Laravel

1. Global installation

- composer global require laravel/installer

-laravel new example-app

2.per project installation

-composer create-project laravel/laravel example-app

3.project run

-php artisan serve

4.project run stop

-ctr+c

5.Laravel Folder & File Structure

-Model Folder -> Database/sql queries Handling Files (app->models)

-controller Folder -> Business Logics Files (app->http->controller)

-view Folder ->Html files (resources->views)

-Routing Folder -> URl defining Files

-Asserts Folder(public Folder) -> images/fonts/music/videos/css/javascript Files

6.Route---->URl

-webpage URL

7.Search any file vscode

-ctr+p

8.All command show

-php artisan

9.controller create

-php artisan make:controller (Name)

10.table create

-php artisan make:migration create_tableName_table

11.Table name create of sql

-php artisan migrate

12.last update table k delete kore dibe

-php artisan migrate:rollback

13.database empty

-php artisan migrate:reset

14.database refresh

-php artisan migrate:refresh

15.model and table create same time

-php artisan make:model (Name) -m

1.Routing

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

3.single line

```
Route::view('/post','post');
```

Laravel Routing Parameters & Constraints

```
Route::get('/post/{id}', function (string $value) {
    return "<h1>Post Id :". $value ."</h1>";
});
```

optional perameter

```
Route::get('/post/{id?}', function (string $value = null) {
    if($value){
        return "<h1>Post Id :". $value ."</h1>";
    }else{
      return "<h1> No id found </h1>";
    }
```

```
});
```

multi perameter passing

```
Route::get('/post/{id?}/comment/{commentid?}', function (string $value = null,string $comment =null) {
   if($value){
       return "<h1>Post Id :". $value ."</h1> <h2>". $comment ."</h2>";
   }else{
     return "<h1> No id found </h1>";
   }

});
```

laravel Constraints

```
Route::get('/post/{id?}', function (string $value = null) {
if($value){
return "<h1>Post Id :". $value ."</h1>";
}else{
return "<h1> No id found </h1>";
}

})->whereNumber('id'); //id== number hobe
```

```
Route::get('/post/{id?}', function (string $value = null) {
   if($value){
       return "<h1>Post Id :". $value ."</h1>";
   }else{
     return "<h1> No id found </h1>";
   }

})->whereIn('id',['moive','song','painting']); // id value ei tin tar modder value dile kaj korbe noy perameter kaj korbe na

Route::get('/post/{id?}', function (string $value = null) {
   if($value){
       return "<h1>Post Id :". $value ."</h1>";
   }else{
     return "<h1> No id found </h1>";
   }

})->where('id','[0-9]+'); // id value regular expersion  er modder dite hobe

Route::get('/post/{id?}', function (string $value = null) {
   if($value){
       return "<h1>Post Id :". $value ."</h1>";
   }else{
     return "<h1> No id found </h1>";
   }
```

```
})->where('id','[a-zA-Z]+'); // id value Aphavetic regular expersion  er modder dite hobe



Route::get('/post/{id?}/comment/{commentid?}', function (string $value = null,string $comment =null) {
   if($value){
       return "<h1>Post Id :". $value ."</h1> <h2>". $comment ."</h2>";
   }else{
     return "<h1> No id found </h1>";
   }

})->where('id','[a-zA-Z]+')->whereAlpha('commentid');
```

Laravel Named Route

```
Route::get('/post/about-us', function () {
   return view('post');
})->name('about');

// <a href='{{route('about)}}'>Post</a>  jokhon post click korbo tokhon about-us route korbe
```

Redirect route

Route::redirect('/about','/test'); // user about route click korle autometic test route niye jabe

Route::redirect('/about','/test',301);

// 301->permanent redirect code

Routes Group

// url-> /page/....

```
Route::prefix('page')->group(function (){
Route::get('/about-us', function () {
    return view('post');
});
Route::get('/about', function () {
    return view('post');
});
Route::get('/view', function () {
    return view('post');
});
});
```

Not found page Route

```
Route::fallback(function(){
   return "<h1>Page Not found</h1>";
})
```

Blade Template Main Directives

1.@include -> jeta korbo sei file add hobe

2.@section

3.@extend

4.@yield

include

@include('pages.header')

@include('pages.header',['name'=>'aminur'])

@include('pages.footer')

jodi na thake page taw error dekhabe na include if

@includeIf('pages.footer')

@@includeWhen(condition, 'view.name', ['some' => 'data']) condition true hole view page show korbe

Layouts Using Template Inheritance

Defining a Layout

```
<!-- resources/views/layouts/app.blade.php -->

<html>
```

```
    <head>
        <title>App Name - @yield('title')</title>
    </head>
    <body>
        @section('sidebar')
            This is the master sidebar.
        @show

        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

Extent layout

```
<!-- resources/views/child.blade.php -->

@extends('layouts.app')

@section('title', 'Page Title')

@section('sidebar')
    @parent
```

```
    <p>This is appended to the master sidebar.</p>
@endsection

@section('content')
    <p>This is my body content.</p>
@endsection
```

Php convert json

```
@php
    $user = "Aminur";
    $fruits = ['Apple',"Orange","Banana","Graps"];
@endphp

<script>
    var data = @json($fruits);
    data.forEach(element => {
        console.log(element)
    });
</script>
```

Defining a Layout

```
<!-- resources/views/layouts/app.blade.php -->

<html>
```

```
    <head>
        <title>App Name - @yield('title')</title>
    </head>
    <body>
        @section('sidebar')
            This is the master sidebar.
        @show

        <div class="container">
            @yield('content')
        </div>
____@stack('javscript') //javascript addd
    </body>
</html>
```

Extent layout

```
<!-- resources/views/child.blade.php -->

@extends('layouts.app')

@section('title', 'Page Title')

@section('sidebar')
```

```
    @parent

    <p>This is appended to the master sidebar.</p>
@endsection

@section('content')
    <p>This is my body content.</p>
@endsection
```

@push(‘javascript’)

Jslink add korbo

@endpush

Laravel Pass Data Route to View

Route define

```
Route::get('/view', function () {
    $name = 'Yahoo Baba';
    return view('post',['user'=>$name]);
});

Route::get('/view', function () {
    $name = 'Yahoo Baba';
    return view('post',['user'=>$name,'city'=>'delhi']);
});
```

View file excute

<h1>{{$user}}</h1>

Controller Tutorial

1.controller file create php artisan make:controller(Name)

2.controller classs create

3.route create

Control page

```
class pageController extends Controller
{
    public function showUser(){
        return view('welcome');
    }

}
```

Route page

```
use Appl\Http\Controllers\pageController;
Route::get('/',[pageController::class,'showUser']);
```

```
class pageController extends Controller
{
    public function showUser(string $id){
        return view('welcome',['id'=>$id]);
```

```
    }
}
```

```
Route::get('/user/{id}',[pageController::class,'showUser']);
```

Group controller in route

```
Route::Controller(pagecontroller::class)->group(function(){
        Route::get('/','showUser');
        Route::get('/user/{id}','showUser');
});
```

Database Migration

Step:1 create database

-.env file database setup korte hobe

Step:2 create database migration(create tables)

1.php artisan make:migration create_tableName_table

2.add to table row name

```
Schema::create('students', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('email');
        $table->timestamps();
    });
```

3.php artisan migrate

Step:3 seeding(insert initial data table)

Step 4:create model

***update column**

1.command

2.add column

3.php artisan migrate

Step to work in Seeder

1. Php artisan make:model databasetable
2. Php artisan make:seeder ModelNameSeeder

3.use App/Models/student;

Class StudentSeeder extends Seeder{

Public function run():void{

Student::create([

'name'=>'Yahoo Baba',

'email=>'noor@gmail.com'

]);

}

}

4.Seeders/DatabaseSeeder.php

$this->call([

StudentSeeder::class

]);

5.php artisan db:seed

Note:database name somoy s dite hobe r baki kkhete s dite hob na

Mutiple data add

```
public function run(): void
  {
     $students =collect(
        [
        [
           'name'=>'a',
           'email'=>'a@gmail.com'
        ],
        [
           'name'=>'b',
           'email'=>'b@gmail.com'
        ],
        [
           'name'=>'c',
           'email'=>'c@gmail.com'
        ],
        [
           'name'=>'d',
           'email'=>'d@gmail.com'
        ],
        ]
     );
```

```
        $students->each(function($student){
          student::insert($student);
        });


    }
}
```

Json file theke data add kora

Json file

```
[
    {
      "name":"aminur",
      "email":"aminur@gmail.com"
    },
    {
      "name":"a",
      "email":"a@gmail.com"
    },
    {
      "name":"b",
      "email":"b@gmail.com"
    },
```

```
    {
      "name":"c",
      "email":"c@gmail.com"
    },
    {
      "name":"d",
      "email":"d@gmail.com"
    }
```

]

Seeder file

```
public function run(): void
  {


    $json = File::get(path:'database/json/students.json');
    $students = collect(json_decode($json));
    $students->each(function($student){
      student::create(
        [
```

```
            'name'=>$student->name,
            'email'=>$student->email

           ]
           );
    });


}
```

Again data add time this command

Php artisan migrate:fresh –seed

Fake data implementation

```
public function run(): void
  {
     student::create([
        'name'=>fake()->name(),
        'email'=>fake()->unique()->email()
     ]);
  }
```

Multiple fake data implementation

