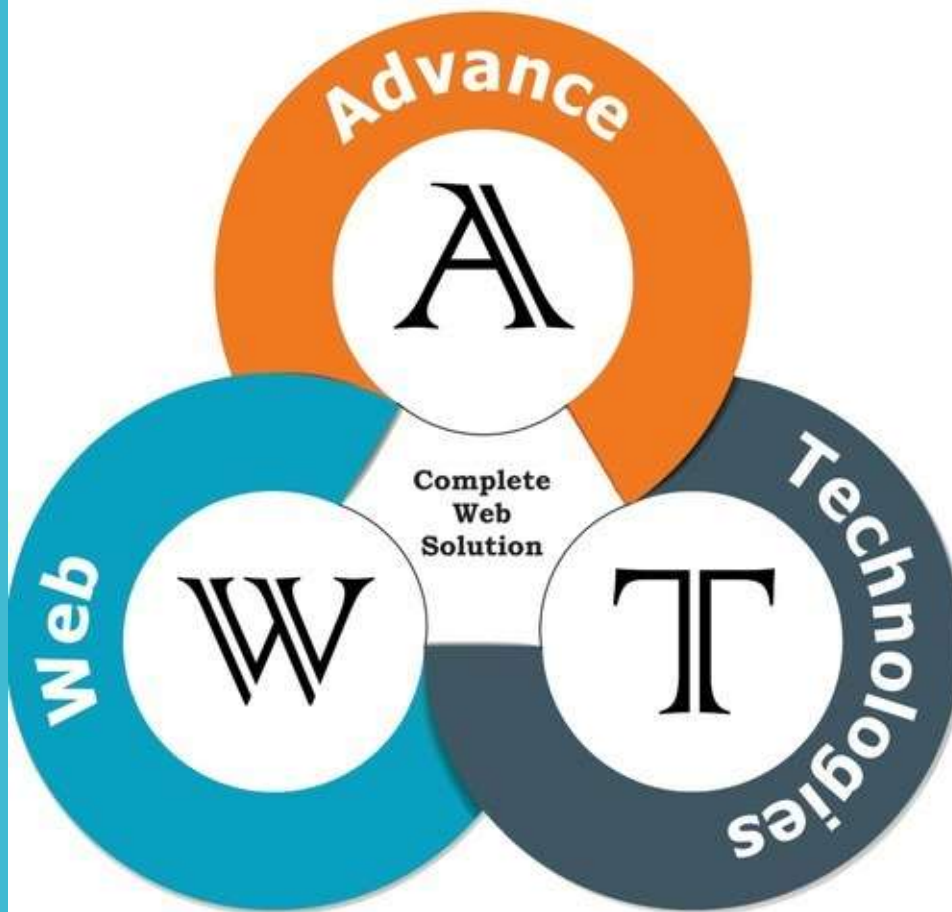Object Oriented PHP

Laravel

Advance PHP

Node.js

Marwadi University

Department of Computer Engineering

Unit 3
PHP MVC Framework:
Laravel

Subject Code: 01IT0701
Advance Web Technology

Prof. Jaydeep K. Ratanpara

Outline

- Introduction to Laravel and MVC
- Environment Setup
- Routes
- Namespaces
- Controllers
- Views
- Request Response
- Redirections
- Forms
- Session
- Cookie
- Database Connectivity and CRUD operations

What is Laravel?

- Laravel is a free open-source PHP web framework, created by Taylor Otwell and intended for the development of web application following the **model-view-controller** (MVC) architectural pattern.

- Laravel most widely used PHP framework for web application development.

- Laravel is an open-source framework that anyone can download and use accordingly for web app development.

- Uses object oriented libraries.

Laravel is a PHP framework that uses the MVC architecture.

- **Framework:** It is the collection of methods, classes, or files that the programmer uses, and they can also extend its functionality by using their code.

- **Architecture:** It is the specific design pattern that the framework follows. Laravel is following the MVC architecture

Laravel Installation

- **Step 1.** Install XAMP

- **Step 2.** Download Composer's installer:

    https://getcomposer.org/download/

- **Step 3.** Place the installer and run it from the added to Path directory – C:\xampp\php\php.exe

- **Step 4:** Now check whether composer is installed or not?
- C:\users\admin>composer

- The installation of the composer is completed. Now we will check whether the composer is installed successfully or not. To check this, open the command prompt and type Composer.


```
Command Prompt                                    _  □  ×

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\admin>Composer_
```



```
Command Prompt                                                        _  □  ×

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\admin>composer

   _____
  / ____/___  ____ ___  ____  ____  _____  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 1.9.0 2019-08-02 20:55:32

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display this help message
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi                     Force ANSI output
      --no-ansi                  Disable ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working directory.
```

Laravel Installation

- **Step 5:** install laravel –
  composer global require laravel/installer

- **Step 6:** Now check whether Laravel installed or not?
  c:\xampp\htdocs>laravel

