

# laravel

**tutorialspoint**  
SIMPLY EASY LEARNING

[www.tutorialspoint.com](http://www.tutorialspoint.com)



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

## About the Tutorial

---

Laravel is a powerful MVC PHP framework, designed for developers who need a simple and elegant toolkit to create full-featured web applications. Laravel was created by Taylor Otwell. This is a brief tutorial that explains the basics of Laravel framework.

## Audience

---

This tutorial will guide the developers and students who want to learn how to develop a website using Laravel. This tutorial is particularly meant for all those developers who have no prior experience of using Laravel.

## Prerequisites

---

Before you start proceeding with this tutorial, we make an assumption that you are familiar with HTML, Core PHP, and Advance PHP. We have used Laravel version 5.1 in all the examples.

## Copyright & Disclaimer

---

©Copyright 2016 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at [contact@tutorialspoint.com](mailto:contact@tutorialspoint.com)

## Table of Contents

---

About the Tutorial.....	i
Audience .....	i
Prerequisites .....	i
Copyright & Disclaimer.....	i
Table of Contents .....	ii
<b>1. LARAVEL – OVERVIEW .....</b>	<b>1</b>
Introduction .....	1
Laravel – Features .....	1
<b>2. LARAVEL – INSTALLATION.....</b>	<b>2</b>
<b>3. LARAVEL – APPLICATION STRUCTURE.....</b>	<b>4</b>
Root Directory.....	4
App Directory.....	5
<b>4. LARAVEL – CONFIGURATION .....</b>	<b>6</b>
Basic Configuration .....	6
Environmental Configuration .....	6
Database Configuration.....	7
Naming the Application.....	8
Maintenance Mode.....	8
<b>5. LARAVEL – ROUTING .....</b>	<b>10</b>
Basic Routing.....	10
Routing Parameters .....	13
<b>6. LARAVEL – MIDDLEWARE .....</b>	<b>16</b>
Define Middleware .....	16
Register Middleware .....	17

<b>Middleware Parameters</b> .....	<b>19</b>
<b>Terminable Middleware</b> .....	<b>22</b>
<b>7. LARAVEL – CONTROLLERS</b> .....	<b>27</b>
<b>Basic Controllers</b> .....	<b>27</b>
<b>Controller Middleware</b> .....	<b>28</b>
<b>Restful Resource Controllers</b> .....	<b>33</b>
<b>Implicit Controllers</b> .....	<b>35</b>
<b>Constructor Injection</b> .....	<b>38</b>
<b>Method Injection</b> .....	<b>39</b>
<b>8. LARAVEL – REQUEST</b> .....	<b>41</b>
<b>Retrieving the Request URI</b> .....	<b>41</b>
<b>Retrieving Input</b> .....	<b>43</b>
<b>9. LARAVEL – COOKIE</b> .....	<b>47</b>
<b>Creating Cookie</b> .....	<b>47</b>
<b>Retrieving Cookie</b> .....	<b>47</b>
<b>10. LARAVEL – RESPONSE</b> .....	<b>51</b>
<b>Basic Response</b> .....	<b>51</b>
<b>Attaching Headers</b> .....	<b>51</b>
<b>Attaching Cookies</b> .....	<b>52</b>
<b>JSON Response</b> .....	<b>53</b>
<b>11. LARAVEL – VIEWS</b> .....	<b>54</b>
<b>Understanding Views</b> .....	<b>54</b>
<b>Passing Data to Views</b> .....	<b>55</b>
<b>Sharing Data with all Views</b> .....	<b>55</b>
<b>Blade Templates</b> .....	<b>57</b>

