Kazaam
PHP Modular Website Framework
User Guide
version 0.5.0 *beta*

by Keith Roberts

March 29, 2013

NB: This user guide is currently a work in progress.

Contents

1 Introduction to Kazaam

1.1 Chapter Overview

This chapter covers:

- Target Audience
- Documentation Overview
- Kazaam Overview
- Kazaam System Requirements

1.2 Target Audience

Kazaam is designed to make it easy for anyone, from novice to professional, to create usefull websites. A novice should find the framework, module structure, and page structure easy to grasp. A seasoned web developer should take to Kazaam like a duck to water. Whatever your level of expertise, using Kazaam should speed up the website development and maintenance process considerably.

1.3 Documentation Overview

About the documentation layout. Do this bit last when the document is finished!

1.4 Kazaam Overview

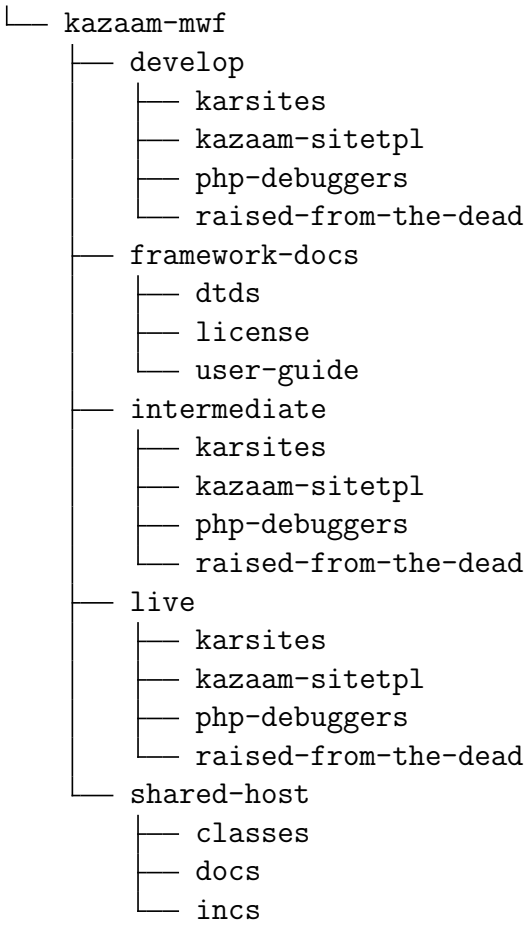
1.4.1 In a Nutshell

The Kazaam framework consists of three code branches called *develop*, *intermediate* and *live*, and their supporting directories and files.

code branches consist of websites and their support files.

Websites consist of modules and their support files.

Modules consist of pages and their support files.



1.4.2 Kazaam Design Goals

To create a website framework that is:

- Modular - allowing modules from one Kazaam website to be 'plugged-in' to other Kazaam websites, promoting code reuse.
- Extensible
- Easy to learn
- Easy to use and navigate
- W3C compliant
- Supports Apache name-based virtual hosts
- Uses 100% native code

Kazaam is a modular website framework for developing and deploying websites. By virtue of it's modular design, this makes the virtual websites in the framework easy to create, update, and extend. As each module in a website uses a uniform directory structure, once you have learnt the layout for one module, then finding your way around other modules is exactly the same and just as easy.

The framework consists of three code branches: a *develop*(ment) code branch, an *intermediate* code branch, and a *live* code branch. The *develop* and *intermediate* code is not intended to be viewable online. Only websites in the *live* directory are. This allows development code to be updated and then tested in the *intermediate* directory, before being moved across to the *live* directory, without fear of breaking an existing live website.

All website development is done in the *develop* code branch. The *intermediate* and *live* branches are copied from the development code, using PHING scripts. The development sequence is:

develop $\triangleright$ *intermediate* $\triangleright$ *live*

You can skip the *intermediate* step if you don't need it.

If you host your websites on a remote server, then you can setup kazaam on your local machine, and use the *develop* directory of kazaam for developing and testing websites on your local machine. You can then use your ftp program to upload the contents of the website in the *develop* directory, from your local machine to the remote host. This will allow you to keep backups of your website code on your local machine.

In the case of using a remote hosting package, the development sequence would be:

local-machine develop $\triangleright$ *remote-machine live*

Each Kazaam website and associated pages uses:

- 100% xhtml
- 100% CSS
- 100% Java-script
- 100% PHP 5

so you do not have to learn any sort of template language at all. With Kazaam you can clone a new website from a standard website template[1]. You can then extend the website's functionality, by adding more modules to the website.

In theory, there is no limit to the number of websites you can create within the Kazaam framework, or the number modules you can create and add to each website. This limit is most likely up to the maximum amount of sub-directories your OS allows, or how much space you have left on your hard drive.

Additionally, any modules you have created for one website in the Kazaam framework, may be copied across to another website (or websites) in the same framework, and 'plugged-in' and re-used by the other website(s).

1.5 Kazaam System Requirements

To use Kazaam you will need the following software installed:

- Apache 2.2.x web server (not tested with any other server).
- Apache 2.2.x PHP 5 module.
- PHP 5 CLI interpreter.
- PHING (Phing is a project build system based on Apache ant)

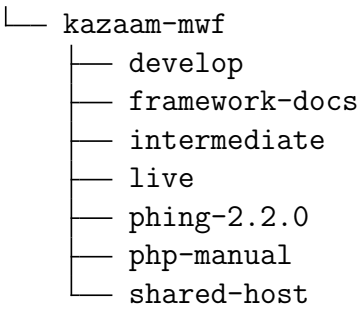
[1]The website template I refer to here is actually the *kazaam-sitetpl* directory, in the *develop* code branch.

2 The Kazaam Framework

2.1 Chapter Overview

This chapter discusses the framework that all virtual hosts live in, and the directories and files shared by those virtual hosts.

2.2 The topmost directory.



develop is where all website development is done. Each sub-dir in *develop* contains the code for a complete website. None of the websites in *develop* are accessible for public viewing on-line.

framework-docs contains documentation for the Kazaam framework.

intermediate contains sub-dirs of websites copied from the *develop* directory using PHING scripts, and is also not accessible for public viewing on-line. A website's code can be tweaked in the *intermediate* dir before being pushed to the *live* directory.

For example, there might be code still under development in the *develop* dir, that you don't want to push into the *live* dir yet. Copying the website code to the *intermediate* dir allows you to check the site is how you want it to be seen by others. So you can make changes to things you don't want copied from the *inter*mediate directory into the *live* directory for public viewing. This can be done automatically with the PHING script **int-build.xml.**

live contains sub directories of all websites that are available on-line for public viewing.

phing-2.2.0 this is where I have put my PHING installation. However, if you have installed PHING using your OS's package manager, or the

PEAR installer, your PHING installation is probably in a different location.

php-manual contains a symbolic link to the PHP5 manual *html* directory. This allows PHP5 to display the relevant manual page when a PHP error occurs.

shared-host contains files that are shared by all virtual hosts on a particular physical server, running a web server such as Apache.

Once you have run the relevant PHING scripts that move the website code from *develop* to *intermediate*, and from *intermediate* to *live*, the directory structure for the *intermediate* and *live* sub-directories should be similar to the *develop* directory. (What actually gets copied across depends on how you have modified your PHING scripts.) So the following example for the *develop* directory applies to *intermediate* and *live* directories too.

2.3 Contents of *develop di*rectory.

```
develop
├── karsites
├── kazaam-sitetpl
├── php-debuggers
└── raised-from-the-dead
```

This directory example contains four websites:

- karsites
- kazaam-sitetpl
- php-debuggers
- raised-from-the-dead

The naming convention used is generally the main part of the website's domain name. *kazaam-sitetpl* is a template used by Kazaam for creating new websites. It must not be modified, unless you know *exactly* what you are doing.

2.4 Contents of *shared-host di*rectory.

This directory contains sub-directories of files shared by all pages in all virtual websites, accesible via the **$SHARED_HOST_CLASSES or $SHARED_HOST_INCS** php variables. *docs* is a place to put any notes with regard to files in *shared-host*.

```
shared-host
├── classes
├── docs
└── incs
```

3 Kazaam Websites

3.1 Chapter Overview

This chapter looks at the directory structure for the *kazaam-sitetpl* website template, using the *develop* directory as an example. (Because websites can be copied from *develop* to *intermediate*, and then from *intermediate* to *live*, this discussion also applies to the *intermediate* and *live* directory structures.)

3.2 A Kazaam website (top-level) directory structure.

```
kazaam-sitetpl/
├── fedora-services/
├── home/
├── module-example1/
├── module-example2/
├── module-example3/
├── new-ext-module/
├── new-module/
├── shared/
├── shared-host/
├── tmp-backups/
├── website-docs/
├── create-new-ext-module.xml
├── create-new-ext-module.xml.bak
├── create-new-module.xml
├── create-new-module.xml.bak
├── int-build.xml
├── int-build.xml.bak
├── int-delete.xml
├── int-delete.xml.bak
├── live-build.xml
├── live-build.xml.bak
├── live-delete.xml
├── live-delete.xml.bak
├── robots.txt
└── robots.txt.bak
```

fedora-services is an example module. It contains some outline pages for listing all the services available in Fedora Linux.

home Each website has a *home* module. The *home* module contains the website's home page, and pages accessed from the top link bar.

module-exampleX are more example modules. You can make backup copies these and fiddle around with the original modules to see how the code works.

new-ext-module is a module template directory. It is used by Kazaam as a template to clone new modules with external page content. This must not be modified unless you are absolutely sure you know what you are doing.

new-module is a module template directory. It is used by Kazaam as a template to clone new modules with combined content. This must not be modified unless you are absolutely sure you know what you are doing.

shared is a set of sub-directories and associated files that are common to all pages in a particular website.

shared-host contains */incs/* sub-dir with symbolic links to files in: */kazaam-mwf/shared-host/incs/*

This directory of symbolic links allows each virtual host to access the host-wide shared include files that are outside of their virtual host DocumentRoot.

The */kazaam-mwf/shared-host* directory is on the same level as *develop*, *intermediate*, and *live*.
So all virtual hosts in all code branches of the kazaam framework can include any of the files in */kazaam-mwf/shared-host/incs/.*

For example: if a physical server machine is going off-line for a while, you can alter *kazaam-mwf/shared-host/incs/host.js* and uncomment the line:

// onPageLoad(showMaintenanceMesg);

This will use DHTML to display a customizable maintenance message, that appears at the top of all pages in all virtual hosts on a particular server. This happens whenever a page is reloaded, or a new page is requested.

tmp-backups contains temporary backups for this directory.

website-docs contains all the development documentation for this particular website.

3.3 Files in the top-level directory

create-new-ext-module.xml is a phing script to create a new external-content module for this website.

create-new-module.xml is a phing script to create a new module for this website.

int-build.xml is a phing script to copy the *develop* code branch of this website to the *intermediate* code branch.

int-delete.xml is a phing script to clean up the contents of the *intermediate* directory for this website.

live-build.xml is a phing script to copy the *intermediate* code branch of this website to the *live* code branch.

live-delete.xml is a phing script to clean up the contents of the *live* directory for this website. Be carefull with this script, as it will delete everything in a live website! Best used before a website is comissioned.

robots.txt is used to control search engine access to any parts of this website.

3.4 Contents of *shared* directory.

```
shared/
├── classes/
├── images/
├── incs/
├── meta-menus/
└── themes/
```

classes contains PHP5 classes that can be included by all pages in this website, if needed.

images contains image files that can be included by all pages in this website, as needed.

incs contains the following:

```
incs/
├── tmp-backups/
├── admin-details.php
├── admin-page-check.php
├── admin_link_bar.php
├── copyright.php
├── disclaimer.php
├── domain.conf.php
├── homepage-menu.php
├── meta-info.php
├── page-head.php
├── right-menu.php
├── session.funcs.php
├── site-config.php
├── site-js.js
├── top_link_bar.php
└── user.conf.php
```

admin-details.php contains the website administrator's logon (username and password) details.

admin-page-check.php is the website administrator's logon page.

admin_link_bar.php provides a customizable top_link_bar for the website administrator's use only.

copyright.php contains copyright bar content which is loaded at the bottom of each page**.**

disclaimer-body.php contains content for the */home/anyuser/disclaimer.php* page.

domain.conf.php Virtual host configuration file. Used to set **$WEBSITE_ROOT** and **$DOMAIN_NAME** values. Different values apply for each virtual host in the *develop*, *intermediate* and *live* code branches.

homepage-menu.php contains meta-menus for the *left_column* <**div**> part of a page. This is the left-hand menu that appears in a 3-column and 2-column page.

meta-info.php contains meta-tags for the <**head**> section of a page. This file is required by *page-head.php*

page-head.php contains the <**head**> section for every page in a particular virtual host.

right-menu.php contains meta-menus for the *right_column* <**div**> part of a page. This is the right-hand menu that appears in a 3-column page.

session.funcs.php contains custom session handling functions for PHP. Needs rewriting to use sqlite database.

site-config.php is a website's global configuration file. It is used to set directory paths for the framework and modules. This file is included by default at the top of every webpage that needs it.

site-js.js global javascript file for all pages in a particular website.

top_link_bar.php is the site navigation bar that appears below the site-logo banner.

user.conf.php global configuration file included by site-config.php.

4 Kazaam Modules

4.1 Chapter Overview

This chapter looks at the structure of a Kazaam module, using the *'new-module'* template in the *kazaam-sitetpl* website template, as an example.

To add new functionality to a website, you can use the PHING script **create-new-module.xml.** This script will create a new module from the *new-module* template directory. The new module will be named using the name you specify at the start of the module creation script. This is how a website can be extended in a modular way.

4.2 A Kazaam module top-level directory structure.

These are the sub-directories within each module:

```
new-module/
├── admin/
├── anyuser/
├── auth-user/
├── local/
├── module-docs/
└── page-templates/
```

admin may contain web-pages or scripts to administer this module. Each module can have it's own administration pages, independent of other modules. If you copy a module to another virtual host, the module's admin pages will also go with it. Access to the *admin* directory can be controlled using Apache's httpd.conf authorization directives. If there are no administration pages to this module the *admin* directory will be empty.

anyuser contains the web-pages that are accessible to anyone on the internet. There are no access restrictions on these pages.

auth-user may contain web-pages with restricted access rights. These pages are only accessible to authorized users. Access to the *auth-user* directory can be controlled using Apache's httpd.conf authorization directives. If there are no pages for this module with restricted access the *auth-user* directory will be empty.

local is a collection of various sub-dirs and files for this module.

module-docs is a place to keep documentation about this particular module.

page-templates contains a set of page templates automatically configured to work with this module. These are created by the *create-new-module.xml* PHING script.

