CS418/518

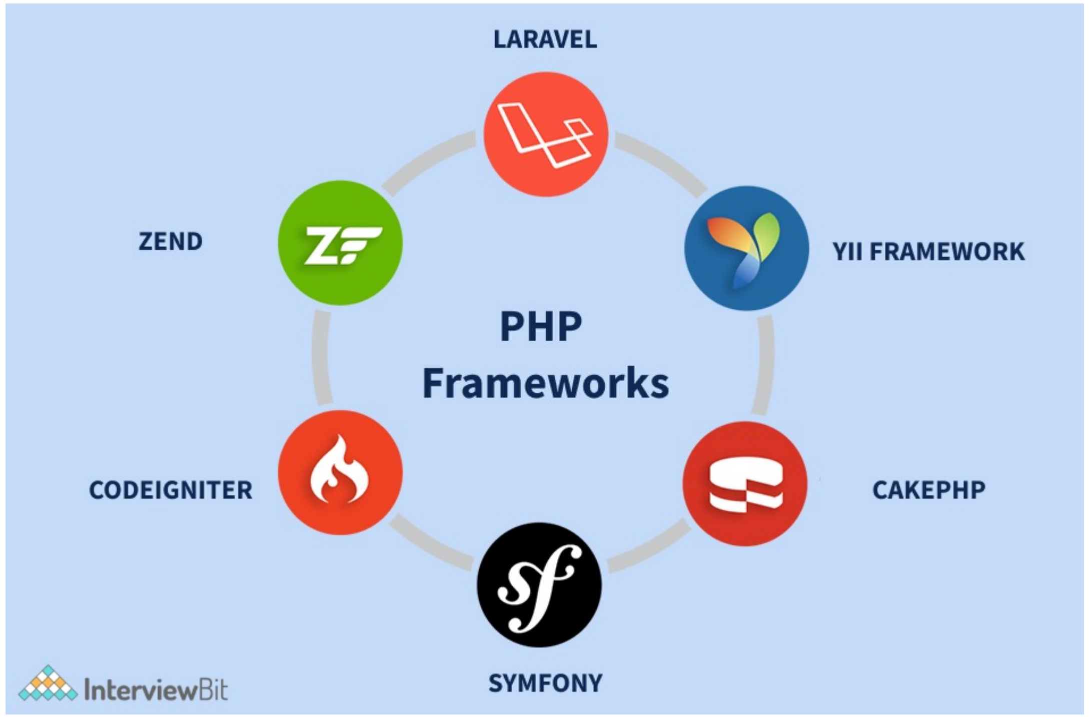
Jian Wu

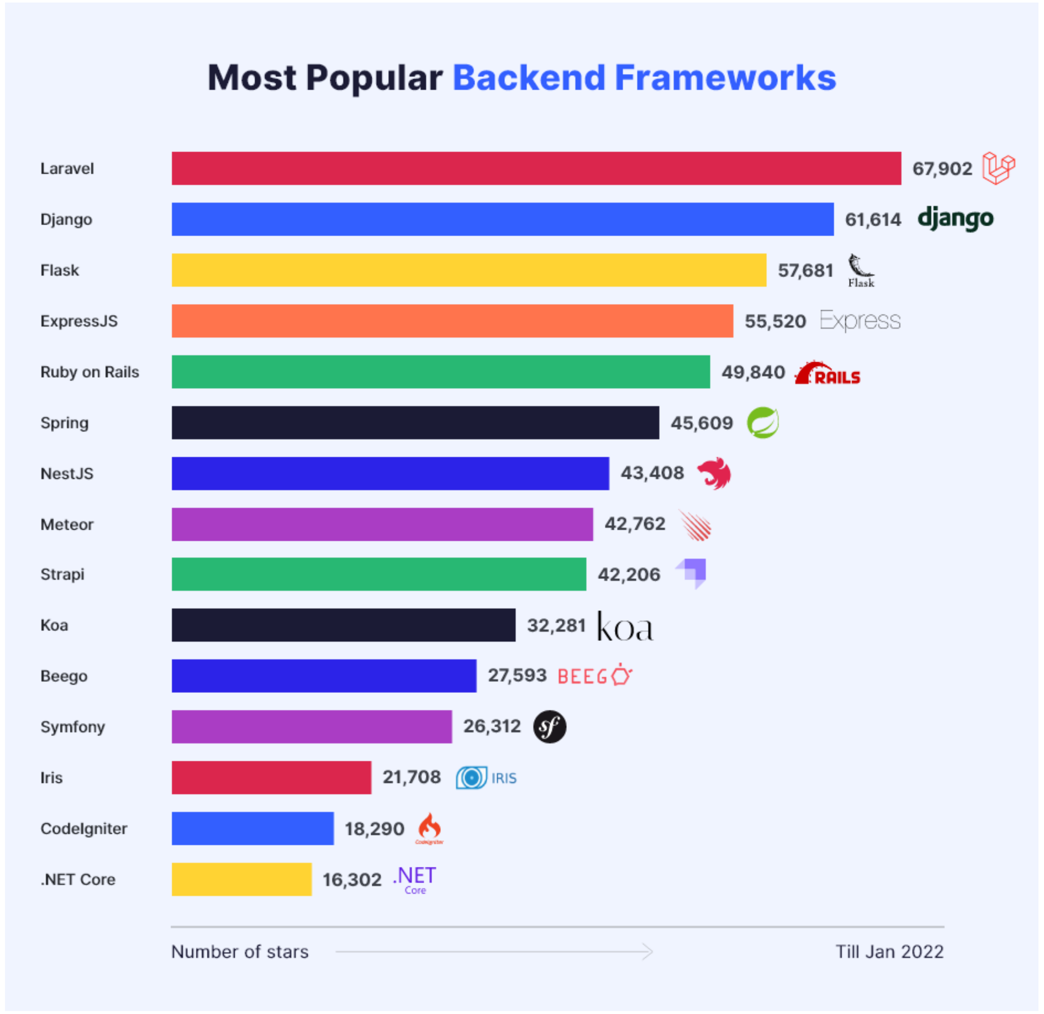
Laravel Frontend

What is Laravel

- Laravel is MVC PHP framework created by Taylor Otwell in 2011
- Free open-source license with many contributors worldwide
- One of the best frameworks together with Symfony and CodeIgniter
- Has powerful features, saving us time
- Uses Symfony packages

PHP Framework Comparison Chart





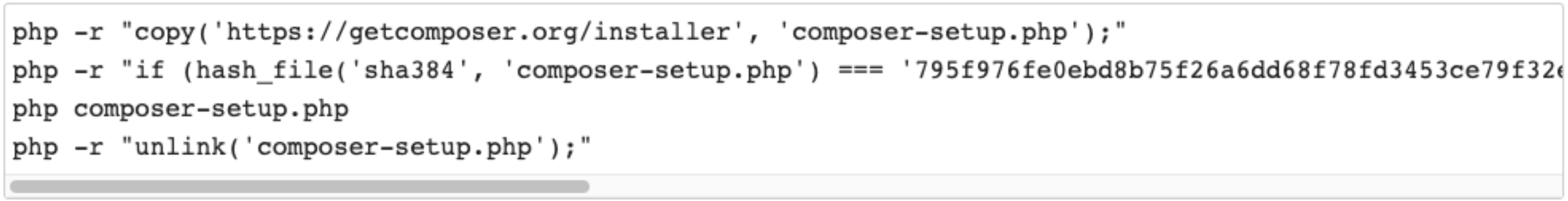
https://webmobtech.com/blog/why-choose-laravel-as-the-best-php-framework/

Features of Laravel

- Eloquent ORM (Object-Relational Mapping)
- Query builder – helps you to build secured SQL queries
- Restful controllers – provides a way for separating the different HTTP requests (GET, POST, DELETE, etc.)
- Blade template engine – combines templates with a data model to produce views
- Migrations – version control system for databases
- Database seeding –populate database tables with test data used for testing
- Pagination – easy to use advanced pagination functionalities
- Form security – provides CSRF token middleware, protecting all the forms

Installation – install composer first

- Go to this page and follow the instruction to install composer
  - https://getcomposer.org/download/
- For most people (if not all), you may copy and paste the following piece of code and composer should automatically install. (please copy from the original website)

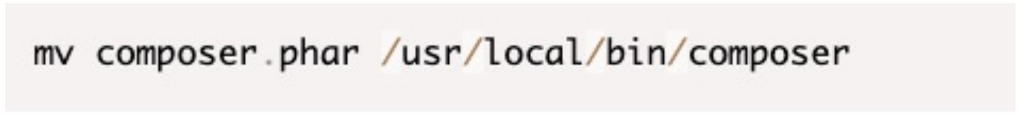


```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === '795f976fe0ebd8b75f26a6dd68f78fd3453ce79f32e
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

- You should see a composer.phar file, which is a PHP archive file.

Install composer

- Next: follow the instructions under “Globally” section on this page to place the PHAR file so it is accessible globally: https://getcomposer.org/doc/00-intro.md



```
mv composer.phar /usr/local/bin/composer
```

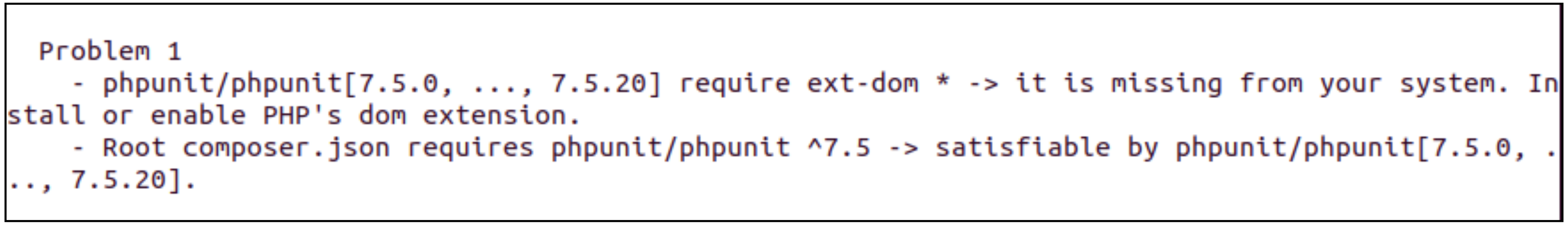
- Using composer. In your shell (Command Prompt in Windows or Terminal in MacOS), use the “composer –V” to verify whether composer is installed.
  - `composer -V`
  - `Composer version 2.6.3 2023-09-15 17:20:17`

Install Laravel

- Now check your PHP version by going to http://localhost/info.php , use the table in the Wikipedia page to find out which version of Laravel should be installed. https://en.wikipedia.org/wiki/Laravel
- Two methods to install Laravel
- **Method 1: install using composer**
- Go to `/var/www/html` and run the following command
  - `mkdir laravel //`use `sudo` if necessary
  - `cd laravel`
- Create the project called `blog` and install the project using composer
  - `composer create-project --prefer-dist laravel/laravel:^7.0 blog`
  - Replace 7.0 with your supportive Laravel version.

Troubleshoot 1

- If you see an error



```
  Problem 1
    - phpunit/phpunit[7.5.0, ..., 7.5.20] require ext-dom * -> it is missing from your system. In
stall or enable PHP's dom extension.
    - Root composer.json requires phpunit/phpunit ^7.5 -> satisfiable by phpunit/phpunit[7.5.0, .
.., 7.5.20].
```

- Run this command to install the package php-xml on your VM
  - `sudo yum install php-xml`

Troubleshoot 2

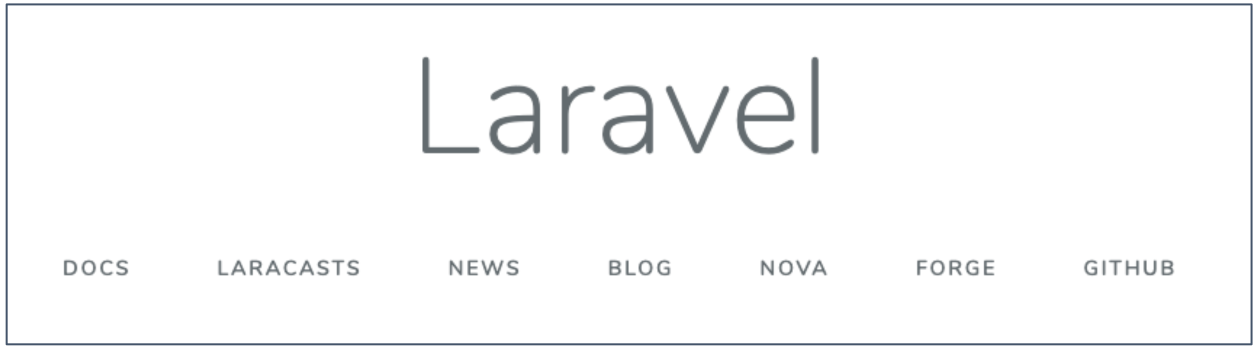
- If you see an error

