What is Laravel ?

- A powerful and scalable web application framework with PHP.

- An API based framework.

Laravel Philosophy

- To make attractive, expressive and elegant syntax and code.

- Take your pain and give you lots of essential things (e.g: authentication, routing, sessions, caching).

- Providing you very powerful tools needed for large, robust applications.

- Offer you Expressive migration system, tightly integrated unit testing.

- To make best framework of any web framework

Installation with composer

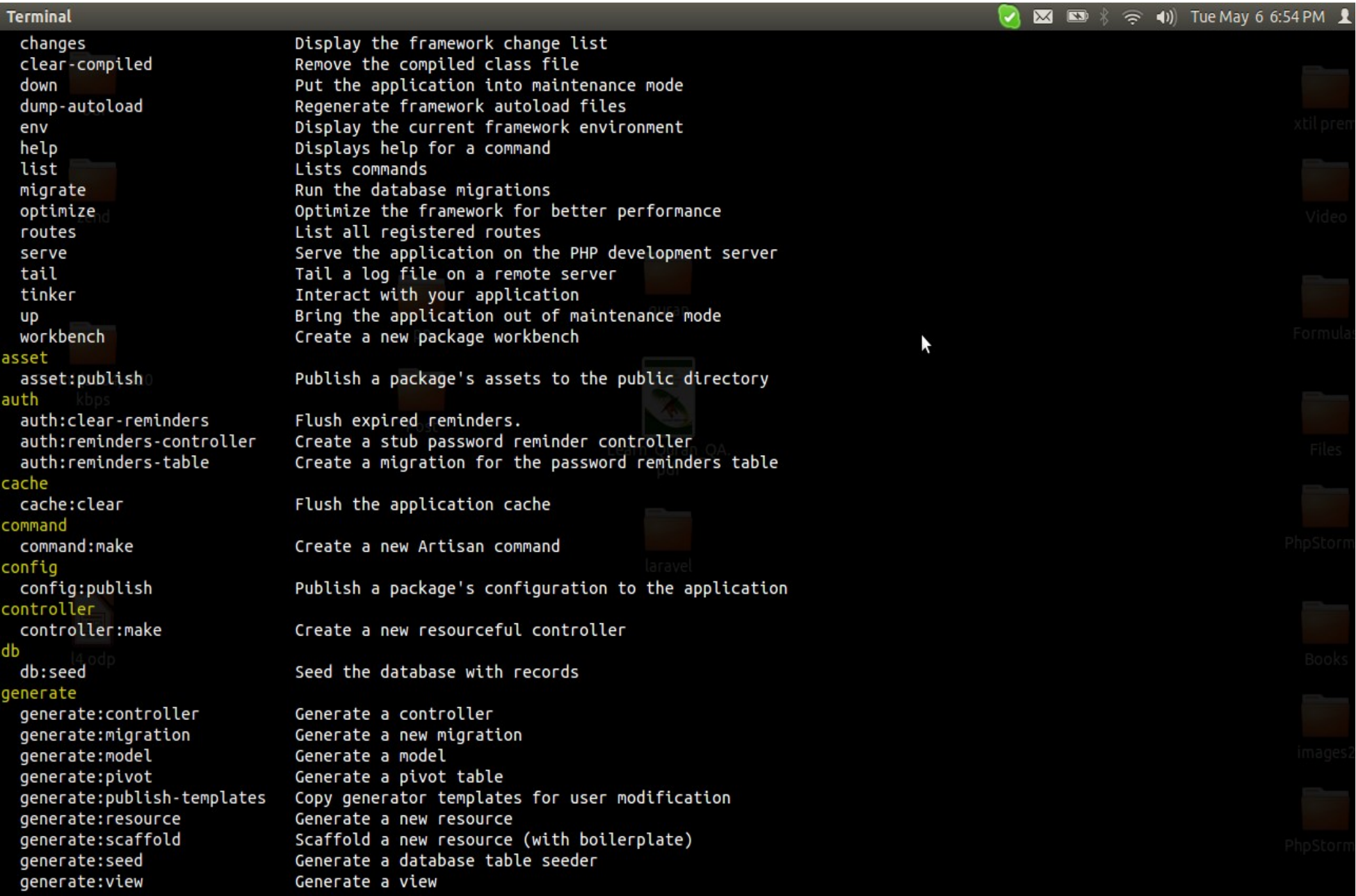
- Laravel installation is based on composer.
- To avoid dependency, updated packages etc.
- You can easily use bower also for js, css.
- To integrated 3rd party tools easily.
- https://packagist.org (like ruby gems)

- composer create-project laravel/laravel your-project-name –prefer-dist
- bower update

Artisan CLI

- Command Line Interface for Laravel
- You will find lots of helpful command here.
- Create controller, view, model with this.
- Database migration, Database seeding etc

