# Laravel

Laravel Request Response :

Request -> Route -> Controller -> Return -> Response

## Controller:

Controller files control the requests and responses

### Controller file (DemoController.php) ->

```php
<?php

namespace App\Http\Controllers;

class DemoController extends Controller {

    public function demo1() {
        return response( 'Hello World!' );
    }
}
```

### Route file (web.php) ->

```php
Route::get( 'demo1/', [DemoController::class, 'demo1'] );
```

## Return types in controller methods:

```php
<?php

namespace App\Http\Controllers;

class DemoController extends Controller {

    public function demo1() {

        return response( 'Hello World!' );

    }
```

```php
    // string
    public function demo2() {
        return response( 'This is a string!' );
    }

    // int
    public function demo3() {
        return response( 1 );
    }

    // null
    public function demo4() {
        return response( null );
    }

    // boolean
    public function demo5() {
        return response( true );
    }

    // array
    public function demo6() {
        return response( ['a', 'b', 'c'] );
    }

    // associative array or assoc
    public function demo7() {
        return response( [
            'name'    => 'Bill Gates',
            'company' => 'Microsoft',
        ] );
    }

    // multidimensional associative array
    public function demo8() {
        return response( [
            ['name1' => 'Bill Gates'],
            ['name2' => 'Microsoft'],
            ['name3' => 'Steve Jobs'],
        ] );
    }

    // binary large object (BLOB) -> files -> images, videos, etc; first save the
    file in public folder then do the operation
    public function demo9() {
        return response()->file( 'image.png' );
    }

    // download response -> no file preview, direct download after hitting route
```

```php
    public function demo10() {
        return response()->download( 'image.png' );
    }

    // redirect to another route
    public function demo11() {
        return redirect( 'demo2/' );
    }
}
```

```php
<?php


use Illuminate\Support\Facades\Route;
use App\Http\Controllers\DemoController;

Route::get( 'demo1/', [DemoController::class, 'demo1'] );
Route::get( 'demo2/', [DemoController::class, 'demo2'] );
Route::get( 'demo3/', [DemoController::class, 'demo3'] );
Route::get( 'demo4/', [DemoController::class, 'demo4'] );
Route::get( 'demo5/', [DemoController::class, 'demo5'] );
Route::get( 'demo6/', [DemoController::class, 'demo6'] );
Route::get( 'demo7/', [DemoController::class, 'demo7'] );
Route::get( 'demo8/', [DemoController::class, 'demo8'] );
Route::get( 'demo9/', [DemoController::class, 'demo9'] );
Route::get( 'demo10/', [DemoController::class, 'demo10'] );
Route::get( 'demo11/', [DemoController::class, 'demo11'] );
```

## Working with Laravel Request

Request Route Methods:

- GET
- POST
- PUT
- PATCH
- DELETE

```php
<?php


namespace App\Http\Controllers;
use Illuminate\Http\Request;

class RequestResponseLearningController extends Controller {
```

```php
    public function demo12( Request $request ) {
        $name = $request->name;
        $age  = $request->age;

        return response( ['name' => $name, 'age' => $age] );
    }

    public function demo13( Request $request ) {
        return response( $request->input() );
    }

    public function demo14( Request $request ) {
        return response( $request->input( "firstName" ) );
        // same:
        // return response( $request->firstName );
    }

    // with header
    public function demo15( Request $request ) {
        return response( $request->header() );
        // for specific header
        // return response( $request->header( "token" ) );
    }

    // query string
    public function demo16( Request $request ) {
        return response( $request->query() );
        // for specific
        // return response( $request->query( "name1" ) );
    }
}
```

```php
Route::get( 'demo12/{name}/{age}/', [RequestResponseLearningController::class,
'demo12'] );
Route::get( 'demo13/', [RequestResponseLearningController::class, 'demo13'] );
Route::get( 'demo14/', [RequestResponseLearningController::class, 'demo14'] );
Route::get( 'demo15/', [RequestResponseLearningController::class, 'demo15'] );
Route::get( 'demo16/', [RequestResponseLearningController::class, 'demo16'] );
```

