Laravel - Response

A web application responds to a users request in many ways depending on many parameters. This chapter explains you in detail about responses in Laravel web applications.

Basic Response

Laravel provides several different ways to return response. Response can be sent either from route or from controller. The basic response that can be sent is simple string as shown in the below sample code. This string will be automatically converted to appropriate HTTP response.

Example

Step 1 – Add the following code to **app/Http/routes.php** file.

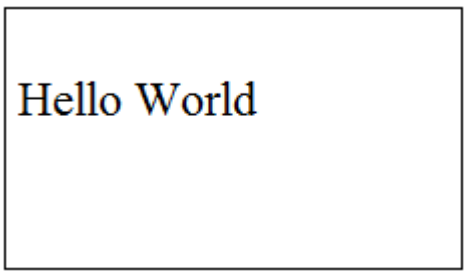
app/Http/routes.php

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

Step 2 – **Visit** the following URL to test the basic response.

http://localhost:8000/basic_response

Step 3 – The output will appear as shown in the following image.



Attaching Headers

The response can be attached to headers using the header() method. We can also attach the series of headers as shown in the below sample code.

```
return response($content,$status)
  ->header('Content-Type', $type)
  ->header('X-Header-One', 'Header Value')
  ->header('X-Header-Two', 'Header Value');
```

Example

Observe the following example to understand more about Response –

Step 1 – Add the following code to **app/Http/routes.php** file.

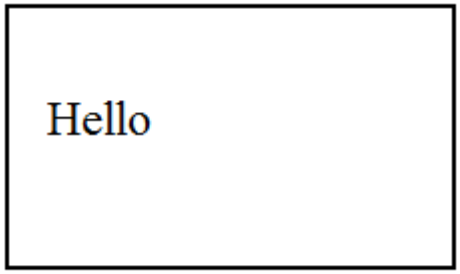
app/Http/routes.php

```
Route::get('/header',function() {
  return response("Hello", 200)->header('Content-Type', 'text/html');
});
```

Step 2 – Visit the following URL to test the basic response.

http://localhost:8000/header

Step 3 – The output will appear as shown in the following image.



Attaching Cookies

The **withcookie()** helper method is used to attach cookies. The cookie generated with this method can be attached by calling **withcookie()** method with response instance. By default, all cookies generated by Laravel are encrypted and signed so that they can't be modified or read by the client.

Example

Observe the following example to understand more about attaching cookies –

Step 1 – Add the following code to **app/Http/routes.php** file.

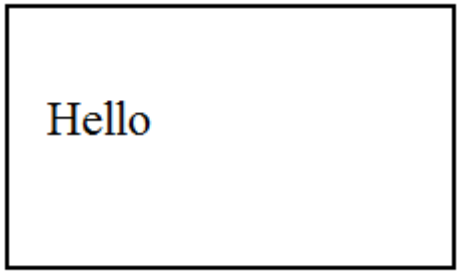
app/Http/routes.php

```
Route::get('/cookie',function() {
  return response("Hello", 200)->header('Content-Type', 'text/html')
    ->withcookie('name','Virat Gandhi');
});
```

Step 2 – **Visit** the following URL to test the basic response.

http://localhost:8000/cookie

Step 3 – The output will appear as shown in the following image.



Laravel - Views

Previous

Quiz

Next

In MVC framework, the letter **V** stands for **Views**. It separates the application logic and the presentation logic. Views are stored in **resources/views** directory. Generally, the view contains the HTML which will be served by the application.

Example

Observe the following example to understand more about Views –

Step 1 – Copy the following code and save it at **resources/views/test.php**

```
<html>
  <body>
    <h1>Hello, World</h1>
```

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

Step 2 – Add the following line in **app/Http/routes.php** file to set the route for the above view.

app/Http/routes.php

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

Step 3 – Visit the following URL to see the output of the view.

http://localhost:8000/test

Step 4 – The output will appear as shown in the following image.

Passing Data to Views

While building application it may be required to pass data to the views. Pass an array to view helper function. After passing an array, we can use the key to get the value of that key in the HTML file.

Example

Observe the following example to understand more about passing data to views –

Step 1 – Copy the following code and save it at **resources/views/test.php**

```
<html>
  <body>
    <h1><?php echo $name; ?></h1>
  </body>
</html>
```

Step 2 – Add the following line in **app/Http/routes.php** file to set the route for the above view.

app/Http/routes.php

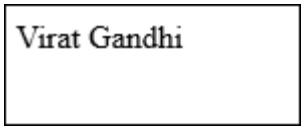
```
Route::get('/test', function() {
  return view('test',[name=>Virat Gandhi]);
});
```

Step 3 – The value of the key name will be passed to test.php file and $name will be replaced by that value.

Step 4 – Visit the following URL to see the output of the view.

http://localhost:8000/test

Step 5 – The output will appear as shown in the following image.



Sharing Data with all Views

We have seen how we can pass data to views but at times, there is a need to pass data to all the views. Laravel makes this simpler. There is a method called **share()** which can be used for this

purpose. The **share()** method will take two arguments, key and value. Typically **share()** method can be called from boot method of service provider. We can use any service provider, **AppServiceProvider** or our own service provider.

Example

Observe the following example to understand more about sharing data with all views –

Step 1 – Add the following line in **app/Http/routes.php** file.

app/Http/routes.php

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

Route::get('/test2', function() {
  return view('test2');
});
```

Step 2 – Create two view files **test.php** and **test2.php** with the same code. These are the two files which will share data. Copy the following code in both the files. **resources/views/test.php & resources/views/test2.php**

```
<html>
  <body>
```

```
    <h1><?php echo $name; ?></h1>
  </body>
</html>
```

Step 3 – Change the code of boot method in the file **app/Providers/AppServiceProvider.php** as shown below. (Here, we have used share method and the data that we have passed will be shared with all the views.) **app/Providers/AppServiceProvider.php**

```
<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider {

  /**
    * Bootstrap any application services.
    *
    * @return void
  */
```

```
  public function boot() {
    view()->share('name', 'Virat Gandhi');
  }

  /**
    * Register any application services.
    *
    * @return void
  */

  public function register() {
    //
  }
}
```

Step 4 – **Visit** the following URLs.

http://localhost:8000/test

http://localhost:8000/test2