```
public function run(): void
  {
```

```
    for($i=1;$i<=10;$i++){
        student::create([
       'name'=>fake()->name(),
       'email'=>fake()->unique()->email()
    ]);
    }
  }
```

Steps to work in Factory (fake data add ,can not using loop only using function then create a multiple fake data)

1.php artisan make:model ModelName

2.php artisan make:factory ModelNameFactory

3.class StudentFactory extends Factory{

Public function definition():array{

Return[

‘name’=>fake()->name(),

‘email’=>fake()->email()

];

}

}

4.Seeders/DatabaseSeeder.php

Use App\Models\student;

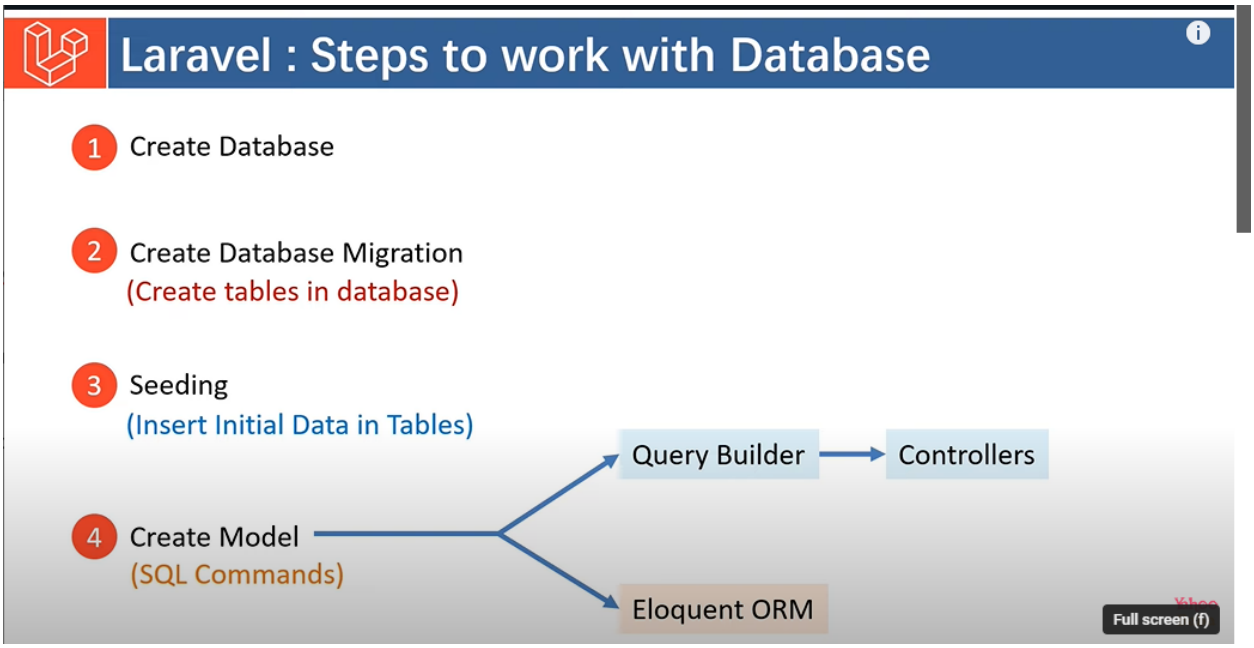
Student::factory()->count(5)->create();

Note:count holo kot gulo data create korbe seta

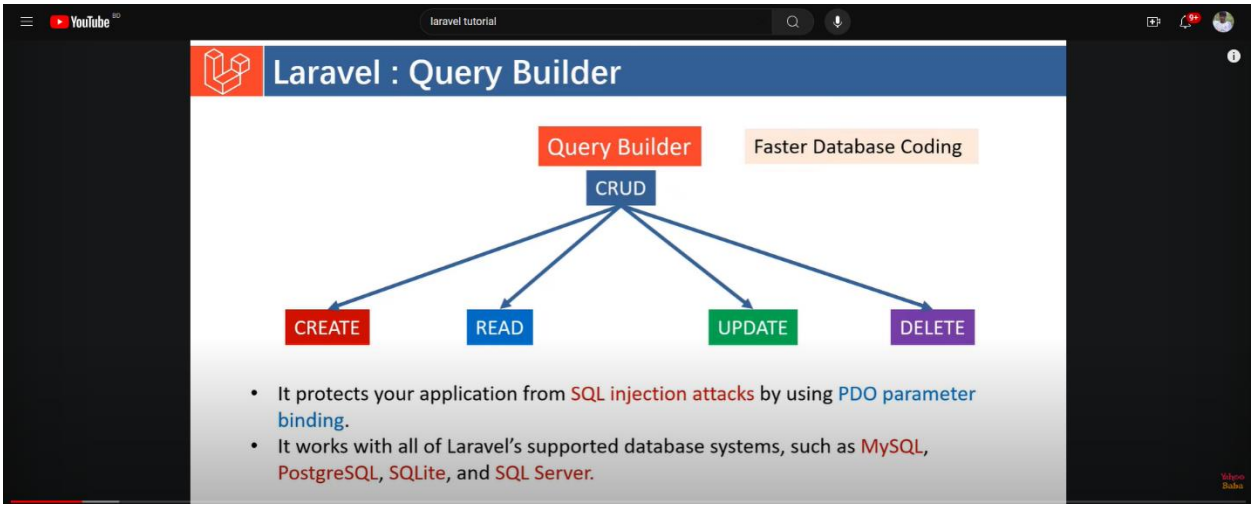
5.php artisan db:seed

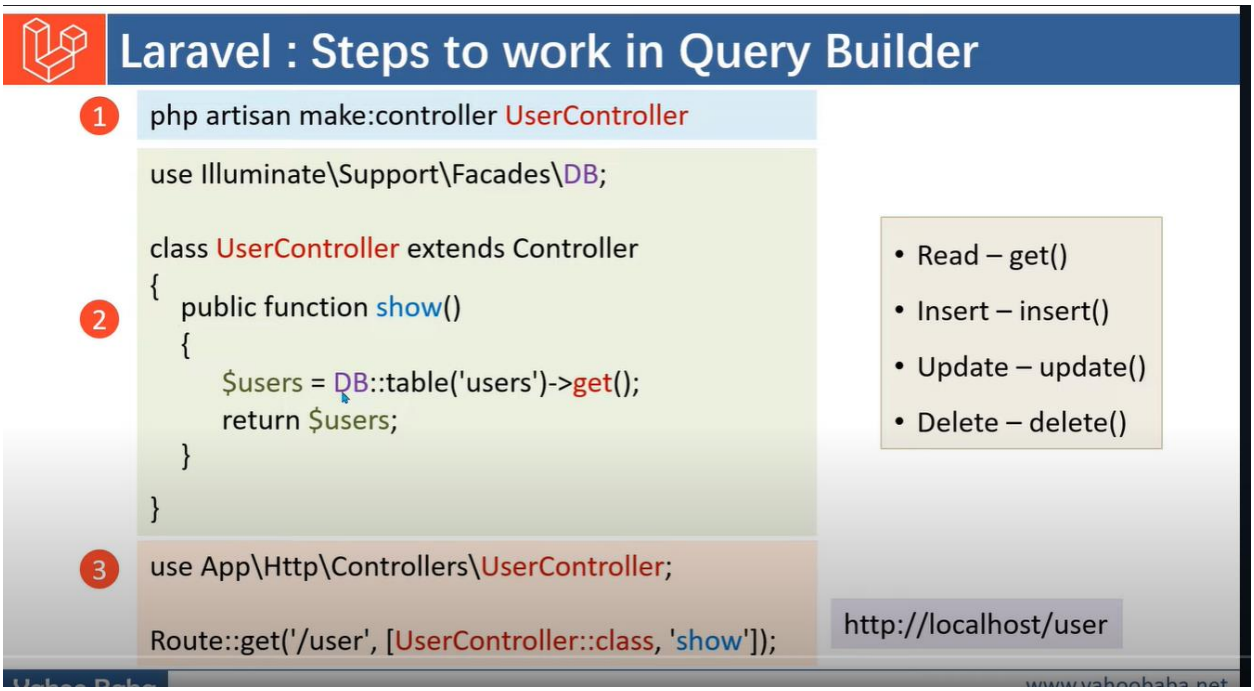
Fake data documentation

https://fakerphp.org/

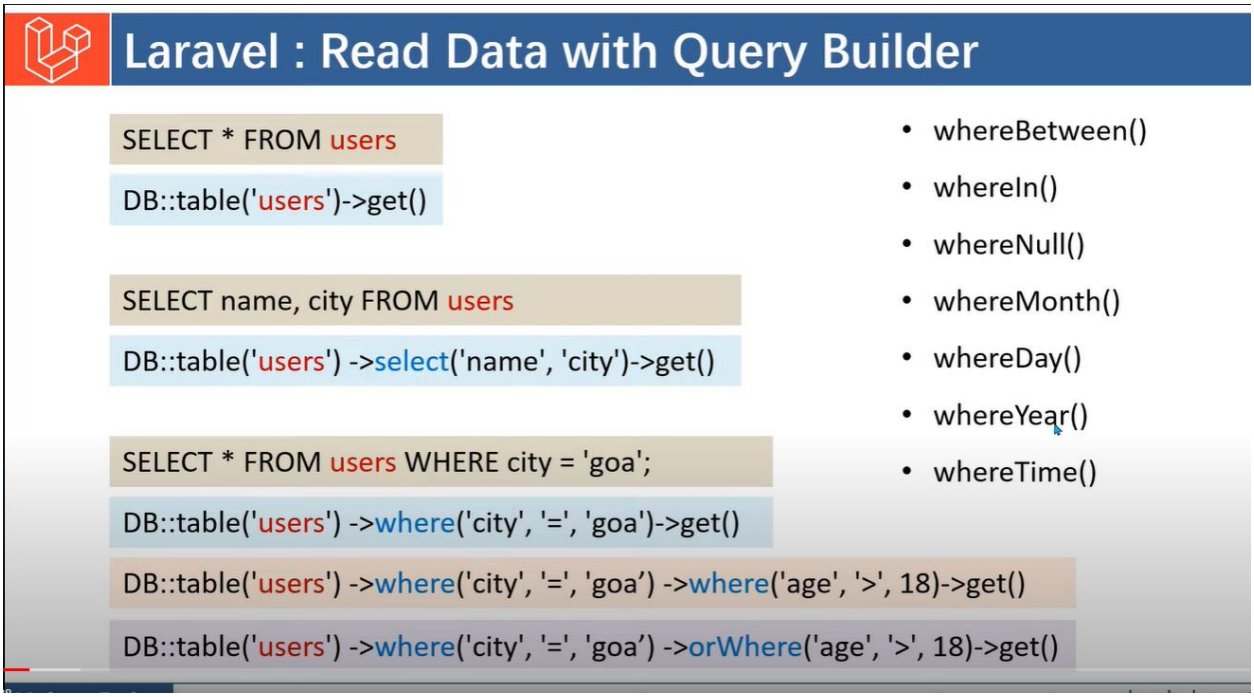


Laravel Query Builder





Read data with query Builder



Crontroller create

-php artisan make:controller UserController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class UserController extends Controller
{
    public function showUsers(){
        $users =DB::table('users')->get();
        return $users;
    }
}
```

Route

```php
<?php

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

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

Route::get('/',[UserController::class,'showUsers']);
```

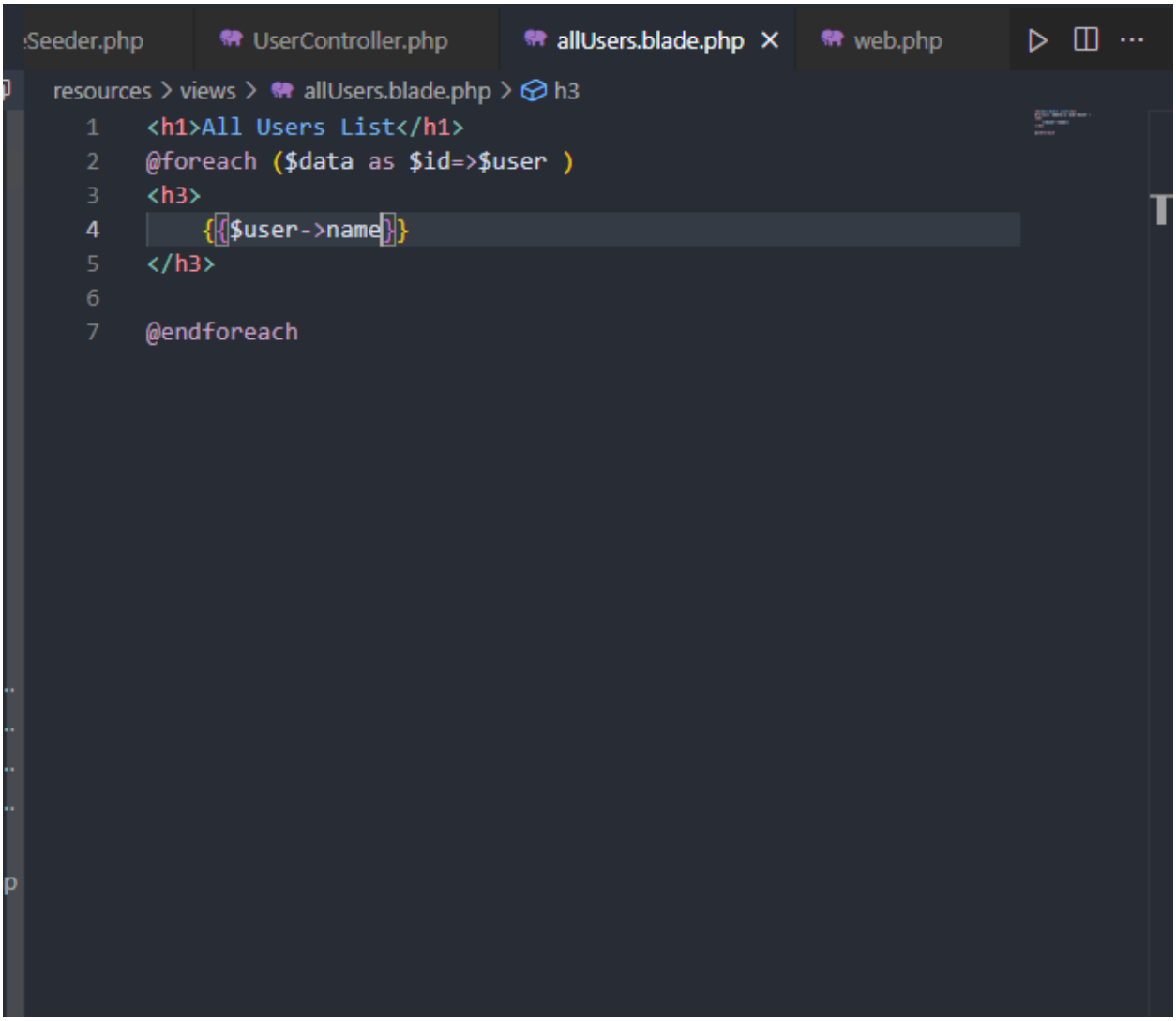
2.controller

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class UserController extends Controller
{
    public function showUsers(){
        $users =DB::table('users')->get();
        // return $users;
        return view ('allUsers',['data'=>$users]);
    }
}
```

Route

```php
Route::get('/',[UserController::class,'showUsers']);
```

View


Seeder.php
UserController.php
allUsers.blade.php
web.php
resources > views > allUsers.blade.php > h3
1 <h1>All Users List</h1>
2 @foreach ($data as $id=>$user)
3 <h3>
4 {{$user->name}}
5 </h3>
6
7 @endforeach

3.

UserSeeder.php | DatabaseSeeder.php | UserController.php | web.php | user.json

app > Http > Controllers > UserController.php > UserController > singleUser

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class UserController extends Controller
{
    public function showUsers(){
        $users =DB::table('users')->get();
        // return $users;
        return view ('allUsers',['data'=>$users]);
    }
    public function singleUser(string $id){
        $users =DB::table('users')->where('id',$id)->get();
        // return $users;
        return view ('allUsers',['data'=>$users]);
    }
}
```

