PHP

1. Echo:

This keyword is used to print.

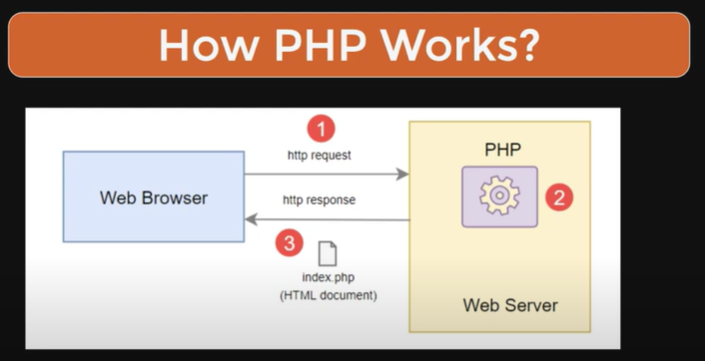
1. Variables:
2. variable starts with $ sign
3. variable name must cannot start with number
4. It must start with letter or underscore character
5. It should not start with special chars
6. It can only contain alpha-numeric chars and underscore
7. $name, $Name and $NAME are different variables

means variable is case sensitive

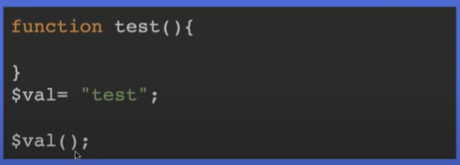
1. Write HTML inside PHP:

We can write html code in PHP code using “”.

1. By default, if any error will come in php then error will not display on browser to display error, we need to go to php.ini file and then find display\_error and set it on.
2. With the help of **const** key word, we can declare constant in php and another way is **define()**
3. Datatypes in php – String, Integer, float, Boolean, Null, Array, Object(class), Recourse
4. How PHP works:



1. Variable functions:



1. 10 Basic important functions in PHP
2. strlen($str);
3. date(“l”);
4. phpinfo()
5. is\_string($var);
6. is\_int()
7. rand()
8. die()/die;
9. exit()/exit
10. substr();
11. dfdfgdfg
12. Array:

Can store multiple values in single variable

1. Super global variables in PHP



Laravel 11

1. In **web.php** routes related information are stored. **Routes -> web.php**
2. In which folder views are stored? **Resources -> views**
3. **How to pass parameter from directly routes/url to view?**

In web.php:

**Route::get('/sendValueUrlToRoutesToView/{value}', function($viewValueHolderVar) {**

**return view('valueUrlToViews', ['viewValueHolderVar' =>$viewValueHolderVar]);**

**});**

In view file:

**<h1>This value is direclty coming from URL without controller: {{ $viewValueHolderVar }}</h1>**

Note: Do not forget to use $ on view file else we will get error.

1. **How we can create controller using PHP command:**

* php artisan make:controller ControllerName
* **Resource Controller** (For CRUD operations): php artisan make:controller BlogController –resource

This will generate methods like index, create, store, show, edit, update, and destroy.

* **API Resource Controller** (Without create & edit methods):

php artisan make:controller BlogController –api

1. **Pass value url parameter to view through controller:**

**UserController:**

function getViewWithParameter($name) {

        echo 'Hello from view with parameter: '. $name;

        return view('viewWithParameter', ['parameterValue' => $name]);

    }

**Web.php:**

Route::get('getViewPageWithParameter/{parameter}', [UserController::class, 'getViewWithParameter']);

**viewWithParameter.blade.php**:

<h1>This is view which display input parameter and parameter is : <span style="color:yellowgreen;">{{ $**valueWeCanKeenAnyNameHere** }}</span></h1>

1. **Print value directly using controller without using view:**

**UserController.java**

function getName($name) {

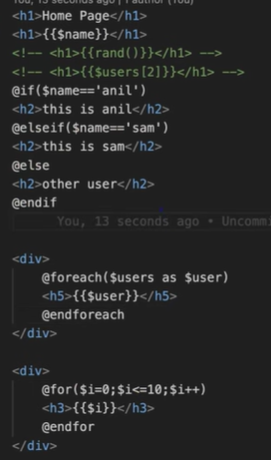
        return 'Hello this is '. $name;

    }

**Web.php**

Route::get('getName/{name}', [UserController::class, 'getName']);

