Laravel - Namespaces

Namespaces can be defined as a class of elements in which each element has a unique name to that associated class. It may be shared with elements in other classes.

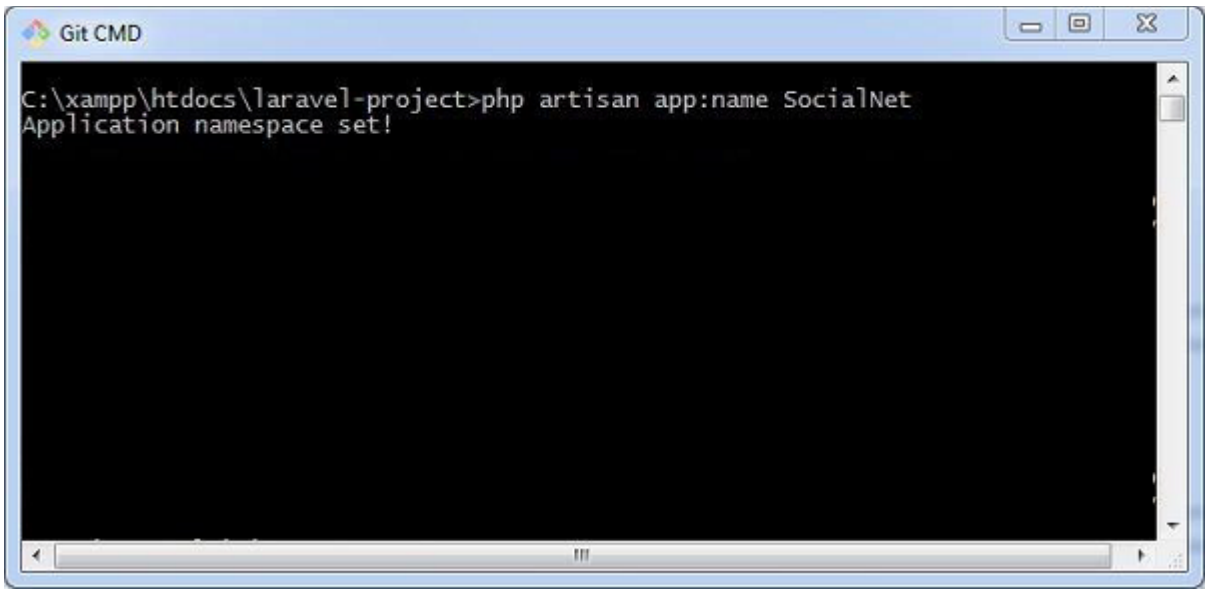
Declaration of namespace

The **use** keyword allows the developers to shorten the namespace.

use <namespace-name>;

The default namespace used in Laravel is App, however a user can change the namespace to match with web application. Creating user defined namespace with artisan command is mentioned as follows –

php artisan app:name SocialNet



The namespace once created can include various functionalities which can be used in controllers and various classes.

Laravel - Controllers

In the MVC framework, the letter C stands for Controller. It acts as a directing traffic between Views and Models. In this chapter, you will learn about Controllers in Laravel.

Creating a Controller

Open the command prompt or terminal based on the operating system you are using and type the following command to create controller using the Artisan CLI (Command Line Interface).

php artisan make:controller <controller-name> --plain

Replace the <controller-name> with the name of your controller. This will create a plain constructor as we are passing the argument **plain**. If you dont want to create a plain constructor, you can simply ignore the argument. The created constructor can be seen at **app/Http/Controllers**.

You will see that some basic coding has already been done for you and you can add your custom coding. The created controller can be called from routes.php by the following syntax.