4.3 Contents of *anyuser* directory

```
anyuser/
├── tmp-backups/
├── 1-column-page.php
├── 2-column-page.php
├── 3-column-page.php
└── new-module.php
```

1-column-page.php is a page template without the *left_column* or *right_column* <**div**> parts. This page's content window will fill the entire width of the screen, without displaying any left hand column or right hand column menus. Usefull for a page with very wide content.

2-column-page.php is a page template with the *left_column* <**div**> part, but without the *right_column* <**div**> part. This page's content window will fill the right-hand side of the screen, and also display the left hand column menu. The left hand column menu can be either the website's default homepage menu, or the module's own customized left hand menu.

3-column-page.php is a page template with the *left_column* and *right_column* <**div**> parts. This page's content window will fill the middle section of the screen, and also display the left hand column menu, and the right hand column menu. The left hand column menu can be either the website's default homepage menu, or the module's own customized left hand column menu. The right hand column menu can be either the website's default right menu, or the module's own customized right hand menu.

new-module.php is the homepage for this module. Each module has a homepage named after the module. When you create a new module with the PHING script *create-new-module.xml* (or *create-new-ext-module.xml*), this page is also created, and renamed to the name of the new module.

4.4 Contents of *local* directory

```
local/
├── classes/
├── images/
└── incs/
```

classes is a container for PHP5 classes owned exclusively by this module. It may be empty if there are no PHP5 classes for this module.

images is a container for images owned exclusively by this module. It may be empty if there are no images for this module.

incs is a sub-dir containing the following:

```
incs/
├── mod/
├── tmp-backups/
├── new-module.conf.php
└── new-module.conf.php.bak
```

new-module.conf.php is the module's configuration file. Here you can set options that will apply to all pages in a particular module, such as the sql-database username and password for this module.

mod is a sub-dir containing the following:

```
mod/
├── tmp-backups/
├── new-module-menu.php
└── new-module-right-menu.php
```

new-module-menu.php is a PHP file that pulls in the ***/shared/incs/homepage-menu.php*** *left_column* menu, or contains content for it's own exclusive *left_column* menu.

new-module-right-menu.php is a PHP file that pulls in the ***/shared/incs/right-menu.php*** *right_column* menu, or contains content for it's own exclusive *right_column* menu.

5 Kazaam Pages

5.1 Chapter Overview

This chapter examines the structure of a typical xhtml page in Kazaam, using the 3-column page template from the 'new-module' module template as an example. We look at each line of the page, and see how the page is built on the server, using Apache's PHP5 module.

5.2 Structure of a 3-Column Kazaam Page

When viewing the source code in a browser, the following 53 lines of code will generate a typical web page that is approximately 1100 lines long.

3-Column Page Template

```
01 <?php require "../../shared/incs/site-config.php"; ?>
02 <?php require "$MODULE_CONFIG/new-module.conf.php"; ?>
03
04 <?php $META_KEYWORDS = '*keywords* meta information -
05 may span multiple lines'; ?>
06
07 <?php $META_DESCRIPTION = '*description* meta information -
08 may span multiple lines'; ?>
09
10 <?php $MODULE_NAME = "$sitename - 3 Column page template"; ?>
11
12 <?php require "$SHARED_INCS/page-head.php"; ?>
13
14 <body>
15
16 <div id="main_window_3col">
17
18 <div class="box2">
19   <h3>New Module</h3>
20 </div>
21
22 <div class="box2">
23   <p>Main content for page goes here.</p>
24 </div>
```

```
25
26 <div class="update">
27   Last updated: Fri 23 Feb 07
28 </div>
29
30 </div> ~<!-- end of main_window_3col content -->
31
32 <img id="top_logo"
33      src="<?php echo $SHARED_IMAGES ?>/site-logo"
34      alt="site-logo" />
35
36 <div id="top_link_bar">
37   <?php require "$SHARED_INCS/top_link_bar.php"; ?>
38 </div>
39
40 <div id="left_column">
41  <?php require "$MODULE_PATH/new-module-menu.php" ?>
42 </div>
43
44 <div id="right_column">
45  <?php require "$MODULE_PATH/new-module-right-menu.php" ?>
46 </div>
47
48 <div id="copyright">
49   <?php require "$SHARED_INCS/copyright.php" ?>
50 </div>
51
52 </body>
53 </html>
```

A Kazaam 2-column page template is based on the 3-column page template, but does not have the *right_column* <**div**> part, as in lines 44-46.

A Kazaam 1-column page template is based on the 3-column page template, but does not have the *left_column or right_column* <**div**> parts, as in lines 40-46.

5.3 How a Kazaam Page Loads

line 01 loads the global configuration file for the website in:
website-name*/shared/incs/site-config.php* , where ***website-name*** is

a website's directory under the *develop*, *intermediate* or *live* code branches. This global configuration file sets a number of PHP5 path variables, for including other parts of the page.

line 02 loads the configuration file for the module this page belongs to: ***website-name****/new-module/local/incs/new-module.conf.php*

lines 04 - 08 these four lines override the value of the **$META_KEYWORDS** and $**META_DESCRIPTION** PHP5 variables. These two variables can be set globally for the whole of the website in ***website-name****/shared/incs/user.conf.php,* overriden in each module's configuration file to apply to all pages in the module, or set individually in each page. This allows these two meta-tags to be customised on a page-by-page basis. Good for SEO. (search engine optimisation.)

line 10 **$MODULE_NAME** is initially set in *new-module.conf.php* . This allows any number of pages in a module to use all or part of this value in the <**title**> tag for a page. Each page may append to this value, or totally overwrite it. The value of **$MODULE_NAME** is the string displayed as the page's <**title**> when it loads in the browser. see line 14, **Inside page-head.php** on the following page, for details of how it's used.

line 12 loads the <**head**> part of the page: ***website-name****/shared/incs/page-head.php* . For more about *page-head.php* see **Inside page-head.php** on the next page.

line 14 is the start of the page's <**body**> section.

lines 16 - 30 *main_window_3col* is the centre column and content for this 3-column page. This is first in the source code to make it easier for search engines to find the page's content.

lines 32 - 34 load the banner to display across the top of each page: ***website-name****/shared/images/site-logo* . Most browsers can identify the image type by the *site-logo* file contents. Using an image filename without an extension allows any image file to be copied to this *site-logo* file, and used as the site banner.

lines 36 - 38 load the ***website-name****/shared/incs/top_link_bar.php* . This is the site navigation bar that appears below the *site-logo*.

lines 40 - 42 load the:
website-name*/new-module/local/incs/mod/new-module-menu.php* file. This file uses it's own custom menu for the module, or pulls in the ***website-name****/shared/incs/homepage-menu.php* file instead. The menu is displayed in the left hand column of the page.

lines 44 - 46 load the:
website-name*/new-module/local/incs/mod/new-module-right-menu.php* file. This file pulls in the ***website-name****/shared/incs/right-menu.php* file, which is displayed in the right hand column of the page.

lines 48 - 50 pull in the ***website-name****/shared/incs/copyright.php* file. This is the copyright bar that is displayed at the bottom of each page.

lines 52 - 53 mark the end of the page's <**body**> and <**html**> sections.

5.4 Inside *page-head.php*

Here we take a look at what's happening inside:
website-name*/shared/incs/page-head.php*

---

page-head.php

```
01 <?php
02  // database session management code
03  // require "$SHARED_INCS/session.funcs.php";
04  // session_start();
05 ?>
06
07 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
08  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
09
10 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
11
12 <head>
13
14 <title><?php echo $MODULE_NAME ?></title>
15
16 <?php require "$CONFIG_PATH/meta-info.php"; ?>
17
```

```
18 <!-- ================================== -->
19 <!-- files common to this particular website only -->
20
21 <script type="text/javascript"
22   src="<?php echo $SHARED_INCS ?>/site-js.js">
23 </script>
24
25 <?php /* for development purposes only */ ?>
26 <style type="text/css">
27   <?php require "$SITE_THEMES/site-css.php"; ?>
28 </style>
29
30 <?php /* for live pages only */ ?>
31 <!-- link rel="stylesheet" type="text/css"
32 href="<?php /* echo $SITE_THEMES */ ?>/site-css.css" / -->
33
34 <!-- ================================== -->
35 <!-- files shared by all websites on this server -->
36
37 <style type="text/css">
38   <?php require "$SHARED_HOST_INCS/host.css"; ?>
39 </style>
40
41 <script type="text/javascript"
42   src="<?php echo $SHARED_HOST_INCS ?>/host.js">
43 </script>
44
45 <!-- ================================== -->
46
47 <script type="text/javascript"
48   src="<?php echo $SHARED_HOST_INCS ?>/x_core.js">
49 </script>
50
51 <script type="text/javascript"
52   src="<?php echo $SHARED_HOST_INCS ?>/x_event.js">
53 </script>
54
55 <!-- ================================== -->
56
57 </head>
```

5.5 How *page-head.php* builds the <head> section

All pages in a kazaam website use ***website-name****/shared/incs/page-head.php*, where ***website-name*** is a website's directory under the *develop*, *intermediate* or *live* code branches. So any modifications made to *page-head.php* will apply globally to one particular website.

lines 03 - 04 uncomment these two lines to enable the use of PHP5's session management functions. **Need to rewrite these as a PHP5 OOP class.**

lines 07 - 10 set the document type for all pages in this website to XHTML 1.0 Transitional.

line 12 is the start of the <**head**> section for the page.

line 14 $MODULE_NAME is initially set in *new-module.conf.php* and may also be overidden in each page before *page-head.php* is loaded. After *page-head.php* has loaded, this line echoes the value of $MODULE_NAME and sets the <title> tag for the current page. See line 10 in the section about page structure starting on page 16 .

line 16 loads ***website-name****/shared/incs/meta-info.php* . This file contains global meta-tags and associated values that are applied to all pages by default. The '*keywords*' and '*description*' meta-tags can be set globally for the whole of a particular website in ***website-name****/shared/incs/user.conf.php,* overriden in each module's configuration file to apply to all pages in the module, or set individually in each page. This allows these two meta-tags to be customised on a page-by-page basis, if required. Good for SEO. (search engine optimisation.)

lines 21 - 23 load the ***website-name****/shared/incs/site-js.js* javascript file for this website.

lines 25 - 28 load the ***website-name****/shared/themes/default/site-css.php* CSS file for this website. This is a PHP-enabled CSS file, used for development purposes only. See **CSS Site Themes** on page ?? for more details.

lines 30 - 32 load the ***website-name****/shared/themes/default/site-css.css* CSS file for this website. This is a standard CSS file, derived from *site-css.php*, and used for the live code branch.

lines 37 -49 load the ***website-name****/shared-host/incs/host.css* CSS file. This is actually a symbolic link to:
kazaam-mwf/shared-host/incs/host.css . This CSS file is part of the Kazaam framework, and used by all pages in all websites.

lines 41 -43 load the ***website-name****/shared-host/incs/host.js* Javascript file. This is actually a symbolic link to:
kazaam-mwf/shared-host/incs/host.js . This Javascript file is part of the Kazaam framework, and used by all pages in all websites.

line 57 is the end of the <**head**> section for the page.

6 Kazaam Configuration files

6.1 Chapter Overview

This chapter looks at the different types of configuration files in Kazaam. It then looks at the order in which these configuration files load, and the structure of each of these files in detail.

6.2 Introduction

There are four types of configuration files in Kazaam:

1. ***website-name***/shared/incs/*site-config.php*
2. ***website-name***/shared/incs/*user.conf.php*
3. ***website-name***/shared/incs/*domain.conf.php*
4. ***module-name***/anyuser/local/incs/*module-name*.conf.php

where ***website-name*** is the name of the respective website you are looking at either in the *develop*, *intermediate* or *live* code branches.

Each module has it's own configuration file, named after the module, as in item 4 above; and ***module-name*** is a module under ***website-name*** . All webpages belong to a module, and each webpage may use the module's configuration file, if needed. See **Contents of *local* directory**, on page 15 for more details.

6.3 Loading order and precedence

1. *site-config.php*
2. *user.conf.php*
3. *domain.conf.php*
4. *module-name*.conf.php

The configuration files are loaded in the above order. So any variables that are set in configuration files that load later, will take precedence over the same variables in configuration files that have loaded earlier.

Variables of the same name set in *user.conf.php*, will override the same variables set in *site-config.php* .

Variables of the same name set in *domain.conf.php,* will override the same variables set in *user.conf.php* and *site-config.php* .

Variables of the same name set in *module-name.*conf.php, will override the same variables set in *domain.conf.php, user.conf.php* and *site-config.php* .

Generally there should be no need to override variables set by configuration files that have loaded earlier. The only variables that you may need to override are those you set for groups of pages in a module's configuration file, or individually for each page, such as **$META_KEYWORDS,** $**META_DESCRIPTION**, and **$MODULE_NAME**.

6.4 site-config.php

This is the main configuration file that is required at the start of a web page. It sets required paths for loading other parts of the page.