12. LARAVEL — REDIRECTIONS.....	61
Redirecting to Named Routes.....	61
Redirecting to Controller Actions .....	62
13. LARAVEL — WORKING WITH DATABASE.....	64
Connecting to Database .....	64
Insert Records .....	64
Retrieve Records .....	67
Update Records.....	70
Delete Records .....	74
14. LARAVEL — ERRORS AND LOGGING .....	78
Errors .....	78
Logging.....	78
15. LARAVEL – FORMS.....	79
16. LARAVEL – LOCALIZATION .....	85
17. LARAVEL — SESSION .....	89
Accessing Session Data.....	89
Storing Session Data.....	89
Deleting Session Data.....	89
18. LARAVEL – VALIDATION .....	93
19. LARAVEL – FILE UPLOADING .....	98
20. LARAVEL – SENDING EMAIL.....	102
21. LARAVEL – AJAX.....	108
22. LARAVEL – ERROR HANDLING.....	111
HTTP Exceptions.....	111

Custom Error pages .....	111
23. LARAVEL – EVENT HANDLING .....	114
24. LARAVEL – FACADES .....	122
25. LARAVEL – SECURITY .....	128

# 1. Laravel – Overview

## Introduction

---

Laravel is an MVC framework with bundles, migrations, and Artisan CLI. Laravel offers a robust set of tools and an application architecture that incorporates many of the best features of frameworks like CodeIgniter, Yii, ASP.NET MVC, Ruby on Rails, Sinatra, and others.

Laravel is an Open Source framework. It has a very rich set of features which will boost the speed of Web Development. If you familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It will save a lot time if you are planning to develop a website from scratch. Not only that, the website built in Laravel is also secure. It prevents the various attacks that can take place on websites.

## Laravel – Features

---

Laravel offers the following key features:

- Modularity
- Testability
- Routing
- Configuration management
- Query builder and ORM (**O**bject **R**elational **M**apper)
- Schema builder, migrations, and seeding
- Template engine
- E-mailing
- Authentication
- Redis
- Queues
- Event and command bus

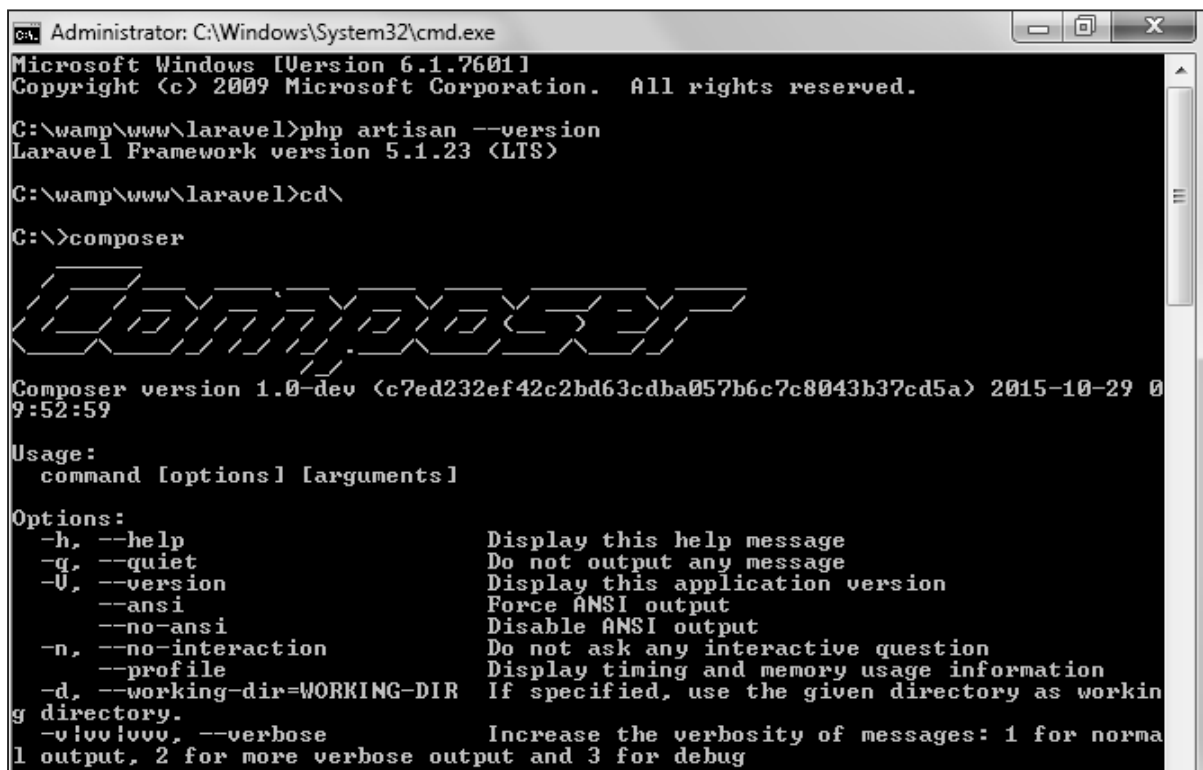
## 2. Laravel – Installation

For managing dependencies, Laravel uses **composer**. Make sure you have a Composer installed on your system before you install Laravel.

**Step 1:** Visit the following URL and download composer to install it on your system.

<https://getcomposer.org/download/>

**Step 2:** After the Composer is installed, check the installation by typing the Composer command in the command prompt as shown in the following screenshot.



```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\wamp\www\laravel>php artisan --version
Laravel Framework version 5.1.23 (LTS)

C:\wamp\www\laravel>cd\

C:\>composer

Composer version 1.0-dev (c7ed232ef42c2bd63cdba057b6c7c8043b37cd5a) 2015-10-29 09:52:59

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi                    Force ANSI output
  --no-ansi                 Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
  --profile                 Display timing and memory usage information
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  -vvvv, --verbose          Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
```

**Step 3:** Create a new directory anywhere in your system for your new Laravel project. After that, move to path where you have created the new directory and type the following command there to install Laravel.

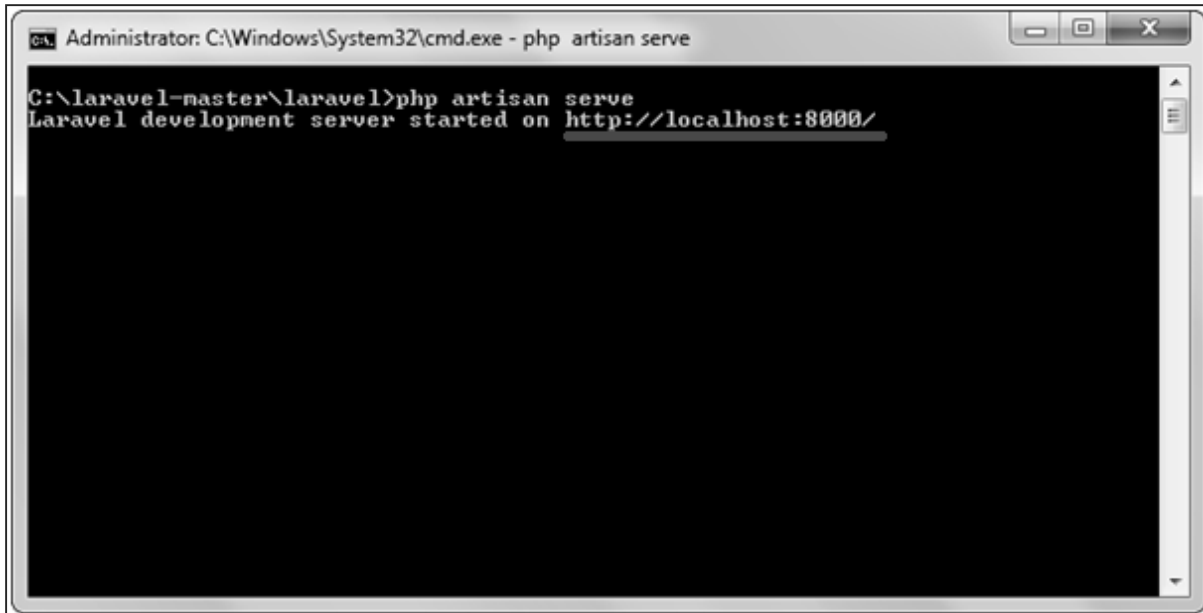
```
composer create-project laravel/laravel --prefer-dist
```

**Step 4:** The above command will install Laravel in the current directory. Start the Laravel service by executing the following command.

```
php artisan serve
```



**Step 5:** After executing the above command, you will see a screen as shown below:



```
Administrator: C:\Windows\System32\cmd.exe - php artisan serve
C:\laravel-master\laravel>php artisan serve
Laravel development server started on http://localhost:8000/
```

**Step 6:** Copy the URL underlined in gray in the above screenshot and open that URL in the browser. If you see the following screen, it implies Laravel has been installed successfully.



