Laravel IMP

1. How to create a route in Laravel:
2. Syntax:

1 method🡺

Route::get('/url',function(){

    return view('page name');

});

1. Syntax:

2 method🡺

Route::view( '/url', 'page name' );

1. Now Example of route:

//home route

Route::get('/', function () {

    return view('welcome');

});

//about route

Route::view("about","about");

1. What if we create same route url with different view name:

//home route

Route::view('/',"welcome");

//contact route

Route::view('/',"contact");

**Fact: It will consider the last one (Updated one) with program flow**

1. How to create a router’s like one page pointing to other and so on… :

//home route

Route::view('/',"welcome");

//contact route

Route::view('/contact',"contact");

//about route

Route::view("/contact/about","about");

**Output:**

****

****

1. How to send data with route(including in url itself):
2. Simply echoing on the top of page itself:

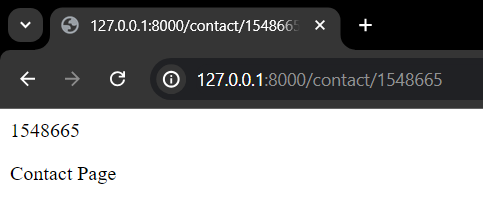
//how to send data throw route

Route::get("/contact/{contact\_number}", function($contact\_number){

    echo($contact\_number);

    return view("contact");

});



1. Now how we can provide it in blade.php page

For printing in php file we are using {{ }} braces with variable name

Syntax: {{$variable\_name}} ==> it will print date in variable

//how to send data throw route

Route::get("/contact/{contact\_number}", function($contact\_number){

    return view("contact",['ph\_no'=>$contact\_number]);

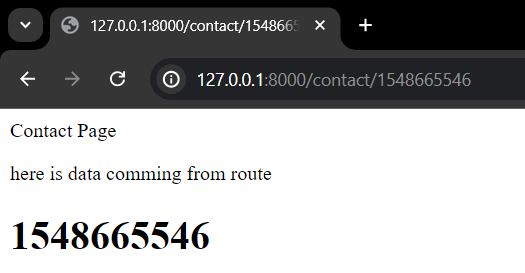
});

//contact.blade.php==>>

<p>Contact Page</p>

<p>here is data comming from route</p>

<h1>{{$ph\_no}}</h1>



1. Now how to navigate in browser without changing url (using Ancher tag):

//Web.php =>

//home route

Route::view('/',"welcome");

//contact route

Route::view('/contact', "contact");

//about route

Route::view("/about","about");

//welcome.blade.php=>

<p> Welcome Page </p>

<a href="/about">About Page</a></br>

<a href="contact">Contact Page</a>

1. How to redirect to some page:

Simply changing home page

Syntax:

Here we are passing page name not url

**redirect(‘blade\_name’);**

If user try to add this url: <http://127.0.0.1:8000/>

Then also automatically this url will open: http://127.0.0.1:8000/contact

//home route

Route::get('/',function(){

    // return view('welcome');

    //how to redirecting to some other page

    return redirect('contact');

});

1. How to create Controller in Laravel
2. Using command:

Syntax:

**php artisan make:controller controller\_name**

after this controller automatically created at place **/app/Http/Controllers/here**

ex:

Laravel/first-project (main)

$ php artisan make:controller user

INFO Controller [Laravel\first-project\app\Http\Controllers\user.php] created successfully.

1. Code inside controller looks like:

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class user extends Controller

{

    //code

}

?>

1. How to create function in controller

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class user extends Controller

{

    //function in controller

    function printing(){

        echo "This is controller echoing";

    }

}

?>

1. How to call controller:

To call controller we need to write code in web.php

1. Import controller class
2. Create route for that with syntax:

Route::get(“url”, [ControllerClassName::class, “function\_name”] );

//web.php

//importing controller

use App\Http\Controllers\user;

//calling controller after laravel 8

/\*

syntax:

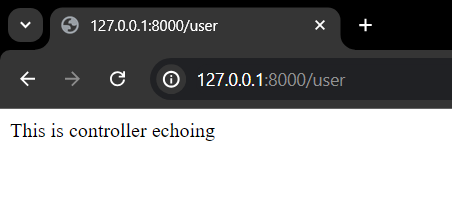
//importing controller

use App\Http\Controllers\user;

Route::get('url', [ControllerClassName::class,"FunctionWhichYouWantToCall"] );

\*/

Route::get("user",[user::class,"printing"]);



1. How does controller called before laravel 8:
2. By creating route only like this

/\*

syntax was:

Route::get("url","ControllerClassName@FunctionName");

\*/

Route::get("laravel7","user@printing");

Why it was changed?

* **This change enhance PHP type safety features and provide better support for IDE’s to offer code completion and type hinting**
* **So this change was made un Laravel 8 to aim to provide modern and robust development experience**

1. How to send data from controller url
2. Accept an variable and print it using echo

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class user extends Controller

{

    //function in controller

    function printing($comming){

        echo "$comming This is controller echoing";

    }

}

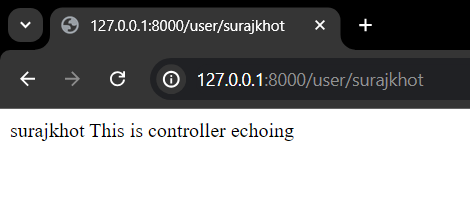
?>

1. Make change in web.php to accept data

//web.php

//passing data from url

Route::get("user/{username}",[user::class,"printing"]);



1. How to call a view from controller:
2. Create it by using cli:

**$ php artisan make:controller DemoController**

INFO Controller [D:\IMPMyWorkplace\Laravel\first-project\app\Http\Controllers\DemoController.php] created successfully.

1. Made Change in controller file:

//DemoController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class DemoController extends Controller

{

    //calling view from controller

    function loadView(){

        //simply returning about view

        return view('about');

    }

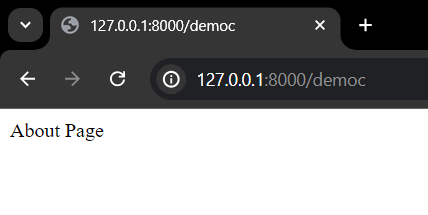
}

1. Make route for it

//Web.php

//democontroller route

Route::get("democ",[DemoController::class,"loadView"]);



1. How to pass data from controller to view:
2. demoController.php

// DemoController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class DemoController extends Controller

{

    //calling view from controller

    function loadView($arg){

        //simply returing about view

        return view('about',['argument'=>$arg]);

    }

}

1. create route with argument

//web.php

//democontroller route

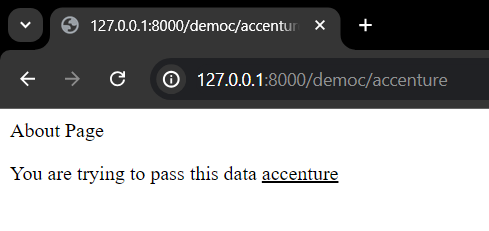
Route::get("democ/{company}",[DemoController::class,"loadView"]);

1. now display in about.blade.php

// about.blade.php

<p> About Page </p>

<p> You are trying to pass this data <u>{{$argument}}</u> </p>



1. how to create component:
2. fire command in root directory

**$ php artisan make:component DemoComponent**

INFO Component [D:\IMPMyWorkplace\Laravel\first-project\ap

p\View\Components\DemoComponent.php] created successfully.

Two files are created after it

- **componentName.php (app/view/components/xxx)**

- **componentName.blade.php**  **(resources/views/component/xxx)**

1. Here code inside component

// first-project\app\View\Components\DemoComponent.php

<?php

namespace App\View\Components;

use Closure;

use Illuminate\Contracts\View\View;

use Illuminate\View\Component;

class DemoComponent extends Component

{

    /\*\*

     \* Create a new component instance.

