JEJ_MIS

Technical Documentation

Revised Draft, April 17, 2015

Author: Corsanes, John Cyrill C.

johncyrillcorsanes@gmail.com

Software Engineer from THE BIG FIVE

TABLE OF CONTENTS

JEJ_MIS TECHNICAL INFORMATION FOR 'THE BIG FIVE' AND FUTURE DEVELOPERS

Abstract

The JEJ_MIS is the Management Information System for Company we've been proposing (Confidentiality). This system is a Web Science study using the latest technologies such as PHP, HTML and MySQL. The study is based on my Web Science Research with Objectives, Scopes and Limitations by the past few chapters of our (THE BIG FIVE) study.

The JEJ_MIS was made with MINI and PHP-LOGIN 2.1.1 by panique. This was inspired from MVC (Model-View-Controller) Concept and Microsoft's MVC Web Framework. We've combined the two frameworks into one and the results are good as usual.

Introduction

Many technologies rapidly evolved in every second of our lives. Web Science is already involved all over the world such as using Social Media (like Facebook, Twitter, Tumblr), Streaming Multimedia Content and even creating projects without paying for proprietary programs and applications.

Also the industry rapidly evolving every year, even every quarter of the year and it occurs constraints and problems for those organizations which system should they use in the industry. Many Organizations aware and finding the suitable system that is reliable, understandable, feasible, accessible and user-friendly.

Major IT Companies like IBM, MICROSOFT, ORACLE, and CISCO leads in the IT industry today. Some IT people does not need to learn anything from complex standards if there is more secured and easy to use for developers, software engineers, and system analysts. Open-source Technologies is one of the leading technologies today because it's free to install and use, and we've decided to develop a Management Information System for Company we've proposed using the Open-Source technologies such as PHP and MySQL.

Technical Objectives

The technical objectives for this system is to ensure the functionality, stability, compatibility, security and user-friendliness.

Technical Features

Here some proposed and to be features to propose.

- Using MVC (Model-View-Controller) skeleton
- Inspired from Microsoft ASP.NET Web framework
- No extra syntax required to learn (Unlike CakePHP and CodeIgniter)
- Multi-Layered Security features (especially login and CRUD actions)
- Mobile Friendly UI
- Backward Compatibility with older PHP versions

System Requirements

Here’s the Recommended Technical Requirements for this system.

End-User Requirements

- **CPU/Processor:** Any Quad Core Intel or AMD 64-bit Processor
- **RAM:** 4-8GB DDR3
- **Hard Drive:** At least 300GB of HDD/SSD (SATA or IDE)
- **Network:** Any internet connection at least 3mbps
- **Operating System:** Any OS such as Windows/Linux/Mac OS X, etc.
- **Others:** XAMPP/WAMPP is required with all installed components

