**CST-256 How to Guides**

Contents

[How to Use Namespaces in Laravel 2](#_Toc528659806)

[How to Upload a Laravel Project to Azure 4](#_Toc528659807)

[How to Set Up Application Path in Laravel 6](#_Toc528659808)

[How to Do a Redirect in Laravel 7](#_Toc528659809)

[How to Configure DB in Laravel 8](#_Toc528659810)

[Install and Use Postman to Test REST APIs 9](#_Toc528659811)

[Install and Configure Composer in a Laravel Project 11](#_Toc528659812)

# How to Use Namespaces in Laravel

Internally, Laravel uses the PSR-4 Autoloader, which is a specification for auto-loading classes based on file paths and the file system. A brief introduction to PSR-4 can be found on the [PSR-4: Autoloader](https://www.php-fig.org/psr/psr-4/) page of the PHP Framework Interop Group website.

A Composer-based Laravel project will use Composer as its auto-loader. The auto-loader configuration can be found in your Laravel project by inspecting the *composer.json* file located in the root of your Laravel project. The actual auto-loader configuration, found within the *autoload* section and the psr-4 configuration parameter, simply defines a mapping between a namespace and a file path. By default, this maps the *App* namespace to the *app/* folder within your Laravel project, which is the root directory for all application code. A brief introduction to PHP namespaces can be found in the following articles:

[A Brief Introduction to PHP Namespacing](https://mattstauffer.com/blog/a-brief-introduction-to-php-namespacing/)

[How to Use PHP Namespaces, Part 1: The Basics](https://www.sitepoint.com/php-53-namespaces-basics/)

[Using Namespaces: Basics](http://php.net/manual/en/language.namespaces.basics.php)

What does this mean when you are developing code for Laravel?

1. You must make sure that you prefix all of your namespace declarations at the top of your class file with *App/* because this is how the Composer auto-loader is configured. The only caveat to this rule is if you truly intend to use relative namespaces as discussed in the [PHP Documentation](http://php.net/manual/en/language.namespaces.basics.php).
2. You must make sure the namespace declaration at the top of your class file, including case, matches the folder names and structure within your Laravel project.
3. You must make sure in your classes that have dependences on classes outside the classes own namespace you resolve these dependencies with a *using* statements with proper paths and proper spelling, including the case of those namespaces.

A brief example is shown in the screenshots below.

If you get an exception from your IDE indicating it cannot resolve the namespace or you get a class cannot be loaded exception at runtime please check the before mentioned rules for using namespace in a Laravel project.

You will also need to be aware of the Global Namespace where many of the classes for the Database and Exception handling (i.e., PDO and Exception respectively) can be located. Global Namespace are discussed in [How to Use PHP Namespaces, Part 1: The Basics](https://www.sitepoint.com/php-53-namespaces-basics/).