for demo12 url : http://127.0.0.1:8000/demo12/John/Doe

for demo16 url : http://127.0.0.1:8000/demo16?id1=1&id2=2&name1=John&name2=Doe

# Working with multipart form data (Files with request):

**working with file (getting the file, size of the file, original/temporary name of the file, extension):**

```php
public function demo20( Request $request ) {

        // picking a image file
        $photo = $request->file( 'image' );

        // size of the file
        $fileSize = $photo->getSize();

        // getting original name of the file
        $fileOriginalName = $photo->getClientOriginalName();

        // getting temporary name of the file
        $fileTemporaryName = $photo->getFilename();

        // getting file extension
        $fileExtensionWithExtension                 = $photo->extension();
        $fileExtensionWithGetClientOriginalExtension = $photo-
>getClientOriginalExtension();

        return response( [
            'fileSize'                                => $fileSize,
            'fileOriginalName'                        => $fileOriginalName,
            'fileTemporaryName'                       => $fileTemporaryName,
            'fileExtensionWithExtension'              =>
$fileExtensionWithExtension,
            'fileExtensionWithGetClientOriginalExtension' =>
$fileExtensionWithGetClientOriginalExtension,
        ] );
    }
```

```php
Route::post( 'demo20/', [WorkingWithMultipartformdataController::class, 'demo20']
);
```

**Binary File Response:**

```php
public function demo29() {

        // getting the saved file path
        $filePath = "image.png";

        // to preview the file
        return response()->file( $filePath );
    }
```

```php
Route::get( 'demo29/', [WorkingWithRedirectionController::class, 'demo29'] );
```

**Binary File Download Response:**

```php
public function demo30() {

        // getting the saved file path
        $filePath = "image.png";

        // to download the file
        return response()->download( $filePath );
    }
```

```php
Route::get( 'demo30/', [WorkingWithRedirectionController::class, 'demo30'] );
```

**Moving the files to Storage/Public directory:**

```php
public function demo21( Request $request ) {

        // picking a image file
        $image = $request->file( 'image' );

        // to store the file in storage directory, with storeAs() method
        // storeAs(directory: 'uploads', save with the name: $image-
>getClientOriginalName() );
        $image->storeAs( 'uploads', $image->getClientOriginalName() );

        // to move the file in public directory, with move() method
        // move(directory: public_path(path: 'uploads' ), save with the name:
$image->getClientOriginalName() );
        $image->move( public_path( 'uploads' ), $image->getClientOriginalName()
);

        // keep that in mind, to save the file in both directory at once, first
execute for storage and then for public

        return true;
    }
```

```php
Route::post( 'demo21/', [WorkingWithMultipartformdataController::class, 'demo21']
);
```

# Laravel IP address and Content:

**ip address (ip address with request):**

```php
public function demo22( Request $request ) {

        // picking a image file
        $ipAddress = $request->ip();

        return response( $ipAddress );
    }
```

```php
Route::post( 'demo22/', [WorkingWithMultipartformdataController::class, 'demo22']
);
```

**Content Negotiations (Content Negotiations with request):**

```php
public function demo23( Request $request ) {

        // working with Content Negotiations
        // most of the time we use Accept header to pick the data format
        // there are many but mainly two types: text/html and application/json
        // mainly we work with application/json
        $acceptableContentTypes = $request->getAcceptableContentTypes();
        // will return an array with all the acceptable content types
        // for this case it will be [*/*], but we can even change this to
text/html or application/json etc, we can check on postman headers

        return response( $acceptableContentTypes );
    }
```

```php
public function demo24( Request $request ) {
                // content type check
        if ( $request->accepts( 'application/json' ) ) {
            return 1; // true
        } else {
            return 0; // false
        }
    }
```

```php
Route::post( 'demo23/', [WorkingWithMultipartformdataController::class, 'demo23']
);
Route::post( 'demo24/', [WorkingWithMultipartformdataController::class, 'demo24']
);
```

# Working with Cookie (cookie with request):

```php
// working with cookie
    public function demo25( Request $request ) {

        // getting all cookie value from request
        $cookieAll = $request->cookie();
        // getting specific cookie value from request
        $cookieSpecific = $request->cookie( 'Cookie_1' );

        return response( ['cookieAll' => $cookieAll,
            'cookieSpecific'            => $cookieSpecific] );
    }
```

```php
Route::post( 'demo25/', [WorkingWithMultipartformdataController::class, 'demo25']
);
```