End-User Requirements for Mobile

For iOS Devices

- **Devices:** At least iOS 7 phone/tablet or higher

For Other Devices

- **Operating System:** At least Android 4.1 Jelly Bean or higher

Server Requirements

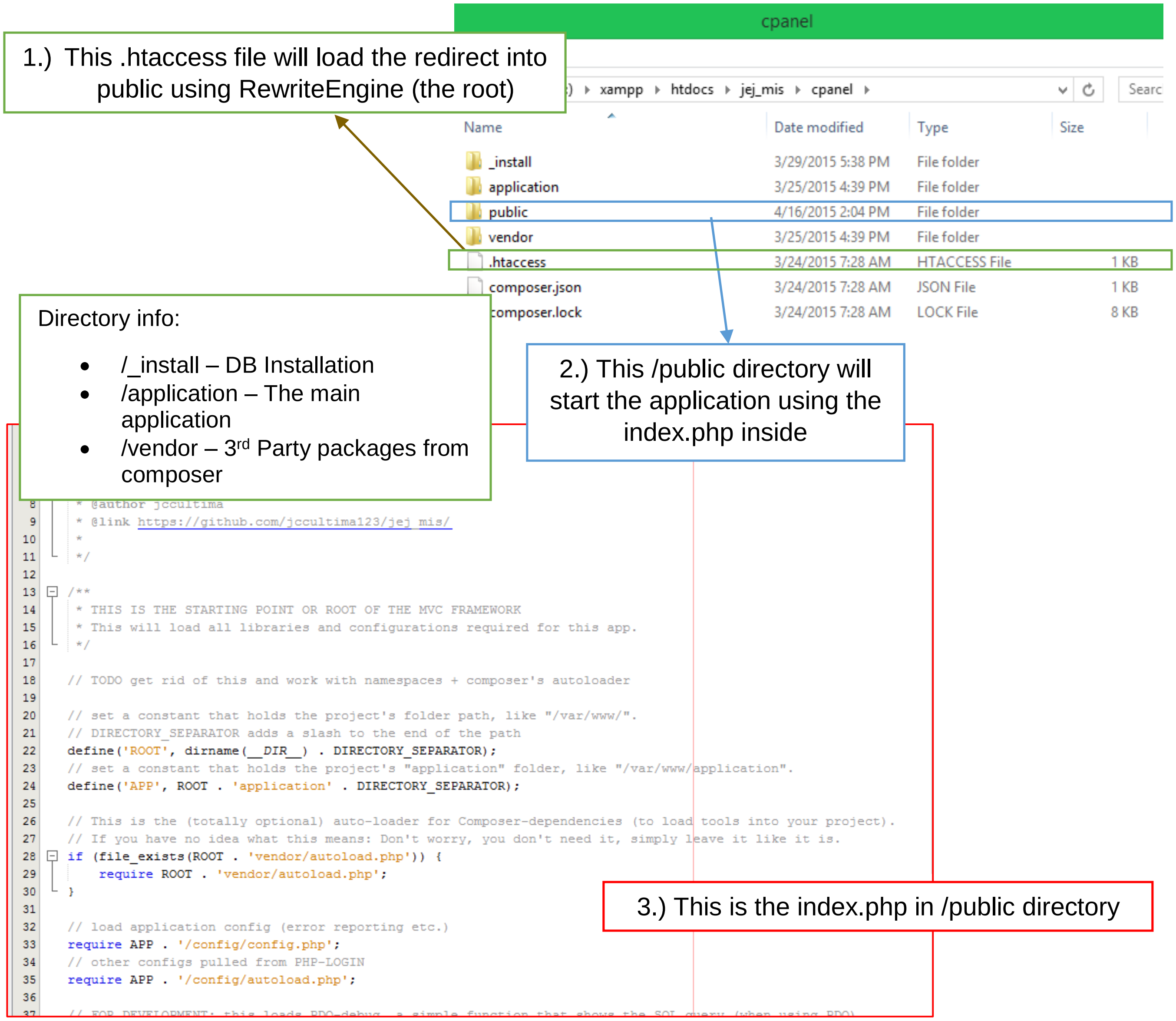
Coming Soon

Installation Instructions

1. Make sure that you are running XAMPP with Apache and MySQL Servers running.
2. Download the ZIP file or pull from master on jejMIS repository.
3. Make sure the repository is on XAMPP's htdocs directory with the new folder (name is optional)
4. Go to MySQL Admin button in XAMPP Control Panel.
5. Make sure that your database is filled with users. (SQL file coming soon)
6. Go to your application and modify settings in **/htdocs/*your new app*/cpanel/application/config/config.php
7. Now run the application (e.g. http://localhost/*your folder name in htdocs*/)

Getting Started

How the MVC Framework skeleton works?



Name	Date modified	Type	Size
_install	3/29/2015 5:38 PM	File folder	
application	3/25/2015 4:39 PM	File folder	
public	4/16/2015 2:04 PM	File folder	
vendor	3/25/2015 4:39 PM	File folder	
.htaccess	3/24/2015 7:28 AM	HTACCESS File	1 KB
composer.json	3/24/2015 7:28 AM	JSON File	1 KB
composer.lock	3/24/2015 7:28 AM	LOCK File	8 KB

1.) This .htaccess file will load the redirect into public using RewriteEngine (the root)

Directory info:

- /_install – DB Installation
- /application – The main application
- /vendor – 3rd Party packages from composer