     \*/

    public function \_\_construct()

    {

        //

    }

    /\*\*

     \* Get the view / contents that represent the component.

     \*/

    public function render(): View|Closure|string

    {

        return view('components.demo-component');

    }

}

// first-project\resources\views\components\demo-component.blade.php

<div>

//code

</div>

1. How to use components

It will allow us to use same code in every file with on line of code

Syntax:

**<x-ComponentName/>**

1. Write code in blade.php

//demo-component.blade.php

<div>

    <p> This is code from component and this file is in resources/views/commponents/ directory </p>

</div>

1. Using in our views

Contact.blade.php

<p>Contact Page</p>

<x-DemoComponent/>

Welcome.blade.php

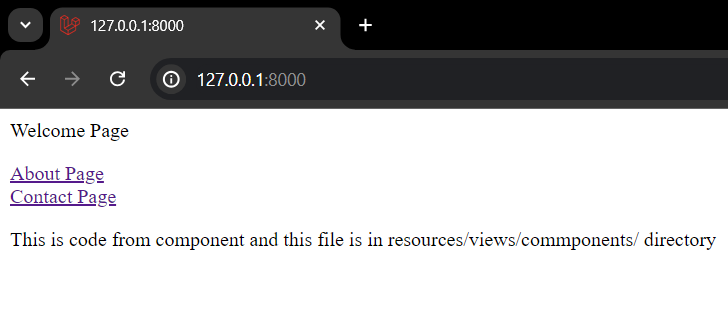
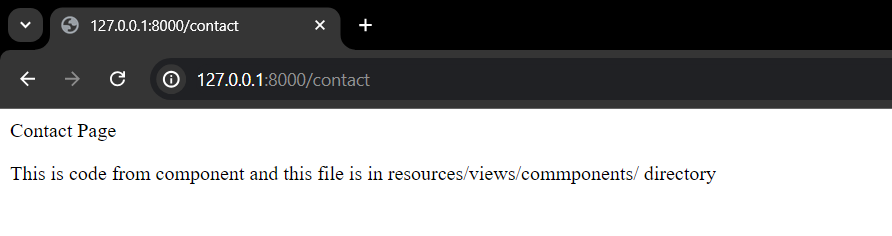
<p> Welcome Page </p>

<a href="/about">About Page</a></br>

<a href="contact">Contact Page</a>

<x-DemoComponent/>

1. And do not forgot to create routes for those views and Outputs :



1. How to pass data through component
2. Create variable(use public) and use this keyword to refer to argument coming from \_\_construct

Syntax :

public $variableName;

    public function \_\_construct($argument)

    {

       $this->variableName=$argument;

    }

// DemoComponent.php

<?php

namespace App\View\Components;

use Closure;

use Illuminate\Contracts\View\View;

use Illuminate\View\Component;

class DemoComponent extends Component

{

    /\*\*

     \* Create a new component instance.

     \*/

    //publicly defining variable

    public $title;

    public function \_\_construct($arg)

    {

        //Refers to the current object and for further use use variable \*\*\*\*title\*\*\*\*\* not arg

       $this->title=$arg;

    }

    /\*\*

     \* Get the view / contents that represent the component.

     \*/

    public function render(): View|Closure|string

    {

        return view('components.demo-component');

    }

}

1. Now print data in components blade file by using variable name not a argument

//demo-component.blade.php

<div>

//printing here a data title

    {{$title}}

    <p> This is code from component and this file is in resources/views/commponents/ directory </p>