6.4.1 Contents of site-config.php

---

site-config.php

```
01 <?php
02
03 // kazaam-sitetpl global website configuration.
04 // file-id: /kazaam-sitetpl/shared/incs/site-config.php
05
06 // include this file at the top of each page.
07 // this config file will be encoded by ioncube encoder
08 // hiding the path layouts
09
10 /* ---------------------------------------- */
11 // module-specific shared files
12
13 // path to module specific config files
14 $MODULE_CONFIG = "../local/incs";
```

```
15
16 // path to this module's local file content
17 $MODULE_PATH = "../local/incs/mod";
18
19 // path to this module's local images
20 $LOCAL_IMAGES = "../local/images";
21
22 // path to this module's local include files
23 $LOCAL_INCS = "../local/incs";
24
25 // path to this module's local class files
26 $LOCAL_CLASSES = "../local/classes";
27
28 /* ---------------------------------------- */
29 // site-wide shared file paths
30
31 // path to site-wide shared include files
32 // and other site-wide includes
33 $CONFIG_PATH = "../../shared/incs";
34
35 // path to site-wide shared include files
36 // and other site-wide includes
37 $SHARED_INCS = "../../shared/incs";
38
39 // path to site-wide shared php classes
40 $SHARED_CLASSES = "../../shared/classes";
41
42 // path to site-wide shared meta-menus
43 $SHARED_META_MENUS = "../../shared/meta-menus";
44
45 // path to site-wide shared images
46 $SHARED_IMAGES = "../../shared/images";
47
48 // path to site-wide style themes
49 $SITE_THEMES = "../../shared/themes/default";
50
51 /* ---------------------------------------- */
52
53 // allow site admin to alter the website configuration globally
54 // this config file will not be encoded
55 // all files ending with .conf.php are not meant
```

```
56 // to be encoded
57 require "$SHARED_INCS/user.conf.php";
58
59 ?>
```

6.4.2 Variables inside site-config.php

Not all of these path variables are used - but they are defined here just in case they are needed.

line 14 $MODULE_CONFIG = "../local/incs";
path to the module's configuration file. It is used in the second line of a page, after site-config.php has loaded. See line 02 of the **3-Column Page Template** on page 16 for how it is used.

line 17 $MODULE_PATH = "../local/incs/mod";
path to the module's menu files.

line 20 $LOCAL_IMAGES = "../local/images";
path to the image files used exclusively by this module.

line 23 $LOCAL_INCS = "../local/incs";
path to the module's local include files.

line 26 $LOCAL_CLASSES = "../local/classes";
path to the module's local PHP classes.

line 33 $CONFIG_PATH = "../../shared/incs";
path to site-wide website configuration files.

line 37 $SHARED_INCS = "../../shared/incs";
path to site-wide shared include files.

line 40 $SHARED_CLASSES = "../../shared/classes";
path to site-wide shared PHP classes.

line 43 $SHARED_META_MENUS = "../../shared/meta-menus";
path to site-wide shared meta-menus. Used to build the LH column and RH column menu structures for each page.

line 46 $SHARED_IMAGES = "../../shared/images";
path to site-wide shared image files. Used for example, to load the site-logo image. See line 33 of the **3-Column Page Template** on page 16 for how it is used.

line 49 $SITE_THEMES = "../../shared/themes/default";
path to site-wide shared CSS stylesheet files.

line 57 require "$SHARED_INCS/user.conf.php";
this line loads the *user.conf.php* configuration file.

6.4.3 Path Synonyms

$MODULE_CONFIG is a synonym for $LOCAL_INCS. Both point to ../local/incs . $MODULE_CONFIG is used for the path to a module's configuration file, whereas $LOCAL_INCS can be used as a path to any other type of module include file(s).

$CONFIG_PATH is a synonym for $SHARED_INCS. Both point to ../../shared/incs . $CONFIG_PATH is used for paths to site-wide configuration files, whereas $SHARED_INCS can be used for any other type of site-wide include file(s).

6.5 user.conf.php

This file is required by *site-config.php* .

6.5.1 Contents of user.conf.php

---

user.conf.php

```
01 <?php
02
03 // kazaam-sitetpl user configuration file
04 // file-id: /kazaam-sitetpl/shared/incs/user.conf.php
05
06 /**
07 this config file is included by site-config.php as plain text,
08 and not encoded. Values set here apply globally to a site,
09 but may be overidden in module-name.conf.php files.
10 **/
11
12 // uncomment this line to turn OFF all error messages globally
13 // error_reporting(0);
```

```
14
15 /* ------------------------------------------- */
16 // this required config file will not be encoded
17 // and contains variables to set domain specific paths
18
19 require "$SHARED_INCS/domain.conf.php";
20 /* ------------------------------------------- */
21
22 // meta keywords tag
23 $META_KEYWORDS = 'keyword default content for
24   kazaam-sitetpl website here - may span multiple lines';
25
26 // meta description tag
27 $META_DESCRIPTION = 'description default content for
28   kazaam-sitetpl website here - may span multiple lines';
29
30 /* ------------------------------------------- */
31 // database connectivity
32 // each website has at least one database with the same
33 // name as the website
34
35 // Mysql hostname for this website
36 $hostname = 'localhost';
37
38 // default database for this website
39 $dbname = 'kazaam-sitetpl';
40
41 // $username is set for each module in
42 // module-name/local/incs/module-name.conf.php
43
44 // $password is set for each module in
45 // module-name/local/incs/module-name.conf.php
46 /* ------------------------------------------- */
47
48 ?>
```

6.5.2 Variables inside user.conf.php

lines 03 - 04 when you create a new website with the PHING script *create-new-website.xml* , the string value 'kazaam-sitetpl' will be replaced with

the name of the new website cloned from the kazaam-sitetpl template. This helps you identify which website the configuration file belongs to.

line 13 // error_reporting(0);
uncomment this line to turn OFF all PHP error messages globally. May not be needed. Setting the values in httpd.conf might be better.

line 19 require "$SHARED_INCS/domain.conf.php";
this line loads the domain.conf.php configuration file, discussed in the next section. Note - this require statement has moved to the end of user.conf.php as of kazaam-mwf

lines 23 - 24 $META_KEYWORDS = 'keyword default content for kazaam-sitetpl website here - may span multiple lines';
Sets the meta-keywords value globally for a website.

lines 27 - 28 $META_DESCRIPTION = 'description default content for kazaam-sitetpl website here - may span multiple lines';
Sets the meta-description value globally for a website.

line 36 $hostname = 'localhost';
sets the hostname for mysql database connectivity.

line 39 $dbname = 'kazaam-sitetpl';
sets the name of the default database for this website.

6.6 domain.conf.php

This is required by *user.conf.php* . This file sets domain-specific paths for a particular website. Each code branch for a website uses different paths to access either the *develop*, *intermediate* or *live* code branches. When you create a new website with the PHING *create-new-website.xml* script, these domain.conf.php files are set up by the script, in the respective code branches.

6.6.1 Contents of domain.conf.php

---

domain.conf.php

```
01 <?php
```

```
02
03 /**
04 This config file is used to set domain specific paths
05 and values for a website in the develop, intermediate
06 and live directories. It is included by user.conf.php
07 as plain text, and not encoded. Values set here apply
08 globally to a site, and may NOT be overidden anywhere else.
09 Do NOT overwrite this file when updating site code from
10 develop or intermediate code branches. Different values
11 apply for each develop, intermediate and live directory.
12 **/
13
14 // URL for this website - used for validating xhtml & CSS.
15 // This needs to be set to the name of the virtual domain name
16 // when in a virtual server live directory container.
17 $DOMAIN_NAME = "http://www.kazaam-mwf.net";
18
19 // Full path to website's base directory relative
20 // to DocumentRoot setting.
21 // This needs to be set to "" when in a virtual server
22 // live directory container.
23 $WEBSITE_ROOT =
24 "/KAR/kazaam-mwf/kazaam-mwf/develop/kazaam-sitetpl";
25
26 // sitename to be used in page <title> tags.
27 $sitename = 'kazaam-sitetpl - DEVELOPMENT';
28
29 // path to shared-host include files.
30 // These include files are shared by all virtual hosts
31 // on this physical web server.
32 $SHARED_HOST_INCS = "../../../../shared-host/incs";
33
34 // Path to shared-host class files.
35 // These include files are shared by all virtual hosts
36 // on this physical web server.
37 $SHARED_HOST_CLASSES = "../../../../shared-host/classes";
38
39 ?>
```

6.6.2 Variables inside domain.conf.php

line 17 $DOMAIN_NAME = "http://www.kazaam-mwf.net";
sets the domain name for the website. Used by the ***website-name****/shared/meta-menus/page-checker.php* meta-menu, for validating xhtml and CSS, with W3C's online validators.

lines 23 - 24 $WEBSITE_ROOT =
"srv/kazaam-mwf/develop/kazaam-sitetpl";
sets the path to the *develop*, *intermediate*, or *live* code branch, and a website within that branch. Used by meta-menus to load pages residing on localhost. This does not include Apache's DocumentRoot, but follows on from there.

line 27 $sitename = 'kazaam-sitetpl - DEVELOPMENT';
sets the name of the website and the code branch it lives in. Used as part of the <**title**> tag for all pages in a website.

line 32 $SHARED_HOST_INCS = "../../../../shared-host/incs";
sets the path to include files that are shared by all virtual hosts on this physical server.

line 37 $SHARED_HOST_CLASSES = "../../../../shared-host/classes";
sets the path to PHP class files that are shared by all virtual hosts on this physical server.

6.7 *module-name*.conf.php

Each module has it's own configuration file. All pages that belong to a module may use the module's configuration file, if needed.

This example looks at
kazaam-sitetpl/new-module*/local/incs/new-module.conf.php* .

6.7.1 Contents of new-module.conf.php

For each new module that you create with *create-new-module.xml* , or *create-new-ext-module.xml* , a module configuration file is also created based on this pattern.

---

new-module.conf.php

```
01 <?php // new-module module config file.
02
03 // include this file at the top of each page
04 // after the main site-config.php file.
05
06 /* ------------------------------------------- */
07
08 // this global variable sets the <title> for this module
09 $MODULE_NAME = "$sitename - New Module";
10
11 // each website has a database corresponding to the same
12 // name as the website
13
14 // Mysql login details for this module
15 $username = 'mysql-username'; // username for this module
16 $password = 'mysql-password'; // password for this module
17
18 // $hostname is set for each website in user.conf.php
19 // $dbname is set for each website in user.conf.php
20
21 /* ------------------------------------------- */
22
23 ?>
```

As a module's configuration file has the highest precedence (apart from individual pages), each module may connect to a different database by overriding the *$dbname* variable here, with a new database name (which is initially set in ***website-name****/shared/incs/user.conf.php).*

6.7.2 Variables inside new-module.conf.php

line 09 $MODULE_NAME = "$sitename - New Module";
sets the string value to be echoed in the <**title**> tag for each page in this module. This applies to all pages belonging to this module that use this configuration file. $MODULE_NAME can also be modified in each page - either appended to, or completely overwritten. This value

is initially set when you use the PHING scripts *create-new-module.xml*, or *create-new-ext-module.xml* .

line 15 $username = 'mysql-username';
the mysql username this module uses to connect to the mysql database.

line 16 $password = 'mysql-password';
the mysql password this module uses to connect to the mysql database.

** up to here **

7 The PHING build scripts

7.1 Chapter Overview

PHING scripts are used by Kazaam to create new websites, new modules for websites, and to copy a website from the *develop* to *intermediate,* and *intermediate* to *live,* code branches. Understanding how these scripts work will be helpfull when you use them in the next chapter.

7.2 create-new-website.xml

```
develop
├── karsites
├── kazaam-sitetpl   <- website-template
├── php-debuggers
├── raised-from-the-dead
└── create-new-website.xml
```

The create-new-website.xml script lives in the *develop* directory of the Kazaam framework, and is used to create a new website in the Kazaam framework.

It prompts for the name of the new website, and copies the *kazaam-sitetpl* website-template directory, to a new website directory in the *develop* code branch. After copying the *kazaam-sitetpl* template to a new directory, various regular-expression string transformations are done on the newly created website's code. This customises the new website's code, for this particular website.

It then sets up the *domain.conf.php* files for this website in the *intermediate* and *live* code branches.

7.2.1 Structure of create-new-website.xml

Here we take a look at the create-new-website.xml script.

---

create-new-website.xml

```
01 <?xml version="1.0" ?>
```

```
02
03 <!--
04
05 Phing script to create a new website,
06 by copying the kazaam-sitetpl website template,
07 to a new directory in the develop code branch.
08
09 to run it use:
10
11 # phing -f create-new-website.xml
12
13 in the CWD of the above script (the develop code branch).
14
15 Tell phing the name of the new website you want to create,
16 and phing will do it for you.
17
18 -->
19
20 <project name="Create new website"
21          default="finished" basedir=".">
22
23 <target name="t1">
24   <echo></echo>
25   <input propertyname="websiteName">
26 Please enter name of new website:</input>
27  <echo>Creating new kazaam '${websiteName}' website</echo>
28 </target>
29
30
31 <!-- copy 'kazaam-sitetpl' website template
32      to the ${websiteName} directory -->
33 <target name="t2" depends="t1">
34
35  <echo msg="Copying kazaam-sitetpl website template
36 to new '${websiteName}' website directory..." />
37
38   <echo msg="" />
39  <echo msg="Please wait. This could take some time..." />
40   <echo msg="" />
41
42   <copy todir="${websiteName}">
```

```
43     <!-- Fileset for all files to be copied -->
44     <fileset dir="kazaam-sitetpl">
45       <include name="**/*" />
46       <!-- exclude name="**/about.*" /-->
47     </fileset>
48   </copy>
49 </target>
50
51
52 <!-- change text strings in all files in newly created
53     ${websiteName} directory to work with this
54      new website directory -->
55 <target name="t3" depends="t2">
56   <echo msg="replacing 'kazaam-sitetpl' text with
57 '${websiteName}'" />
58
59   <reflexive>
60     <fileset dir="${websiteName}">
61       <include name="**/*" />
62     </fileset>
63
64     <filterchain>
65       <replaceregexp>
66         <regexp pattern="kazaam-sitetpl"
67                 replace="${websiteName}" />
68       </replaceregexp>
69     </filterchain>
70
71   </reflexive>
72
73 </target>
74
75
76 <target name="t4" depends="t3">
77  <echo msg="New kazaam website '${websiteName}' created OK!" />
78 </target>
79
80
81 <!-- initialise intermediate directory
82      with domain.conf.php files -->
83
```

```
84 <!-- copy '${websiteName}/shared/incs/domain.conf.php
85     files to intermediate/${websiteName}/shared/incs/
86      directory -->
87
88 <target name="t5" depends="t4">
89   <echo msg="Copying domain.conf.php files to new
90 'intermediate/${websiteName}/shared/incs/' directory..." />
91
92  <copy todir="../intermediate/${websiteName}/shared/incs/">
93     <!-- Fileset for all files to be copied -->
94     <fileset dir="./${websiteName}/shared/incs/">
95       <include name="**/domain.conf.php*" />
96       <!-- exclude name="**/about.*" /-->
97     </fileset>
98   </copy>
99 </target>
100
101
102 <!-- change text strings in domain.conf.php files
103      in newly created ${websiteName} directory,
104      to work with this new website -->
105
106 <target name="t6" depends="t5">
107  <echo msg="replacing 'develop' text with 'intermediate'" />
108  <echo msg="replacing 'DEVELOPMENT' text with 'INTERMEDIATE'" />
109
110   <reflexive>
111    <fileset dir="../intermediate/${websiteName}/shared/incs/">
112       <include name="**/domain.conf.php*" />
113     </fileset>
114
115     <filterchain>
116       <replaceregexp>
117         <regexp pattern="develop\/${websiteName}"
118              replace="intermediate/${websiteName}" />
119       <regexp pattern="DEVELOPMENT" replace="INTERMEDIATE" />
120       </replaceregexp>
121     </filterchain>
122
123 </reflexive>
124
```

```
125 </target>
126
127
128 <!-- initialise live directory
129      with domain.conf.php files -->
130
131 <!-- copy '${websiteName}/shared/incs/domain.conf.php
132      files to live/${websiteName}/shared/incs/
133      directory -->
134
135 <target name="t7" depends="t6">
136   <echo msg="Copying domain.conf.php files to new
137 'live/${websiteName}/shared/incs/' directory..." />
138
139  <copy todir="../live/${websiteName}/shared/incs/">
140     <!-- Fileset for all files to be copied -->
141     <fileset dir="./${websiteName}/shared/incs/">
142       <include name="**/domain.conf.php*" />
143       <!-- exclude name="**/about.*" /-->
144     </fileset>
145   </copy>
146 </target>
147
148
149 <!-- change text strings in domain.conf.php files
150      in newly created ${websiteName} directory,
151      to work with this new website -->
152
153 <target name="t8" depends="t7">
154  <echo msg="replacing 'develop' text with 'live'" />
155  <echo msg="replacing 'DEVELOPMENT' text with 'LIVE'" />
156
157   <reflexive>
158    <fileset dir="../live/${websiteName}/shared/incs/">
159       <include name="**/domain.conf.php*" />
160     </fileset>
161
162     <filterchain>
163       <replaceregexp>
164         <regexp pattern="develop\/${websiteName}"
165                 replace="live/${websiteName}" />
```

```
166       <regexp pattern="DEVELOPMENT" replace="LIVE" />
167       </replaceregexp>
168     </filterchain>
169
170   </reflexive>
171
172 </target>
173
174
175 <target name="finished" depends="t8">
176   <echo msg="New kazaam website '${websiteName}' created OK!" />
177
178   <echo msg="intermediate directory for new kazaam website
179 '${websiteName}' created OK!" />
180
181   <echo msg="live directory for new kazaam website
182 '${websiteName}' created OK!" />
183 </target>
184
185 </project>
```

