RESPONES

Creating Responses

Strings & Arrays

All routes and controllers should return a response to be sent back to the user's browser. Laravel provides several different ways to return responses. The most basic response is returning a string from a route or controller. The framework will automatically convert the string into a full HTTP response:

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

In addition to returning strings from your routes and controllers, you may also return arrays. The framework will automatically convert the array into a JSON response:

```
Route::get('/', function () {
    return [1, 2, 3];
});
```

Response Objects

Typically, you won't just be returning simple strings or arrays from your route actions. Instead, you will be returning full **Illuminate\Http\Response** instances or views.

Returning a full Response instance allows you to customize the response's HTTP status code and headers. A Response instance inherits from the Symfony\Component\HttpFoundation\Response class, which provides a variety of methods for building HTTP responses:

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

file upload with laravel

Example

Step 1 – To start, you need to create a view file that will handle the file upload. You can name this file uploadfile.php and place it in the resources/views directory. This is the code you should copy into the file:

```
<html>
 <body>
      <?php
         echo Form::open(array('url' => '/uploadfile','files'=>'true'));
         echo 'Select the file to upload.';
         echo Form::file('image');
         echo Form::submit('Upload File');
         echo Form::close();
      ?>
   </body>
</html>
```

Step 2 − Create a controller called UploadFileController by executing the following command.

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

Step 3 − After successfully executing the command, you will receive the output.

Step 4 – Next, copy the following lines of code in the app/Http/Controllers/UploadFileController.php file. app/Http/Controllers/UploadFileController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Requests;
use App\Http\Controllers\Controller;

class UploadFileController extends Controller {
   public function index() {
      return view('uploadfile');
   }
   public function showUploadFile(Request $request) {
      $file = $request->file('image');

      //Display File Name
      echo 'File Name: '.$file->getClientOriginalName();
      echo '<br>';

      //Display File Extension
      echo 'File Extension: '.$file->getClientOriginalExtension();
      echo '<br>';

      //Display File Real Path
      echo 'File Real Path: '.$file->getRealPath();
      echo '<br>';

      //Display File Size
      echo 'File Size: '.$file->getSize();
      echo '<br>';

      //Display File Mime Type
      echo 'File Mime Type: '.$file->getMimeType();
      //Move Uploaded File
      $destinationPath = 'uploads';
      $file->move($destinationPath,$file->getClientOriginalName());
   }
}
```

Step 5 −Add the following lines in app/Http/routes.php.

```
Route::get('/uploadfile', 'UploadFileController@index');
Route::post('/uploadfile', 'UploadFileController@showUploadFile');
```

Step 6 – Test the upload file functionality.

eloquent

1. Eloquent is Laravel's built-in ORM (Object-Relational Mapping) system.
2. It allows you to interact with your database using PHP objects instead of writing SQL queries.
3. Eloquent models represent database tables and define relationships between them.
4. You can use Eloquent to create, read, update, and delete records in your database.
5. Eloquent provides a convenient and expressive way to build complex database queries.
6. It supports features like eager loading, pagination, and more for efficient data retrieval.
7. Eloquent models make it easy to work with related data through methods and relationships.
8. Laravel's Eloquent ORM is designed to streamline database interactions and improve code readability.
9. Eloquent is widely used in Laravel applications for database management and data manipulation.
10. Learning Eloquent is essential for developing Laravel applications efficiently.