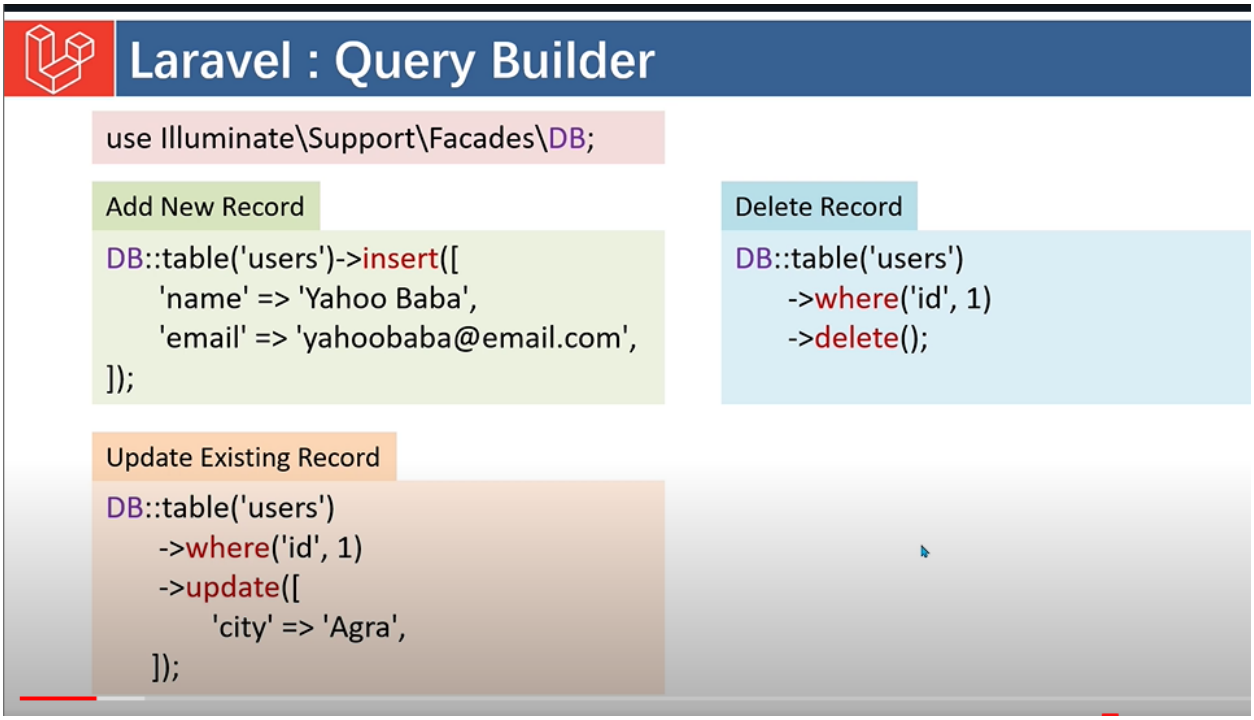
```php
Route::get('/',[UserController::class,'showUsers']);
Route::get('/user/{id}',[UserController::class,'singleUser'])->name('view.user');
```

//button

```
  <td><a href="{{ route('view.user', $user->id) }}" class="btn
/tr>
```

//column select

```php
public function showuser(){
    $user=DB::table('users')->select('name','email')->get();
    return $user;
}
```




```php
public function addUser(){
    $user = DB::table('users')->insert([
        'name'=>'Aminur',
        'email'=>'noor@gmail.com',
        'password'=>'12345678'

    ]);
    if($user){
        echo "<h1>Data Successfully Added.</h1>";
    }
}
```

```php
Route::get('/add',[UserController::class,'addUser']);
```

Multiple value insert

```php
public function addUser(){
    $user = DB::table('users')->insert([
        [
        'name'=>'d',
        'email'=>'d@gmail.com',
        'password'=>'12345678'

        ],
        [
        'name'=>'a',
        'email'=>'a@gmail.com',
        'password'=>'12345678'

        ],
        [
        'name'=>'b',
        'email'=>'b@gmail.com',
        'password'=>'12345678'

        ]
    ]);
    if($user){
        echo "<h1>Data Successfully Added.</h1>";
    }
}
```

Check unique email

```php
public function addUser(){
    $user = DB::table('users')->insertOrIgnore([
        'name'=>'d',
        'email'=>'d@gmail.com',
        'password'=>'12345678'

        ]);
    if($user){
        echo "<h1>Data Successfully Added.</h1>";
    }else{
        echo "<h1>Data not Added.</h1>";
    }
}
```

Update

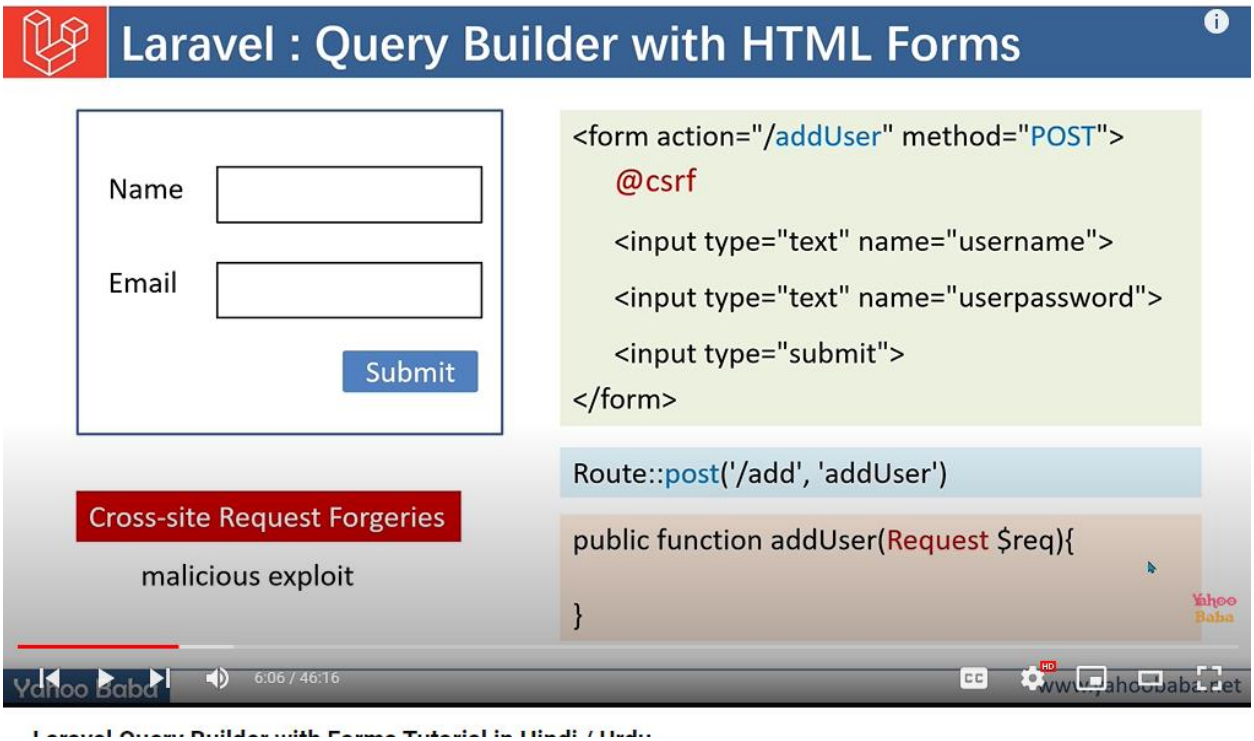
```php
public function updateUser(){
    $user = DB::table('users')->where('id',1)->update(['name'=>'noor']);
}
```

```php
Route::get('/update',[UserController::class,'updateUser']);
```

Delete

```php
public function deleteUser(){
    $user = DB::table('users')->where('id',1)->delete();
}
```

Query Builder with Html Forms



Controller

```php
public function addUser(Request $req){
    $user = DB::table('users')->insert([
        'name'=>$req->username, //form input field er name je thakbe setai dite hobe
        'email'=>$req->useremail,
        'password'=>$req->userpassword

        ]);
    if($user){
        echo "<h1>Data Successfully Added.</h1>";
    }else{
        echo "<h1>Data not Added.</h1>";
    }
}
```

//route

```php
Route::post('/add',[UserController::class,'addUser'])->name('addUser');
Route::view('newuser','/adduser');
```

//form

```blade
<body>
    <div class="container">
        <h1>Add New User</h1>
        <form action="{{route('addUser')}}" method="POST">
            @csrf
            <input type="text" placeholder="Enter the Name" name="username">
            <input type="email" placeholder="Enter the Email" name="useremail">
            <input type="password" placeholder="Enter the password" name="userpassword
            <button type="submit">submit</button>
        </form>
    </div>
</body>
```

Update

```php
public function updateUser(Request $req, $id){
    $user = DB::table('users')
                ->where('id', $id)
                ->update([
                    'name' => $req->username,
                    'email' => $req->useremail,
                    'age' => $req->userage,
                    'city' => $req->usercity,
                ]);

    if($user){
        echo "<h1>Data Successfully Update.</h1>";
    }else{
        echo "<h1>Data not Updated.</h1>";
    }
```

37:02 / 46:16

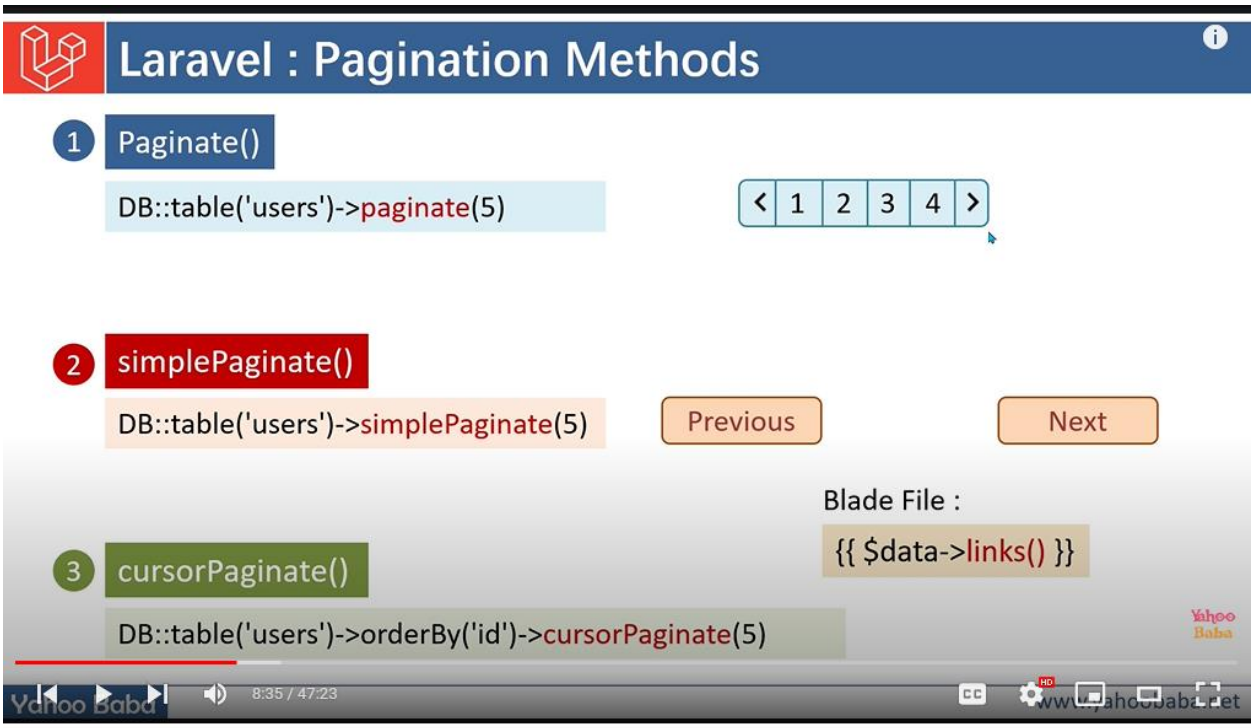
1.first route->form theke data control kore

2.second route->home page update page er niye jay

```php
Route::post('/update/{id}', 'updateUser')->name('update.user');
Route::get('/updatepage/{id}', 'updatePage')->name('update.page');
```

```blade
<div class="container">
  <div class="row">
    <div class="col-4">
      <h1>Update User Data</h1>
      <form action="{{ route('update.user',$data->id) }}" method="POST">
        @csrf
        <div class="mb-3">
          <label class="form-label">Name</label>
          <input type="text" value="{{ $data->name }}" class="form-control" name="usern
        </div>
        <div class="mb-3">
          <label class="form-label">Email</label>
          <input type="text" value="{{ $data->email }}" class="form-control" name="user
        </div>
        <div class="mb-3">
          <label class="form-label">Age</label>
          <input type="text" value="{{ $data->age }}" class="form-control" name="userag
        </div>
```

