PHP 101
An Introduction to PHP 5

Rob Richards

http://xri.net/=rob.richards

www.cdatazone.org

What Is PHP?

- Loosely typed
  - Variables can be of any type
  - Variables set by context
  - Functions can return different types
- Dynamic Language
  - Dynamic typing
  - Run time code modification
  - Interpretation vs compilation

C# .NET Example

```
XmlDocument doc = new XmlDocument();

XmlElement root = doc.CreateElement("root");
doc.AppendChild(root);

XmlElement child = doc.CreateElement("child");
root.AppendChild(child);

XmlText textnode = doc.CreateTextNode("childcontent");
child.AppendChild(textnode);

Console.Write(doc.OuterXml);
```

PHP Example

```
$doc = new DOMDocument();

$root = $doc->CreateElement("root");
$doc->AppendChild($root);

$child = $doc->CreateElement("child");
$root->AppendChild($child);

$root = $doc->CreateTextNode("childcontent");
$child->AppendChild($root);

echo $doc->saveXML();
```

What Is PHP?

- Cross Platform
- Supports numerous types of Web servers
- Typically run on Web server
- Also used for command line scripting (CLI)
- Supports development of cross platform GUI applications (PHP-GTK)

History of PHP

- 1994
- Personal Home Page
- Created by Rasmus Lerdorf
- Track Online Resume

History of PHP

- 1995
- PHP/FI version 2
- Personal Home Page / Forms Interpreter
- Growing needs of web pages
- mSQL support
- Open Source

History of PHP

- 1997
- PHP 3
- PHP: Hypertext Preprocessor
- Parser re-written by Andi Gutmans and Zeev Suraski
- needed more for an e-commerce application
- language was extendable

History of PHP

1999/2000

- PHP 4
- Core re-write
- Powered by Zend Engine 1.0
- Handle more complex applications
- Improve modularity of PHP core
- Support for numerous databases and web servers

History of PHP

- 2004
- PHP 5
- Zend Engine 2
- New Object Model
- Better XML and Web service support
- Better database support
- Other miscellaneous changes

The Future of PHP

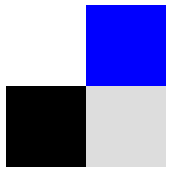
- PHP 5.3 (Early 2008)
  - Namespaces
    - namespace keyword
    - use keyword
  - Late Static Binding
- PHP 6.0 (???)
  - Unicode

PHP 4 EOL

08/08/08

PHP
Is
Stable

Who Uses PHP?



del.icio.us

flickr

digg

YAHOO!

facebook

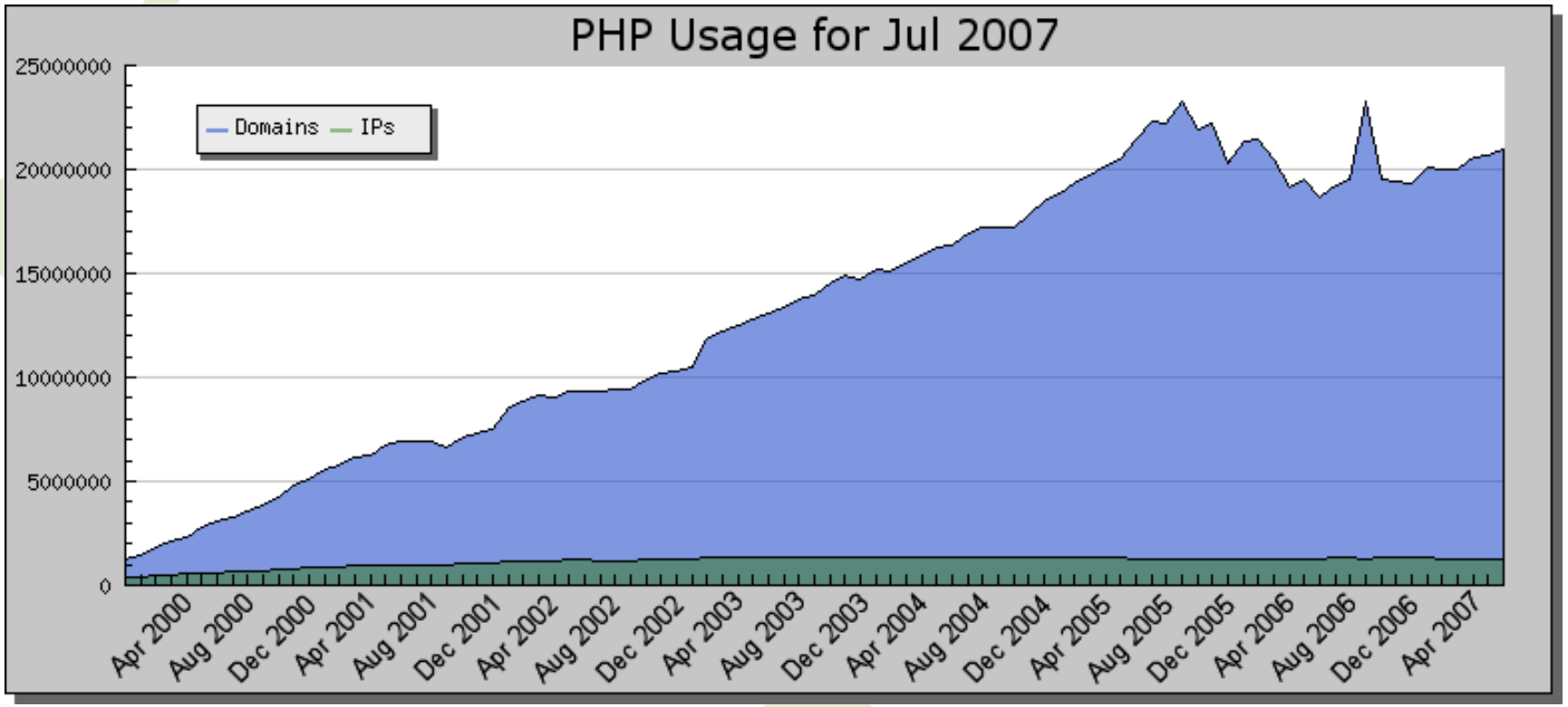
WIKIPEDIA

PHP Usage

Usage Stats for July 2007

Source: Netcraft

PHP: 20,917,850 domains, 1,224,183 IP addresses



Why Trust PHP?

- Open Source
  - Review the code
  - Make your own changes
- Active Community
  - ~500 Active committers
  - Documentation (http://doc.php.net)
  - mailing lists (http://news.php.net)
  - Extensions (http://pecl.php.net)
  - Libraries (http://pear.php.net)

Corporate Involvement

PHP and Web 2.0

What Is Web 2.0?

- Services, not packaged software, with cost-effective scalability
- Control over unique, hard-to-recreate data sources that get richer as more people use them
- Trusting users as co-developers
- Harnessing collective intelligence
- Leveraging the long tail through customer self-service
- Software above the level of a single device
- Lightweight user interfaces, development models, AND business models

5 second Atom parser

Flickr Atom Feed

5 second Atom parser

```
<?php
$url = 'http://www.flickr.com/services/feeds/photos_public.gne';

foreach(simplexml_load_file($url)->entry as $it)
    echo $it->content;
?>
```

Yahoo Maps
in
30 seconds

LOCAL THUNDER

Geocoding

Mapping & Geocoding Combined

LOCAL THUNDER

Maptiles

Mashups

- Combining data from multiple sources
- Consumer Mashups
- Data Mashups
- Enterprise Mashups

Dude, Where's My Used Car?