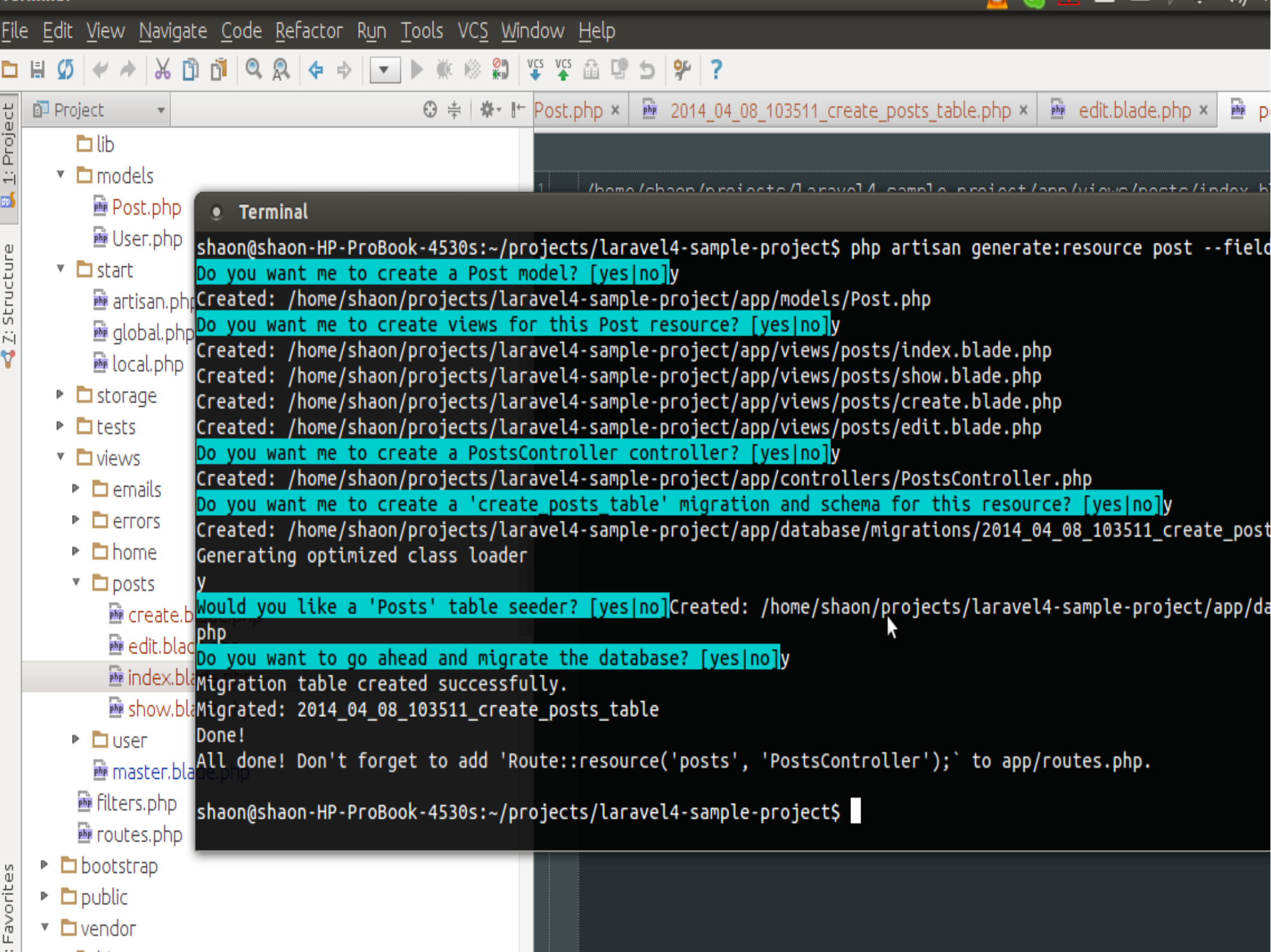
Artisan CLI

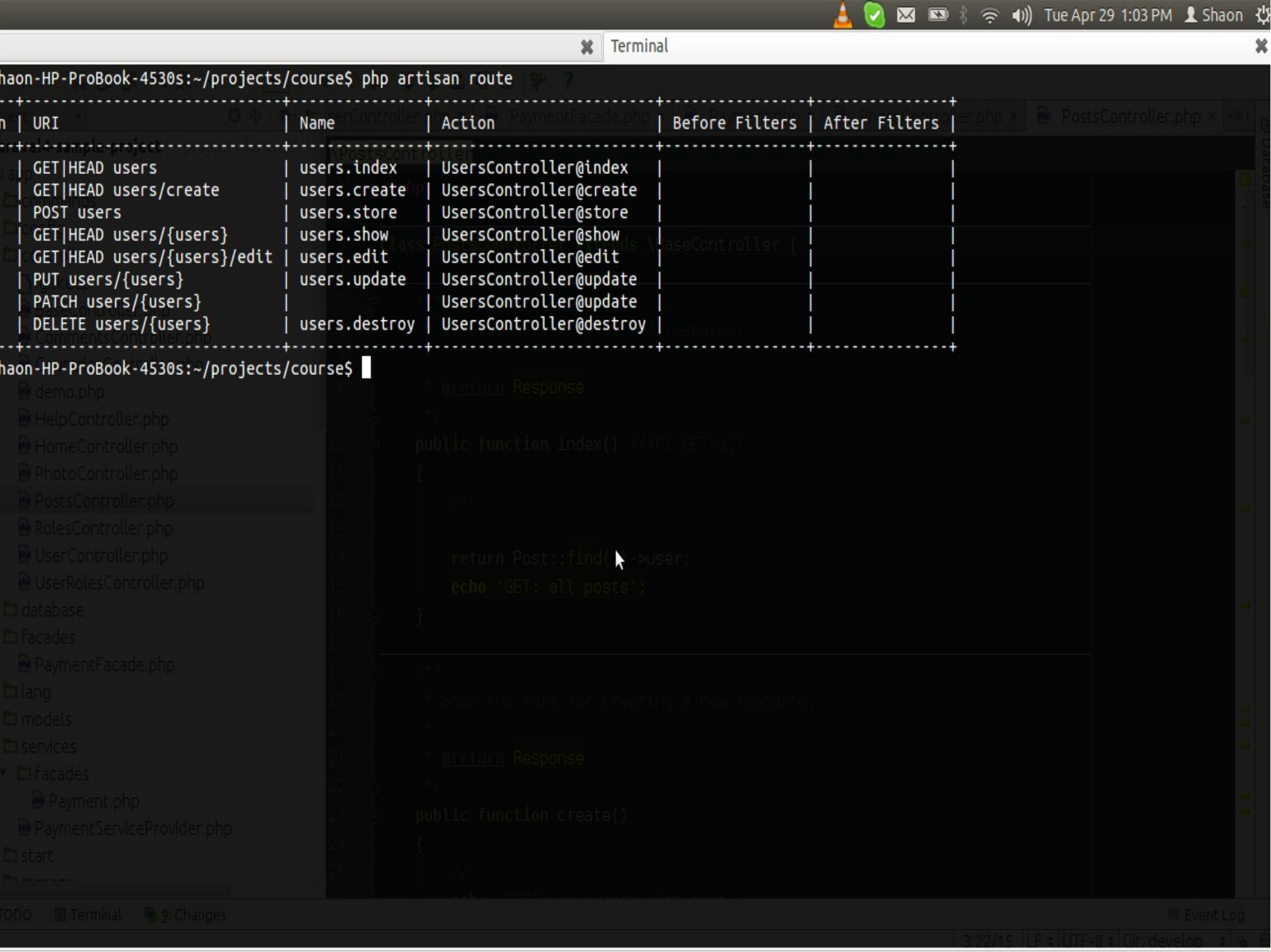

Terminal
Tue May 6 6:54 PM
changes Display the framework change list
clear-compiled Remove the compiled class file
down Put the application into maintenance mode
dump-autoload Regenerate framework autoload files
env Display the current framework environment
help Displays help for a command
list Lists commands
migrate Run the database migrations
optimize Optimize the framework for better performance
routes List all registered routes
serve Serve the application on the PHP development server
tail Tail a log file on a remote server
tinker Interact with your application
up Bring the application out of maintenance mode
workbench Create a new package workbench
asset
asset:publish Publish a package's assets to the public directory
auth
auth:clear-reminders Flush expired reminders.
auth:reminders-controller Create a stub password reminder controller
auth:reminders-table Create a migration for the password reminders table
cache
cache:clear Flush the application cache
command
command:make Create a new Artisan command
config
config:publish Publish a package's configuration to the application
controller
controller:make Create a new resourceful controller
db
db:seed Seed the database with records
generate
generate:controller Generate a controller
generate:migration Generate a new migration
generate:model Generate a model
generate:pivot Generate a pivot table
generate:publish-templates Copy generator templates for user modification
generate:resource Generate a new resource
generate:scaffold Scaffold a new resource (with boilerplate)
generate:seed Generate a database table seeder
generate:view Generate a view

Laravel Resource Controller

- You can create a resource controller
- With resource controller you can easily implement Restful API.
- Resource controller give you all end point methods.