Pagination




```
public function showUsers(){
    $users = DB::table('users')
                ->simplePaginate(4);

    return view('allusers',['data' =>  $users]);
}
```

Show the pagination button

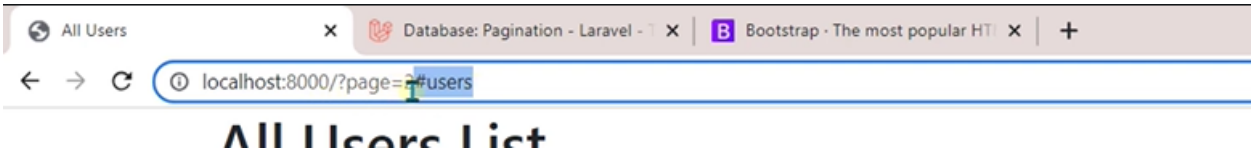
```
<div class="mt-5">
    {{ $data->links() }}
</div>
```

Urlbar

```php
public function showUsers(){
    $users = DB::table('users')
                ->paginate(5)
                ->appends(['sort' => 'votes','test' => 'abc']);

    //    return $users;
    return view('allusers',['data' =>  $users]);
}
```

Note appends kaj korte hobe shudu

url bar hash value add




```php
public function showUsers(){
    $users = DB::table('users')
                ->paginate(5)
                ->fragment('users');

    //    return $users;
    return view('allusers',['data' =>  $users]);
}
```

($fragment = null)

@param string|null

Note hash value jonno shudu fragment add korte hobe

Join Table

Laravel : Join Table

Students Table

Id	Name	Age	City
1	Ram Kumar	19	1
2	Salman Khan	18	3
3	Meera Khan	19	2
4	Sarita Kumari	21	1

Cities Table

Cid	City
1	Agra
2	Bhopal
3	Delhi

```
SELECT *
FROM students
INNER JOIN cities
ON students.city = cities.cid;
```

```
DB::table('students')
->join('cities','students.city','=','cities.cid')
->get();
```

Laravel : MySQL Joins

- INNER JOIN `join()`
- LEFT JOIN `leftJoin()`
- RIGHT JOIN `rightJoin()`
- CROSS JOIN `crossJoin()`

```php
public function showStudents(){
    $students = DB::table('students')
                ->join('cities','students.city','=','cities.id')
                ->get();

        return $students;
}
```

```php
Route::get('/',[StudentController::class,'showStudents']);
```

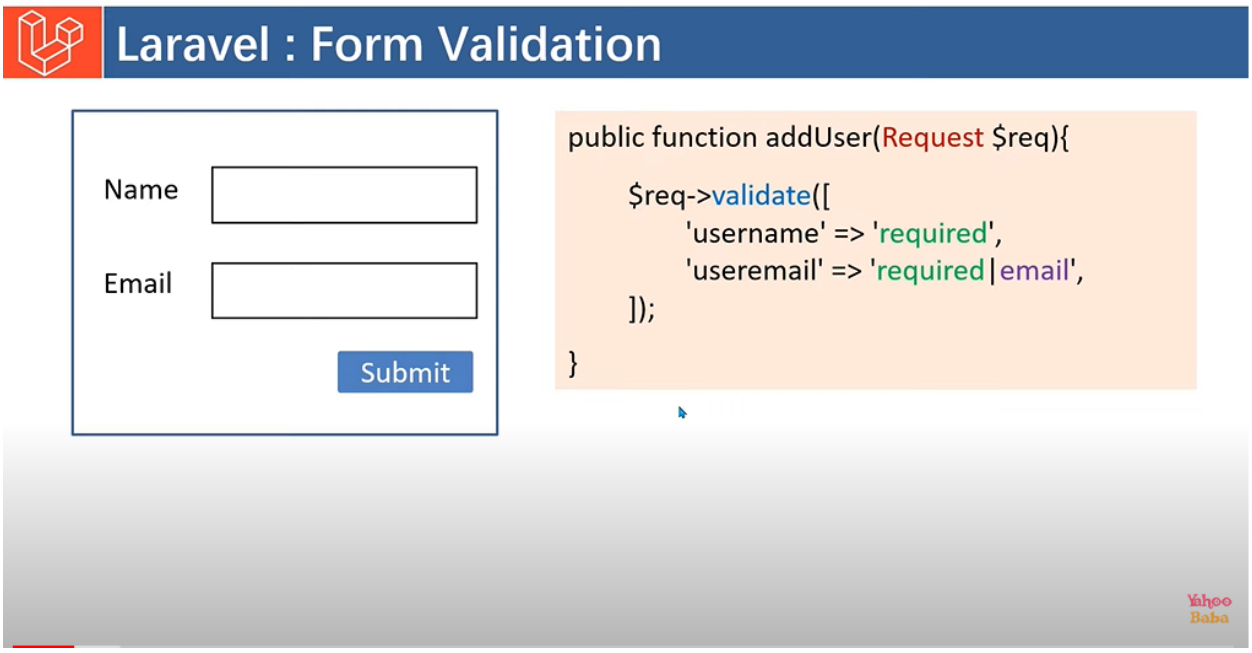
Select using

```php
public function showStudents(){
    $students = DB::table('students')
                ->join('cities','students.city','=','cities.id')
                ->select('students.*','cities.city_name')
                ->get();

        // return $students;

    return view('welcome',compact('students'));
}
```

Left join

```php
$students = DB::table('students')
            ->leftJoin('cities',function(JoinClause $join){
                $join->on('students.city','=','cities.id')
                    ->where('students.name','like','a%');
            })
            ->select('students.*','cities.city_name')
            ->get();

    // return $students;

return view('welcome',compact('students'));
```

Form Validation



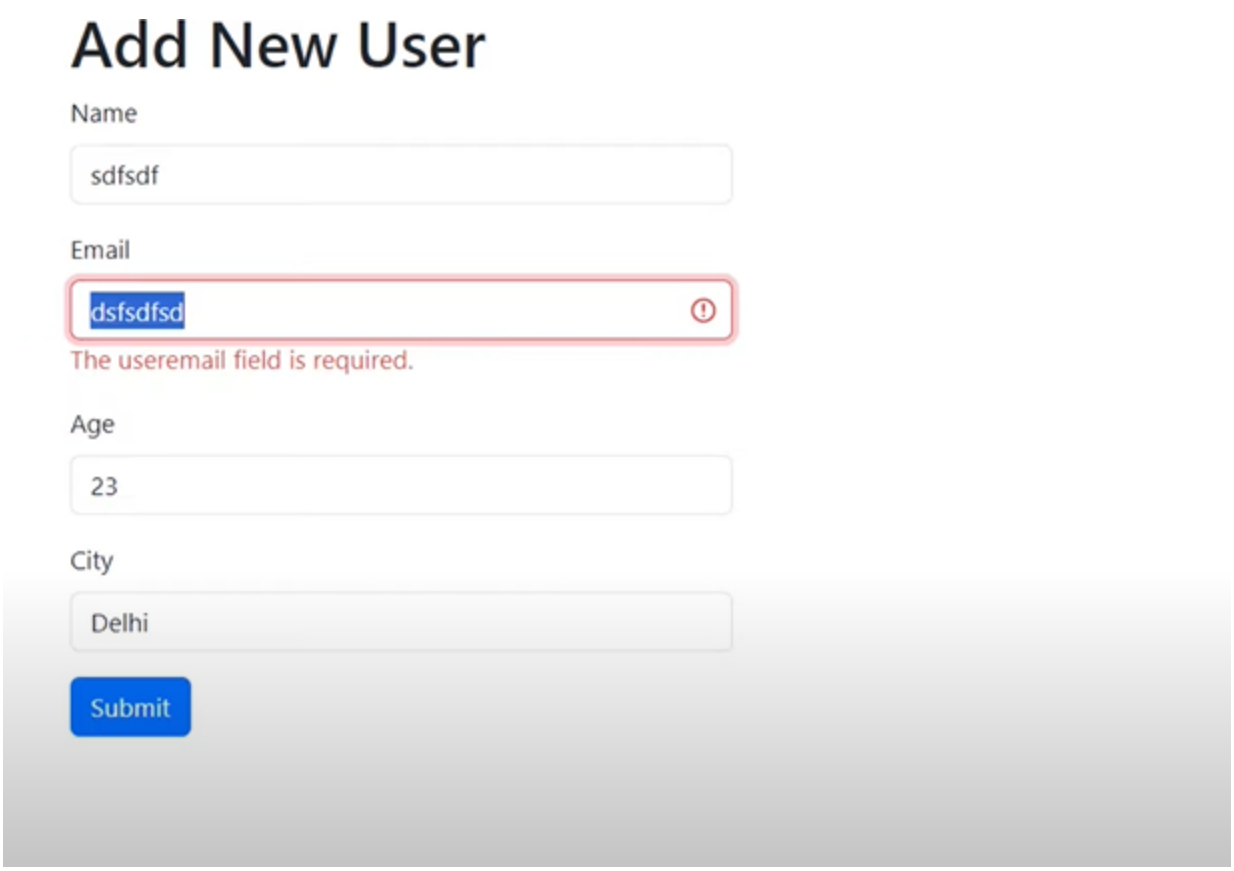
Error message

```
<h1>Add New User</h1>
@if ($errors->any())
    <ul class="alert alert-danger">
        @foreach ($errors->all() as $error)
            <li>{{ $error }}</li>
        @endforeach
    </ul>
@endif
```

Single error message show

```
<div class="mb-3">
    <label class="form-label">Name</label>
    <input type="text" class="form-control" name="username">
    <span class="text-danger">
        @error('username')
            {{ $message }}
        @enderror
    </span>
</div>
```

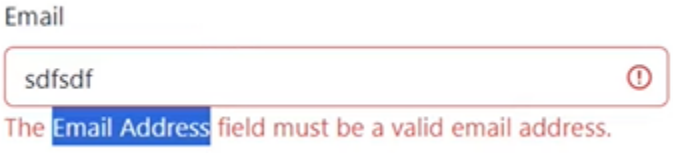
Invalid error show

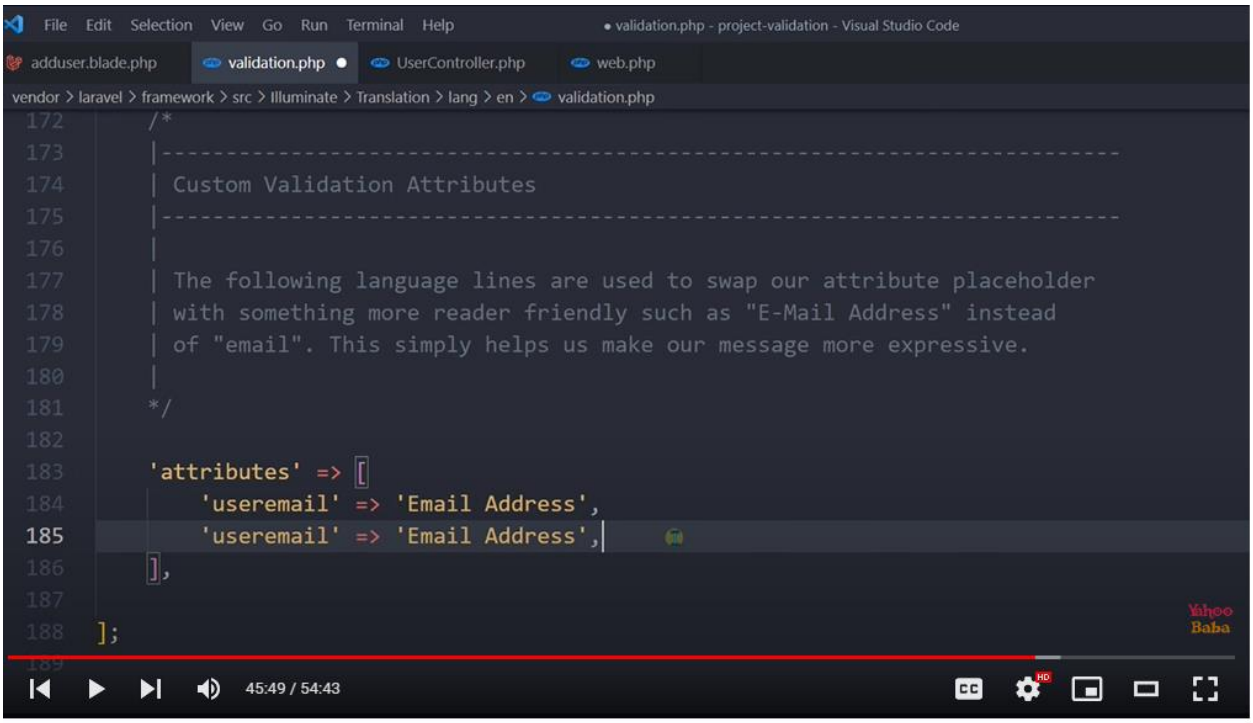



```
<div class="mb-3">
  <label class="form-label">Name</label>
  <input type="text" class="form-control @error('username') is-invalid @enderror" name
  <span class="text-danger">
    @error('username')
      {{ $message }}
    @enderror
  </span>
</div>
```