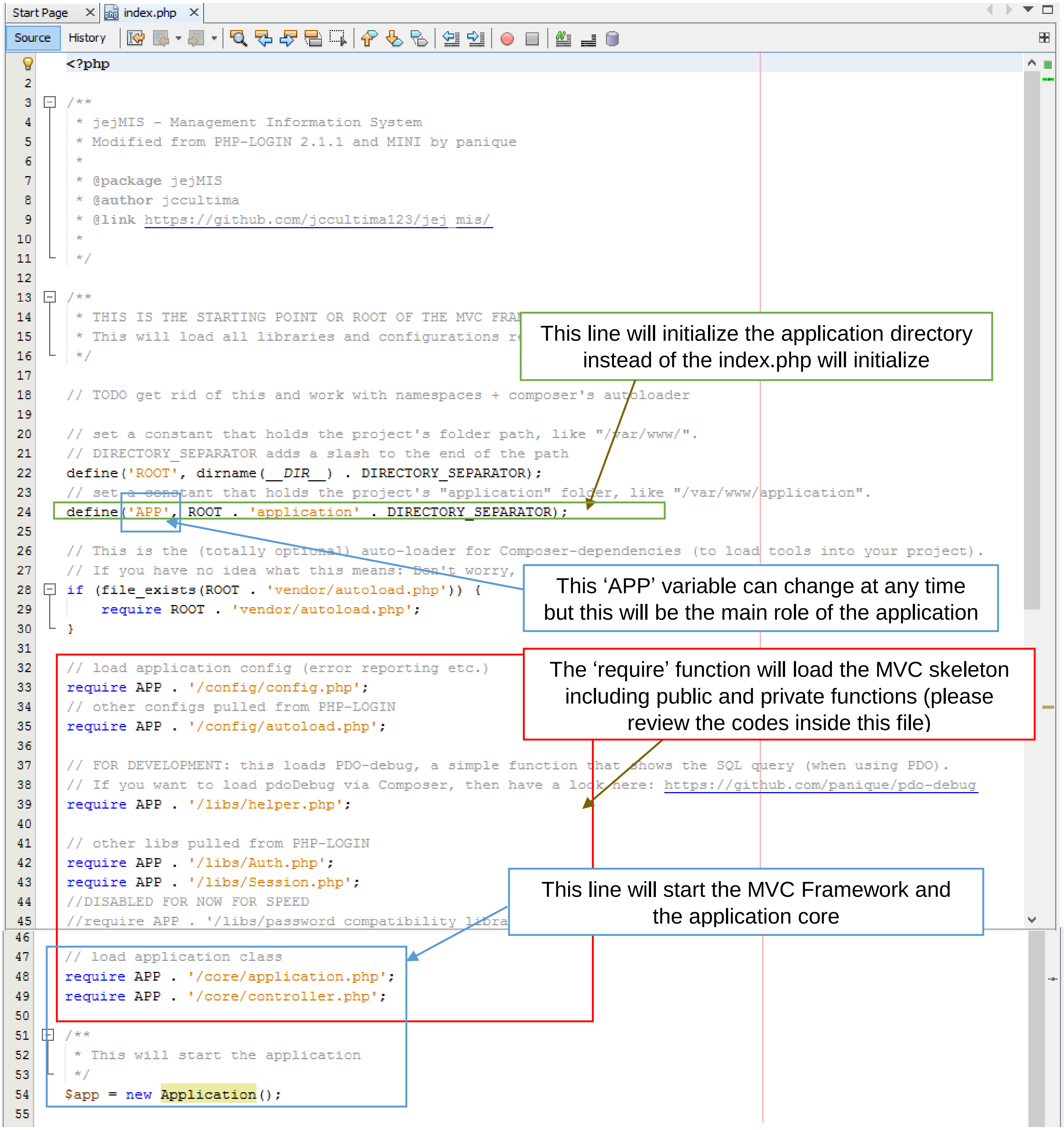
2.) This /public directory will start the application using the index.php inside

```
 * @author jccultima
 * @link https://github.com/jccultima123/jej_mis/
 *
 */

/**
 * THIS IS THE STARTING POINT OR ROOT OF THE MVC FRAMEWORK
 * This will load all libraries and configurations required for this app.
 */

// TODO get rid of this and work with namespaces + composer's autoloader

// set a constant that holds the project's folder path, like "/var/www/".
// DIRECTORY_SEPARATOR adds a slash to the end of the path
define('ROOT', dirname(__DIR__) . DIRECTORY_SEPARATOR);
// set a constant that holds the project's "application" folder, like "/var/www/application".
define('APP', ROOT . 'application' . DIRECTORY_SEPARATOR);

// This is the (totally optional) auto-loader for Composer-dependencies (to load tools into your project).
// If you have no idea what this means: Don't worry, you don't need it, simply leave it like it is.
if (file_exists(ROOT . 'vendor/autoload.php')) {
    require ROOT . 'vendor/autoload.php';
}

// load application config (error reporting etc.)
require APP . '/config/config.php';
// other configs pulled from PHP-LOGIN
require APP . '/config/autoload.php';

// FOR DEVELOPMENT: this loads PDO-debug, a simple function that shows the SQL query (when using PDO)
```

3.) This is the index.php in /public directory

Accessing the Main Application

As we can see this code below in /public/index.php. This is the starting point of the application.



```php
<?php

/**
 * jejMIS - Management Information System
 * Modified from PHP-LOGIN 2.1.1 and MINI by panique
 *
 * @package jejMIS
 * @author jccultima
 * @link https://github.com/jccultima123/jej_mis/
 *
 */

/**
 * THIS IS THE STARTING POINT OR ROOT OF THE MVC FRA
 * This will load all libraries and configurations re
 */

// TODO get rid of this and work with namespaces + composer's autoloader

// set a constant that holds the project's folder path, like "/var/www/".
// DIRECTORY_SEPARATOR adds a slash to the end of the path
define('ROOT', dirname(__DIR__) . DIRECTORY_SEPARATOR);
// set a constant that holds the project's "application" folder, like "/var/www/application".
define('APP', ROOT . 'application' . DIRECTORY_SEPARATOR);

// This is the (totally optional) auto-loader for Composer-dependencies (to load tools into your project).
// If you have no idea what this means: Don't worry,
if (file_exists(ROOT . 'vendor/autoload.php')) {
    require ROOT . 'vendor/autoload.php';
}

// load application config (error reporting etc.)
require APP . '/config/config.php';
// other configs pulled from PHP-LOGIN
require APP . '/config/autoload.php';

// FOR DEVELOPMENT: this loads PDO-debug, a simple function that shows the SQL query (when using PDO).
// If you want to load pdoDebug via Composer, then have a look here: https://github.com/panique/pdo-debug
require APP . '/libs/helper.php';

// other libs pulled from PHP-LOGIN
require APP . '/libs/Auth.php';
require APP . '/libs/Session.php';
//DISABLED FOR NOW FOR SPEED
//require APP . '/libs/password_compatibility_libra

// load application class
require APP . '/core/application.php';
require APP . '/core/controller.php';

/**
 * This will start the application
 */
$app = new Application();