- http://www.dudewheresmyusedcar.com/
- Written by Adam Trachtenberg
- Combination of Ebay Motors and Google Maps
- Written using PHP 5
- Uses ext/soap and other new features

Yahoo Pipes

- http://pipes.yahoo.com/pipes/
- A powerful composition tool to aggregate, manipulate, and mashup content from around the web
  - ✧ combine many feeds into one, then sort, filter and translate it.
  - ✧ geocode your favorite feeds and browse the items on an interactive map.
  - ✧ power widgets/badges on your web site.
  - ✧ grab the output of any Pipes as RSS, JSON, KML, and other formats

QEDWiki

- http://services.alphaworks.ibm.com/qedwiki/
- A browser-based assembly canvas used to create simple mashups
- environment in which the creator of a mashup uses software components (or services) made available by content providers
- Business users can quickly prototype and build ad hoc applications without depending on software engineers

OpenSocial

http://code.google.com/apis/opensocial/

XML and Web services

XML

- Describes data in a structured format
- Platform independent
- Allows creation of markup languages
- Understandable to humans and computers

```
<person>
    <identifier>1001</identifier>
    <first_name>Rob</first_name>
    <last_name>Richards</last_name>
</person>
```

XML Document Snippet

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://purl.org/rss/1.0/" xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
         xmlns:admin="http://webns.net/mvcb/" . . .>
    <channel rdf:about="http://planet-php.net">
        <title>Planet PHP</title>
        <link>http://planet-php.net</link>
        <description>People blogging about PHP</description>
        <dc:language>en</dc:language>
        <dc:date>2007-11-03T14:50:00Z</dc:date>
        <admin:generatorAgent rdf:resource="http://planet-php.net"/>
        <admin:errorReportsTo rdf:resource="mailto:chregu@bitflux.ch"/>
        <sy:updateBase>2000-01-01T12:00+00:00</sy:updateBase>
        <items>
            <rdf:Seq>
                <rdf:li rdf:resource="http://www.phpguru.org/#154"/>
            </rdf:Seq>
        </items>
    </channel>
```

XML in PHP 4

- Limited functionality
- Multiple xml and xslt libraries
  - expat
  - libxml2 and libxslt
  - sablotron
- Poor performance
- Maintenance Issues
- Little to no interoperability

XML in PHP 5

- Supports numerous XML standards and technologies
- Extensions standardized on libxml2 and libxslt
- Leverage PHP Streams
- XML based extensions support interoperability with little to no performance penalties

XML in PHP 5

- XML
- DOM
- SimpleXML
- XSL
- XMLReader
- XMLWriter

XML Extension

- Simple API for XML (SAX)
- Push/Event based parser
- libxml2 now used for underlying library
- Compatible with code from PHP 4
- Ability to use expat if necessary

XML Example

```
<?php
$url = "http://pipes.yahoo.com/pipes/pipe.run?
_id=ogx67iCK3BG3xh9lnkartA&_render=rss";

$inItem = False;
$dumpContent = False;

function char_handler($xml, $data) {
   if ($GLOBALS['dumpContent']) {
      echo $data;
   }
}
```

XML Example

```
function elementHandler($xml, $tag, $attributes) {
   if ($GLOBALS['inItem']) {
     switch ($tag) {
        case 'TITLE':
        case 'LINK':
        case 'DESCRIPTION':
           print $tag.": ";
           $GLOBALS['dumpContent'] = True;
           break;
        default:
           $GLOBALS['dumpContent'] = False;
     }
   } elseif ($tag == 'ITEM') {
      $GLOBALS['inItem'] = True;
   }
}
```

XML Example

```
function endHandler($xml, $tag) {
   if ($GLOBALS['dumpContent']) {
      print "<br />";
      $GLOBALS['dumpContent'] = False;
   }
   if ($tag == 'ITEM' && $GLOBALS['inItem']) {
      $GLOBALS['inItem'] = False;
      print "<br />";
   }
}

$xml = xml_parser_create();

xml_set_element_handler($xml, 'elementHandler', 'endHandler');
xml_set_character_data_handler($xml, 'char_handler');

xml_parse($xml, file_get_contents($url));
```

DOM

- Tree based parser
- Allows for creation and editing of XML documents
- W3C Specification with DOM Level 2/3 compliancy
- Provides XPath support
- Provides XInclude Support
- Ability to work with HTML documents
- Zero copy interoperability with SimpleXML
- Replacement for ext/domxml from PHP 4

DOM Example

```
$url = "http://pipes.yahoo.com/pipes/pipe.run?_id=ogx67iCK3BG3xh9lnkartA&_render=rss";
$dom = new DOMDocument();
$dom->load($url);

foreach ($dom->getElementsByTagName("item") AS $itemNode)
{
    foreach ($itemNode->childNodes AS $child)
    {
        if ($child->nodeName == 'title' || $child->nodeName == 'link'
            || $child->nodeName == 'description') {
                print $child->nodeName.": ";

                foreach ($child->childNodes AS $textNode) {
                    print $textNode->nodeValue;
                }
                print "<br />";
        }
    }
    print "<br />";
}
```

.NET C# Example

```
string url = "http://pipes.yahoo.com/pipes/pipe.run_id=ogx67iCK3BG3xh9lnkartA&_render=rss";
XmlDocument doc = new XmlDocument();
doc.Load(url);

foreach (XmlElement itemNode in doc.GetElementsByTagName("item"))
{
    foreach (XmlNode child in itemNode.ChildNodes)
    {
        if (child.Name == "title" || child.Name == "link" || child.Name == "description") {
                Console.Write(child.Name + ": ");

                foreach (XmlNode textNode in child.ChildNodes) {
                    Console.Write(textNode.Value);
                }
                Console.WriteLine("<br />");
        }
    }
    Console.WriteLine("<br />");
}
```

SimpleXML

Provides simple access to XML documents

Operates only on elements and attributes

Contains XPath support

Allows for modifications to the XML

Zero copy interoperability with DOM

SimpleXML Example

```
$url = "http://pipes.yahoo.com/pipes/pipe.run?
_id=ogx67iCK3BG3xh9lnkartA&_render=rss";

$rss = simplexml_load_file($url);

foreach ( $rss->channel->item AS $item ) {
    echo "Title: " . $item->title . "<br />";
    echo "Link: " . $item->link . "<br />";
    if (isset($item->description)) {
        echo "Description: " . $item->description . "<br />";
    }
    echo "<br />";
}
```

XSL

- Used to transform XML data
- XSLT based on XPath
- Works with DOM and SimpleXML, although the DOM extension is required.
- Provides the capability of calling PHP functions during a transformation
- DOM nodes may be returned from PHP functions
- The LIBXML_NOCDATA and LIBXML_NOENT constants are your friends.

XSL Example

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes" />

  <xsl:template match="/">
    <xsl:apply-templates select="rss/channel/item" />
  </xsl:template>

  <xsl:template match="/rss/channel/item">
    Title: <xsl:value-of select="./title" /><br />
    Link: <xsl:value-of select="./link" /><br />
    <xsl:if test="description">
      Description: <xsl:value-of select="./description" /><br />
    </xsl:if>
    <br />
  </xsl:template>

</xsl:stylesheet>
```

XSL Example

```
$url = "http://pipes.yahoo.com/pipes/pipe.run?
_id=ogx67iCK3BG3xh9lnkartA&_render=rss";

$stylesheet = new DOMDocument();
$stylesheet->load("xsl.xsl");

$xslProc = new xsltprocessor();
$xslProc->importStylesheet($stylesheet);

$dom = new DOMDocument();
$dom->load($url);

print $xslProc->transformToXML($dom);
```

XMLReader

- Forward moving stream based parser
- It is a Pull parser
- Based on the C# XmlTextReader API
- Advantages:
  - ✧ Low memory footprint
  - ✧ Namespace support
  - ✧ Simple API
  - ✧ Validation support
  - ✧ Advanced Feature Set
  - ✧ Faster Processing

XMLReader Simple Example

```
$url = "http://pipes.yahoo.com/pipes/pipe.run?
_id=ogx67iCK3BG3xh9lnkartA&_render=rss";

$reader = new XMLReader();
$reader->open($url);

while ($reader->read()) {
    print $reader->name . "<br />";
}
```

XMLReader Example

```
$url = "http://pipes.yahoo.com/pipes/pipe.run?_id=ogx67iCK3BG3xh9lnkartA&_render=rss";

        $reader = new XMLReader();
        $reader->open($url);

        while ($reader->read()) {
            if ($reader->name == 'item')  {    break;    }
        }

        $level = $reader->depth;
        do {
            while ($reader->read() && ($reader->depth > $level)) {
                if ($reader->name == 'title' || $reader->name == 'link' ||
                  $reader->name == 'description') {
                    print $reader->name.": ";
                    $elemLevel = $reader->depth;
                    while ($reader->read() && ($reader->depth > $elemLevel)) {
                        print $reader->value;
                    }
                    print "<br />";
                }
            }
            print "<br />";
        } while ($reader->next("item"));
```

LOCAL THUNDER

XMLWriter

- Lightweight and forward-only API for generating well formed XML
- Automatically escapes data
- Works with PHP 4.3+ available at http://pecl.php.net/package/xmlwriter
- Object Oriented API available for PHP 5+
- Part of core PHP distribution since PHP 5.1.2

XMLWriter Example

```
<Stock Symbol="MSFT">
 <Price>74.125</Price>
 <Change>5.89</Change>
 <Volume>69020000</Volume>
</Stock>
```

XMLWriter Example

```
$writer = new XMLWriter();
$writer->openURI("php://output");
$writer->setIndent(True);
$writer->startElement("Stock");
$writer->writeAttribute("Symbol", "MSFT");
$writer->writeElement("Price", 74.125);
$writer->writeElement("Change", 5.89);
$writer->writeElement("Volume", 69020000);
$writer->endElement();
$writer = NULL;
```