- **Step 7:** Create a new Project in your Directory.
  c:\xampp\htdocs>laravel new project1

- **Step 8:** c:\xampp\htdocs>cd project1

- **Step 9:** c:\xampp\htdocs\project1>php artisan serve
  (artisan is a command line interface/tool included with laravel)

- **Step 10:** Now load the project on http://127.0.0.1:8000

Composer Installation
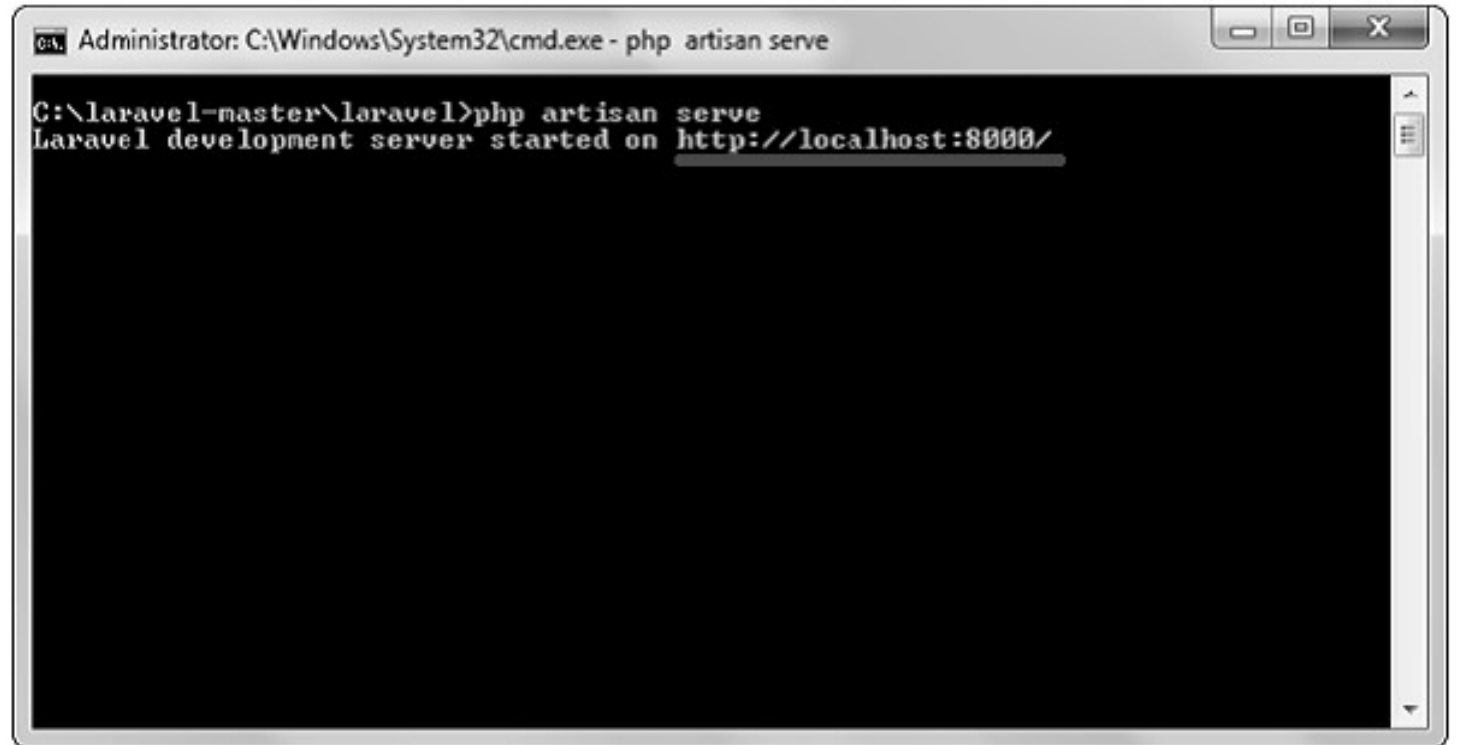
What is Composer?

- Composer is a dependency manager for a PHP programming language that manages the dependencies of PHP software and required libraries.

- Composer runs through the **command line**. The main purpose of the composer is to install the **dependencies** or **libraries** for an application.

- The composer also provides the users to install the PHP applications available on the **Packagist**, where Packagist is the main repository that contains all the available packages.

Click on the link given below to download the Composer: https://getcomposer.org/download/.
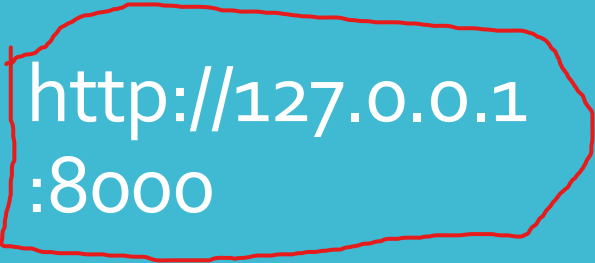
How to run Project:

- The above command will install Laravel in the current directory. Start the Laravel service by executing the following command.

php artisan serve

http://127.0.0.1
:8000

- Copy the URL underlined in gray in the above screenshot and open that URL in the browser. If you see the following screen, it implies Laravel has been installed successfully.

Laravel

DOCS   LARACASTS   NEWS   BLOG   NOVA   FORGE   VAPOR   GITHUB

Directory structure

- After fresh install Laravel have following directory structure.

❑App
- Commands
- Console
- Events
- Handlers
    - Commands
    - Events
- Http
    - Controllers
    - Middleware
    - Requests
- Provider
- Services

❑Bootstrap

❑config

❑Database
- Migrations
- Seeds

❑Public
- Package

❑Resources
- Lang
- Views

❑Storage
- Cache
- Log
- Sessions
- Views
- Work

❑tests

The MVC architecture:

- MVC is an acronym for 'Model View Controller'. It represents architecture developers adopt when building applications.

- MVC is a software architecture, that separates application into three parts: the model, the view, and the controller.

- **Model:**

- The **Model** manages fundamental behaviors and data of the application.

- It can respond to requests for information, respond to instructions to change the state of its information, and even notify observers in event-driven systems when information changes.

- This could be a database or any number of data structures or storage systems. In short, it is the data and data-management of the application.

The MVC architecture:

- **View:**
- The view effectively provides the user interface element of the application. It'll render data from the model into a form that is suitable for the user interface.

- **Controller:**
- The controller receives user input and makes calls to model objects and the view to perform appropriate actions.

- All in all, these three components work together to create the three basic components of MVC.
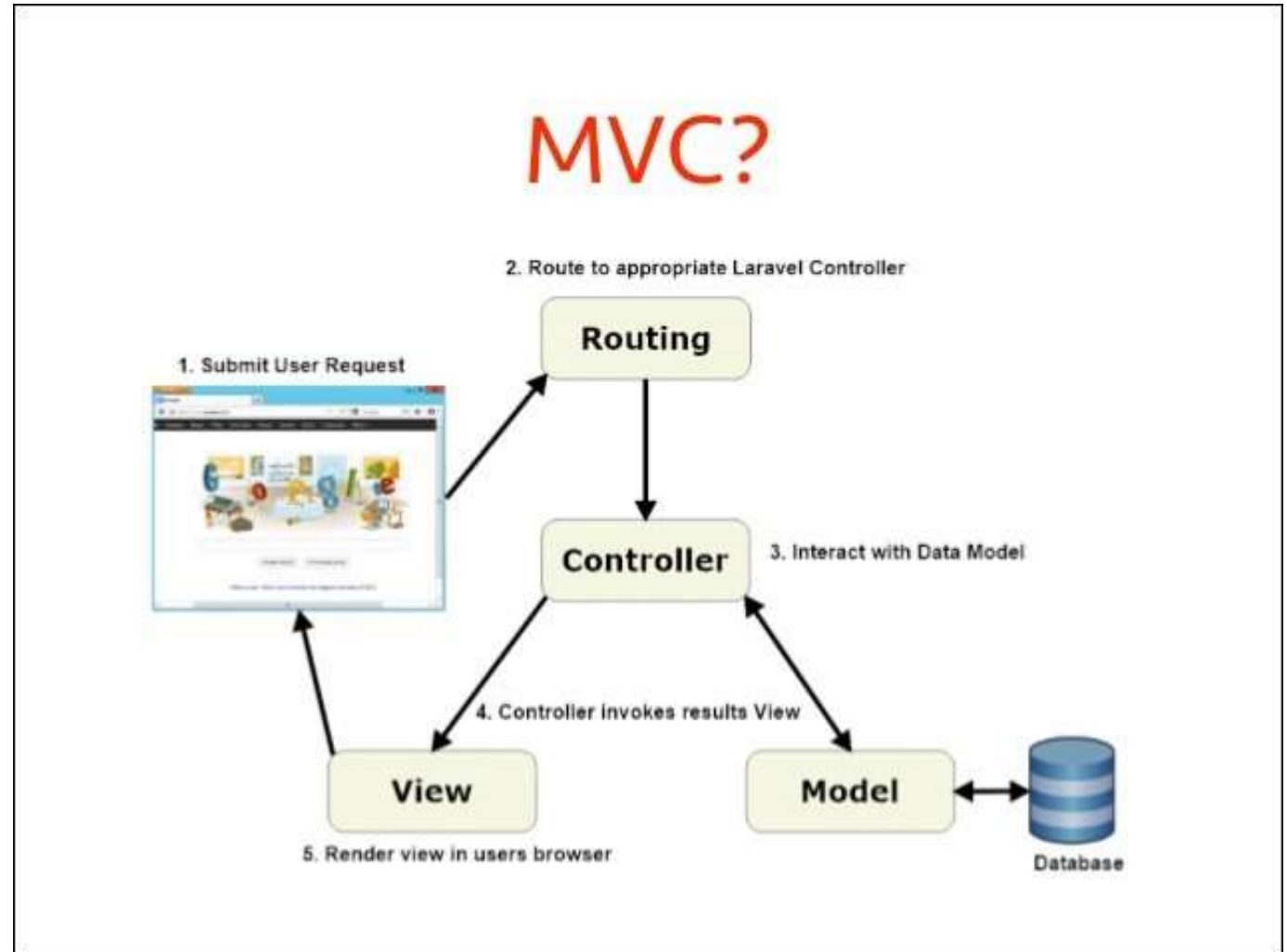
The MVC architecture:
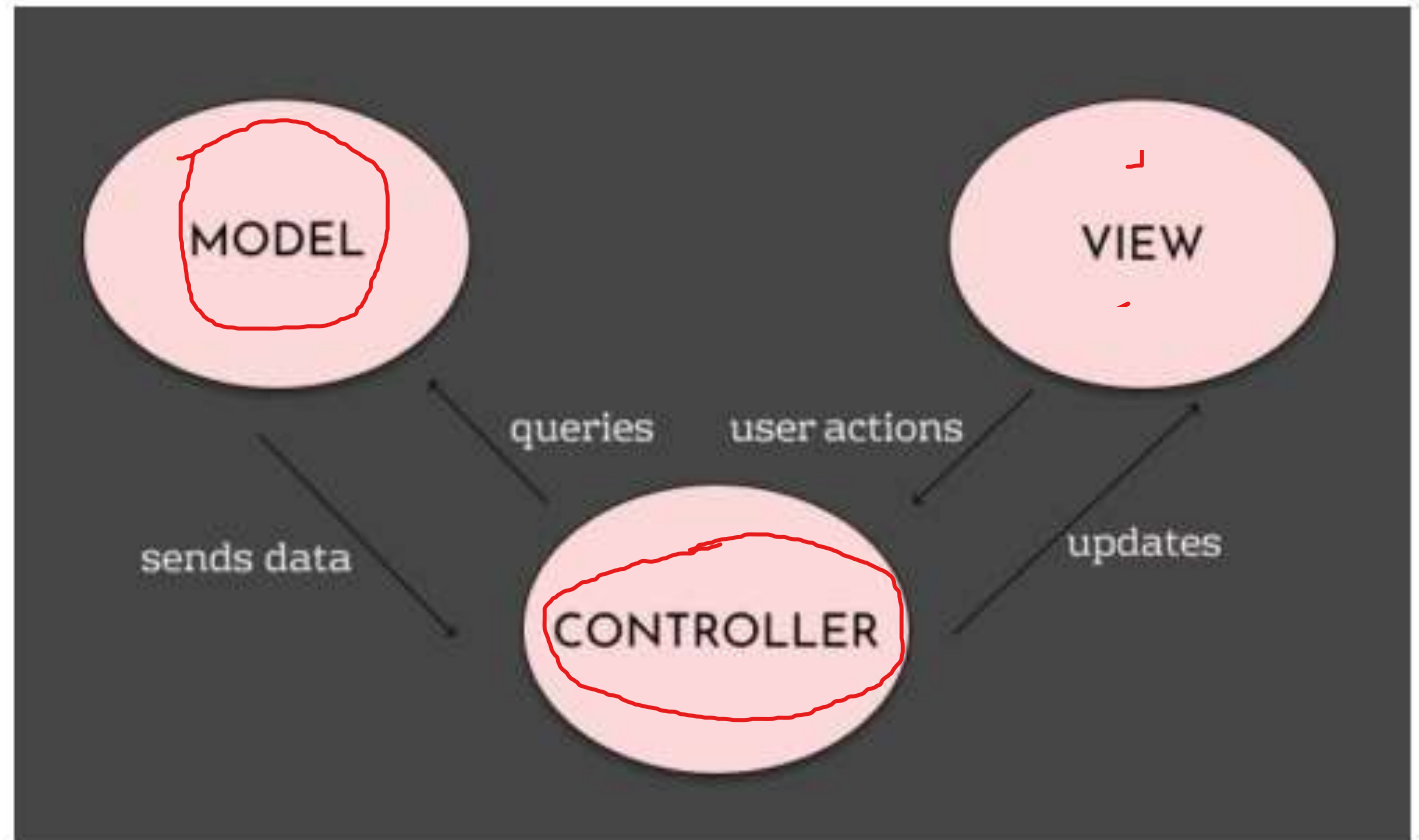
The MVC architecture:

- A Model is a representation of a real-life instance or object in our code base.

- The View represents the interface through which the user interacts with our application.

- When a user takes an action, the Controller handles the action and updates the Model if necessary.

The MVC architecture:

The MVC architecture:

- Let's look at a simple scenario.

- If you go to an e-commerce website, the different pages you see are provided by the View layer.

- When you click on a particular product to view more, the Controller layer processes the user's action. This may involve getting data from a data source using the Model layer.

- The data is then bundled up together and arranged in a View layer and displayed to the user. Rinse and repeat.

Artisan console

- Artisan is the name of the command-line interface included with Laravel. It provides a number of helpful commands for your use while developing your application. It IS driven by the powerful Symfony Console component. To view a list of all available Artisan commands, YOU may use the list command:

- — php artisan list

- Every command also includes a "help" screen which displays and describes the command's available arguments and options. To View a help screen, simply precede the name of the command with help:

- — php artisan help migrate

Where exactly the Display page comes from:

1. Open Project in Editor : / or go to Root Folder
2. Root/resource/views/ **welcome.blade.php**
   In vews folder we need to create views here

1. **If you make any changes on this page you can do it, like:**
   Change the heading Laravel to
   "Laravel Projects".

2. from where welcome page is called:
   - Go to Routes folder and then web.php like :
     - Routes/web.php

You will find welcome page and
that is exactly : welcome.blade.php

```
Route::get('/', function () {
return view('welcome');
});
```

Create a new page:

Go to View:

Right click -> new file :abc.blade.php

According to Laravel file name start with small letter only like : "abc.blade.php"

abc.blade.php

```
<html>
<head>
welcome to ABC View testing
</head>
<body>
<h1>welcome to ABC View
testing</h1>
</body>
</html>
```

Web.php

```
Route::get('/', function () {
return view('welcome');
});
Route::get('/abc', function () {
return view('abc');
});
```

| How to call in URL: | http://127.0.0.1:8000/abc |

Routing:

- What is Routing?
- Making a new page with routing
- Anchor tag with routing
- Pass URL parameters with routing
- Pass data with routing
- Redirection

Routing:

- What is basically Routing?

- When you create a page in Laravel, so basically Laravel don't know what to do???
- So we need to do define routing,

Which means something comes in routing, or we write in routing, then we need to open particular page.

To open page in Routing : go to Route Folder first

Route/web.php

Route::get('/', function () {
return view('welcome');
});

Basically it is called welcome.blade.php

So its simply opens welcome.blade.php in browser

Routing:

- Where we need to defined routing ??

Routing defined after " / "

```
Route::get('/', function () {
return view('welcome');
});
```

Here route is : "abc"

```
Route::get('/abc', function () {
return view('abc');
});
```

Anchor Tag With Routing

- Lets make anchor tag in welcome.blade.php

- welcome.blade.php

- We need to write this code below to main heading:

- We need to put route in href=" ". So abc is route put in

- `<a href="abc">abc</a>`

- web.php

```
Route::get('/abc', function () {
return view('abc');
});
```

Controller:

- What is controller
- How to make controller
- How call controller with routing
- How pass data with controllers

Controller:

- A Controller is that which controls the behavior of a request. It handles the requests coming from the Routes.

- In Laravel, a controller is in the 'app/Http/Controllers' directory. All the controllers, that are to be created, should be in this directory.

- It is central unit of any MVC based application, which actually manages database operations as well as GUIs.

- It is mainly used to fetch the data from the database and display in GUI.

- It is also used for sessions and cookies management

- It can also be used to pass a data using query string and the fetch the data of it.

- (Which means you need to write business logic in controller).

How Controller Works?

- Step 1: Create Controllers

- Step2: make relevant or required functions in controller

- Step3: call this function / call this controller , using Routing(Web.php)

Controller:

create controller

- Where to find controller in leravel?
- **App/Https/Controllers**

- To create a new controller the code is:
- **Php artisan make : controller Users**

- After executing this command successful ,
- Users.php will be created in **App/Https/Controllers**

- Controller name always start with a capital letter because it is a class. It follows OOPs concept.

Controller Continues…

Controller: add index ( ) in controller (Users.php)

- **Users.php – generated by command**

```php
<?php

    namespace
    App\Http\Controllers;

    use Illuminate\Http\Request;

    class Users extends
    Controller
    {
    //
    }
```

Now add index function to Uesrs.php

```php
<?php

    namespace
    App\Http\Controllers;

    use Illuminate\Http\Request;

    class Users extends Controller
    {
        public function index($id)

        {
        echo $id;
        }
    }
```

How to call function index() / controller

- Step1 :we need to define routing
- Syntax:
- **Route::get(" name of route " ," name of Class @name of function");**
- Route::get("test","Users@index");
- Or
- Route::get("/test/","App\Http\Controllers\Users@index");

To run now :

http://127.0.0.1:8000/test

How to pass parameters with controllers

```php
Users.php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{
public function index($id)
{
echo $id;
}
}
```

```php
Web.php

Route::get("test/{id}","Users@index");
Or
Route::get("/test/{id}","App\Http\Controllers\Users@index");
```

To run this :
http://127.0.0.1:8000/test/101

Views:

- What is view?
- How to make view?
- Call a view with routing
- Call a view with Controller
- Pass data while calling view

What is View?

- View mainly deals with GUI part of Laravel application
- It contains all the HTML parts to display the data in to browser.

How to make a view?

- Go to Your Folder, Search for RESOURCES

- Resources/Views/Welcome.Blade.Php → Default file

- To create a new view:

- Right click on views & select new file (named it anything u like: as following)

- Resources/Views/**mobiles.Blade.Php**

mobiles.Blade.Php

```
<html>
<head>
welcome to Creating Views
</head>
<body>
<h1>Mobile.blade.php</h1>
</body>
</html>
```

How to call View with routing?

To open a view page using routing goto web.php (Routing)

Syntax:

Route::view('/route name', 'View name/ file name');

Example:

Route::view('/iphone', 'Mobiles');

To run this :
http://127.0.0.1:8000/iphone

How to call View with Controller?

- We already created controller named Users.php so we will use same file, we have also created INDEX() in that file .

- Now To call View with Controller do as follows:

- Step1: Go to Controller:

```php
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{
public function index($id)
{
return view('Mobiles');
//Mobiles is the view name only
//echo $id;
}
}
```

To run this :
http://127.0.0.1:8000/test/101

How to pass data While calling a view?

We already created controller named **Users.php** so we will use same file , we have also created **INDEX()** in that file .

Now To call View with Controller do as follows:

Step1: Go to Controller: Users.php

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{
public function index($id)
{
return view('Mobiles', ['name'=>"Alex"]);
//echo $id;
}
}
```

To run this :
http://127.0.0.1:8000/test/101

Continues…

How to pass data While calling a view?

- We already created view name **mobiles.blade.php** so we will use same file..

- Now To call View with Controller do as follows:

- Step2: Go to Views: Mobiles.blade.php

```html
<html>
<head>
welcome to Creating Views
</head>
<body>
<h1>Mobile.blade.php</h1>
<h2>
{{$name}}
// $name is the variable that is passed in controller and fetch in view.
// {{}}: Shortcut to define Php which also known as Blade Templating.
</h2>
</body>
</html>
```

To run this :
http://127.0.0.1:8000/test/101

Https Request & Response

- It is mainly used to send any type of request from one page to another page.

- When we need to fetch any data from data server, so basically request sent through HTTPS only.

- It is also used for knowing the URL of Site from where the data is being sent .

- It can also be used to know through which method (GET/POST) the data has been sent .

- It is also used to know the data of the query string in the form of array.

Https Request & Response :

To know all types of Request

Examples:1

- To know all the types of request is being sent go to **App/http/controller/users.php**

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{
    public function index(Request $req)
    {
        var_dump($req);
    }
}
```

Now perform below to check what we have ??
- **var_dump($req->input()); ---- > it will display Blank Array**

Now type if url as enter the data:
- http://127.0.0.1:8000/user?data=100

You can add more data to above url like &id=1000811.

Https Request & Response :

To know URL:

Examples:1

• To know URL : **App/http/controller/users.php**

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{
    public function index(Request $req)
    {
        echo $req->url();
    }
}
```

i/p in url :http://127.0.0.1:8000/user?data=100&id=100000

o/p: http://127.0.0.1:8000/user, now we get upto /user only what about rest?

Continues…

Https Request & Response :

To know URL:

Examples:1

Now To get Full URL :

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{
        public function index(Request $req)
        {
                echo $req->fullUrl();
        }
}
```

Https Request & Response :

To know PATH:

Examples:1

To know Path : **App/http/controller/users.php**

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{
        public function index(Request $req)
        {
                echo $req->path();
        }
}
```

Input: http://127.0.0.1:8000/user?data=100&id=100000
Output: user
Which means from this path our data is coming.

Https Request & Response :

To know Data come from: GET /POST

Examples:1

To check : data comes from get or Post?

App/http/controller/users.php

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;

class Users extends Controller
{
    public function index(Request $req)
    {
        echo $req->method();
    }
}
```

O/P : GET

Https Request & Response :

GET /POSTused : to display something

Examples:1

Now if Any condition we are using like , if I m using GET or POST , I wanna do something or if get then display something / if post then display something: **App/http/controller/users.php**

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{

    public function index(Request $req)
    {

        //echo $req->method();
        if($req->isMethod('GET'))
        {

            echo "Data coming through GET";
        }
        else
        {

            echo "Else";
        }
    }
}
```

OutPut: if

Continues... but condition change in next..

Https Request & Response :

GET /POSTused : to display something

Examples:2

Now if Any condition we are using like , if I m using GET or POST , I wanna do something or if get then display something / if post then display something : **App/http/controller/users.php**

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Users extends Controller
{
    public function index(Request $req)
    {
        //echo $req->method();
        if($req->isMethod('POST'))
        {
            echo "if";
        }
        else
        {
            echo "else";
        }
    }
}
```

Output: else

To know Query parameters : **App/http/controller/users.php**

Https Request & Response :

how to Get Query Parameters

Examples:1

```php
<?php

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

class Users extends Controller
{
    public function index(Request $req)
    {
        var_dump($req->query());
    }
}
```

i/p in url: http://127.0.0.1:8000/user?data=100&id=100000

Form Processing with Validation:

Formcontrol.php : Controller File

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class Formcontrol extends Controller
{
    //
public function submit(Request $req)
    {
$req->validate([
"name"=>"required",
"pass"=>"min:2 | max:8",
"phone"=>"numeric | min:10",
"email"=>"email"
]);
//Print Data from POST Method
            return $req->input();
    }
}
```

Continues…

Form Processing with Validation:

Form.Blade.php : Regular HTML Form File

```html
<html>
<head>
<title>Forms</title>
</head>
<body>
<div>
<ul>
@foreach($errors->all() as $e)
<li>{{$e}}</li>
@endforeach
</ul>
</div>
  <form action="formdata" method="POST">
  @csrf
  Name: <input type="text" name ="name">
  @error('name')
  <div>{{$message}}</div>
  @enderror
  <br>
  Password: <input type="password" name="pass">
  @error('pass')
  <div>{{$message}}</div>
  @enderror
  <br>
  Mobile<input type="text" name="phone">
  @error('phone')
  <div>{{$message}}</div>
  @enderror
  <br>
  Email<input type="text" name="email">
  @error('email')
  <div>{{$message}}</div>
  @enderror
  <br>
  <input type="submit" value="OK">
  </form>
</body>
</html>
```

Continues…

Form Processing with Validation:

Web.PHP : Route File

- Route::view("/regform",'form');

- Route::post("/formdata",'Formcontrol@submit');

CSRF Protection:

Introduction

- Laravel makes it easy to protect your application from cross-site request forgery(CSRF) attacks. Cross-site request forgeries are a type of malicious exploit whereby unauthorized commands are performed on behalf of an authenticated user.

- Laravel automatically generates a CSRF "token" for each active user session managed by the application. This token is used to verify that the authenticated user is the one actually making the requests to the application.

- Anytime you define an HTML form in your application, you should include a hidden CSRF token field in the form so that the CSRF protection middleware can validate the request. You may use the @csrf Blade directive to generate the token field:

CSRF Protection:

Introduction

- CSRF refers to Cross Site Forgery attacks on web applications. CSRF attacks are the unauthorized activities which the authenticated users of the system perform. As such, many web applications are prone to these attacks.

- Laravel offers CSRF protection in the following way –

- Laravel includes an in built CSRF plug-in, that generates tokens for each active user session. These tokens verify that the operations or requests are sent by the concerned authenticated user.

How to Define?

CSRF

```
<form method="POST" action="/profile">

@csrf ...

</form>
```

Laravel & Database

- First create a table in database

- With any field that you required like: id, name , address, email.

- Now need to configure and check database file of Laravel, for that go to **config/database.php**

- In this file you will find different configurations for different type of databases, supported by Laravel.

- For. Ex. SQLite , MySql, PGSql, SQLSRV(sqlserve),

- you need to configure you database as per you requirement

- We do not change this file, but all the configurations related to database are made in .env file of Laravel.

- Open .env file & go to DB_connection=mysql and change the :
  - Host,
  - Port,
  - Database
  - Username & password as per your requirement .
  - DB_SOCKET=/Applications/MAMP/tmp/mysql/mysql.sock (also add in MAC)

Laravel & Database

- After the Configuration restart the Laravel server.

- Create a Controller for performing database operations named dbcontroller then go to dbcontroller.php file.

- Now we need to call the Controller to perform database operations , so we create one route in web.php

Fetch the data

- Now to fetch the data from the database, we required DB Class or namespace, now Syntax to import DB Class is as follows:

- **use Illuminate\Support\Facades\DB;**

- Using above name space we can perform any type of database operations

- For. Ex. To Fetch the data from the table. the syntax is as follows:

Fetch Data /

Perform Selection operations

- Step 1: Create controller **dbcontrol**

- **Type Command:** Php artisan make:controller dbcontroller

- So controller file is at : **Controllers/dbcontrol.php**

Continues…

dbcontrol.php

```php
public function dbselect()
{

// Fetch the Data from the table
// Array Format
$a = DB::select('select * from user');
print_r($a);
// Json Format
return DB::select('select * from user');

// Other Way of Fetching the Data with Where Clause
$b = DB::table('user')
->where('Name','Munindra')
->get();
print_r($b);

// Find the Data only on basis of ID
$c = DB::table('user')->find(5);
print_r($c);

//Count the Records
$d = DB::table('user')->count();
print_r($d);
}
```

Continues…

Web.php

Now after making controller and function we need to define Route: Which is as Follows

```
Route::get("dbsel",'dbcontrol@dbselect');
```

Inserting Data in Database :

dbcontrol.php

&

web.php

```php
public function dbinsert()
{
// Insert New Records
$i = DB::table('user')
->insert([
'Name'=>'Mahendra',
'Email'=>'M@m.com',
'Address'=>'Chennai'
]);
print_r($i);
}
```

Now after making function we need to define Route: Which is as Follows

```php
Route::get("dbins",'Dbcontrol@dbinsert');
```

Updating Data in Database :

dbcontrol.php

&

web.php

```php
public function dbupdate()
{
    // Update Any Records
    $u = DB::table('user')
    ->where('Id','6')
    ->update([
    'Name'=>'Mahendra Reddy',
    'Email'=>'Mahendra@mu.ac.in'
    ]);
    print_r($u);
}
```

Now after making function we need to define Route: Which is as Follows

```php
Route::get("dbupt",'Dbcontrol@dbupdate');
```

Deleting Data in Database :

dbcontrol.php

&

web.php

```php
public function dbdelete()
{

    // Delete Any Records
    $del = DB::table('user')
    ->where('Id','6')
    ->delete();
    print_r($del);
}
}
```

Access Database through Model

- Step 1: Create Model First :

- Php artisan make:model dbmodel

Now go to dbmodel.php file In App Folder and update the code as follows:

Step 2 : Create a Function in controller:

Step 3: To Access the dbmodel (Model), Create route in web.php

Note: Model can be used to only fetch the data

dbmodel.php

```php
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class dbmodel extends Model
{
 //
    protected $table = 'user'; // (Any Table name to show in o/p)
}
```

Dbcontrol.php

public function dbmodel()

{

return dbmodel::all();

(It will return all the data of the Table in form of Json . for the table name passed in the model file)

}

Web.php

Route::get("dbmodel",'Dbcontrol@dbmodel');

Laravel:

Session

Step 1: First create signin.blade.php in views

Step 2: Create Route for it in Web.php

Step 3: Create a controller login.php

Step 4: Create profile.blade.php in view

Step 5-1: Create a Route for Logout in Web.php

Step 5-2: Create a Route in Web.php for Profile.Blade.php

STEP 1:

Signin.blade.php

```
<form action="signon" method="post">


Username:<input type="text" name="user"> <br>


Password:<input type="password" name="pwd"><br>


@csrf


<button type="submit">Login</button>


</form>
```

STEP 2:

Web.php

Route::view("signin",'signin');

Route::post("signon",'login@submit'); // Create a Route to get Post data via controller

STEP 3:

Login.php (Controller)

```php
public function submit(Request $req)


  {


     $req->session()->put('data',$req->input());  // "data" is Session Key and $req->input() is Value to be stored in Session

     return redirect('profile');  // If session is Set, then redirect to Profile.Blade.php

  }
```

STEP 4 : Profile.blade. Php (View)

```
<h1>Profile Page</h1>


<h3>Welcome {{session('data')['user']}}</h3>  // Displays the
Username sent via POST and Session


<br>


<a href="logout">Logout</a>   // Clicking Logout removes session
and redirects to signin page
```

STEP:
5-1
&
5-2
Web.php

Route::view("profile",'profile');

- **If session is not created, then user should not be able to access the profile page directly withoutlogin. So to manage this, go to web.php and put the following code:**

```
Route::get('profile', function () {
  if(!session()->has('data'))
  {
    return redirect('signin');
  }
  else
  {
  return view('profile');
  }
});
```

//LOGOUT

```
Route::get('/logout', function () {
session()->forget('data');
return redirect('signin');
});
```

Laravel:

Cookie

- Step 1: First Create a Controller for Cookies. -> cookiecontrol

- Step 2: Now Create 2 Functions to Set and Get Cookies in the Class (Controller)

- Step 3: Create Routes in Web.php for each controller methods of cookies

Step 1: cookiecontrol.php

Add following Namespaces in header section of Controller

use Illuminate\Http\Response;

use App\Http\Requests;

use App\Http\Controller\Controllers;

Step 2: create Functions in cookiecontrol.php (Controller)

```php
public function setcookie(Request $req)
{
// How Long Cookie Should Remain in minutes
$minutes = 5;
// Response message of the cookie
$response = new Response('Hello Students');
//set cookie and its value
$response->withCookie(cookie('name','Marwadi University',$minutes));
$response->withCookie(cookie('address','Rajkot',$minutes));
// set cookie
return $response;
}
```

Step 2: create Functions in cookiecontrol.php (Controller)— Continue..

```php
public function getcookie(Request $req)
{
// Retrieve value of name Cookie
$result = $req->cookie('name');
// Retrieve value of address Cookie
$result1 = $req->cookie('address');
// Print values of Cookies
echo $result;
echo $result1;
}
```

Step 3: Creating Routes: web.php

```php
// Route for Cookies
// Route for Setting Cookie Value
Route::get('setcookie','cookiecontrol@setcookie');
// Route for Getting Cookie Value
Route::get('getcookie','cookiecontrol@getcookie');
```

THANK YOU