</div>

1. Now pass data using argument(use without $ sign)

And use the correct parameter name

Contact.blade.php

<p>Contact Page</p>

<x-DemoComponent arg="This is argument passing here in Contact Page"/>

Welcome.blade.php

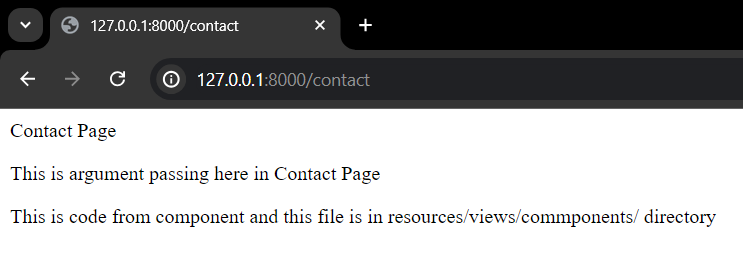
<p> Welcome Page </p>

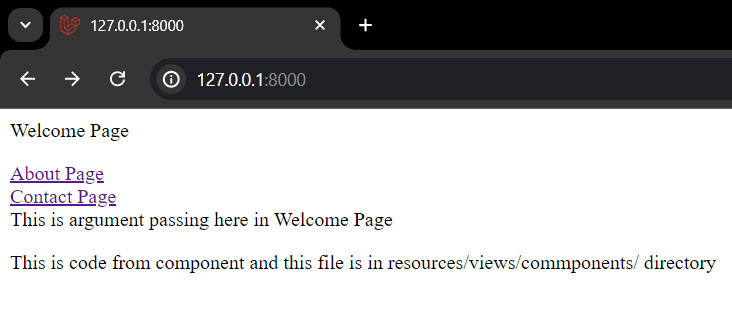
<a href="/about">About Page</a></br>

<a href="contact">Contact Page</a>

{{-- do not include $ here with variable --}}

<x-DemoComponent arg="This is argument passing here in Welcome Page"/>





1. Now Its time to learn blade template engine :
2. How to comment in blade

Syntax:

**{{-- commenting --}}**

Same for single line and multiline

Ex:

{{-- this is comment in blade --}}

1. How we use php in blade template

We can not directly add php in blade we have to use such thing like php tags and for printing {{ }} braces like this. 🡺

**For printing php in blade**

1. Double braces

{{ //code }}

1. Php tags

<?php

//code

?>

1. Php attribute tags

@php

//code

@endphp

Ex:

//welcome.blade.php

{{-- 1. operations --}}

10+2  {{--  not work --}}

{{10+2}}  {{-- 12 --}}

@php

echo 10+2; //12

@endphp

<?php

echo 10+2; //12

?>

1. If conditions:
2. Simply using blade format

{{-- if condition --}}

<?php

$a = 10;

$b = 20;

if ($a > $b) {

    echo "$a>$b";

} elseif ($b > $a) {

    echo "$b>$a";

} else {

    echo "$a==$b";

}

?>

@if ($a > $b)

    {{ "$a>$b" }}

@elseif ($b > $a)

    {{ "$b>$a" }}

@else

    {{ "$a==$b" }}

@endif

1. Using controller
2. Create a controller

$ php artisan make:controller PhpExampleController

1. Pass data and load view

//PhpExampleController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PhpExampleController extends Controller

{

    function loadView()

    {

        $a = 10;

        $b = 20;

        $arr = [12, 45, 65, 74, 68, 12];

        return view('php\_code', ['a' => $a, 'b' => $b, 'arr' => $arr]);

    }

}

1. Create route:

//php code controller

Route::get("php", [PhpExampleController::class, "loadView"]);

1. Code for blade

//php\_code.blade.php

{{-- data comming from controller --}}

<h1>data comming from controller</h1>

{{ "a:$a \n b:$b \n array:arr" }}

<br>

{{-- if condition  --}}

@if ($a > $b)

    {{ "$a is greater than $b" }}

@elseif($b > $a)

    {{ "$a is leass than $b " }}

@endif

<br>

<h1>length of array</h1>

{{ count($arr) }}

<br>

<h1>for loop</h1>

{{-- for loop  --}}

@php

    $len = count($arr);

@endphp

@for ($i = 0; $i < $len; $i++)

    <h5> {{ $arr[$i] }}</h5>

@endfor

<br>

<h1>for each loop</h1>

{{-- for each loop  --}}

@foreach ($arr as $value)

    <h5>{{ $value }}</h5>

@endforeach