eMarketting portal

Software Engineering Project 2014
Coding Standards Document

Project ID: WE-SEP-004

Submitted by:

1. IT12088942– (K. M. K. N. B. Gamhatha)
2. IT12009978– (A. N. M. R. H. S Athurupana)
3. IT12015122– (H. M. C. I. Gunathunga)
4. IT12021512– (U. D. R. Piumal)

Class and Method Naming

Class names are always starting with an uppercase letter. Multiple words are separated with an underscore. Camel Cased notation is not used. The class name reflects the work done by that particular class. Avoid using long and verbose names

Eg – class **Account_settings_controller**

class **Comment_controller**

class **Membership_model**

All other class methods are entirely lowercased and underscores are used to separate words. Camel cased notation is not used. Methods are named to clearly indicate their function, preferably including a verb.

Eg - function **validate_credentials()**

function **get_user_type()**

function **send_email()**

Variable Names

Guidelines are very similar to that used for class methods. Variables names have only lowercase letters and use underscore separators. Variables are named to indicate their purpose and content. Avoid using non meaningful and single lettered variables .

Eg - $is_logged_in

$seller_email

$advertisement

Commenting

The code should be commented prolifically. It helps us to understand the code after writing it and keeping for a long time. We would be easily able to understand the work we have done. It also help others to understand the code who go through it.

DocBlock style comments used to define class and method details

```
/*
 * Email_controller class whick handles tasks related to sending email to the seller
 *
 */
```

Multiline comments are used to describe larger contents using many sentences.

```
/*
*/
```

Use of single line comments , leaving a blank line between comment and code. Single line comments are used to describe small sections in lesser number of sentences.

```
//Getting the sellers email from sellers user  ID
$data['sellers_email'] = $this->email_model->get_user_email($sellers_user_id->user_id);
//Validating name, phone fields
 $this->form_validation->set_rules('name', 'Name', 'required|apha');
```

Constants

Constants follow the same guidelines as variables, but constants should always be fully uppercase. Always use CodeIgniter constants when appropriate, i.e. SLASH, LD, RD, PATH_CACHE, etc.

Eg - **MY_CONSTANT**

TRUE, FALSE, and NULL

TRUE, FALSE, and NULL are always defined in uppercases

```
Eg -    if ($this->form_validation->run() == FALSE)
        if($result)
        {
                return TRUE;
         }
        else
        {
                return FALSE;
        }
```

Logical Operators

|| For OR operator

&& For AND operator

! For not

!= For not equal

```
Eg-     if ( ! isset($is_logged_in) || $is_logged_in != TRUE )
```

Comparing Return Values and Typecasting

Comparisons inside if and else if operators are done using = = , ! =

= = Checks whether two values are equal

! = Checks whether two values are not equal

Eg- if ($current_password = = $current_password_entered)

if ($is_logged_in != TRUE)

Class and File Names using Common Words

When class or variable names are to be named with common words append or prepend with another word.

Eg - class **Email_controller**

class

class

Whitespace in Files

No whitespace can precede the opening PHP tag or follow the closing PHP tag. Output is buffered, so whitespace in files can cause output to begin before CodeIgniter outputs its content.

```
<?php

    // No white spaces present before opening and after closing tags of php

?>
```

Code Indenting

Allman style indenting is used. With the exception of Class declarations, braces are always placed on a separate line by themselves, and indented at the same level as the control statement that "owns" them.

```
if ($this->form_validation->run() == FALSE)
 {
        $this ->view_load();
 }
else
{
        if ($current_password = = $current_password_entered)
        {
                $email = $this->session->userdata('email');
                //Update the password of the logged user
                $this ->load->model('Account_settings_model');
                $this ->Account_settings_model->update_password($email);
                $this ->update_success();
        }
}
```

MySQL Queries

MySQL keywords are always capitalized

SELECT, INSERT, UPDATE, WHERE, AS, JOIN, ON, IN.

Strings

Always use single quoted strings unless you need variables parsed, and in cases where you do need variables parsed, use braces to prevent greedy token parsing. You may also use double-quoted strings if the string contains single quotes, so you do not have to use escape characters.

Eg- `$data['error'] = 'Entered Current Password not matched with actual password';`

Bracket and Parenthetic Spacing

A space should always follow PHP control structures that accept arguments with parenthesis (such as if, elseif, for, foreach, switch, while), to help distinguish them from functions and increase readability.

No whie spaces arounf array keys

Eg- `$data['error']`

No spaces around parenthesis in function declarations

Eg- `public function delete($user_id)`

Other Guidelines

- Use a separate file for one class.
- Use tabs instead of white spaces.
- Write one statement per line.