Visual Basic Example

```
Dim writer As New XmlTextWriter(Console.Out)
writer.Formatting = Formatting.Indented
writer.WriteStartElement("Stock")
writer.WriteAttributeString("Symbol", "MSFT")
writer.WriteElementString("Price", "74.125")
writer.WriteElementString("Change", "5.89")
writer.WriteElementString("Volume", "69020000")
writer.WriteEndElement()
writer.Close()
```

JSON

- Javascript Object Notation
- Data interchange format
- Simplifies client side development
- Compiled into PHP by default as of 5.2
- Encode/Decode functionality

JSON: XML Data

```
<?xml version="1.0" encoding="UTF-8"?>
<contacts>
    <contact id="1">
        <name>John Doe</name>
        <phone>123-456-7890</phone>
        <address>
            <street>123 JFKStreet</street>
            <city>Any Town</city>
            <state>Any State</state>
            <zipCode>12345</zipCode>
        </address>
    </contact>
</contacts>
```

JSON: JSON Data

```
{ "contacts" : {
      "contact" : {
        "@attributes" : {
          "id" : "1"
        },
        "name" : "John Doe",
        "phone" : "123-456-7890",
        "address" : {
          "street" : "123 JFK Street",
          "city" : "Any Town",
          "state" : "Any State",
          "zipCode" : "12345"
        }
      }
   }
}
```

JSON: PHP Example

```
class Person {
   public $first = 'Rob';
   public $last = 'Richards';
   public $address = NULL;
}

class Address {
    public $address1 = 'PO Box 123';
    public $city = 'Portland';
    public $state = 'Maine';
}

$objPerson = new Person();
$objPerson->address = new Address();

$data = json_encode($objPerson);
echo $data;
```

Output:

```
{
 "first":"Rob",
 "last":"Richards",
 "address":{
    "address1":"PO Box 123",
    "city":"Portland",
    "state":"Maine"
 }
}
```

Web services and PHP 4

- Extensions
  - XML
  - DOM
  - XML-RPC
- PEAR Packages
- NuSOAP
- String Manipulation

Web services and PHP 5

Much Easier
More Choices
Faster

REST Based Services

- Representational State Transfer
- Uses common Internet technologies
- Resources Identified by URI
- Cacheable
- Stateless
- It's Simple

REST: Yahoo Shopping

```
$prodSearchUrl =
'http://api.shopping.yahoo.com/ShoppingService/V1/productSearch';
$prodSearchUrl .= '?query='.rawurlencode('microsoft vista');
$prodSearchUrl .= "&appid=zzz&results=2&start=1";

$merchantUrl =
'http://api.shopping.yahoo.com/ShoppingService/V1/catalogListing';
$merchantUrl .= "?appid=zzz&results=2&start=1&catalogid=";

function mypad() {
    return str_repeat(" ", 5);
}
```

REST: Yahoo Shopping

```
$results = simplexml_load_file($prodSearchUrl);

foreach ($results->Result AS $result) {
    foreach ($result->Catalog AS $catalog) {
        echo "Name: $catalog->ProductName <br />";
        echo "URL: ".$catalog->Url."<br />";
        echo "Price: ".$catalog->PriceFrom." - ".$catalog->PriceTo."<br />";

        $items = simplexml_load_file($merchantUrl . $catalog["ID"]);
        foreach ($items->Offer AS $item) {
            print mypad()."Merchant: ".$item->Merchant->Name."<br />";
            print mypad()."Price: ".$item->TotalPrice."<br />";
            print mypad()."URL: ".$item->Url."<br />";
        }
    }
    echo "<br />";
}
```

SOAP Based Services

- Protocol for exchanging XML-based messages in a distributed environment
- HTTP/HTTPS common protocols for transport
- Promote interoperability
- Extendable

SOAP: Example

- Integration with ExactTarget
- Provide tools and services for email communications
- Focus on Email Marketing
- On-Demand Access
- Personal Interaction via Web-based GUI
- Integration via REST and/or SOAP
- SOA based SOAP interface

SOAP: Retrieve API information

```
$wsdl = 'https://webservice.exacttarget.com/etframework.wsdl';

$client = new SOAPClient($wsdl, array('trace'=>1));

$types = $client->__getTypes();
foreach ($types AS $type) {
    print $type."\n\n";
}

$functions = $client->__getFunctions();
foreach ($functions AS $function) {
    print $function."\n\n";
}
```

SOAP: Function List

CreateResponse Create(CreateRequest $parameters)

RetrieveResponseMsg Retrieve(RetrieveRequestMsg $parameters)

UpdateResponse Update(UpdateRequest $parameters)

DeleteResponse Delete(DeleteRequest $parameters)

ExecuteResponseMsg Execute(ExecuteRequestMsg $parameters)

SOAP: RetrieveRequestMsg

```
struct RetrieveRequestMsg {
    RetrieveRequest RetrieveRequest;
}

struct RetrieveRequest {
    ClientID ClientIDs;
    string ObjectType;
    string Properties;
    FilterPart Filter;
    AsyncResponse RespondTo;
    APIProperty PartnerProperties;
    string ContinueRequest;
    boolean QueryAllAccounts;
    boolean RetrieveAllSinceLastBatch;
}
```

SOAP: Basic List Retrieval Request

```
$client = new ExactTargetSoapClient($wsdl, $options);
/* WS-Security Username handling */

$request = new ExactTarget_RetrieveRequestMsg();

$rr = new ExactTarget_RetrieveRequest();

$rr->ObjectType = "List";
$rr->Properties = array("ListName");

$request->RetrieveRequest = $rr;
$response = $client->Retrieve($request);
```

SOAP: List Retrieval Response

```
object(stdClass)#6 (3) {
  ["OverallStatus"]=> string(2) "OK"
  ["RequestID"]=> string(36) "6bb27c71-16e6-4167-a57c-3..."
  ["Results"]=>
  array(2) {
    [0]=> object(stdClass)#12 (3) {
      ["PartnerKey"]=> NULL
      ["ObjectID"]=> NULL
      ["ListName"]=> string(8) "RobsList"
    }
    [1]=> object(stdClass)#8 (3) {
      ["PartnerKey"]=> NULL
      ["ObjectID"]=> NULL
      ["ListName"]=> string(12) "New Rob List"
```

PHP 5
and
The Enterprise

What Is Enterprise?

Any Organization?

Large Organizations?

Infrastructure Applications?

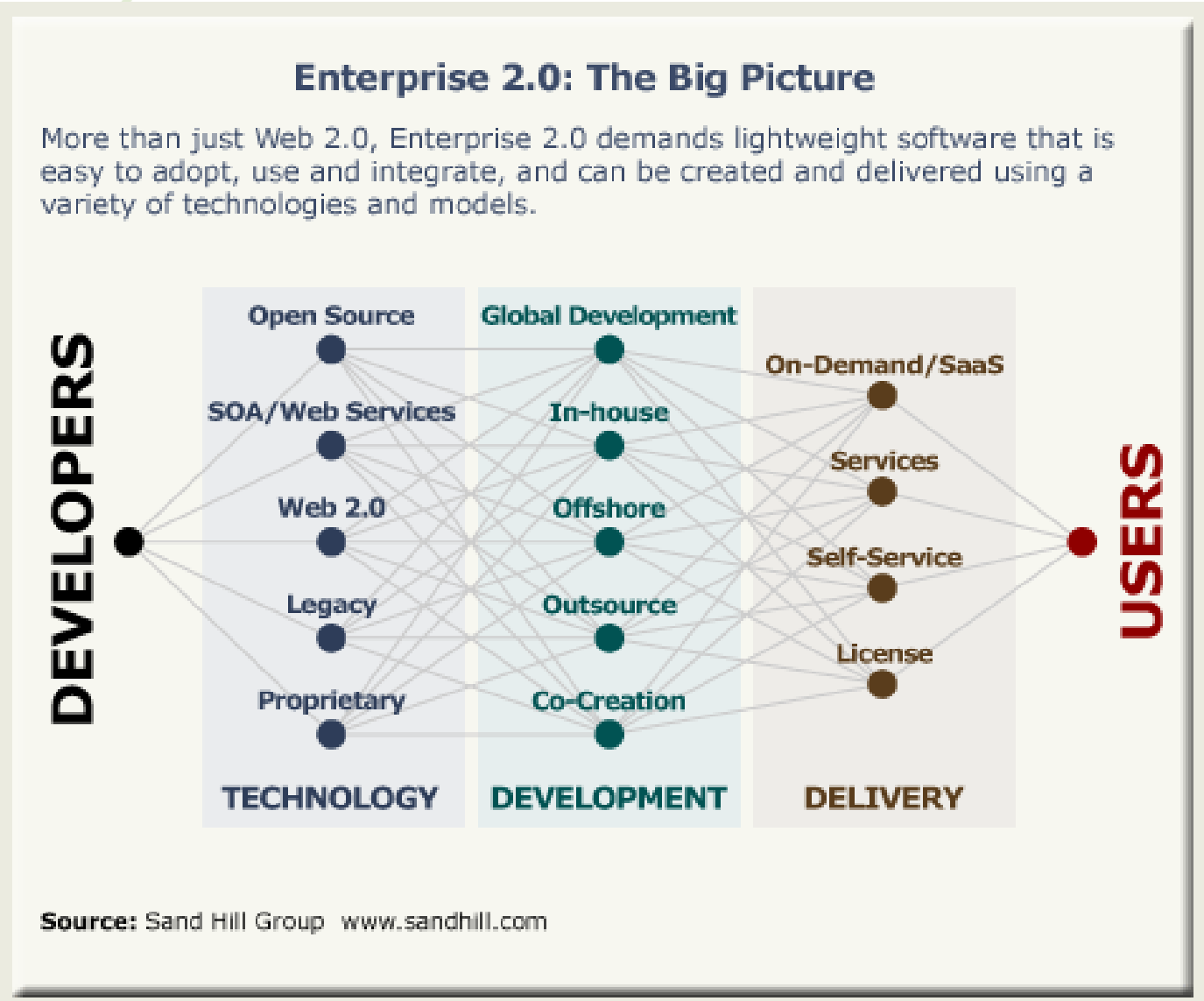
Internal Business Applications?

