BitWasp Architecture Analysis

ampedup [amphetamine@tormail.org]

Contents

1 Introduction

This document aims to provide a clear, well specified summary of the BItWasp project and very clear indiciations of its progress so far based solely on the existing source for BitWasp.

1.1 Rationale

Properly specifying BitWasp and evaluating its progress is a key step before performing any modifications to the software. This information will be invaluable to further project decision making and will also greatly benefit the original BitWasp project.

1.2 Methodology

The structure analysis of the software has been done using Visual Paradigm to extract class data organized by directory. Dependencies and coding conventions are discovered through both the UML class design and manual code auditing.

2 Coding Conventions

2.1 Naming Conventions

2.1.1 Classes

Class names are created using CapitalisedCase.

2.1.2 Methods

Method names are created using camelCase.

2.1.3 Attributes

Attribute names are created using camelCase.

2.1.4 Instance Variables

Attribute names are created using camelCase.

2.2 Syntax Conventions

2.2.1 PHP Tags

<?php is used for opening tags. No closing tag is used. The *<?=* syntax was used but replaced with *<?php echo* instead to allow for compatibility with Visual Paradigm software.

2.2.2 Classes

The opening bracket for classes is on the same line as the beginning of the class definition.

2.3 Layout Conventions

Tabs are used for indentation, not spaces.

3 Dependencies

3.1 Frameworks

3.1.1 CodeIgniter

3.2 PHP Libraries

3.2.1 php-gd

3.3 External Software

3.3.1 GPG

3.3.2 bitcoind

4 Path Structure

4.1 Application (application/)

Description: Contains the bulk of the BitWasp web application code.

Files:

index.html
HTML file to prevent directory browsing.

4.1.1 application/config/

Description: Contains application configuration data for CodeIgniter based applications.

Files:

autoload.php
Specifies which systems should be loaded by default.

config.php
Contains configuration settings for the BitWasp installation, such as Bitcoin configuration, etc.

constants.php
Contains definitions of constant values.

doctypes.php
Contains an array for DOCTYPE definitions.

foreign_chars.php
Contains an array of foreign characters for transliteration conversion used by the Text helper.

form_validation.php
Contains definitions for data sanitization from sent form data.

hooks.php
Lets you define "hooks" to extend CI without hacking the core files. Unused in BitWasp.

index.html
HTML file to prevent directory browsing.

migration.php
Contains data for schema migrations. Disabled by default and not used in BitWasp.

mimes.php
Contains an array of MIME types to help identify allowed file types.

profiler.php
Configures whether or not the Profiler data is shown when enabled. Unused in BitWasp.

routes.php
Contains URL mappings to specific controller functions.

smileys.php
Contains an array of smileys for converting smiley text to images.

user_agents.php
Contains arrays of user agent data to assist in identifying connected clients' platforms and browsers.

4.1.2 application/controllers/

Description: Contains the PHP files for the controller classes.

Files:

account.php
Contains code for the Account class.

admin.php
Contains code for the Admin class.

bitcoin.php
Contains code for the Bitcoin class.

error.php
Contains code for the Error class.

home.php
Contains code for the Home class.

index.html
HTML file to prevent directory browsing.

items.php
Contains code for the Items class.

listings.php
Contains code for the Listings class.

messages.php
Contains code for the Messages class.

orders.php
Contains code for the Orders class.

pages.php
Contains code for the Pages class.

users.php
Contains code for the Users class.

4.1.3 application/errors/

Description: Contains HTML to be displayed for various HTTP error codes.

Files:

file1

file2

4.1.4 application/language/english

Description: Contains the translation data for English.

Files:

form_validation_lang.php

index.html

4.1.5 application/libraries/

Description: Contains library classes of the BitWasp application.

Files:

file1

file2

4.1.6 application/models/

Description: Contains the Model definition data for BitWasp objects.

Files:

file1

file2

4.1.7 application/views/

Description: Contains layout data for pages of BitWasp.

Files:

file1

file2

4.2 System (system/)

4.2.1 core

Description: Contains the .

Files:

file1

file2

4.2.2 database

Description: TBD.

Files:

file1

file2

4.2.3 helpers

Description: TBD.

Files:

file1

file2

4.2.4 language

Description: TBD.

Files:

file1

file2

4.2.5 libraries

Description: TBD.

Files:

file1

file2

4.3 Root directory (/)

Description: TBD.

Files:

file1

file2

5 Class Structure

5.1 Controllers

5.1.1 Account

Description: The class handles the Account management interface. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Account object. Calls the parent constuctor and loads required libraries and models.

index()
Loads data for the account/index page ("My Account") and loads the layout library.

edit()
Loads data for the account/edit page ("Edit Account") and loads the layout library.

replacePGP()
Loads data for the account/replacePGP page ("Replace PGP key"), handles the updating of PGP keys and loads the layout library.

update()
Handles updating of the account information.

deletePubKey()
Delete the stored pubKey for the user, and disable two-step authentication if necessary.

5.1.2 Admin

Description: The class handles the Admin management interface. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Account object. Calls the parent constuctor and loads required the model.

index()
Loads data for the admin/siteConfig page ("Admin Panel") and loads the layout library.

users($userHash = NULL)
Loads data for the admin/users page ("Users") and loads the layout library.

editConfig()
Loads data for the admin/editConfig page ("Edit Configuration") and loads the layout library.

updateConfig()
Handles updating of the site configuration.

fixOrphans()
Handles the removal of orphan categories.

removeCategory()
Loads data for the admin/removeCategory page ("Remove Category"), handles category removal and loads the layout library.

addCategory()
Loads data for the admin/addCategory page ("Add Category"), handles category addition and loads the layout library.

check_parentID_exists($parentID)
Method does not follow naming conventions! Checks that the category identified by $parentID has a parent that exists.

check_category_exists($id)
Method does not follow naming conventions! Checks that the category identified by $id exists.

