# CI:IRL

By Beth Tucker Long

# Who am I?

- Beth Tucker Long (@e3betht)
- Editor-in-Chief - php[architect] magazine
- Freelancer under Treeline Design, LLC
- Stay-at-home-mom

- User group organizer – Madison PHP

# Audience Participation?

- Yes, there will be. So, when I ask the audience a question, don’t be shy about answering.

# Continuous Integration

That’s only for the big guys.

My team is small, my projects are small.

So, why am I up here?

# What is continuous integration?

According to Wikipedia:

In software engineering, continuous integration (CI) implements continuous processes of applying quality control — small pieces of effort, applied frequently. Continuous integration aims to improve the quality of software, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control after completing all development.

http://en.wikipedia.org/wiki/Continuous_integration

- Martin Fowler - http://martinfowler.com/articles/continuousIntegration.html

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

# Step 1

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day.

# Step 2

Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

# Code Quality

Why?

- Easier to test
- Easier to measure
- Easier to follow
- Streamlines the development process

# Measuring Code Quality

- PEAR coding standard - http://pear.php.net/manual/en/standards.php
- PEAR2 coding standard - http://pear.php.net/manual/en/pear2cs.rules.php
- PHP Framework Interoperability Group (PHP-FIG) - http://www.php-fig.org/

# PEAR Coding Standard

- Use an indent of 4 spaces, with no tabs.
- Control Structures:

```
<?php
if ((condition1) || (condition2)) {
    action1;
} elseif ((condition3) && (condition4)) {
    action2;
} else {
    defaultAction;
}
?>
```

# Custom Standards

- Broad
- Strict, but flexible
- Based on a “standard” standard
- Everyone must follow

# PHP_CodeSniffer

"tokenizes your PHP, JavaScript and CSS files and detects violations of a defined set of coding standards“
http://pear.php.net/package/PHP_CodeSniffer

- PEAR package
- Single file or entire directory
- Preset and customizable

# Output

```
$ phpcs /myDir/myFile.php

FILE: /myDir/myFile.php
----------------------------------------------------------------------
FOUND 3 ERROR(S) AFFECTING 3 LINE(S)
----------------------------------------------------------------------
 2  | ERROR | Missing file doc comment
20 | ERROR | PHP keywords must be lowercase;
                  expected "false" but found "FALSE"
47 | ERROR | Line not indented correctly;
                   expected 4 spaces but found 1
----------------------------------------------------------------------
```

# Unit Tests

- **Unit** - the smallest piece of testable code within my application or script.

- **Unit test** - a piece of code that executes the unit and then evaluates the result returned.

# Tips

- Make sure the unit is small enough so the test is testing a single function.
- Make sure the test is efficient enough to run repeatedly, perhaps even a thousand times a day.
- Make sure the tests do not depend on each other. Each test should be able to run completely separately from other tests.

```
function validateName($name) {
    if ((strlen($name) > 1) && (strlen($name) < 50)) {
        if (ctype_alpha(str_replace(array(" ","","","-","'"),"",$name))) {
            return true;
        }
        else {
            return false;
        }
    }
    else {
        return false;
    }
}

assert(validateName("Beth's Test Name"));
```

# How Many Tests?

Enough to test every basic function of the code.

# Testing Frameworks

- Standardize test format
- Easily run tests
- Analyze results

PHPUnit - http://www.phpunit.de

Pros:

- Good documentation
- Lots of examples online
- Integrates with many other popular tools and platforms

Cons:

- Command line only

SimpleTest - http://simpletest.sourceforge.net/

Pros:

- Run on command line or in browser
- Can test front-end functionality

Cons:

- Not as integrated as PHPUnit

Selenium-WebDriver - http://seleniumhq.org/

Pros:

- Can test front-end functionality
- Makes direct calls to the browser using each browser's native support for automation

Cons:

- Not a native PHP tool, but bindings are available from several third-parties, including one from Facebook

# Automate The Build

- Perform a DB query to update the schema, clear a cache, upload files, run cron tasks, etc.