```
The stream or file "/var/www/html/storage/logs/laravel.log" could not be opened: failed to open stream: Permission denied ...
```

- This is because the user to install Laravel (root) is different from the user who installed the composer (you) or they are not in the same group. To solve this problem, you can put your username to the root group. Do this under root:
  - `usermod -aG root your_username`
- Then change the permissions of all files and folders recursively under the blog project folder. Do this under the laravel folder:
  - `chmod -R 755 blog`

Laravel Deployed

- You should have a website up at
  - `http://localhost/laravel/blog/public/index.php`



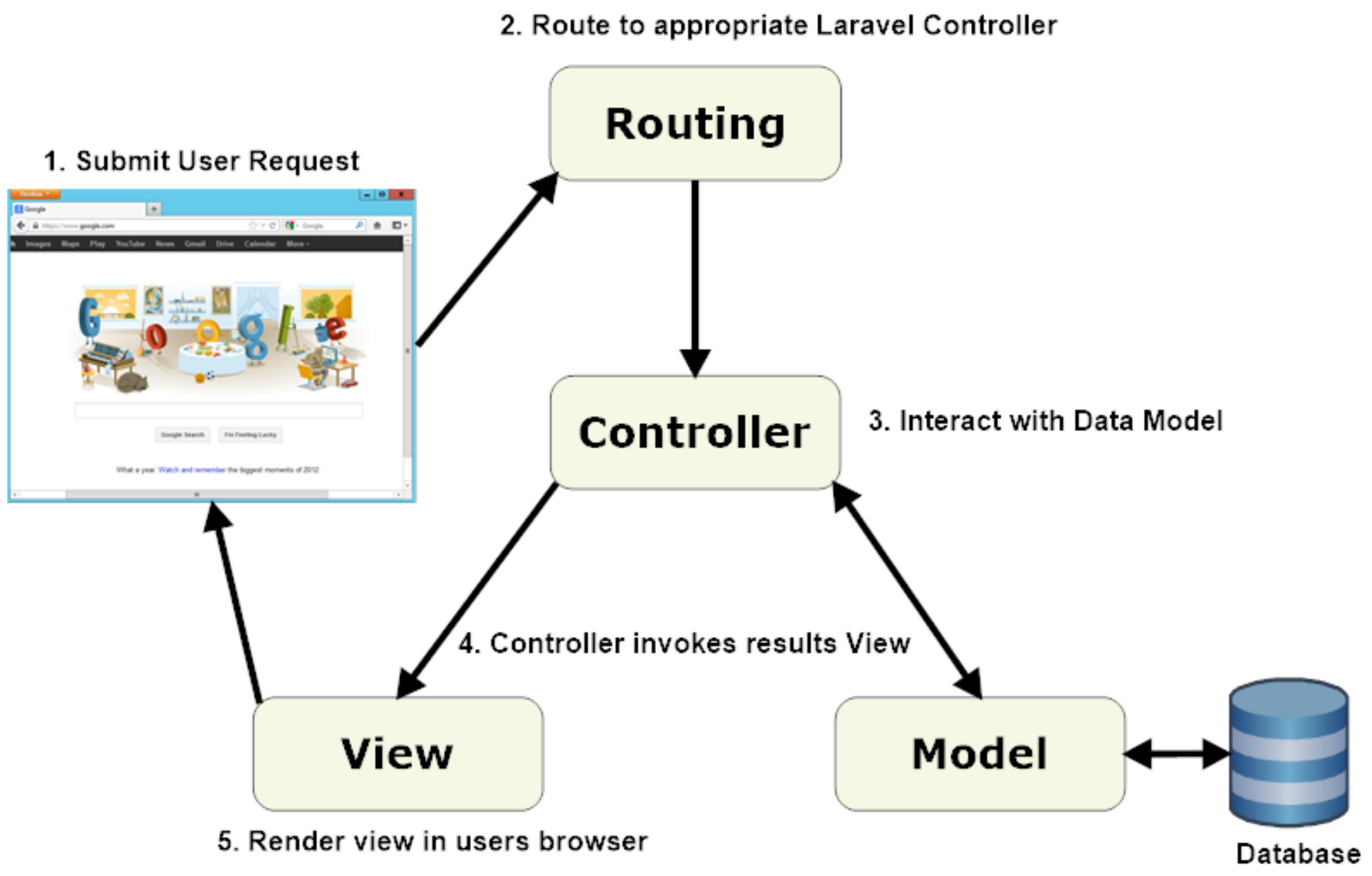
Install Laravel Method 2: using artisan

- To install Laravel, switch to your user account and create a directory under your home directory such as `~/myclasses/cs418518/laravel` and run the command
  - `composer create-project --prefer-dist laravel/laravel:^7.0 blog`
  - The `blog/` folder should be created.
  - `cd` into `blog`/
- Then run `php artisan serve`. This will start a new web server, so your website can be deployed at http://127.0.0.1/8000. You may run this command inside a `screen` session.
- This will start the PHP built-in development server. This supports PHP 5.4+.
- Optionally, you may deploy it at other port, such as 8080
  - `php artisan serve --port=8080`

Other Installation Options

- Laravel Homestead https://laravel.com/docs/8.x/homestead
  - Change the version number (8.x) to your version number.
  - Basically, a virtual machine
  - For more advanced users
- Laravel Valet https://laravel.com/docs/8.x/valet
  - Valet is a Laravel development environment for Mac minimalists.
  - No Windows users
  - Laravel Valet configures your Mac to always run Nginx in the background when your machine starts.

Architecture

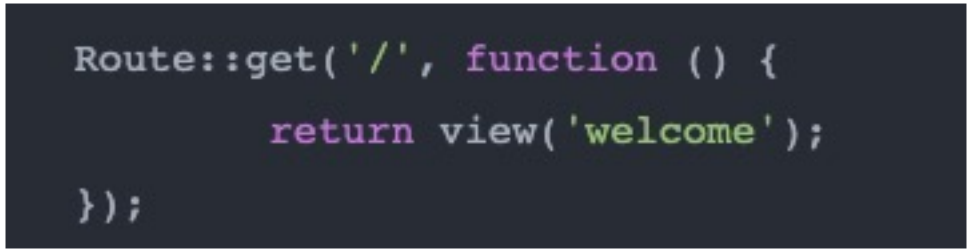

2. Route to appropriate Laravel Controller
Routing
1. Submit User Request
Controller
3. Interact with Data Model
4. Controller invokes results View
View
Model
Database
5. Render view in users browser

Routing

- Routing is a way of creating a request URL for your application.
- The best thing about the Laravel routing is you are free to define your routes the way you want it to look alike.
- Routing enables use of URLs that are descriptive of the user actions and are more easily understood by the users.
- For example, instead of
  - http://myapplication/Users.php?id=1
  - we can
  - http://myapplication/Users/1

Routing in Laravel

An example of `web.php`

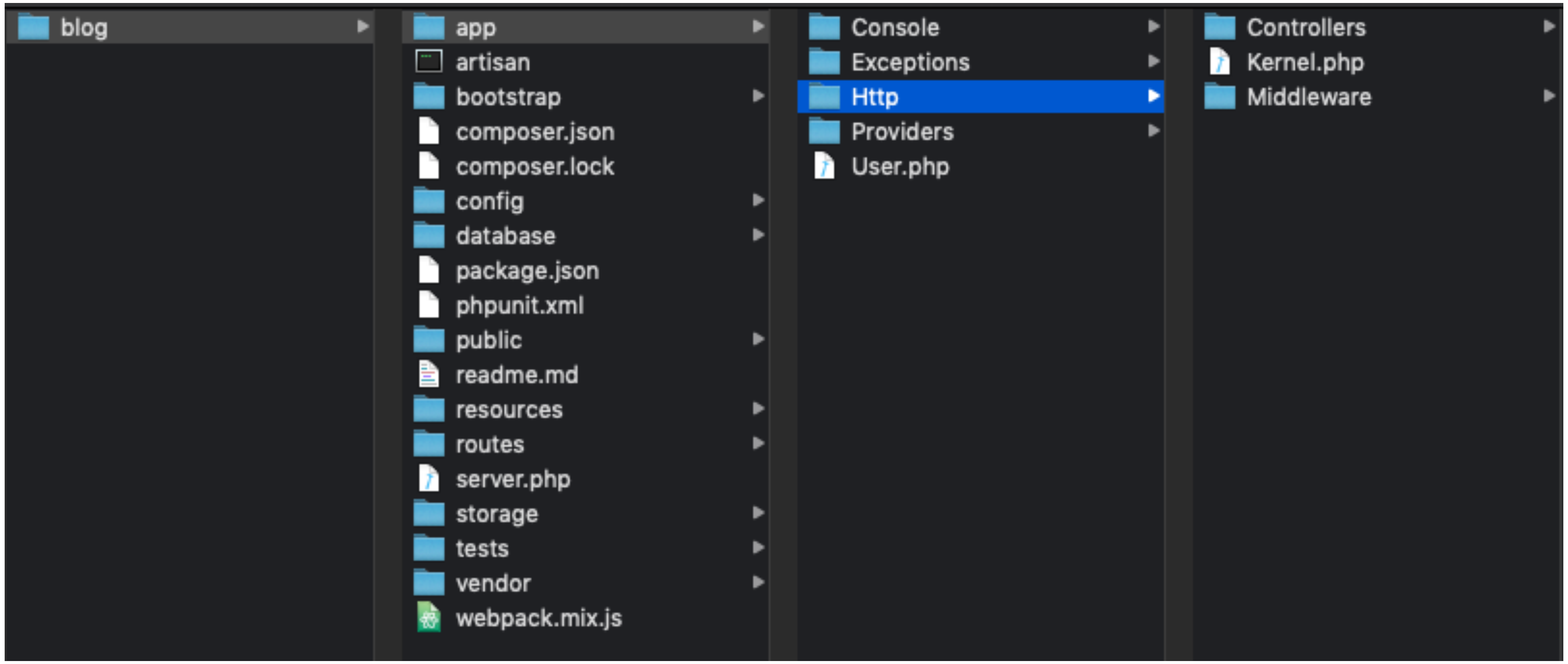


