Using PHP with JAVA

By Harish Kamath

Table of Contents

Extending Yourself

One of the coolest things about PHP has to be the wide range of extensions available for the language. Having problems communicating with a particular database? PHP has a wide range of extensions for different databases. Want to process credit cards transactions? PHP includes an extension that allows you to integrate CyberMUT on your Web site. Want to parse and transform XML documents? PHP comes with a built−in SAX parser and XSLT engine.

The moral of the story? You can do a lot of very interesting things with PHP. And one of the more interesting ones includes hooking PHP up to Java and accessing Java classes in a PHP script. Which, coincidentally, happens to be just what this article is about.

Getting Started

PHP doesn't come with Java support turned on by default, so you'll need to recompile it to enable this support. You can do this by adding the "−−with−java" compile−time parameter to the PHP "configure" script. Note that you'll also need a copy of the latest JDK − you can get this from http://java.sun.com/ (the examples in this article use version 1.3.0 of the JDK).

If you're using Windows, you've already got a pre−built Java extension with your PHP distribution − all you need to do is enable it. First, make sure that your system's PATH variable includes the path to the JDK − this can easily be accomplished by altering the PATH variable in your "autoexec.bat" file.

Next, pop open the "php.ini" configuration file, and skip over all the cryptic configuration commands to the "Java" section. There, set values for the following variables:

java.library − this refers to the location of the Java Virtual Machine file (jvm.dll), usually located in the JRE directory;

java.library.path − this refers to the location of PHP's Java extension;

java.home − this refers to the JDK's "bin" directory;

java.class.path − this refers to the Java CLASSPATH, which contains your custom Java classes. You should make sure that this variable always includes the location of the "php_java.jar" file (usually the /java/ directory)

While you're editing this file, also drop by the "Windows Extensions" section of "php.ini" and uncomment the "php_java.dll" extension.

Finally, check to make sure that the file "php_java.dll" is in the /extensions/ sub−directory, restart your Web server, and you're ready to roll!

Rank And File

Let's begin with a simple example to verify that everything is working as advertised. This script uses built−in Java methods to check for the existence of a particular file on your system and display its size.

```
<?php

$fp = new Java("java.io.File", "/home/john/test.txt");

if($fp−>exists())
{
echo "The file ". $fp−>getAbsolutePath() . " is ".
$fp−>length()
. " bytes"; }
else
{
echo "The file " . $fp−>getAbsolutePath() . " does not exist";
}

?>
```

This is fairly simple, and should be clearly understandable to anyone who knows basic Java programming. The first line of the script instantiates an object of Java's File class, and stores it in the PHP variable $fp.

```
$fp = new Java("java.io.File", "/home/john/test.txt");
```

In case you're wondering, it's possible to pass parameters to the class constructor by specifying them as arguments when creating the object. In the example above, the second argument to the class constructor contains the location of the file to be displayed.

Once the object has been instantiated, it becomes possible to access the methods and properties of the Java class using regular OOP syntax.

```
if($fp−>exists())
{
echo "The file ". $fp−>getAbsolutePath() . " is ".
$fp−>length()
. " bytes"; }
else
{
echo "The file " . $fp−>getAbsolutePath() . " does not exist";
}
```

In this case, the exists(), getAbsolutePath() and length() methods are all methods of the File class; however, I've managed to use them in a PHP script by virtue of the Java connectivity built into PHP.

Here's what the output of the script above looks like:

```
The file /home/john/test.txt is 385 bytes
```

A Custom Job

Now, the example on the previous page used a built−in Java class to demonstrate PHP/Java connectivity. It's also possible to instantiate and use a custom Java class in a PHP script. I'll demonstrate this by encapsulating the functionality of the core File class in my own FileReader class.

Here's the code for my custom class:

```
import java.io.*;

public class FileReader
{

public File LocalFile;

// set file location
public void loadFile(String FileName)
{
LocalFile = new File(FileName);
}

// return file size
public long getFileSize()
{
return LocalFile.length();
}

// check if file exists
public boolean FileExists()
{
return LocalFile.exists();
}

// return file path
public String getFilePath()
{
return LocalFile.getAbsolutePath();
}

}
```

There's no rocket science here − this class is only a wrapper for core File class methods. However, it will serve to demonstrate the basics of using a custom Java class in a PHP script.