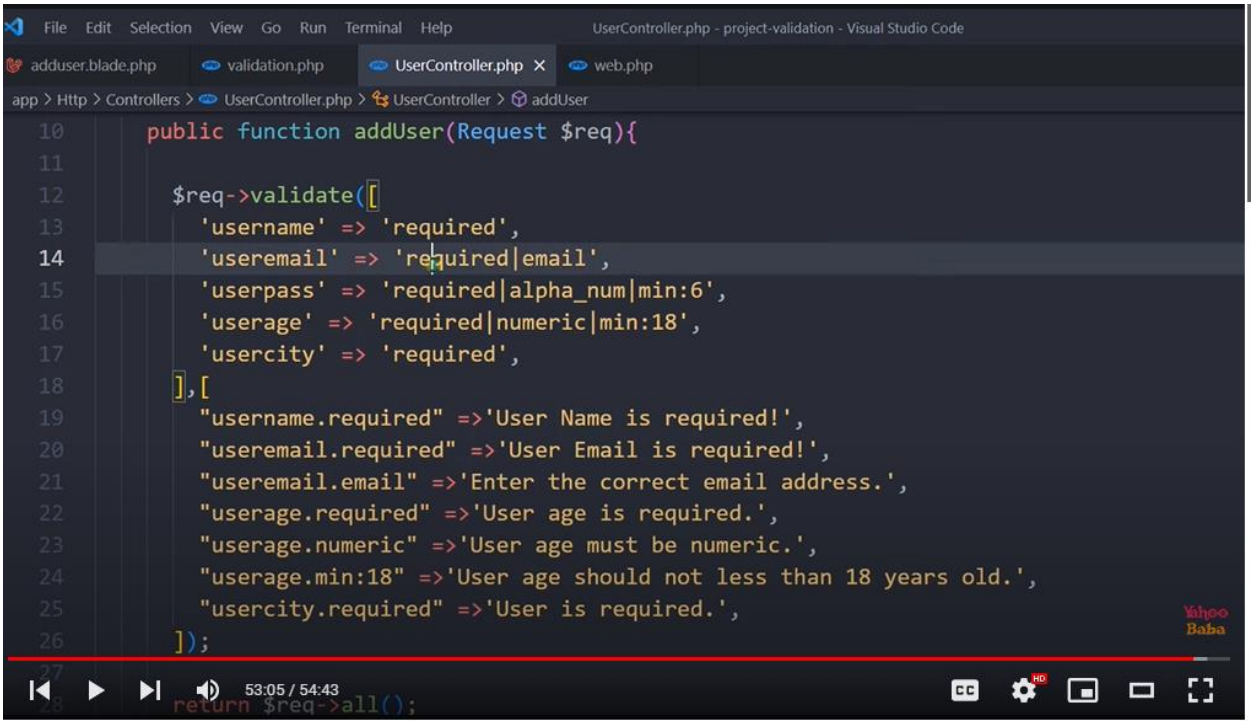
https://laravel.com/docs/11.x/validation#main-content