```
Route::get('/', function () {
        return view('welcome');
});
```

- In Laravel, all requests are mapped with the help of routes, all routes are created inside **routes** folder. There are two important files:
  - `web.php:` define application related routes
  - `api.php:` define API related routes
- The above code example defines a route which will receive a **/** request and will return **welcome** view.
- Defining routes in Laravel is simple. You start a new route with `Route::` façade followed by the request type you want to assign to that route.
- The function that processes the request will come after.
- Route methods or request types in Laravel: **get, post, put, delete, patch, options**

Folder Structure – Directories

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



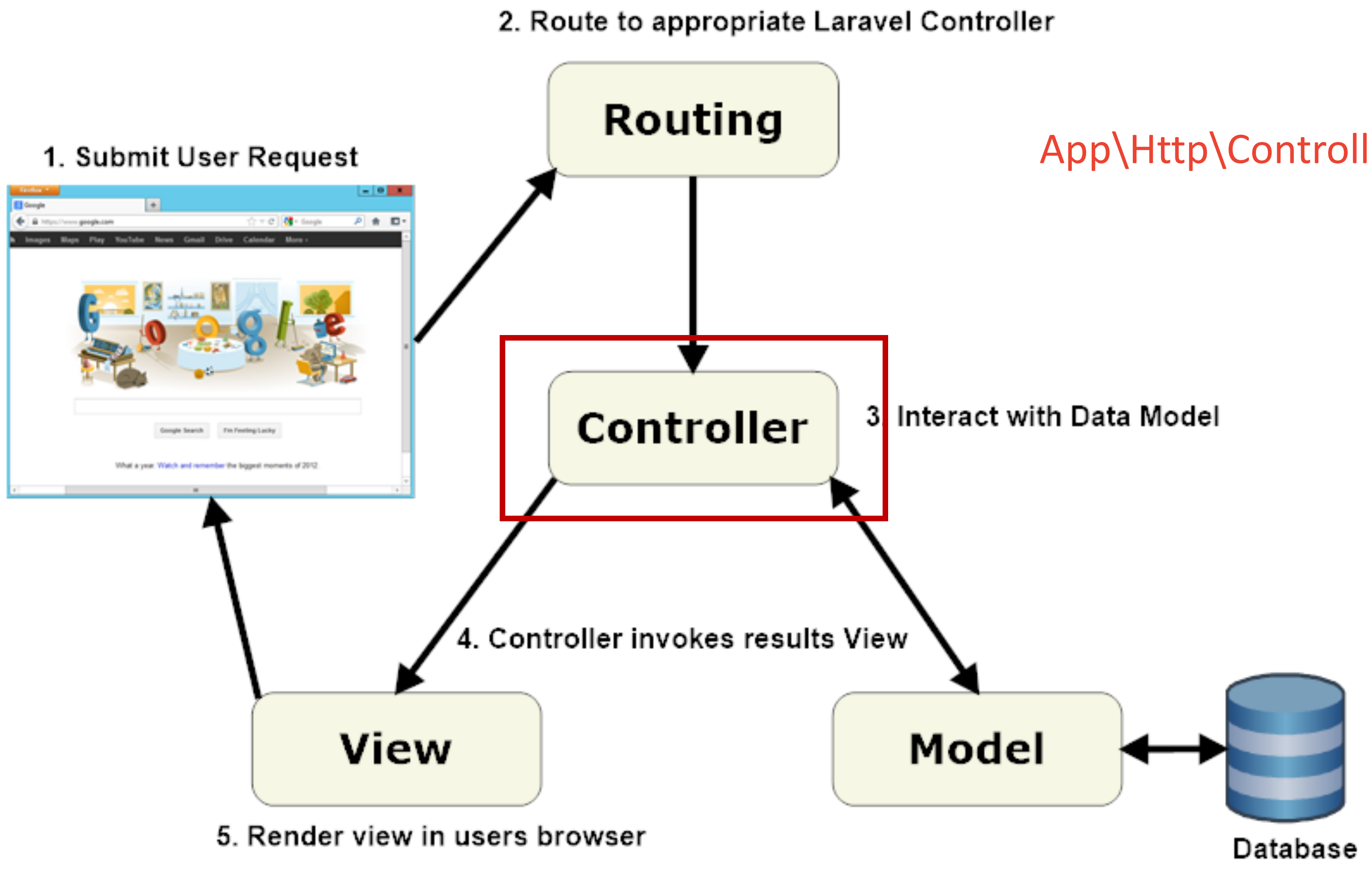
```
blog
```

```
app
artisan
bootstrap
composer.json
composer.lock
config
database
package.json
phpunit.xml
public
readme.md
resources
routes
server.php
storage
tests
vendor
webpack.mix.js
```

```
Console
Exceptions
Http
Providers
User.php
```

```
Controllers
Kernel.php
Middleware
```

Controller


2. Route to appropriate Laravel Controller
Routing
1. Submit User Request
Controller
3. Interact with Data Model
4. Controller invokes results View
View
Model
Database
5. Render view in users browser

App\Http\Controllers\Controller

Laravel model – Eloquent ORM

- Models allow you to query for data and insert new records into database table.
- Eloquent ORM provides a simple implementation for working with your database.
- Each database table has a corresponding "Model" used to interact with that table.

ClassName	Singular of table name	`protected $table='custom_name'`
Primary key	id	`protected $primaryKey`
Timestamp	created_at, updated_at	`protected $timestamp = false`
Guarded	array of fields name	`protected $guarded = array('id', 'password')`
Fillable	array of fields name	`protected $fillable = array('id', 'password')`

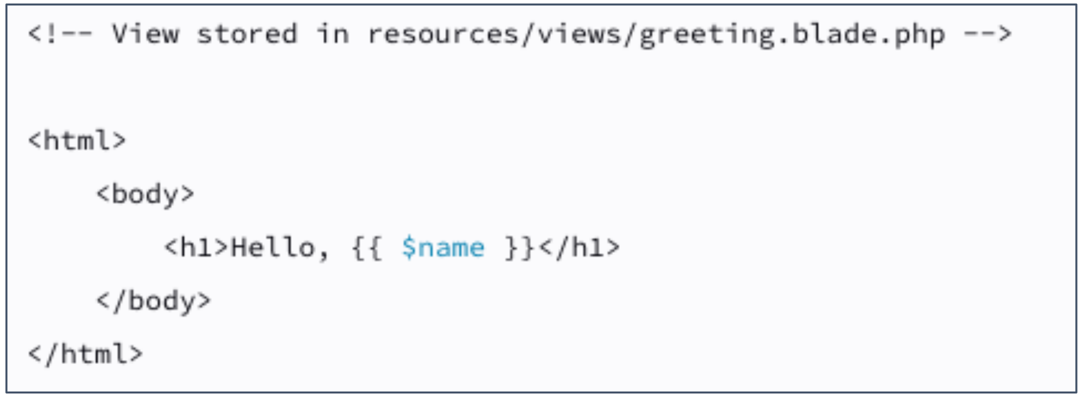
Laravel Model – Eloquent ORM

- Pros
  - Extend from fluent
  - Model relationships
  - Easy to use
  - Code readability

- Cons
  - Handle complex SQL queries
  - Increase SQL execute time
  - Learning time

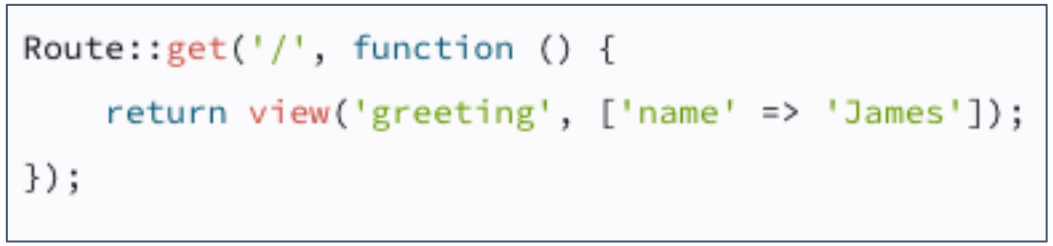
Laravel view

- Views contain the HTML served by your application and separate your controller/application logic from your presentation logic.
- Views are stored in the `resources/views` directory. A simple view might look something like this:



```
<!-- View stored in resources/views/greeting.blade.php -->

<html>
    <body>
        <h1>Hello, {{ $name }}</h1>
    </body>
</html>
```

This view is stored at `resources/views/greeting.blade.php`, we may return it using the global view helper like so:

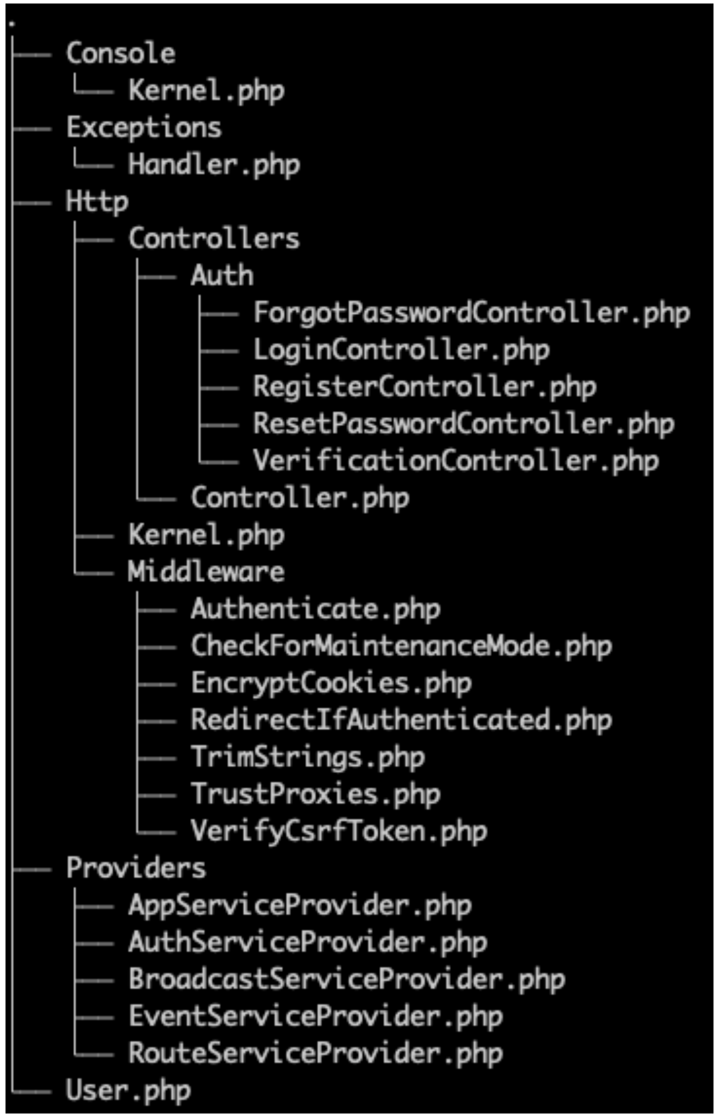


```
Route::get('/', function () {
    return view('greeting', ['name' => 'James']);
});
```

1. The first argument (‘greeting’): the name of the view file in the `resources/views` directory.
2. The second argument ([‘name’=>’James’]): an array of data that should be made available to the view. In this case, we are passing the `name` variable,

app/ Directory

- Contains the core code of your application. A variety of other directories will be generated inside the app directory as you use the `make artisan` commands to generate class.
- The exact layout may vary depending your Laravel version.

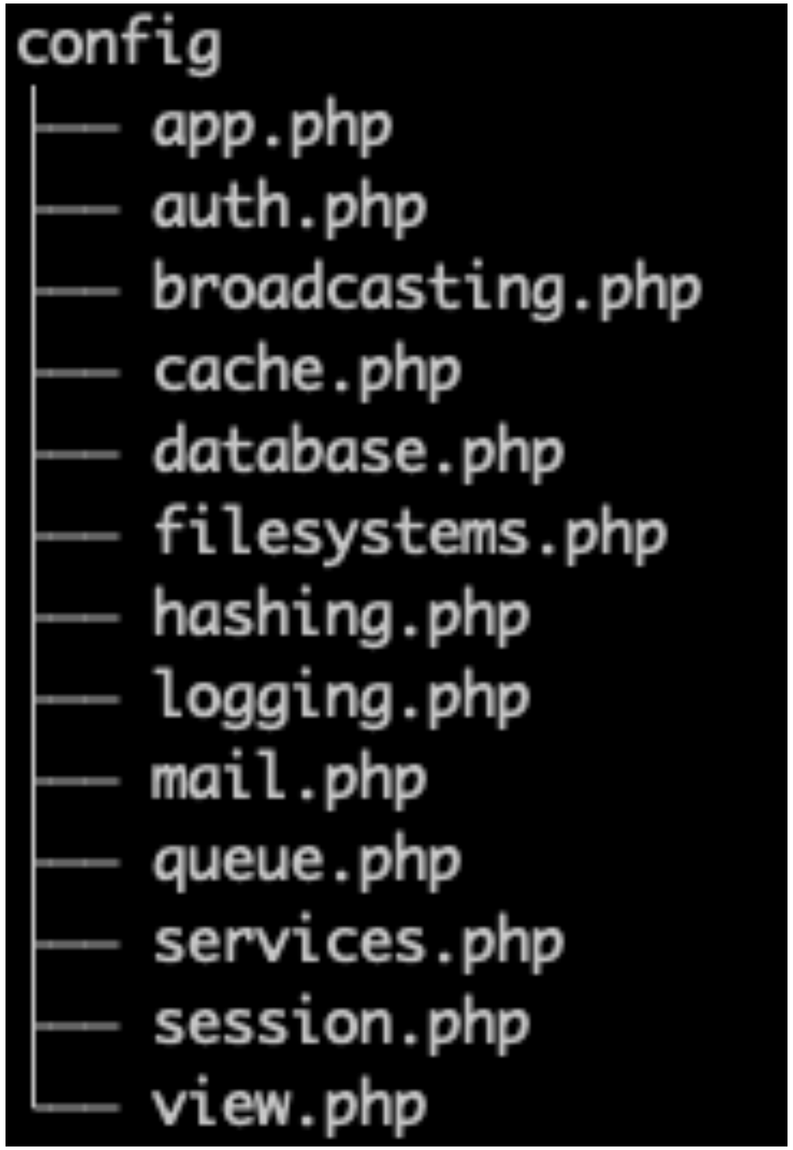


