**CST-256 Activity 1 Guide**

Contents

[Part 1: Laravel Tools Installation 2](#_Toc528659233)

[Part 2: Laravel Routes and Plain Controller 7](#_Toc528659234)

This activity has multiple parts. All parts must be completed prior to documentation submission.

## Part 1: Laravel Tools Installation

**Overview**

In this activity, students will set up Eclipse PHP IDE for Laravel development and investigate the project scaffolding that is created for Laravel Projects.

**Execution**

Execute this activity according to the following guidelines:

1. Download Composer (Optional):
   1. This is an optional step. If you do not install Composer, ensure that the 'Download Composer per project' setting is selected in Eclipse Preferences->PHP->Tools->Composer.
   2. If desired, install a global version of Composer as follows:
      1. Go to http://getcomposer.org.
      2. Go to the Getting Started menu.
      3. Follow the instructions for downloading and installing composer. Make sure composer.phar is located in your path.
2. Set up a PHP Runtime Configuration:
   1. Start MAMP and get the current PHP version and location of the current INI file.
   2. Go to Eclipse Preferences.
   3. Select the Installed PHPs option under the PHP section.
   4. Click the Add button.
   5. Enter a name of MAMP.
   6. Browse to the [MAMP\_HOME]/bin/php/[version of PHP]/bin/php (or php.exe Windows).
   7. Browse to the [MAMP\_HOME]/bin/php/[version of PHP]/conf/php.ini.
   8. Click the Debugger tab. Select XDebug from the debugger dropdown list.
      1. NOTE: If you get a warning that Xdebug is not installed, follow the configuration from steps 4e below.
   9. Click the Finish button. Click the Apply and Close button.
3. Create default Laravel Project:
   1. File->New->PHP Project from Composer Package.
   2. Name your Project and select the "Use project specific settings" to set the desired version of PHP. Click Next.
   3. Enter laravel/laravel as the find package and click the Select button for the laravel/laravel package. Select highest version of 5.x of Laravel. Click Finish.
   4. Wait for the entire project and its dependencies to be downloaded. This will take a few minutes to complete.
      1. NOTE: If you get an error that Open SSL cannot be loaded, copy the active php.ini file to the location of the Configuration File Path specified in phpinfo. Delete the failed Laravel Project and repeat step 3a again.
   5. Select the Eclipse Project->Clean menu option to ensure all "build issues" are resolved.
4. Set up MAMP Auto-deploy, XDebug, and Arisan CLI Features:
   1. Set up Auto-Deployment (optional but highly recommended):
      1. Install Java Development Tools to get Ant Builder:
         1. Select the Eclipse Help->Install New Software menu options.
         2. Select the Work with All Available Sites, enter Java Development Tools in the search, check the Eclipse Java Development Tools, click all default options to install.
         3. When prompted, restart Eclipse.
      2. Set up Ant Build Files:
         1. Copy the sample build.xml and build.properties to root of project.
         2. In build properties set the MAMP htdocs Location in the apache.htdocs.dir variable. For Windows, make sure to use // for a file path separator and not the / character.
         3. In build properties, set the MAMP App Name in the web.root.dir variable.
      3. Set up Ant Builder for Project:
         1. In Project Properties, select Builders, add new, select Ant Builder:
         2. Select the Main tab. Under Buildfile set to the build.xml in the root of the project. Under Base Directory set to root of the Project.
         3. Select Targets tab. Click the Set Targets button for Auto Build and simply click the OK button in the Set Targets dialog. Click the Set Targets button for During Clean and simply click the OK button in the Set Targets dialog.
         4. Click the Apply button. Click the OK button to finish the configuration. Click the Apply and Close button.
         5. Validate that the entire project by selecting the Project->Clean menu or by making any incremental file changes to the project. Ensure that the entire project is automatically deployed to the proper app directory within the MAMP htdocs directory.
         6. Start MAMP. Open the following URL in your browser: http://localhost:[MAMP Port]/[App Name]/public/. Validate the default Laravel Project page is displayed.
   2. Setup Debugger:
      1. Enable the Zend Debug Plugin in the active php.ini file.
         1. If there is no [xdebug] section add one to the bottom of the php ini file.
         2. Add or uncomment zend\_extension property with a value set to the full path to the Zend Extension (php\_xdebug.dll on Windows or xdebug.so on OSX). The Zend Extension can be found under your MAMP install in bin/php/php[version]/php/extensions (on OSX) or bin/php/php[version]/php/ext (on Windows).
         3. Enable the debugger by adding a xdebug.remote\_enable property with a value set to 1.
         4. See the Figure 1 below.
         5. Restart MAMP.
         6. Validate that the phpinfo has an xdebug section.
         7. See Figure 2.

**Figure 1**

Example INI xdebug configuration settings for Windows and OSx:

[xdebug]

xdebug.remote\_enable=1

zend\_extension="C:\MAMP\bin\php\[Version of PHP]\ext\php\_xdebug.dll"

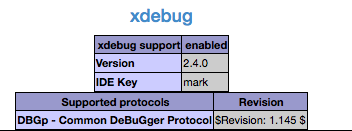
[xdebug]

xdebug.remote\_enable=1

zend\_extension="/Applications/MAMP/bin/php/[Version of PHP]/lib/php/extensions/no-debug-non-zts-20151012/xdebug.so"

**Figure 2**

Example screenshot of xdebug section in phpinfo when XDebug is enabled:



* 1. Test Artisan CLI:
     1. Open a Terminal Window in Eclipse by selecting the Window->Show View->Other and then filter on Terminal. Select Terminal and click the Open button.
     2. Click on the Terminal Tab in the Eclipse Views pane. Click the Open Terminal icon.
     3. Navigate to the location of your Project in your Workspace directory.
     4. Test artisan by entering the following command: php artisan env and validate the local environment is displayed (you must enter an absolute path to php.exe if php is not in your current environment path).
     5. Validate that a message 'Current application environment: local' is displayed.

1. Validate default Laravel Project:
   1. Start MAMP. Open the following URL in your browser: http://localhost:[MAMP Port]/[App Name]/public/. Validate the default Laravel Project page is displayed.
   2. Take a screenshot of the default Laravel Page (including the address bar of your browser).
   3. Select Debug, Debug Configurations, Debug as PHP Web Application, and click the New icon:
      1. Name your Configuration.
      2. Under PHP Server click the Configure button.
      3. Under the Server Tab set up the Base URL to the host and port to match MAMP and set up the Document Root to the MAMP htdocs directory.
      4. Under the Debugger Tab select the XDebug type from the list.
      5. Under the Path Mapping Tab, click add. Set the Path on Server to the full path the application in the MAMP htdocs directory (i.e. MAMP/htdocs/[Project Name]) and set up the Path in Workspace to the root of the project. Click OK.
      6. Click the Finish button.
      7. Set File entry to the Controller.php file located in the app/http/controllers.
      8. Uncheck the URL Auto Generate option. Fix up the URL [Project Name]/public/. Click the Apply button.
      9. Click the Debug button.
      10. The Debugger should break on the first line in index.php. Click the Resume icon to continue execution. See note below regarding setting the break on file line of code setting.
2. Add PDT Extensions Plugin (optional but highly recommended):
   1. Go to Eclipse Help->Add New Software menu options.
   2. Click the Manage button and add the following site http://p2-dev.pdt-extensions.org to the list of update sites. Click the Apply and Close button.
   3. Select the PDT Extension update site from the list.
   4. Check the PDT Extensions and click the default buttons (accepting the warning) to finish the plugin installation. Restart Eclipse when prompted.
   5. Refer to http://p2.pdt-extensions.org/phpfeatures.html for the list of features: Code Formatter, Undefined Variable warning, etc.
   6. Set up Code Formatter by going to Eclipse Preferences->PHP->Code Style->Formatter. Set up the Code Style per your liking.
3. Fix up the Laravel Default 'public' URI:
   1. Copy and paste the server.php file located in the root of the project to index.php. Delete the server.php file from your project.
   2. Show hidden files in Eclipse package explorer by selecting the little down arrow in the upper right corner of the Project Explorer. From the dropdown menu select Filters and Customizations. Uncheck the \* resources option. Click the OK button.
   3. Copy the (hidden) .htaccess file from the public directory to the root of the project. Delete the file from the public directory.
   4. Deploy your changes to MAMP.
   5. Hide the by checking the \* resources option from the Filters and Customization package explorer.
   6. Open the following URL in your browser: http://localhost:[MAMP Port]/[App Name] and validate the default Laravel Page is displayed.
   7. Take a screenshot of the default Laravel Page (including the address bar of your browser).
4. Complete the "Project Structure Tutorial" and "Project Configuration Overview Tutorial,'" located in the required readings.