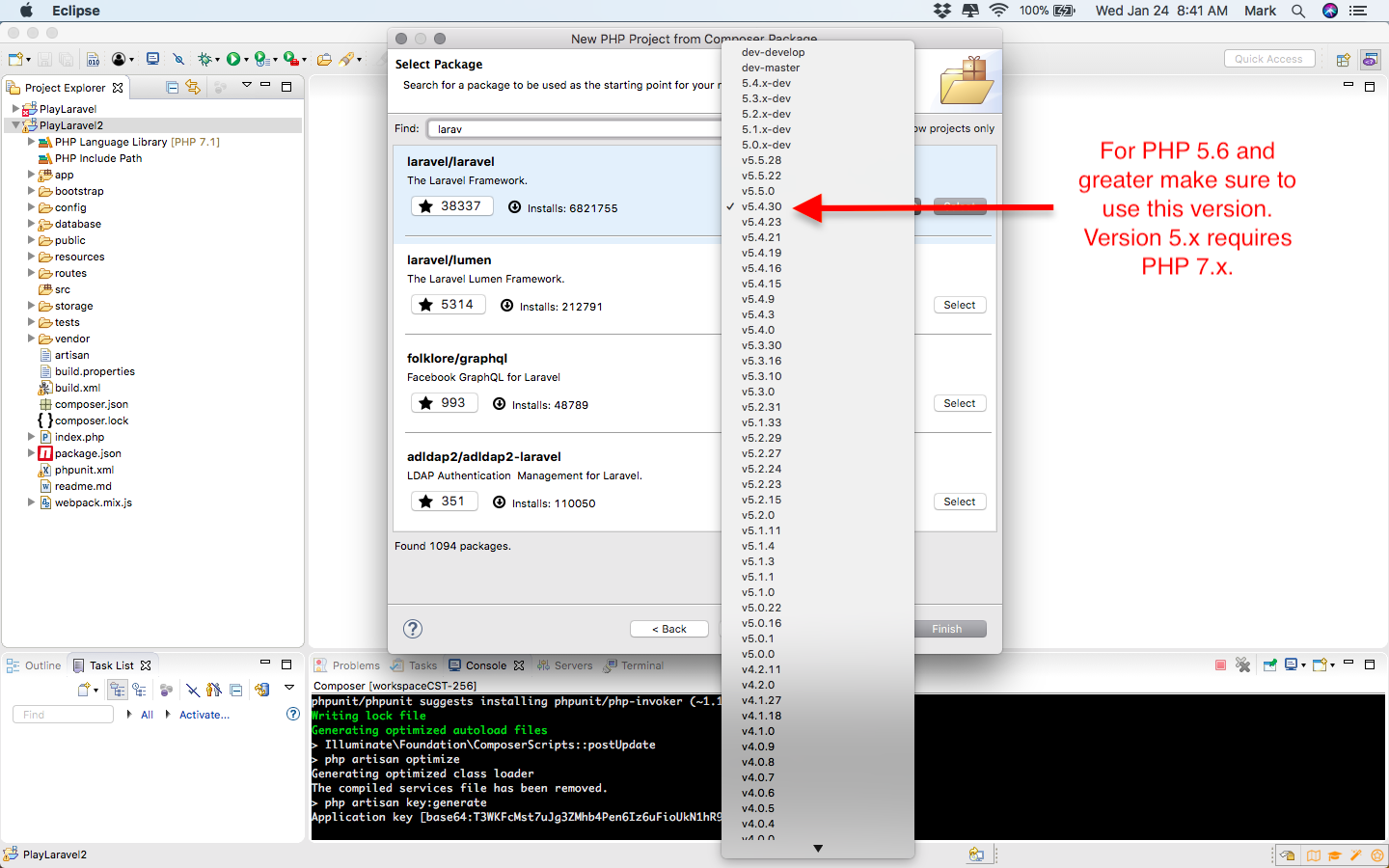


# How to Upload a Laravel Project to Azure

Laravel Projects contains a large number of visible and hidden files that all must be uploaded to Azure in order to get your application hosted properly. The following are a few guidelines for building and uploading your Laravel Project. **Note:** Refer to the various resources available within the Topic 1 required readings to provide assistance.

1. Make sure you have built your Laravel Project with the right version to match the version of PHP you are using:
   1. The version of PHP can be set in your Applications Configuration section in Azure.
   2. Use v5.4.x of the default Laravel Project if you are using PHP 5.6 or greater. v5.5.x of the Laravel Project requires PHP 7.x or greater. See Figure 1.
2. You can upload and configure your Laravel Project as follows:
   1. Create a default PHP application in Azure:
      1. Click the '+ Create a resource' icon from the Azure Portal and search for PHP.
      2. Select the PHP Starter Kit Application.
      3. Open your application from your Dashboard.
   2. Deploy your Application:
      1. For a Laravel copy the web.config from the public directory to the root of your project
      2. Zip up your PHP project into a file named [appname].zip
      3. Under the Development Tools section click the Advanced Tools icon, select the Go link, and select the Tools->Zip Push Deploy menu
      4. Delete the Azure created default files from the application (if they exist)
      5. Drag and drop your zip file onto the page
   3. Deploy your Database in Azure:
      1. Under the Settings section click the MySQL In App icon and click the Manage icon to open phpMyAdmin.
      2. Import your Database DLL.
      3. Under the Development Tools section click the Console icon.
      4. In the Console enter 'type D:\home\data\mysql\MYSQLCONNSTR\_localdb.txt' to get your MySQL Connection Properties.
      5. Update your MySQL Database Connection properties in your application (note your hostname will need to be formatted as hostname:port).
   4. Test your application using your Azure Domain name:
      1. Go to https://[app name].azurewebsites.net.