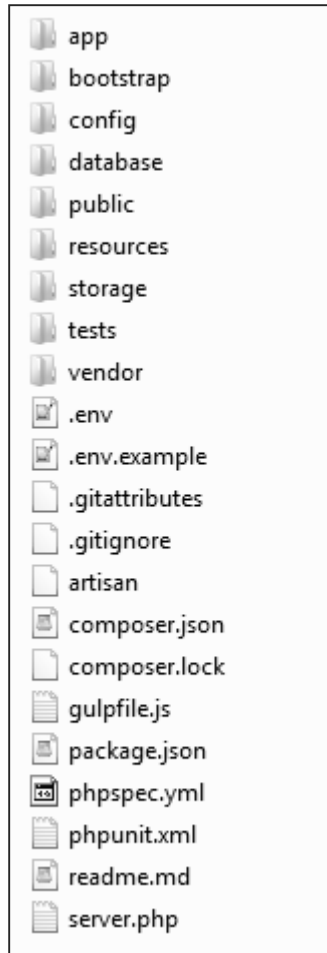
Laravel 5

# 3. Laravel – Application Structure

## Root Directory

---

The root directory of Laravel contains various folders and files as shown in the following figure.



- **app:** This directory contains the core code of the application.
- **bootstrap:** This directory contains the application bootstrapping script.
- **config:** This directory contains configuration files of application.
- **database:** This folder contains your database migration and seeds.
- **public:** This is the application's document root. It starts the Laravel application. It also contains the assets of the application like JavaScript, CSS, Images, etc.

- **resources:** This directory contains raw assets such as the LESS & Sass files, localization and language files, and Templates that are rendered as HTML.
- **storage:** This directory contains App storage, like file uploads etc. Framework storage (cache), and application-generated logs.
- **test:** This directory contains various test cases.
- **vendor:** This directory contains composer dependencies.

## App Directory

---

This is the application directory. It contains a variety of additional directories, which are described below:

- **Console:** All the artisan commands are stored in this directory.
- **Events:** This directory stores events that your application can raise. Events may be used to alert other parts of your application that a given action has occurred, providing a great deal of flexibility and decoupling.
- **Exceptions:** This directory contains your application's exception handler and is also a good place to stick any exceptions thrown by your application.
- **Http:** This directory contains your controllers, filters, and requests.
- **Jobs:** This directory contains the queueable jobs for your application.
- **Listeners:** This directory contains the handler classes for your events. Handlers receive an event and perform logic in response to the event being fired. For example, a UserRegistered event might be handled by a SendWelcomeEmail listener.
- **Policies:** This directory contains various policies of the application.
- **Providers:** This directory contains various service providers.