Syntax

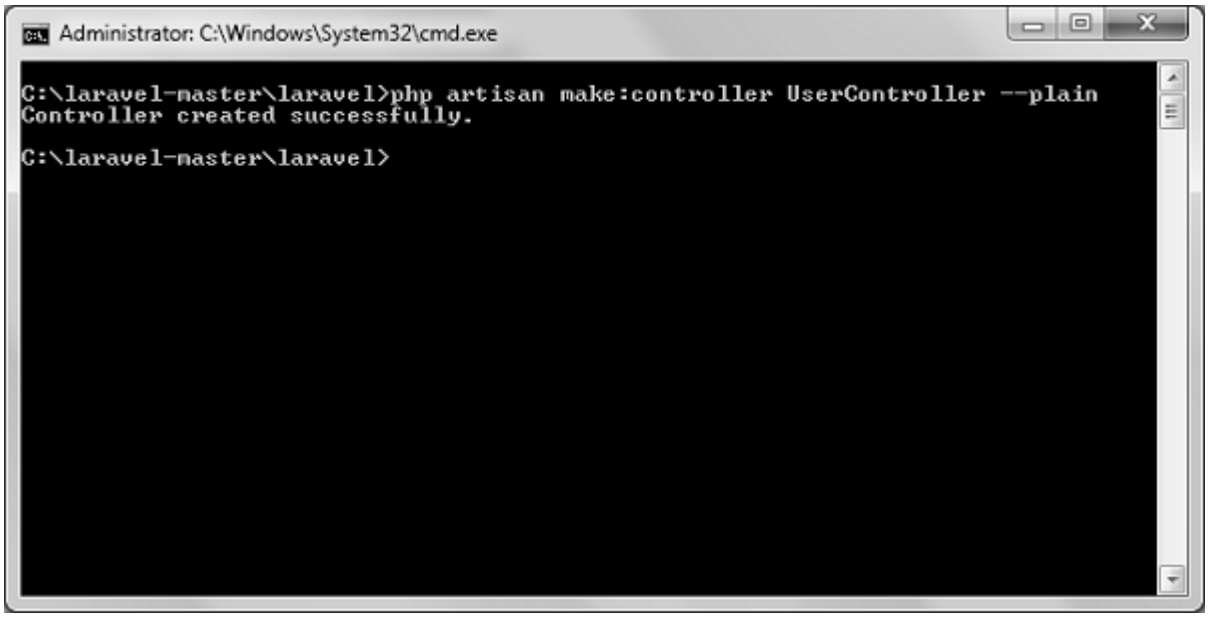
Route::get(base URI,controller@method);

Example

Step 1 – Execute the following command to create **UserController**.

php artisan make:controller UserController --plain

Step 2 – After successful execution, you will receive the following output.



Step 3 – You can see the created controller at **app/Http/Controller/UserController.php** with some basic coding already written for you and you can add your own coding based on your need.

```
<?php


namespace App\Http\Controllers;


use Illuminate\Http\Request;

use App\Http\Requests;

use App\Http\Controllers\Controller;


class UserController extends Controller {
```

```
  //
}
```

Advertisement

Controller Middleware

We have seen middleware before and it can be used with controller also. Middleware can also be assigned to controllers route or within your controllers constructor. You can use the middleware method to assign middleware to the controller. The registered middleware can also be restricted to certain method of the controller.

Assigning Middleware to Route

```
Route::get('profile', [
  'middleware' => 'auth',
  'uses' => 'UserController@showProfile'
]);
```

Here we are assigning auth middleware to UserController in profile route.

Assigning Middleware within Controllers constructor

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
```

```
use App\Http\Requests;

use App\Http\Controllers\Controller;


class UserController extends Controller {

  public function __construct() {

    $this->middleware('auth');

  }

}
```

Here we are assigning **auth** middleware using the middleware method in the **UserController** constructor.

Example

Step 1 – Add the following lines of code to the **app/Http/routes.php** file and save it.

routes.php

```
<?php

Route::get('/usercontroller/path',[

  'middleware' => 'First',

  'uses' => 'UserController@showPath'

]);
```

Step 2 – Create a middleware called **FirstMiddleware** by executing the following line of code.

php artisan make:middleware FirstMiddleware

Step 3 – Add the following code into the **handle** method of the newly created FirstMiddleware at **app/Http/Middleware**.

FirstMiddleware.php

```
<?php


namespace App\Http\Middleware;

use Closure;


class FirstMiddleware {

  public function handle($request, Closure $next) {

    echo '<br>First Middleware';

    return $next($request);