```
.
├── Console
│   └── Kernel.php
├── Exceptions
│   └── Handler.php
├── Http
│   ├── Controllers
│   │   ├── Auth
│   │   │   ├── ForgotPasswordController.php
│   │   │   ├── LoginController.php
│   │   │   ├── RegisterController.php
│   │   │   ├── ResetPasswordController.php
│   │   │   └── VerificationController.php
│   │   └── Controller.php
│   ├── Kernel.php
│   └── Middleware
│       ├── Authenticate.php
│       ├── CheckForMaintenanceMode.php
│       ├── EncryptCookies.php
│       ├── RedirectIfAuthenticated.php
│       ├── TrimStrings.php
│       ├── TrustProxies.php
│       └── VerifyCsrfToken.php
├── Providers
│   ├── AppServiceProvider.php
│   ├── AuthServiceProvider.php
│   ├── BroadcastServiceProvider.php
│   ├── EventServiceProvider.php
│   └── RouteServiceProvider.php
└── User.php
```

Laravel default tree structure in the `app` directory.

The bootstrap/ directory

- Contains the `app.php` file which bootstraps the framework.
- It is used to initialize (setting up path & environments) the framework.
- Laravel's bootstrap folder has nothing to do with the Bootstrap CSS framework



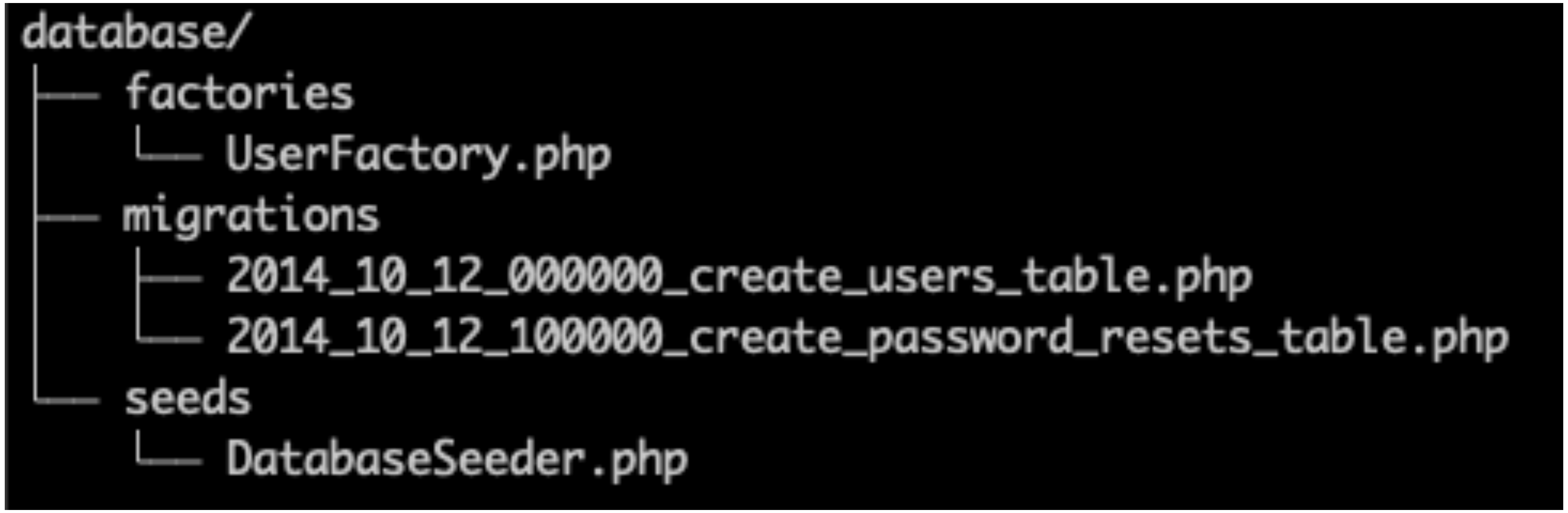
```
config
├── app.php
├── auth.php
├── broadcasting.php
├── cache.php
├── database.php
├── filesystems.php
├── hashing.php
├── logging.php
├── mail.php
├── queue.php
├── services.php
├── session.php
└── view.php
```

config/ directory

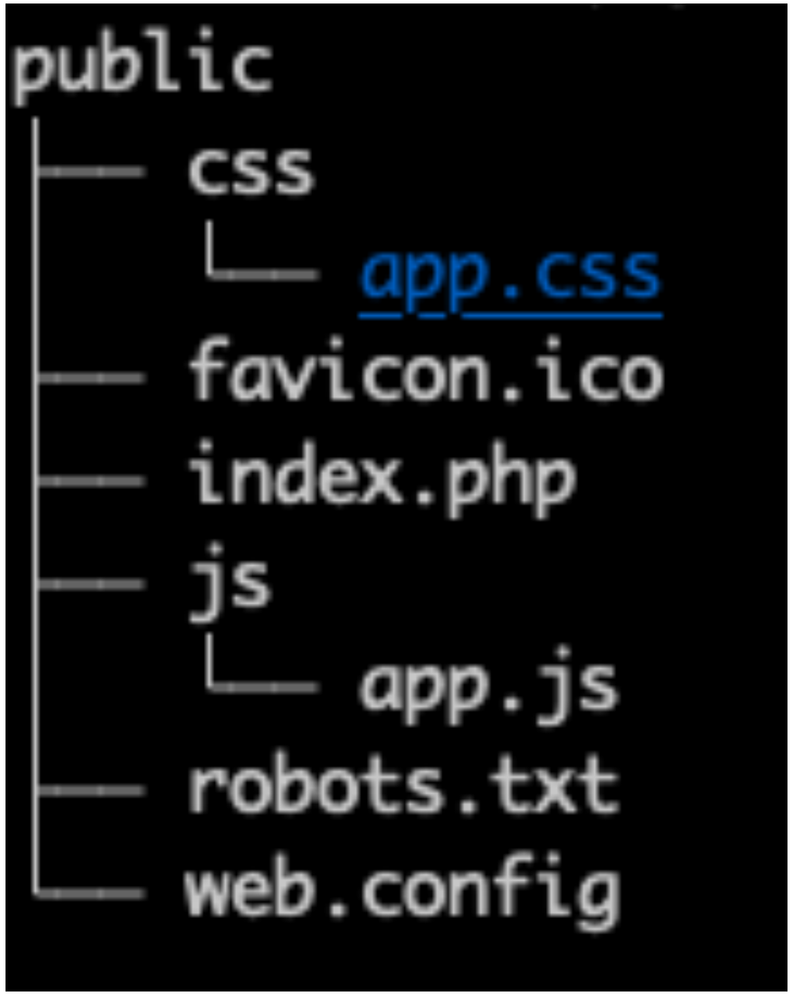
- Contains all of your application's configuration files.

database/ directory

- Contains your database migration and seeds.



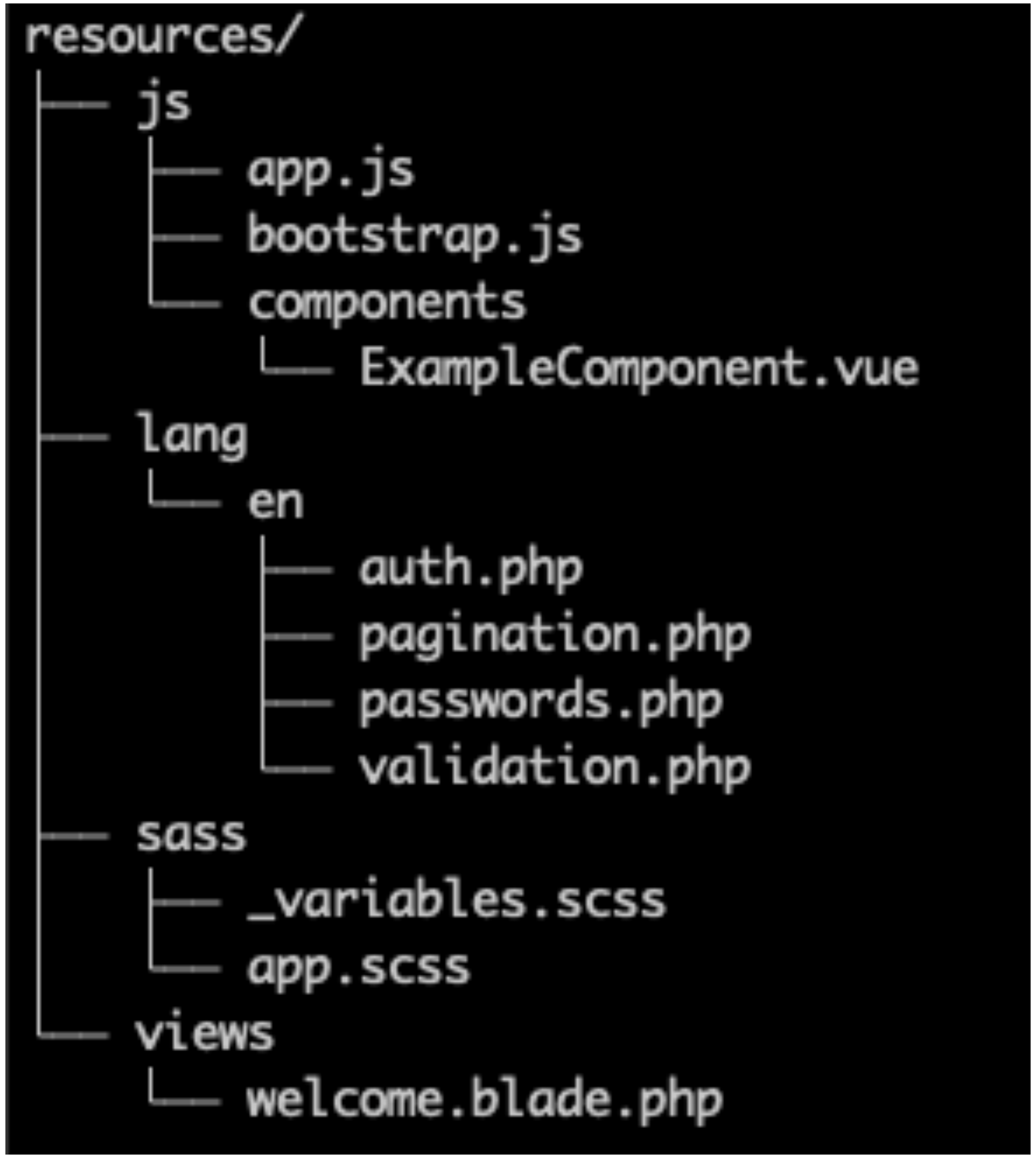
```
database/
├── factories
│   └── UserFactory.php
├── migrations
│   ├── 2014_10_12_000000_create_users_table.php
│   └── 2014_10_12_100000_create_password_resets_table.php
└── seeds
    └── DatabaseSeeder.php
```



