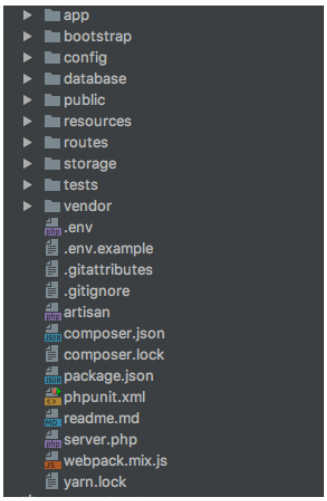
Chapter 2: Building Our First Website

Now that we know how to install Laravel, let's start working our way into our first Laravel website. In this chapter, you will learn about Laravel structure, routes, Controllers, Blade templates, Artisan commands, Elixir and many basic features that will come handy when building Laravel applications.

Exploring Laravel structure

I assume that you've installed Laravel at **~/Code/Laravel**. Let's go there and open the **Laravel** directory.



To build applications using Laravel, you will need to understand truly Laravel.

Laravel follows **MVC (Model View Controller)** pattern, so if you've already known about MVC, everything will be simple. Don't worry if you don't know what MVC is, you will get to know soon.

As you may have seen, every time you visit a Laravel app, you'll see **these folders**.

1. app
2. bootstrap
3. config
4. database
5. public
6. resources
7. routes
8. storage
9. tests
10. vendor

I'm not going to tell you everything about them right now because I know that it's boring.

Trust me.

But we have to take a quick look at them to know what they are, anyway.

App

This directory holds all our application's logic. We will put our controllers, services, filters, commands and many other custom classes here.

Bootstrap

This folder has some files that are used to bootstrap Laravel. The cache folder is also placed here.

Config

When we want to configure our application, check out this folder. We will configure database, mail, session, etc. here.

Database

As the name implies, this folder contains our database migrations and database seeders.

Public

The public folder contains the application's images, CSS, JS and other public files.

Resources

We should put our views (.blade.php files), raw files and other localization files here.

Routes

All our **route files** will be stored here.

We'll learn about this **routes directory** in the next section.

Storage

Laravel will use this folder to store sessions, caches, templates, logs, etc.

Tests

This folder contains the test files, such as PHPUnit files.

Vendor

Composer dependencies (such as Symfony classes, PHPUnit classes, etc.) are placed here.

To understand more about Laravel structure, you can read the official documentation here:

https://laravel.com/docs/master/structure

Understand the routes directory

One of the most important features of Laravel is "**routing**".

What does it mean?

Routing means that you will tell Laravel to get URL requests and assign them to specific actions that you want. For instance, when someone visits **homestead.test**, which is the home page of our current application, Laravel will think: **"Oh, this guy is going to the home page, I need to display something!"**

We usually register all routes in the **routes.php** file, but since Laravel 5.3, the **routes.php** file has been moved to the new **routes** directory. We now have three files: **web.php**, **console.php** and **api.php**.

Changing Laravel home page

To change Laravel home page, we need to edit the **web.php** file.

Let's see what's inside the file.

routes/web.php

```
Route::get('/', function () {
    return view('welcome');
});
```

We have only one route by default.

This route tells Laravel to return the **welcome view** when someone make a **GET request** to our **root URL** (which represents by the /).

So if we want to edit the home page, we need to modify the **welcome view**.

What is view and where is the welcome view?

Views contain the HTML served by our application. A simple view may look like a simple HTML file:

```
<html>
    <body>
      <p> A simple view </p>
    </body
</html>
```

All views are stored at the **resources/views** directory.

Go to the **views** folder, find a file called **welcome.blade.php**. It's the **welcome view**.

Open it:

resources/views/welcome.blade.php

```
<!doctype html>
<html lang="{{ app()->getLocale() }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title>Laravel</title>

        <!-- Fonts -->
        <link href="https://fonts.googleapis.com/css?family=Nunito:200,600" rel="stylesheet" type="text/css">

        <!-- Styles -->
        <style>
            html, body {
                background-color: #fff;
                color: #636b6f;
```

```
                color: #636b6f;
                font-family: 'Nunito', sans-serif;
                font-weight: 200;
                height: 100vh;
                margin: 0;
            }

            .full-height {
                height: 100vh;
            }

            .flex-center {
                align-items: center;
                display: flex;
                justify-content: center;
            }

            .position-ref {
                position: relative;
            }

            .top-right {
                position: absolute;
                right: 10px;
                top: 18px;
            }

            .content {
                text-align: center;
            }

            .title {
                font-size: 84px;
            }

            .links > a {
                color: #636b6f;
                padding: 0 25px;
                font-size: 12px;
                font-weight: 600;
                letter-spacing: .1rem;
                text-decoration: none;
                text-transform: uppercase;
            }

            .m-b-md {
                margin-bottom: 30px;
            }
        </style>
    </head>
    <body>
        <div class="flex-center position-ref full-height">
            @if (Route::has('login'))
                <div class="top-right links">
                    @auth
                        <a href="{{ url('/home') }}">Home</a>
                    @else
                        <a href="{{ route('login') }}">Login</a>
                        <a href="{{ route('register') }}">Register</a>
                    @endauth
                </div>
            @endif

            <div class="content">
                <div class="title m-b-md">
                    Laravel
                </div>

                <div class="links">
                    <a href="https://laravel.com/docs">Documentation</a>
                    <a href="https://laracasts.com">Laracasts</a>
                    <a href="https://laravel-news.com">News</a>
                    <a href="https://nova.laravel.com">Nova</a>
                    <a href="https://forge.laravel.com">Forge</a>
                    <a href="https://github.com/laravel/laravel">GitHub</a>
                </div>
            </div>
        </div>
    </body>
</html>
```

This welcome view is used to display the home page. It just looks like a basic HTML file!

I assume that you've already known HTML, so you should understand the content of this file clearly.

> **Tip:** If you don't, **w3schools** is a good place to learn HTML and PHP: http://www.w3schools.com/html

It's time to modify the homepage!

What should we do?...

How about display a custom quote?