  }

}
```

Step 4 – Create a middleware called **SecondMiddleware** by executing the following command.

php artisan make:middleware SecondMiddleware

Step 5 – Add the following code in the handle method of the newly created SecondMiddleware at **app/Http/Middleware**.

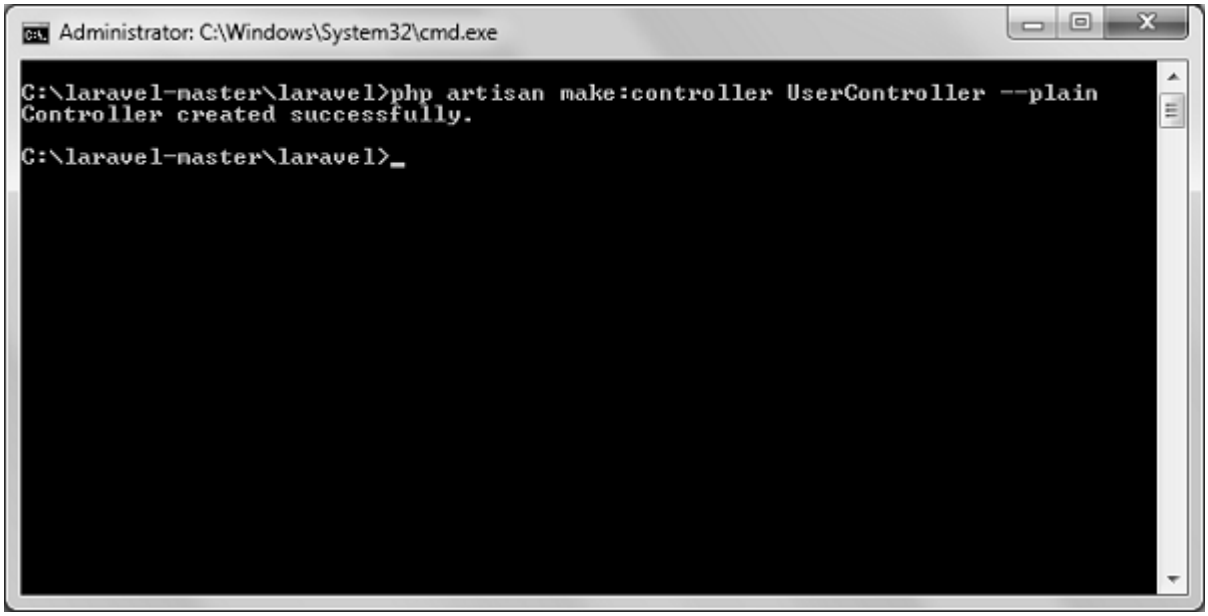
SecondMiddleware.php

```
<?php


namespace App\Http\Middleware;

use Closure;


class SecondMiddleware {

  public function handle($request, Closure $next) {

    echo '<br>Second Middleware';

    return $next($request);

  }

}
```

Step 6 – Create a controller called **UserController** by executing the following line.

```
php artisan make:controller UserController --plain
```

Step 7 – After successful execution of the URL, you will receive the following output –



Step 8 – Copy the following code to **app/Http/UserController.php** file.

app/Http/UserController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;

use App\Http\Controllers\Controller;

class UserController extends Controller {

  public function __construct() {

    $this->middleware('Second');
```

```
  }
  public function showPath(Request $request) {
    $uri = $request->path();
    echo '<br>URI: '.$uri;

    $url = $request->url();
    echo '<br>';

    echo 'URL: '.$url;
    $method = $request->method();
    echo '<br>';

    echo 'Method: '.$method;
  }
}
```

Step 9 – Now launch the phps internal web server by executing the following command, if you havent executed it yet.

```
php artisan serve
```

Step 10 – Visit the following URL.

http://localhost:8000/usercontroller/path

Restful Resource Controllers

Often while making an application we need to perform **CRUD (Create, Read, Update, Delete)** operations. Laravel makes this job easy for us. Just create a controller and Laravel will automatically provide all the methods for the CRUD operations. You can also register a single route for all the methods in routes.php file.

Example

Step 1 – Create a controller called **MyController** by executing the following command.

php artisan make:controller MyController

Step 2 – Add the following code in

app/Http/Controllers/MyController.php file.

app/Http/Controllers/MyController.php

```
<?php


namespace App\Http\Controllers;


use Illuminate\Http\Request;

use App\Http\Requests;

use App\Http\Controllers\Controller;


class MyController extends Controller {
```

```
public function index() {
  echo 'index';
}
public function create() {
  echo 'create';
}
public function store(Request $request) {
  echo 'store';
}
public function show($id) {
  echo 'show';
}
public function edit($id) {
  echo 'edit';
}
public function update(Request $request, $id) {
  echo 'update';
}
public function destroy($id) {
  echo 'destroy';
```

```
  }

}
```

Step 3 – Add the following line of code in **app/Http/routes.php** file.

app/Http/routes.php

```
Route::resource('my','MyController');
```