```
public
├── css
│   └── app.css
├── favicon.ico
├── index.php
├── js
│   └── app.js
├── robots.txt
└── web.config
```

public/ directory

- Contains the `index.php` file, which is the entry point for all requests entering your application and configures autoloading.
- This directory also houses your assets such as images, JavaScript, and CSS.



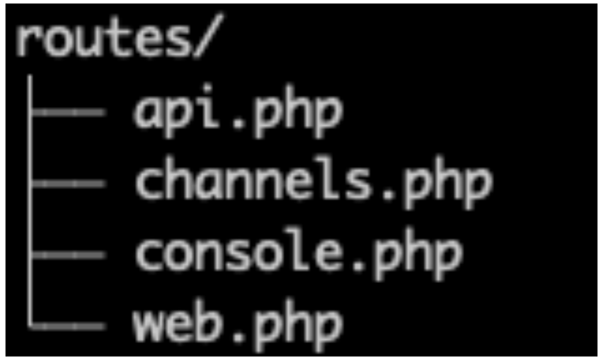
```
resources/
├── js
│   ├── app.js
│   ├── bootstrap.js
│   └── components
│       └── ExampleComponent.vue
├── lang
│   └── en
│       ├── auth.php
│       ├── pagination.php
│       ├── passwords.php
│       └── validation.php
├── sass
│   ├── _variables.scss
│   └── app.scss
└── views
    └── welcome.blade.php
```

resource/ directory

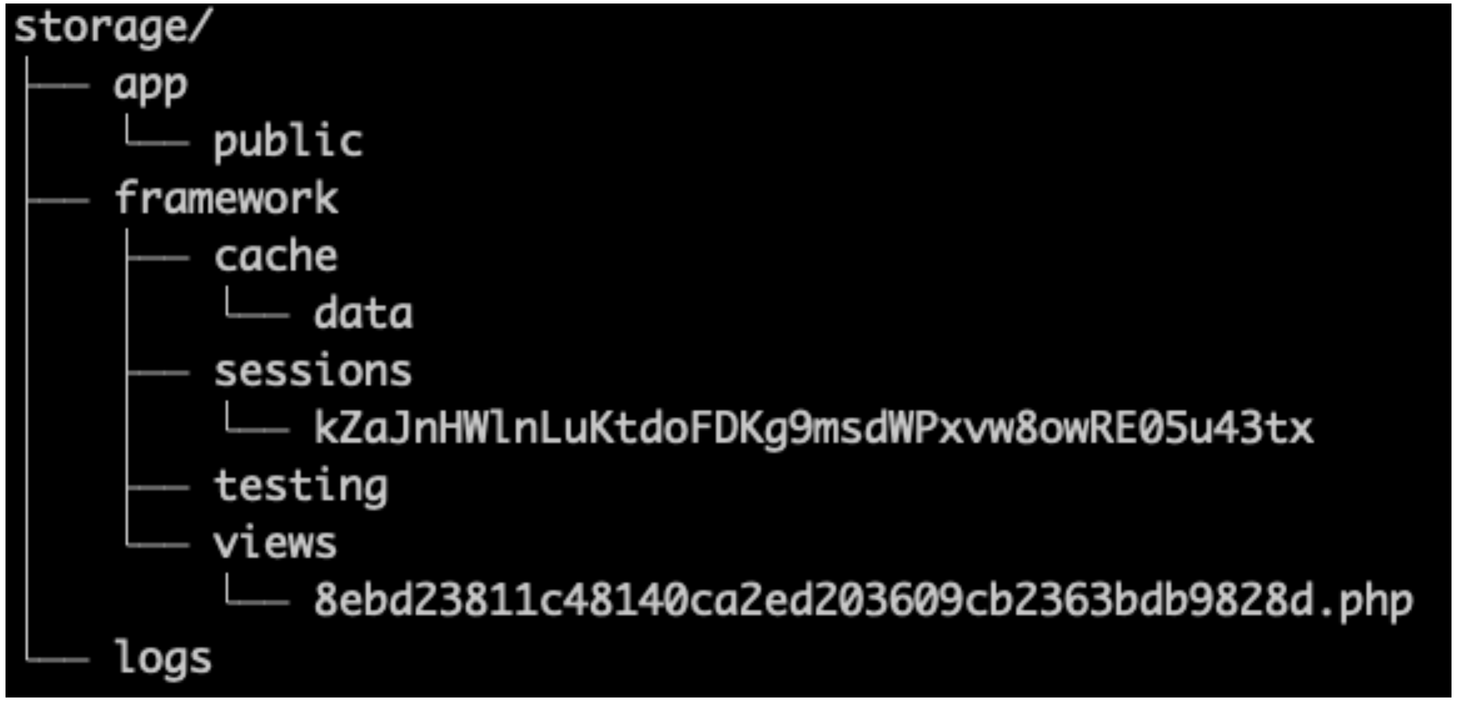
- Contains your views as well as your raw, un-compiled assets such as LESS, SASS, or JavaScript.

routers/ directory

- Contains all of the route definitions for your application.



```
routes/
├── api.php
├── channels.php
├── console.php
└── web.php
```



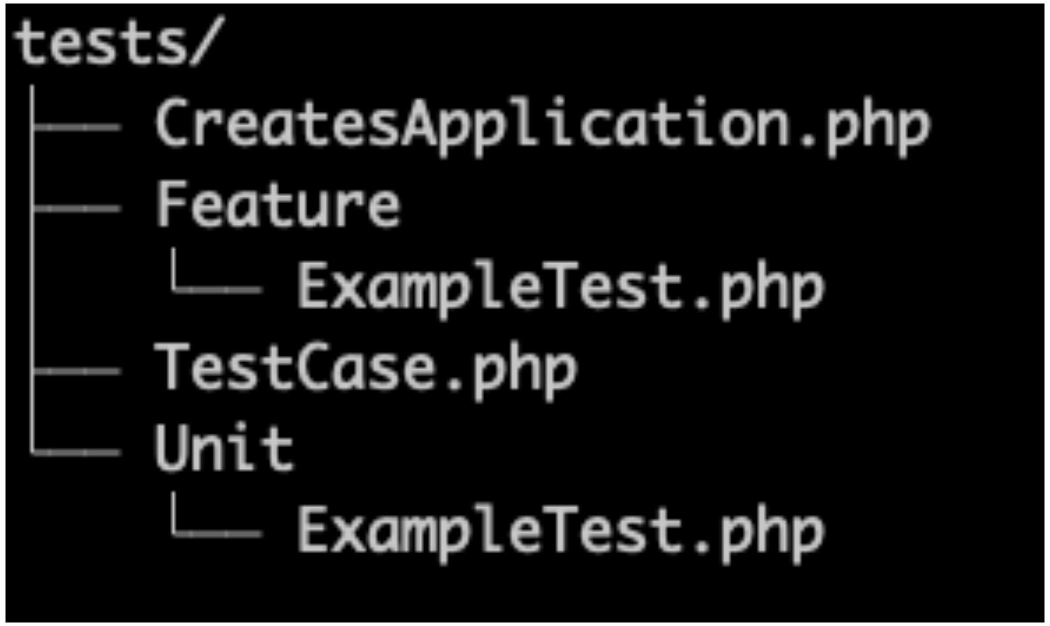
```
storage/
├── app
│   └── public
├── framework
│   ├── cache
│   │   └── data
│   ├── sessions
│   │   └── kZaJnHWlnLuKtdoFDKg9msdWPxvw8owRE05u43tx
│   ├── testing
│   └── views
│       └── 8ebd23811c48140ca2ed203609cb2363bdb9828d.php
└── logs
```

storage/ directory

- Contains your compiled
  - Blade templates
  - file based sessions
  - file caches
  - other files generated by the framework.

test/ directory

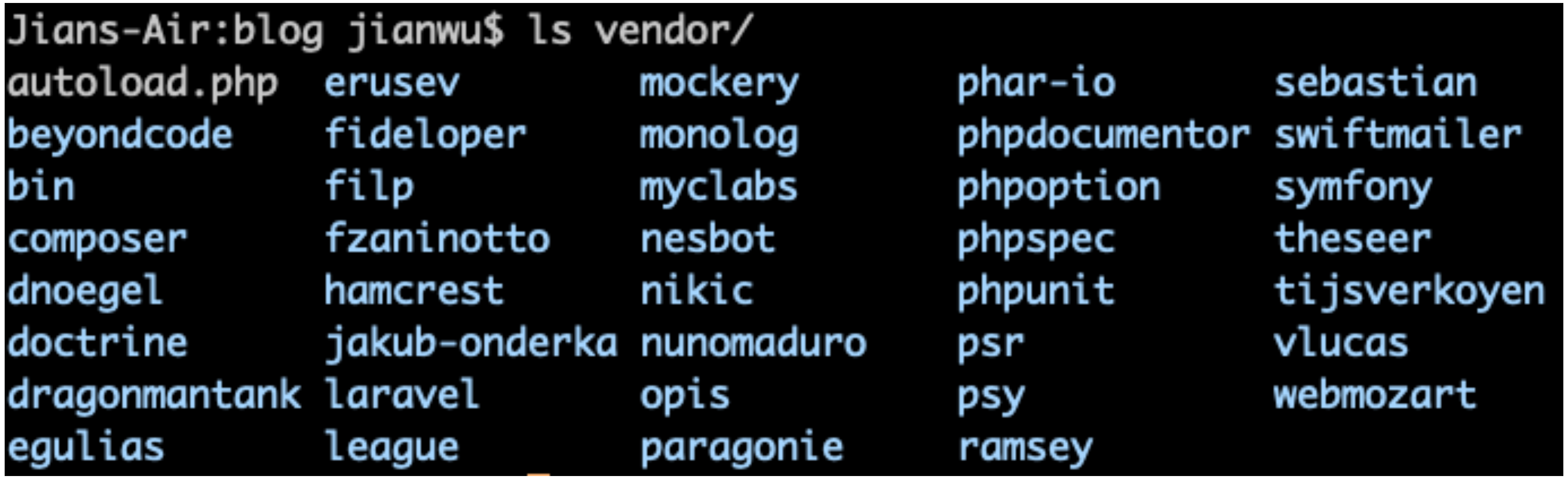
- Contains your automated tests.



```
tests/
├── CreatesApplication.php
├── Feature
│   └── ExampleTest.php
├── TestCase.php
└── Unit
    └── ExampleTest.php
```

vendor/ directory

- Contains your composer dependencies.



```
Jians-Air:blog jianwu$ ls vendor/
autoload.php  erusev        mockery       phar-io        sebastian
beyondcode    fideloper     monolog       phpdocumentor  swiftmailer
bin           filp          myclabs       phpoption      symfony
composer      fzaninotto    nesbot        phpspec        theseer
dnoegel       hamcrest      nikic         phpunit        tijsverkoyen
doctrine      jakub-onderka nunomaduro    psr            vlucas
dragonmantank laravel       opis          psy            webmozart
egulias       league        paragonie     ramsey
```

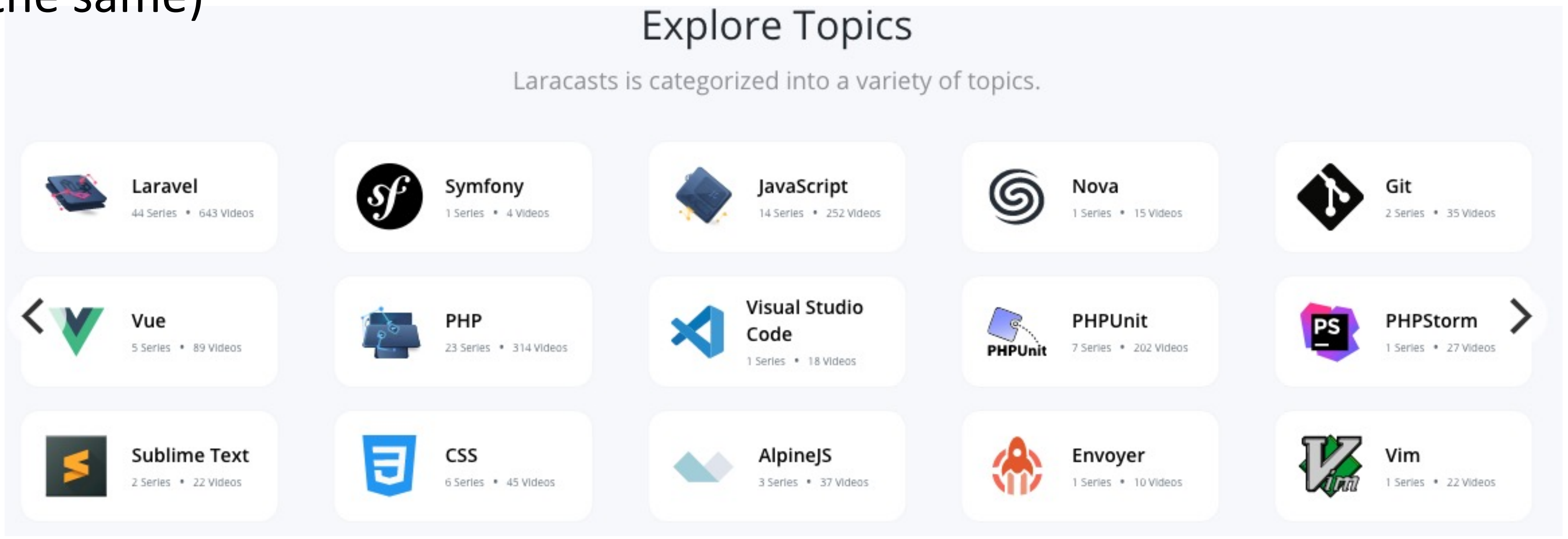
Default Laravel technologies

- **Jquery**: A cross-platform JS library designed to simplify the client-side scripting of HTML.
- **Bootstrap:** Bootstrap is the most popular HTML, CSS, and Js framework for developing responsive, mobile-first web sites
- **Axios:** A Js library used to make http requests from node.js or XMLHttpRequests from the browser and it supports the Promise API that is native to JS ES6.
- **Vue:** Popular Js framework for building user interfaces.
- **Lodash:** A Js library that helps programmers write more concise and easier to maintain Js.
- **Cross-env:** Run scripts that set and use environment variables across platforms

Laraval videos

https://laracasts.com/ (may need subscription but some are free)

https://laracasts.com/series/laravel-6-from-scratch (Laravel 6 and 7 are almost the same)



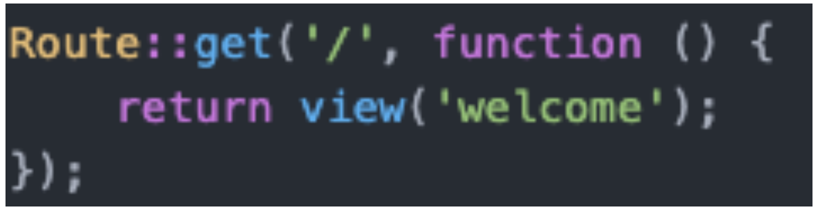
Explore Topics

Laracasts is categorized into a variety of topics.

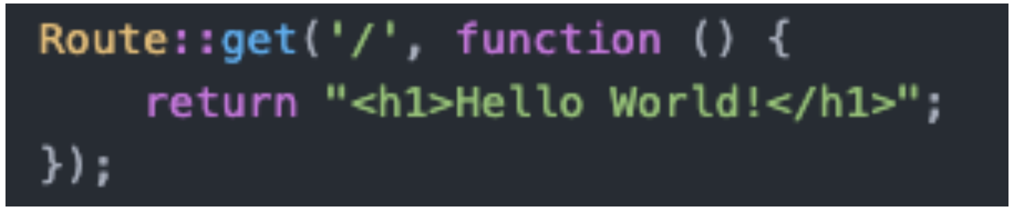
Laravel	44 Series • 643 Videos
Symfony	1 Series • 4 Videos
JavaScript	14 Series • 252 Videos
Nova	1 Series • 15 Videos
Git	2 Series • 35 Videos
Vue	5 Series • 89 Videos
PHP	23 Series • 314 Videos
Visual Studio Code	1 Series • 18 Videos
PHPUnit	7 Series • 202 Videos
PHPStorm	1 Series • 27 Videos
Sublime Text	2 Series • 22 Videos
CSS	6 Series • 45 Videos
AlpineJS	3 Series • 37 Videos
Envoyer	1 Series • 10 Videos
Vim	1 Series • 22 Videos

Experiment 1: Displaying “Hello World!”

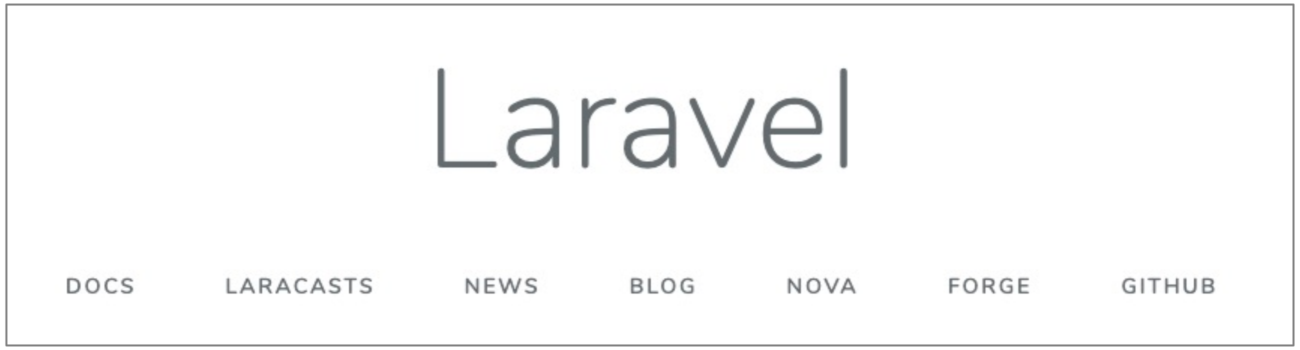
- Route -> web.php



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

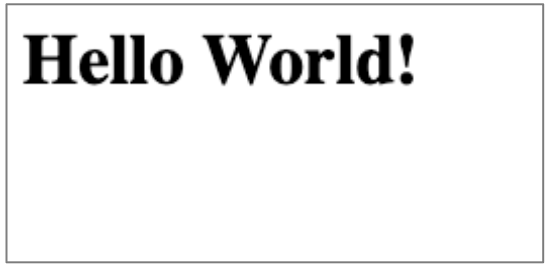


