

More About PHP

Jonathan Francis Roscoe

`<jjr6@aber.ac.uk>`

`http://users.aber.ac.uk/jjr6`



November 29th 2011

Outline

Introduction

Some PHP tips

Object-Orientation

Frameworks

Debugging, Testing and Continuous Integration

Secure PHP Code

Environments

Related Reading

Introduction

These slides are based on experience from my industrial year when I developed backed end code for web control panels.

What I hope you'll learn about today:

- Techniques for improving code quality
 - Speed
 - Efficiency
 - Reliability
 - (Re)Usability
- Advanced features of PHP
- Frameworks
- Making your PHP Applications secure
- Debugging and Analysis of PHP applications

I'll mainly focus on PHP but most of this applies to any web development project.

Some PHP tips

With dynamic weak typing, PHP is a laid back language allowing functional code to be written fast.

- Use a strong editor with supporting tools - Notepad++, Eclipse (with plugin), vim, Netbeans
- Use server-side includes and mix PHP and HTML as little as possible
- Test your code (yes - unit testing and acceptance testing is possible)
- Debug your code
- Take security seriously
- Frameworks can significantly improve the speed of your develop, as well as providing features for efficiency, security and reliability.

Server-Side Includes - index.php

```
<?php session_start (); $page = "My Page"
$name = "Jonathan"; $t= gettime (); ?>
<html><head>
  <style type="text/css">
    body {      color: purple;
              background-color: #d8da3d }</style>
<title ><?=$page; ?></title >
</head><body>
Hello <?=$name; ?>
<div id="menu">
<a href="#" Menu item 1/>
<a href="#" Menu item 2/>
</div>
<body></html>
```

Keeping PHP tidy

A better way to handle this code would be to place aspects that are probably wanted in other places into separate files, then we can reuse the same code.

This will make code easier to read and will allow others to focus on front end development (HTML, CSS) without worrying too much about the back-end aspects.

```
<?php
//start the session
session_start();
//set the variables we need on the page
$name = "Jonathan"
$page = "My Page";
$t = time();
?>
```

```
<div id="menu">  
<a href="#" Menu item 1/>  
<a href="#" Menu item 2/>  
</div>
```

```
<html><head>
  <style type="text/css">
    body {
      color: purple;
      background-color: #d8da3d }
  </style>
<title <?=$page; ?></title >
</head>
```

The new index.php

```
<?php
//grab supporting functions
require_once('functions.inc.php');
//grab header
require_once('header.php');
?>
<body>
Hello <?=$name; ?>
<?php require_once('menu.php');
</body>
</html>
```

PHP 5 and Object-Orientation

Introduced in PHP 5, there is a lot in common with Java's syntax, rules and conventions.

- Interfaces and abstract classes
- Magic Methods (e.g. `__toString()`)
- Exception classes and `try..catch`

With simple yet powerful polymorphism:

- Instantiation from String
- `class_exists()`, `method_exists()` and other tricks

PHP 5 introduced a few other significant features:

- Type Hinting in methods
- Reflection
- Method overloading (quite different from Java)

OO PHP Example

```
interface Animal {
    function speak();
}

class Cow implements Animal{
    function speak(){
        print "Moooooo!";
    }
}

$classname = "Cow";
$a = new $classname;
$a -> speak();
```

Reflection in PHP

Reflection is a fantastic concept in programming that you might already be familiar with.

As PHP is weak and dynamic, we can use reflection with powerful results.

For example:

```
$r = new ReflectionClass( 'Cow' );  
$methods = $r->getMethods();
```

Will give you a list of all the methods available. This may not sound like much, but we can use it for quick and easy polymorphic constructions.

Many Reflection functions exist. It's also possible to create a tool to describe undocumented classes you might need to work with.

Frameworks

Frameworks can be compared to more specific software such as blogs (e.g. Wordpress) or Content Management Systems (eg. Drupal).

- Concepts
 - Faster development
 - Reduced overhead
 - Reusability (DRY)
- Support for a number of common needs
 - Session management
 - Database interaction (usually Object Relational Mapping)
 - Ajax
 - Internationalisation
 - Testing
 - Deployment
 - Templating
- Typically based on an MVC approach and often object-oriented

MVC Dispatcher

A dispatcher script is used to handle routes and control the flow of the MVC structure.