7.2.2 How create-new-website.xml works

Here we look at some important lines of code in this script.

line 01 declares the document type as xml version one.

lines 03 - 18 are comments about the script's usage.

lines 20 - 21 the start of the PHING script.

lines 23 - 28 target ***t1*** is the first target in the script. It asks for the name of the new website you want to create. This value is stored in the PHING variable *${websiteName}*.

lines 33 - 49 target ***t2*** copies the *kazaam-sitetpl* website template directory in the *develop* code branch, to the name of the new website directory specified in *${websiteName}*. New websites are only created in the *develop* directory.

lines 55 - 73 target ***t3*** performs a regular expression find and replace on all files in the new website. Any strings found that are equal to 'kazaam-sitetpl' will be replaced with whatever value is in *${websiteName}.*

lines 76 - 78 target ***t4*** outputs the message: "New kazaam website *${websiteName}* created OK!"

lines 88 - 99 target ***t5*** copies the domain.conf.php file(s) from the newly created website (in /develop/*${websiteName}*/shared/incs/*,)* to /intermediate/*${websiteName}*/shared/incs/ . It also creates the *${websiteName}*/shared/incs/ directories in the *intermediate* directory first, before copying the files.

lines 106 - 125 target ***t6*** performs a regular expression find-and-replace on the domain.conf.php files copied to /intermediate/*${websiteName}*/shared/incs/ . This changes the **$WEBSITE_ROOT** path variable to point to the *intermediate* code branch. It also changes the *$sitename* variable from 'DEVELOPMENT' to 'INTERMEDIATE'. This string is shown as part of the <**title**> tag for each page in a website, and helps identify what code branch you are in.

lines 135 - 146 target ***t7*** copies the domain.conf.php file(s) from the newly created website (in /develop/*${websiteName}*/shared/incs/*,)* to /live/*${websiteName}*/shared/incs/ . It also creates the *${websiteName}*/shared/incs/ directories in the *live* directory first, before copying the files.

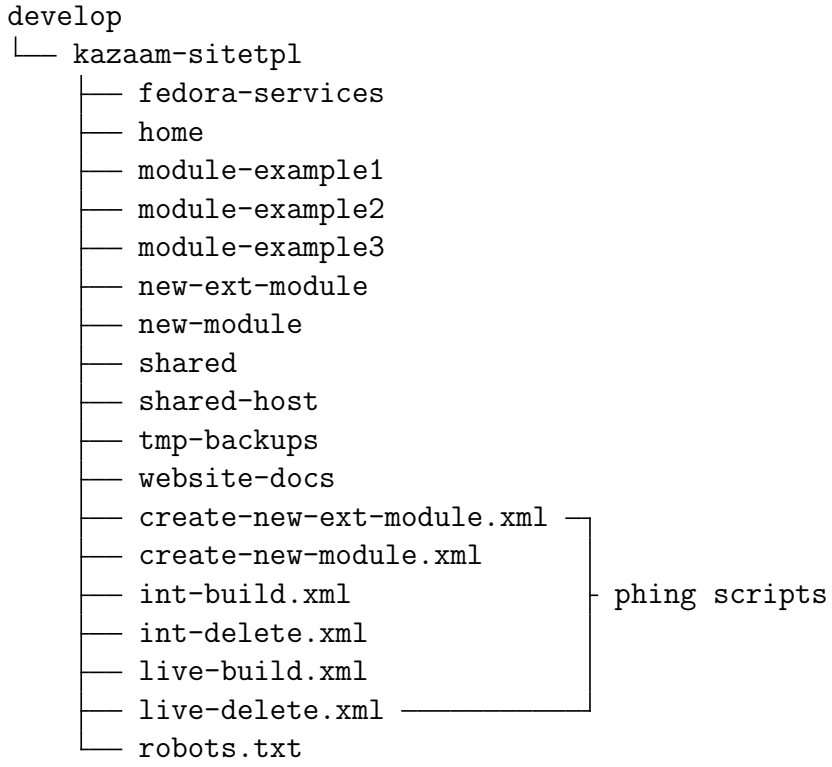
lines 153 - 172 target ***t8*** performs a regular expression find-and-replace on the domain.conf.php files copied to /live/*${websiteName}*/shared/incs/ . This changes the **$WEBSITE_ROOT** path variable to point to the *live* code branch. It also changes the *$sitename* variable from 'DEVELOPMENT' to 'LIVE'. This string is shown as part of the <**title**> tag for each page in a website, and helps identify what code branch you are in.

lines 175 - 183 target ***finished*** outputs a message to say the website has been created OK, and that the *intermediate* and *live* code branch directories have been created and populated with their respective copies of the domain.conf.php files for this particular website.

7.3 Other PHING Scripts

The other PHING scripts are inside a ***website's*** directory, in the *develop* code branch. Each website in the *develop* code branch has it's own set of PHING scripts, customised to work with that particular website. These customisations to a website's PHING scripts are done when the website is first created using **create-new-website.xml** .

Using the *kazaam-sitetpl* website template as an example, here is the directory structure showing the PHING scripts it contains:



7.4 create-new-ext-module.xml

This script is used to create a new module for a website. This module's pages are designed to make it easy to convert an existing website's content, to work with the Kazaam framework. Normally you would just use *create-new-module.xml* when creating a new module with new content.

7.4.1 Structure of create-new-ext-module.xml

Here we take a look at the create-new-ext-module.xml script.

create-new-ext-module.xml

```
01 <?xml version="1.0" ?>
02
03 <!--
04
05 Phing script to create a new website module,
06 with external body content, from kazaam's
07 'new-ext-module' (module template) directory.
08
09 to run it use:
10
11 # phing -f create-new-ext-module.xml
12
13 in the CWD of the above script.
14
15 Tell phing the name of the new module you want to create,
16 and phing will do it for you.
17
18 -->
19
20 <project name="Create new external-content website module"
21          default="meta-menu-created" basedir=".">
22
23 <target name="get-module-name">
24   <input propertyname="moduleName">
25 Please enter new module name:</input>
26  <echo>Creating new kazaam '${moduleName}' module</echo>
27 </target>
28
29
30 <!-- copy 'new-ext-module' template
31      to the ${moduleName} directory -->
32 <target name="copy-module-files"
33         depends="get-module-name">
34
35   <echo msg="Copying files to new '${moduleName}'
36 module directory..." />
37
38   <copy todir="${moduleName}">
39     <!-- Fileset for all files to be copied -->
```

```
40      <fileset dir="new-ext-module">
41        <include name="**/*" />
42        <!-- exclude name="**/about.*" /-->
43      </fileset>
44    </copy>
45
46    <!-- ================================= -->
47
48    <!-- rename new-ext-module.php
49         to ${moduleName}.php -->
50    <property name="path1"
51              value="${moduleName}/anyuser" />
52
53    <move file="${path1}/new-ext-module.php"
54          tofile="${path1}/${moduleName}.php"
55          overwrite="true" />
56
57    <move file="${path1}/new-ext-module.php.bak"
58          tofile="${path1}/${moduleName}.php.bak"
59          overwrite="true" />
60
61    <!-- rename backup directory filenames -->
62    <property name="path2"
63           value="${moduleName}/anyuser/tmp-backups" />
64
65    <move file="${path2}/new-ext-module.php"
66          tofile="${path2}/${moduleName}.php"
67          overwrite="true" />
68
69    <move file="${path2}/new-ext-module.php.bak"
70          tofile="${path2}/${moduleName}.php.bak"
71          overwrite="true" />
72
73    <!-- ================================= -->
74
75    <!-- rename new-ext-module.conf.php
76         to ${moduleName}.conf.php -->
77    <property name="path3"
78              value="${moduleName}/local/incs" />
79
80    <move file="${path3}/new-ext-module.conf.php"
```

```
81          tofile="${path3}/${moduleName}.conf.php"
82          overwrite="true" />
83
84    <move file="${path3}/new-ext-module.conf.php.bak"
85          tofile="${path3}/${moduleName}.conf.php.bak"
86          overwrite="true" />
87
88    <!-- rename backup directory filenames -->
89    <property name="path4"
90            value="${moduleName}/local/incs/tmp-backups" />
91
92    <move file="${path4}/new-ext-module.conf.php"
93          tofile="${path4}/${moduleName}.conf.php"
94          overwrite="true" />
95
96    <move file="${path4}/new-ext-module.conf.php.bak"
97          tofile="${path4}/${moduleName}.conf.php.bak"
98          overwrite="true" />
99
100    <!-- ================================= -->
101
102    <!-- rename new-ext-module-menu.php
103          to ${moduleName}-menu.php -->
104    <property name="path5"
105             value="${moduleName}/local/incs/mod" />
106
107    <move file="${path5}/new-ext-module-menu.php"
108          tofile="${path5}/${moduleName}-menu.php"
109          overwrite="true" />
110
111    <move file="${path5}/new-ext-module-menu.php.bak"
112          tofile="${path5}/${moduleName}-menu.php.bak"
113          overwrite="true" />
114
115    <!-- rename backup directory filenames -->
116    <property name="path6"
117            value="${moduleName}/local/incs/mod/tmp-backups" />
118
119    <move file="${path6}/new-ext-module-menu.php"
120          tofile="${path6}/${moduleName}-menu.php"
121          overwrite="true" />
```

```
122
123  <move file="${path6}/new-ext-module-menu.php.bak"
124      tofile="${path6}/${moduleName}-menu.php.bak"
125        overwrite="true" />
126
127   <!-- ================================= -->
128
129   <!-- rename new-ext-module-right-menu.php
130        to ${moduleName}-right-menu.php -->
131   <property name="path7"
132           value="${moduleName}/local/incs/mod" />
133
134  <move file="${path7}/new-ext-module-right-menu.php"
135      tofile="${path7}/${moduleName}-right-menu.php"
136        overwrite="true" />
137
138  <move file="${path7}/new-ext-module-right-menu.php.bak"
139      tofile="${path7}/${moduleName}-right-menu.php.bak"
140        overwrite="true" />
141
142   <!-- rename backup directory filenames -->
143   <property name="path8"
144          value="${moduleName}/local/incs/mod/tmp-backups" />
145
146  <move file="${path8}/new-ext-module-right-menu.php"
147      tofile="${path8}/${moduleName}-right-menu.php"
148        overwrite="true" />
149
150  <move file="${path8}/new-ext-module-right-menu.php.bak"
151      tofile="${path8}/${moduleName}-right-menu.php.bak"
152        overwrite="true" />
153
154   <!-- ================================= -->
155
156   <!-- rename new-ext-module.cont.php
157        to ${moduleName}.cont.php -->
158   <property name="path9"
159             value="${moduleName}/anyuser" />
160
161   <move file="${path9}/new-ext-module.cont.php"
162         tofile="${path9}/${moduleName}.cont.php"
```

```
163         overwrite="true" />
164
165   <move file="${path9}/new-ext-module.cont.php.bak"
166       tofile="${path9}/${moduleName}.cont.php.bak"
167         overwrite="true" />
168
169    <!-- rename backup directory filenames -->
170    <property name="path10"
171           value="${moduleName}/anyuser/tmp-backups" />
172
173    <move file="${path10}/new-ext-module.cont.php"
174          tofile="${path10}/${moduleName}.cont.php"
175          overwrite="true" />
176
177   <move file="${path10}/new-ext-module.cont.php.bak"
178       tofile="${path10}/${moduleName}.cont.php.bak"
179         overwrite="true" />
180
181    <!-- ================================= -->
182
183 </target>
184
185
186 <!-- remove the new-ext-module* template files
187     from the newly created ${moduleName} module -->
188 <target name="clean-up" depends="copy-module-files">
189   <echo msg="Cleaning up unnecessary template files" />
190
191    <delete>
192      <!-- Fileset for all files to be deleted -->
193      <fileset dir="${moduleName}">
194        <include name="**/new-ext-module*" />
195       <include name="**/tmp-backups/new-ext-module*" />
196      </fileset>
197    </delete>
198
199 </target>
200
201
202 <!-- change text strings in all files in newly created
203      ${moduleName} directory to work with this
```

```
204      new module directory -->
205 <target name="replace-module-text" depends="clean-up">
206
207   <echo msg="replacing 'new-ext-module' text
208 with '${moduleName}'" />
209
210   <echo msg="replacing 'New Ext Module' text
211 with '${moduleName}'" />
212
213   <reflexive>
214     <fileset dir="${moduleName}">
215       <include name="**/*" />
216     </fileset>
217
218     <filterchain>
219       <replaceregexp>
220         <regexp pattern="new-ext-module"
221                 replace="${moduleName}" />
222
223         <regexp pattern="New Ext Module"
224                 replace="${moduleName}" />
225       </replaceregexp>
226     </filterchain>
227
228   </reflexive>
229
230 </target>
231
232
233 <target name="module-created"
234         depends="replace-module-text">
235  <echo msg="New kazaam module '${moduleName}' created OK!" />
236 </target>
237
238
239 <!-- create the meta-menu for the new module -->
240
241 <target name="create-ext-meta-menu"
242         depends="module-created">
243   <echo></echo>
244   <echo>Creating meta-menu for new kazaam
```

```
245 module '${moduleName}'</echo>
246 </target>
247
248
249 <target name="copy-meta-menu-files"
250         depends="create-ext-meta-menu">
251
252  <echo msg="Copying 'new-ext-module.php' (meta-menu
253 template files) to '${moduleName}.php'" />
254
255  <!-- copy './shared/meta-menus/new-ext-module.php'
256      meta-menu template file to '${moduleName}.php' -->
257   <property name="path11"
258             value="./shared/meta-menus" />
259
260   <copy file="${path11}/new-ext-module.php"
261         tofile="${path11}/${moduleName}.php"
262         overwrite="true"/>
263
264   <copy file="${path11}/new-ext-module.php.bak"
265         tofile="${path11}/${moduleName}.php.bak"
266         overwrite="true"/>
267
268   <!-- copy backup directory files -->
269   <property name="path12"
270           value="./shared/meta-menus/tmp-backups" />
271
272   <copy file="${path12}/new-ext-module.php"
273         tofile="${path12}/${moduleName}.php"
274         overwrite="true"/>
275
276   <copy file="${path12}/new-ext-module.php.bak"
277         tofile="${path12}/${moduleName}.php.bak"
278         overwrite="true"/>
279
280 </target>
281
282
283 <!-- tweak text strings in newly created
284      ${moduleName}.php files -->
285 <target name="replace-meta-menu-text"
```

```
286         depends="copy-meta-menu-files">
287
288   <echo msg="replacing 'new-ext-module' text
289 with '${moduleName}'" />
290
291   <echo msg="replacing 'New Ext Module' text
292 with '${moduleName}'" />
293
294   <reflexive>
295     <fileset dir="./shared/meta-menus/">
296       <include name="${moduleName}.*" />
297      <include name="**/tmp-backups/${moduleName}.*" />
298     </fileset>
299
300     <filterchain>
301       <replaceregexp>
302         <regexp pattern="new-ext-module"
303                 replace="${moduleName}" />
304
305         <regexp pattern="New Ext Module"
306                 replace="${moduleName}" />
307       </replaceregexp>
308     </filterchain>
309
310   </reflexive>
311
312 </target>
313
314
315 <target name="meta-menu-created"
316         depends="replace-meta-menu-text">
317   <echo msg="meta-menu for new kazaam module
318 '${moduleName}' created OK!" />
319 </target>
320
321 </project>
```