validation attributes change

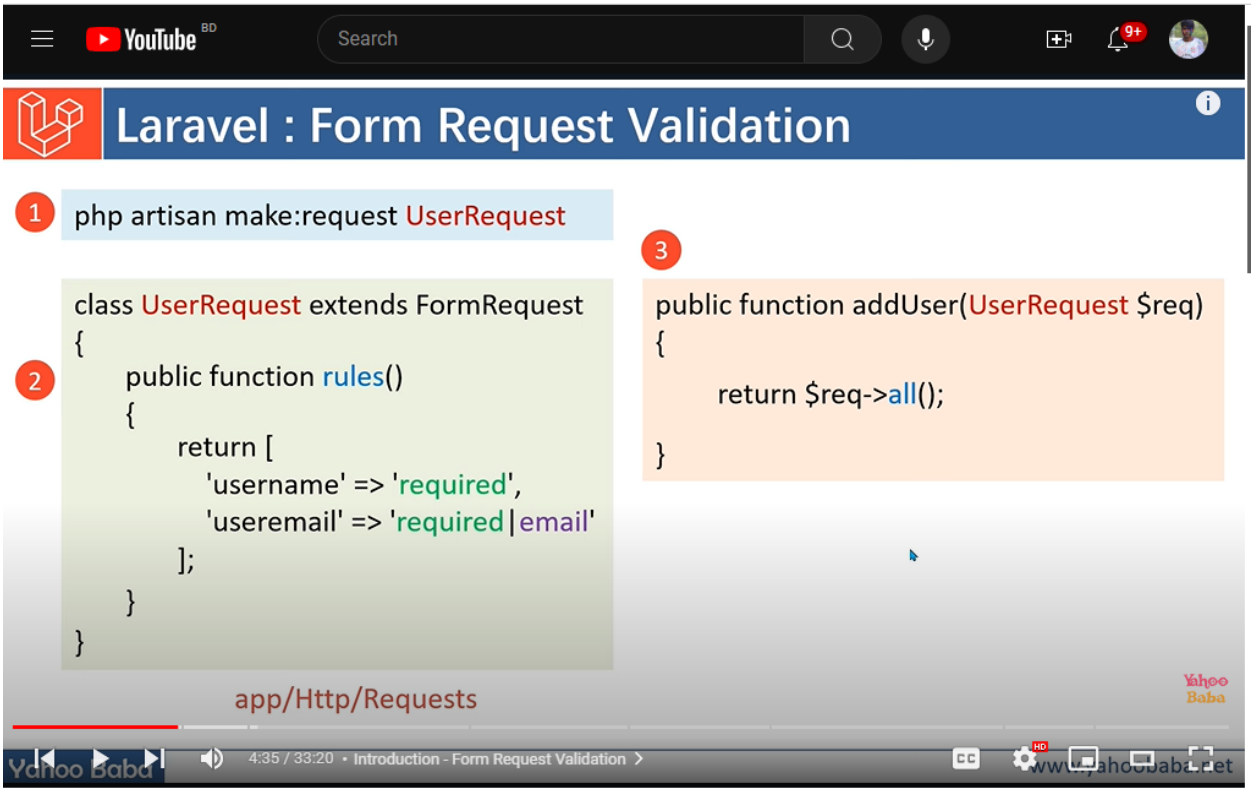




Error required message show

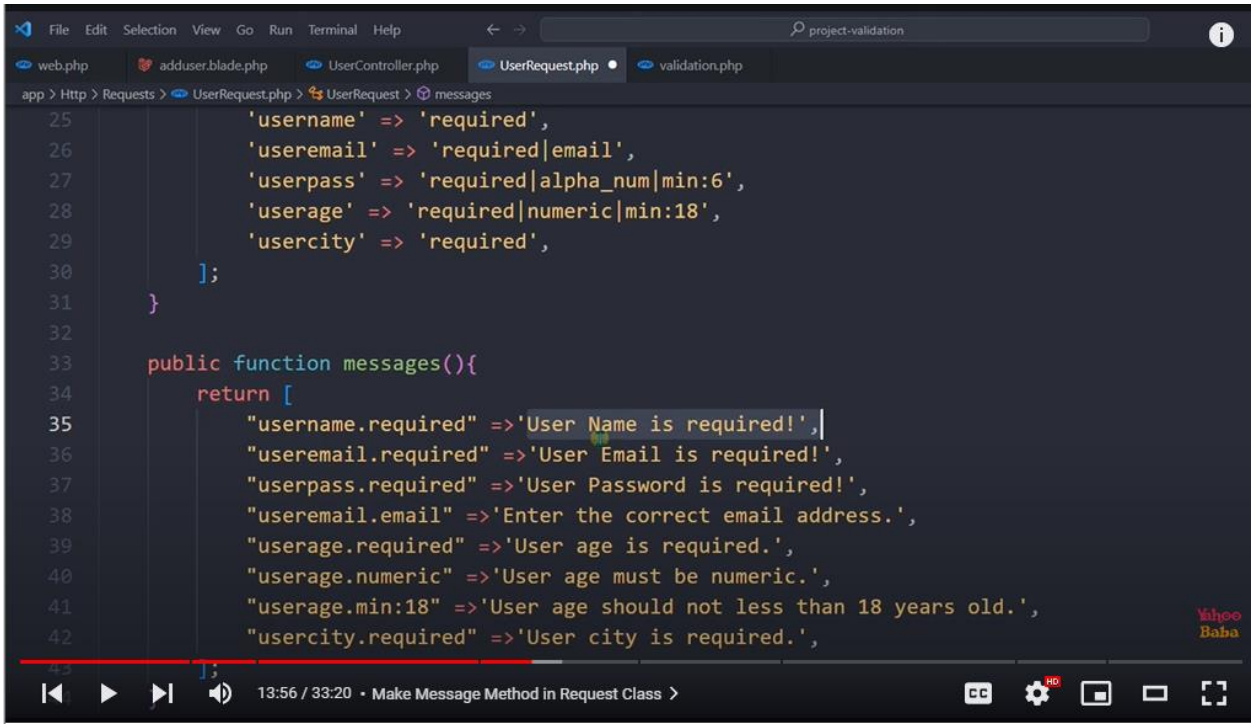


Form Request Validation

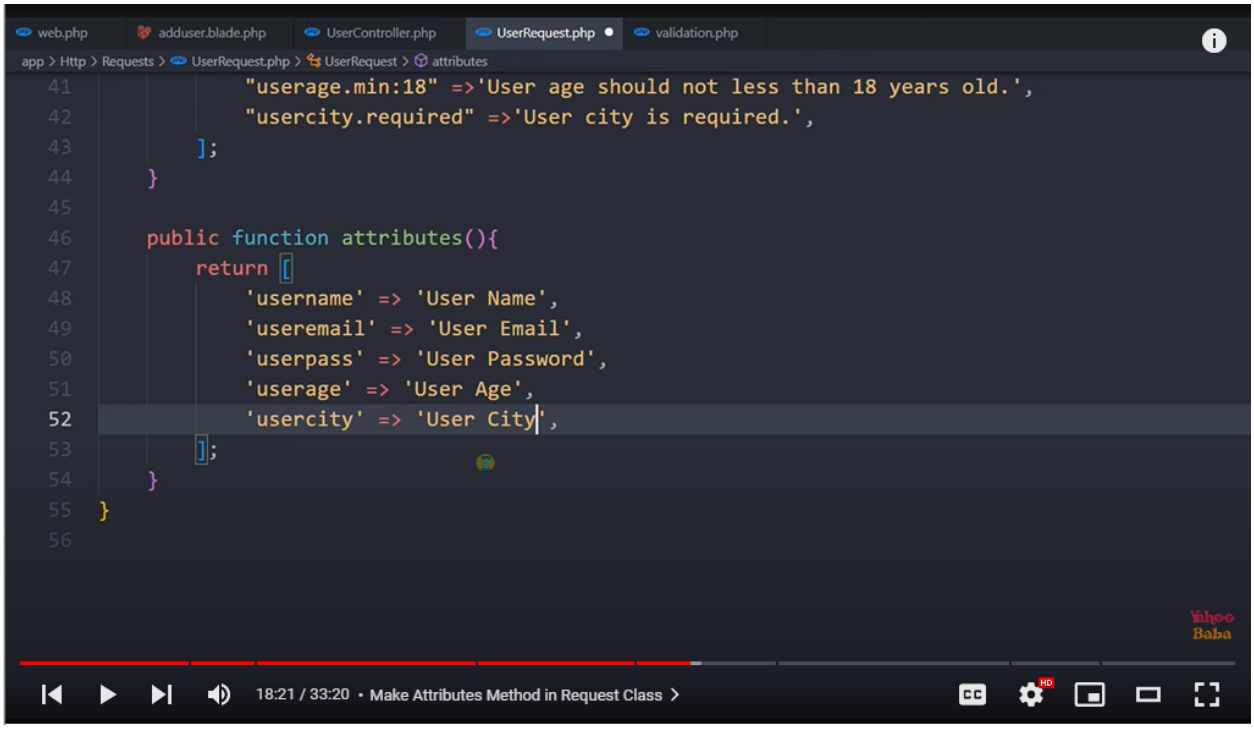


Note: controller class er UserRequest import korbo

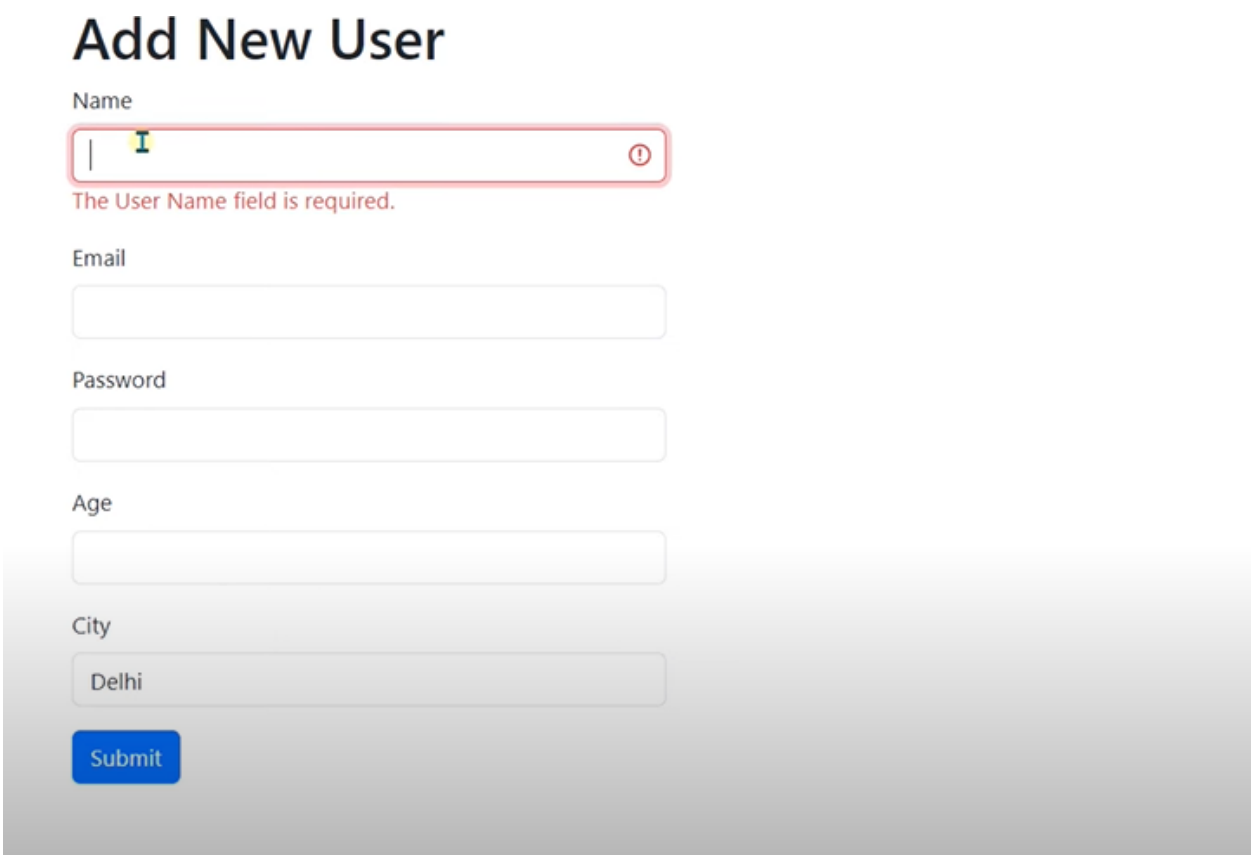
Message show

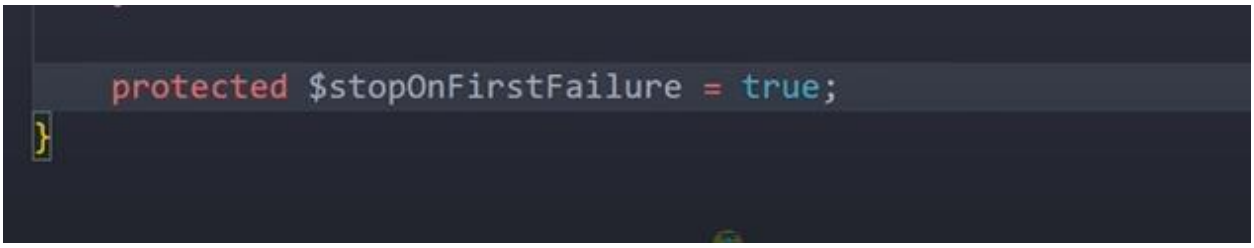


Attributes

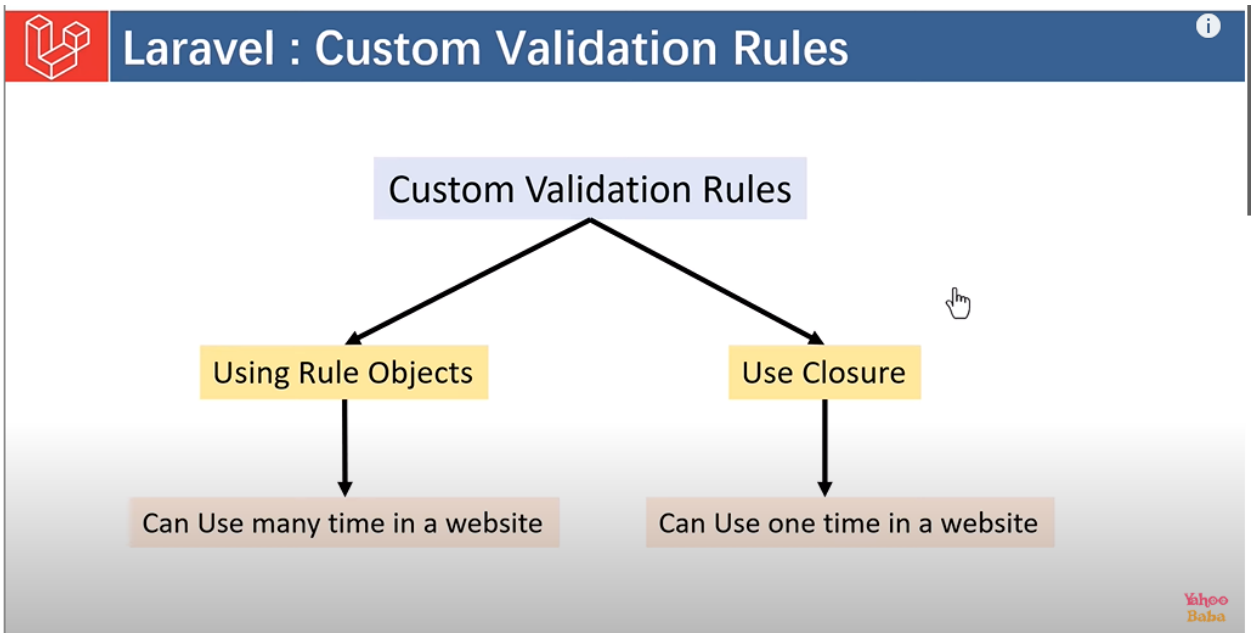


Only one error message show

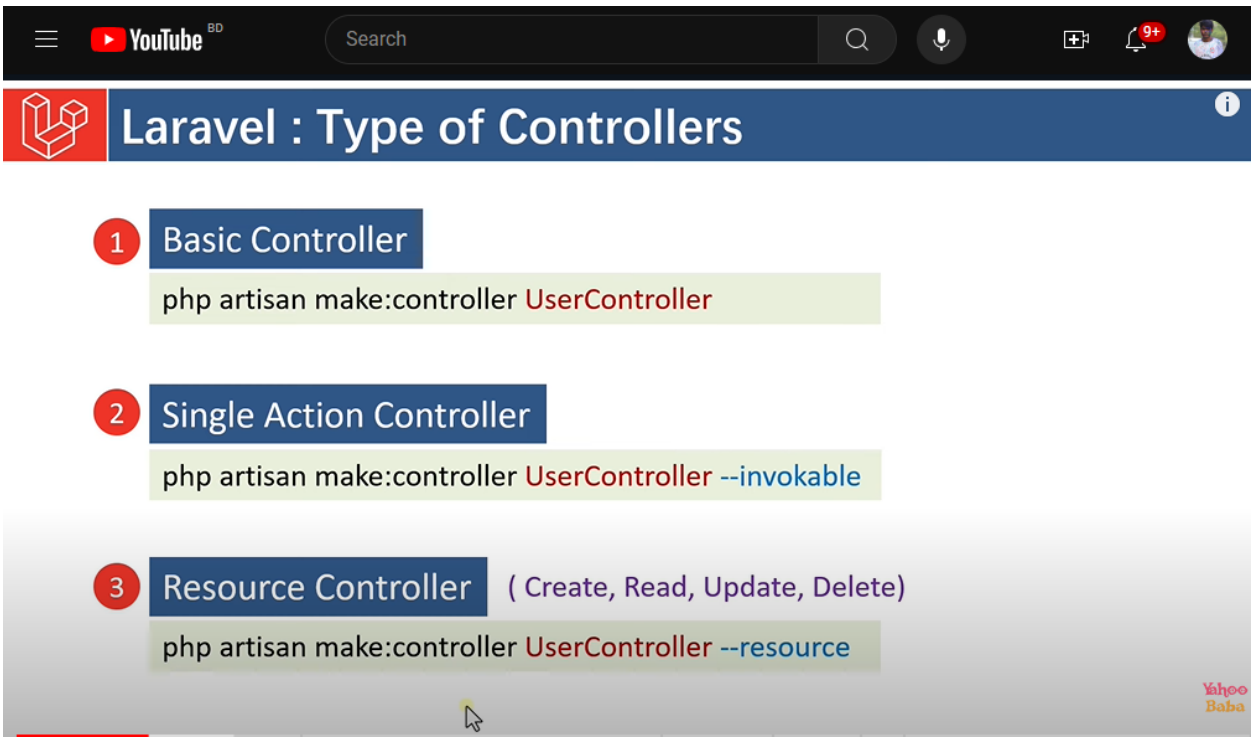




Custom Validation Rule



Resource controller



Note automatic create many function

Example

Laravel : Resource Controller

UserController.php

```
class UserController extends Controller
{
    public function index(){ }

    public function create(){ }

    public function store(Request $request){ }

    public function show(User $user){ }

    public function edit(User $User){ }

    public function update(Request $request, User $user){ }

    public function destroy(User $user){ }
}
```

Laravel : Route for Resource Controller

routes/web.php

```
use App\Http\Controllers\UserController;

Route::resource('users', UserController::class);
```

Verb	URI	Action	Route Name
GET	/users	index	users.index
GET	/users/create	create	users.create
POST	/users	store	users.store
GET	/users/{user}	show	users.show
GET	/users/{user}/edit	edit	users.edit
PUT/PATCH	/users/{user}	update	users.update
DELETE	/users/{user}	destroy	users.destroy

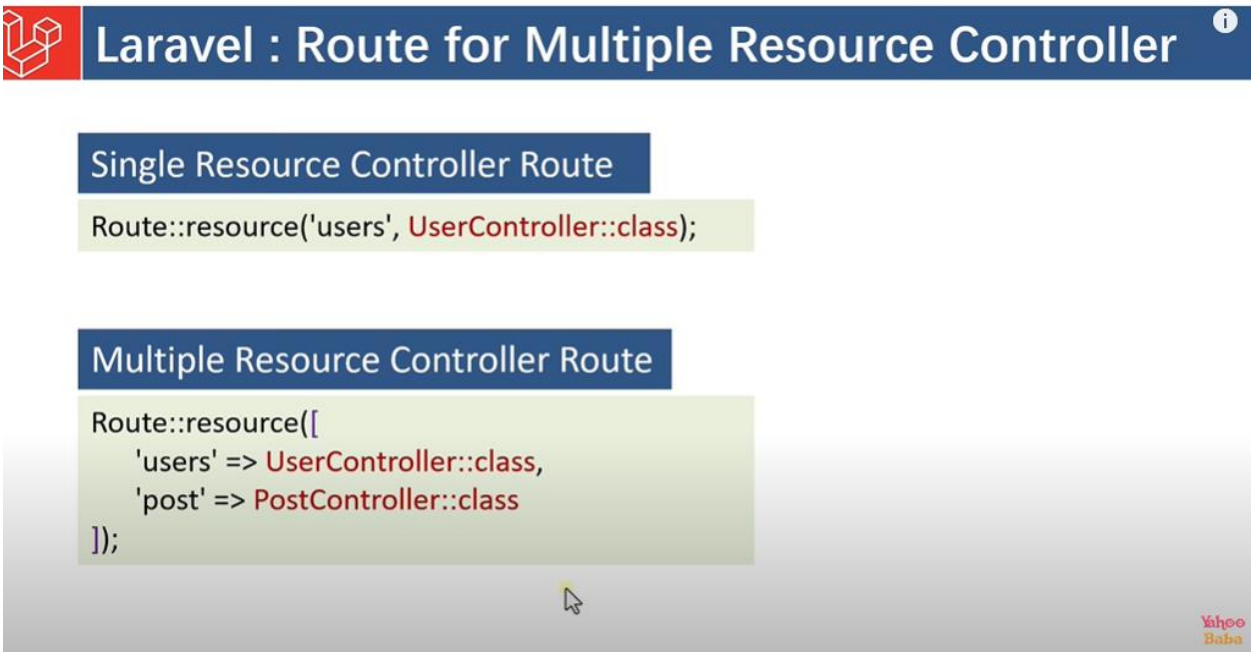
Only method show create update show

```php
Route::resource('users',UserController::class)->only([
    'create','update','show'
]);
```