Let's replace these lines:

```
<div class="links">
    <a href="https://laravel.com/docs">Documentation</a>
```

```
    <a href="https://laracasts.com">Laracasts</a>
    <a href="https://laravel-news.com">News</a>
    <a href="https://nova.laravel.com">Nova</a>
    <a href="https://forge.laravel.com">Forge</a>
    <a href="https://github.com/laravel/laravel">GitHub</a>
</div>
```

with:

```
<div class="quote">Put your custom quote here!</div>
```

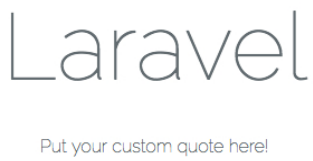
Our new **welcome view** should look like this:

```
<!doctype html>
<html lang="{{ app()->getLocale() }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title>Laravel</title>

        <!-- Fonts -->
        <link href="https://fonts.googleapis.com/css?family=Nunito:200,600" rel="stylesheet" type="text/css">

        <!-- Styles -->
        <style>
            html, body {
                background-color: #fff;
                color: #636b6f;
                font-family: 'Nunito', sans-serif;
                font-weight: 200;
                height: 100vh;
                margin: 0;
            }

            .full-height {
                height: 100vh;
            }

            .flex-center {
                align-items: center;
                display: flex;
                justify-content: center;
            }

            .position-ref {
                position: relative;
            }

            .top-right {
                position: absolute;
                right: 10px;
                top: 18px;
            }

            .content {
                text-align: center;
            }

            .title {
                font-size: 84px;
            }

            .links > a {
                color: #636b6f;
                padding: 0 25px;
                font-size: 12px;
                font-weight: 600;
                letter-spacing: .1rem;
                text-decoration: none;
                text-transform: uppercase;
            }

            .m-b-md {
                margin-bottom: 30px;
            }
        </style>
    </head>
    <body>
        <div class="flex-center position-ref full-height">
            @if (Route::has('login'))
                <div class="top-right links">
                    @auth
                        <a href="{{ url('/home') }}">Home</a>
                    @else
                        <a href="{{ route('login') }}">Login</a>
                        <a href="{{ route('register') }}">Register</a>
                    @endauth
                </div>
            @endif

            <div class="content">
                <div class="title m-b-md">
                    Laravel
```

```
                </div>

                <div class="quote">Put your custom quote here!</div>

            </div>
        </div>
    </body>
</html>
```

> **Note:** Laravel welcome view is changed frequently. If your view is different, just copy the code above to **create a new welcome view**.

Save the file, head over to your browser and run **homestead.test**.



Awesome! We have just changed our home page by changing the **welcome.blade.php** template!

In fact, we may change any page without returning a view:

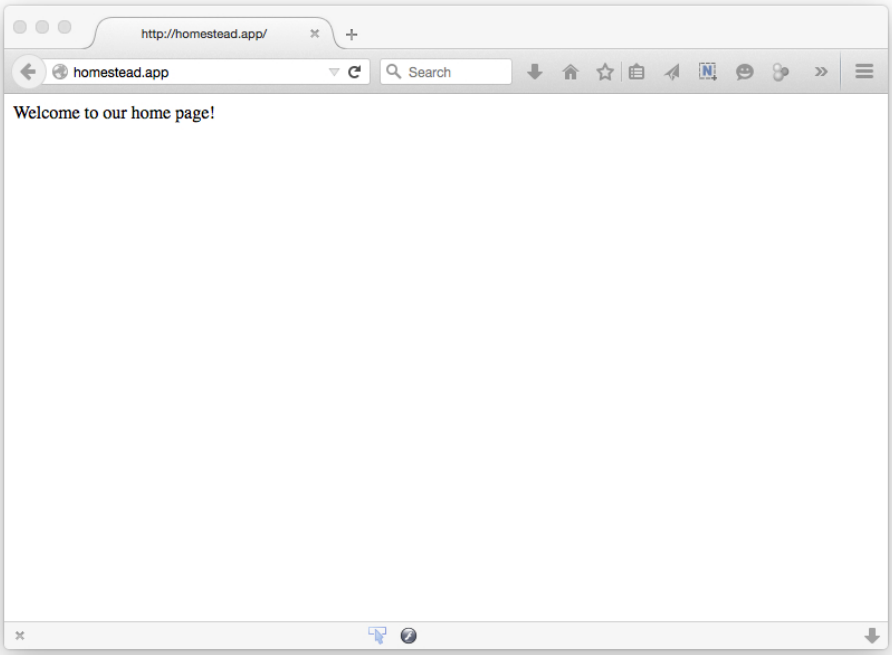
Modify the first route:

routes/web.php

```
Route::get('/', function () {
    return view('welcome');
});
```

to

```
Route::get('/', function()
{
return 'Welcome to our home page!';
});
```



Amazing! Right?

We have just used a **anonymous function** to change our home page. In PHP, we called this function: "**Closure**".

A Closure is a function that doesn't have a name.

We use **Closure** to handle "routing" in small applications. In large applications, we use **Controllers**.

You can find Controllers documentation here:

http://laravel.com/docs/master/controllers

It's recommended that you should always use Controllers because Controllers help to structure your code easier. For instance, you may group all user actions into **UserController**, all post actions into **PostsController**.

The disadvantage of Controllers is, you will need to create a file for each Controller, thus it takes a bit more time.

To understand what **Controllers** is, we will be creating some pages using **Controllers**.

Adding more pages to our first website

When we have a basic understanding of how we can edit the home page, adding more pages is easy.

Imagine that we're going to build a website to introduce the Learning Laravel book. Our website will have three pages:

1. Home page.
2. About page.
3. Contact page.

Because we have many pages, and we might add more pages to our website, we should organize all our pages in **PagesController**.

As you've noticed, we don't have the **PagesController** yet, thus we're going to create it.

Create our first controller

There are two methods to create a controller:

Create a controller manually

To start off with things, create a new file called **PagesController.php** with your favorite text editor (PHPStorm, Sublime Text, etc.).

Place the file in **app/Http/Controllers/** directory.