7.4.2 How create-new-ext-module.xml works

Here we look at some important lines of code in this script.

line 01 declares the document type as xml version one.

lines 03 - 18 are comments about the script's usage.

lines 20 - 21 This is the start of the PHING script.

lines 23 - 27 target ***get-module-name*** is the first target in the script. It asks for the name of the new module you want to create. This value is stored in the PHING variable *${moduleName}*.

lines 32 - 183 target ***copy-module-files*** copies the *new-ext-module* module template directory to the name of the new module directory specified in *${moduleName}* . New modules are only created in websites under the *develop* directory. Lines 48 - 180 change the names of certain files in the module to match the new module just created.

lines 188 - 199 target ***clean-up*** removes any unnecessary module template files from the newly created module.

lines 205 - 230 target ***replace-module-text*** performs a regular expression search-and-replace on the contents of all the files in the newly created module. It replaces any occurence of 'new-ext-module' or 'New Ext Module' with the value in *${moduleName}* .

lines 233 - 236 target ***module-created*** outputs a message saying the new module has been created OK.

lines 241 - 246 target ***create-ext-meta-menu*** outputs a message saying a meta-menu is being created for the new module.

lines 249 - 280 target ***copy-meta-menu-files*** creates a *meta-menu* for this new module from the ***website-name****/shared/meta-menus/*new-ext-module.php meta-menu template. *(A meta-menu* can be added to either the *homepage-menu.php,* or the *right-menu.php*, both which live in: ***website-name****/shared/incs/ .)*

lines 285 - 312 target ***replace-meta-menu-text*** performs a regular expression search-and-replace on the contents of the newly created meta-menu files for this new module. It replaces any occurence of 'new-ext-module' or 'New Ext Module' with the value in *${moduleName}* .

lines 315 - 319 target ***meta-menu-created*** outputs a message saying the meta-menu for the new module has been created OK.

7.5 create-new-module.xml

Each website has it's own customised copy of *create-new-module.xml* . This script will create a new module for a given website.

7.5.1 Structure of create-new-module.xml

Here we take a look at the create-new-module.xml script.

create-new-module.xml

```
01 <?xml version="1.0" ?>
02
03 <!--
04
05 Phing script to create a new website module,
06 from kazaam's 'new-module' (module template) directory.
07
08 to run it use:
09 # phing -f create-new-module.xml
10 in the CWD of the above script.
11
12 Tell phing the name of the new module you want to create,
13 and phing will do it for you.
14
15 -->
16
17 <project name="Create new website module"
18 default="meta-menu-created" basedir=".">
19
20 <target name="get-module-name">
21   <echo></echo>
22   <input propertyname="moduleName">
23 Please enter new module name:</input>
24  <echo>Creating new kazaam '${moduleName}' module</echo>
25 </target>
26
27 <!-- copy 'new-module' template to the ${moduleName}
```

```
28 directory -->
29 <target name="copy-module-files"
30         depends="get-module-name">
31
32 <echo msg="Copying files to '${moduleName}' module directory..." />
33
34 <copy todir="${moduleName}">
35   <!-- Fileset for all files to be copied -->
36   <fileset dir="new-module">
37     <include name="**/*" />
38     <!-- exclude name="**/about.*" /-->
39   </fileset>
40 </copy>
41
42 <!-- ================================== -->
43
44 <!-- rename new-module.php to ${moduleName}.php -->
45 <property name="path1"
46           value="${moduleName}/anyuser" />
47
48 <move file="${path1}/new-module.php"
49       tofile="${path1}/${moduleName}.php"
50       overwrite="true" />
51
52 <move file="${path1}/new-module.php.bak"
53       tofile="${path1}/${moduleName}.php.bak"
54       overwrite="true" />
55
56 <!-- rename backup directory filenames -->
57 <property name="path2"
58         value="${moduleName}/anyuser/tmp-backups" />
59
60 <move file="${path2}/new-module.php"
61       tofile="${path2}/${moduleName}.php"
62       overwrite="true" />
63
64 <move file="${path2}/new-module.php.bak"
65       tofile="${path2}/${moduleName}.php.bak"
66       overwrite="true" />
67
68 <!-- ================================== -->
```

```
69
70 <!-- rename new-module.conf.php to ${moduleName}.conf.php -->
71 <property name="path3"
72           value="${moduleName}/local/incs" />
73
74 <move file="${path3}/new-module.conf.php"
75       tofile="${path3}/${moduleName}.conf.php"
76       overwrite="true" />
77
78 <move file="${path3}/new-module.conf.php.bak"
79       tofile="${path3}/${moduleName}.conf.php.bak"
80       overwrite="true" />
81
82 <!-- rename backup directory filenames -->
83 <property name="path4"
84         value="${moduleName}/local/incs/tmp-backups" />
85
86 <move file="${path4}/new-module.conf.php"
87       tofile="${path4}/${moduleName}.conf.php"
88       overwrite="true" />
89
90 <move file="${path4}/new-module.conf.php.bak"
91       tofile="${path4}/${moduleName}.conf.php.bak"
92       overwrite="true" />
93
94 <!-- ================================= -->
95
96 <!-- rename new-module-menu.php to ${moduleName}-menu.php -->
97 <property name="path5"
98           value="${moduleName}/local/incs/mod" />
99
100 <move file="${path5}/new-module-menu.php"
101       tofile="${path5}/${moduleName}-menu.php"
102       overwrite="true" />
103
104 <move file="${path5}/new-module-menu.php.bak"
105       tofile="${path5}/${moduleName}-menu.php.bak"
106       overwrite="true" />
107
108 <!-- rename backup directory filenames -->
109 <property name="path6"
```

```
110 value="${moduleName}/local/incs/mod/tmp-backups" />
111
112 <move file="${path6}/new-module-menu.php"
113       tofile="${path6}/${moduleName}-menu.php"
114       overwrite="true" />
115
116 <move file="${path6}/new-module-menu.php.bak"
117      tofile="${path6}/${moduleName}-menu.php.bak"
118       overwrite="true" />
119
120 <!-- ================================= -->
121
122 <!-- rename new-module-right-menu.php
123      to ${moduleName}-right-menu.php -->
124 <property name="path7"
125           value="${moduleName}/local/incs/mod" />
126
127 <move file="${path7}/new-module-right-menu.php"
128     tofile="${path7}/${moduleName}-right-menu.php"
129       overwrite="true" />
130
131 <move file="${path7}/new-module-right-menu.php.bak"
132     tofile="${path7}/${moduleName}-right-menu.php.bak"
133       overwrite="true" />
134
135 <!-- rename backup directory filenames -->
136 <property name="path8"
137     value="${moduleName}/local/incs/mod/tmp-backups" />
138
139 <move file="${path8}/new-module-right-menu.php"
140     tofile="${path8}/${moduleName}-right-menu.php"
141       overwrite="true" />
142
143 <move file="${path8}/new-module-right-menu.php.bak"
144     tofile="${path8}/${moduleName}-right-menu.php.bak"
145       overwrite="true" />
146
147 </target>
148
149 <!-- remove the new-module* template files
150     from the newly created ${moduleName} module -->
```

```
151 <target name="clean-up" depends="copy-module-files">
152
153   <echo msg="Cleaning up unnecessary template files" />
154    <delete>
155
156      <!-- Fileset for all files to be deleted -->
157      <fileset dir="${moduleName}">
158        <include name="**/new-module*" />
159       <include name="**/tmp-backups/new-module*" />
160      </fileset>
161
162    </delete>
163 </target>
164
165
166 <!-- change text strings in all files in newly created
167       ${moduleName} directory to work with this
168       new module directory -->
169 <target name="replace-module-text" depends="clean-up">
170   <echo msg="replacing 'new-module' text with '${moduleName}'" />
171   <echo msg="replacing 'New Module' text with '${moduleName}'" />
172
173 <reflexive>
174    <fileset dir="${moduleName}">
175      <include name="**/*" />
176    </fileset>
177
178    <filterchain>
179      <replaceregexp>
180       <regexp pattern="new-module" replace="${moduleName}" />
181       <regexp pattern="New Module" replace="${moduleName}" />
182      </replaceregexp>
183    </filterchain>
184 </reflexive>
185
186 </target>
187
188 <target name="module-created" depends="replace-module-text">
189   <echo msg="New kazaam module '${moduleName}' created OK!" />
190 </target>
191
```

```
192
193 <!-- create the meta-menu for the new module -->
194
195 <target name="create-meta-menu" depends="module-created">
196   <echo></echo>
197   <echo>Creating meta-menu for new kazaam module
198    '${moduleName}'</echo>
199 </target>
200
201
202 <target name="copy-meta-menu-files"
203         depends="create-meta-menu">
204
205  <echo msg="Copying 'new-module.php' (meta-menu template files)
206    to '${moduleName}.php'" />
207
208 <!-- copy './shared/meta-menus/new-module.php' meta-menu template
209 file to '${moduleName}.php' -->
210 <property name="path9"
211           value="./shared/meta-menus" />
212
213 <copy file="${path9}/new-module.php"
214       tofile="${path9}/${moduleName}.php"
215       overwrite="true"/>
216
217 <copy file="${path9}/new-module.php.bak"
218       tofile="${path9}/${moduleName}.php.bak"
219       overwrite="true"/>
220
221 <!-- copy backup directory files -->
222 <property name="path10"
223         value="./shared/meta-menus/tmp-backups" />
224
225 <copy file="${path10}/new-module.php"
226       tofile="${path10}/${moduleName}.php"
227       overwrite="true"/>
228
229 <copy file="${path10}/new-module.php.bak"
230       tofile="${path10}/${moduleName}.php.bak"
231      overwrite="true"/>
232
```

```
233 </target>
234
235
236 <!-- tweak text strings in newly created
237      ${moduleName}.php files -->
238 <target name="replace-meta-menu-text"
239         depends="copy-meta-menu-files">
240
241  <echo msg="replacing 'new-module' text with '${moduleName}'" />
242  <echo msg="replacing 'New Module' text with '${moduleName}'" />
243
244   <reflexive>
245     <fileset dir="./shared/meta-menus/">
246       <include name="${moduleName}.*" />
247      <include name="**/tmp-backups/${moduleName}.*" />
248     </fileset>
249
250     <filterchain>
251       <replaceregexp>
252       <regexp pattern="new-module" replace="${moduleName}" />
253       <regexp pattern="New Module" replace="${moduleName}" />
254       </replaceregexp>
255     </filterchain>
256
257   </reflexive>
258
259 </target>
260
261
262 <target name="meta-menu-created"
263         depends="replace-meta-menu-text">
264   <echo msg="meta-menu for new kazaam module
265    '${moduleName}' created OK!" />
266 </target>
267
268 </project>
```

7.5.2 How create-new-module.xml works

Here we look at some important lines of code in this script.

line 01 declares the document type as xml version one.

lines 03 - 15 are comments about the script's usage.

lines 17 - 18 This is the start of the PHING script.

lines 20 - 25 target ***get-module-name*** is the first target in the script. It asks for the name of the new module you want to create. This value is stored in the PHING variable *${moduleName}.*

lines 29 - 147 target ***copy-module-files*** copies the *new-module* module template directory to the name of the new module directory specified in *${moduleName}* . New modules are only created in websites under the *develop* directory. Lines 44 - 145 change the names of certain files in the module to match the new module just created.

lines 151 - 163 target ***clean-up*** removes any unnecessary module template files from the newly created module.

lines 169 - 186 target ***replace-module-text*** performs a regular expression search-and-replace on the contents of all the files in the newly created module. It replaces any occurence of 'new-module' or 'New Module' with the value in *${moduleName}* .

lines 188 - 190 target ***module-created*** outputs a message saying the new module has been created OK.

lines 195 - 199 target ***create-meta-menu*** outputs a message saying a meta-menu is being created for the new module.

lines 202 - 233 target ***copy-meta-menu-files*** creates a *meta-menu* for this new module from the
website-name*/shared/meta-menus/*new-module.php meta-menu template. *(A meta-menu* can be added to either the *homepage-menu.php,* or the *right-menu.php*, both which live in:
website-name*/shared/incs/ .)*

lines 238 - 259 target ***replace-meta-menu-text*** performs a regular expression search-and-replace on the contents of the newly created meta-menu files for this new module. It replaces any occurence of 'new-module' or 'New Module' with the value in *${moduleName}* .

lines 262 - 266 target ***meta-menu-created*** outputs a message saying the meta-menu for the new module has been created OK.