LOCAL THUNDER

PHP and Enterprise

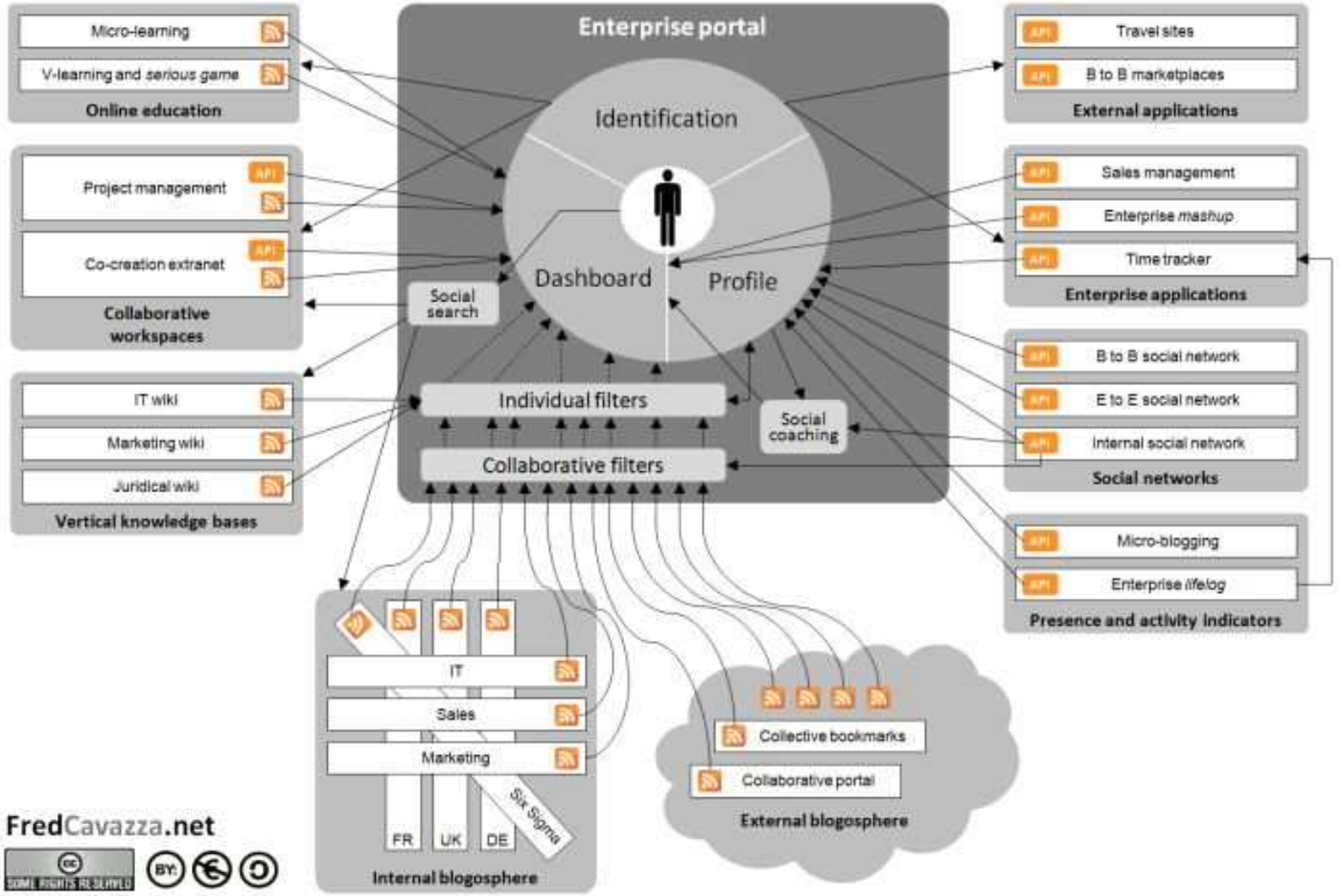
- PHP will not replace your infrastructure
- Rapid development
- Enhanced OO features
- Enterprise 2.0
- Enterprise Integration
- SOA integration

Enterprise 2.0



LOCAL THUNDER

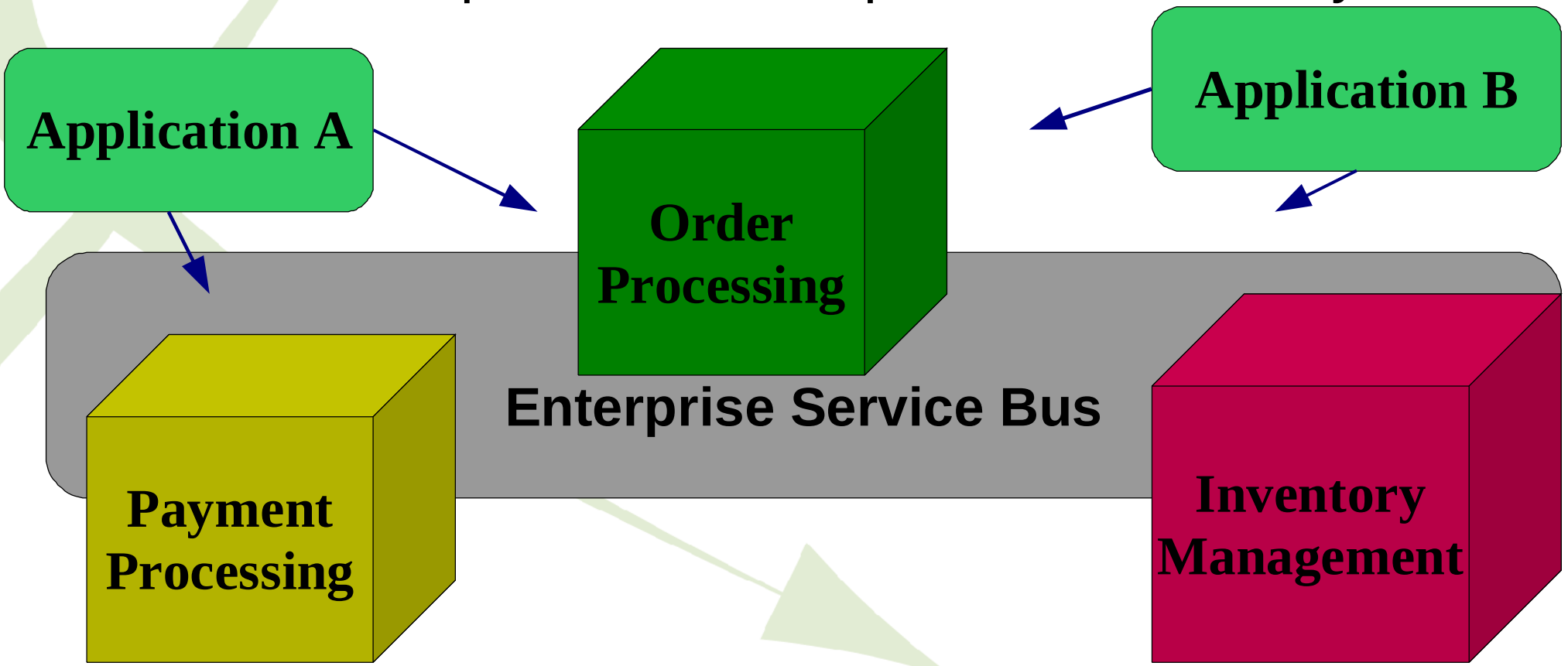
Enterprise 2.0



LOCAL THUNDER

Enterprise: SOA

Service-oriented architecture is a style of building applications based on independent and re-usable building blocks that provide some specific functionality.