Now, compile the class, copy the compiled code to your Java CLASSPATH, and write a script to use it.

```
<?php

$myClass = new Java("FileReader");

$myClass->loadFile("/home/john/test.txt");

if($myClass->FileExists())
{
echo "File size is " . $myClass->getFileSize() . "<br>";
echo "File path is " . $myClass->getFilePath();

}
else
{
echo "Sorry, the file " . $myClass->getFilePath() . " could
not
be found"; }

?>
```

This is similar to the first example, except that, this time, I've used my own custom Java class in the script. Once an instance of the class has been instantiated, class methods can be accessed in the normal manner.

Here's what the output looks like:

```
File size is 385
File path is /home/john/test.txt
```

Next up, passing one Java object to another.

Passing The Parcel

This next example demonstrates how one object instance can be passed to another in a single PHP script. Here's the code:

```
import java.io.Writer;
import java.io.IOException;
import java.lang.Exception;

public class ReverseString
{
public Writer out;

public void setWriter(Writer out)
{
this.out = out;
}

public void reverse(String string) throws Exception
{
try
{
// reverse the string here
char tempArray[] = new char[string.length()];

for (int i = 0; i < string.length(); i++)
tempArray[i] =
string.charAt(string.length() - i - 1);
String reversedString = new String(tempArray);
out.write(reversedString);

}
catch (IOException e)
{
// catch IO exceptions
throw new Exception("Something bad
happened");

}
catch (Exception e)
{
// catch everything else
throw new Exception("Something really
bad happened");
}

}
```

```
}
```

This class uses a Writer object for script communication; this Writer object is instantiated via the setWriter() class method.

```
public void setWriter(Writer out)
{
this.out = out;
}
```

Next, the class method reverse() accepts a string, reverses it and returns it via the Writer object.

```
public void reverse(String string) throws Exception
{
try
{
// reverse the string here
char tempArray[] = new char[string.length()];

for (int i = 0; i < string.length(); i++)
tempArray[i] =
string.charAt(string.length() - i - 1);
String reversedString = new String(tempArray);
out.write(reversedString);

// snip
}


}
```

Compile this class, copy it to your Java CLASSPATH, and write a simple PHP script to access it.

```
<?php

// create an instance of the ReverseString object
$obj = new Java("ReverseString");

// create an instance of the StringWriter object
$writer = new Java("java.io.StringWriter");

$obj->setWriter($writer);
$obj->reverse("The cow jumped over the moon");
```

```
echo $writer->toString();
$writer->flush();
$writer->close();

?>
```

In this case, I've instantiated two objects, one for the ReverseString class and the other for the StringWriter class. Next, I've passed the StringWriter object, as stored in the PHP object $writer, to the ReverseString class via the class' setWriter() method, and then used the reverse() method to reverse a string and return the result.

Here's what the output looks like:

```
noom eht revo depmuj woc ehT
```

An Exceptionally Clever Cow

Next, let's look at exception handling. If you go back to the example on the previous page, you'll see that the Java class includes a couple of exception handlers, one for IOExceptions and one for everything else. Let's now modify the PHP script so that it includes some basic exception handling.

```
<?php

// create an instance of the ReverseString object
$obj = new Java("ReverseString");

// create an instance of the StringWriter object
$writer = new Java("java.io.StringWriter");

$obj->setWriter($writer);
$obj->reverse("The cow jumped over the moon");

// get the last exception
$e = java_last_exception_get();

if ($e)
{
// print error
echo $e->toString();
}
else
{
echo $writer->toString();
$writer->flush();
$writer->close();
}

// clear the exception
java_last_exception_clear();
?>
```

This version of the script uses two built-in PHP functions to check whether or not any exceptions were raised during the execution of the class methods. If any exceptions were raised, they would be stored in the $e PHP variable, and printed via a call to the toString() method.