```

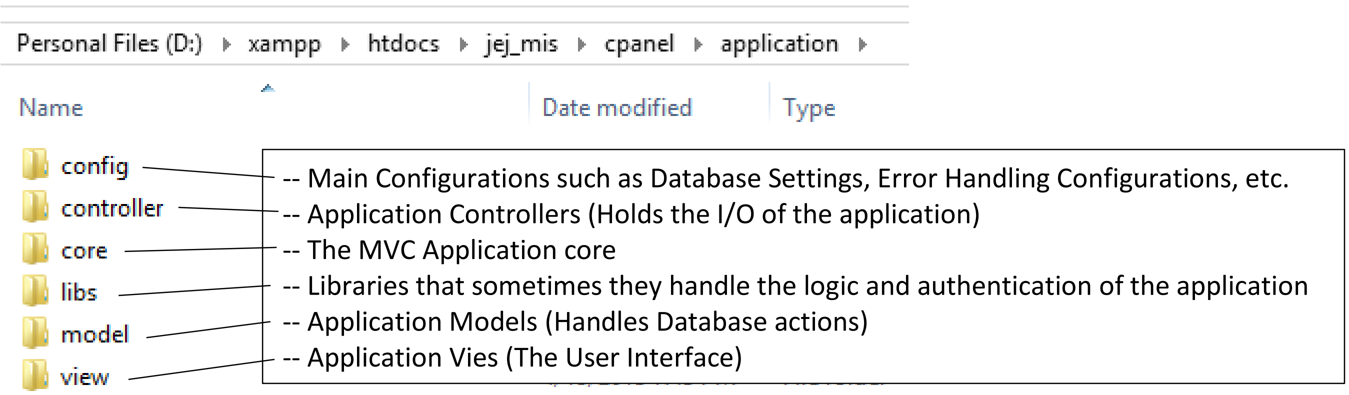
This line will initialize the application directory instead of the index.php will initialize

This 'APP' variable can change at any time but this will be the main role of the application

The 'require' function will load the MVC skeleton including public and private functions (please review the codes inside this file)

This line will start the MVC Framework and the application core

Accessing the Main Application


Personal Files (D:) ▸ xampp ▸ htdocs ▸ jej_mis ▸ cpanel ▸ application ▸
Name
Date modified
Type
config
controller
core
libs
model
view

-- Main Configurations such as Database Settings, Error Handling Configurations, etc.
-- Application Controllers (Holds the I/O of the application)
-- The MVC Application core
-- Libraries that sometimes they handle the logic and authentication of the application
-- Application Models (Handles Database actions)
-- Application Vies (The User Interface)

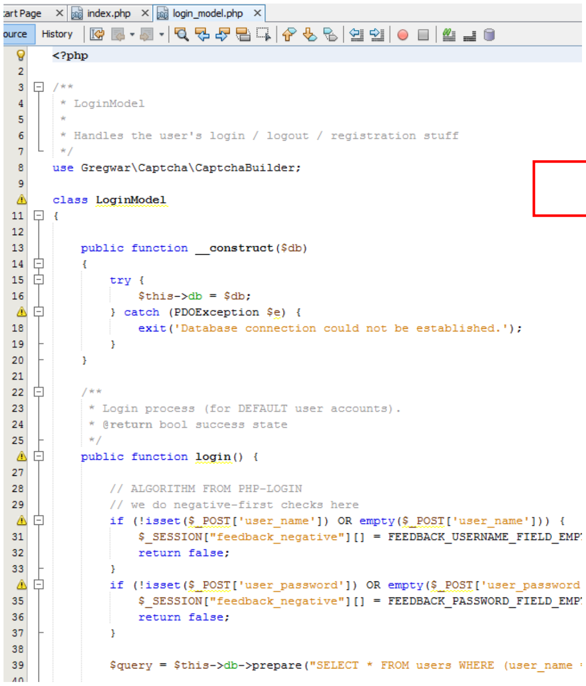
This is the MVC Structure of the Main Application

Creating Application Features/Page

By creating a page or by adding a feature, the developer must presence the three needs for this application, The Model – View – Controller concept. If one of them misses, the application might occur unknown errors.

The Model

The Model will handle the query actions to the database. The model can do everything to the database using the PDO Controller. PDO is widely used in large-scale systems and it’s suitable in this application. PDO can do the CRUD (Create, Rename, Update, and Delete) actions. If you are creating a page without any feature (i.e. about page), the model might not be needed. But if you are creating a feature outside the home page, you must call a model or instantiate it in /core/controller.php.

