# Laravel - Controllers

**Topics :** Laravel
**Written on** December 21, 2023

In Laravel, controllers play a vital role in handling HTTP requests and managing the application's logic. Controllers are responsible for processing user input, interacting with models, and returning appropriate responses. Here's a basic overview of controllers in Laravel:

## Creating Controllers:

You can create a controller using the Artisan command-line tool. Open your terminal and run:

```
php artisan make:controller MyController
```

This command generates a new controller class in the `app/Http/Controllers` directory.

## Controller Structure:

A typical controller in Laravel has methods that correspond to different actions. For example:

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class MyController extends Controller
{
    public function index()
    {
        // Logic for displaying the index page
    }

    public function show($id)
    {
        // Logic for displaying a specific resource
    }

    public function store(Request $request)
    {
        // Logic for storing a new resource
    }
```

```
    // ... other methods
}
```

### Routing to Controllers:

You can define routes in your `web.php` or `api.php` file to map to controller actions:

```
use App\Http\Controllers\MyController;

Route::get('/my-resource', [MyController::class, 'index']);
Route::get('/my-resource/{id}', [MyController::class, 'show']);
Route::post('/my-resource', [MyController::class, 'store']);
```

### Dependency Injection:

Laravel controllers often use dependency injection for services and other dependencies. For example, the `Request` object can be injected into a controller method:

```
public function store(Request $request)
{
    $data = $request->validate([
        'name' => 'required|string',
        'email' => 'required|email',
    ]);

   // Logic for storing a new resource
}
```

### Resource Controllers:

Laravel provides a convenient way to generate controllers for CRUD operations using the `--resource` option with the `make:controller` command:

```
php artisan make:controller MyResourceController --resource
```

This generates a controller with methods for handling CRUD operations (index, create, store, show, edit, update, destroy).

### Middleware:

You can apply middleware to controllers to perform actions before or after a controller method is called. Middleware is specified in the controller's constructor or using the `middleware` method:

```
public function __construct()
{
    $this->middleware('auth');
}
```

### Naming Conventions:

By convention, controller class names are in CamelCase, and the file name matches the class name. For example, `MyController` class is stored in `MyController.php`.

### Implicit Controller Route Binding:

Laravel provides implicit route model binding, allowing you to inject model instances directly into your controller methods. For example:

```
public function show(User $user)
{
    // $user is automatically resolved by Laravel
}
```