Update the **PagesController.php** file to look like this:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    public function home()
    {
        return view('welcome');
    }
}
```

Good job! You now have PagesController with a **home action**.

You may notice that the **home action** tells Laravel to "**return the welcome view**":

```php
return view('welcome');
```

Create a controller by using Artisan

Rather than manually creating a controller, we can use **Artisan** to generate it automatically.

Artisan is Laravel command line tool that helps us to perform tasks that we hate to do manually. Using Artisan, we can create models, views, controllers, migrations and many other things.

You have known how to use Terminal or Git Bash, let's create a controller by running this command:

```
php artisan make:controller PagesController
```

> **Note: vagrant ssh** to your VM, **cd** to Laravel folder, and use the command there. Delete the old controller if you've created it.

You'll see:

```
Homestead [master] vagrant ssh
Welcome to Ubuntu 14.10 (GNU/Linux 3.16.0-23-generic x86_64)
```

```
 * Documentation:  https://help.ubuntu.com/
Last login: Mon Jun  1 05:19:19 2015 from 10.0.2.2
vagrant@homestead:~$ cd Code/Laravel
vagrant@homestead:~/Code/Laravel$ php artisan make:controller PagesController
Controller created successfully.
vagrant@homestead:~/Code/Laravel$
```

By default, Laravel creates a plain controller:

app/Http/Controllers/PagesController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    //
}
```

Next, let's add a new a home action:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    public function home()
    {
        return view('welcome');
    }
}
```

Alternatively, you can use this command to generate a new **RESTful PagesController**:

```
php artisan make:controller PagesController --resource
```

Good job! You now have **PagesController** with the **home action**.

You may notice that the home action tells Laravel to "**return the welcome view**":

```php
return view('welcome');
```

Using our first controller

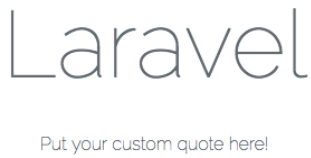
Cool! When you have **PagesController**, the next thing to do is modifying our **routes**!

Open the **web.php** file. Change the default route to:

routes/web.php

```php
Route::get('/', 'PagesController@home');
```

This route tells Laravel to execute the **home action** (which can be found in **PagesController**) when someone makes a **GET request** to our **root URL** (which represents by the **/**).



Well done! You've just displayed the front page using your own controller!

Create other pages

Now that we have **PagesController** class. Adding more pages is not difficult.

> **Suggestion:** How about trying to add the about and contact page yourself?

Edit the **web.php** file. Add the following lines:

```
Route::get('/about', 'PagesController@about');
Route::get('/contact', 'PagesController@contact');
```

Open **PagesController**, add:

```
public function about()
{
    return view('about');
}

public function contact()
{
    return view('contact');
}
```

Here is the updated **PagesController.php** file:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    public function home()
    {
        return view('welcome');
    }

    public function about()
    {
        return view('about');
    }

    public function contact()
    {
        return view('contact');
    }
}
```

Next you will want to create **about view** and **contact view**. Create two new files in the **resources/views** directory named **about.blade.php** and **contact.blade.php**.

Finally, add the following contents (copy from the **welcome view**) to each file:

resources/views/about.blade.php

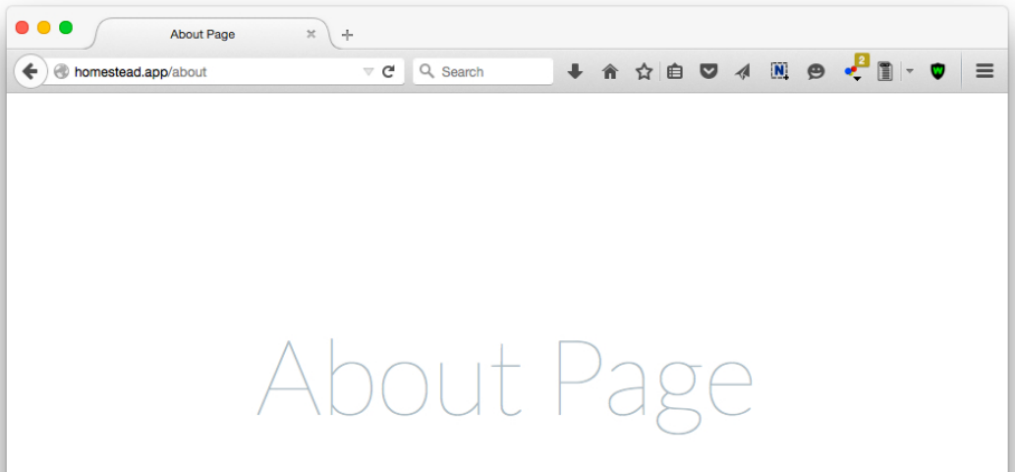
```
<html>
    <head>
        <title>About Page</title>

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

        <style>
            body {
                margin: 0;
                padding: 0;
                width: 100%;
                height: 100%;
                color: #B0BEC5;
                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;
```

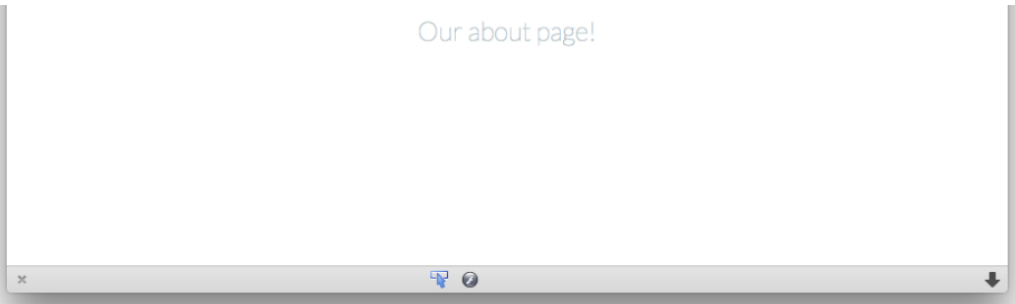
```
                margin-bottom: 40px;
            }

            .quote {
                font-size: 24px;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="content">
                <div class="title">About Page</div>
                <div class="quote">Our about page!</div>
            </div>
        </div>
    </body>
</html>
```