Figure 1



# How to Set Up Application Path in Laravel

By default Laravel configures their framework such that the application requires a /public URI within the Application URL. The following instructions can be used to eliminate the public URI from within your application URL:

1. Set up your operating system to View Hidden Files:
   1. OSX:
      1. Open the Terminal from your Applications Folder.
      2. Run the following commands:
         1. defaults write com.apple.finder AppleShowAllFiles YES
         2. killall Finder
         3. To restore run the above commands with NO in the argument
   2. Windows 10:
      1. From the Start Menu open File Explorer.
      2. Select the View Menu. Check the 'Hidden items' option.
   3. Windows 7:
      1. From the Start Menu open Control Panel.
      2. Select Folder Options.
      3. Select the View tab.
      4. Under 'Hidden files and folders' select the 'Show hidden files' option.
2. Open Windows File Explore or OSX Finder.
3. Navigate to your Laravel Project directory within your PhpstormProjects directory.
4. Set up Non 'public' URI:
   1. Rename the server.php file located in the root of your project to index.php.
   2. Copy (the hidden) file .htaccess file from the public directory within your project to the root of your project. See instructions below if hidden files are not displayed in your operating system.
   3. Repeat these steps for both your Project and Deployed applications.

# How to Do a Redirect in Laravel

If you are not using the Laravel Security Framework you will need to validate your security token maintained in the HTTP Session and in various pages redirect a user to the Login View in the event they are not authenticated. Redirect in Laravel can be done by updating your desired URIs to secure in your routes file as follows:

Route::get('/profile', function() {

if(Session::get("user") == null) <= Your security token check and logic here

{

return view('login'); <= Redirect to Login View

}

return view('memberProfile'); <= Continue to View as you had already defined

});

# How to Configure DB in Laravel

Generally it is a bad practice to hard code application configuration data or use a home grown configuration utility. Laravel provides a means to configure your application (via the app.php file in the config directory of your project) and also configure your database (via the database.ph file in the config directory in your project). The following instructions can be used to configure your database and access that configuration from your PHP code in a Laravel application:

1. Navigate using File Explore or Finder to your Laravel project.
2. Update .env configuration file:
   1. Locate the (hidden) .env file located in the root of your project.
   2. Open this file with PhpStorm or a safe text editor.
   3. Comment out (using a ;) all the environment settings that start with 'DB\_'.
   4. Save the file.
   5. Deploy the file to MAMP.
3. Update the database configuration file:
   1. Locate the database.php file located in the config directory in the root of your project.
   2. Open this file with PhpStorm or a safe text editor.
   3. Locate the 'connections' settings.
   4. Locate the 'mysql' settings with 'connections'.
   5. Update the 'host', 'port', 'database', 'username', and 'password' configuration to your desired settings.
   6. Save the file.
   7. Deploy the file to MAMP.
4. Restart your MAMP if it was running.
5. Access your Configuration in PHP Code:
   1. Access a database configuration setting using the Config class in your PHP class code:

config("database.connections.mysql.host");