```
Route::get('/', function () {
    return "<h1>Hello World!</h1>";
});
```



Laravel

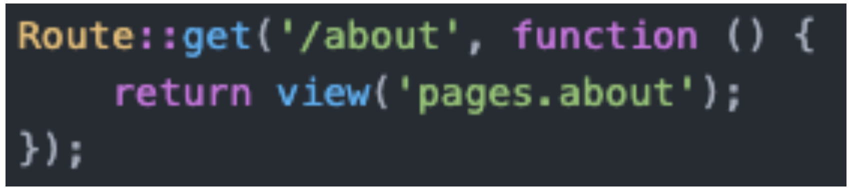
DOCS LARACASTS NEWS BLOG NOVA FORGE GITHUB



Hello World!

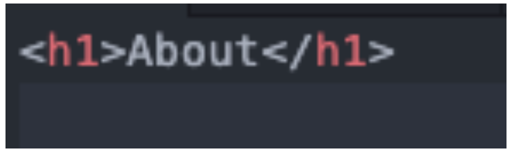
Experiment 2: Create an “About” page as a new view

- route->web.php: open a new Route statement

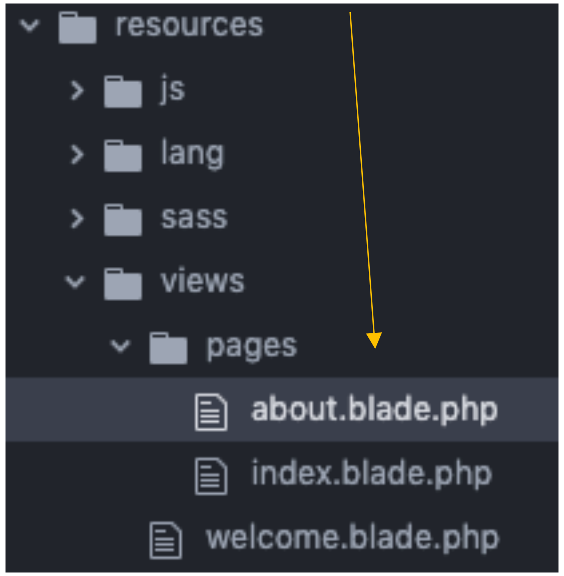


```
Route::get('/about', function () {
    return view('pages.about');
});
```

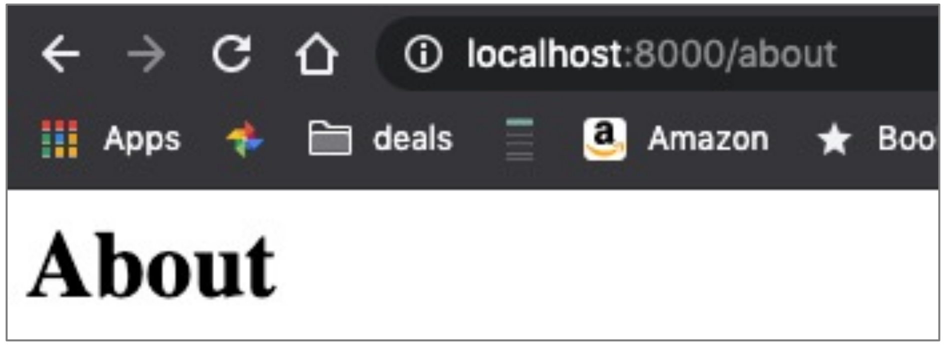
- resources->views->pages->about.blade.php



```
<h1>About</h1>
```



```
resources
  js
  lang
  sass
  views
    pages
      about.blade.php
      index.blade.php
    welcome.blade.php
```

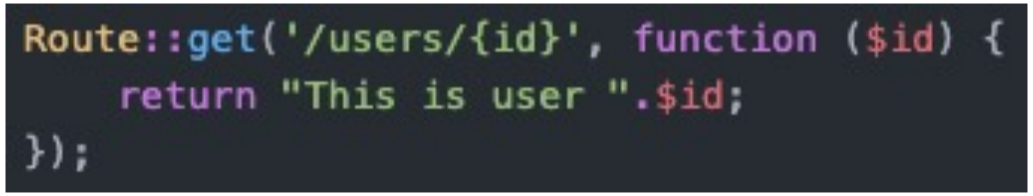


localhost:8000/about

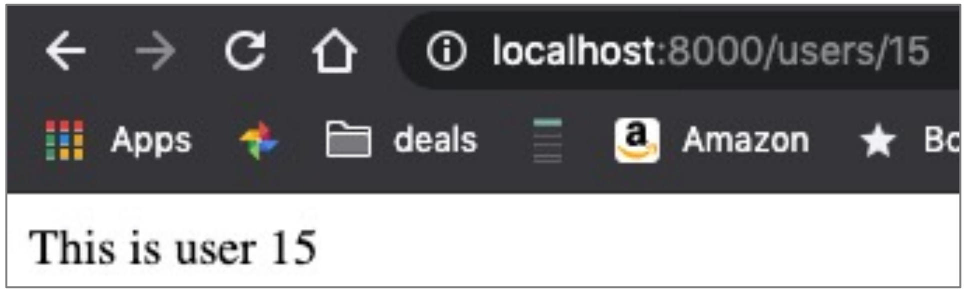
About

Experiment 3: Show a user's ID

- route->web.php: open a new Route statement



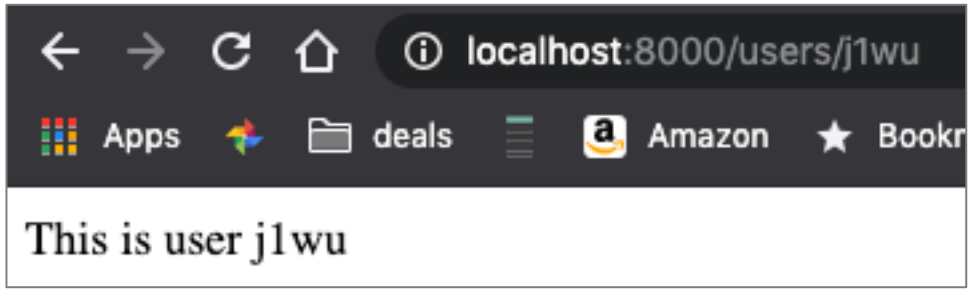
```php
Route::get('/users/{id}', function ($id) {
    return "This is user ".$id;
});
```



```
localhost:8000/users/15
```

Apps deals Amazon Bc

This is user 15



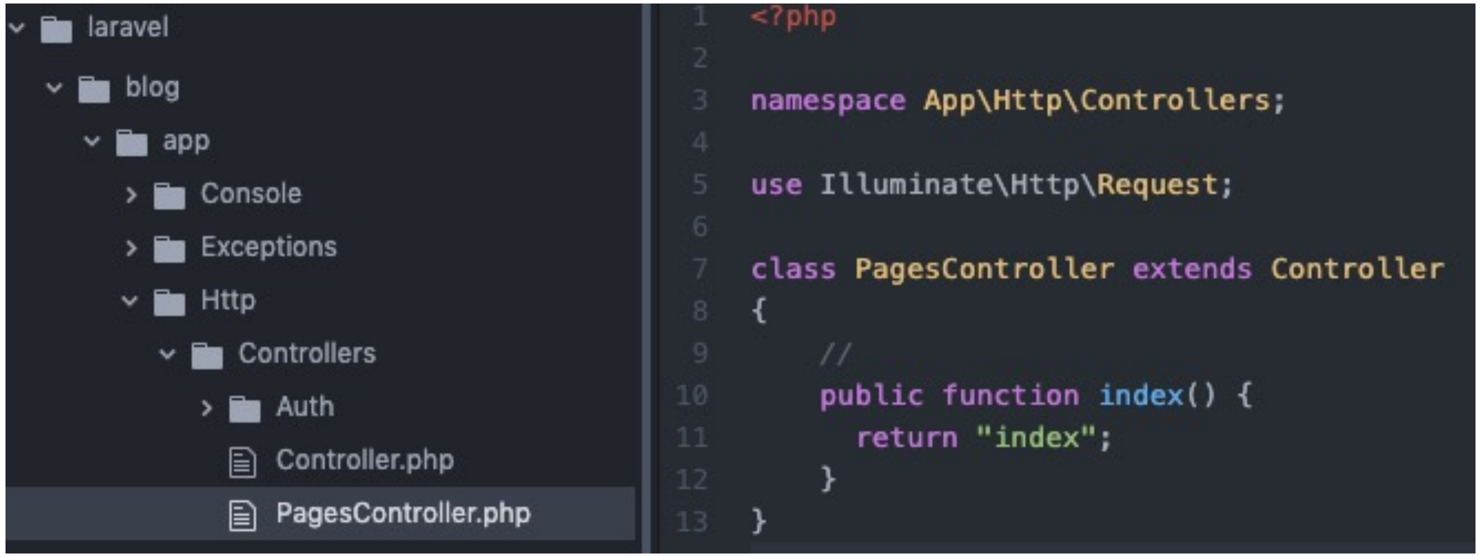
```
localhost:8000/users/j1wu
```

Apps deals Amazon Bookr

This is user j1wu

Experiment 4: Create a new controller

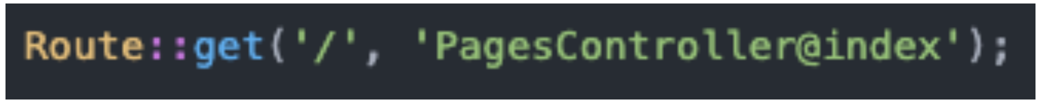
- Make a PagesController (note the camelCase naming convention)
  - `php artisan make:controller PagesController`



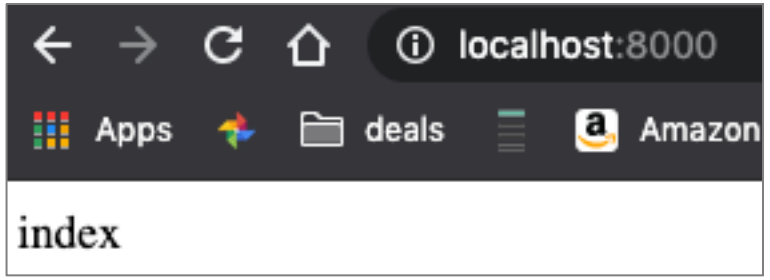
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    //
    public function index() {
      return "index";
    }
}
```