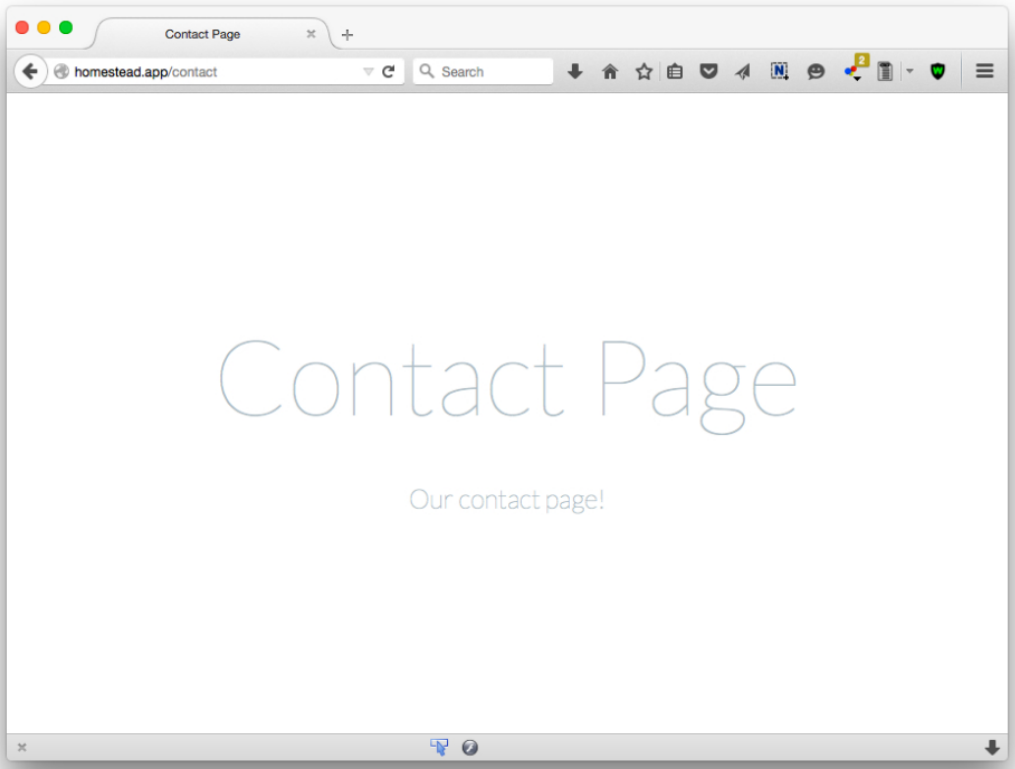
resources/views/contact.blade.php

```
<html>
    <head>
        <title>Contact Page</title>

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

        <style>
            body {
                margin: 0;
                padding: 0;
                width: 100%;
                height: 100%;
                color: #B0BEC5;
                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;
                margin-bottom: 40px;
            }

            .quote {
                font-size: 24px;
            }
        </style>
    </head>
    <body>
    <div class="container">
        <div class="content">
            <div class="title">Contact Page</div>
            <div class="quote">Our contact page!</div>
        </div>
    </div>
    </body>
</html>
```

Save these changes, go to **homestead.test/about** and **homestead.test/contact**:







Yayyyy! We have created the about and contact page with just a few lines of code!

Now, let's create a **home view** for our homepage, and change the **PagesController** to use it:

resources/views/home.blade.php

```
<html>
<head>
    <title>Home Page</title>

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

    <style>
        body {
            margin: 0;
            padding: 0;
            width: 100%;
            height: 100%;
            color: #B0BEC5;
            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 {
```

```
            .title {
                font-size: 96px;
                margin-bottom: 40px;
            }

            .quote {
                font-size: 24px;
            }
        </style>
    </head>
    <body>
    <div class="container">
        <div class="content">
            <div class="title">Home Page</div>
            <div class="quote">Our Home page!</div>
        </div>
    </div>
    </body>
    </html>
```

PagesControllers

Find:

```
    public function home()
    {
        return view('welcome');
    }
```

Replace with:

```
public function home()
{
    return view('home');
}
```

Integrating Bootstrap

Nowadays, the most popular front-end framework is **Bootstrap (aka Twitter Bootstrap)**. It's free, open source and has a large active community. I've been using Bootstrap for all projects.

Using Bootstrap, we can quickly develop responsive, mobile-ready web applications. Millions of beautiful and popular sites across the world are built with Bootstrap.

In this section, we will learn how to integrate Bootstrap into our Laravel application.

You can get Bootstrap and read its official documentation here:

https://getbootstrap.com/

> **Note:** We're using the new Bootstrap 4 in this book. If you want to use Bootstrap 3, check the old version of the book.

There are many ways to integrate Bootstrap. You can install it using Bootstrap CDN, Bower, npm, etc.

I'll show you the most three popular methods:

1. Using Bootstrap CDN
2. Using Precompiled Bootstrap Files
3. Using Bootstrap Source Code (Sass)

You can choose to use any method that you like.

By the way, before going to the next section, you may notice that we've added some CSS styles and Lato font into our **home.blade.php** file before. **Remove** those first:

```
<style>
    body {
        margin: 0;
        padding: 0;
        width: 100%;
        height: 100%;
        color: #B0BEC5;
        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;
        margin-bottom: 40px;
    }

    .quote {
        font-size: 24px;
    }
</style>
```

and

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

Using Bootstrap CDN

The fastest way to integrate Bootstrap is using CDN.

Open **home.blade.php**, place these links inside the **head** tag:

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
```

Find:

```
</body>
```

Add above:

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
```

Done! You now have fully integrated Twitter Bootstrap into our website!

> **Note:** All the links can be found at https://getbootstrap.com/docs/4.1/getting-started/introduction/

Here is our new **home.blade.php**:

```
<html>
<head>
    <title>Home Page</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
</head>
<body>

<div class="container">
    <div class="content">
        <div class="title">Home Page</div>
        <div class="quote">Our Home page!</div>
    </div>
</div>

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>

</body>
</html>
```

Using Precompiled Bootstrap Files

Bootstrap 4 is built using Sass - a CSS pre-processor.

Sass allows us to use variables, mixins, functions and other techniques to enhance CSS. You can learn more about Less here:

https://sass-lang.com/

Unfortunately, browsers don't understand Sass. You must **compile** all Less files using **Sass compiler** to produce CSS files. Of course, you have to learn Sass.

Luckily, Bootstrap has provided compiled CSS, JS and fonts for us. We can download and use them without worrying about the Sass files. Go to:

https://getbootstrap.com/docs/4.1/getting-started/download/

> **Note:** I'm using **version 4.1** here. You may use a newer version if you want.

Find the "**Compiled CSS and JS**" section and click **Download** to download the latest compiled Bootstrap files.