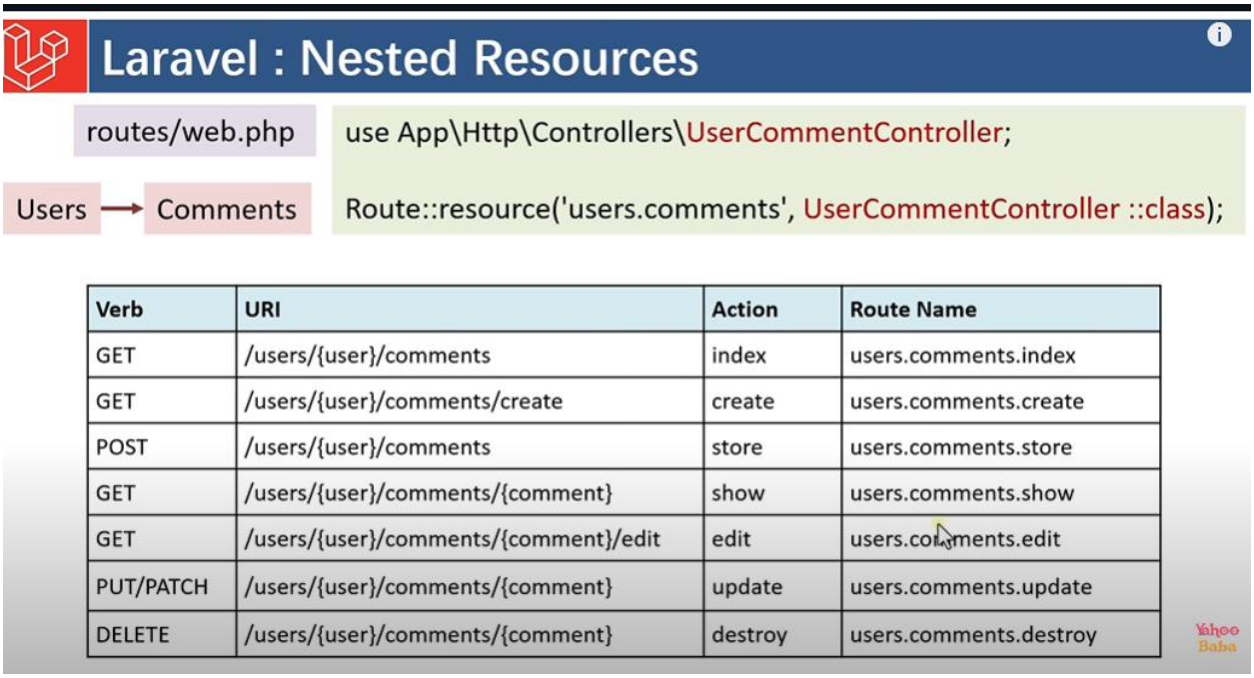
All method show without update and show method

```php
Route::resource('users',UserController::class)->except([
    'update','show'
]);
```