Experiment 4 (continued)

- route->web.php: revise the “/” controller, changing it to

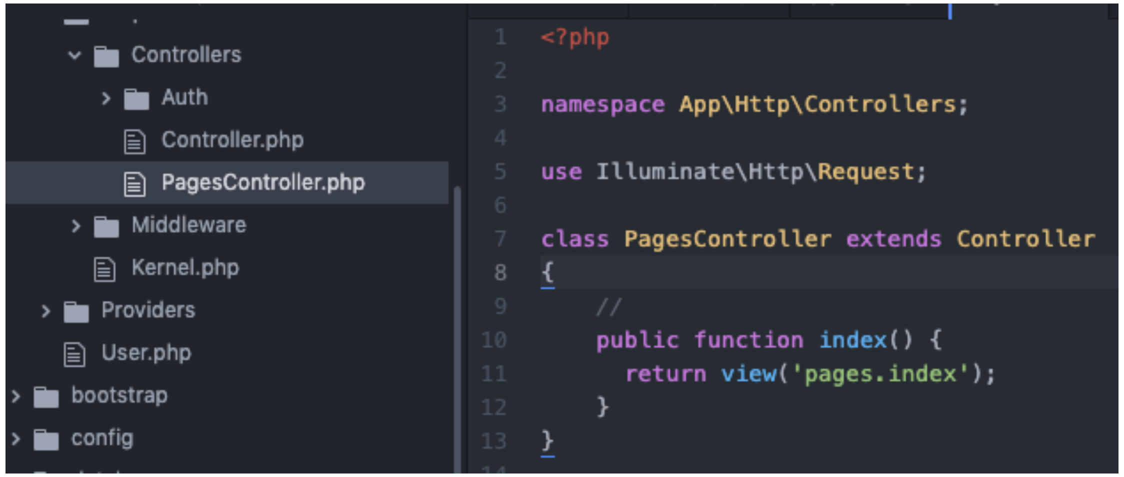


```
Route::get('/', 'PagesController@index');
```



index

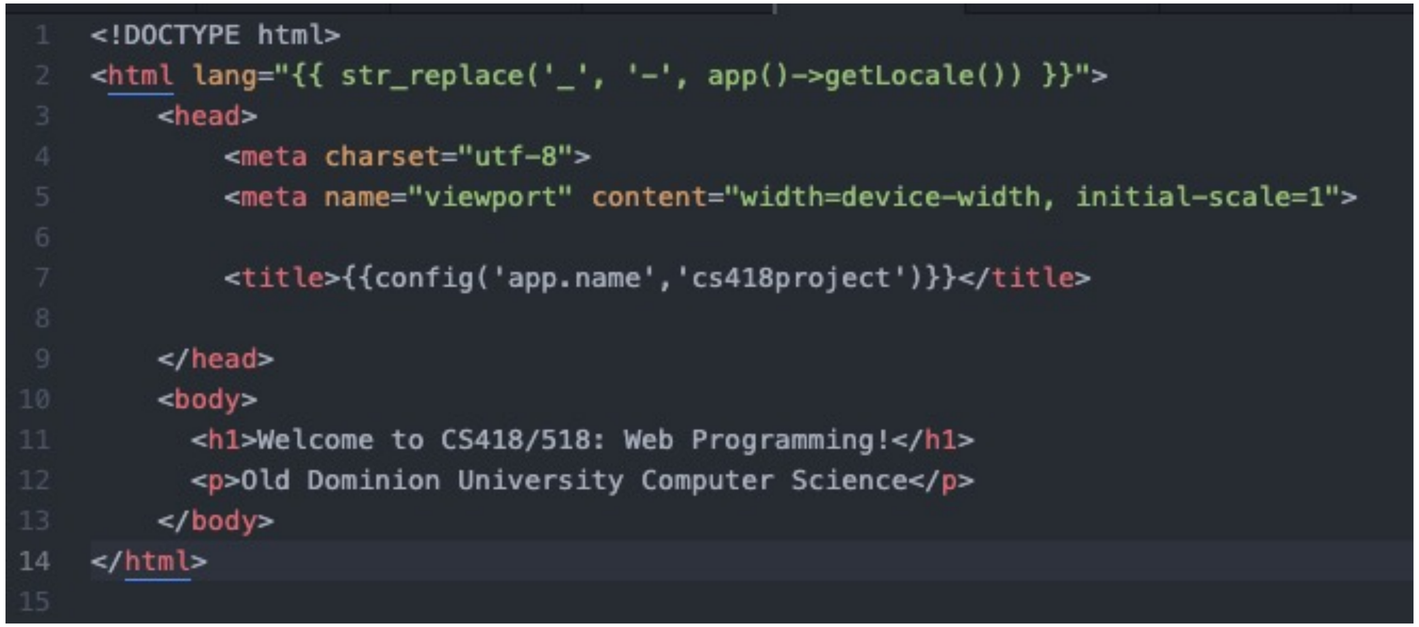
- Next: edit PagesController.php



```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    //
    public function index() {
      return view('pages.index');
    }
}
```

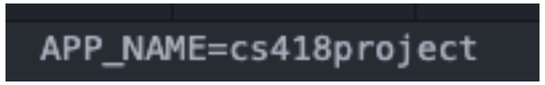
Experiment 4 (continued)

- resources->views->pages->index.blade.php



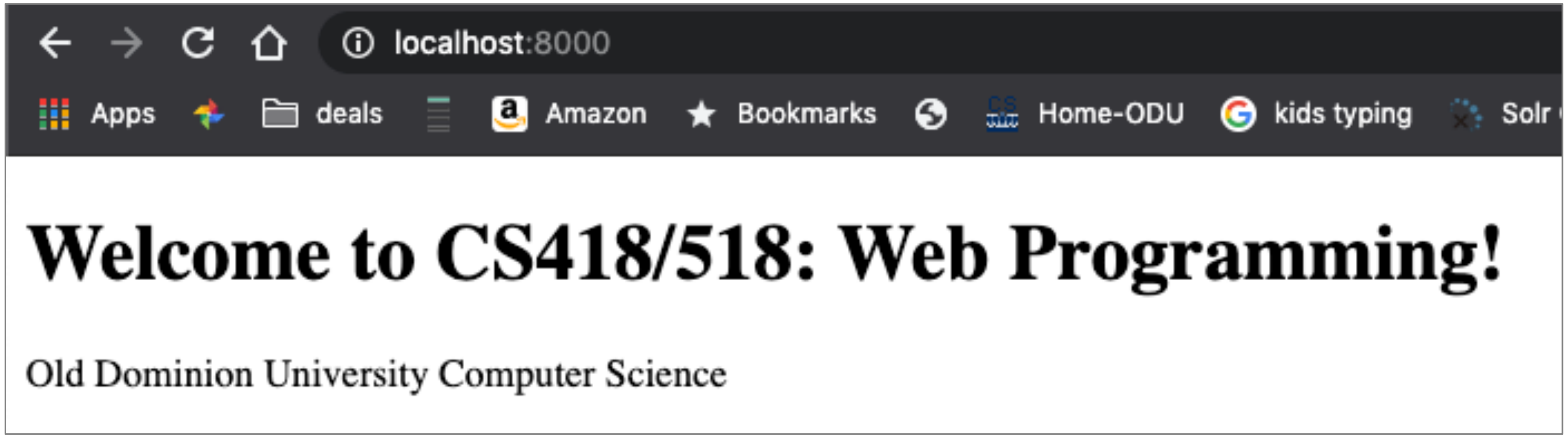
```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title>{{config('app.name','cs418project')}}</title>

    </head>
    <body>
      <h1>Welcome to CS418/518: Web Programming!</h1>
      <p>Old Dominion University Computer Science</p>
    </body>
</html>

```

- edit .env file



```
APP_NAME=cs418project
```

Experiment 4:

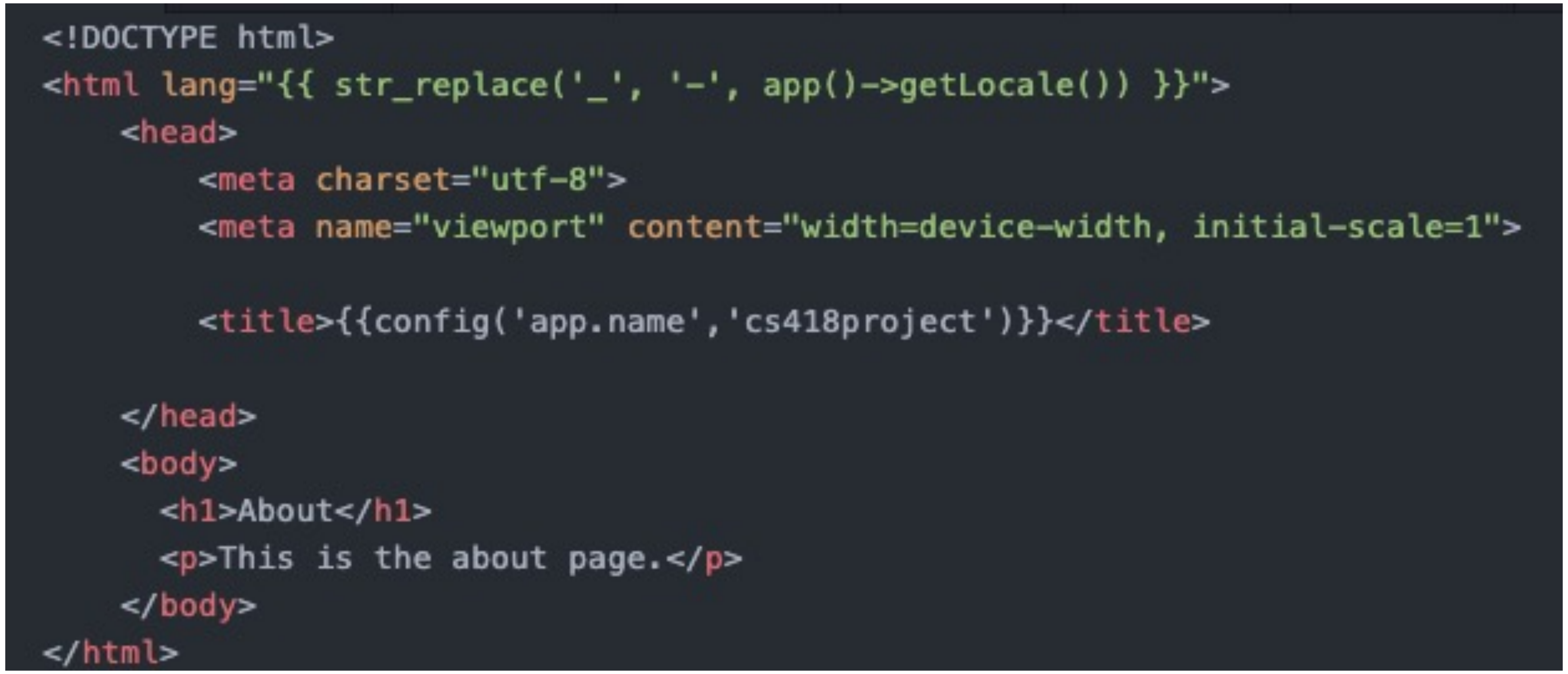


Welcome to CS418/518: Web Programming!