- Instantiate model
- Execute controller
- Render view
- `__autoload()` magic method often used to enable source includes on the fly

Typically takes the place of `index.php`.

The following code is a simple example demonstrating how an MVC structured blog might be structured.

Business/domain logic.

```
<?
    class PostModel{
        function static getPosts(){
            return mysql_query("SELECT * FROM
                                blog_posts;");
        }
    }
?>
```

Controls actions of model and view. May handle input from view and instruct model.

```
<?  
    class PostController(){  
        function execute(){  
            $posts = PostModel::getPosts();  
        }  
    }  
?>
```

User input and feedback.

```
<html>
```

```
...
```

```
<? foreach($post in $posts): ?>  
    <h1><?=$post['title'];?></h1>  
    <p><?=$post['body'];?></p>  
<? endforeach; ?>
```

```
...
```

```
</html>
```

Popular Frameworks

- Zend Framework
- Symfony
- Codeigniter
- Rails
- Zope

It can sometimes be difficult to understand how and why PHP code is behaving the way it is, but there are some useful tools:

- Static Analysis - lint, PHP Codesniffer, IDE features
- Dynamic Analysis - xdebug, Valgrind

PHP has error reporting of its own, and some frameworks may provide debugging features.

Browser Extensions

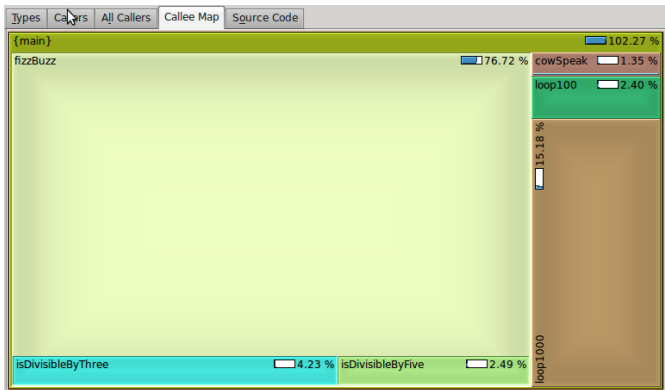
Web development can be a deviation from your familiar development techniques, fortunately various extensions can be invaluable in debugging the information that the client has, particularly when working with Ajax.

- Web Developer Toolbar
 - Firefox
 - Chrome
- Firebug
 - Firefox
 - Chrome, Internet Explorer, etc (Lite)
- IE WebDeveloper

Safari has a built in debug toolbar?..

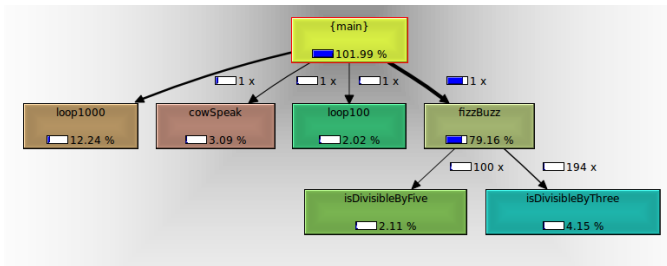
- Speed is particularly important when we have many users placing load on a server
- "Stepping through" is usually not possible
- Xdebug is a PHP extension that provides debugging and performance analysis:
 - Stack & function traces
 - Memory allocation
 - Callgrind profiles
- Callgrind information can be investigated and visualised with a tool such as KCacheGrind, Carica or Webcachegrind.

Profiling



Example call map showing relative runtime of each method called.

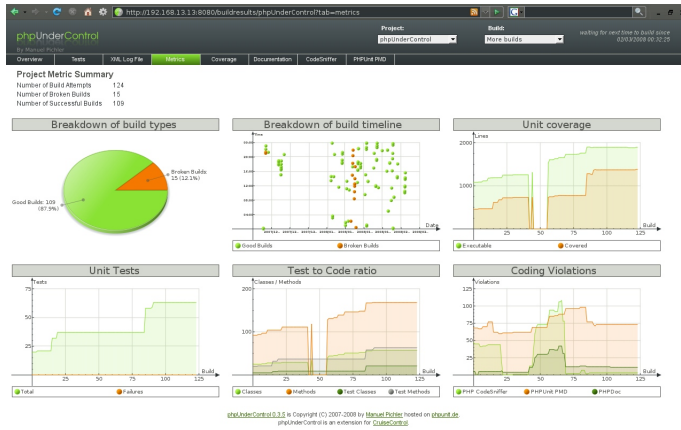
Profiling



Example call graph showing relative runtime of each method called.