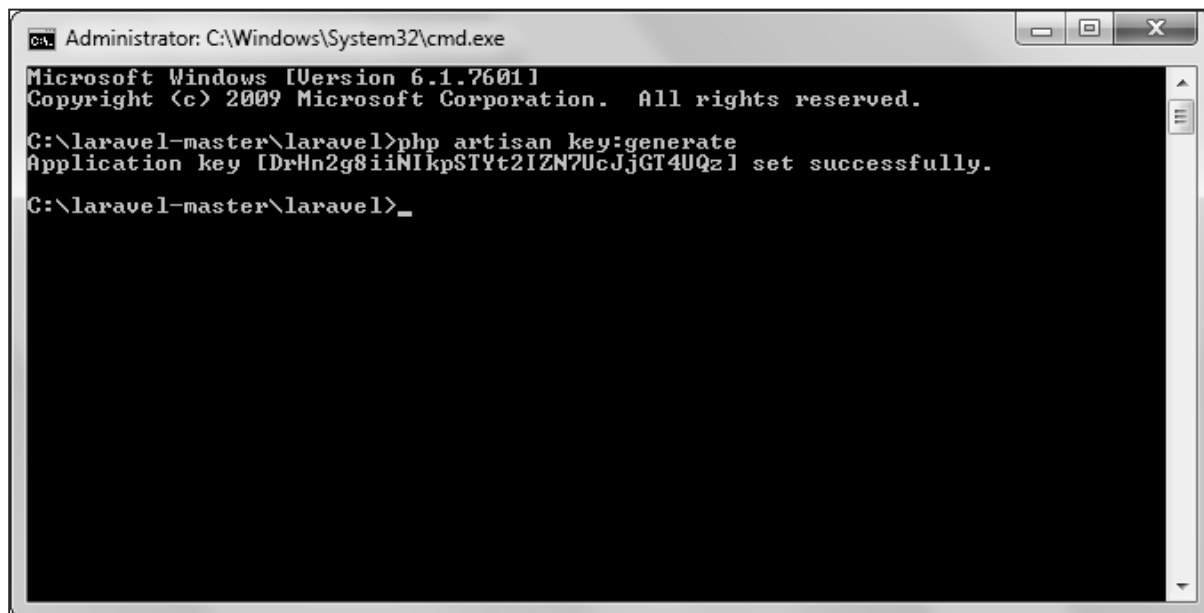
# 4. Laravel – Configuration

The config directory, as the name implies, contains all of your application's configuration files. In this directory, you will find various files needed to configure database, session, mail, application, services etc.

## Basic Configuration

---

- After installing Laravel, the first thing we need to do is to set the write permission for the directory **storage** and **bootstrap/cache**.
- Generate Application key to secure session and other encrypted data. If the root directory doesn't contain the **.env** file then rename the **.env.example** to **.env** file and execute the following command where you have installed Laravel. The newly generated key can be seen in the **.env** file.



```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\laravel-master\laravel>php artisan key:generate
Application key [DrHn2g8iiNIkpSTYt2IZN7UcJjGT4UQz1] set successfully.

C:\laravel-master\laravel>_
```

- You can also configure the locale, time zone, etc. of the application in the **config/app.php** file.

## Environmental Configuration

---

Laravel provides facility to run your application in different environment like testing, production etc. You can configure the environment of your application in the **.env** file of the root directory of your application. If you have installed Laravel using composer, this file will automatically be created.

In case you haven't installed Laravel, you can simply rename the **.env.example** file to **.env** file. A sample of Laravel.env file is shown below.

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=DrHn2g8iiNIkpSTYt2IZN7UcJjGT4UQz

DB_HOST=localhost
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

MAIL_DRIVER=smtp
MAIL_HOST=mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

Notice the text underlined gray in the above image. **Local** environment variable has been set. It can further be changed to **production** or **testing** as per your requirement.

## Database Configuration

---

The database of your application can be configured from **config/database.php** file. You can set configuration parameters that can be used by different databases and you can also set the default one to use.

```
'connections' => [
    'sqlite' => [
        'driver' => 'sqlite',
        'database' => storage_path('database.sqlite'),
        'prefix' => '',
    ],
    'mysql' => [
        'driver' => 'mysql',
        'host' => 'localhost',
        'database' => 'test_db',
        'username' => 'root',
        'password' => '',
        'charset' => 'utf8',
        'collation' => 'utf8_unicode_ci',
        'prefix' => '',
        'strict' => false,
    ],
],
```

## Naming the Application

The App Directory, by default, is namespaced under App. To rename it, you can execute the following command and rename the namespace.

```
php artisan app:name <name-of-your-application>
```

Replace the <name-of-your-application> with the new name of your application that you want to give.

