PHP / Java Integration

Introduction

There are two possible ways to bridge PHP and Java: you can either integrate PHP into a Java Servlet environment, which is the more stable and efficient solution, or integrate Java support into PHP. The former is provided by a SAPI module that interfaces with the Servlet server, the latter by this Java extension.

The Java extension provides a simple and effective means for creating and invoking methods on Java objects from PHP. The JVM is created using JNI, and everything runs in-process.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension?including the names of its functions and any other documentation surrounding this extension?may change without notice in a future release of PHP. This extension should be used at your own risk.

Installing/Configuring

Requirements

You need a Java VM installed on your machine to use this extension.

Installation

This » PECL extension is not bundled with PHP.

In PHP 4 this PECL extensions source can be found in the *ext/* directory within the PHP source or at the PECL link above. In order to use these functions you must compile PHP with Java support by using the *--with-java[=DIR]* where DIR points to the base install directory of your JDK. This extension can only be built as a shared extension. Additional build instructions can be found in *php-src/ext/java/README*.

Windows users will enable *php_java.dll* inside of *php.ini* in order to use these functions. In PHP 4 this DLL resides in the *extensions/* directory within the PHP Windows binaries download. The DLL for this PECL extension may be downloaded from either the » PHP Downloads page or from » http://pecl4win.php.net/

Note
In order to enable this module on a Windows environment with PHP <= 4.0.6, you must make *jvm.dll* available to your systems PATH. No additional DLL is needed for PHP versions > 4.0.6.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

Java configuration options

Name	Default	Changeable	Changelog
java.class.path	NULL	PHP_INI_ALL	
java.home	NULL	PHP_INI_ALL	
java.library.path	NULL	PHP_INI_ALL	
java.library	JAVALIB	PHP_INI_ALL	

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Java Servlet SAPI

The Java Servlet SAPI builds upon the mechanism defined by the Java extension to enable the entire PHP processor to be run as a servlet. The primary advantage of this from a PHP perspective is that web servers which support servlets typically take great care in pooling and reusing JVMs. Build instructions for the Servlet SAPI module can be found in *php4/sapi/README*. Notes:

- While this code is intended to be able to run on any servlet engine, it has only been tested on Apache's Jakarta/tomcat to date. Bug reports, success stories and/or patches required to get this code to run on other engines would be appreciated.
- PHP has a habit of changing the working directory. sapi/servlet will eventually change it back, but while PHP is running the servlet engine may not be able to load any classes from the CLASSPATH which are specified using a relative directory syntax, or find the work directory used for administration and JSP compilation tasks.

Examples

Example #1 - Java Example

```
<?php
// get instance of Java class java.lang.System in PHP
$system = new Java('java.lang.System');

// demonstrate property access
echo 'Java version=' . $system->getProperty('java.version') . '<br />';
echo 'Java vendor=' . $system->getProperty('java.vendor') . '<br />';
echo 'OS=' . $system->getProperty('os.name') . ' ' .
            $system->getProperty('os.version') . ' on ' .
            $system->getProperty('os.arch') . ' <br />';

// java.util.Date example
$formatter = new Java('java.text.SimpleDateFormat',
                      "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

echo $formatter->format(new Java('java.util.Date'));
?>
```

Example #2 - AWT Example

```
<?php
// This example is only intended to be run using the CLI.

$frame  = new Java('java.awt.Frame', 'PHP');
$button = new Java('java.awt.Button', 'Hello Java World!');

$frame->add('North', $button);
$frame->validate();
$frame->pack();
$frame->visible = True;

$thread = new Java('java.lang.Thread');
$thread->sleep(10000);

$frame->dispose();
?>
```

Notes:

- *new Java()* will create an instance of a class if a suitable constructor is available. If no parameters are passed and the default constructor is useful as it provides access to classes like *java.lang.System* which expose most of their functionallity through static methods.
- Accessing a member of an instance will first look for bean properties then public fields. In other words, *print $date.time* will first attempt to be resolved as *$date.getTime()*,

then as *$date.time*.

- Both static and instance members can be accessed on an object with the same syntax. Furthermore, if the java object is of type *java.lang.Class*, then static members of the class (fields and methods) can be accessed.
- Exceptions raised result in PHP warnings, and **NULL** results. The warnings may be eliminated by prefixing the method call with an "@" sign. The following APIs may be used to retrieve and reset the last error:
  - java_last_exception_get()
  - java_last_exception_clear()
- Overload resolution is in general a hard problem given the differences in types between the two languages. The PHP Java extension employs a simple, but fairly effective, metric for determining which overload is the best match. Additionally, method names in PHP are not case sensitive, potentially increasing the number of overloads to select from. Once a method is selected, the parameters are coerced if necessary, possibly with a loss of data (example: double precision floating point numbers will be converted to boolean).
- In the tradition of PHP, arrays and hashtables may pretty much be used interchangably. Note that hashtables in PHP may only be indexed by integers or strings; and that arrays of primitive types in Java can not be sparse. Also note that these constructs are passed by value, so may be expensive in terms of memory and time.

Java Functions

java_last_exception_clear

java_last_exception_clear -- Clear last Java exception

Description

void **java_last_exception_clear** (void)

Clears last Java exception.

Return Values

No value is returned.

Examples

See java_last_exception_get() for an example.

Notes

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

java_last_exception_get

java_last_exception_get -- Get last Java exception

Description

object **java_last_exception_get** (void)

Gets last Java exception.

Return Values

Returns an exception object.

Examples

The following example demonstrates the usage of Java's exception handler from within PHP:

Example #3 - Java exception handler

```
<?php
$stack = new Java('java.util.Stack');
$stack->push(1);

// This should succeed
$result = $stack->pop();
$ex = java_last_exception_get();
if (!$ex) {
 echo "$result\n";
}

// This should fail (error suppressed by @)
$result = @$stack->pop();
$ex = java_last_exception_get();
if ($ex) {
 echo $ex->toString();
}

// Clear last exception
java_last_exception_clear();
?>
```

Notes

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This

function should be used at your own risk.