- Laravel support nested resource controller.

e.g: GET lists/1/tasks/1


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project
Post.php
2014_04_08_103511_create_posts_table.php
edit.blade.php
lib
models
Post.php
User.php
start
artisan.php
global.php
local.php
storage
tests
views
emails
errors
home
posts
create.b
edit.blac
index.bl
show.bl
user
master.bla
filters.php
routes.php
bootstrap
public
vendor
Terminal
shaon@shaon-HP-ProBook-4530s:~/projects/laravel4-sample-project$ php artisan generate:resource post --field
Do you want me to create a Post model? [yes|no]y
Created: /home/shaon/projects/laravel4-sample-project/app/models/Post.php
Do you want me to create views for this Post resource? [yes|no]y
Created: /home/shaon/projects/laravel4-sample-project/app/views/posts/index.blade.php
Created: /home/shaon/projects/laravel4-sample-project/app/views/posts/show.blade.php
Created: /home/shaon/projects/laravel4-sample-project/app/views/posts/create.blade.php
Created: /home/shaon/projects/laravel4-sample-project/app/views/posts/edit.blade.php
Do you want me to create a PostsController controller? [yes|no]y
Created: /home/shaon/projects/laravel4-sample-project/app/controllers/PostsController.php
Do you want me to create a 'create_posts_table' migration and schema for this resource? [yes|no]y
Created: /home/shaon/projects/laravel4-sample-project/app/database/migrations/2014_04_08_103511_create_post
Generating optimized class loader
y
Would you like a 'Posts' table seeder? [yes|no]Created: /home/shaon/projects/laravel4-sample-project/app/da
php
Do you want to go ahead and migrate the database? [yes|no]y
Migration table created successfully.
Migrated: 2014_04_08_103511_create_posts_table
Done!
All done! Don't forget to add 'Route::resource('posts', 'PostsController');` to app/routes.php.
shaon@shaon-HP-ProBook-4530s:~/projects/laravel4-sample-project$
1: Project
7: Structure
Favorites