# Install and Use Postman to Test REST APIs

1. Where do I get Postman?

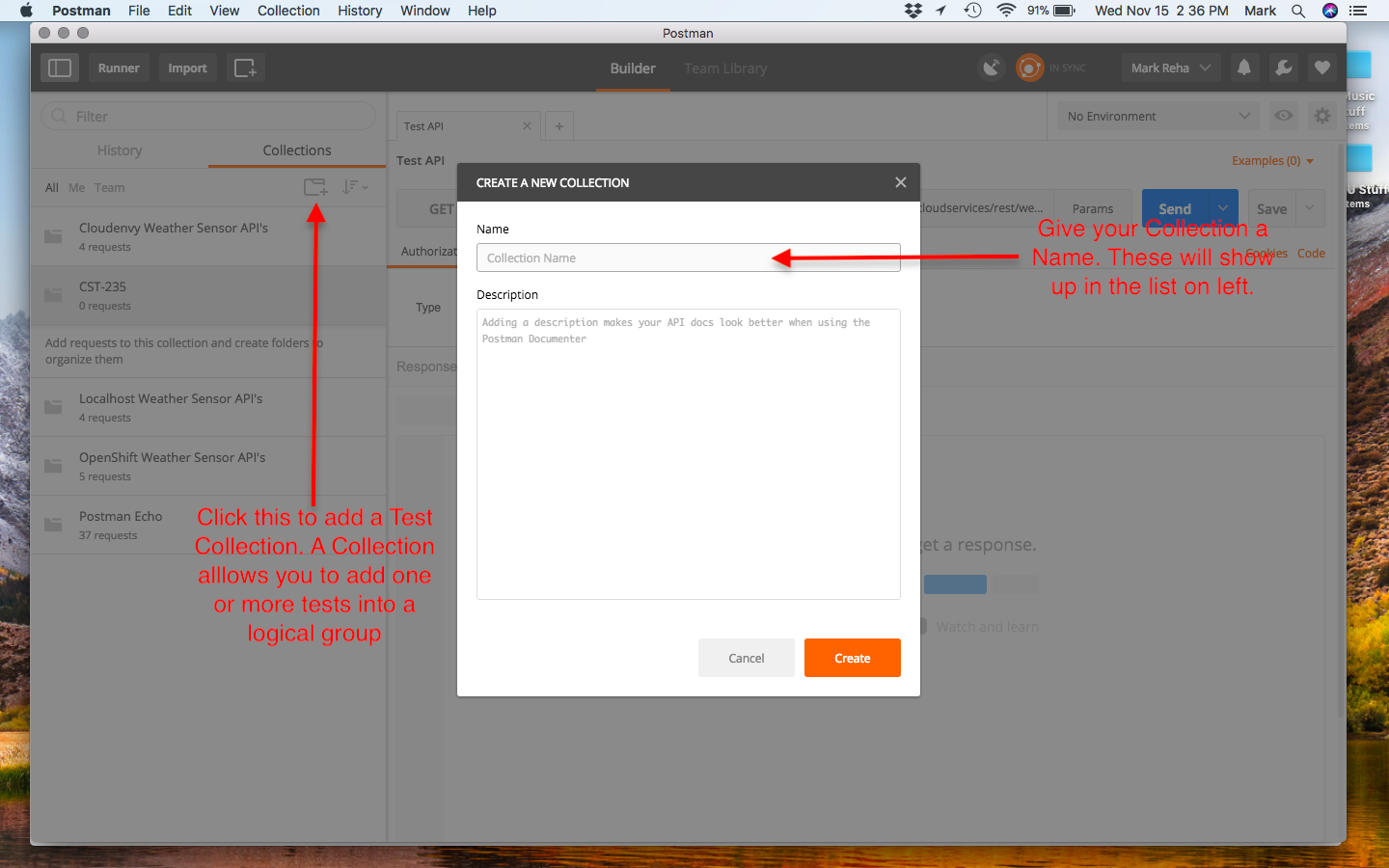
Postman can be downloaded at <https://www.getpostman.com>.