If you'd like to see how an exception is handled, modify the PHP script above to look like this:

```
<?php

// create an instance of the ReverseString object
```

```
$obj = new Java("ReverseString");

// create an instance of the StringWriter object
$writer = new Java("java.io.StringWriter");

// deliberately introduce an error by commenting out the next
line //
$obj->setWriter($writer);

// suppress errors here
@$obj->reverse("The cow jumped over the moon");

// get the last exception
$e = java_last_exception_get();

if ($e)
{
// print error
echo $e->toString();
}
else
{
echo $writer->toString();
$writer->flush();
$writer->close();
}

// clear the exception
java_last_exception_clear();
?>
```

Here's what the output looks like:

```
java.lang.Exception: Something really bad happened
```

Of course, this simple and graceful error message gets a little more unfriendly if you remove the @ error-suppression operator.

```
Warning: java.lang.Exception: Something really bad happened in
/usr/local/apache/htdocs/java/exception.php on line 10
java.lang.Exception: Something really bad happened
```

Beanie Baby

Finally, how about connecting PHP up to a Java Bean? Here's the Bean, a simple Celsius−to−Fahrenheit−to−Celsius converter:

temperature.zip

The Bean exposes the following methods:

getCelsius() − get the current value of the Celsius property

getFahrenheit() − get the current value of the Fahrenheit property

setCelsius(num) − set the current value of the Celsius property to num

setFahrenheit(num) − set the current value of the Fahrenheit property to num

convertCelsiusToFahrenheit(value) − convert Celsius value to Fahrenheit

convertFahrenheitToCelsius(value) − convert Fahrenheit value to Celsius

And here's a PHP script which combines user input with the Temperature Bean to perform temperature conversion:

```
<html>
<head><basefont face="Arial"></head>
<body>
<h2>Temperature Converter</h2>
<?php

if($submit)
{

// data type conversion
setType($temp,"integer");

$myClass = new Java("Temperature");

if($units == "celsius" &$temp != "")
{

// use the Celsius functions
$myClass->setCelsius($temp);

// print result
echo $myClass->getCelsius()," <sup>o</sup> Celsius is
",$myClass->convertCelsiusToFahrenheit($myClass->getCelsius()),"
```

```
<sup>o</sup> Fahrenheit.";

}
else if($units == "fahrenheit" &$temp != "")
{

// use the Fahrenheit functions
$myClass->setFahrenheit($temp);

// print result
echo $myClass->getFahrenheit() , " <sup>o</sup>
Fahrenheit is ",
$myClass->convertFahrenheitToCelsius($myClass->getFahrenheit())
, "
<sup>o</sup> Celsius.";

}
else
{
echo "Please enter a valid temperature and scale";
}

}
else
{
?>

<form action="<? echo $PHP_SELF; ?>" method="post">
<input type ="text" name="temp" size="4" maxlength="4">
<select name="units">
<option value="celsius">Celsius</option>
<option value="fahrenheit">Fahrenheit</option>
</select>
<input type ="submit" name="submit" value="Convert">
</form>

<?
}
?>

</body>
</html>
```

This script consists of two parts: the form which allows the user to select a temperature scale and enter a tempterature value, and the form processor which actually uses the Bean to perform the conversion and display the result.

Here's what the form looks like:

Once this form has been submitted, an object is instantiated from the Temperature class, and the information provided by the user is used to perform temperature conversion using the Bean methods described above. The result is then displayed to the user.

Here's what the result looks like:

Note that it's necessary to convert the type of the form variable $temp from string to integer in order to make it compatible with the arguments expected by the Bean − this data type conversion is one of the important issues you will face when accessing Java classes through PHP.

And that's about it for the moment. In case you'd like to learn more, take a look at the following links:

The PHP manual's Java pages, at http://www.php.net/manual/en/ref.java.php

O'Reilly's PHP and Java tutorial on O'Reilly ONLamp.com, at
http://www.onlamp.com/pub/a/php/2001/06/14/php_jav.html

PHPBuilder's PHP and Java tutorial, at http://www.phpbuilder.com/columns/marknold20001221.php3

See you soon! Note: All examples in this article have been tested on Linux/i586 with JDK 1.3.0, Apache 1.3.20 and PHP 4.1.1. Examples are illustrative only, and are not meant for a production environment. Melonfire provides no warranties or support for the source code described in this article. YMMV!