## Maintenance Mode

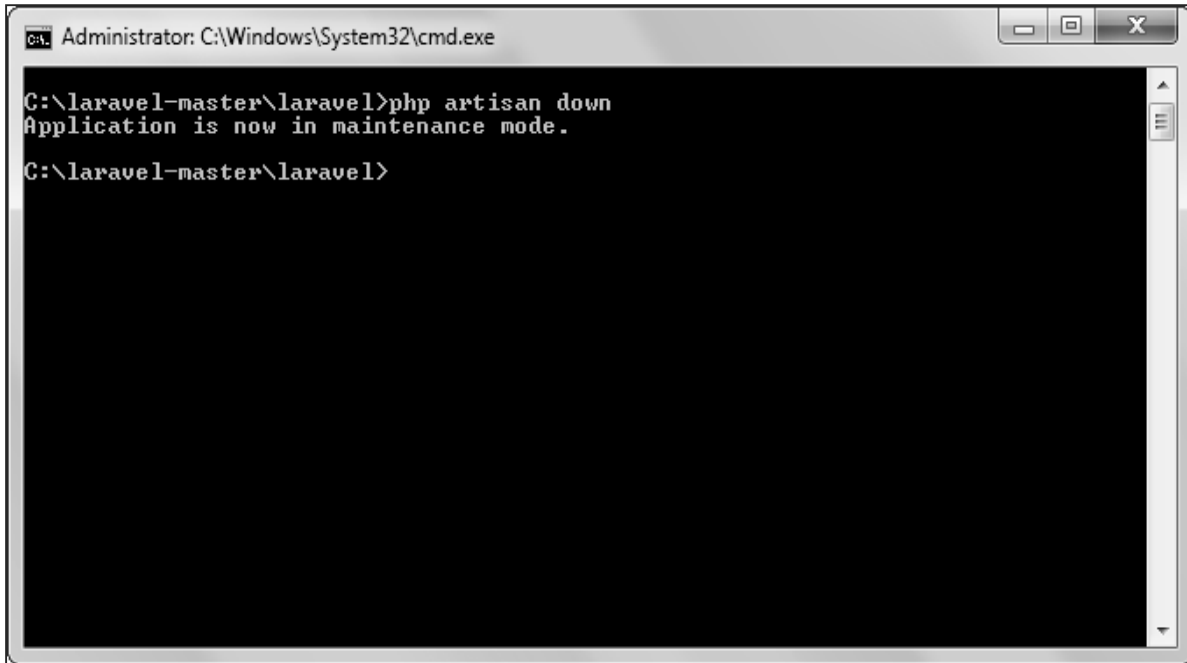
We need to modify our website on a regular basis. The website needs to be put on maintenance mode for this. Laravel has made this job easier. There are two artisan commands which are used to start and stop the maintenance mode which are described below.

### Start Maintenance Mode

**To start the maintenance mode, simply execute the following command.**

```
php artisan down
```

**After successful execution, you will receive the following output:**



```
Administrator: C:\Windows\System32\cmd.exe
C:\laravel-master\laravel>php artisan down
Application is now in maintenance mode.
C:\laravel-master\laravel>
```

**It will activate the Maintenance mode and all the request to server will be redirected to a single maintenance page as shown in the following screenshot.**