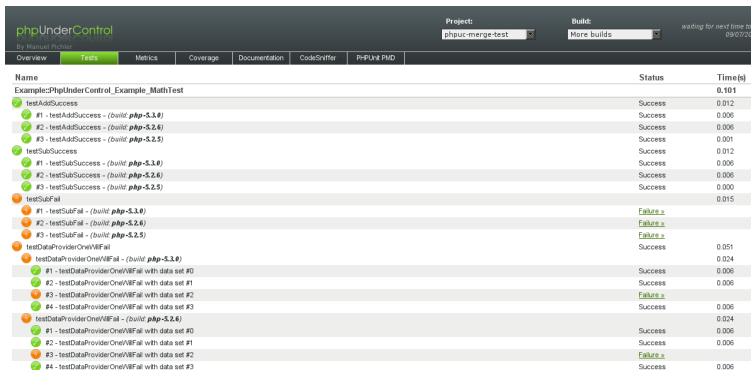
- Code coverage is a measure of how well our tests explore every potential avenue of execution; strive for 100%.
- Unit tests
 - Test each module of code
 - Core to test driven development methodology
 - PHP-Unit is the PHP equivalent of JUnit, a tool you probably already know. You can use it to run subsections of your code, and test expected output.
- Functional tests
 - Tests of end user experience
 - Selenium is a powerful tool that allows automatic execution of your webpages. This makes it easy to fully automate the testing of your entire site.
- Continuous Integration
 - Automated, routine build and testing of code
 - Alert developers of new errors in code before it is released

Continuous Integration



Overview of phpUnderControl showing various statistics. Source: <http://phpundercontrol.org/>

Continuous Integration



Name	Status	Time(s)
Example:PhpUnderControl_Example_MathTest		0.101
testAddSuccess		
✔ #1 - testAddSuccess - (build: php-5.3.0)	Success	0.006
✔ #2 - testAddSuccess - (build: php-5.2.6)	Success	0.006
✔ #3 - testAddSuccess - (build: php-5.2.5)	Success	0.001
testSubSuccess		
✔ #1 - testSubSuccess - (build: php-5.3.0)	Success	0.012
✔ #2 - testSubSuccess - (build: php-5.2.6)	Success	0.006
✔ #3 - testSubSuccess - (build: php-5.2.5)	Success	0.000
testSubFail		0.015
✘ #1 - testSubFail - (build: php-5.3.0)	Failure »	
✘ #2 - testSubFail - (build: php-5.2.6)	Failure »	
✘ #3 - testSubFail - (build: php-5.2.5)	Failure »	
testDataProviderOneWiFiFail		0.051
✘ testDataProviderOneWiFiFail - (build: php-5.3.0)	Success	0.024
✔ #1 - testDataProviderOneWiFiFail with data set #0	Success	0.006
✔ #2 - testDataProviderOneWiFiFail with data set #1	Success	0.006
✘ #3 - testDataProviderOneWiFiFail with data set #2	Failure »	
✔ #4 - testDataProviderOneWiFiFail with data set #3	Success	0.006
testDataProviderOneWiFiFail - (build: php-5.2.6)		0.024
✔ #1 - testDataProviderOneWiFiFail with data set #0	Success	0.006
✔ #2 - testDataProviderOneWiFiFail with data set #1	Success	0.006
✘ #3 - testDataProviderOneWiFiFail with data set #2	Failure »	
✔ #4 - testDataProviderOneWiFiFail with data set #3	Success	0.006

Example result of a phpUnderControl build, highlights code with errors. Source: <http://manuel-pichler.de/>