7.6 int-build.xml

This script is used to copy a complete website from the *develop* code branch, to the *intermediate* code branch.

7.6.1 Structure of int-build.xml

Here we take a look at the int-build.xml script.

---

int-build.xml

```
01 <?xml version="1.0" ?>
02
03 <project name="kazaam-sitetpl"
04          default="finished" basedir=".">
05
06 <target name="t1">
07   <echo>Copying kazaam-sitetpl development files
08           to intermediate directory...
09   </echo>
10
11   <echo>Except the following files:
12 /shared/incs/domain.* files...
13 /shared-host/incs/*.* symbolic link files...
14   </echo>
15
16   <echo>Please wait. This could take some time
17   </echo>
18
19   <copy todir="../../intermediate/kazaam-sitetpl"
20         overwrite="true">
21
22     <!-- Fileset for all files to be copied -->
23     <fileset dir=".">
24       <include name="**" />
25
26       <!-- don't overwrite domain.* files -->
27      <!-- each website's code branch has it's own domain.* files
28            with specific settings -->
29       <exclude name="**/domain.*" />
30
```

```
31       <!-- do not copy these directories for now -->
32       <!-- shared-host contains symbolic link files
33          so we don't want to overwrite them with REAL files -->
34       <exclude name="**/shared-host/**" />
35
36      <!-- also do not copy these example files for now -->
37       <exclude name="**/file1.*" />
38       <exclude name="**/file2.*" />
39
40     </fileset>
41   </copy>
42 </target>
43
44 <target name="finished" depends="t1">
45   <echo msg="All development files copied
46 to intermediate directory OK!" />
47 </target>
48
49 </project>
```

7.6.2 How int-build.xml works

Here we look at some important lines of code in this script.

line 01 declares the document type as xml version one.

lines 03 - 04 This is the start of the PHING script.

line 06 is the start of target ***t1.***

lines 07 - 09 output an information message about the script.

lines 11 - 14 output a message saying which files are NOT being copied.

lines 16 - 17 output an information message about the script.

line 19 sets the directory destination where the files will be copied to.

line 20 overwrites any existing directories or files while copying.

line 23 sets the source directory to copy from.

line 24 copies all directories & files from the source directory.

line 29 does not copy any files matching this pattern.

line 34 does not copy anything from the directory matching this pattern.

lines 37 - 38 example pattern to exclude any files (or directories) for this website you do not want copied to the *intermediate* code branch.

line 42 is the end of target ***t1.***

lines 44 - 47 target ***finished*** outputs a message saying the development code for this website has been copied to the *intermediate* code branch ok.

7.7 int-delete.xml

Use this script to clean up the contents of the *intermediate* directory for a particular website. By default it will delete everything in the *intermediate* code branch for this website, apart from the *domain.** files, and anything in the *shared-host* sub-directory, which should be symbolic links.

7.7.1 Structure of int-delete.xml

Here we take a look at the int-delete.xml script.

---

int-delete.xml

```
01 <?xml version="1.0" ?>
02
03 <project name="kazaam-sitetpl"
04          default="finished" basedir=".">
05
06 <target name="t1">
07   <echo>Deleting all files in kazaam-sitetpl
08           intermediate directory...
09   </echo>
10
11   <echo>Except the following files:
12 /shared/incs/domain.* files...
13 /shared-host/incs/*.* symbolic link files...
14   </echo>
```

```
15
16   <echo>Please wait. This could take some time
17   </echo>
18
19   <delete includeemptydirs="true">
20     <!-- Fileset for all files to be deleted -->
21    <fileset dir="../../intermediate/kazaam-sitetpl">
22       <include name="**" />
23
24       <!-- do not delete domain.* files -->
25      <!-- each website's code branch has it's own domain.* files
26            with specific settings -->
27       <exclude name="**/domain.*" />
28       <exclude name="**/tmp-backups/domain.*" />
29
30       <!-- also do not delete these directories -->
31      <!-- this directory contains symbolic links that are needed -->
32       <exclude name="**/shared-host/**" />
33
34      <!-- also do not delete these example files for now -->
35       <exclude name="**/file1.*" />
36       <exclude name="**/file2.*" />
37     </fileset>
38   </delete>
39 </target>
40
41 <target name="finished" depends="t1">
42   <echo msg="All files in kazaam-sitetpl
43 intermediate directory deleted OK!" />
44   </target>
45
46 </project>
```

7.7.2 How int-delete.xml works

Here we look at some important lines of code in this script.

line 01 declares the document type as xml version one.

lines 03 - 04 This is the start of the PHING script.

line 06 is the start of target ***t1.***

lines 07 - 09 output an information message about the script.

lines 11 - 14 output a message saying which files are NOT being deleted.

lines 16 - 17 output an information message about the script.

line 19 is the start of the <delete> task. It also removes empty directories from the target directory.

line 21 sets the target directory to delete contents from.

line 22 deletes all directories and files from the target directory.

lines 27 - 28 does not delete any files matching this pattern from the target directory.

line 32 does not delete anything from the sub-directory matching this pattern in the target directory.

lines 35 - 36 example pattern to exclude any files (or directories) you do not want deleted from the *intermediate* code branch for this website.

line 39 is the end of target ***t1.***

lines 41 - 44 target ***finished*** outputs a message saying all files in the *intermediate* code branch for this website have been deleted ok.

7.8 live-build.xml

This script is used to copy a complete website from the *intermediate* code branch, to the *live* code branch.

7.8.1 Structure of live-build.xml

Here we take a look at the live-build.xml script.

---

live-build.xml

```
01 <?xml version="1.0" ?>
02
```

```
03 <project name="kazaam-sitetpl"
04          default="finished" basedir=".">
05
06 <target name="t1">
07   <echo>Copying kazaam-sitetpl intermediate files
08           to live directory...
09   </echo>
10
11   <echo>Except the following files:
12 /shared/incs/domain.* files...
13 /shared-host/incs/*.* symbolic link files...
14   </echo>
15
16   <echo>Please wait. This could take some time
17   </echo>
18
19   <copy todir="../../live/kazaam-sitetpl"
20         overwrite="true">
21
22     <!-- Fileset for all files to be copied -->
23     <fileset dir=".">
24       <include name="**" />
25
26       <!-- don't overwrite domain.* files -->
27      <!-- each website's code branch has it's own domain.* files
28            with specific settings -->
29       <exclude name="**/domain.*" />
30
31      <!-- do not copy these directories for now -->
32       <!-- shared-host contains symbolic link files
33          so we don't want to overwrite them with REAL files -->
34       <exclude name="**/shared-host/**" />
35
36       <exclude name="**/new-ext-module/**" />
37       <exclude name="**/new-module/**" />
38       <exclude name="**/website-docs/" />
39       <exclude name="**/tmp-backups/" />
40
41       <!-- do not copy any of these files
42            to the live directory -->
43       <exclude name="**/*.BASH" />
```

```
44        <exclude name="**/*.NOTES" />
45        <exclude name="**/robots.*" />
46        <exclude name="**/*.bak" />
47        <exclude name="**/*.xml" />
48
49      </fileset>
50    </copy>
51 </target>
52
53 <target name="finished" depends="t1">
54    <echo msg="All intermediate files copied
55 to live directory OK!" />
56 </target>
57
58 </project>
```

7.8.2 How live-build.xml works

Here we look at some important lines of code in this script.

line 01 declares the document type as xml version one.

lines 03 - 04 This is the start of the PHING script.

line 06 is the start of target ***t1.***

lines 07 - 09 output an information message about the script.

lines 11 - 14 output a message saying which files are NOT being copied.

lines 16 - 17 output an information message about the script.

line 19 sets the directory destination where the files will be copied to.

line 20 overwrites any existing directories or files while copying.

line 23 sets the source directory to copy from.

line 24 copies all directories & files from the source directory.

line 29 does not copy any files matching this pattern.

line 34 does not copy anything from the directory matching this pattern.

lines 36 - 39 does not copy anything from the directories matching these patterns.

lines 43 - 47 does not copy any files matching these patterns.

line 51 is the end of target ***t1.***

lines 53 - 56 target ***finished*** outputs a message saying the *intermediate* code for this website has been copied to the *live* code branch ok.

7.9 live-delete.xml

Only use this script to clean up the contents of the *live* directory for a particular website ***Before*** the website has gone live.

WARNING: Running this script on a live operational website is NOT RECOMMENDED. It will delete all of the website, except for the domain.conf.php and robots.txt files, and symbolic link files in shared-host!

7.9.1 Structure of live-delete.xml

Here we take a look at the live-delete.xml script.

---

live-delete.xml

```
01 <?xml version="1.0" ?>
02
03 <project name="kazaam-sitetpl"
04          default="finished" basedir=".">
05
06 <target name="t1">
07   <echo>Deleting all files in kazaam-sitetpl
08           live directory...
09   </echo>
10
11   <echo>Except the following files:
12 /shared/incs/domain.* files...
13 /shared-host/incs/*.* symbolic link files...
```

```
14   </echo>
15
16   <echo>Please wait. This could take some time
17   </echo>
18
19   <delete includeemptydirs="true">
20     <!-- Fileset for all files to be deleted -->
21     <fileset dir="../../live/kazaam-sitetpl">
22       <include name="**" />
23
24       <!-- do not delete domain.* files -->
25      <!-- each website's code branch has it's own domain.* files
26            with specific settings -->
27       <exclude name="**/domain.*" />
28       <exclude name="**/tmp-backups/domain.*" />
29
30       <!-- also do not delete these directories -->
31      <!-- this directory contains symbolic links that are needed -->
32       <exclude name="**/shared-host/**" />
33
34       <!-- do not delete any of these files -->
35       <exclude name="**/robots.*" />
36       <exclude name="**/tmp-backups/robots.*" />
37
38     </fileset>
39   </delete>
40 </target>
41
42 <target name="finished" depends="t1">
43   <echo msg="All files in kazaam-sitetpl
44 live directory deleted OK!" />
45 </target>
46
47 </project>
```

7.9.2 How live-delete.xml works

Here we look at some important lines of code in this script.

line 01 declares the document type as xml version one.

lines 03 - 04 This is the start of the PHING script.

line 06 is the start of target ***t1.***

lines 07 - 09 output an information message about the script.

lines 11 - 14 output a message saying which files are NOT being deleted.

lines 16 - 17 output an information message about the script.

line 19 is the start of the <delete> task. It also removes empty directories from the target directory.

line 21 sets the target directory to delete contents from.

line 22 deletes all directories and files from the target directory.

lines 27 - 28 does not delete any files matching this pattern from the target directory.

line 32 does not delete anything from the sub-directory matching this pattern in the target directory.

lines 35 - 36 do not delete the robots.* files from the *live* code branch for this website.

line 40 is the end of target ***t1.***

lines 42 - 45 target ***finished*** outputs a message saying all files in the *live* code branch for this website have been deleted ok.

---

8 Kazaam Tutorials Part 1

8.1 Chapter Overview

Having looked at how the PHING build scripts work, this chapter now looks at using the PHING scripts to:

- create a new website
- create a new module for a website
- copy a website to the *intermediate* code branch
- copy a website to the *live* code branch

8.2 Creating a new website

cd to the root of the *develop* directory. It should look something like this:

```
develop
├── karsites
├── kazaam-sitetpl   <- website template
├── php-debuggers
├── raised-from-the-dead
└── create-new-website.xml
```

Now run the PHING script *create-new-website.xml* :

```
[root@karsites develop]# phing -f create-new-website.xml
Buildfile: /kazaam-mwf/develop/create-new-website.xml

Create new website > t1:
[echo]
Please enter name of new website: first-site
[echo] Creating new kazaam 'first-site' website

Create new website > t2:
[echo] Copying kazaam-sitetpl website template to new 'first-site'
       website directory...

[echo]
[echo] Please wait. This could take some time...
[echo]

[copy] Copying 861 files to /kazaam-mwf/develop/first-site

[copy] Copied 28 empty directories to
       /kazaam-mwf/develop/first-site

Create new website > t3:
[echo] replacing 'kazaam-sitetpl' text with 'first-site'

[reflexive] Applying reflexive processing to 861 files.
```

```
Create new website > t4:
[echo] New kazaam website 'first-site' created OK!

Create new website > t5:
[echo] Copying domain.conf.php files to new
     'intermediate/first-site/shared/incs/' directory...

[copy] Copying 4 files to
       /kazaam-mwf/intermediate/first-site/shared/incs

Create new website > t6:
[echo] replacing 'develop' text with 'intermediate'
[echo] replacing 'DEVELOPMENT' text with 'INTERMEDIATE'

[reflexive] Applying reflexive processing to 4 files.

Create new website > t7:
[echo] Copying domain.conf.php files to new
       'live/first-site/shared/incs/' directory...

[copy] Copying 4 files to
       /kazaam-mwf/live/first-site/shared/incs

Create new website > t8:

[echo] replacing 'develop' text with 'live'
[echo] replacing 'DEVELOPMENT' text with 'LIVE'

[reflexive] Applying reflexive processing to 4 files.

Create new website > finished:
[echo] New kazaam website 'first-site' created OK!

[echo] intermediate directory for new kazaam website 'first-site'
       created OK!

[echo] live directory for new kazaam website 'first-site'
       created OK!

BUILD FINISHED
```

```
Total time: 49.9379 seconds
```

If we do a directory listing, we should now have a new website in the *develop* directory, called **first-site**.

```
develop
├── first-site
├── karsites
├── kazaam-sitetpl   <- website template
├── php-debuggers
├── raised-from-the-dead
└── create-new-website.xml
```

We can check this out by opening a web browser, and going to:
http://localhost/kazaam-mwf/develop/first-site/home/anyuser/home.php
You should now see the homepage for the new site you have just created.

