Workshop Handout

Phix Project

October 2011

Stuart Herbert – stuart@stuartherbert.com

Attribution-Share Alike 2.0 UK: England & Wales

You are free:

to copy, distribute, display, and perform the work

to make derivative works

under the following conditions:

Attribution: you must give the original author credit.

Share Alike: If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.
- Nothing in this license impairs or restricts the author's moral rights.

Your fair dealing and other rights are in no way affected by the above.

This is a human-readable summary of the license. The full legal text can be found on the Creative Commons website.

Table of Contents

Workshop Introduction

Phix Project

Thank you for your interest in learning how to make packaged components of PHP code. This handout accompanies your workshop today.

What Is Phix?

Phix is a simple command-line tool that makes it very easy to create and maintain packaged components of PHP code. It standardises and automates the more tedious aspects of creating PHP components.

What Are PHP Components?

PHP components are (normally) reusable libraries of PHP code. Each component is designed to do one job, and do it well.

You will find reusable code quicker to create, easier to test and maintain ... and being able to reuse each others' code frees your time up for what makes your app unique.

What We Cover In The Workshop

In this workshop, you will learn the basics of creating PHP components:

1. How to setup your development environment for creating PHP components
2. How to create, test, and publish your component
3. The key practices that your component needs to follow

Please do ask questions as you go along!

Where To Go For Further Information

We hope this workshop answers all of your questions, and enables you to start creating your own PHP components straight away.

If you need more information after this workshop, please visit our websites:

- http://phix-project.org/
- http://blog.stuartherbert.com/php/php-components/

Copyright © 2011 Stuart Herbert

Commands Cheat Sheet

Phix Project

Use this cheat sheet to remind yourself which command gets the job done when you're working on your PHP components.

Commands List

To …	Run This Command …
Create a new component in the current folder	`phix php-library:init .`
To upgrade a component's skeleton files to the latest version	`phix php-library:upgrade .`
Download all of the component's dependencies into the 'vendor' folder	`phing build-vendor`
Check a component for PHP syntax errors	`phing lint`
Run the unit tests	`phing test`
Create a PEAR package	`phing pear-package`
Install the PEAR package into the 'vendor' folder for further testing	`phing pear-package && phing install-vendor`
Install the PEAR package onto your local computer	`phing pear-package && sudo phing install-system`
Publish the PEAR package to a local copy of your PEAR channel	`phing pear-package && phing publish-local`

Division Of Responsibilities

The general rule of thumb is:

- Use the 'phix' command to create and maintain the component's skeleton files
- Use the 'phing' command *inside your component's folder* to test, package, and publish your component.

New Component Cheat Sheet

Phix Project

Use this cheat sheet to remind you of the steps you need to follow whenever you create a new component using Phix.

Create An Empty Component

Run the following commands to create a brand-new, empty component. Substitute $HOME/Devel for wherever you normally keep your source code files.

```
mkdir $HOME/Devel/<component>
cd $HOME/Devel/<component>
phix php-library:init .
```

Editing The Component's Metadata Files

You need to edit your component's metadata files next:

Edit This File ...	To ...
build.properties	Define your component's name, version number, PEAR channel.
LICENSE.txt	Define the license terms that your component will be released under.
package.xml	Define your component's dependencies.
README.md	Document the information your users will need to read first.

Finally ...

Before you start writing your component's code, download the component's dependencies into the 'vendor' folder. This will allow your unit tests to run.

```
cd $HOME/Devel/<component>
phing build-vendor
```

Remember to rebuild the vendor folder if you change the dependencies listed in the package.xml file!

Copyright © 2011 Stuart Herbert

Version Numbers Cheat Sheet

Phix Project

Use this cheat sheet to remind you when to change the version number of your PHP components.

When To Update The Version Numbers

When you change your PHP component, remember to change the component's version number inside the build.properties file:

When You Make This Change ...	... The Version Number Becomes
Fix a bug	X.Y.Z+1
Add a new feature	X.Y+1.0
Break backwards-compatibility	X+1.0.0
Add major new features	X+1.0.0

Further Advice

You can find further guidance on changing your version numbers from the Semantic Versioning website:

- http://semver.org/

Prep: Dev Environment

Phix Project

Before we begin the exercises today, let's take a moment to make sure that your dev environment is setup and working, to avoid problems throughout the day.

Setting Up Your Dev Environment

If you haven't already, you need to get PHP 5.3 CLI, the PEAR Installer, and phix all installed onto your computer or virtual machine.