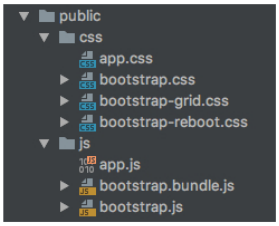
Uncompress the downloaded .zip file. We have three folders:

1. css
2. js

Put them all into the **public** folder of your app. (**~/Code/Laravel/public**).

> Note: By default, Laravel has created **css** and **js** folder for us.

When you visit your public folder, it should look like this:



To load Bootstrap framework for styling our home page, open **home.blade.php** and place this link inside the **head** tag

```
<link rel="stylesheet" href="{!! asset('css/bootstrap.min.css') !!}" >
```

Find:

```
</body>
```

Add above:

```
<script src="{!! asset('js/bootstrap.min.js') !!}"></script>
```

Twitter Bootstrap requires **jQuery** and **Popper.js** to work properly.

You can download **jQuery** here:

https://jquery.com/download

You can download **Popper.js** here:

https://unpkg.com/popper.js/dist/umd/popper.min.js

Put the downloaded files into the **public** directory as well, then use the following code to reference them:

```
<script src="{!! asset('js/jquery-3.3.1.min.js') !!}"></script>
<script src="{!! asset('js/popper.min.js') !!}"></script>
```

> **Note:** Your jQuery version may be different.

Here is our updated homepage:

home.blade.php

```
<html>
<head>
    <title>Home Page</title>
        <link rel="stylesheet" href="{!! asset('css/bootstrap.min.css') !!}">
</head>
<body>

<div class="container">
    <div class="content">
        <div class="title">Home Page</div>
        <div class="quote">Our Home page!</div>
    </div>
</div>

<script src="{!! asset('js/jquery-3.3.1.min.js') !!}"></script>
<script src="{!! asset('js/popper.min.js') !!}"></script>
<script src="{!! asset('js/bootstrap.min.js') !!}"></script>

</body>
</html>
```

Done! You now have fully integrated **Bootstrap** into our website!

We've used **asset** function to link CSS and JS files to our app. You can also use the **asset** function to link images, fonts and other public files. If you don't want to use asset function, you can use relative links instead:

```
<link rel="stylesheet" href="/css/bootstrap.min.css" >
```

```
<script src="/js/jquery-3.3.1.min.js"></script>
<script src="/js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
```

Using Bootstrap Source Code (Sass)

The good news is, the latest version of Laravel has officially supported **Sass**. Sass files are placed at **resources/assets/sass**.

We'll use **Laravel Mix** to automatically compile Sass files to **app.css** file.

To load Twitter Bootstrap framework for styling our home page, open **home.blade.php** and place this link inside the **head** tag:

```
<link rel="stylesheet" href="/css/app.css">
```

Find:

```
</body>
```

Add above:

```
<script src="/js/app.js"></script>
```

Done! You now have fully integrated Bootstrap into your website!

Here is our new **home.blade.php**:

```
<html>
<head>
    <title>Home Page</title>
    <link rel="stylesheet" href="/css/app.css">
</head>
<body>

<div class="container">
    <div class="content">
        <div class="title">Home Page</div>
        <div class="quote">Our Home page!</div>
    </div>
</div>
<script src="/js/app.js"></script>
</body>
</html>
```

Oh, I've mentioned **Laravel Mix**, what is it?

Introducing Laravel Mix

One of the best Laravel 5 new features is **Laravel Mix**. We can use **Laravel Mix** to compile Sass files, Coffee Scripts or automate other manual tasks.

> **Note:** Laravel Mix is a new feature of Laravel 5.4+. In older versions of Laravel, we have **Elixir**, which is pretty similar.

Laravel Mix official documentation

Basically, Laravel Mix is based on **Webpack**. If you don't know about **Webpack** yet, you can find more information about it here:

Webpack Official Home Page

To use Laravel Mix, we must have **NPM**(Node Package Manager) and **Node.js** installed.

Luckily, Homestead has NPM and Node.js by default, we can use Laravel Mix right away.

If you don't use Homestead, you have to install Node.js and NPM.

> **Tip:** Actually, you should run Node.js/Laravel Mix directly on your local machine (Windows, Mac, etc.). It's faster and less buggy.

To install Node.js, visit:

https://nodejs.org

> **Note:** Be sure to use Node.js v10.11.0 or newer. The direct download links can also be found at:
> https://nodejs.org/en/blog/release/v10.11.0

Follow instructions on the site, you should have installed Node.js easily.

You may check the version of Node.js by running:

```
node -v
```

You may check the version of NPM by running:

```
npm -v
```

The last step is to install Laravel Mix. Laravel 5 has included a file called **package.json**. You use this file to install Node modules. Open the file, you should see:

package.json

```
{
    "private": true,
    "scripts": {
        "dev": "npm run development",
        "development": "cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_m
        "watch": "npm run development -- --watch",
        "watch-poll": "npm run watch -- --watch-poll",
        "hot": "cross-env NODE_ENV=development node_modules/webpack-dev-server/bin/webpack-dev-server.js --inline --hot --config=noc
        "prod": "npm run production",
        "production": "cross-env NODE_ENV=production node_modules/webpack/bin/webpack.js --no-progress --hide-modules --config=node_
    },
    "devDependencies": {
        "axios": "^0.18",
        "bootstrap": "^4.0.0",
        "cross-env": "^5.1",
        "jquery": "^3.2",
        "laravel-mix": "^2.0",
        "lodash": "^4.17.5",
        "popper.js": "^1.12",
        "vue": "^2.5.7"
    }
}
```

Do you see the `laravel-mix` there?

Good! Navigate to our **app root** (~/Code/Laravel), run this command to install **Laravel Mix**:

```
npm install
```

> **Note:** If you install Node.js on your local machine, you should run this command directly on your machine (Windows, macOS, etc.).

