

<?php

PHP 5

The Perfect Enterprise Integration Platform

Zeev Suraski
Paris, November 2004

Zend The *php* company

Overview

- **What is an Integration Platform?**
- **XML**
- **Web Services**
- **Java Integration**
- **Summary**

<?php

Integration Platform

Zend The *php* company

PHP as an Integration Platform

- An Integration Platform is a system that can communicate with many different, disparate components, allowing the creation of new applications
- PHP has grown to be a powerful integration platform for Web applications
 - It is still growing!
- The integration capabilities make PHP more attractive even for non-Web tasks

PHP as an Integration Platform

Typical use cases for integration:

- **Sharing data between applications**
 - XML as a file format
 - XML as a communications protocol
- **Creating hybrid applications**
 - Java backend components, PHP frontend and Web logic
 - Same idea with .NET
- **Connecting to remote services**
 - Web Services (or XML-RPC), e.g. Google, E-bay, etc.

<?php

XML

Zend

The *php* company

When should I use XML?

- Short answer:

Always.

- Longer answer:

Always, unless it really, *really* makes no sense.

- Use XML to replace proprietary (or ad-hoc) formats
- Use XML as a communications protocol (e.g. RSS)

XML in PHP 4

- **Two main options:**
 - Using SAX (**S**imple **A**PI for **X**ML)
 - Using DOM (**D**ocument **O**bject **M**odel)
- **Neither was (is) very good in terms of ease of use.**
- **DOM support is also not that stable in PHP 4**

XML Made Simple - SimpleXML

- **Very simple motto:**

Make XML what it should be.

Simple.

- Takes advantage of the Zend Engine II Overloading API, allowing PHP to treat XML data as if it was a native type
- Built on the powerful and high-performance libxml2 library
- Supports advanced features such as Xpath and Schema validation

Credits:

Daniel Veillard (libxml2)

Sterling Hughes (SimpleXML)

Parsing XML

```
- <veggies foo="bar">
- <veggie>
  <name>Carrot</name>
  <desc>an orangey root</desc>
</veggie>
- <veggie>
  <name>Tomato</name>
  <desc>a red fruit</desc>
</veggie>
</veggies>
```

```
1 <?php
2 $dom = domxml_open_file("./veggies.xml") or die("failed!");
3 $veggies = $dom->get_elements_by_tagname("veggie");
4 foreach($veggies as $veggie) {
5     $name_node = $veggie->get_elements_by_tagname("name");
6     $name = $name_node[0]->get_content();
7     $desc_node = $veggie->get_elements_by_tagname("desc");
8     $desc = $desc_node[0]->get_content();
9     print "$name is $desc\n";
10 }
11?>
```

```
1 <?php
2 $veggies = simplexml_load_file('veggies.xml');
3 foreach ($veggies->veggie as $veggie) {
4     print "$veggie->name is $veggie->desc\n";
5 }
6?>
```

Manipulating XML

```
1 <?php
2 $veggies = simplexml_load_file("veggies.xml");
3 print_r($veggies);
4 $veggies->veggie[1]->desc = "I love it!";
5 print_r($veggies);
6
7 // Save the updated XML file
8 $newXML = fopen("/tmp/veggies-new.xml", "w")
9     or die("Unable to open file");
10 fwrite($newXML, $veggies->asXML());
11 fclose($newXML);
12 ?>
```

```
- <veggies foo="bar">
- <veggie>
  <name>Carrot</name>
  <desc>an orange root</desc>
</veggie>
- <veggie>
  <name>Tomato</name>
  <desc>a red fruit</desc>
</veggie>
</veggies>
```



```
<?xml version="1.0" ?>
- <veggies foo="bar">
- <veggie>
  <name>Carrot</name>
  <desc>an orange root</desc>
</veggie>
- <veggie>
  <name>Tomato</name>
  <desc>I love it!</desc>
</veggie>
</veggies>
```

<?php

Web Services

Zend The *php* company

What are Web Services?

- Means for integrating Web-based applications using XML, SOAP, WSDL and UDDI open standards.
- Designed to replace other integration infrastructures, such as CORBA and DCOM
- SOA - Service Oriented Architecture
 - Loose coupling
- Cross platform
- Cross language

Web Services in PHP 5

- In PHP 4, there was no standard SOAP implementation.
- There were several non-standard implementations.
 - NuSOAP
 - PEAR::SOAP
 - pecl/soap
- No single implementation which is complete, high-performance and reliable
 - At least not all-in-one
- In PHP 5 - bundled, official SOAP implementation.
 - Use `--enable-soap` to activate

Credits:
Dmitry Stogov <dmitry@zend.com>

Using Web Services: Google Speller

```
1 <?php
2
3 if (isset($_GET["spellString"])) {
4     $spellString = $_GET["spellString"];
5     $google = new SoapClient("googleapi/GoogleSearch.wsdl");
6
7     $result = $google->doSpellingSuggestion("ectg6UlQFHJruG1h5gXnE9LRx0PPGblv",
8                                             $spellString);
9     if ($result) {
10         print "Google's spelling suggestion for <b>$spellString</b> is <b>$result</b>.";
11     } else {
12         print "No spelling suggestion for $spellString.";
13     }
14 }
15 ?>
16
```

Exposing Web Services: A Fortune Server

```
1 <?php
2
3 class Fortune {
4     function getFortune()
5     {
6         return `usr/games/fortune`;
7     }
8
9     function getExtendedFortune()
10    {
11        return array(    "timestamp" => time(),
12                        "fortuneText" => `usr/games/fortune`);
13    }
14 };
15
16 $SOAP = new SoapServer("Fortune.wsdl");
17 $SOAP->setClass("Fortune");
18 $SOAP->handle();
19
20 ?>
```

The Fortune Client

```
1 <pre>
2 <?php
3
4 $fortuneClient = new SoapClient("Fortune.wsdl");
5
6 if (isset($_GET["extended"])) {
7     print_r($fortuneClient->getExtendedFortune());
8 } else {
9     print $fortuneClient->getFortune();
10 }
11 ?>
12 </pre>
```

SOAP in PHP 4?