NOTE: If you do not want the Debugger to break at the first line of code:

1. Go to Eclipse Settings->PHP->Debug options and uncheck the 'Break at first line' option.
2. Go to Eclipse Debug Configurations, select the Debugger tab, and uncheck the 'Break at first line' option.

NOTE: By default your Laravel application will run using the following URL convention:

http://localhost:[WAMP/MAMP Port]/[App Name]/public

For example: http://localhost:8888/PlayLaravel/public/

**Documentation**

All documentation will be submitted at the end of the activity to the learning management system. Ensure documentation of the following:

1. Create a Project Report using a GCU standard Project Header Page/Information containing your name, course, assignment name, and date. Make sure to include any screenshots demonstrating working code and write-ups as instructed.
2. After first creating a Laravel Project, run the default project and attach a screenshot of your working application in your Project Report.
3. After updating the Laravel Project to remove the /public URI, attach a screenshot of your working application in your Project Report.
4. After finishing the Laravel Project Structure and Configuration Tutorials, outline the basic structure of a Laravel project discussing all the relevant folders and their general purpose. In addition, document what file and location is used for general application configuration and how databases are configured in Laravel. This should be 50-100 words and documented in your Project Report.

## Part 2: Laravel Routes and Plain Controller

**Overview**

In this activity, students will explore Laravel Routes and a Plain Controller.

**Execution**

Execute this activity according to the following guidelines:

1. Ensure that Part 1 is 100% complete.
2. Create a new Route that renders a Text response:
   1. Go to the /routes directory and open the web.php files. This is where all your Routes will be defined.
   2. Create a new Route '/hello' that renders the text "Hello World" and test this URI.
   3. Take a screenshot of your result.
3. Create a new Route that renders a View response:
   1. Go to the /resources/views directory and create a very simple PHP page, helloworld.php, that simply echoes the string “Hello World” as a response.
   2. Create a new Route '/helloworld' that renders "Hello World" via helloworld.php. View and test this URI.
   3. Take a screenshot of your result.
4. Create a Plain Controller:
   1. In Eclipse PHP, select the Terminal View (if this is not visible go to the Window->Show View->Other->Terminal menu options). Open a Terminal session by clicking on the Open Terminal icon. This will open the Command Line Tools Console where you can run Laravel Artisan commands. In the Terminal navigate thru the file system to reach the root directory of your project. One of the Artisan commands can be used to create a Laravel Controller class.
   2. Run the following Artisan command: artisan make:controller TestController
   3. Go to the /app/Http/Controllers directory and verify that a TestController.php was created.
   4. Add a method named test() to the TestController with signature of public function test() and that returns a text response of "Hello World from Test Controller".
   5. Create a new Route ‘/test’ that maps to the TestController test():
      1. Route::get('/test', 'TestController@test');
      2. View and test this URI.
      3. Take a screenshot of your result.
   6. Add a new method test2() to the TestController that returns the helloworld.php View:
      1. test2() method return code: return view('helloworld');
      2. Update the '/test' Route to map to the test2().
      3. View and test this updated URI.
      4. Take a screenshot of your result.
5. Research and Design:
   1. Review the Registration and Login pages you created in CST-236. Browse the Internet and review several registration pages for professional networking sites, similar to the one needed in this course.
   2. Draw and provide a UML class diagram illustrating the elements and methods for a model and controller class.
   3. Draw and provide a logical diagram illustrating how the MVC design pattern would be incorporated into the design of your application.

**Documentation**

Submit the following to the learning management system:

A Project Report to include:

1. A screenshot of your working application in your Project Report after running the default project.
2. A screenshot of your working application in your Project Report after updating the Laravel Project to remove the /public URI.
3. An outline of the basic structure of the Laravel project discussing all the relevant folders and their general purpose. In 50-100 words explain what file and location is used for general application configuration and how databases are configured in Laravel.
4. Screenshots for the following URIs:
   1. /hello
   2. /helloworld
   3. /test (with TestController.test())
   4. /test (with TestController.test2())
5. A drawn block diagram for each of the components (M, V, C) of the MVC design pattern. Map each component of the MVC design pattern to its equivalent component within the Laravel Framework and the code you wrote in Step 4f of Part 2: Laravel Routes and Plain Controller.
6. UML Class Diagram of a Login Module and Registration Module.
7. Logical Diagram of the MVC Design Pattern for a Login Module and Registration Module.