Laravel - Response

A web application responds to a user's 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

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

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

JSON Response JSON response can be sent using the json method. This method will automatically set the Content-Type header to application/json. The json method will automatically convert the array into appropriate json response.

Example Observe the following example to understand more about JSON Response −

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

app/Http/routes.php

Route::get('json',function() { return response()->json(['name' => 'Virat Gandhi', 'state' => 'Gujarat']); }); Step 2 − Visit the following URL to test the json response.

http://localhost:8000/json

Laravel provides various tools and features to help build modern, efficient web applications. One of the key features of Laravel is its ability to handle HTTP requests and generate responses. In this tutorial, you will learn how to use the Laravel Response class to generate different types of responses in your application.

In Laravel, different types of responses can be returned from routes and controllers. Some common types of responses include:

The most basic way is to use the response helper function, which returns an instance of the Illuminate\Http\Response class. Here's an example of using the response function to send a simple text response:

return response('Hello World', 200); The first argument in the above code is the response's content, and the second is the HTTP status code. In this case, we're sending a "200 OK" status code, which indicates that the request was successful.

The response function can also be used to send a view template as a response:

Example:

return response()->view('welcome'); The above code will render the welcome.blade.php view from the resources/views directory and send it as a response.

In Laravel, an HTML response can be returned by using the response() helper function and calling the view() function onto it. The view() function takes the name of the view as its first argument and an array of data as its second argument. The data array will be passed to the view as variables, which can then be used in the HTML code.

Route::get('/', function () { return response()->view('welcome', ['name' => 'Alex',]); }); In the above code, the view function will look for a file called welcome.blade.php in the resources/views directory, pass a variable $name with the value 'Alex' to the view, and then render it as an HTML response.

In Laravel, a JSON response can be returned from a route or controller using the json helper function or the response() helper function with the json function. Here's an example of how to return a JSON response from a route using the json helper function:

Route::get('/users', function () { $users = [['userid' => 1, 'name' => 'Alex'], ['userid' => 2, 'name' => 'Jane'],]; return json($users); }); A redirect response is an HTTP response that tells the client to redirect to a different URL. Redirect responses often redirect the user to another page after completing a form submission, login, logout, or other action. Redirect responses can be created using the redirect helper function or the response() helper function with the redirect function.

Route::post('/signin', function () { // Action: user authenticated

// Create a redirect response using the redirect helper function: return redirect('/dashboard');

// or

// Use the response() helper function to create a redirect response: return response()->redirect('/dashboard'); }); The above code will return a header location with a 302 Found HTTP status code and value /dashboard, telling the client to redirect to the /dashboard URL.

A download response is an HTTP response that tells the client to download a file. Download responses often allow the user to download a file from the server. Download responses can be created using the redirect helper function or the response() helper function with the redirect function.

Route::post('/download', function () { return download('file_path/file_name.zip');

// or

return response()->download('file_path/file_name.zip'); }); The above code will set the Content-Disposition header of the response to "attachment; filename="file_name.zip" and the Content-Type header for the MIME type of the file, which tells the client to download the file.

The Laravel File class provides functions for creating HTTP responses for sending files to the client. This tells the browser to treat the response as a file, prompting the user to download or display it in the browser if possible.

return response()->download($pathToFile, 'filename.pdf');

// or

return response()->file($pathToFile);

// or

```
return response()->stream($pathToFile, ’filename.pdf’);
```