Tue Apr 29 1:03 PM
Shaon
Terminal
haon-HP-ProBook-4530s:~/projects/course$ php artisan route
n | URI | Name | Action | Before Filters | After Filters |
GET	HEAD users	users.index	UsersController@index		
GET	HEAD users/create	users.create	UsersController@create		
POST users	users.store	UsersController@store		
GET	HEAD users/{users}	users.show	UsersController@show		
GET	HEAD users/{users}/edit	users.edit	UsersController@edit		
PUT users/{users}	users.update	UsersController@update		
PATCH users/{users}		UsersController@update		
DELETE users/{users}	users.destroy	UsersController@destroy		
haon-HP-ProBook-4530s:~/projects/course$

Laravel Route

- Get Route
- Post Route
- Route Parameters
- Route Filter Binding
- Route Model Binding
- more...

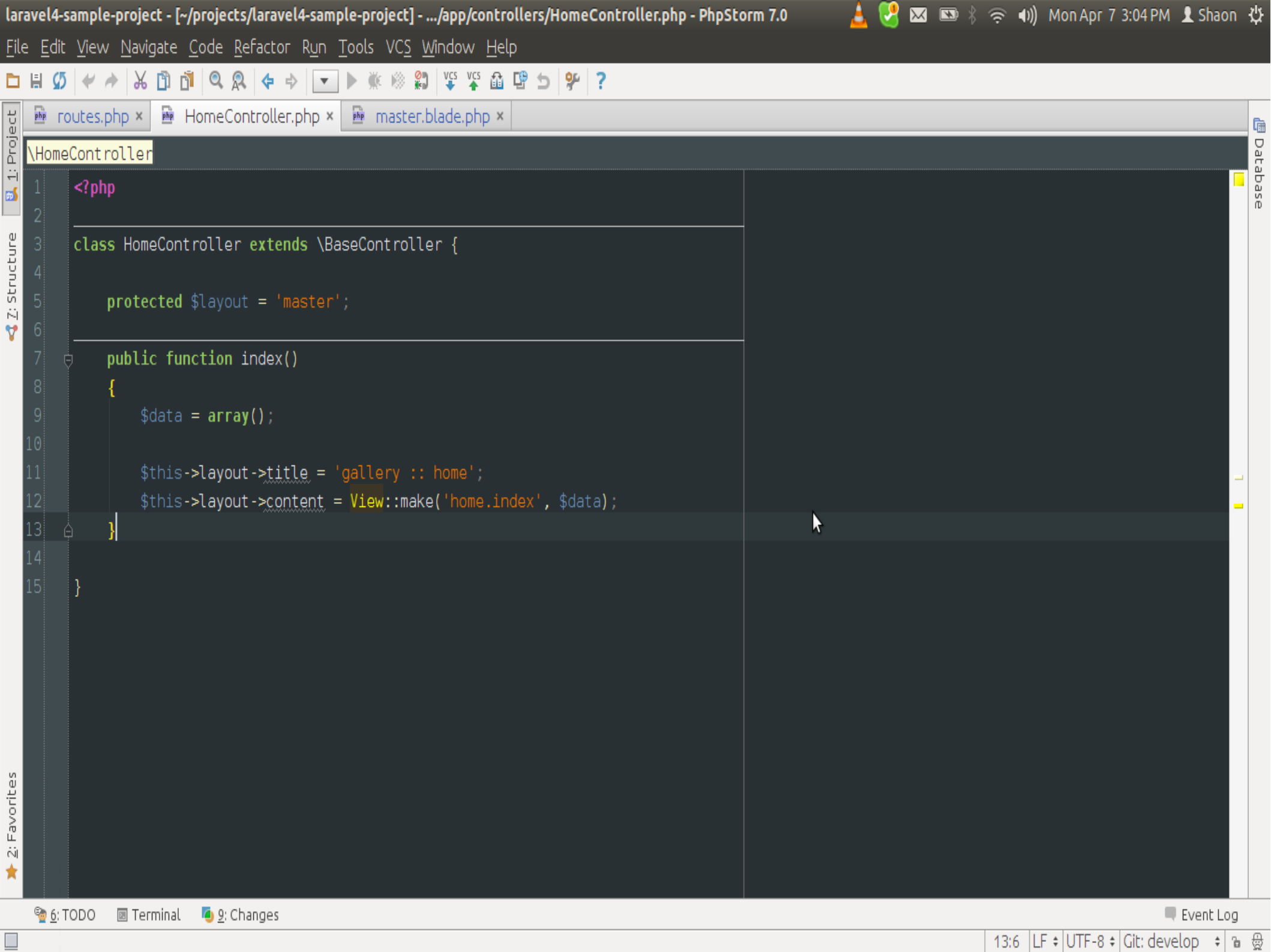
object(stdClass)[475] | object(stdClass)[475] | object(stdClass)[474] | <? /* | untitled | object(stdClass)[569] | object(stdClass)[475]

```php
<?php

GET ROUTE
Route::get('login',    array('uses' => 'UserController@login'));

POST ROUTE
Route::post('login',   array('uses' => 'UserController@postLogin'));

PARAMS BASED ROUTE
Route::get('user/{name?}', function($name = null)
{
    return $name;
});

FILTER BASED
Route::filter('old', function()
{
    if (Input::get('age') < 200)
    {
        return Redirect::to('home');
    }
});

GROUP BASED
Route::group(array('before' => 'auth'), function()
{
    Route::get('/', function()
        {
            // Has Auth Filter
        });

    Route::get('user/profile', function()
        {
            // Has Auth Filter
        });
});
```