Phing - http://phing.info

- PHP project build system
- Based on Apache Ant
- XML build files and PHP "task" classes
- Integrates with both PHPUnit and SimpleTest as well as phpDocumentor
- Platform independent
- No required external dependencies

Maven - http://maven.apache.org

- Supports Ant tasks
- Large library of third-party plug-ins to integrate other continuous integration tools
- Helps shield you from the details of the build
- For Java-based projects, so you'll need Maven for PHP: http://www.php-maven.org/

**Phing Buildfile:**

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="FooBar" default="dist">

  <!-- ============================================  -->
  <!-- Target: prepare                               -->
  <!-- ============================================  -->
  <target name="prepare">
    <echo msg="Making directory ./build" />
    <mkdir dir="./build" />
  </target>

  <!-- ============================================  -->
  <!-- Target: build                                 -->
  <!-- ============================================  -->
  <target name="build" depends="prepare">
    <echo msg="Copying ./about.php to ./build directory..." />
    <copy file="./about.php" tofile="./build/about.php" />
  </target>
  <!-- ============================================  -->
  <!-- (DEFAULT)  Target: dist                       -->
  <!-- ============================================  -->
  <target name="dist" depends="build">
    <echo msg="Creating archive..." />
    <tar destfile="./build/build.tar.gz"
     compression="gzip">
      <fileset dir="./build">
        <include name="*" />
      </fileset>
    </tar>

    <echo msg="Files copied and compressed in build
     directory OK!" />
  </target>
</project>
```

# Documentation

- phpDocumentor 2: http://www.phpdoc.org/
- Merging phpDocumentor and DocBlox
- Automates documentation
- Tutorial:

  http://manual.phpdoc.org/HTMLSmartyConverter/HandS/phpDocumentor/tutorial_phpDocumentor.howto.pkg.html

```
/**
 * Put your short description here.
 *
 * Put your long description here.
 * You may use multiple lines.
 * You can even use Markdown.
 *
 * @author  Beth Tucker Long <beth@musketeers.me>
 *
 * @since 1.0
 *
 * @param int $exampleA  This is a method parameter description.
 * @param string $exampleB This is another example.
 */
```

\ global

NAMESPACES

global

## Functions

Creates a packager object with all basic options set.

```
createPackager(string $original_file, string[] $options) : \
PEAR_Error | \PEAR_PackageFileManager2
```

### Parameters

**$original_file**

`string` Path of the original package.xml.

**$options**

`string[]` Set of options to merge in.

### Returns

`\PEAR_Error` `\PEAR_PackageFileManager2`

Generate a XHProf Display View given the various URL parameters as arguments.