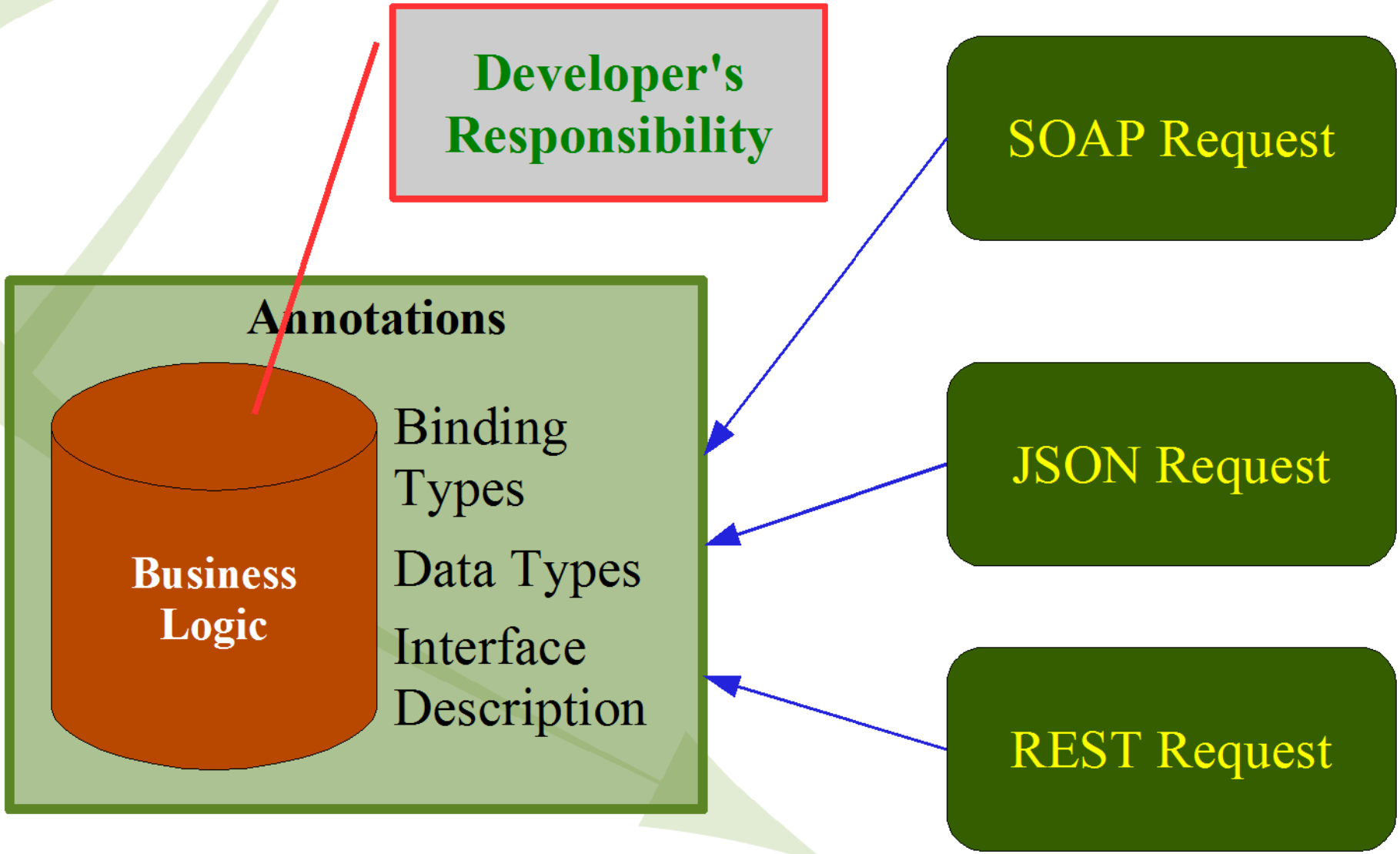
Enterprise PHP Integration

- Numerous Database Extensions
- REST Based
- SOAP Based
- Message Based
- JAVA (Java Bridge)
- COM and .NET

Enterprise Integration: SCA

- Project by the Open Service Oriented Architecture (OSOA) collaboration: http://www.osoa.org/display/Main/Home
- Allows developer to concentrate on the business logic rather than how it is all connected together
- http://www.osoa.org/display/PHP/SOA+PHP+Homepage
- Combined with the SDO extension
- Pecl repository: http://pecl.php.net/package/SCA_SDO
- Still in development

SCA

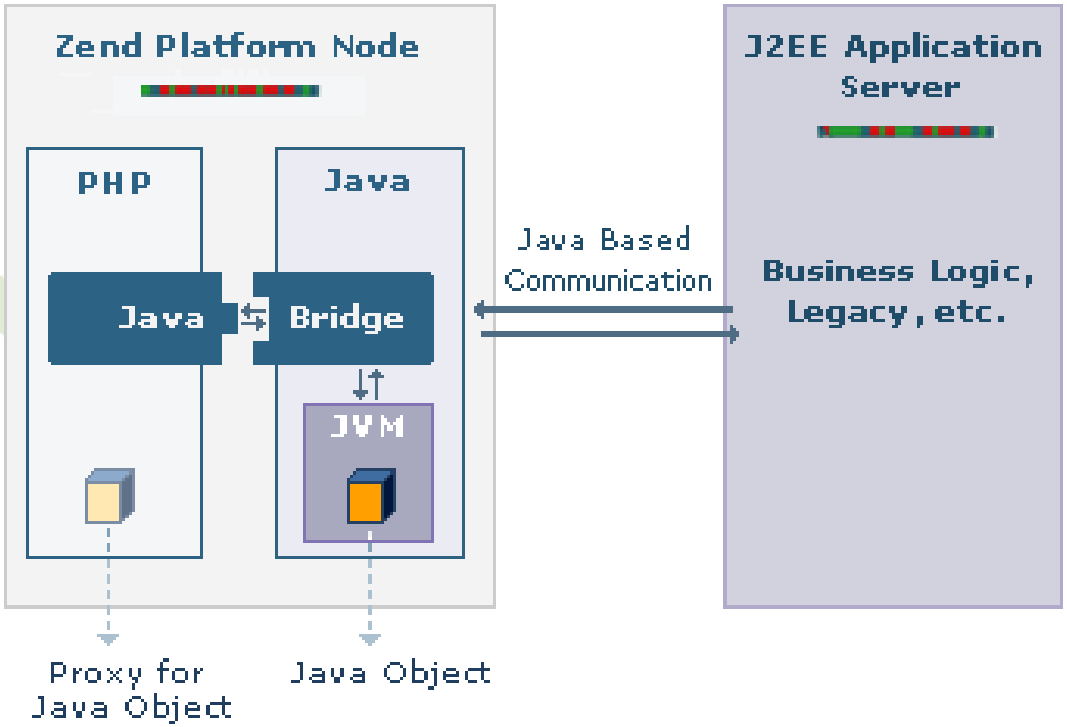

Developer's Responsibility
SOAP Request
Annotations
Business Logic
Binding Types
Data Types
Interface Description
JSON Request
REST Request

SAM: Simple Asynchronous Messaging

- http://project-sam.awardspace.com/index.htm
- PECL Repository – http://pecl.php.net/package/sam
- MQTT (MQ Telemetry Transport) Messaging Protocol
- Optionally interfaces with IBM Messaging and Queuing using XMS
  - ✧ WebSphere Application Server
  - ✧ WebSphere MQ
  - ✧ Other support IBM middleware products
- Pluggable architecture for additional messaging systems
- Add/Retrieve message capabilities
- Publish/Subscribe capabilities

Zend Platform Java Bridge

Access Java methods and properties from PHP



.NET Integration

```
<?php
$Engine = new COM("SAPI.SpVoice");

$Engine->Speak("Hello World!", 0);

unset($Engine);
?>
```

OOP in PHP 5

Object Oriented Programming

- Modularity
  - Independently operating components
- Inheritance
  - Deriving new classes (specializing classes) based on existing ones
- Polymorphism
  - allows you to treat derived class members just like their parent class's members
- Encapsulation
  - conceals the functional details of a class from objects that send messages to it