Layouts

- Need a layout file
- Call layout from controller
- and your view loaded into that layout

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    {{HTML::style('bower_components/bootstrap/dist/css/bootstrap.css');}}
</head>

<body>

    @if (!Auth::check())
        <ul class="nav navbar-nav">
            <li>
                {{ HTML::link('register', 'Register') }}
            </li>
            <li>
                {{ HTML::link('login', 'Log In') }}
            </li>
        </ul>
    @else
        <div class="welcome-user">Hi, {{Auth::user()->name}} <a title="Logout" href="/logout"><i class="fa fa-sign-out icon-v
    @endif

    <div class="container theme-showcase l4body">
        <!-- Content -->
        @yield('content')
        <!-- ./ content -->
    </div>

    {{HTML::script('bower_components/jquery/jquery.js');}}

</body>
</html>
```


laravel4-sample-project - [~/projects/laravel4-sample-project] - .../app/controllers/HomeController.php - PhpStorm 7.0
Mon Apr 7 3:04 PM
Shaon
File Edit View Navigate Code Refactor Run Tools VCS Window Help
routes.php
HomeController.php
master.blade.php
\HomeController
<?php
class HomeController extends \BaseController {
protected $layout = 'master';
public function index()
{
$data = array();
$this->layout->title = 'gallery :: home';
$this->layout->content = View::make('home.index', $data);
}
}
1: Project
7: Structure
2: Favorites
Database
6: TODO
Terminal
9: Changes
Event Log
13:6
LF
UTF-8
Git: develop

```
1  @section('content')
2      <div class="row">
3          <h1>Welcome from l4</h1>
4      </div>
5  @stop
```

Forms and Input

- Opening A Form
- CSRF Protection
- Form Model Binding
- Labels
- Text, Text Area, Password & Hidden Fields
- Checkboxes and Radio Buttons
- File Input
- Drop-Down Lists
- Buttons
- Custom Macros
- Generating URLs

```
1  @section('content')
2
3      {{ Form::open(array('url' => '/register', 'class' => 'form-horizontal')) }}
4
5
6          {{ Form::label('name', 'Name', array('class' => 'col-lg-2 col-sm-2 col-xs-2 control-label')) }}
7
8          <div class="col-lg-10 col-sm-10 col-xs-10">
9              {{ Form::text('fullname', '', array('class'=>'form-control', 'id'=>'fullname', 'placeholder' => 'Full Name')) }}
10
11             @if ($errors->has('fullname'))
12                 @foreach ($errors->get('fullname', '<p class="l4-form-error">:message</p>') as $name_error)
13                     {{ $name_error }}
14                 @endforeach
15             @endif
16         </div>
17
18         <div class="col-sm-offset-2 col-xs-offset-2 col-sm-10">
19             {{ Form::button('Register Me!', array( 'type' => 'submit', 'class'=>"btn btn-default")) }}
20         </div>
21
22     {{ Form::close() }}
23
24 @stop
```

```
38
39
40
41      public function postRegister()
42      {
43          $data = array();
44          $data['input'] = $input = Input::all();
45          $rules = array(
46              'name' => 'required|regex:/[a-zA-Z0-9]*/',
47              'email' => 'required|email|unique:users',
48              'password' => 'confirmed|required|min:5|max:20',
49              'birth_date' => 'required',
50              'terms'    => 'accepted'
51          );
52          $validation = Validator::make($input, $rules);
53          if ($validation->passes()) {
54              User::saveUser($input);
55              return Redirect::to('success');
56          }
57          return Redirect::back()->withInput()->withErrors($validation->errors());
58      }
59
60
```

Views and Responses

- Basic Responses
- Redirects
- Views
- View Composers
- Special Responses
- Response Macros

Examples - Response

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

```php
$response = Response::make($contents, $statusCode);