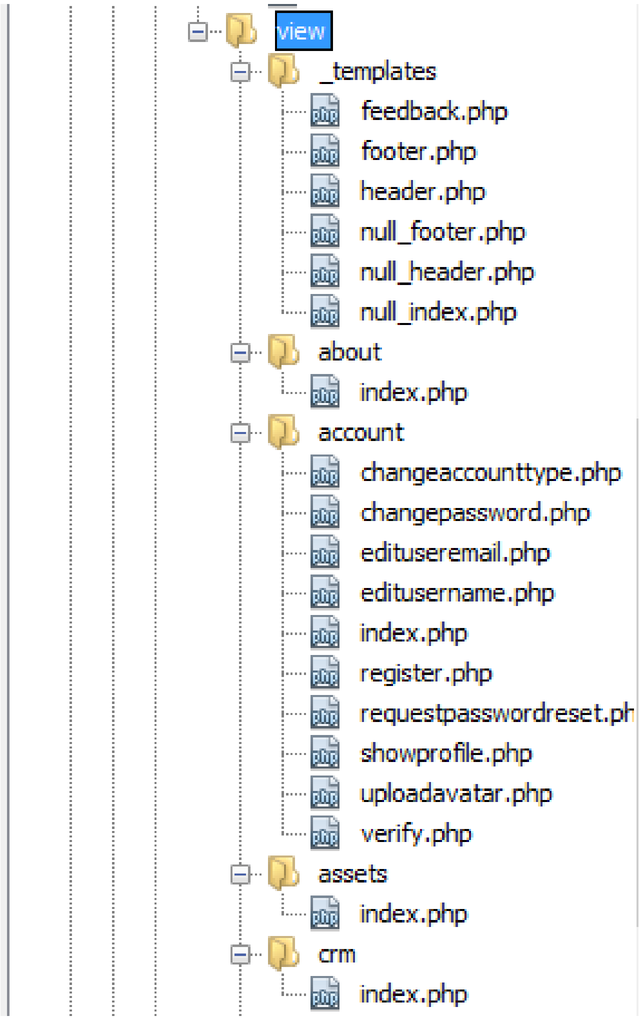


```php
<?php

/**
 * LoginModel
 *
 * Handles the user's login / logout / registration stuff
 */
use Gregwar\Captcha\CaptchaBuilder;

class LoginModel
{

    public function __construct($db)
    {
        try {
            $this->db = $db;
        } catch (PDOException $e) {
            exit('Database connection could not be established.');
        }
    }

    /**
     * Login process (for DEFAULT user accounts).
     * @return bool success state
     */
    public function login() {

        // ALGORITHM FROM PHP-LOGIN
        // we do negative-first checks here
        if (!isset($_POST['user_name']) OR empty($_POST['user_name'])) {
            $_SESSION["feedback_negative"][] = FEEDBACK_USERNAME_FIELD_EMP
            return false;
        }
        if (!isset($_POST['user_password']) OR empty($_POST['user_password
            $_SESSION["feedback_negative"][] = FEEDBACK_PASSWORD_FIELD_EMP
            return false;
        }

        $query = $this->db->prepare("SELECT * FROM users WHERE (user_name
```

The login_model.php file in Model directory

The View

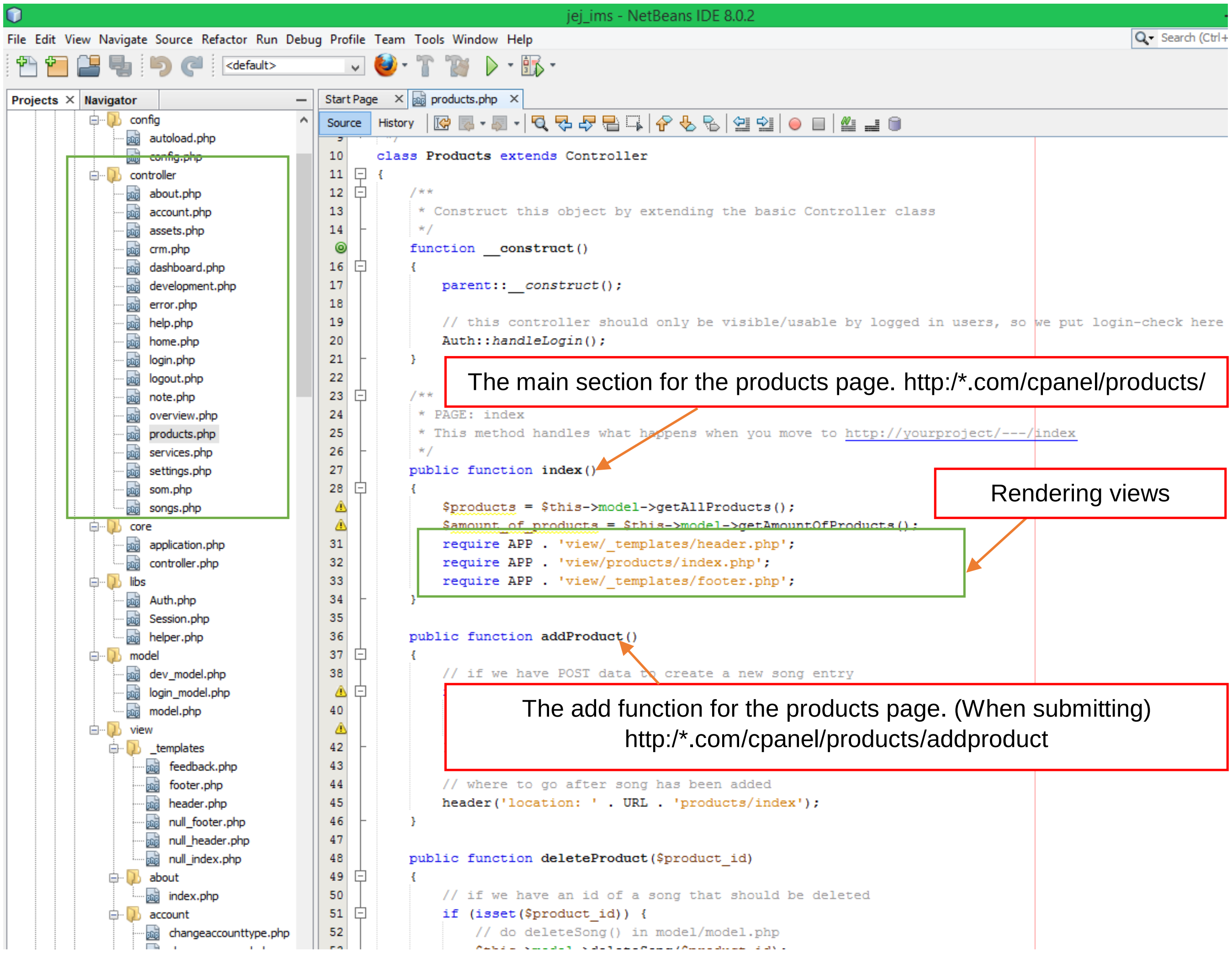
When creating a view, you must finish at least one page with 3 files, the header, content, and the footer. To make it cleaner, the author of this application prefer that make the default header and footer in _templates in Views directory. Here’s the sample.



```
view
    _templates
        feedback.php
        footer.php
        header.php
        null_footer.php
        null_header.php
        null_index.php
    about
        index.php
    account
        changeaccounttype.php
        changepassword.php
        edituseremail.php
        editusername.php
        index.php
        register.php
        requestpasswordreset.ph
        showprofile.php
        uploadavatar.php
        verify.php
    assets
        index.php
    crm
        index.php
```