PHP and OOP

- PHP 3 introduced basic Object Oriented concepts
- PHP 4 did little to improve the object model
  - Assignment by copy
  - Objects passed by-value

Object References

```
<?php
  class OS {
      var $name;
      function OS($name) {
          $this->name = $name;
      }
  }

  function changeName(&$obj,  $name) {
      $obj->name = $name;
  }

  $linux = new OS('linux');
  $win = $linux;
  changeName($win, 'windows');
  echo $linux->name. "\n";
  echo $win->name;
?>
```

Output in PHP4:

linux

windows

Output in PHP 5:

windows

windows

LOCAL THUNDER

PHP 5 OO

- Constructors/Destructors
- Supports visibility (public, private, protected)
- Static Members
- Final Members
- Abstract Members
- Interfaces
- Magic Methods

PHP5 OO - *structors

```
class OS {
    function __construct($name) {
        echo "Constructor called<br />";
    }

    function __destruct() {
        echo "Destructor called<br />";
    }
}

$linux = new OS('linux');
echo "Step 1<br />";
$ref  = $linux;
echo "Step 2<br />";
$linux = NULL;
echo "Step 3<br />";
$ref = NULL;
echo "Step 4<br />";
```

PHP 4 OO

```
class OS {
    function privatemethod () {
        return "No public access";
    }
}

class newOS extends OS {
    function mymethod() {
        return $this->privatemethod();
    }
}

$myObj = new newOS();
echo $myObj->mymethod();
```

Output:

No public access

PHP5 OO - Private

```
class OS {
    private function privatemethod () {
        return "No public access";
    }
}

class newOS extends OS {
    function mymethod() {
        return $this->privatemethod();
    }
}

$myObj = new newOS();
echo $myObj->mymethod();
```

Output:

Fatal error: Call to private method OS::privatemethod() from context 'newOS' in foo.php on line 10

PHP5 OO - Protected

```
class OS {
    protected function privatemethod () {
        return "No public access";
    }
}

class newOS extends OS {
    function mymethod() {
        return $this->privatemethod();
    }
}

$myObj = new newOS();
echo $myObj->mymethod();
echo $myObj->privatemethod();
```

Output:

No public access

Fatal error: Call to protected method OS::privatemethod() from context '' in foo.php on line 16

PHP5 OO - Static Members

```
class OS {
   static public $counter = 0;
   public function __construct() {
      self::$counter++;
   }
   public function __destruct() {
      self::$counter--;
   }
}

print OS::$counter . "<br />";
$myOS = new OS();
$myOS2 = new OS();
print OS::$counter . "<br />";
$myOS = NULL;
print OS::$counter . "<br />";
```

Output:

0
2
1

PHP5 OO - Interfaces

```
interface MyBehavior {
    public function speak();
}

class Dog implements MyBehavior {
   public function speak() {
       return "Woof!";
   }
}

$myDog = new Dog();
if ($myDog instanceof MyBehavior) {
   echo $myDog->speak();
}
```

Output:

Woof!

PHP5 OO - Type Hinting

```
class Animation {
    private $myObj;

    function __construct(MyBehavior $objToAnimate) {
       $this->myObj = $objToAnimate;
    }
}

$myDog = new Dog();
echo "Passing Dog Object: <br />";
$objAnim = new Animation($myDog);
echo "Passing DOMElement: <br />";
$objAnim = new Animation(new DOMElement("test"));
```

PHP5 OO - Type Hinting

Output:

Passing Dog Object:

Passing DOMElement:

Catchable fatal error: Argument 1 passed to Animation::__construct() must implement interface MyBehavior, instance of DOMElement given, called in foo.php on line 26 and defined in foo.php on line 17

PHP 4 Object Iteration

```
class MyClass {
   var $var1 = 1;
   var $var2 = 2;
   var $var3 = 3;
}

$myObj = new MyClass();
foreach ($myObj AS $key=>$val) {
   echo "{$key} => {$val} \n";
}
```

Output:

```
var1 => 1
var1 => 2
var1 => 3
```

PHP 5 Iterators

```
class Alphabet implements Iterator {
    private $cur = 65;

    public function current() {   return chr($this->cur); }
    public function key()       {     return $this->cur;      }
    public function next()      {    $this->cur++;            }
    public function rewind()  {   $this->cur = 65;          }
    public function valid()     {   return ($this->cur <= 90);    }
}

$alphabet = new Alphabet();

foreach ($alphabet AS $ndx=>$letter) {
   print $letter;
}
```

Output:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

The Standard PHP Library (SPL)

- Captures some common patterns
- Advanced Iterators
- Advanced Array access
- File and directory handling
- Makes __autoload() useable
- Exception hierarchy with documented semantics
- http://www.php.net/manual/en/ref.spl.php

Error Handling

- Select Error levels
- Log Errors
- Display Errors
- Format Errors
- Custom Error Handlers

Error Handling

```
error_reporting = E_ALL & ~E_NOTICE
display_errors = On
display_startup_errors = On
log_errors = On
ignore_repeated_errors = On
ignore_repeated_source = On
html_errors = Off
error_log = /var/log/httpd/php-error.log
```

User Error Handling

```
function myErrorHandler($errno, $errstr, $errfile, $errline) {
    if ($errno == E_USER_ERROR) {
        echo "<b>User ERROR</b> [$errno] $errstr<br />\n";
        exit(1);
    }
    echo "Unknown error type: [$errno] $errstr<br />\n";

    return true; // Don't execute PHP internal error handler
}

$old_error_handler = set_error_handler("myErrorHandler");

fdsfs;

trigger_error ("Thrown in script", E_USER_ERROR);
```

New Error Levels

- E_STRICT
  - New in PHP 5.0
  - Not included in E_ALL error level
  - Included in E_ALL with PHP 6.0
- E_RECOVERABLE_ERROR
  - Introduced in PHP 5.2
  - Developer decides if application should continue
  - For example: Used with Type Hinting

Exceptions

- Basic Rules
  - Exceptions are exceptions
  - Never use exceptions for control flow
  - Never ever use exceptions for parameter passing
- Exceptions should be specialized
- Exceptions inherit from exception class
- Exception blocks can be nested
- Exceptions can be re thrown

Exceptions

```
function inverse($x) {
    if (!$x)
        throw new Exception('Division by zero.');
    else return 1/$x;
}

try {
    echo inverse(5) . "\n";
    echo inverse(0) . "\n";
} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}

// Continue execution
echo 'Hello World';
```

Database Support

Databases with PHP 5

- MySQL Improved extension (mysqli)
- SQLite
- PHP Data Objects (PDO)

MySQLi

- Supports MySQL 4.1+ Client API
- Prepared statement support
- Support for tracing, debugging, load balancing and replication functionality
- Multi-Statement queries
- SQLSTATE error codes
- Procedural and OOP Interface
- Fix legacy issues from ext/mysql
  - mysql_pconnect()
  - Automatic connections

MySQLi Example

```
$objLink = new mysqli('localhost', <username>, <password>, 'mydb');

$state = "ME";
$sql = 'SELECT name FROM lkp_state WHERE state=?';

if ($objStmt = $objLink->prepare($sql)) {

    $objStmt->bind_param("s", $state);
    $objStmt->execute();

    $objStmt->bind_result($name);

    $objStmt->fetch();

    print "State: $name\n";
}
```

SQLite

- Bundled with PHP 5 (including library)
- Enabled by default
- Fast and easy interface to 'flatfiles'
- Supports in-memory databases
- No daemon required
- Implements most of SQL92
- Supports both OO and procedural API
- Can use PHP to extend SQL language

SQLite Example

```
/* open connection to memory database */
$db = new SQLiteDatabase("test.db");

/* execute a regular query */
$db->query("CREATE TABLE test(a,b)");
$db->query("INSERT INTO test VALUES('1','2')");

/* retrieve data using an unbuffered query */
$r = $db->unbufferedQuery("SELECT * FROM test",
                                    SQLITE_ASSOC);

/* use object iterators to retrieve the data */
foreach ($r as $row) {
    print_r($row);
}
```

Output

```
Array
(
    [a] => 1
    [b] => 2
)
```

PDO: The Problem

- No consistency of API between DB extensions
- Sometimes no self-consistency within a given extension
- Duplicated code (but not)
- High maintenance

PDO: The PDO Solution

- Move PHP specific stuff into one extension
- Database specific stuff (only) in their own
- extensions
- Data access abstraction, not database
- abstraction

PDO: Features

- Performance
  - Native C code beats a scripted solution
  - Takes advantage of latest PHP 5 internals
- Power
  - Gives you common DB features as a base
  - Still be able to access specialist functions

PDO: What can it do?