$response->header('Content-Type', $value);

return $response;
```

```php
return Response::download($pathToFile);

return Response::download($pathToFile, $name, $headers);
```

Examples - Redirects

```
return Redirect::to('user/login');
```

Returning A Redirect With Flash Data

```
return Redirect::to('user/login')->with('message', 'Login Failed');
```

```
if (Auth::attempt(array('email' => $email, 'password' => $password)))
{
    return Redirect::intended('dashboard');
}
```

View Composers

```
View::composer('profile', function($view)
{
    $view->with('count', User::count());
});
```

Now each time the `profile` view is rendered, the `count` data will be bound to the view.

You may also attach a view composer to multiple views at once:

```
View::composer(array('profile','dashboard'), function($view)
{
    $view->with('count', User::count());
});
```

Database Features

- Basic Usage
- Query Builder
- loquent ORM
- Schema Builder
- Migrations & Seeding
- Redis

Basic Uses

- Configuration
- Read / Write Connections
- Running Queries
- Database Transactions
- Accessing Connections
- Query Logging

```
'mysql' => array(
    'read' => array(
        'host' => '192.168.1.1',
    ),
    'write' => array(
        'host' => '196.168.1.2'
    ),
    'driver'    => 'mysql',
    'database'  => 'database',
    'username'  => 'root',
    'password'  => '',
    'charset'   => 'utf8',
    'collation' => 'utf8_unicode_ci',
    'prefix'    => '',
),
```

Basic Uses

```
$results = DB::select('select * from users where id = ?', array(1));
```

The `select` method will always return an `array` of results.

Running An Insert Statement

```
DB::insert('insert into users (id, name) values (?, ?)', array(1, 'Dayle'));
```

Running An Update Statement

```
DB::update('update users set votes = 100 where name = ?', array('John'));
```

Running A Delete Statement

```
DB::delete('delete from users');
```

Query Builder

- Selects
- Joins
- Advanced Wheres
- Aggregates
- Raw Expressions
- Inserts
- Updates
- Deletes
- Unions
- Pessimistic Locking
- Caching Queries

```php
$users = DB::table('users')->select('name', 'email')->get();

$users = DB::table('users')->distinct()->get();

$users = DB::table('users')->select('name as user_name')->get();
```

Adding A Select Clause To An Existing Query

```php
$query = DB::table('users')->select('name');

$users = $query->addSelect('age')->get();
```

Using Where Operators

```php
$users = DB::table('users')->where('votes', '>', 100)->get();
```

Or Statements

```php
$users = DB::table('users')
                    ->where('votes', '>', 100)
                    ->orWhere('name', 'John')
                    ->get();
```

```php
DB::table('users')->increment('votes');

DB::table('users')->increment('votes', 5);

DB::table('users')->decrement('votes');

DB::table('users')->decrement('votes', 5);
```

Best Features of Laravel

- Eloquent ORM
- Cache
- Ioc Container
- Facades
- Database Migration and seeding
- Unit Testing

Eloquent ORM

- The Eloquent ORM give you facilities your database works.
- Eloquent ORM recommended every table have a corresponding Model which is used to interact with that table.

Features of Eloquent ORM

- Basic Usage
- Mass Assignment
- Insert, Update, Delete
- Soft Deleting
- Timestamps
- Query Scopes
- Relationships
- Querying Relations
- Eager Loading
- Inserting Related Models
- Touching Parent Timestamps
- Working With Pivot Tables
- Collections
- Accessors & Mutators
- Date Mutators
- Model Events
- Model Observers
- Converting To Arrays / JSON

Retrieving All Models

```
$users = User::all();
```

Retrieving A Record By Primary Key

```
$user = User::find(1);

var_dump($user->name);
```

```
$model = User::findOrFail(1);

$model = User::where('votes', '>', 100)->firstOrFail();
```

```
// Create a new user in the database...
$user = User::create(array('name' => 'John'));

// Retrieve the user by the attributes, or create it if it doesn't exist...
$user = User::firstOrCreate(array('name' => 'John'));

// Retrieve the user by the attributes, or instantiate a new instance...
$user = User::firstOrNew(array('name' => 'John'));
```

```php
class Post extends Eloquent {

    public function comments()
    {
        return $this->hasMany('Comment');
    }

}
```

Now we can access the post's comments through the dynamic property:

```php
$comments = Post::find(1)->comments;
```

```php
class Comment extends Eloquent {

    public function post()
    {
        return $this->belongsTo('Post');
    }

}
```

```php
$posts = Post::has('comments')->get();
```

Eager Loading

```php
foreach (Book::all() as $book)
{
    echo $book->author->name;
}
```

```php
foreach (Book::with('author')->get() as $book)
{
    echo $book->author->name;
}
```

```php
$books = Book::all();

$books->load('author', 'publisher');
```

Touching Parent Timestamps

```php
class Comment extends Eloquent {

    protected $touches = array('post');

    public function post()
    {
        return $this->belongsTo('Post');
    }

}
```

Model Observers

```php
class UserObserver {

    public function saving($model)
    {
        //
    }

    public function saved($model)
    {
        //
    }

}
```

Accessors & Mutators

```php
class User extends Eloquent {

    public function getFirstNameAttribute($value)
    {
        return ucfirst($value);
    }

}
```

Laravel Cache

- Laravel supports popular caching backends like File System, Memcached and Redis.

- You can configure your cache easily.

Storing An Item In The Cache Permanently

```
Cache::forever('key', 'value');
```

Removing An Item From The Cache

```
Cache::forget('key');
```

Accessing A Tagged Cache

You may store a tagged cache by passing in an ordered list of tag names as arguments, or as an ordered array of tag names.

