Writable Directories and Zend Deployment

Issue

When an application is deployed to Zend Server, the target directory is not writable for the web server user. This is not a bug, but a design decision, based on general security considerations. Of course, in some use cases this can be considered a limitation. This article suggests some methods to workaround this "limitation".

Environment

Zend Server, mostly GNU / Linux systems.

Resolution

Method 1 - Using a Designated Writable Directory

This method is the simplest in implementation and general maintainability. It is very convenient, if you need the write permissions for application log and/or cache. It has the additional advantage of having the log/cache files in a predictable location, which is especially important, if you have some additional log aggregation or cache manipulation system in place.

The summary is simple - modify your applications code, so that it writes to an absolute path, instead of a path relative to application's root, for example, to */var/log/MyApp*. Of course, you need to set the correct write permissions upfront.

Method 2 - Setting the Correct Permissions During Deployment

This method is very versatile and will work on any system, even if it has not been prepared for your application. Unfortunately, the writable directory location is less predictable with this method - it will end up somewhere in */usr/local/zend/var/apps.*

To implement this method, you need to usethe hook scriptsin your application's package. For example, you can use code like this in your "post_stage.php" script:

```
$appDir = getenv("ZS_APPLICATION_BASE_DIR");
chmod("$appDir/logs", 0777);
```

Method 3 = Method 2 + Method 1

There are several possible combinations of the methods above. The most popular combination is defining a fixed writable directory during the deployment process itself. This gives both predictability of the directory's path and the advantage of making sure that the system is configured correctly.

There is probably no need to try and cover as many scenarios as possible. Here is one example:

1. Add deployment parameters to your package (in "deployment.xml"):

```
<parameters>
   <parameter display="Log File" id="logfile" readonly="false" required="true" type="string">
      <defaultvalue>/usr/local/zend/var/log/MyApp.log</defaultvalue>
   </parameter>
   <parameter display="Cache Directory" id="cachedir" readonly="false" required="true" type="string">
      <defaultvalue>/usr/local/zend/tmp/MyApp_Cache</defaultvalue>
   </parameter>
</parameters>
```

2. Prepare these resources after the staging phase of deployment (in "post_stage.php"):

```
$cacheDir = getenv("ZS_CACHEDIR");
$logFile = getenv("ZS_LOGFILE");

mkdir($cacheDir, 0777, true);
file_put_contents($logFile, " ", FILE_APPEND);
chmod($logFile, 0777);
```

3. Finally, create symbolic links in the application's directory (same file - "post_stage.php"):

```
$appDir = getenv("ZS_APPLICATION_BASE_DIR");
symlink($cacheDir, "$appDir/cached_data");
symlink($logFile, "$appDir/application.log");
```