- Prepare/execute, bound parameters
- Transactions
- LOBS
- Scrollable Cursors
- SQLSTATE error codes, flexible error handling
- Portability attributes

PDO: Available Drivers

Pretty much all of them:

- MySQL, PostgreSQL
- ODBC, DB2, OCI
- SQLite (2 and 3)
- Sybase/FreeTDS/MSSQL
- Firebird (needs love)

PDO: Persistence

- Connection stays alive between requests
- Can specify your own cache key
- The ODBC driver runs with connection pooling enabled by default
- "better" than PHP-level persistence
  - Pool is shared at the process level
- Can be forced off by setting
  - pdo_odbc.connection_pooling=off

PDO: Data Typing

- Very loose
- Uses strings for data
- Gives you more control over data conversion
- Supports integers where 1:1 mapping exists
- Is float agnostic
  - PDO is precise

PDO: Portability Aids

- PDO aims to make it easier to write db independent apps
- Number of tweaks for this purpose
  - Oracle style NULLs - Translates empty strings into NULLs when fetching data
  - Case folding
    - The ANSI SQL standard says that column names are returned in upper case
    - High end databases (eg: Oracle and DB2) respect this
    - Most others don't

PDO: Portability Aids

- Number of tweaks for this purpose (cont'd)
  - Case folding
    - The ANSI SQL standard says that column names are returned in upper case
    - High end databases (eg: Oracle / DB2) respect this
    - Most others don't
  - Error Handling
    - PDO offers 3 different error modes
    - Attempts to map native codes to SQLSTATE standard codes
    - But still offers native info too

PDO: LOBs via Streams

- Large objects are usually 4kb or more in size
- Advantageous to avoid fetching them until you really need to
- Mature RDBMS offer LOB APIs for this
- PDO exposes LOBS as Streams

PDO: Scrollable Cursors

- Allow random access to a rowset
- Higher resource usage than forward-only cursors
- Can be used to emulate limit, offset style paged queries
- Can be used for positioned updates (more useful for CLI/GUI apps)

PDO: Positioned updates

- An open (scrollable) cursor can be used to target a row for another query
- Name your cursor by setting PDO::ATTR_CURSOR_NAME during prepare()
- UPDATE foo set bar = ? WHERE CURRENT OF cursor_name

Major Hurdles

Performance

&

Security

Overall Performance

- PHP is rarely the bottleneck
- 80-90% front-end
- Usability and perception
- See YSlow - http://developer.yahoo.com/yslow

SquirrelMail

Siege: http://www.joedog.org/JoeDog/Siege

Benchmarking SquirrelMail:

```
% siege -H "Cookie: squirrelmail_language=; SQMSESSID=10c85281f476
        -c 10 http://mail.ubuntu/src/right_main.php -b -r 200
** SIEGE 2.66
** Preparing 10 concurrent users for battle.
The server is now under siege..      done.
Availability:                 100.00 %
Elapsed time:                  64.69 secs
Data transferred:               2.81 MB
Response time:                  0.26 secs
Transaction rate:              30.92 trans/sec
Throughput:                     0.04 MB/sec
Concurrency:                    8.06
Successful transactions:        2000
Failed transactions:               0
Longest transaction:           27.14
Shortest transaction:           0.02
```

Code Caching

Enable APC

```
% siege -H "Cookie: squirrelmail_language=; SQMSESSID=10c85281f47660
        -c 10 http://mail.ubuntu/src/right_main.php -b -r 1000
** SIEGE 2.66
** Preparing 10 concurrent users for battle.
The server is now under siege..      done.
Availability:                 100.00 %
Elapsed time:                  85.32 secs
Data transferred:              13.72 MB
Response time:                  0.08 secs
Transaction rate:             117.21 trans/sec
Throughput:                     0.16 MB/sec
Concurrency:                    9.25
Successful transactions:       10000
Failed transactions:               0
Longest transaction:           11.02
Shortest transaction:           0.00
```

Quick Code Optimization

right_main.php

```
define('SM_PATH','../');

/ SquirrelMail required files. /
include_once(SM_PATH . 'include/validate.php');
//include_once(SM_PATH . 'functions/global.php');
require_once(SM_PATH . 'functions/imap.php');
require_once(SM_PATH . 'functions/date.php');
require_once(SM_PATH . 'functions/mime.php');
require_once(SM_PATH . 'functions/mailbox_display.php');
require_once(SM_PATH . 'functions/display_messages.php');
require_once(SM_PATH . 'functions/html.php');
//require_once(SM_PATH . 'functions/plugin.php');
...
```

Quick Code Optimization

Get rid of dynamic paths and _once

```
include '../include/validate.php';
require '../functions/imap.php';
require '../functions/date.php';
require '../functions/mime.php';
require '../functions/mailbox_display.php';
require '../functions/display_messages.php';
require '../functions/html.php';
```

Quick Code Optimizations

Big Difference from the original 30.92 trans/sec

```
** SIEGE 2.66
** Preparing 10 concurrent users for battle.
The server is now under siege..      done.
Availability:                 100.00 %
Elapsed time:                  57.91 secs
Data transferred:              13.72 MB
Response time:                  0.05 secs
Transaction rate:             172.68 trans/sec
Throughput:                     0.24 MB/sec
Concurrency:                    8.31
Successful transactions:       10000
Failed transactions:               0
Longest transaction:           11.84
Shortest transaction:           0.00
```

Quick Code Optimizations

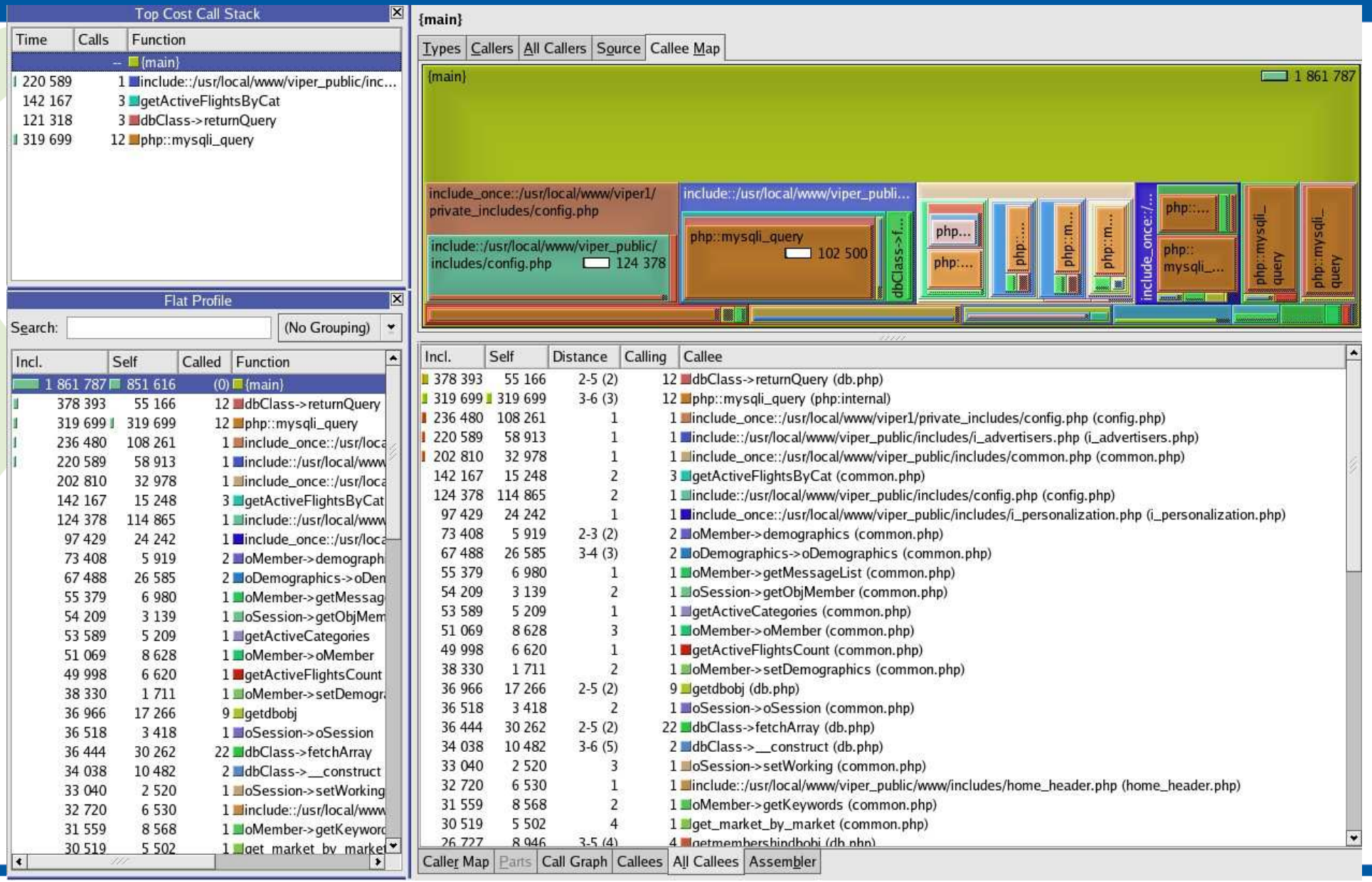
```
% valgrind --tool=callgrind –dump-instr=yes \
        -v --instr-atstart=no /usr/sbin/apache -X

% chmod a+rw callgrind.out.*

 - hit the server to warm it up -
% callgrind_control -i on

% kcachegrind callgrind.out.*
```