- Yes, compatible with PHP 4
- Works in very much the same way
 - Except there are no exceptions
- How to obtain?

```
cvcs -d :pserver:cvcsread@cvcs.php.net:/repository login
```

```
Password: phpfi
```

```
cvcs -d :pserver:cvcsread@cvcs.php.net:/repository co pecl/soap
```

<?php

Java Integration

Zend The *php* company

Java and PHP

- **The idea of using Java components from PHP is relatively old**
- **Rationales:**
 - A lot of companies, especially medium/large ones, have corporate logic implemented as Java components, but would like to use PHP to write the Web front end instead of JSP
 - J2EE features which are not available in PHP prevent certain companies today from using it
- **An initial Java extension, based on ZE I's (PHP 4) extensibility API was developed by Sam Ruby**
 - Not practical for real world, production Web server usage due to performance / memory considerations
 - Limited functionality, due to limited mapping of PHP syntax to Java syntax

Java and PHP, cont.

- PHP 5, through ZE II, provides much-improved extensibility APIs that allow for a much better Java extension
- JSR (Java Specification Request) 223, a process started by Sun Microsystems and Zend Technologies, and joined by a dozen other companies, is aimed to define a formal spec for integrating Java with other languages in general, and PHP in particular
 - Endorsement from Sun for using hybrid PHP-Java applications

Java and PHP, cont.

- Like any other JSRs, the spec will become open, making it possible for anybody to implement.
- Zend intends to provide a commercially-backed implementation of the spec in 2005.

Example - Using Java components from PHP

```
1 <?php
2
3 $host = 'localhost';
4 $db = 'test';
5 $user = 'test';
6 $pws = '';
7
8 Java("com.mysql.jdbc.Driver"); // init
9
10 $conn = Java("java.sql.DriverManager")
11         ->getConnection("jdbc:mysql://$host/$db", $user, $pwd);
12 $stmt = $conn->createStatement();
13
14 $rs = $stmt->executeQuery("SELECT * FROM news");
15
16 while($rs->next()) {
17     printf("On %s article %s\n", $rs->getDate("when"),
18         $rs->getString("title"));
19 }
20
21 ?>
```

<?php

Summary

Zend

The *php* company

Summary

- PHP 5 is the ideal platform for creating Web applications that integrate information from multiple, disparate sources.
- PHP can communicate easily and efficiently with any XML-based system, with any Web Service, and in the near future - any Java based system.
- Additional integration options:

- C / C++ extensions

- Even Perl code!

```
1 <?php
2
3 print "Hello from PHP!\n";
4 $perl = new Perl();
5 $perl->eval('print "Hello from Perl!\n"');
6 print "Bye!\n";
7
8 ?>
```

<?php

Thank You!

Zend The *php* company

More Information

Web Services

- <http://www.oreillynet.com/lpt/a/webservices/2002/02/12/webservicefaqs.html>
 - <http://tinyurl.com/6ue8w>
- <http://www.w3.org/TR/soap/>

Web Services in PHP 5

- <http://www.zend.com/php5/articles/php5-SOAP.php>

Future Directions

- SOAP attachments
- Integration with tools

Example #2 - A Web Service with Persistence

```
1 <?php
2
3 class Counter {
4     private $counter=0;
5
6     function getCount()
7     {
8         return ++$this->counter;
9     }
10 };
11
12 $SOAP = new SoapServer("Counter.wsdl");
13 $SOAP->setClass("Counter");
14 $SOAP->setPersistence(SOAP_PERSISTENCE_SESSION);
15 $SOAP->handle();
16
17 ?>
```

What is WSDL?

- **Stands for Web Service Definition Language**
- **Describes the expected input and output of one or more Web Services**
- **Used by both the client and server**
- **Not strictly necessary in PHP**
 - Web Services can be programmatically described, in both the client and server

What is SOAP?

- **Used to stand for ‘Simple Object Access Protocol’**
 - Now stands for just ‘Soap’
- **XML-based communications protocol, for exchanging data**
- **Stateless**
- **Typically used in a request/response paradigm, over HTTP**
 - a.k.a. Web Services

Troubleshooting

- **Control both ends**
 - That's not always possible, unfortunately
- **Throw SoapFault's on every server error!**
- **Disable WSDL caching (soap.wsdl_cache_enabled=off)**
- **Disable 'Friendly HTTP errors'**
- **Use SoapClient::__getLastRequest() and SoapClient::__getLastResponse()**
- **Use try..catch and look into SoapFault**
- **Use a debugger!**

Using Web Services w/o WSDL: Google Speller in manual mode

```
1 <?php
2
3 if (isset($_GET["spellString"])) {
4     $spellString = $_GET["spellString"];
5     $google = new SoapClient(null, array(
6         "location"=>"http://api.google.com/search/beta2",
7         "uri"=>"urn:GoogleSearch"));
8
9     $result = $google->doSpellingSuggestion(
10         new SoapParam("ectg6UlQFHJruG1h5gXnE9LRx0PFGB1v", "key"),
11         new SoapParam($spellString, "phrase"));
12
13     if ($result) {
14         print "Google's spelling suggestion for <b>$spellString</b> is <b>$result</b>.";
15     } else {
16         print "No spelling suggestion for $spellString.";
17     }
18 }
19 ?>
```