Before we go any further, we need to change the site-logo, so we know which website we are working on. Look under */kazaam-mwf/framework-docs/tutorials/website-images,* and you will find a file called *first-site-logo-1.jpg* . Copy this file into the: */kazaam-mwf/develop/first-site/shared/images/* directory. Now copy this file again to overwrite the *site-logo* image file in the same directory. If you reload the page, you should now see a different yellow-colored site banner, with the text 'first-site' on it. If the new *site-logo* does not appear, you may need to clear your browser cache and reload the page again, to force the browser to load the new *site-logo* banner.

8.3 Creating a new module

Now we have created a new website, we can extended it by adding new modules to it. **cd** to the ***first-site*** directory, in the *develop* directory. It should look like this:

```
first-site
├── fedora-services
├── home
├── module-example1
├── module-example2
├── module-example3
├── new-ext-module  <- module template
├── new-module      <- module template
├── shared
├── shared-host
├── tmp-backups
├── website-docs
├── create-new-ext-module.xml
├── create-new-module.xml
├── int-build.xml
├── int-delete.xml
├── live-build.xml
├── live-delete.xml
└── robots.txt
```

We want to create a new module named *'sports-cars'* to keep details of some high performance sports cars we are interested in. So lets run the PHING script *create-new-module.xml* :

```
[root@karsites first-site]# phing -f create-new-module.xml
Buildfile: /kazaam-mwf/develop/first-site/create-new-module.xml

Create new website module > get-module-name:
Please enter new module name: sports-cars
[echo] Creating new kazaam 'sports-cars' module

Create new website module > copy-module-files:
[echo] Copying files to 'sports-cars' module directory...

[copy] Copying 78 files to
        /kazaam-mwf/develop/first-site/sports-cars

[copy] Copied 4 empty directories to
```

```
        /kazaam-mwf/develop/first-site/sports-cars

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /anyuser

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /anyuser

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /anyuser/tmp-backups

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /anyuser/tmp-backups

[move] Copying 1 file to
       /kazaam-mwf/develop/first-site/sports-cars
       /local/incs

[move] Copying 1 file to
       /kazaam-mwf/develop/first-site/sports-cars
       /local/incs

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/tmp-backups

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/tmp-backups

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/mod

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/mod
```

```
[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/mod/tmp-backups

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/mod/tmp-backups

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/mod

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/mod

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/mod/tmp-backups

[move] Copying 1 file to
        /kazaam-mwf/develop/first-site/sports-cars
        /local/incs/mod/tmp-backups

Create new website module > clean-up:
[echo] Cleaning up unnecessary template files
[delete] Deleting 16 files from sports-cars

Create new website module > replace-module-text:
[echo] replacing 'new-module' text with 'sports-cars'
[echo] replacing 'New Module' text with 'sports-cars'
[reflexive] Applying reflexive processing to 78 files.

Create new website module > module-created:
[echo] New kazaam module 'sports-cars' created OK!

Create new website module > create-meta-menu:
[echo]
[echo] Creating meta-menu for new kazaam module 'sports-cars'
```

```
Create new website module > copy-meta-menu-files:
[echo] Copying 'new-module.php' (meta-menu template files)
        to 'sports-cars.php'

[copy] Copying 1 file to
        /kazaam-mwf/develop/first-site
        /shared/meta-menus

[copy] Copying 1 file to
        /kazaam-mwf/develop/first-site
        /shared/meta-menus

[copy] Copying 1 file to
        /kazaam-mwf/develop/first-site
        /shared/meta-menus/tmp-backups

[copy] Copying 1 file to
        /kazaam-mwf/develop/first-site
        /shared/meta-menus/tmp-backups

Create new website module > replace-meta-menu-text:
[echo] replacing 'new-module' text with 'sports-cars'
[echo] replacing 'New Module' text with 'sports-cars'
[reflexive] Applying reflexive processing to 4 files.

Create new website module > meta-menu-created:
[echo] meta-menu for new kazaam module 'sports-cars' created OK!

BUILD FINISHED
Total time: 9.2949 seconds
```

If we do a directory listing, we should now have a new module in the *first-site* website directory, called **sports-cars**:

```
first-site
├── fedora-services
├── home
├── module-example1
├── module-example2
├── module-example3
├── new-ext-module
├── new-module
├── shared
├── shared-host
├── sports-cars   <- new website module
├── tmp-backups
├── website-docs
├── create-new-ext-module.xml
├── create-new-module.xml
├── int-build.xml
├── int-delete.xml
├── live-build.xml
├── live-delete.xml
└── robots.txt
```

Now we have created a new module, we can access the module by 'plugging it into' the website in two different ways. We can:

1 add the *sports-cars* meta-menu to the website's homepage menu that appears in the left-hand colum of all 2 or 3 column pages;

2 or add a link to the *sports-cars* module's homepage, into an existing meta-menu of the website's homepage menu.

8.3.1 Adding the sports-cars meta-menu

To add the sports-cars meta-menu to the website's homepage menu open: */develop/first-site/shared/incs/homepage-menu.php* ,

and add the following line to the top of the file:

```
<?php require "$SHARED_META_MENUS/sports-cars.php"; ?>
```

Save the file and reload the *first-site* homepage. You should now see the *sports-cars* meta-menu added at the top of the other meta-menu items in the left-hand column homepage menu. This is OK for a module that only links

to a few pages. Clicking on any of the links in the *sports-cars* meta-menu will take you to that particular page template. You will then be presented with that page template, and the module's own custom left-hand menu, in place of the homepage menu you have just come from.

NB. A module's custom left-hand column menu is *not* the same as the module's meta-menu. You should prefer the module's custom left-hand menu over the meta-menu, as it is more flexible and extendable, and will go with the module when you copy a module from one website to another website.

As our collection of sports-cars could grow considerably, we will opt to use the module's own custom left-hand column menu, instead of the homepage menu, and we will put one link on the homepage menu to the *sports-cars.php* module's homepage.

Lets now remove the sports-cars meta-menu from the homepage-menu:

Restore */develop/first-site/shared/incs/homepage-menu.php*
to what it was like before adding the:

```
<?php require "$SHARED_META_MENUS/sports-cars.php"; ?>
```

line to it, and reload the first-site homepage. The sports-cars meta-menu should be gone now.

8.3.2 Adding a single link to the sports-cars.php homepage

Another way of adding the *sports-cars* module to the website is to link directly to the module's homepage, in:

/develop/first-site/sports-cars/anyuser/sports-cars.php .

This will only use one link on the homepage menu. This link can be added to an existing meta-menu in:
/develop/first-site/shared/meta-menus/ .

Lets add a link from the 'Module Example1' meta-menu (fourth one down from the top of the homepage menu), to the *sports-cars* module's homepage.

Open */develop/first-site/shared/meta-menus/module-example1.php*
and add the following lines of code as the first href link:

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/sports-cars.php"
title="Some very nice sports cars!"
>Sports Cars</a>
```

Save the file and reload the *first-site* homepage. You should now see the link to the *sports-cars* module's homepage as the first link in the 'Module Example1' meta-menu. Clicking on this link will take you to the homepage for the *sports-cars* module. This will load the *sports-cars* own custom navigation menu in the left-hand column, in place of the website's homepage menu.

This custom menu will be used by all pages in the *sports-cars* module.
You can use this menu to navigate around the default page templates, which we will customise shortly. Clicking on the title of the module's menu ('sports-cars') will take you back to the homepage for that module.

By default, the *create-new-module.xml* script creates the following page templates with integrated <**body**> content, under the module's *anyuser* sub-dir for the particular module you have just created:

```
anyuser
├── tmp-backups
├── 1-column-page.php
├── 2-column-page.php
├── 3-column-page.php
└── sports-cars.php
```

8.3.3 Adding pages to the module

Now we have a new *sports-cars* module, we can use the default page templates and add some usefull content to this module. First lets add some content to the module's homepage, to tell users what this section of the website is all about.

Open */develop/first-site/sports-cars/anyuser/sports-cars.php*
for editing.

In the section, change the contents of the

second , so it looks like this:

```
<div class="box2">

<p>Here is a collection of really nice high performance sports
cars, designed to be used on the public highway.  I hope to add
to this part of the website as more of these cars are found.</p>

</div>
```

Save the file and click on the title for the sports-car menu. The page should reload with the new contents above.

That should do for the sports-cars homepage. Now we need to add some pages with details of various sports cars.

Lets add 3 pages with details of the following cars:

- Audi GT3
- Corvette ZR1
- Shelby Ultimate Aero Twin Turbo (my favourite)

We will create a simple page for each sports car, showing the car's image, and a few details about the car. We'll use the *2-column-page.php* template in the *anyuser* directory as the basis for each car's page.

8.3.4 creating the Audi GT3 page

We'll call this page *audi-GT3.php* . Copy the *2-column-page.php* template to a new file called *audi-GT3.php* .

Open */develop/first-site/sports-cars/anyuser/audi-GT3.php*
for editing.

Edit the section so it looks the same as the example xhtml code in:
/framework-docs/tutorials/sports-cars/Audi-GT3/audi-GT3.html .

Now copy the image file *audi-GT3-1.jpg* from the *Audi-GT3* directory to

the *sports-cars* module image directory:
/develop/first-site/sports-cars/local/images/

That's the *audi-GT3.php* page contents sorted. Now we need to add a link from the *sports-cars* module's custom left-hand menu to the *audi-GT3.php* page.

Open */develop/first-site/sports-cars/local/incs/mod/sports-cars-menu.php* for editing.

As the first menu item after the end of the add this code:

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/audi-GT3.php"
title="Audi GT3 sports car"
>Audi GT3</a>
```

You can also delete the other menu items that refer to the 3-column, 2-column and single column page templates, as they are not needed now.

There should now be only one menu item in *sports-cars-menu.php*, a href link to the *audi-GT3.php* page, and the file should now look like this:

```
<?php
// get the global left-side menu file
// require "$SHARED_INCS/homepage-menu.php";
?>

<!-- use a customised LHS column menu instead of the
default one from the site homepage -->

<div class="internal_menu">

<div class="box1">
<span class="module_title_box">
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/sports-cars.php"
 id="module_home_link" title="sports-cars Home"
>sports-cars</a>
</span>
```

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/audi-GT3.php"
 title="Audi GT3 sports car"
>Audi GT3</a>

</div>
</div>
```

If we click on the title bar of the *sports-cars* menu, the homepage for this module should reload, and we should now see only one menu entry called 'Audi GT3' - the link to the *audi-GT3.php* page. Clicking on this link will take us to the page with a picture of the Audi GT3, and some details about the car. So that's the first of our three pages about high performance sports cars finished.

We can now create a page for the Corvette ZR1, and a page for the Shelby Ultimate Aero TT. We will use the *audi-GT3.php* page as a pattern for these other two cars' pages.

8.3.5 creating the Corvette ZR1 page

We'll call this page *corvette-ZR1.php* . Copy the *audi-GT3.php* page to a new file called *corvette-ZR1.php* .

Open */develop/first-site/sports-cars/anyuser/corvette-ZR1.php*
for editing.

Edit the section so it looks the same as the example xhtml code in:
/framework-docs/tutorials/sports-cars/Corvette-ZR1/corvette-ZR1.html .

Now copy the image file *corvette_ZR1-1.jpg* from the *Corvette-ZR1* directory to the *sports-cars* module image directory:
/develop/first-site/sports-cars/local/images/

That's the *corvette-ZR1.php* page contents sorted. Now we need to add a link from the *sports-cars* module's custom left-hand menu to the *corvette-ZR1.php* page.

Open */develop/first-site/sports-cars/local/incs/mod/sports-cars-menu.php*
for editing.

As the second menu item after the *audi-GT3.php* href link, add this code:

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/corvette-ZR1.php"
title="Corvette ZR1 sports car"
>Corvette ZR1</a>
```

After saving *sports-cars-menu.php*, if we click on the title bar of the *sports-cars* menu, the homepage for this module reloads, and we should now see the second menu entry called 'Corvette ZR1' - the link to the *corvette-ZR1.php* page. Clicking on this link will take us to the page with a picture of the Corvette ZR1, and some details about the car. So that's the second of our three pages about high performance sports cars finished.

8.3.6 creating the Shelby Ultimate Aero Twin Turbo page

We'll call this page *shelby-UA-TT.php* . Copy the *audi-GT3.php* page to a new file called *shelby-UA-TT.php* .

Open */develop/first-site/sports-cars/anyuser/shelby-UA-TT.php*
for editing.

Edit the section so it looks the same as the example xhtml code in:
/framework-docs/tutorials/sports-cars/Shelby-UA-TT/shelby-UA-TT.html .

Now copy the image file *ssc-ultimate-aero-1.jpg* from the *Shelby-UA-TT* directory to the *sports-cars* module image directory:
/develop/first-site/sports-cars/local/images/

That's the *shelby-UA-TT.php* page contents sorted. Now we need to add a link from the *sports-cars* module's custom left-hand menu to the *shelby-UA-TT.php* page.

Open */develop/first-site/sports-cars/local/incs/mod/sports-cars-menu.php*
for editing.