Windows system users may need to run this command instead:

```
npm install --no-bin-links
```

Mac users may need to run this command instead (if the above command doesn't work):

```
sudo npm install
```

```
├─┬ axios@0.15.3
│ └─┬ follow-redirects@1.0.0
│   └─┬ debug@2.6.0
│     └── ms@0.7.2
├── bootstrap-sass@3.3.7
├── jquery@3.1.1
├─┬ laravel-mix@0.6.3
│ ├─┬ babel-core@6.22.1
│ │ ├─┬ babel-code-frame@6.22.0
│ │ │ ├── esutils@2.0.2
│ │ │ └── js-tokens@3.0.1
│ │ ├─┬ babel-generator@6.22.0
│ │ │ ├─┬ detect-indent@4.0.0
│ │ │ │ └─┬ repeating@2.0.1
│ │ │ │   └── is-finite@1.0.2
│ │ │ └── jsesc@1.3.0
│ │ ├── babel-helpers@6.22.0
│ │ ├── babel-messages@6.22.0
│ │ ├─┬ babel-register@6.22.0
│ │ │ ├── core-js@2.4.1
│ │ │ ├─┬ home-or-tmp@2.0.0
│ │ │ │ └── os-tmpdir@1.0.2
│ │ │ └── source-map-support@0.4.11
│ │ ├─┬ babel-runtime@6.22.0
│ │ │ └── regenerator-runtime@0.10.1
│ │ ├── babel-template@6.22.0
│ │ ├─┬ babel-traverse@6.22.1
│ │ │ ├── globals@9.14.0
│ │ │ └─┬ invariant@2.2.2
│ │ │   └── loose-envify@1.3.1
│ │ ├─┬ babel-types@6.22.0
│ │ │ └── to-fast-properties@1.0.2
│ │ ├── babylon@6.15.0
│ │ ├── convert-source-map@1.3.0
```

It may take a while to download. Be patient.

Once complete, there is a new folder called **node_modules** in our app. You can find Laravel Mix and other Node.js packages here.

Running first Laravel Mix task

You can write a new Mix task in **webpack.mix.js**. By default, you can find a Mix task that compiles **app.sass** file into **app.css**, and bundles all JS files there:

```
mix.js('resources/assets/js/app.js', 'public/js')
   .sass('resources/assets/sass/app.scss', 'public/css');
```

Feel free to add more tasks by reading the official Laravel Mix documentation:

https://laravel.com/docs/master/mix

To execute your Mix task, run this command:

```
npm run dev
```

```
DONE  Compiled successfully in 4692ms

                                                    Asset       Size  Chunks
  fonts/glyphicons-halflings-regular.eot?f4769f9bdb7466be65088239c12046d1  20.1 kB          [emitted]
  fonts/glyphicons-halflings-regular.svg?89889688147bd7575d6327160d64e760   109 kB          [emitted]
  fonts/glyphicons-halflings-regular.ttf?e18bbf611f2a2e43afc071aa2f4e1512  45.4 kB          [emitted]
 fonts/glyphicons-halflings-regular.woff?fa2772327f55d8198301fdb8bcfc8158  23.4 kB          [emitted]
fonts/glyphicons-halflings-regular.woff2?448c34a56d699c29117adc64c43affeb    18 kB          [emitted]
                                                               /js/app.js  1.16 MB       0  [emitted]
                                                              /css/app.css   146 kB       0  [emitted]
                                                         mix-manifest.json  66 bytes          [emitted]
```

By running the task, Laravel will automatically compile the app.sass file and bundle all JS files. The output files can be found in your **public** directory.

Adding Bootstrap components

Cool! Now we can add **Bootstrap components** into our website.

There are many reusable Bootstrap components built to provide navbars, labels, dropdowns, panels, etc. You can see a full list of components here:

https://getbootstrap.com/docs/4.1/components/alerts/

To use the components, you can copy the example codes, and paste them into your application. For instance, we can add Bootstrap navbar to our app by adding these codes to **home.blade.php**:

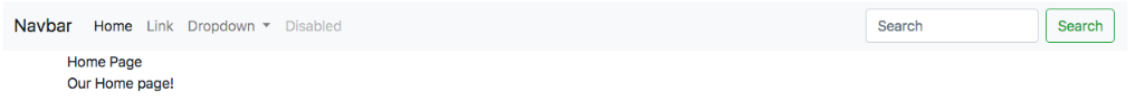
```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSup
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" a
          Dropdown
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>
```

Here is our new **home.blade.php** after adding the navbar:

home.blade.php

```
<html>
<head>
    <title>Home Page</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarS
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Link</a>
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup=
                    Dropdown
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                    <a class="dropdown-item" href="#">Action</a>
                    <a class="dropdown-item" href="#">Another action</a>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="#">Something else here</a>
                </div>
            </li>
            <li class="nav-item">
                <a class="nav-link disabled" href="#">Disabled</a>
            </li>
        </ul>
        <form class="form-inline my-2 my-lg-0">
            <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
            <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
        </form>
    </div>
</nav>

<div class="container">
    <div class="content">
        <div class="title">Home Page</div>
        <div class="quote">Our Home page!</div>
    </div>
</div>

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
</body>
</html>
```

It's time to refresh our browser:



Home Page

Our Home page!

Responsive meta tag

To make our site more responsive to display well across all devices, we should put a responsive meta tag in the head section of our template:

Find:

```
<head>
```

Add below:

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

> **Note:** this step is optional, but it's recommended to include the tag when using Bootstrap.

Learning Blade templates

It's time to learn about **Blade**!

Blade is an official Laravel's templating engine. It's very powerful, but it has very simple syntax. We use Blade to build layouts for our Laravel applications.

Blade view files have **.blade.php** file extension. The **home view** and **other views** that we've been using are Blade templates.

Usually, we put all Blade templates in **resources/views** directory. The great thing is, we can use plain PHP code in a Blade view.

All Blade expressions begin with `@` . For example: `@section` , `@if` , `@for` , etc.

Blade also supports all PHP loops and conditions: `@if` , `@elseif` , `@else` , `@for` , `@foreach` , `@while` , etc. For instance, you can write a `if else` statement in Blade like this:

```
@if ($product == 1)
 {!! $product->name !!}
@else
There is no product!
@endif
```

Equivalent PHP code:

```
if ($product ==1) {
echo $product->name;
} else {
echo("There is no product!");
}
```

To understand Blade's features, we will use Blade to build our first website's layout.

Creating a master layout

The typical web application has a **master layout**. The layout consists header, footer, sidebar, etc. Using a master layout, we can easily place one element in every view. For instance, we can use the same header and footer for all pages. It helps to make our code look clearer and save our time a lot.