The Controller

The controller handles the logic (Input/output of the user such as submitting forms) and Authentications of the Application. (NOTE: The model does not connected to the Controller and View based on the Microsoft's MVC Concept.). In this application, the developer must encode the "public function index()" which is it's an default index of the page. But if you want to create a sub-page, you can create a public function in current controller you've been editing, but this time, the name of the function must be also in the sub-page. Here's the example.



```
class Products extends Controller
{
    /**
     * Construct this object by extending the basic Controller class
     */
    function __construct()
    {
        parent::__construct();

        // this controller should only be visible/usable by logged in users, so we put login-check here
        Auth::handleLogin();
    }

    /**
     * PAGE: index
     * This method handles what happens when you move to http://yourproject/---/index
     */
    public function index()
    {
        $products = $this->model->getAllProducts();
        $amount_of_products = $this->model->getAmountOfProducts();
        require APP . 'view/_templates/header.php';
        require APP . 'view/products/index.php';
        require APP . 'view/_templates/footer.php';
    }

    public function addProduct()
    {
        // if we have POST data to create a new song entry


        // where to go after song has been added
        header('location: ' . URL . 'products/index');
    }

    public function deleteProduct($product_id)
    {
        // if we have an id of a song that should be deleted
        if (isset($product_id)) {
            // do deleteSong() in model/model.php
            $this->model->deleteSong($product_id);
```

The main section for the products page. http:/*.com/cpanel/products/

Rendering views

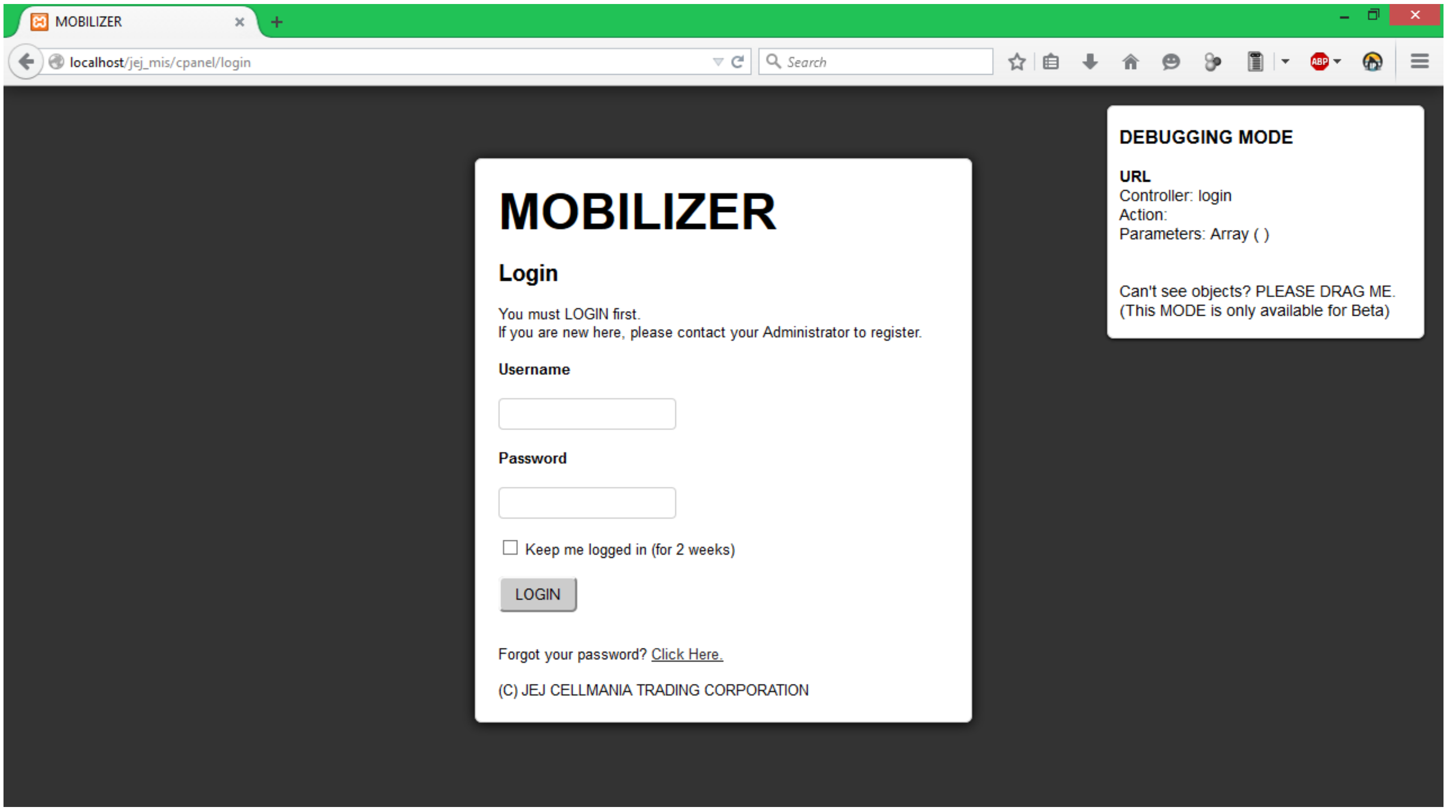
The add function for the products page. (When submitting) http:/*.com/cpanel/products/addproduct