There are instructions and one-line installers available for several popular operating systems:

- Instructions for Apple OS X
- Instructions for CentOS Linux
- Instructions for Debian Linux
- Instructions for Fedora Linux
- Instructions for Ubuntu Linux

If you are not using one of these operating systems, generic installation instructions are also available on the Phix Project's website.

Setting Up Your PEAR Channel

During this workshop, you will publish components to a PEAR channel. Please setup the default vhost on your laptop to publish the PEAR channel.

- http://blog.stuartherbert.com/php/2011/03/30/setting-up-your-own-pear-channel/

Copyright © 2011 Stuart Herbert

Exercise #1: HelloWorld

Phix Project

This is a very simple exercise to help you create your very first component as quickly as possible.

Requirements

Create a component that:

1. Defines a class called 'HelloWorld'.
2. Defines a method called 'HelloWorld::getGreeting()' that returns the string “greetings, programs!”.
3. Defines a method called 'HelloWorld::getPurpose()' that returns the string “I fight for the users!”.

Unit Tests

You should create unit tests for your component that provides 100% code coverage.

Question: how will you prove that your unit tests provide 100% code coverage?

Publish To A PEAR Channel

Once your component is completed and tested, setup a PEAR channel and publish your component.

Test your PEAR channel by installing your component using the PEAR installer, and then write a simple PHP script to call the methods on the component.

When You Have Finished

Once you have published your own component, please download, install, and test the components created by the person sat next to you in this workshop.

Exercise #1 Review

Phix Project

Here's a review of everything we learned in Exercise #1. During the workshop, please don't read ahead until you have completed Exercise #1 for yourself; you'll learn more that way!

What We Learned

1. How to create an empty component
2. How to edit the component's metadata
3. Where to put our PHP classes
4. Where to put our component's unit tests
5. How to generate a code-coverage test report
6. How to publish a PEAR channel
7. The importance of namespaces for our components

Copyright © 2011 Stuart Herbert

Exercise #2: Adding Features

Phix Project

This simple exercise shows you the steps to follow to add features to your PHP component.

Part #1

Requirement:

1. Add a new method, HelloWorld::getCompleted(), which returns the string "End of Line, man!" to the caller.

Unit tests:

- Ensure that there is 100% code coverage of the unit tests

Publish:

- Publish this component to your PEAR channel

Questions:

1. What version number will you use for this component?

Part #2

Requirement:

1. Add a namespace (either a PHP 5.3 namespace, or just a prefix to the classname) to the HelloWorld class.

Unit tests:

- Ensure that there is 100% code coverage of the unit tests

Publish:

- Publish this component to your PEAR channel

Questions:

1. What version number will you use for this component?
2. How will you decide what namespace to use?

Exercise #2 Review

Phix Project

Here's a review of everything we learned in Exercise #2. During the workshop, please don't read ahead until you have completed Exercise #2 for yourself; you'll learn more that way!

What We Learned

1. When we add minor new features, we increment the 'Y' part of the version number; our components moved from v1.0.0 to v1.1.0.
2. When we break backwards-compatibility, we increment the 'X' part of the version number; our components moved from v1.1.0 to v2.0.0.
3. We need to choose namespaces that are not generic, that we can reasonably claim as our own for all eternity.
4. The namespace has two components to it: the top-level identifies our project or organisation, and the level underneath it identifies the component name itself.
5. Adopt PHP 5.3 namespaces for your components to make reusing the code cleaner.

Copyright © 2011 Stuart Herbert

Exercise #3: Converting Legacy Code

Phix Project

In this exercise, we're going to take some legacy code and practice converting it into a modern PHP component.

This is a group exercise.

Exercise Steps

1. Pick an existing package from pear.php.net. It doesn't matter which one, as long as you are confident you can convert it in the time available.
2. Convert the package to a PHP component, in the 'PEAR\<packagename>' namespace.
3. Add 100% code coverage to the tests.
4. Publish the completed component on your PEAR channel.

When You Have Finished

... we will discuss the problems you encountered, how you overcame then, and especially how the whole process made you feel.

Exercise #3 Review

Phix Project

Here's a review of everything we learned in Exercise #3. During the workshop, please don't read ahead until you have completed Exercise #3 for yourself; you'll learn more that way!

What We Learned

1. Good practices for creating PHP components have evolved a lot since the PEAR project was initially created.
2. Using PHP 5.3 namespaces changes how we would structure reusable PHP code, compared to just prefixing class names with our own namespace string.
3. Adding unit tests after the event is hard (and sometimes frustrating!) work.
4. Incomplete unit tests make refactoring code harder than it needs to be.

Copyright © 2011 Stuart Herbert