Old Dominion University Computer Science

Experiment 5: multiple views

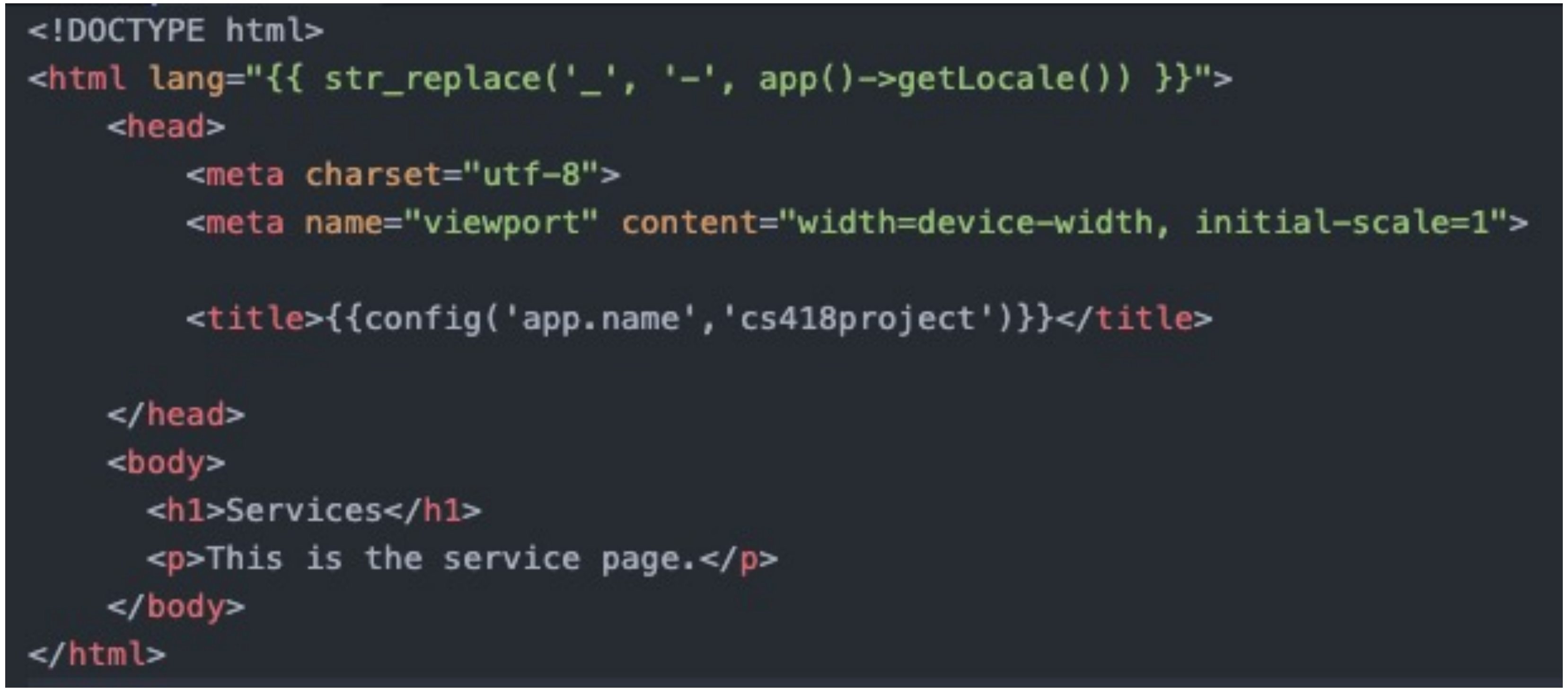
- resources->views->pages->about.blade.php
  - copy content from index.blade.php



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

        <title>{{config('app.name','cs418project')}}</title>

    </head>
    <body>
      <h1>About</h1>
      <p>This is the about page.</p>
    </body>
</html>
```

Experiment 5:

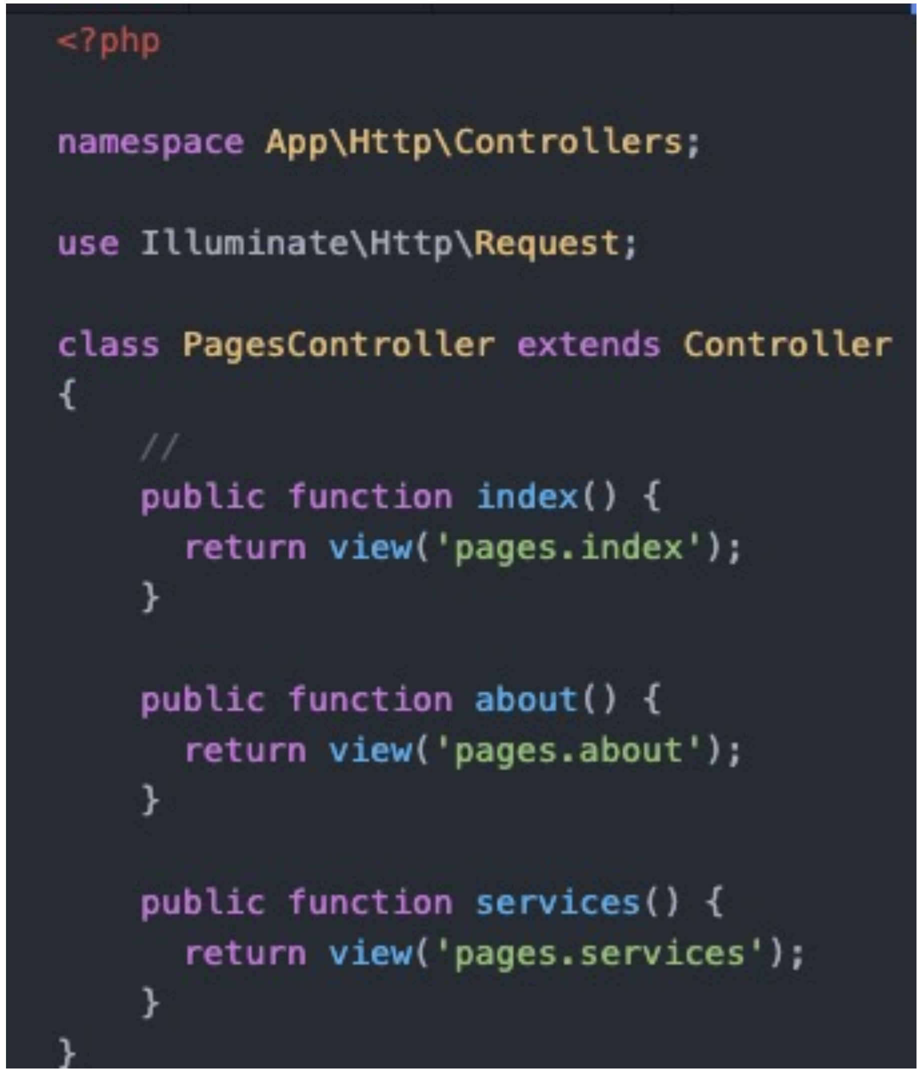
- resources->views->pages->services.blade.php



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

        <title>{{config('app.name','cs418project')}}</title>

    </head>
    <body>
      <h1>Services</h1>
      <p>This is the service page.</p>
    </body>
</html>
```

Experiment 5:

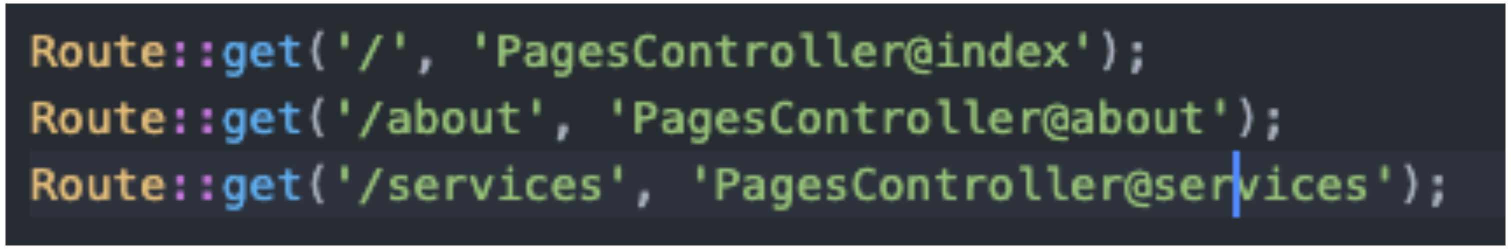
- PagesController.php



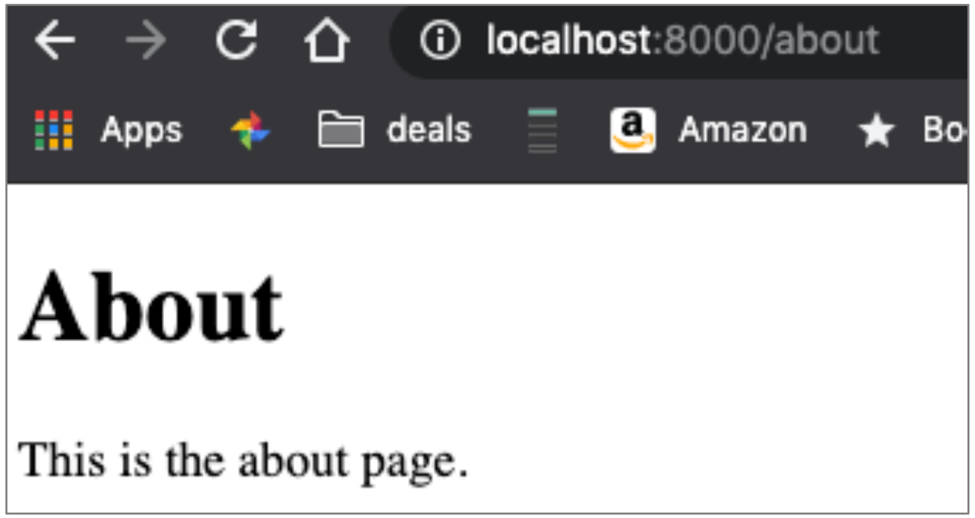
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PagesController extends Controller
{
    //
    public function index() {
      return view('pages.index');
    }

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

    public function services() {
      return view('pages.services');
    }
}
```

Experiment 5:

- routes->web.php

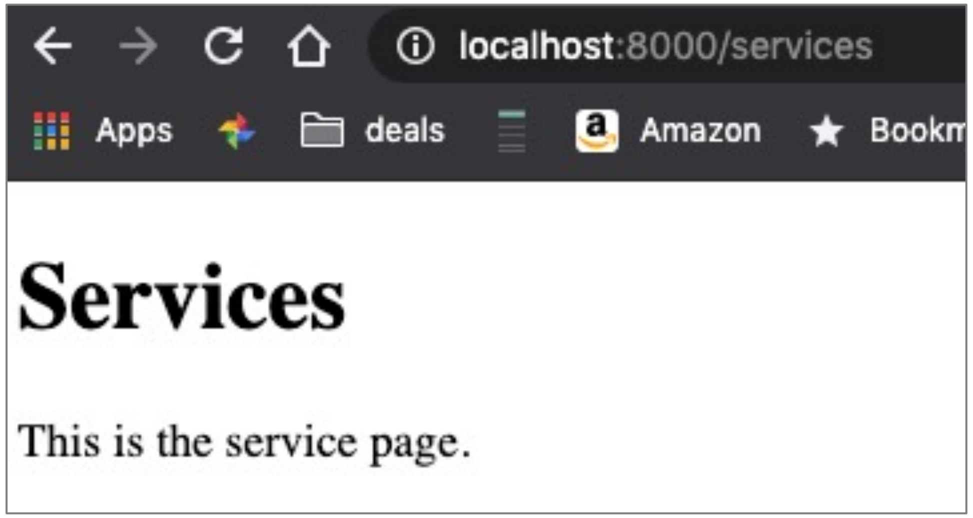


```
Route::get('/', 'PagesController@index');
Route::get('/about', 'PagesController@about');
Route::get('/services', 'PagesController@services');
```



About

This is the about page.



Services

This is the service page.

Backup slides beyond this point

CS418/518