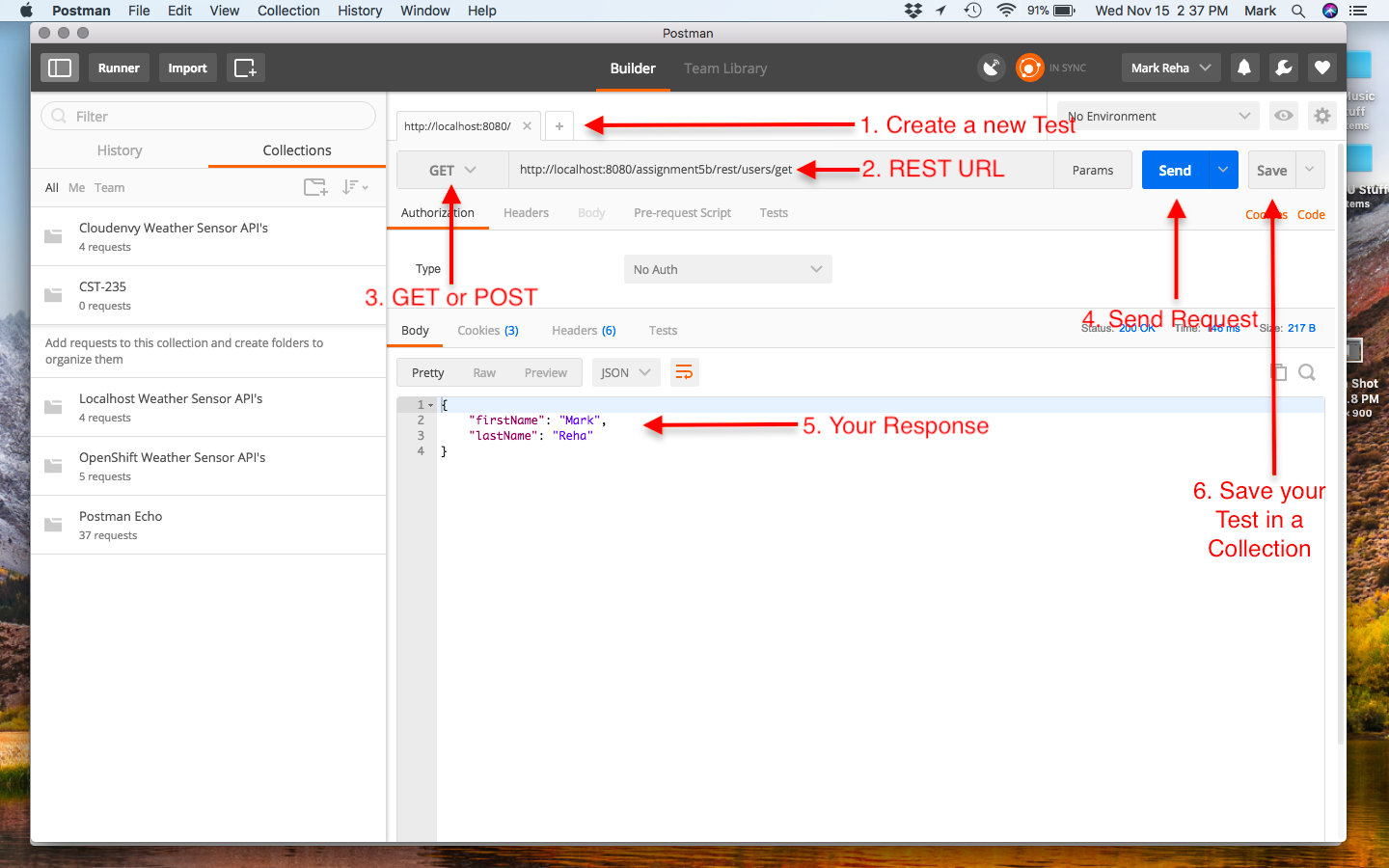
1. How do I make a GET Request with Postman?

You can always use your browser for sending GET requests. But this is error prone and you have to write down all your URLs and test scenarios for all your APIs. Postman is a very nice desktop tool for testing your REST APIs. This tool allows you to set up Collections of API Tests and supports making HTTP GET and POST requests. If you sign up for an account with Postman, you can even sync your Collections and Tests across development environments and team members. Postman also allows you to create unit tests and validate that you get the proper responses. This capability is available in the Tests tab of the UI. Reference the Postman home page for more information.

1. Select the GET operation and the URL (with optional HTTP parameters).
2. Click the Send button and observe the Raw response.

See examples below.



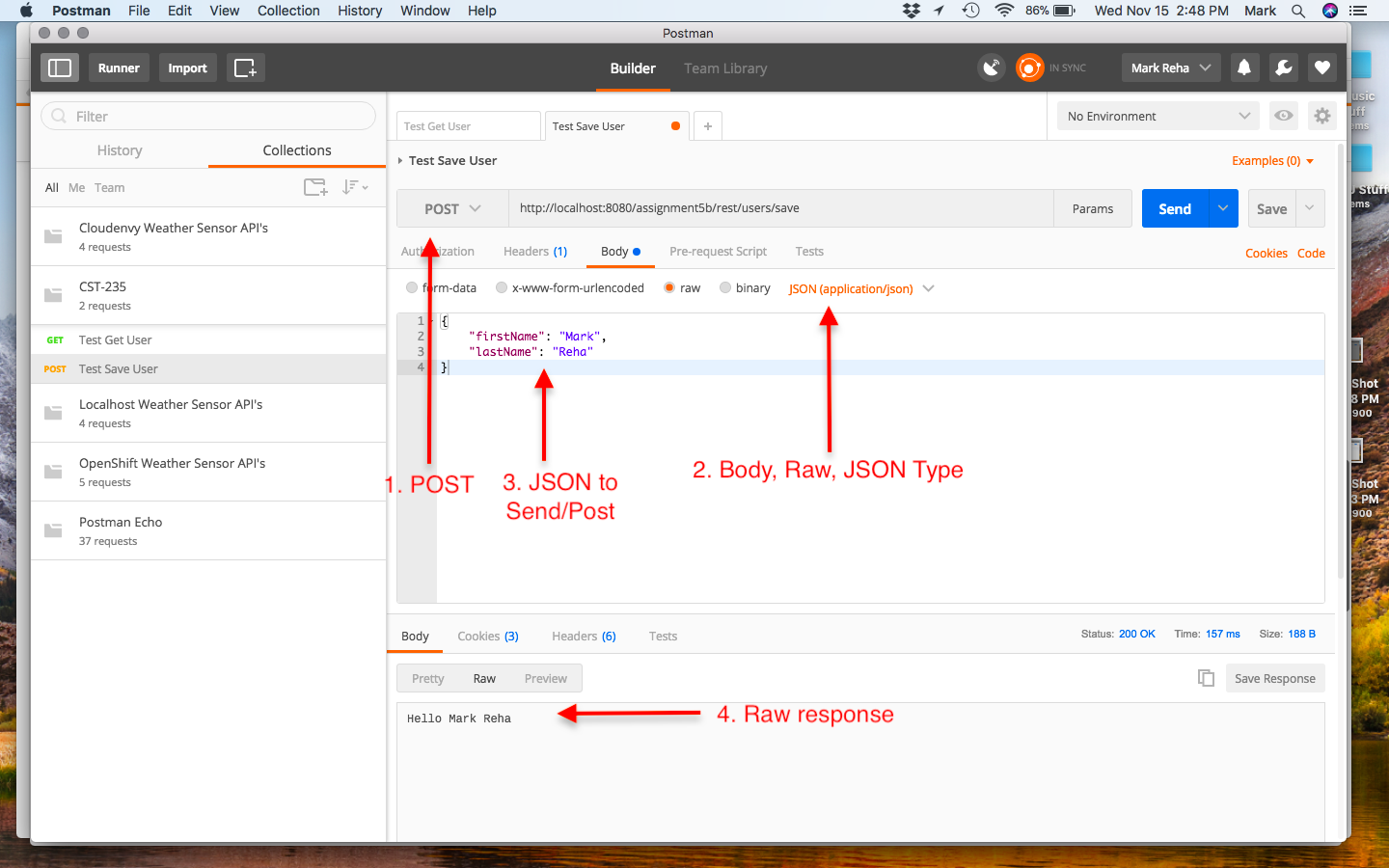


1. How do I make a POST Request with Postman?

Postman easily supports making HTTP POST requests. This cannot be done straight up using your browser or without writing a client page. It should be noted that if you are going to POST JSON data, as shown below, you also need to click on the Headers table and add the following key value pair: key set to Content-type and value set to application/json.

1. Select the POST operation and the URL.
2. Set the Body Tab to Raw.
3. Set the Body content to the JSON to POST.
4. Set the Header tab to Content-type and application/json.
5. Click the Send button and observe the Raw response.

See example below.



# Install and Configure Composer in a Laravel Project

1. What is Composer?

Composer is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you. More information can be obtained at <https://getcomposer.org>.

1. Where do I get information on Composer?

See <http://guzzle.readthedocs.io/en/latest/overview.html>

1. How do I add Composer to a PHP Project in Eclipse?

Update composer.json file in project located in the root of the project. Under the requires section add entry "guzzlehttp/guzzle": "~6.0". See image below.

Right click on project in Eclipse and select the Composter->Update Build Path menu options.

Run the following command from an Eclipse Terminal Session in the root of the project:

php ../composer.phar require guzzlehttp/guzzle:~6.0

Right click on the vendor folder and select the Refresh menu option. A new library named guzzlehttp should be listed.