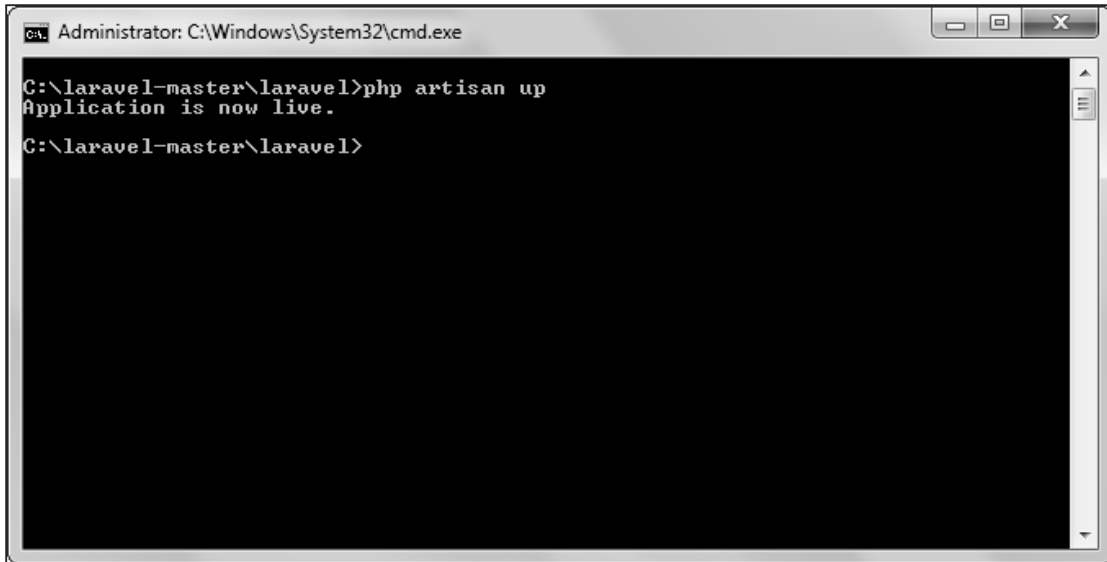


### Stop Maintenance Mode

- After making changes to your website and to start it again, execute the following command.

```
php artisan up
```

- After successful execution, you will receive the following output:



```
Administrator: C:\Windows\System32\cmd.exe
C:\laravel-master\laravel>php artisan up
Application is now live.
C:\laravel-master\laravel>
```



# 5. Laravel – Routing

## Basic Routing

---

Basic routing is meant to route your request to an appropriate controller. The routes of the application can be defined in **app/Http/routes.php** file. Here is the general route syntax for each of the possible request.

```
Route::get('/', function () {
    return 'Hello World';
});

Route::post('foo/bar', function () {
    return 'Hello World';
});

Route::put('foo/bar', function () {
    //
});

Route::delete('foo/bar', function () {
    //
});
```

Let us now understand how to see the Laravel homepage with the help of routing.

### Example

#### app/Http/routes.php

```
<?php
Route::get('/', function () {
    return view('welcome');
```

```
});
```

### resources/view/welcome.blade.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Laravel</title>

    <link href="https://fonts.googleapis.com/css?family=Lato:100"
rel="stylesheet" type="text/css">

    <style>
      html, body {
        height: 100%;
      }

      body {
        margin: 0;
        padding: 0;
        width: 100%;
        display: table;
        font-weight: 100;
        font-family: 'Lato';
      }

      .container {
        text-align: center;
        display: table-cell;
        vertical-align: middle;
      }