1. If, else and for and foreach loop in blade file:



1. **URL generation:**

{{ <URL::current()> }}

{{ <URL::full()> }}

{{ url()->current() }}

{{ url()->full() }}

{{ <URL::previous()> }}

<a href=”{{URL::to(‘about’)}}”>About Page</a>

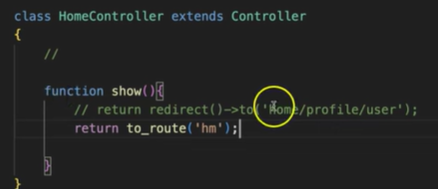
<a href=”{{URL::to(‘about’, [‘vikash’])}}”>About Page</a>

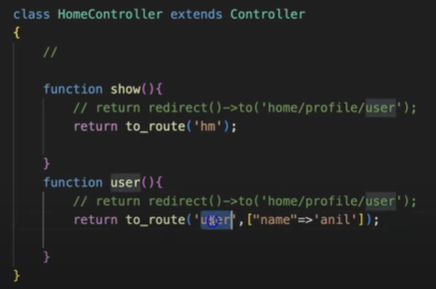
1. **Named routes:**

We used named routes to define short name of a complex routes.

Route::view(‘home/profiler/user’, ‘home’)->name(‘hm’);

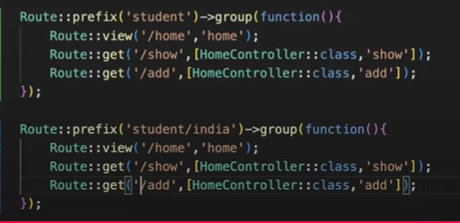
<a href=’{{route(‘hm’)}}’>Go to Home page</a>





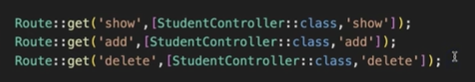


1. **Route group with prefix:**



1. **Route group with controller:**

To remove this below:



Use this:



1. **Middleware/Filter:**

Middleware is same like filter in Java. To create middle ware we can use – php artisan make:middleware AgeCheck

Then open **bootstrap -> app.php**

->withMiddleware(function (Middleware $middleware) {

        $middleware -> append(AgeCheck::class);

    })

class AgeCheck

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next

     \*/

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

    {

**echo 'Welcome from Middleware!<br>';**

**$age = $request-> age;**

**if(!empty($age) && $age < 18) {**

**echo 'Value of age from pathparame: '.$age.'<br>';**

**die("<span style='background-color:#ff00004f; color:red;'>You can not access this page</span>");**

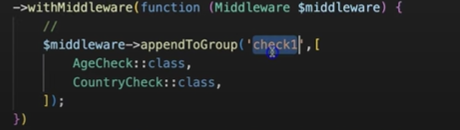
**}**

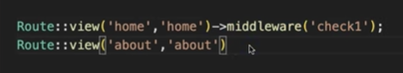
**return $next($request);**

    }

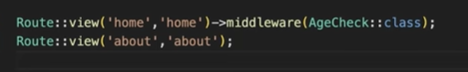
}

1. Middleware group:





How to apply single Middleware.



1. In Laravel just create a model using command – **PHP artisan make:model Student**

Then it will create a model and if a table name with **Students** will be available in db then it will be auto connected with database and automatically all the columns and attributes will be mapped no need to map column and class property.

In Model class we can write out own function also and there will be pre-defined methods as well since this model class we extended from **Model** class.

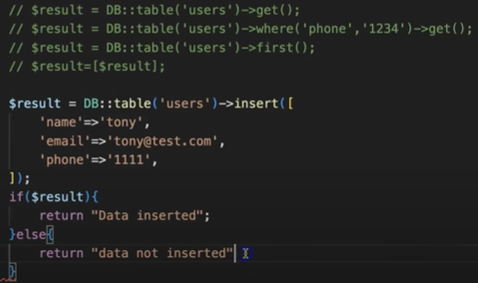
To get all student we need to write **Student::all()**, it will return all students records.

And if we want to access the out custom method then we will have to write **$dummyFunctiontxt = new Student();** and then to access this method **$dummyFunctiontxt -> dummy();**

1. We can see all the details of database table Student using command **php artisan model:show Student**.
2. We can user query builder to get all the student as well using **$students = DB::table('students') -> get();**
3. To get particular class value **$students = DB::table('students') -> where('class', $class) -> get();**

And then return **return response() -> json($students);**

1. Other db related operations:



Update:

