**Day 1**

Required installations

* Xampp
* Composer
* Git bash
* Php storm( text editor)

To check proper installation of the composer , go to cmd prompt, type “compser” enter

Successful installation.

Go to git bash , launch it, change directory to c, go to xampp/htdocs folder

Create a laravel page using composer - **composer create –project laravel/laravel filename --prefer-dist/5.2(versions)**

If error occurred as showing composer command not found-

* Change environmental variables,- go to advance system settings –path- edit- apply-apply-ok
* Or run git bash as administrator-(git bash launch right click- run as administrator)

Initial settings for xampp

* Xampp control panel start apache to check the port numbers, -config – change port numbers from 80 to 3000
* Config –ssl – change port number 443 t0 1443
* Check other settings and config and save.

Install php storm which is a text editor where we actually write the codes

**Day 2**

Once the file is created we can go and check that in xampp- htdocs – filename –(createdf ile)

Onto the **browser “localhost:3000”**

Php installation page appears

On a new browser” **localhost/filename” created laravel page appears**

In the php storm created file appears – as far as concerned all the database related files of the file created is present in the**” app**” menu

Majorly we concentrate on**” http**” menu

We write the database constants that we plugin to the page in **.env** file

**“Public** “ menu has all css files

We use **sass** menu to compile our css file

After compilation the output goes to **“public** “ menu

In laravel if we type “**filename.dev “**it takes to the root

First inorder to create a route to the page in php storm go **the filename- apps – http – routes.php**

**In routes .php -**  we specify the what type of path the user need to approach to get that page , sohere we create a method, like

**Route::get(‘/path/ (array can be used as optional to wrap up all) function() {**

$(begin tag) url ( It specifies the complete path on display) = admin.home( as given by the user in array specified In the previous route like array (‘as’=>’admin.home’). Therefore it looks like

**$url=admin.home**

Return’ this is url; ( it specifies what displays on the screen)

So ,code will be like

***Route::get(‘admin/home/example ‘ array(‘as’=>admin.home function()***

***{***

***$url =sdmin.home***

***Return “/the url is” .$url***

***}***

***));***

Generally , return acts as printf statements

Now, go to the terminal of the phpstorm go to that file which we created using cd command like **cd filename/**

In that, type **php artisan route:list** then the list appears…..

General lectures

Of course, sometimes you will need to capture segments of the URI within your route. For example, you may need to capture a user's ID from the URL. You may do so by defining route parameters:

Route::get('user/{id}', function ($id) {

return 'User '.$id;

});

For giving constraints we use **where** class to the constraints as specified

Where=>(‘{id}/{name}’ [0-9]+ [a-z]+)

Once you have assigned a name to a given route, you may use the route's name when generating URLs or redirects via the global route function:

// Generating URLs...

$url = route('profile');

// Generating Redirects...

return redirect()->route('profile');

CSRF – ( cross site request forgery) is a type of protection which denies unauthenticated users to make changes.

Permission methods f r the access:

**Router Methods**

* Get
* Put
* Post
* Patch
* Delete
* Options

**Day 3**

**Controllers: controllers act as middlemen between the model(classes) and the view**

They are located at **app- http – controllers** in php storm

“**Namespace”** is a keyword that can be used to a class which may occur at different places of a program with same name but different functionalities, so that when a function is called ,no errors will be generated

“**USE**” keyword is as same as the same use in oops which import sources from required directories or libraries

To create a new php file**, controllers- new- php file- filename – Ok**

**Or** by using terminal command **php artisan make: controller( folder) Post controller ( file you wanna create under it )**

“ **Make”** is a command to create

By using the keyword “—resources “ we can add delete or modify the contents of the file created.

EG: php artisan make: controller –resources postcontroller

Resources contains , **index, create, store , show , edit, update and destroy.**

**Controllers lecture :** [**http://laravel.com/docs/5.2/controllers**](http://laravel.com/docs/5.2/controllers)**.**

Index is used to displays whatever the user gives on the browser in the view, it uses pass by parameter value.

Eg1 : router ::get(‘/post/’{id} , postcontroller**@index( creates a request –bolded word**))

;

Here we can enter the number as ‘id’ is referencing of the type numbers

Eg2 : router: :get(/post/{name} , postcontroller@index);

Here we can enter the alphabets

Route :: resources(‘posts’, ‘postcontroller’); - creates all the parameters of the resources on the browser

Command **: php artisan routes: list**

Lists all the resources with their names and methods which are created by resources.

Command : **php artisan**

Lists all the commands of the php that we can perform

Command : **type-hint**

List all the repositories to resolve issue.

Command : **php artisan route: cache**

To generate a route cache

Command : **php artisan route: clear**

To clear all the previously created route cache files.

**VIEWS**

Views are located at resources –views

So we create the views app – resources- view rightclick –php file – filename.**blade**.php(with extension) (where html codes are written for the view of the page)- ok

Create a route for the view in the routes folder – route::get(‘/filename’,controllermethod@contact);

So the method has to be defined in the controller which has a php file.

**Public function contact()**

**{**

**Return view (‘contact’);**

**}**

How to pass data to the views:

* **Blade engine is a templating engine that laravel uses to show php execution in a very simple way**
* **We can chain the functions or the method with the parameter using (“->with” keyword) it takes two parameters ,variable name and variable value itself.**

**Eg: public function show\_post(id)**

**{**

**return view (‘post’)->with(id, $id);**

**}**

* **And make changes in the post blade view page by entering curly braces**

**eg: <h1> post {{$id}}</h1>**

* **There is a native function called** *compact*, **as we can pass many parameters that has same name and variable it will convert it into the variable.**
* **Eg: public function show-post($id, $name, $password)**

**{**

**Retrun view (‘post’) compact {‘id’ , ‘name, ‘ password’);**

**}**

Blade is a php templating agent .

**Database**

**Command:** php artisan make:migration add \_is\_admin\_column \_to\_posts\_table ---table =”posts”(table name)

Path **: App – database- migration.**

To refresh and to get updated one click on synchronize which is there at rightclick of the migrations. **Command :** php artisan migrate to migrate the table

In order to create the new migration file

Command : php artisan make:migration create\_posts\_table –create(flag whih comes with 2 --) = “posts”.

No sensitive imformation has to be kept n the .env(environmental )file because chances that git bash ignores.

When timestrap is used in the creation of the database it generates the **“created at” and “updated** **at”** in the database table

Command: **php artisan migrate:reset** it will rollback all the previous migrations/delete all the migrations

Command: **php artisan migrate:refresh** it will automatically rollback all the tables and migrate them.

**RAW SQL**

Route::get(‘/update’,function(){

$update =DB::update(‘update post set title =”update the title” where id= ?’,[1]);

Return $updated;

}

Route ::set(‘/delete’, function()

{

$delete = DB::delete(‘delete from Posts where id-?’[1]);

Return $deleted;

}

Route::get(‘/insert’, function()

{

DB::insert(‘insert into posts(posts here is a tablename)(title, content) values(?,?)’,[‘PHP with Laravel’,’Laravel is a best thing]);

}

Route::get(‘/read’, function()

{

$results = DB::select(“select \* from Posts where id=?’,[1]);

Return $results

});

**Var\_dump** – displays the content with its type and number of characters in it.

Route:;get(‘/update’, function()

{

$updated =DB::update(update posts set title =”update new title”where id=’?’,[1]);

Return $updated;

});

Create a page:

Route::get(‘/create’, function()

{

Post::create([‘title’=>’create method’, ‘content’ => ‘I am learning’]);

}

);

Route::get(‘/forcedelete’, function()

{

Post::withtrashed()->where(is admin ,0)->forcedelete);

}

**); It deletes the entire method**

Route::get(‘/delete’, function()

{

$post= Post::find(3) ; **[here we can pass parameters to delete the specific id or name etc]**

$post->delete();

});

**Withtrashed() is a method to delete the items which are present in the method along with the trash items.**

**Onlytrashed() is a method to delete the items which are there in the trash.**

Route::get(‘/delete’, function()

{

Post::destroy([3,4]); **(insert al the ids that has to be deleted, we can delete multiple methods)**

});

**Basic insert:**

Route::get(‘/basicinsert’,function()

{

$post **= new** post**;…..[if u wanna insert at specific location or file then use $post = post::find[2]; ]**

$post->title=”my new post”;

$post->content = “ ksdjflsughmcBV.VFSLJGH;jfduol”;

$post->save();

});

Route::get(‘/findmore’, function()

{

$post = post::findorfail( it is a predefined method which takes id’s as the parameters,2 );

Return $post;

});

OR

Route::get(‘/findmore’, function()

{

$post=post::where(‘user\_count’,’<’,50)->firstorfail();

});

**To delete a record**

Route::get(‘/deleted’,function()

{

Sdeleted=DB::delete(delete from posts where id=’?’, 2);

});

**Command :** php artisan make:model filename ( it creates a model)

Models will be there in the apps folder

Route::get(‘/findwhere’ , function()

{

$post =post::where(‘id’, 2)->orderBy(‘id’, ‘desc’) ->take(1)->get();

Return post;

});

// **here B should be in capital for** orderBy **clause. And ‘**desc**’ is descending order.**

**To restore a record which we have deleted, inorder to do that first we need to find them.**

Route::get(‘/restore’, function()

{

Post::withtrashed()->where(‘is\_admin’,0)->restore();

});