      .content {
```

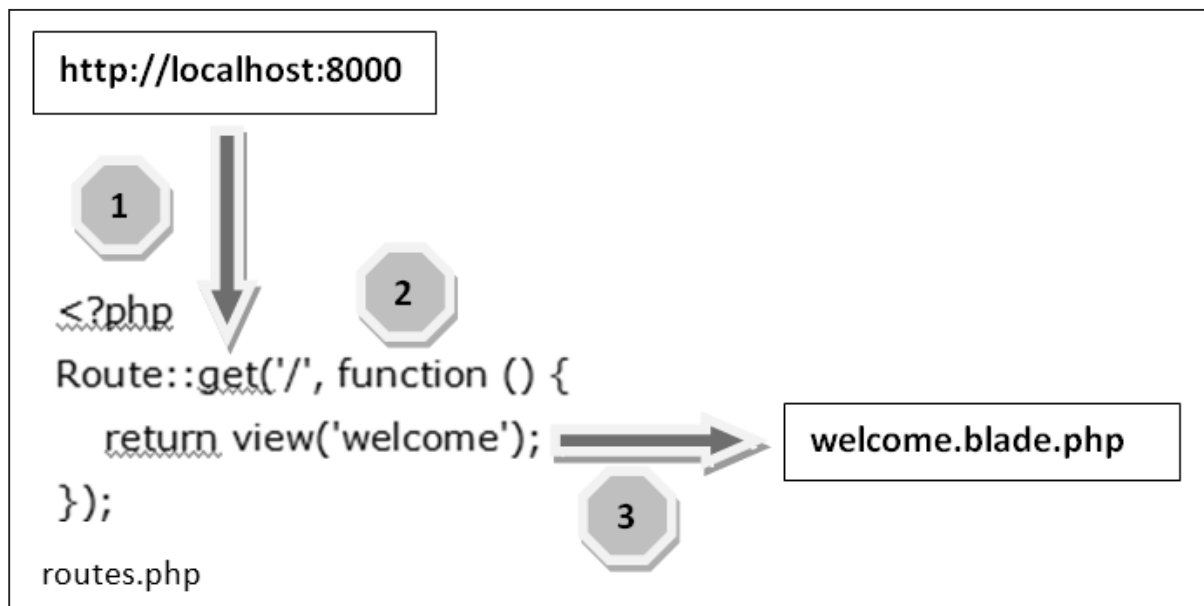
```

        text-align: center;
        display: inline-block;
    }

    .title {
        font-size: 96px;
    }
</style>
</head>
<body>
    <div class="container">
        <div class="content">
            <div class="title">Laravel 5</div>
        </div>
    </div>
</body>
</html>

```

The routing mechanism is depicted in the following image:



Let us now understand the steps in detail:

- **Step 1:** First, we need to execute the root URL of the application.

- **Step 2:** The executed URL will match with the appropriate method in the route.php file. In our case, it will match to get the method and the root ( '/') URL. This will execute the related function.
- **Step 3:** The function calls the template file **resources/views/welcome.blade.php**. The function later calls the **view()** function with argument **'welcome'** without using the **blade.php**. It will produce the following HTML output.



## Routing Parameters

---

Often in the application, we intend to capture the parameters passed with the URL. To do this, we need to modify the code in routes.php file accordingly. There are two ways by which we can capture the parameters passed with the URL.

- Required Parameters
- Optional Parameters

### Required Parameters

These parameters must be present in the URL. For example, you may intend to capture the ID from the URL to do something with that ID. Here is the sample coding for **routes.php** file for that purpose.

```
Route::get('ID/{id}',function($id){
    echo 'ID: '.$id;
});
```

Whatever argument that we pass after the root URL (**http://localhost:8000/ID/5**), it will be stored in `$id` and we can use that parameter for further processing but here we are simply displaying it. We can pass it onto view or controller for further processing.

## Optional Parameters

There are some parameters which may or may not be present in the URL and in such cases we can use the optional parameters. The presence of these parameters is not necessary in the URL. These parameters are indicated by "?" sign after the name of the parameters. Here is the sample coding for **routes.php** file for that purpose.

```
Route::get('/user/{name?}',function($name = 'Virat'){
    echo "Name: ".$name;
});
```

## Example

### routes.php

```
<?php
// First Route method - Root URL will match this method
Route::get('/', function () {
    return view('welcome');
});
// Second Route method - Root URL with ID will match this method
Route::get('ID/{id}',function($id){
    echo 'ID: '.$id;
});

// Third Route method - Root URL with or without name will match this method
Route::get('/user/{name?}',function($name = 'Virat Gandhi'){
    echo "Name: ".$name;
});
```

**Step 1:** Here, we have defined 3 routes with get methods for different purposes. If we execute the below URL then it will execute the first method.

```
http://localhost:8000
```

**Step 2:** After successful execution of the URL, you will receive the following output:



**Step 3:** If we execute the below URL, it will execute the 2<sup>nd</sup> method and the argument/parameter ID will be passed to the variable `$id`.

```
http://localhost:8000/ID/5
```

**Step 4:** After successful execution of the URL, you will receive the following output:

```
ID: 5
```

**Step 5:** If we execute the below URL, it will execute the 3<sup>rd</sup> method and the optional argument/parameter name will be passed to the variable `$name`. The last argument '**Virat**' is optional. If you remove it, the default name will be used that we have passed in the function as '**Virat Gandhi**'

```
http://localhost:8000/user/Virat
```

**Step 6:** After successful execution of the URL, you will receive the following output:

Name: Virat

**Note:** Regular expression can also be used to match the parameters.

End of ebook preview  
If you liked what you saw...  
Buy it from our store @ <https://store.tutorialspoint.com>