## Returning Response in JSON format:

```php
// working with json response
    public function demo26( Request $request ) {

        $array = [
            "firstName" => "John",
            "lastName"  => "Doe",
        ];

        return response()->json( $array, 201 ); // by default the status code
will be set to 200 which means 'ok', if we don't even pass any status code, but
if we pass any, then that will be set as the status code
    }
```

```php
Route::post( 'demo26/', [WorkingWithMultipartformdataController::class, 'demo26']
);
```

## Redirecting a Response to one route to another route:

```php
public function demo27() {
        return redirect( 'demo28/' );
    }

    public function demo28() {
        return response( 'hello world!' );
```

```
        }
```

```
Route::get( 'demo27/', [WorkingWithRedirectionController::class, 'demo27'] );
Route::get( 'demo28/', [WorkingWithRedirectionController::class, 'demo28'] );
```

## Setting Cookie:

```
public function demo31() {

        // generating and setting cookie
        $cookieName      = 'Cookie_1';
        $cookieValue     = 'Value_1';
        $cookieTimeInMin = 120;
        $path            = '/'; // will work on entire application
        $domain          = $_SERVER['SERVER_NAME']; // dynamically getting the
server name
        $secure          = false;
        $httpOnly        = true;

        return response( "hello" )->cookie( $cookieName, $cookieValue,
$cookieTimeInMin, $path, $domain, $secure, $httpOnly );
    }
```

```
Route::get( 'demo31/', [WorkingWithCookiesController::class, 'demo31'] );
```

## Inserting Header:

```
public function demo32() {
        return response( "Hello" )
            ->header( 'key1', 'value1' )
            ->header( 'key2', 'value2' )
            ->header( 'key3', 'value3' );
    }
```

```
Route::get( 'demo32/', [RequestResponseLearningController::class, 'demo32'] );
```

## View:

## Log:

```php
public function demo34( Request $request ) {

    $num1 = $request->num1;
    $num2 = $request->num2;
    $sum  = $num1 + $num2;

    Log::info( $sum );
    Log::error( $sum );
    Log::warning( $sum );

    return response( $sum );
}
```

```php
Route::get( 'demo34/{num1}/{num2}', [LogController::class, 'demo34'] );
```

## Session:

```php
// to put a data in session
    public function sessionPut( Request $request ) {
        $email = $request->email;
        $request->session()->put( 'userEmail', $email );

        return true;
    }

    // to pull session data
    public function sessionPull( Request $request ) {
        // with pull method, session data can be retrieved only one time, then
that will be flushed
        $sessionPull = $request->session()->pull( 'userEmail', 'defualt' );

        return response( $sessionPull );
    }

    // to get session data
    public function sessionGet( Request $request ) {
        // with get method, session data can be retrieved many times, won't be
flushed
        $sessionGet = $request->session()->get( 'userEmail', 'defualt' );

        return response( $sessionGet );
    }

    // to forget a specific session data
    public function sessionForget( Request $request ) {
        $request->session()->forget( 'userEmail' );
```

```php
        return true;
    }

    // to flush all session data
    public function sessionFlush( Request $request ) {
        $request->session()->flush();

        return true;
    }
```

```php
Route::get( 'demo35/{email}', [SessionController::class, 'sessionPut'] );
Route::get( 'demo36/', [SessionController::class, 'sessionPull'] );
Route::get( 'demo37/', [SessionController::class, 'sessionGet'] );
Route::get( 'demo38/', [SessionController::class, 'sessionForget'] );
Route::get( 'demo39/', [SessionController::class, 'sessionFlush'] );
```

## Middleware:

How we define middleware:

- First we create middleware
- Define the logics in that middleware
- Create alias of that middleware class, or give a name of that middleware class, in bootstrap folder -> app.php file
- Then we call that middleware by that alias name, with the routes at the end, or we can even create groups

php artisan make:middleware 'miidlewareName'

php artisan make:middleware DemoMiddleware

**with Header property check:**

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class DemoMiddleware {
    /**
     * Handle an incoming request.
```

```
         *
         * @param   \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
         */
    public function handle( Request $request, Closure $next ): Response {
        $key = $request->header( 'apiKey' );
// condition to check
        if ( $key == 1234 ) {
            return $next( $request );
        } else {
            return response()->json( 'unauthorized', 401 );
        }
    }
}
```

```
<?php

namespace App\Http\Controllers;

class MiddlewareController extends Controller {

    public function demo40() {
        return "hello";
    }
}
```

```
Route::get( 'demo40/', [MiddlewareController::class, 'demo40'] )->middleware(
[DemoMiddleware::class] );
```

**Redirection with middleware:**

```
public function handle( Request $request, Closure $next ): Response {

        $key = $request->key;

// condition to check
        if ( $key == 1234 ) {
            return $next( $request );
        } else {
            return redirect( 'demo41/' );
        }
    }
```

```
public function demo40() {
```

```
        return "hello1";
    }

    public function demo41() {

        return "hello2";
    }
```

```
Route::get( 'demo40/{key}', [MiddlewareController::class, 'demo40'] )-
>middleware( [DemoMiddleware::class] );
Route::get( 'demo41/', [MiddlewareController::class, 'demo41'] );
```

**Middleware Groups:**

Directly declaring the class:

```
// directly declaring the class
Route::middleware( [DemoMiddleware::class] )->group( function () {
    Route::get( 'demo41/{key}', [MiddlewareController::class, 'demo41'] );
    Route::get( 'demo42/{key}', [MiddlewareController::class, 'demo42'] );
    Route::get( 'demo43/{key}', [MiddlewareController::class, 'demo43'] );
    Route::get( 'demo44/{key}', [MiddlewareController::class, 'demo44'] );
} );
```

Using the alias name, after declared the class in the app.php

```
// using the alias name, after declared the class in the app.php
Route::middleware( ['demoMiddleware'] )->group( function () {
    Route::get( 'demo41/{key}', [MiddlewareController::class, 'demo41'] );
    Route::get( 'demo42/{key}', [MiddlewareController::class, 'demo42'] );
    Route::get( 'demo43/{key}', [MiddlewareController::class, 'demo43'] );
    Route::get( 'demo44/{key}', [MiddlewareController::class, 'demo44'] );
} );
```

With alias(),

```
<?php

use Illuminate\Foundation\Application;
use App\Http\Middleware\DemoMiddleware;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure( basePath: dirname( __DIR__ ) )
    ->withRouting(
```

```php
        web: __DIR__ . '/../routes/web.php',
        commands: __DIR__ . '/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware( function ( Middleware $middleware ) {
        $middleware->alias( [
            'demoMiddleware' => DemoMiddleware::class,
        ] );
    } )
    ->withExceptions( function ( Exceptions $exceptions ) {
        //
    } )->create();
```

to apply middleware in all routes declared in web.php or api.php (go to bootstrap file -> app.php file):

```php
<?php

use Illuminate\Foundation\Application;
use App\Http\Middleware\DemoMiddleware;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure( basePath: dirname( __DIR__ ) )
    ->withRouting(
        web: __DIR__ . '/../routes/web.php',
        commands: __DIR__ . '/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware( function ( Middleware $middleware ) {
        $middleware->web( append: [
            'demoMiddleware' => DemoMiddleware::class,
        ] );
        $middleware->api( prepend: [
            'demoMiddleware' => DemoMiddleware::class,
         ] );
    } )
    ->withExceptions( function ( Exceptions $exceptions ) {
        //
    } )->create();
```

**Manipulate Headers in Middleware:**

**Adding Header in middleware:**

```php
<?php
```

```php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class DemoMiddleware {
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle( Request $request, Closure $next ): Response {
        //adding header from middleware, after hitting the route, it will come to
middleware and this header will be added
        $request->headers->add( ['email' => 'john@gmail.com'] );
        return $next( $request );
    }
}
```

```php
public function demo45( Request $request ) {
        return $request->header();
    }
```

```php
Route::get( 'demo45', [MiddlewareController::class, 'demo45'] )->middleware(
[DemoMiddleware::class] );
```

**Replacing Header in middleware:**

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class DemoMiddleware {
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle( Request $request, Closure $next ): Response {
        //Replacing header from middleware, after hitting the route, it will come
```

```
to middleware and this header will be replaced, to that previously set header
        $request->headers->replace( ['email' => 'john@gmail.com'] );
        return $next( $request );
    }
}
```

```
public function demo45( Request $request ) {
        return $request->header();
    }
```

```
Route::get( 'demo45', [MiddlewareController::class, 'demo45'] )->middleware(
[DemoMiddleware::class] );
```

**Removing Header in middleware:**

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class DemoMiddleware {
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle( Request $request, Closure $next ): Response {
        //Removing header from middleware, after hitting the route, it will come
to middleware and this previously set header will be removed
        $request->headers->remove( 'email' ); // for removing, we need to only
set the key
        return $next( $request );
    }
}
```

```
public function demo45( Request $request ) {
        return $request->header();
    }
```

```
Route::get( 'demo45', [MiddlewareController::class, 'demo45'] )->middleware(
[DemoMiddleware::class] );
```

**Request rate limiting in middleware (for one route):**

```php
public function demo46() {
        return "hello";
    }
```

```php
// This was for working with only one route
Route::get( 'demo46', [MiddlewareController::class, 'demo46'] )->middleware(
'throttle:5,1' );
```

**Request rate limiting in middleware (for entire application, web.php):**

```php
$middleware->web( append: [
        'throttle:10,1',
    ] );
```

# To get a view as response:

```php
public function demo33() {

        // go to views folder -> create component folder -> create home.blade.php
        return view( 'component.home' );
    }
```

```php
Route::get( 'demo33/', [ViewController::class, 'demo33'] );
```

# Constructor in Controller:

As we all know, constructors executed by themselves automatically at the beginning, no need to call particularly. We can define middleware through these constructors. Like - if we want that a middleware will work only in a specific controller, then for that controller we can add that middleware, no need to add middleware with the routes particularly.

# Controller Types:

- **Basic Controller**

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class BasicController extends Controller {

    public function demo47( Request $request ) {
        $name = $request->name;

        return response( $name );
    }
}
```

```php
Route::get( 'demo47/{name}', [BasicController::class, 'demo47'] );
```

- **Single Action Controller** : There will be only one function and it's executed by a method called _invoke()__ :
- php artisan make:controller SingleActionController --invokable

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class BasicController extends Controller {

    public function demo47( Request $request ) {
        $name = $request->name;

        return response( $name );
    }
}
```

In route, no need to take an array [] to pass the controller class and the method. Here for this, just we need to mention the class.

```php
Route::get( 'demo47/{name}', SingleActionController::class );
```

- **Resource Controller** : There will be generated all the methods, which we need for CRUD operations

- php artisan make:controller ResourceController --resource

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ResourceController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //
    }

    /**

     * Show the form for creating a new resource.

     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store( Request $request )
    {
        //
    }

    /**
     * Display the specified resource.
     */
    public function show( string $id )
    {
        //
    }

    /**
     * Show the form for editing the specified resource.

     */
    public function edit( string $id )
```

```
    {
        //
    }

    /**
     * Update the specified resource in storage.
     */
    public function update( Request $request, string $id )
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy( string $id )
    {
        //
    }
}
```

In route, no need to take an array [] to pass the controller class and the method. Here for this, just we need to mention the class and the basic endpoint, rest resource method will generate automatically. Keep that in mind, methods generated in ResourceController should not be changed at any cost, otherwise routes won't work.

```
Route::resource( 'demo49/', ResourceController::class );
```

Route Lists for that ResourceController:

```
GET|HEAD        demo49 ........................................ index ›
ResourceController@index
  POST          demo49 ........................................ store ›
ResourceController@store
  GET|HEAD      demo49/create ................................. create ›
ResourceController@create
  GET|HEAD      demo49/{} ..................................... show ›
ResourceController@show
  PUT|PATCH     demo49/{} ..................................... update ›
ResourceController@update
  DELETE        demo49/{} ..................................... destroy ›
ResourceController@destroy
  GET|HEAD      demo49/{}/edit ................................ edit ›
ResourceController@edit
```

**Blade:**

```php
<?php

namespace App\Http\Controllers;

class BladeController extends Controller
{
    public function demo51()
    {
        // multidimensional associative array
        $billGates = [
            ["firstName" => "Bill", "lastName" => "Gates", "age" => 68],
            ["firstName" => "Mark", "lastName" => "Zuckerberg", "age" => 62],
            ["firstName" => "Jeff", "lastName" => "Bezos", "age" => 57],
        ];

        return view( 'demo51', compact( 'billGates' ) );
        // can also write without compact() function:
        // return view( 'demo51', ['billGates' => $billGates] );
    }
}
```

in demo51.blade.php in views folder:

```html
<body>

    <h1>This is demo 51</h1>

    <ol>
        @foreach ($billGates as $person)
            <li>
                User name is = {{ $person['firstName'] }} and age is = {{ $person['age'] }}
            </li>
        @endforeach
    </ol>

</body>
```

```php
Route::get( 'demo51/', [BladeController::class, 'demo51'] );
```