```
Cache::tags('people', 'authors')->put('John', $john, $minutes);

Cache::tags(array('people', 'artists'))->put('Anne', $anne, $minutes);
```

Caching Queries

You may easily cache the results of a query using the `remember` method:

Caching A Query Result

```
$users = DB::table('users')->remember(10)->get();
```

In this example, the results of the query will be cached for ten minutes. While the results are cached, the query will not be run against the database, and the results will be loaded from the default cache driver specified for your application.

If you are using a supported cache driver, you can also add tags to the caches:

```
$users = DB::table('users')->cacheTags(array('people', 'authors'))->remember(10)->get();
```

Storing An Item In The Cache

```
Cache::put('key', 'value', $minutes);
```

Using Carbon Objects To Set Expire Time

```
$expiresAt = Carbon::now()->addMinutes(10);

Cache::put('key', 'value', $expiresAt);
```

Storing An Item In The Cache If It Doesn't Exist

```
Cache::add('key', 'value', $minutes);
```

The `add` method will return `true` if the item is actually **added** to the cache. Otherwise, the method will return `false`.

Checking For Existence In Cache

```
if (Cache::has('key'))
{
    //
}
```

Ioc Container

- Inversion Of Control Container
- A powerful tool for managing class dependencies.
- Removing hard-coded class dependencies.
- Facades are based on Ioc Container

Facades

- Facades provide a "static" interface to classes that are available IoC container.

- Laravel ships with many facades, and you have probably been using them without even knowing it!

- Laravel "facades" serve as "static proxies" to underlying classes in the IoC container

How to create Facades

- An IoC binding
- A facade class.
- A facade alias configuration.

Facades Example

```php
$value = Cache::get('key');
```

```php
class Cache extends Facade {

    /**
     * Get the registered name of the component.
     *
     * @return string
     */
    protected static function getFacadeAccessor() { return 'cache'; }

}
```

Laravel Events

The Laravel Event class provides a simple observer implementation, allowing you to subscribe and listen for events in your application.

- Wildcard Listeners
- Using Classes As Listeners
- Queued Events
- Event Subscribers

Event Examples

```php
Event::listen('auth.login', function($user)
{
    $user->last_login = new DateTime;

    $user->save();
});
```

```php
$event = Event::fire('auth.login', array($user));
```

```php
Event::listen('auth.login', 'LoginHandler', 10);

Event::listen('auth.login', 'OtherHandler', 5);
```

Errors and logging

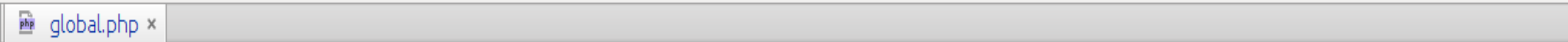
- Easy Configuration
- Handling Errors
- HTTP Exceptions
- Handling 404 Errors
- Logging

Example

```
$logFile = 'laravel.log';

Log::useDailyFiles(storage_path().'/logs/'.$logFile);
```

```
App::error(function(Exception $exception)
{
    Log::error($exception);
});
```

```
App::missing(function($exception)
{
    return Response::view('errors.missing', array(), 404);
});
```




```
46  | even register several error handlers to handle different types of
47  | exceptions. If nothing is returned, the default error view is
48  | shown, which includes a detailed stack trace during debug.
49  |
50  */
51
52  App::error(function($exception, $code)
53  {
54      switch ($code)
55      {
56          case 403:
57              return Response::view('errors.403', array(), 403);
58
59          case 404:
60              return Response::view('errors.404', array(), 404);
61
62          case 500:
63              return Response::view('errors.500', array(), 500);
64
65          default:
66              return Response::view('errors.default', array(), $code);
67      }
68  });
69  /*
70  |--------------------------------------------------------------------------
```

Queue

- The Laravel Queue component provides a unified API
- Using this API you can do different queue services.
- Queues allow you to defer the processing of a time consuming task
- Such as sending an e-mail, until a later time
- Thus drastically speeding up the web requests to your application.

Queue Example

```php
Queue::push('SendEmail', array('message' => $message));
```

```php
class SendEmail {

    public function fire($job, $data)
    {
        //
    }

}
```

Security

- Storing Passwords
- Authenticating Users
- Manually Logging In Users
- Protecting Routes
- HTTP Basic Authentication
- Password Reminders & Reset
- Encryption
- Authentication Drivers
- Laravel provides facilities for strong AES-256 encryption via the mcrypt PHP extension.

Some Example

```php
$password = Hash::make('secret');
```

```php
if (Auth::attempt(array('email' => $email, 'password' => $password)))
{
    return Redirect::intended('dashboard');
}
```