Conclusion

Open-Source projects might not be good but there’s no big deal about it. Still, the technology we have today especially on the web/internet would be the way to improve of our lives and also to the industry. We are hoping that this might be using in the future for the next generations.

Appendix

Running in Windows 8.1 64-bit



DEBUGGING MODE

URL
Controller: login
Action:
Parameters: Array ( )

Can't see objects? PLEASE DRAG ME.
(This MODE is only available for Beta)

MOBILIZER

Login

You must LOGIN first.
If you are new here, please contact your Administrator to register.

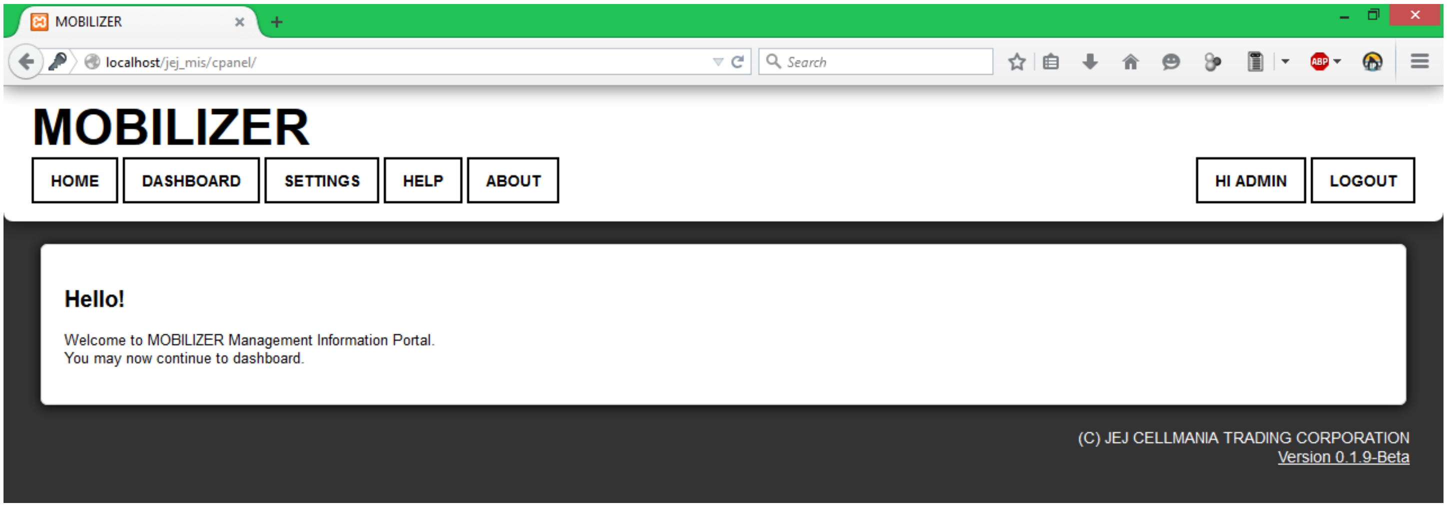
Username

Password

☐ Keep me logged in (for 2 weeks)

Forgot your password? Click Here.