```
displayXHProfReport(object $xhprof_runs_impl, array $url_par
ams, string $source, string $run, string $wts, string $symbo
l, $sort, string $run1, string $run2)
```

# Continuous Integration Tools

- CruiseControl
  - Written in Java
  - Binary distribution, a Windows Installer and the source distribution
  - Flexible scheduling system
  - Notifications via e-mail, messaging or viewing HTML reports
  - Integrates with Phing and Maven
  - PHPUnderControl - optional add-on application for integrating PHP_CodeSniffer and PHPUnit

# Hudson and Jenkins

- Built on Java
- Installed via native packages or a war file
- Easily configured via a GUI web interface
- Extensive library of third-party plug-ins
- RSS, e-mail or instant messaging options for build notifications
- Template for Jenkins Jobs for PHP Projects (by Sebastian Bergmann)

# Reporting

- Sonar
  - Integrates with Hudson and Jenkins
  - PHP plug-in to integrate it directly with other PHP-based tools
  - Web-based application
  - Overall “health” of project, drill down for details
  - Includes TimeMachine

# Technical Debt Plugin

Assigns a technical debt value

- **The debt ratio -** The debt ratio gives a percentage of the current technical debt of the project versus the total possible debt for the project.
- **The cost to reimburse –** A dollar amount for what it would cost to clean all defects.
- **The work to reimburse -** The cost expressed in work days.
- **The breakdown -** A pie chart showing how the debt is distributed across 6 categories: Duplication, Violations, Complexity, Coverage, Documentation and Design.

# A Little Help

- TeamCity by JetBrains
  **TeamCity** is a user-friendly continuous integration (CI) server for professional developers and build engineers, like ourselves. It is trivial to set up and absolutely free for small teams.
  http://www.jetbrains.com/teamcity/

- Introductory blog post:
  http://blog.jetbrains.com/webide/2013/01/continuous-integration-for-php-using-teamcity/

# A Little Help

- NetBeans has support for continuous integration (Template for Jenkins Jobs for PHP Projects)

- More info: https://blogs.oracle.com/netbeansphp/entry/continuous_integration_support

# Yes, But…

- Project is small, budget is small…
- Evaluate which tools are worthwhile to your specific project.

# Make It a Deliverable

- Consider including unit tests or code cost/coverage reports in your deliverables to your customers as an added value to them (and you down the road).

# Quick Recap

- Coding Standards -> PHP_CodeSniffer
- Unit Tests -> PHPUnit, SimpleTest, Selenium
- Build -> Phing or Maven
- CI Tools -> CruiseControl, Hudson/Jenkins
- Documentation -> PHP_Documentor 2
- Reporting -> Sonar

Project:

A customer hires you to create a registration form for a one-time event. It's a small customer with a small budget. It should take a couple hundred lines of code in a single file, results will be e-mailed. It will be tested by the event staff and the marketing department on the live site as they do not have a test environment, and it will only be live for two months.

What they need:

1. If they do not have an in-house standard for you to follow, write it using one of the main coding standards, like PEAR.
2. Create unit tests for it.

What they don’t need:

1. In-depth reporting
2. Full automation, including build.
3. Documentation

Project:

A customer hires you for an ongoing project. On the 15th of every month, they need you to go in and add a new survey to collect data and write it to a database. The previous month's survey data needs to be backed up and cleared out of the database when the new survey goes live.

What they need:

1. If they do not have an in-house standard for you to follow, write it using one of the main coding standards, like PEAR.
2. Create unit tests for it and use a testing framework.
3. Automate the build.

What they don’t need:

1. In-depth reporting (Maybe)
2. Documentation (Maybe)

Project:

A customer hires you to write one part of a very large application. Other consultants that you do not have access to will be working on other parts of the application at the same time.

What they need:

1. All of it

In this situation, see if you can convince them to get everyone working on a unified continuous integration platform utilizing a complete suite of continuous integration tools from standards to documentation and fully automated everywhere in between.

# Take Away #1

Not everything is beneficial enough to use in every situation, so choose the right tools for your project and needs.

# Take Away #2

The fewer steps I have to remember to do manually,

the more successful

my project will be.

# Resources

- CruiseControl - http://cruisecontrol.sourceforge.net
- Guide to writing your own PHP_CodeSniffer standards (Official) - http://pear.php.net/manual/en/package.php.php-codesniffer.coding-standard-tutorial.php
- Guide to writing your own PHP_CodeSniffer standards (Alternate) - http://luhman.org/blog/2009/12/17/setting-custom-coding-standards-php-codesniffer
- Hudson - http://hudson-ci.org
- Jenkins - http://jenkins-ci.org
- Maven - http://www.php-maven.org
- PEAR coding standard - http://pear.php.net/manual/en/standards.php
- PEAR Package Manager Installation - http://pear.php.net/manual/en/installation.php
- PEAR Packages Installation - http://pear.php.net/manual/en/guide.users.commandline.installing.php
- PEAR2 coding standard - http://pear.php.net/manual/en/pear2cs.rules.php
- Phing - http://phing.info
- PHP Standards Working Group - http://groups.google.com/group/php-standards
- PHP_CodeSniffer - http://pear.php.net/package/PHP_CodeSniffer
- phpDocumentor 2 - http://www.phpdoc.org/
- PHPUnit - http://www.phpunit.de/manual/3.6/en/automating-tests.html
- phpUnderControl - http://phpundercontrol.org
- Selenium - http://seleniumhq.org/
- SimpleTest - http://simpletest.sourceforge.net/
- Sonar - http://www.sonarsource.org
- Sonar PHP Plug-in - http://docs.codehaus.org/display/SONAR/PHP+Plugin
- Sonar Technical Debt Plugin - http://docs.codehaus.org/display/SONAR/Technical+Debt+Plugin
- Template for Jenkins Jobs for PHP Projects by Sebastian Bergmann - http://jenkins-php.org


# php[architect] magazine


## http://phparch.com


Ask me about writing articles for the magazine!

Feedback

# https://joind.in/10543

Twitter: e3betht

E-mail: Beth@musketeers.me

Slides Available:
http://www.TreelineDesign.com/slides