Sanitising Input

You've probably seen or implemented some form of client-side security - such as a Javascript popup if you haven't don't complete a form properly. These mainly exist to be quick and look pretty, they are not at all secure. So you must **sanitise input on the server-side**.

```
$username = filter_var(FILTER_SANITIZE_STRING,  
    $_POST['username']);
```

Regular expressions can be used to go beyond simply removing harmful characters and can validate some requirements:

```
if(preg_match("/^http/", $_POST['url']))  
echo "Got a URL!";  
else  
echo " URL is no good.;"
```

Securing Your Code

To show you why server-side sanitisation is important:

```
..  
mysql_query("SELECT * FROM users  
WHERE password ='" .$_POST['password']."'");  
..
```

What if the input was something like:

```
' ; DROP users —
```

Fortunately, tools exist to statically analyse your code for oversights and potentially dangerous behaviour. Pixy is a common script used on multi-user systems - such as your Aberystwyth user web hosting..

Security - Server Side Code Scanners

Dear Jonathan ,

Please disable your "PHP-Shell" program as soon as possible . At the moment, what you have basically allows anyone in the world to remove the entire contents of your account!

(if they enter "rm -r /aber/jjr6 /." as the command).

Cheers ,
Alun .

General Rules for Securing PHP

PHP is often used to create public facing websites on the Internet, which typically include privileged areas. There are a few general security issues to be aware of:

- Ensure user is authenticated & authorised as appropriate
- Input should be validated & sanitised
- Avoid careless debugging/error reporting (configure php.ini not to print error messages on a live system)
- Development data (test files, svn metadata) should be removed
- Server should be appropriately configured
- Utilise security software that can isolate holes in your application..

Security Testing Tools

There are plenty of tools available to help you test your site, popular ones include:

- <http://code.google.com/p/skipfish> - Skipfish is an open source web application security scanner from Google
- Metasploit - a generic vulnerability scanner modules exist to test off-the-shelf (wordpress, drupal, etc) software
<http://www.metasploit.com/modules/>
- Nikto2 - another open-source tool
- <http://evuln.com/tools/php-security/> - a free static code scanner for PHP
- <http://loadimpact.com/> - Remote load testing (Are your database queries efficient enough?!)

Environments

Bugs in new code will have a serious impact on usability and security. A common technique for releasing code is to develop on isolated systems, then gradually deploy them in a limited or beta form, before going live.

- Development

Usually the programmer's local machine. May use specialist data sets. Incomplete/untested functionality. Full debugging and error output.

- Testing

A local network server. Provides common test data for all developers. All code should have been partially tested and obtained from development branch of version control. Full debugging and error output.

- Staging/Pre-Production ("Bleeding Edge")

A live server with limited accessibility. May be available to beta users. Should be treated as the live system. All code should have been fully tested and taken from stable branch of version control.

- Production

The live server. Code on this server should not be edited directly. System should be updated through some form of scheduled deployment procedure. Debugging information printed here is a security risk and unpleasant to end users - errors should be caught and politely reported.

Useful PHP Tidbits

- Server configuration information: `phpinfo()`
- Server Side Inclusion:
`include()`, `require()`, `include_once()`, `require_once()`
- Validation/Sanitisation functions:
`preg_match()`, `strip_tags()`, `filter_var()`,
`mysql_real_escape_string()/pg_escape_string()`
- HTML string output: `htmlspecialchars()`
- Special predefined variables:
`$_SERVER[]`, `$_SESSION[]`, `$_ENV[]`
- Find files on the system - `glob()`
- PEAR is a framework for managing PHP extensions, available from repositories such as PECL
- Apache module for nice URLs: `mod_rewrite`
- Many configuration options in `php.ini` - if you manage the server
- Doxygen (generic Javadoc) works well with PHP

Related Reading

- "Billions of Hits: Scaling Twitter" presentation
- <http://www.unmaskparasites.com/> - tool for website vulnerabilities
- <https://www.owasp.org/> Lots of security information including an injections cheat sheet
- <http://www.symfony-project.org/book/> - the Symfony book, lots of good MVC information
- <http://www.php.net/>
- <http://www.phpframeworks.com/> - comparison of frameworks
- <http://browsershots.org/> - screenshots of your website in dozens of configurations (platform, browser, resolution)