To get started, we will create a master layout called **master.blade.php**

resources/views/master.blade.php

```
<html>
<head>
    <title> @yield('title') </title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
</head>
<body>
```

```
    @include('shared.navbar')

    @yield('content')

    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
</body>
</html>
```

This view look likes the home view, but we've changed something. Let's see the code line by line.

```
<title> @yield('title') </title>
```

Instead of putting a title here, we use **@yield** directive to get the title from another section.

```
@include('shared.navbar')
```

We use **@include** directive to include other Blade views. You may notice that we've embedded a view called **navbar** here.

However, we don't have the navbar view yet, let's create a **shared** directory and put **navbar.blade.php** there.

Copy the Twitter Bootstrap navbar and put it into the **navbar.blade.php**:

resources/views/shared/navbar.blade.php

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarS
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Link</a>
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup=
                    Dropdown
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                    <a class="dropdown-item" href="#">Action</a>
                    <a class="dropdown-item" href="#">Another action</a>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="#">Something else here</a>
                </div>
            </li>
            <li class="nav-item">
                <a class="nav-link disabled" href="#">Disabled</a>
            </li>
        </ul>
        <form class="form-inline my-2 my-lg-0">
            <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
            <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
        </form>
    </div>
</nav>
```

> **Note:** You may change the **shared** folder name to **partials**, **embed** or whatever you like.

```
@yield('content')
```

As you see it's really convenient for us to not display the content of any pages here. We simply use **@yield** directive to embed a section called **content** from other views.

Great! You've just created a master layout!

Extending the master layout

Now we can change the **home** view to extend our master layout.

resources/views/home.blade.php

```
@extends('master')
@section('title', 'Home')

@section('content')
    <div class="container">
        <div class="content">
```

```
            <div class="title">Home Page</div>
            <div class="quote">Our Home page!</div>
        </div>
    </div>
@endsection
```

As you can observe we use **@extends** directive to inherit our **master layout**.

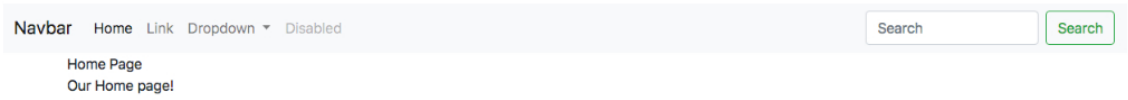
To set the title for our home page, we use **@section** directive.

```
@section('title', 'Home')
```

It's the "short way". If we have a long content, we can use **@section** and **@endsection** to inject our **content** into the master layout.

```
@section('content')
    <div class="container">
        <div class="content">
            <div class="title">Home Page</div>
            <div class="quote">Our Home page!</div>
        </div>
    </div>
@endsection
```

Refresh your browser, you should see the same home page, but this time our code look much cleaner.



Home Page
Our Home page!

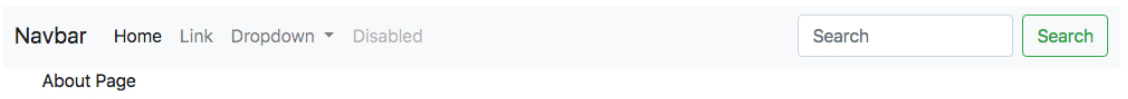
Using the same technique, we can easily change the about and contact page:

about.blade.php

```
@extends('master')
@section('title', 'About')

@section('content')
    <div class="container">
        <div class="content">
            <div class="title">About Page</div>
            <div class="quote">Our about page!</div>
        </div>
    </div>
@endsection
```

contact.blade.php

```
@extends('master')
@section('title', 'Contact')

@section('content')
    <div class="container">
        <div class="content">
            <div class="title">Contact Page</div>
            <div class="quote">Our contact page!</div>
        </div>
    </div>
@endsection
```



About Page
Our about page!

Using other Bootstrap themes

The best thing about Bootstrap is, it has many themes for you to "switch". If you don't like the default Bootstrap theme, you can easily find another one to use. Here are some popular themes for Bootstrap:

1. http://bootswatch.com
2. http://fezvrasta.github.io/bootstrap-material-design
3. http://designmodo.github.io/Flat-UI

If you like a theme, you can change the layout to that theme by following the instructions on its website.

> **Note:** In the old versions of this book, we've learned how to apply **Bootstrap Material Design** theme for our website. However, I think it's best to learn Laravel using the default Bootstrap 4 theme first, so we won't use another theme.

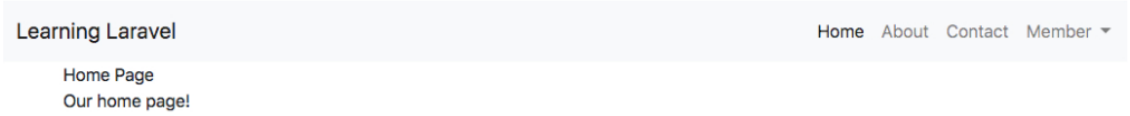
Refine our website layouts

Currently, the site looks messy. We're going to change a few things and re-design all views to make our application look better and professional.

Changing the navbar

Open **navbar.blade.php**, and update it:

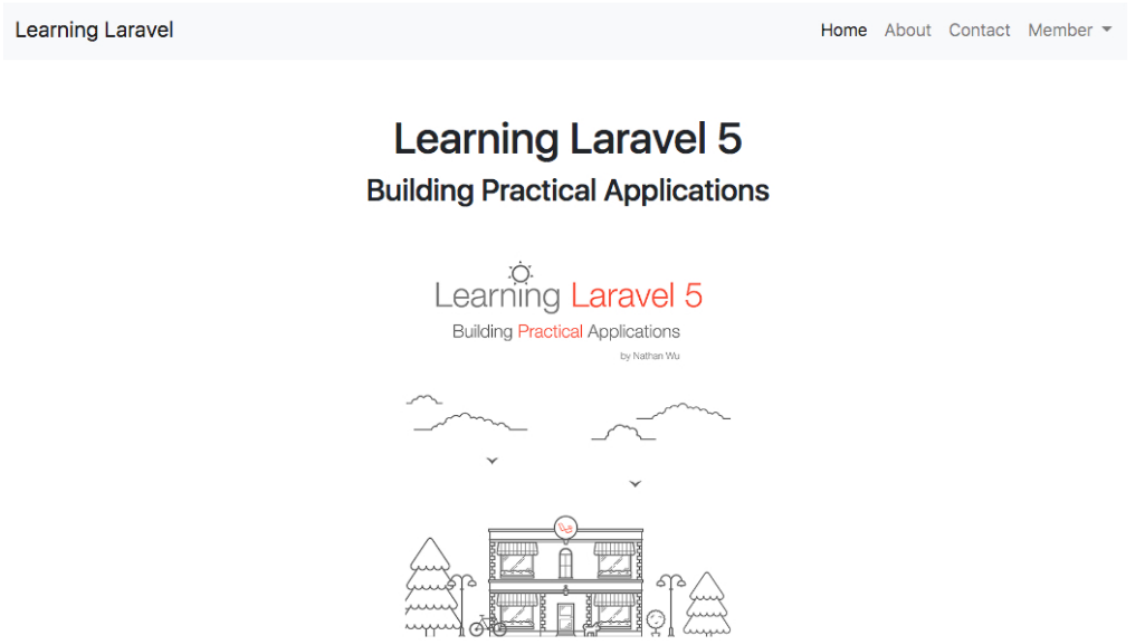
```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Larabook</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarS
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav ml-auto">
            <li class="nav-item active">
                <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/about">About</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/contact">Contact</a>
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup=
                    Member
                </a>
                <div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">
                    <a class="dropdown-item" href="/users/register">Register</a>
                    <a class="dropdown-item" href="/users/login">Login</a>
                </div>
            </li>
        </ul>
    </div>
</nav>
```



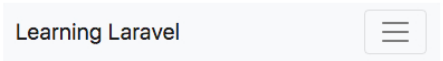
Changing the home page

Open **home.blade.php**, and update it:

```
@extends('master')
@section('title', 'Home')

@section('content')
    <div class="container">
        <div class="row banner">

            <div class="col-md-12">

                <h1 class="text-center mt-5 editContent">
                    Learning Laravel 5
                </h1>

                <h3 class="text-center mt-2 editContent">Building Practical Applications</h3>

                <div class="text-center">
                    <img src="https://learninglaravel.net/img/LearningLaravel5_cover0.png" width="302" height="391" alt="">
                </div>

            </div>

        </div>
    </div>
@endsection
```



Try to resize your browser, you'll notice that we now have a cool responsive home page.



You may try to navigate to the about and contact page to see if everything is working correctly.

Feel free to change minor things like images, contents, colors, and fonts to your liking.

Chapter 2 Summary

Good job! We now have a fully responsive template! We will use this template to build other applications to learn more about Laravel.

In this chapter, you've learned many things:

1. You've known about Laravel structure, how Laravel works and where to put the files.
2. You've learned about Laravel routes.
3. You've learned Controllers. Now you can be able to create web pages using Controllers.
4. You've known what Blade is. It's easy to create Blade templates for your next amazing applications.
5. You've known how to integrate Twitter Bootstrap, CSS, JS and apply different Bootstrap themes.
6. You've known Laravel Mix, how to install Node.js and how to create a basic Mix task.

In the next chapter, we will learn how to create a basic **CRUD** (Create, Read, Update, Delete) application to learn more about Laravel's features.

Chapter 2 Source Code

You can view and download the source code at:

> Learning Laravel 5 Book: Chapter 2 Source Code

wow,
i am the first one
1 ^ | v • Reply • Share ›

Rui M. • 16 days ago
Hi,

Have problems with npm run development

I Use windows... stucked here. Any idea about this?
thanks

> npm run development

> @ development D:\laravel57\code\experiencia
> cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js

'cross-env' is not recognized as an internal or external command,
operable program or batch file.
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! @ development: `cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js`

see more

^ | v • Reply • Share ›

learninglaravel Mod ➚ Rui M. • 16 days ago
You need to make cross-env working globally instead of having it in the project. Please check this out:

https://stackoverflow.com/q...
^ | v • Reply • Share ›

Owain Davies • 9 months ago
How do we open the project on the virtual machine from phpStorm??
^ | v • Reply • Share ›

kifftfer • a year ago
Also, at the stage "Congratulations! You now have a beautiful Material Design website!" I have the attached screen - did anybody get this to work? I also uploaded the end of chapter files and installed those and got a similar result, not the green menu bar.

^ | v • Reply • Share ›

jul2003 ➚ kifftfer • 7 months ago
Hi,

I've the same with google chrome on my macbook but it works with safari , the green menu bar is displayed. So something related to Chrome ?

Thanks for your help
^ | v • Reply • Share ›

Rydeck ➚ jul2003 • 7 months ago
For me the Material design theme didn't work either. not in Chrome not in Firefox. I proceed with the default look
^ | v • Reply • Share ›

learninglaravel Mod ➚ Rydeck • 7 months ago
Hi, you have to use v0.5.10. The current version of Material Design supports Bootstrap 4 (the new version of Bootstrap).
^ | v • Reply • Share ›

Peter Manoukian ➚ learninglaravel • 6 months ago
Hello, is this the full ebook? or there is a more larger ebook:
https://vuejsbook.com/books...
Please email me if there is a larger ebook on:
absolute_wizard@hotmail.com
^ | v • Reply • Share ›

learninglaravel Mod ➚ kifftfer • a year ago
You may download the source code and take a look at it to find out the problems :)
Reply • Share ›

kifftfer ➚ learninglaravel • a year ago
Thanks again! I did that and got the same result, that's why I posted - was curious if folks found the chapter worked as shown. But will try again.
Reply • Share ›

kifftfer • a year ago
"Open master.blade.php, modify its content to look like this:" "https:" is missing from the first three stylesheet urls.
Reply • Share ›

learninglaravel Mod ➚ kifftfer • a year ago
Hi, we use //, which means that it will accept http and https.
Reply • Share ›

kifftfer ➚ learninglaravel • a year ago
Thank you! I'm new to all this, much appreciated!
Reply • Share ›

Kapil • a year ago
Great stuff. Responsiveness is not functioning well.
Reply • Share ›

Wallace Belem Teixeira ➚ Kapil • a year ago
I've noticed that as well.
Luckily, I knew why it wasn't functioning as expected. That's because there is a specific meta tag missing.

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

This piece of code can be found in the "Bootstrap getting started" section.
1 Reply • Share ›

learninglaravel Mod ➚ Wallace Belem Teixeira • a year ago
That's right. We have to include that meta tag. Thank you.
Reply • Share ›

Wallace Belem Teixeira ➚ learninglaravel • a year ago
My pleasure.
33 Reply • Share ›