As the third menu item after the *corvette-ZR1.php* href link, add this code:

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/shelby-UA-TT.php"
title="Shelby Ultimate Aero Twin Turbo sports car"
>Shelby Ultimate Aero Twin Turbo</a>
```

After saving *sports-cars-menu.php*, if we click on the title bar of the *sports-cars* menu, the homepage for this module reloads, and we should now see the third menu entry called 'Shelby Ultimate Aero Twin Turbo' - the link to the *shelby-UA-TT.php* page. Clicking on this link will take us to the page with a picture of the Shelby Ultimate Aero Twin Turbo, and some details about the car. So that's the final page of our three pages about high performance sports cars finished.

8.4 Creating a new 'external content' module

The only difference between *create-new-module.xml* and *create-new-ext-module.xml* is this latter script creates the following page templates with external <**body**> content, under the module's *anyuser* sub-dir for the particular module you have just created:

```
anyuser
├── tmp-backups
├── 1-column-ext-page.php
├── 1-column-page.cont.php
├── 2-column-ext-page.php
├── 2-column-page.cont.php
├── 3-column-ext-page.php
├── 3-column-page.cont.php
├── sports-cars2.cont.php
└── sports-cars2.php
```

If we **cd** to the ***first-site*** directory, in the *develop* directory. It should now look like this:

```
first-site
├── fedora-services
├── home
├── module-example1
├── module-example2
├── module-example3
├── new-ext-module  <- module template
├── new-module      <- module template
├── shared
├── shared-host
├── sports-cars
├── tmp-backups
├── website-docs
├── create-new-ext-module.xml
├── create-new-module.xml
├── int-build.xml
├── int-delete.xml
├── live-build.xml
├── live-delete.xml
└── robots.txt
```

Lets now create another new module named *sports-cars2* . This will contain the same car pages as the *sports-cars* module, but will demonstrate how to use pages with external <**body**> content, instead of pages with integrated <**body**> content. This makes it easy to convert existing website pages to integrate into the Kazaam framework.

So lets run the PHING script *create-new-ext-module.xml* :

```
[root@karsites first-site]# phing -f create-new-ext-module.xml
Buildfile: /kazaam-mwf/develop/first-site/
            create-new-ext-module.xml

Create new external-content website module > get-module-name:
Please enter new module name: sports-cars2
[echo] Creating new kazaam 'sports-cars2' module

Create new external-content website module > copy-module-files:
```

```
[echo] Copying files to new 'sports-cars2' module directory...

[copy] Copying 94 files to
        /kazaam-mwf/develop/first-site/sports-cars2

[copy] Copied 4 empty directories to
        /kazaam-mwf/develop/first-site/sports-cars2

...
...
...

Create new external-content website module > clean-up:
[echo] Cleaning up unnecessary template files
[delete] Deleting 20 files from sports-cars2

Create new external-content website module > replace-module-text:
[echo] replacing 'new-ext-module' text with 'sports-cars2'
[echo] replacing 'New Ext Module' text with 'sports-cars2'
[reflexive] Applying reflexive processing to 94 files.

Create new external-content website module > module-created:
[echo] New kazaam module 'sports-cars2' created OK!

Create new external-content website module > create-ext-meta-menu:
[echo]
[echo] Creating meta-menu for new kazaam module 'sports-cars2'

Create new external-content website module > copy-meta-menu-files:
[echo] Copying 'new-ext-module.php' (meta-menu template files)
        to 'sports-cars2.php'

...
...
...

Create new external-content website module > replace-meta-menu-text:
[echo] replacing 'new-ext-module' text with 'sports-cars2'
[echo] replacing 'New Ext Module' text with 'sports-cars2'
[reflexive] Applying reflexive processing to 4 files.
```

```
Create new external-content website module > meta-menu-created:
[echo] meta-menu for new kazaam module 'sports-cars2' created OK!

BUILD FINISHED
Total time: 12.5390 seconds
```

If we do a directory listing, we should now have a new module in the *first-site* website directory, called **sports-cars2**:

```
first-site
├── fedora-services
├── home
├── module-example1
├── module-example2
├── module-example3
├── new-ext-module
├── new-module
├── shared
├── shared-host
├── sports-cars   <- new website module
├── sports-cars2  <- new 'external content' website module
├── tmp-backups
├── website-docs
├── create-new-ext-module.xml
├── create-new-module.xml
├── int-build.xml
├── int-delete.xml
├── live-build.xml
├── live-delete.xml
└── robots.txt
```

Now we have created a new module, we can access the module by 'plugging it into' the website in two different ways. We can:

1 add the *sports-cars2* meta-menu to the website's homepage menu that appears in the left-hand colum of all 2 or 3 column pages;

2 or add a link to the *sports-cars2* module's homepage, into an existing meta-menu of the website's homepage menu.

8.4.1 Adding the sports-cars2 meta-menu

To add the *sports-cars2* meta-menu to the website's homepage menu open: */develop/first-site/shared/incs/homepage-menu.php* ,

and add the following line to the top of the file:

```
<?php require "$SHARED_META_MENUS/sports-cars2.php"; ?>
```

Save the file and reload the *first-site* homepage. You should now see the *sports-cars2* meta-menu added at the top of the other meta-menu items in the left-hand column homepage menu. This is OK for a module that only links to a few pages. Clicking on any of the links in the *sports-cars2* meta-menu will take you to that particular page template. You will then be presented with that page template, and the module's own custom left-hand menu, in place of the homepage menu you have just come from.

NB. A module's custom left-hand column menu is *not* the same as the module's meta-menu. You should prefer the module's custom left-hand menu over the meta-menu, as it is more flexible and extendable, and will go with the module when you copy a module from one website to another website.

As our collection of sports-cars could grow considerably, we will opt to use the module's own custom left-hand column menu, instead of the homepage menu, and we will put one link on the homepage menu to the *sports-cars2.php* module's homepage.

Lets now remove the *sports-cars2* meta-menu from the homepage-menu:

Restore */develop/first-site/shared/incs/homepage-menu.php*
to what it was like before adding the:

```
<?php require "$SHARED_META_MENUS/sports-cars2.php"; ?>
```

line to it, and reload the first-site homepage. The *sports-cars2* meta-menu should be gone now.

8.4.2 Adding a single link to the sports-cars2.php homepage

Another way of adding the *sports-cars2* module to the website is to link directly to the module's homepage, in:

/develop/first-site/sports-cars2/anyuser/sports-cars2.php .

This will only use one link on the homepage menu. This link can be added to an existing meta-menu in:
/develop/first-site/shared/meta-menus/ .

Lets add a link from the 'Module Example1' meta-menu (fourth one down from the top of the homepage menu), to the *sports-cars2* module's homepage.

Open */develop/first-site/shared/meta-menus/module-example1.php*
and add the following lines of code as the second href link:

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars2/anyuser/sports-cars2.php"
title="Some very nice sports cars!"
>Sports Cars 2</a>
```

Save the file and reload the *first-site* homepage. You should now see the link to the *sports-cars2* module's homepage as the second link in the 'Module Example1' meta-menu (after *sports-cars*). Clicking on the *sports-cars2* link will take you to the homepage for the *sports-cars2* module. This will load the *sports-cars2* module's own custom navigation menu in the left-hand column, in place of the website's homepage menu.

This custom menu will be used by all pages in the *sports-cars2* module.
You can use this menu to navigate around the default page templates, which we will customise shortly. Clicking on the title of the module's menu ('sports-cars2') will take you back to the homepage for that module.

8.4.3 Adding pages to the module

Now we have a new *sports-cars2* module, we can use the default page templates and add some usefull content to this module. First lets add some content to the module's homepage, to tell users what this section of the website is all about.

If you look at the module's homepage:
/develop/first-site/sports-cars2/anyuser/sports-cars2.php ,

you will see that the section, consists of the following:

```
<div id="main_window_2col">
 <?php require "./sports-cars2.cont.php" ?>
</div>
```

So the actual content for the *sports-cars2.php* homepage has been moved out into *sports-cars2.cont.php*. This is the file we now need to edit to change the *sports-cars2.php* homepage. *sports-cars2.php* is now just a wrapper that includes *sports-cars2.cont.php*.

Open:
/develop/first-site/sports-cars2/anyuser/sports-cars2.cont.php
for editing.

Change the contents of the second , so it looks like this:

```
<div class="box2">

<p>Here is a collection of really nice high performance sports
cars, designed to be used on the public highway.  I hope to add
to this part of the website as more of these cars are found.</p>

</div>
```

Save the file and click on the title for the sports-cars2 menu. The page should reload with the new contents above.

That should do for the sports-cars2 homepage. Now we need to add some pages with details of various sports cars.

Lets add 3 pages with details of the following cars:

- Audi GT3
- Corvette ZR1
- Shelby Ultimate Aero Twin Turbo (my favourite)

We will create a simple page for each sports car, showing the car's image, and a few details about the car. We'll use the *2-column-ext-page.php* template as the wrapper for the each of the car pages, and put the content for each page in *2-column-page.cont.php*. These files are in the *anyuser* directory of the *sports-cars2* module.

8.4.4 creating the Audi GT3 page

We'll call this page *audi-GT3.php* .

Copy the *2-column-ext-page.php* template to a new file called *audi-GT3.php* .

Copy the *2-column-page.cont.php* template to a new file called *audi-GT3.cont.php* .

Open */develop/first-site/sports-cars/anyuser/audi-GT3.php*
for editing.

Edit the section so it looks the same as the example xhtml code in:
/framework-docs/tutorials/sports-cars/Audi-GT3/audi-GT3.html .

Now copy the image file *audi-GT3-1.jpg* from the *Audi-GT3* directory to the *sports-cars* module image directory:
/develop/first-site/sports-cars/local/images/

That's the *audi-GT3.php* page contents sorted. Now we need to add a link from the *sports-cars* module's custom left-hand menu to the *audi-GT3.php* page.

Open */develop/first-site/sports-cars/local/incs/mod/sports-cars-menu.php*
for editing.

As the first menu item after the end of the
add this code:

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/audi-GT3.php"
title="Audi GT3 sports car"
>Audi GT3</a>
```

You can also delete the other menu items that refer to the 3-column, 2-column and single column page templates, as they are not needed now.

There should now be only one menu item in *sports-cars-menu.php*, a href link to the *audi-GT3.php* page, and the file should now look like this:

```
<?php
// get the global left-side menu file
// require "$SHARED_INCS/homepage-menu.php";
?>

<!-- use a customised LHS column menu instead of the
default one from the site homepage -->

<div class="internal_menu">

<div class="box1">
<span class="module_title_box">
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/sports-cars.php"
 id="module_home_link" title="sports-cars Home"
>sports-cars</a>
</span>

<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/audi-GT3.php"
 title="Audi GT3 sports car"
>Audi GT3</a>

</div>
</div>
```

If we click on the title bar of the *sports-cars* menu, the homepage for this module should reload, and we should now see only one menu entry called 'Audi GT3' - the link to the *audi-GT3.php* page. Clicking on this link will take us to the page with a picture of the Audi GT3, and some details about the car. So that's the first of our three pages about high performance sports cars finished.

We can now create a page for the Corvette ZR1, and a page for the Shelby Ultimate Aero TT. We will use the *audi-GT3.php* page as a pattern for these other two cars' pages.

8.4.5 creating the Corvette ZR1 page

We'll call this page *corvette-ZR1.php* . Copy the *audi-GT3.php* page to a new file called *corvette-ZR1.php* .

Open */develop/first-site/sports-cars/anyuser/corvette-ZR1.php*
for editing.

Edit the section so it looks the same as the example xhtml code in:
/framework-docs/tutorials/sports-cars/Corvette-ZR1/corvette-ZR1.html .

Now copy the image file *corvette_ZR1-1.jpg* from the *Corvette-ZR1* directory to the *sports-cars* module image directory:
/develop/first-site/sports-cars/local/images/

That's the *corvette-ZR1.php* page contents sorted. Now we need to add a link from the *sports-cars* module's custom left-hand menu to the *corvette-ZR1.php* page.

Open */develop/first-site/sports-cars/local/incs/mod/sports-cars-menu.php*
for editing.

As the second menu item after the *audi-GT3.php* href link, add this code:

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/corvette-ZR1.php"
title="Corvette ZR1 sports car"
>Corvette ZR1</a>
```

After saving *sports-cars-menu.php*, if we click on the title bar of the *sports-cars* menu, the homepage for this module reloads, and we should now see the second menu entry called 'Corvette ZR1' - the link to the *corvette-ZR1.php* page. Clicking on this link will take us to the page with a picture of the Corvette ZR1, and some details about the car. So that's the second of our three pages about high performance sports cars finished.

8.4.6 creating the Shelby Ultimate Aero Twin Turbo page

We'll call this page *shelby-UA-TT.php* . Copy the *audi-GT3.php* page to a new file called *shelby-UA-TT.php* .

Open */develop/first-site/sports-cars/anyuser/shelby-UA-TT.php*
for editing.

Edit the section so it looks the same as the example xhtml code in:
/framework-docs/tutorials/sports-cars/Shelby-UA-TT/shelby-UA-TT.html .

Now copy the image file *ssc-ultimate-aero-1.jpg* from the *Shelby-UA-TT* directory to the *sports-cars* module image directory:
/develop/first-site/sports-cars/local/images/

That's the *shelby-UA-TT.php* page contents sorted. Now we need to add a link from the *sports-cars* module's custom left-hand menu to the *shelby-UA-TT.php* page.

Open */develop/first-site/sports-cars/local/incs/mod/sports-cars-menu.php*
for editing.

As the third menu item after the *corvette-ZR1.php* href link, add this code:

```
<a href="<?php echo $WEBSITE_ROOT ?>/sports-cars/anyuser/shelby-UA-TT.php"
title="Shelby Ultimate Aero Twin Turbo sports car"
>Shelby Ultimate Aero Twin Turbo</a>
```

After saving *sports-cars-menu.php*, if we click on the title bar of the *sports-cars* menu, the homepage for this module reloads, and we should now see the third menu entry called 'Shelby Ultimate Aero Twin Turbo' - the link to the *shelby-UA-TT.php* page. Clicking on this link will take us to the page with a picture of the Shelby Ultimate Aero Twin Turbo, and some details about the car. So that's the final page of our three pages about high performance sports cars finished.

A Start of Appendix