```php
$user = User::find(1);

Auth::login($user);
```

```php
Auth::loginUsingId(1);
```

```php
if (Auth::viaRemember())
{
    //
}
```

```php
if (Auth::once($credentials))
{
    //
}
```

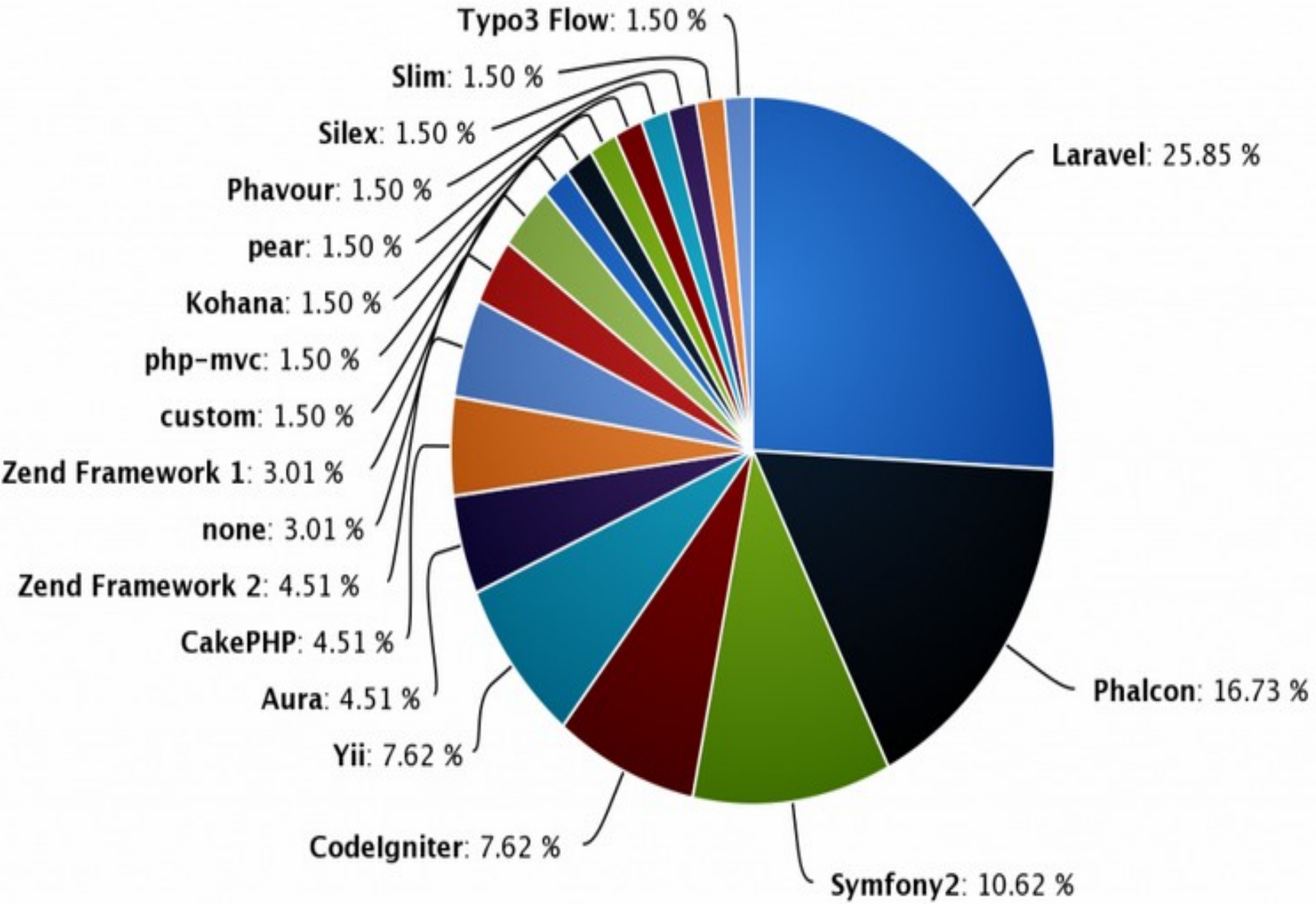
Security Check

```php
Route::get('profile', array('before' => 'auth.basic', function()
{
    // Only authenticated users may enter...
}));
```

```php
$encrypted = Crypt::encrypt('secret');
```

```php
$decrypted = Crypt::decrypt($encryptedValue);
```

```php
Crypt::setMode('ctr');

Crypt::setCipher($cipher);
```


Typo3 Flow: 1.50 %
Slim: 1.50 %
Silex: 1.50 %
Phavour: 1.50 %
pear: 1.50 %
Kohana: 1.50 %
php-mvc: 1.50 %
custom: 1.50 %
Zend Framework 1: 3.01 %
none: 3.01 %
Zend Framework 2: 4.51 %
CakePHP: 4.51 %
Aura: 4.51 %
Yii: 7.62 %
CodeIgniter: 7.62 %
Symfony2: 10.62 %
Phalcon: 16.73 %
Laravel: 25.85 %