are text editors with basic development tools slapped on. While affordable, they lacked features that made them a true integrated development environment (IDE). To get these features, we had to purchase powerful and expensive IDEs. Even then, our choices were limited to NuSphere's PhpED or Zend Studio.

Things began to change in 2001. IBM released Eclipse, a powerful Java IDE, as an open source project. Developers saw the potential of Eclipse's extensible, plug-in-based architecture. Thanks to this community, Eclipse soon became much more than an editor and spoke many more languages than just Java. In 2003, a team of developers released the PHPEclipse plug-in. Finally the gap between PHP and Eclipse was closed. Developers now have a free and powerful IDE for PHP development.

In this book, we will explore using Eclipse for PHP web development using the PHPEclipse plug-in. We will take a tutorial-style approach throughout most of this book. Installation and setup walkthroughs are provided. Features of Eclipse and PHPEclipse that are helpful for PHP development will be explained.

What This Book Covers

This book is organized to get you quickly up and running with Eclipse for PHP development. The beginning chapters cover the basics of Eclipse, and then we move on to writing PHP code in Eclipse. From there, we move to more advanced features that are helpful, but not essential for PHP development, like source-code control and database querying.

Chapter 1 covers Eclipse's history and its architecture, and introduces PHPEclipse.

In *Chapter 2*, we install the necessary core software for developing applications in PHPEclipse—Apache, PHP, Java, Eclipse, and PHPEclipse.

Chapter 3 explains the feature of the Eclipse interface and how to customize it.

Chapter 4 is where we start writing PHP code. We will go through creating a project and examine in depth the features available in PHPEclipse for PHP development.

In *Chapter 5*, we debug our application. We will explain debugging terms and concepts, and how Eclipse debugs. This chapter covers the installation and setting up of the DBG debugger.

For More Information: www.packtpub.com/phpeclipse/book

In *Chapter 6*, we set up the Quantum DB plug-in and learn how to use it to manipulate databases. We will also install a JDBC driver and connect to it using the Quantum DB plug-in.

In *Chapter 7*, we explore the CVS integration of Eclipse. We will show how to manage and store a project completely in CVS as well as explain general CVS and versioning concepts.

Finally, in *Chapter 8*, we publish our website to a web server. We will use Eclipse's Update Manager to add an FTP client functionality.

Where to buy this book

You can buy *PHPEclipse: A User Guide* from the Packt Publishing website:

<http://www.packtpub.com/phpeclipse/book>.

Free shipping to the US, UK, Europe, Australia, New Zealand and India.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.



www.PacktPub.com

For More Information: www.packtpub.com/phpeclipse/book