5.1.3 Bitcoin

Description: The class handles the Bitcoin management interface. It is in its very early stages. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Account object. Calls the parent constuctor and loads required the model.

index()
Loads some test data using a JSON interface to a hardcoded bitcoin server using hardcoded credentials and loads the layout library.

5.1.4 Error

Description: The class handles displaying of errors. Extends CI_Controller.

Attributes:

$params
Array instance variable not actually used in the code.

Methods:

__construct()
Empty constructor.

index($params)
Loads data pertaining to the error and displays the error.

5.1.5 Home

Description: The class handles display of the home page. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Home object. Calls the parent constuctor.

index()
Loads data for the home/index page ("Home") and loads the layout library.

5.1.6 Items

Description: The class handles the display of items in the marketplace. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Items object. Calls the parent constuctor and loads required the models and libraries.

tmp_items_per_page()
Method does not follow naming conventions! Handles the changing of items shown per page for the session.

index()
Loads data for the items/index page ("Items"), "paginates" the lastest items and loads the layout library.

view($itemHash)
Loads data specified item and displays it if it exists, otherwise it "paginates" the lastest items loads the layout library. Display of user feedback is commented out.

cat($catID = FALSE)
Loads items for specified category if it exists and "paginates" the items, otherwise it "paginates" the latest items and loads the layout library.

check_category_exists($id)
Method does not follow naming conventions! Checks whether or not the specified category exists.

5.1.7 Listings

Description: The class handles the management of listings in the marketplace. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Listings object. Calls the parent constuctor and loads required the models and libraries.

edit($itemHash)
Loads data for the listings/editConfig page ("Edit Configuration") and loads the layout library.

manage()
Loads data for the listings/manage page ("Manage Listings") and loads the layout library.

images($itemHash)
Loads data for the listings/images page ("Item Images") and loads the layout library.

mainImage($imageHash)
Handles the selection of the main image for listings, loads the data for the result and loads the layout library.

imageRemove($imageHash)
Handles the removal of images for listings, loads the data for the result and loads the layout library.

imageUpload($itemHash)
Handles the upload of images for listings, loads the data for the result and loads the layout library.

remove($itemHash)
Handles the removal of items from vendor listings, loads the data for the result and loads the layout library.

create()
Loads data for the listings/addItem page ("Add Item") and loads the layout library.

5.1.8 Messages

Description: The class handles the management of user messages. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Messages object. Calls the parent constuctor and loads required the models and libraries.

inbox()
Loads data for the messages/inbox page ("Inbox") and loads the layout library.

delete($messageHash)
Handles the deletion of messages, loads the data for the result and loads the layout library.

read($messageHash)
Loads data for the specified message for the messages/read page and loads the layout library.

remove($itemHash)
Handles the removal of items from vendor listings, loads the data for the result and loads the layout library.

send($toHash = NULL)
Loads data for the messages/send page ("Send Message"), handles message sending and loads the layout library.

5.1.9 Orders

Description: The class handles the management of orders. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Orders object. Calls the parent constuctor and loads required the models and libraries.

index()
Loads data for the orders/index page ("Orders") and loads the layout library.

recount()
Handles the update of order quantities, loads the data for the result and loads the layout library.

orderItem($itemHash)
Handles the creation of a new order for the specified item, loads the data for the result and loads the layout library.

place($sellerHash)
Handles the placement of orders with vendors, loads the data for the result and loads the layout library.

review($id)
Handles the placement of feedback on orders, loads the data for the result and loads the layout library.

confirmDispatch($buyerHash)
Handles the update of dispatch status on orders, loads the data for the result and loads the layout library.

confirmPayment($buyerHash)
Handles the payment confirmation on orders, loads the data for the result and loads the layout library.

purchases()
Loads data for the orders/purchases page ("Purchases") and loads the layout library.

callback_check_rating($param)
Method does not follow naming conventions! Checks whether the feedback value provided is within the specified range (1-5 inclusive).

5.1.10 Pages

Description: The class handles display of the custom pages. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
Constructor for the Pages object. Calls the parent constuctor.

view($page = 'home')
Loads data for the specified page and loads the layout library.

5.1.11 Users

Description: The class handles the user management interface. Extends CI_Controller.

Attributes: None.

Methods:

__construct()
: Constructor for the Users object. Calls the parent constuctor and loads required the models and libraries.

registerPGP()
: Loads data for the users/registerPGP page ("Setup PGP"), handles PGP key registration and loads the layout library.

twoStep()
: Loads data for the users/twoStep page ("Two Step Authentication"), handles verification of the two step authentication and loads the layout library.

logoutInactivity()
: Loads data for the users/login page ("Login"), notifies the user that the previous session has timed out, and loads the layout library.

view($userHash)
: Loads data for the users/individual page for the specified user and loads the layout library.

login()
: Loads data for the users/login page ("Login"), handles user credential and CAPTCHA authentication and loads the layout library.

logout()
: Logs out the user and redirects them to the users/login page.

register()
: Loads data for the users/register page ("Register"), handles user registration and CAPTCHA authentication and loads the layout library.

find_role($roleId)
: **Method does not follow naming conventions!** Resolves the role ID to a string "Vendor" or "Buyer".

check_captcha($string)
: **Method does not follow naming conventions!** Checks the validity of the CAPTCHA answer.

register_check_role($role)
: **Method does not follow naming conventions!** Checks whether the role value provided is within the specified range (1-2 inclusive).