(C) JEJ CELLMANIA TRADING CORPORATION

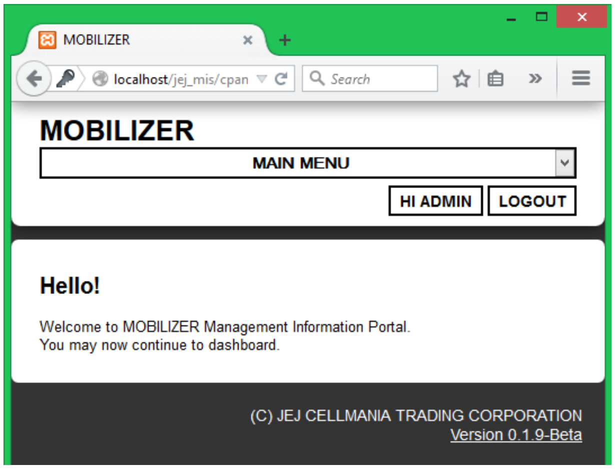


Hello!

Welcome to MOBILIZER Management Information Portal.
You may now continue to dashboard.

(C) JEJ CELLMANIA TRADING CORPORATION
Version 0.1.9-Beta

Desktop Version



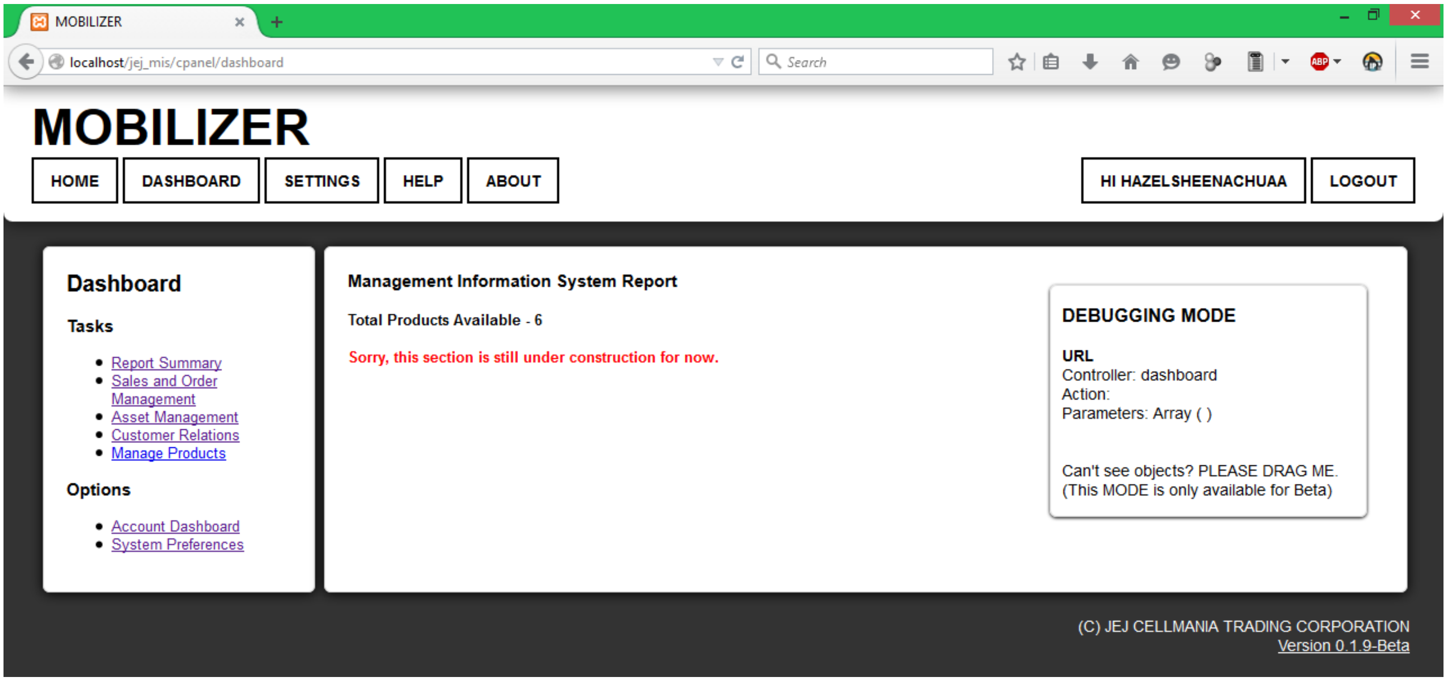
MOBILIZER

Hello!

Welcome to MOBILIZER Management Information Portal.
You may now continue to dashboard.

(C) JEJ CELLMANIA TRADING CORPORATION
Version 0.1.9-Beta

Mobile Version



MOBILIZER

Dashboard

Tasks

- Report Summary
- Sales and Order Management
- Asset Management
- Customer Relations
- Manage Products

Options

- Account Dashboard
- System Preferences

Management Information System Report

Total Products Available - 6

Sorry, this section is still under construction for now.

DEBUGGING MODE

URL
Controller: dashboard
Action:
Parameters: Array ( )

Can't see objects? PLEASE DRAG ME.
(This MODE is only available for Beta)

(C) JEJ CELLMANIA TRADING CORPORATION
Version 0.1.9-Beta

More Information:

jejMIS Repository Site: https://github.com/jccultima123/jej_mis