Quick Code Optimizations



LOCAL THUNDER

Quick Optimizations

XDebug

```
zend_extension=/usr/local/.../xdebug.so
xdebug.profiler_enable=1
xdebug.profiler_aggregate = On
xdebug.profiler_output_dir=/tmp
```

XDebug

- Written and Maintained by Derick Rethans
- http://www.xdebug.org/index.php
- Profiling
- Code Coverage
- Debugging (including remote)
- Variable Inspection
- Stack Traces
- Function Traces

Xdebug: Code Coverage

eZ Components

Current view: /home/derick/dev/ezcomponents/trunk/Tree/src

Date: Fri Sep 7 11:03:39 CEST 2007

Executable lines: 923

Code covered: 99.89%

Executed lines: 922

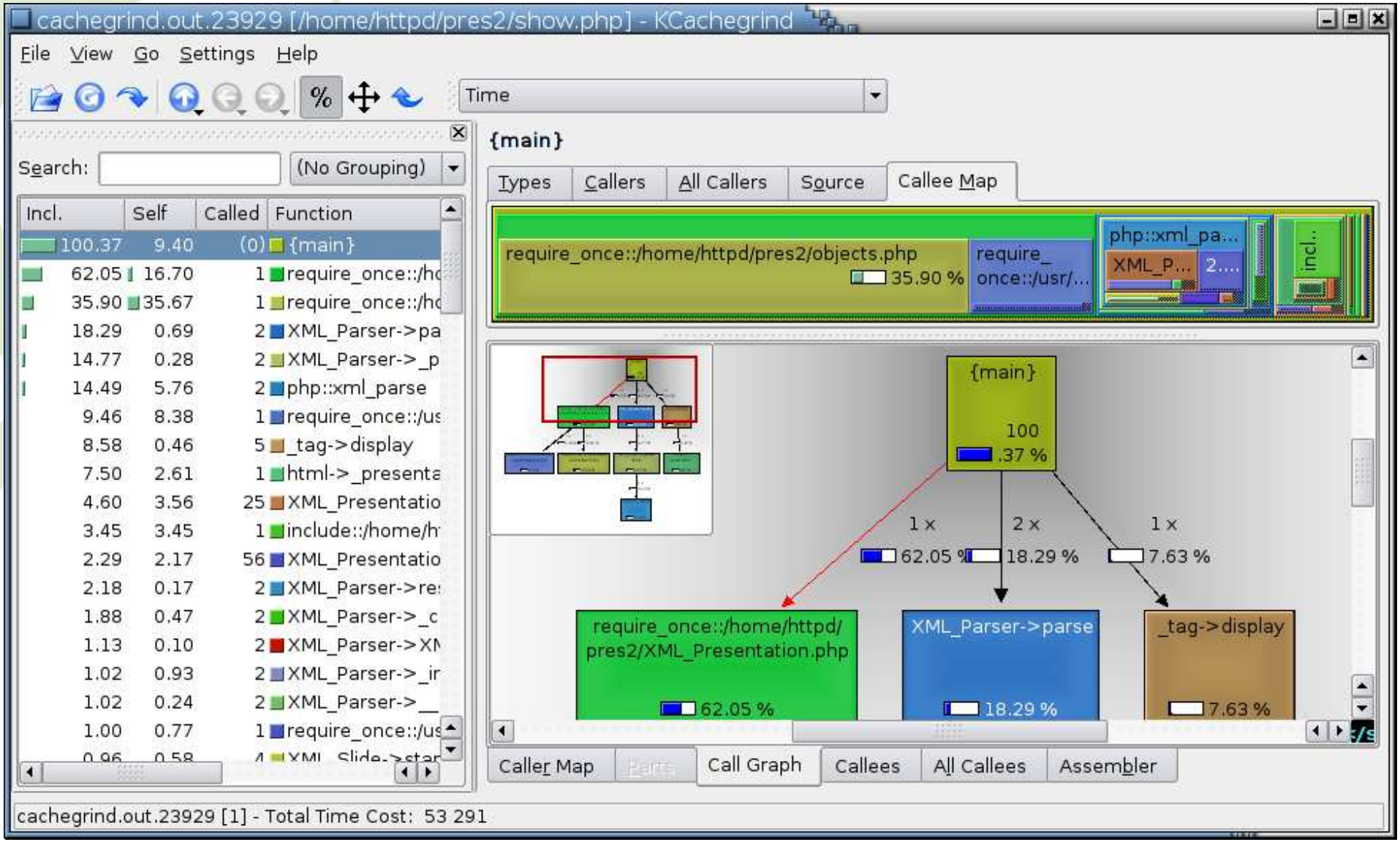
Legend: Low: 0% to 35% | Medium: 35% to 70% | High: 70% to 100%

	Coverage (show details)		
backends		100.00%	391 / 391 lines
exceptions		100.00%	42 / 42 lines
interfaces		100.00%	2 / 2 lines
options		100.00%	39 / 39 lines
stores		100.00%	41 / 41 lines
structs		100.00%	13 / 13 lines
visitors		100.00%	180 / 180 lines
tree.php		98.80%	82 / 83 lines
tree_node.php		100.00%	68 / 68 lines
tree_node_list.php		100.00%	38 / 38 lines
tree_node_list_iterator.php		100.00%	26 / 26 lines

Generated by: PHPUnit 3.1.7 and Xdebug 2.0.1-dev.

Logfile: Plain Text | TAP | XML | Code Coverage

Xdebug: Profiling



LOCAL THUNDER

Security

SQL Injections

```
<?php
    $sql = "
        SELECT card_num, card_expiry
        FROM   credit_cards
        WHERE  uid = '{$_GET['uid']}'
    ";
?>
```

SQL Injections

http://example.com/script.php?uid=42

```
SELECT card_num, card_expiry
FROM   credit_cards
WHERE  uid = '42'
```

SQL Injections

http://example.com/script.php?uid=42'%20or%20"='

```
SELECT card_num, card_expiry
FROM   credit_cards
WHERE  uid = '42' or "="
```

Cross-site scripting (XSS)

```
<?php
    $name = '';
    if ( isset( $_GET['name'] ) ) {
        $name = $_GET['name'];
    }
?>
<html>
    <head><title>Example</title></head>
    <body>
        Name: <?php echo $name; ?>
    </body>
</html>
```

Cross-site scripting (XSS)

http://example.com/script.php?name=rob

Name: rob

Cross-site scripting (XSS)

http://example.com/script.php?
name=<script>alert('!');</script>

Filter Extension

- Comes with PHP 5.2
- Enabled by default
- Provides a default filter
- Provides two groups of accessing filters: sanitizing and logical
- Filters can have options to configure their behavior

Filter Example

```
<?php
if ( isset( $_GET['name'] ) ) {
    $name = filter_input(
        INPUT_GET,
        'name',
        FILTER_SANITIZE_STRING
    );
}
?>
```

Filter Example

http://example.com/script.php?
name=<script>alert('!');</script>

Name: alert('!');

Arbitrary file inlcusion

Security Example

```
<html>
    <head><title>Example</title></head>
    <body>
        <form action="index.php" method="post">
            <select name="language">
                <option value="en">English</option>
                <option vlaue="de">German</option>
                <option value="fr">French</option>
            </select>
        </form>
    </body>
</html>
```

Security Example

```
<?php
$lang = 'en';
if (isset($_POST['language'])) {
    $lang = $_POST['language'];
}

include $lang;
?>
```

Disabling Remote Includes

Within php.ini

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
allow_url_fopen = On

; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
allow_url_include = Off

Questions?

PHP 101
An Introduction to PHP 5

Rob Richards

http://xri.net/=rob.richards

www.cdatazone.org

Resources

Presentations from PHP developers

- http://talks.php.net/index.php/PHP

PHP Manual

- http://www.php.net/manual/en/

PHP 5 Changes

- http://us.php.net/manual/en/migration5.php
- http://us.php.net/manual/en/migration51.php
- http://us.php.net/manual/en/migration52.php