Multiple Resource controller route

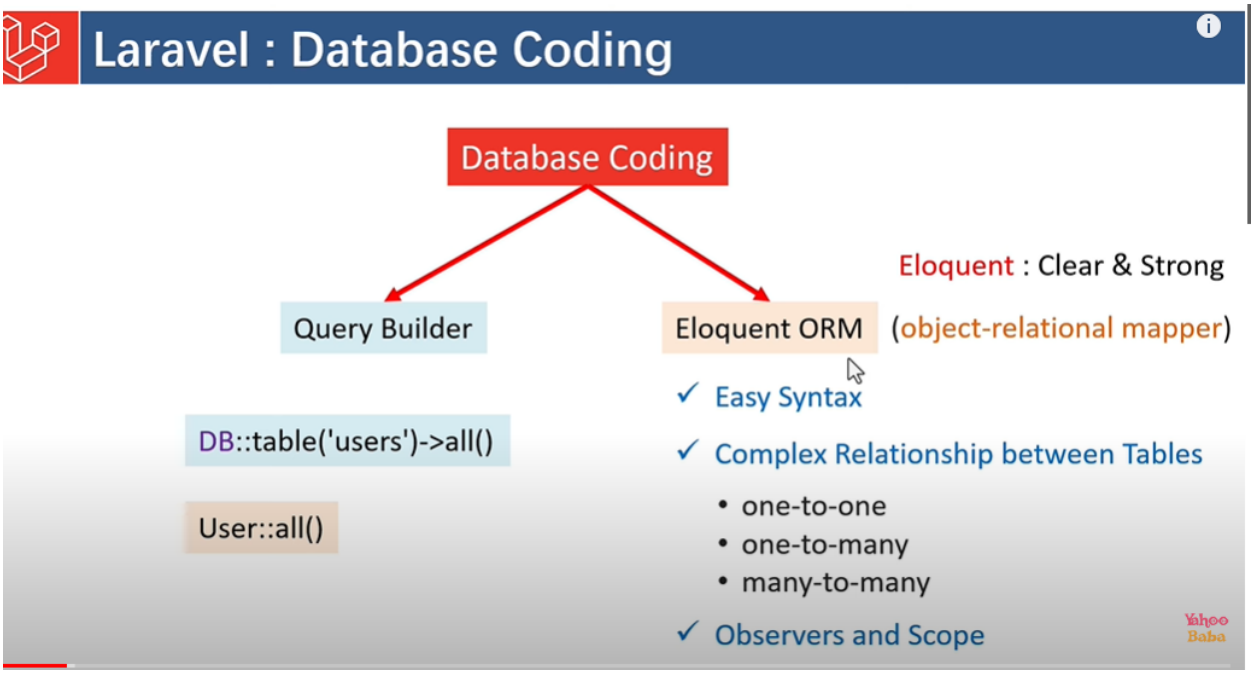


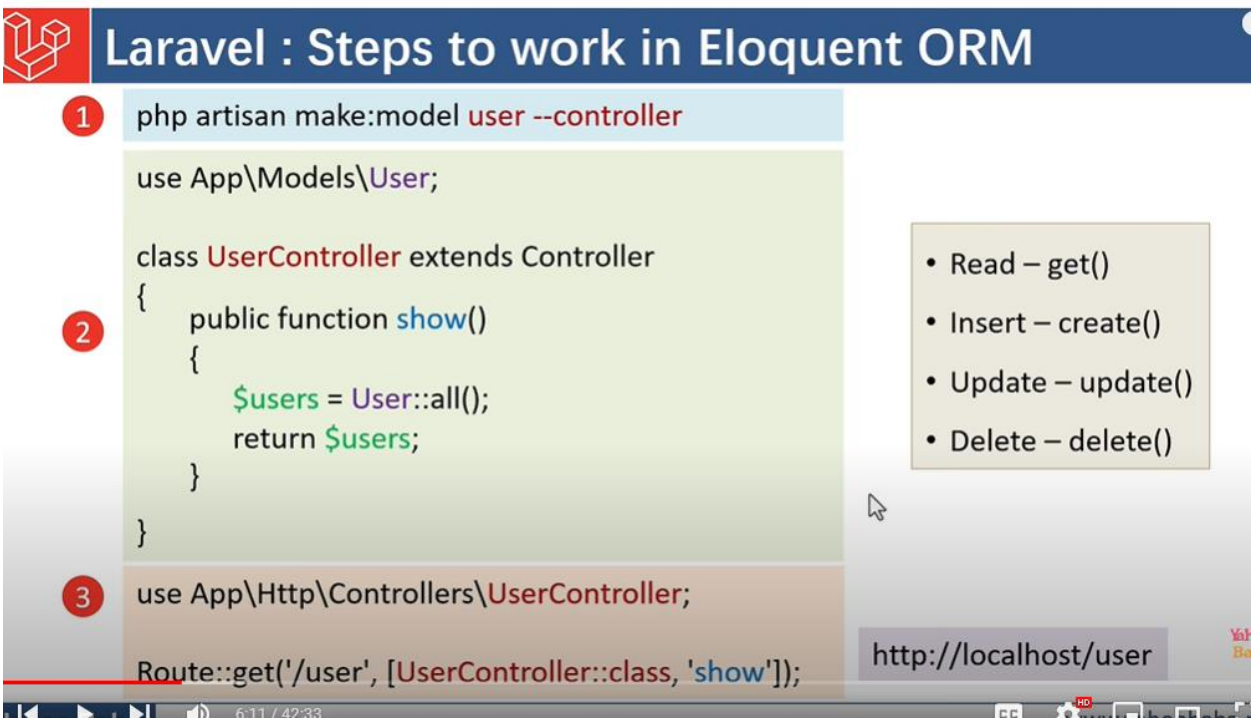
Nested Resource

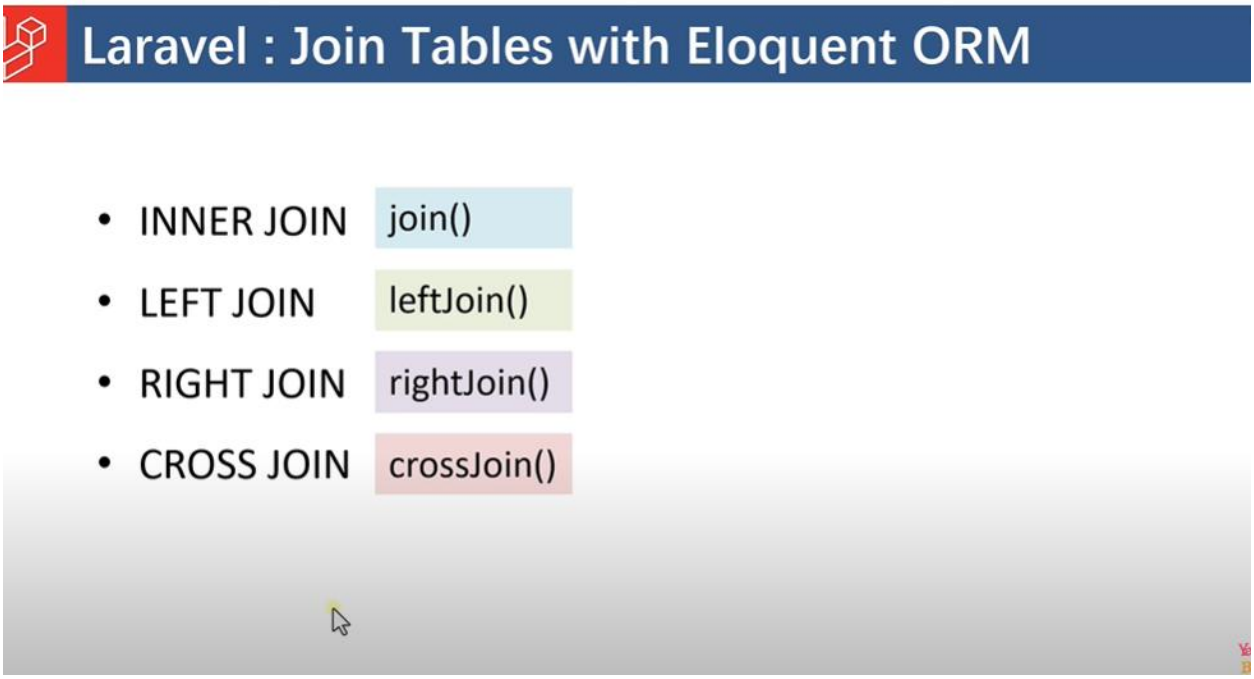


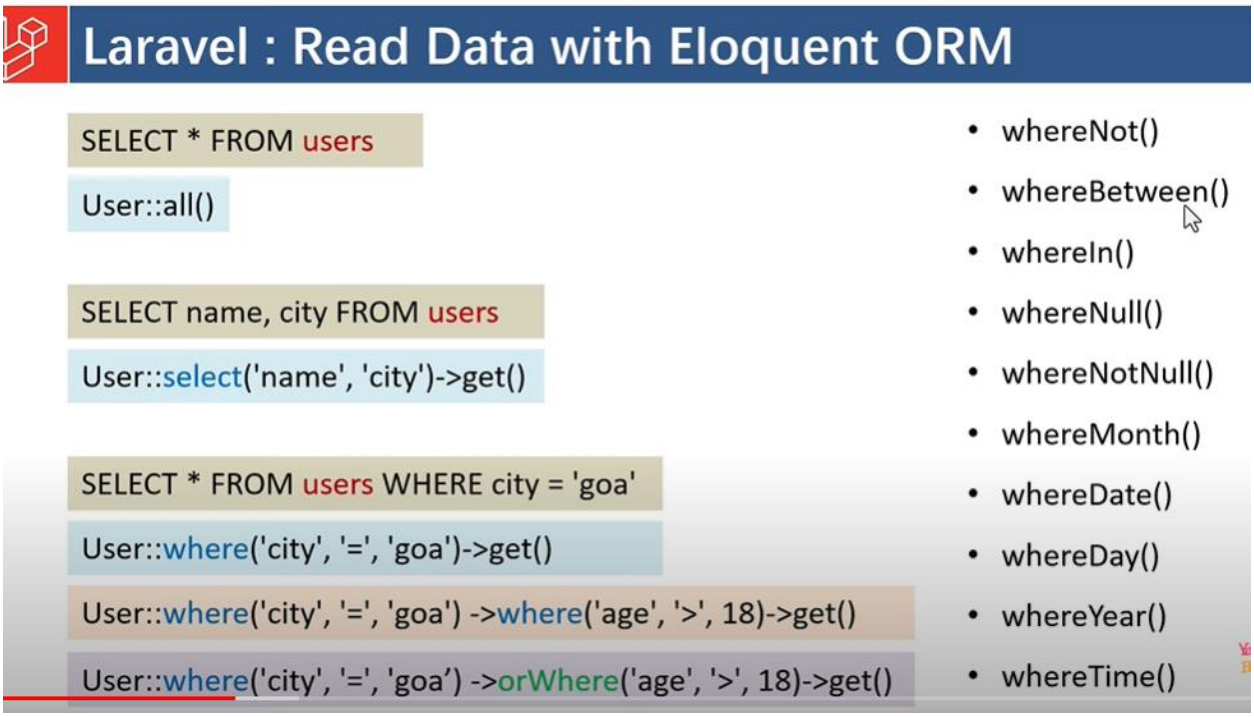
Nested resource

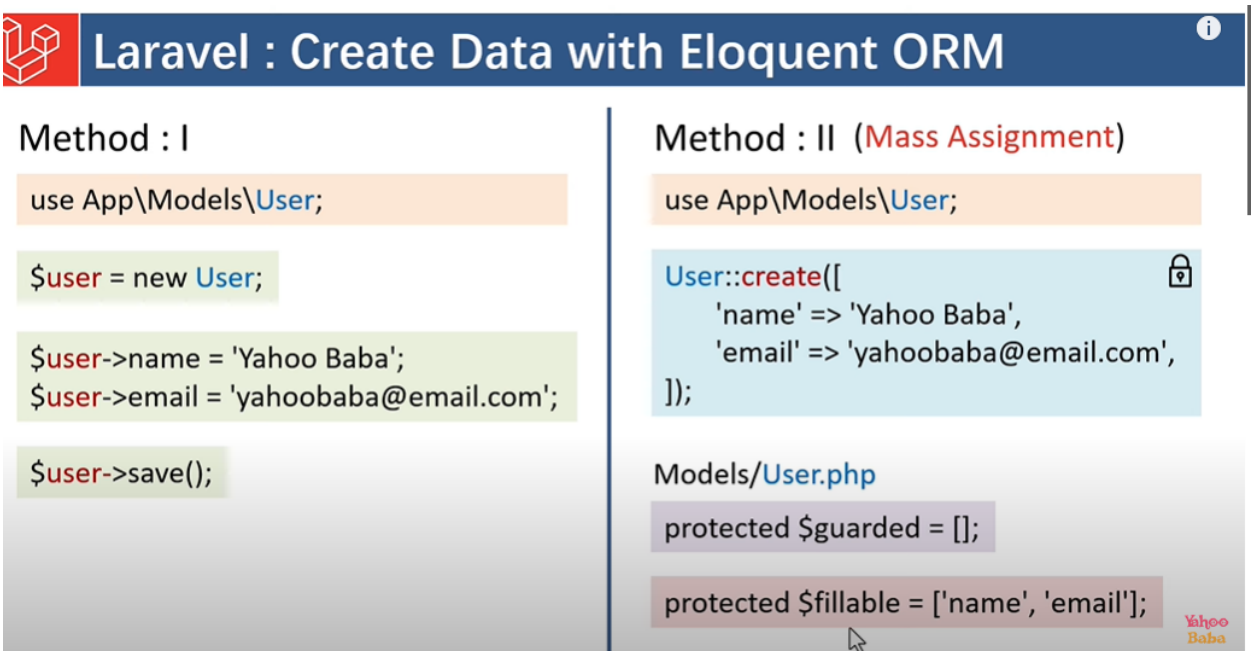
Eloquent


Laravel : Database Coding
Database Coding
Query Builder
Eloquent ORM
Eloquent : Clear & Strong
(object-relational mapper)
DB::table('users')->all()
User::all()
✓ Easy Syntax
✓ Complex Relationship between Tables
• one-to-one
• one-to-many
• many-to-many
✓ Observers and Scope


Laravel : Steps to work in Eloquent ORM
1
php artisan make:model user --controller
2
use App\Models\User;
class UserController extends Controller
{
public function show()
{
$users = User::all();
return $users;
}
}
3
use App\Http\Controllers\UserController;
Route::get('/user', [UserController::class, 'show']);
• Read – get()
• Insert – create()
• Update – update()
• Delete – delete()
http://localhost/user


Laravel : Join Tables with Eloquent ORM
INNER JOIN
join()
LEFT JOIN
leftJoin()
RIGHT JOIN
rightJoin()
CROSS JOIN
crossJoin()


Laravel : Read Data with Eloquent ORM
SELECT * FROM users
User::all()
SELECT name, city FROM users
User::select('name', 'city')->get()
SELECT * FROM users WHERE city = 'goa'
User::where('city', '=', 'goa')->get()
User::where('city', '=', 'goa') ->where('age', '>', 18)->get()
User::where('city', '=', 'goa') ->orWhere('age', '>', 18)->get()
whereNot()
whereBetween()
whereIn()
whereNull()
whereNotNull()
whereMonth()
whereDate()
whereDay()
whereYear()
whereTime()



Note guarded holo….je gulo data show korte chai na seigulo guarded bitore rakhobo

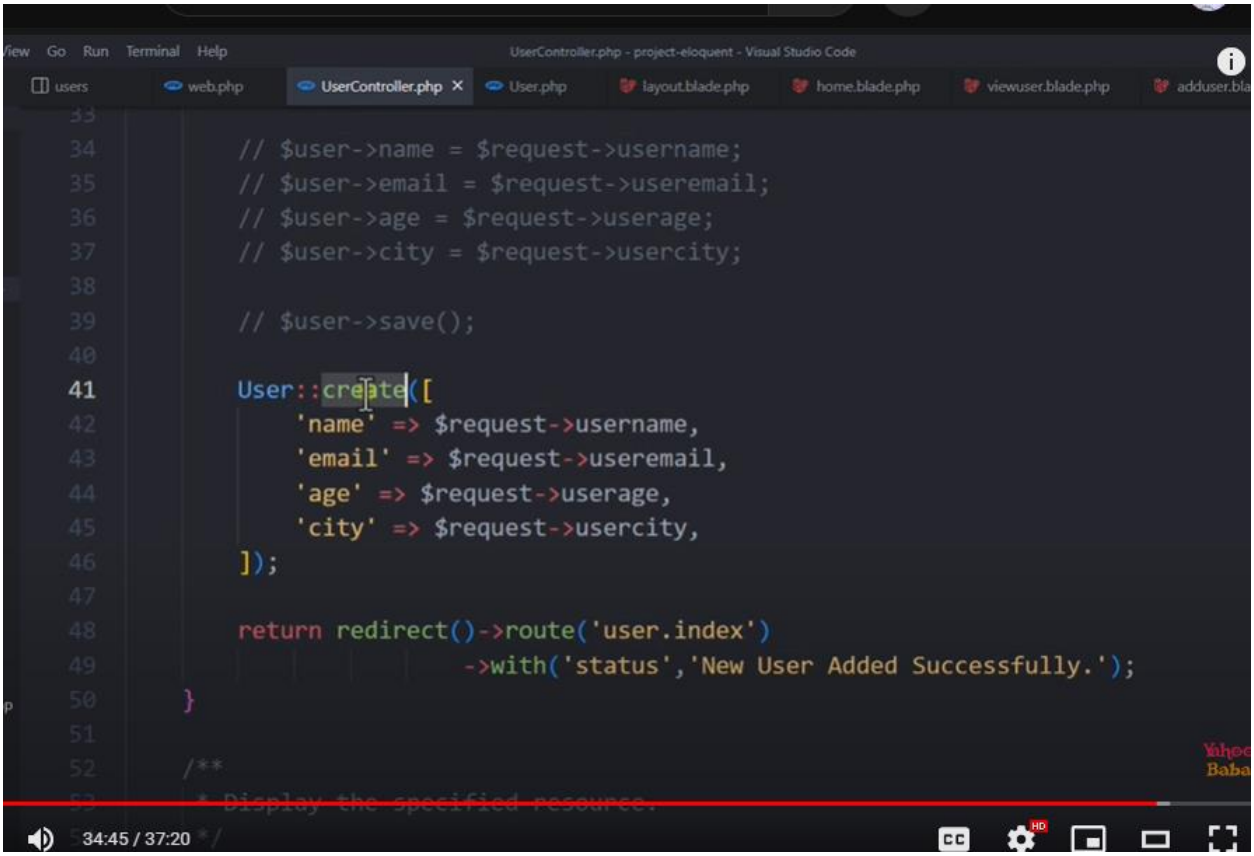
Fillable je gulo show korbo fillable bitore rakhobo

Create data

```
public function store(Request $request)
{
    $user = new User;

    $user->name = $request->username;
    $user->email = $request->useremail;
    $user->age = $request->userage;
    $user->city = $request->usercity;

    $user->save();

    return redirect()->route('user.index');
}
```

Second method

Step-1 controller



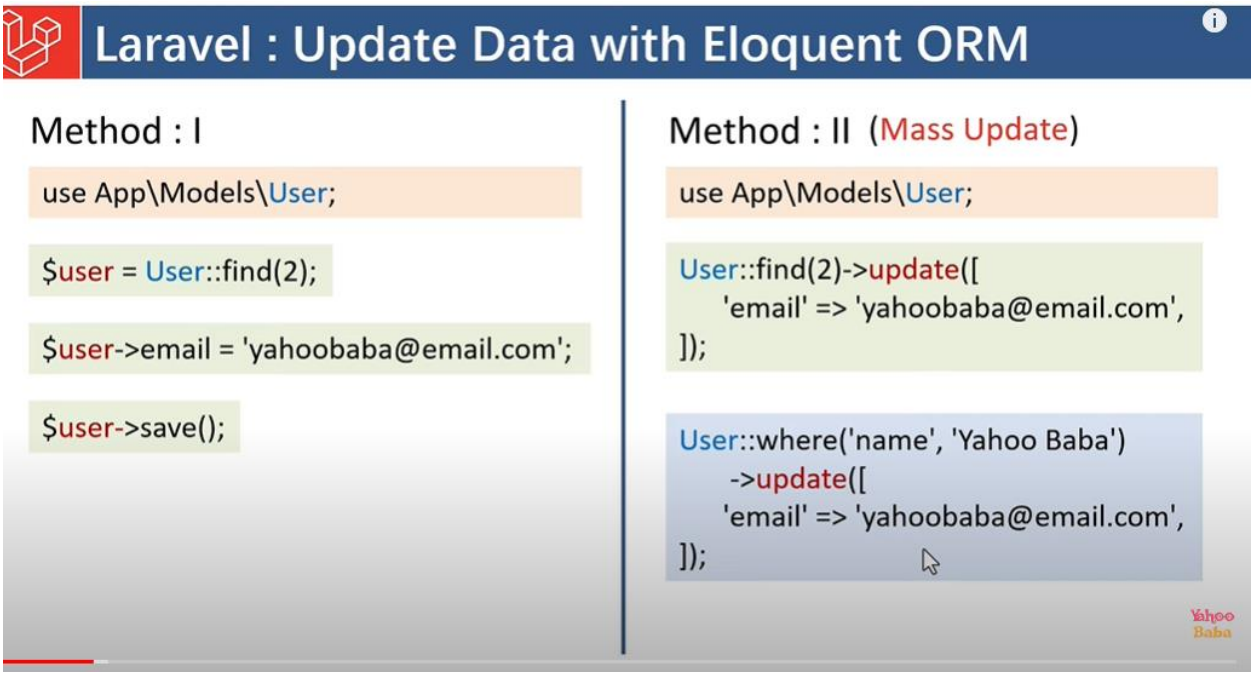
Step-2 model

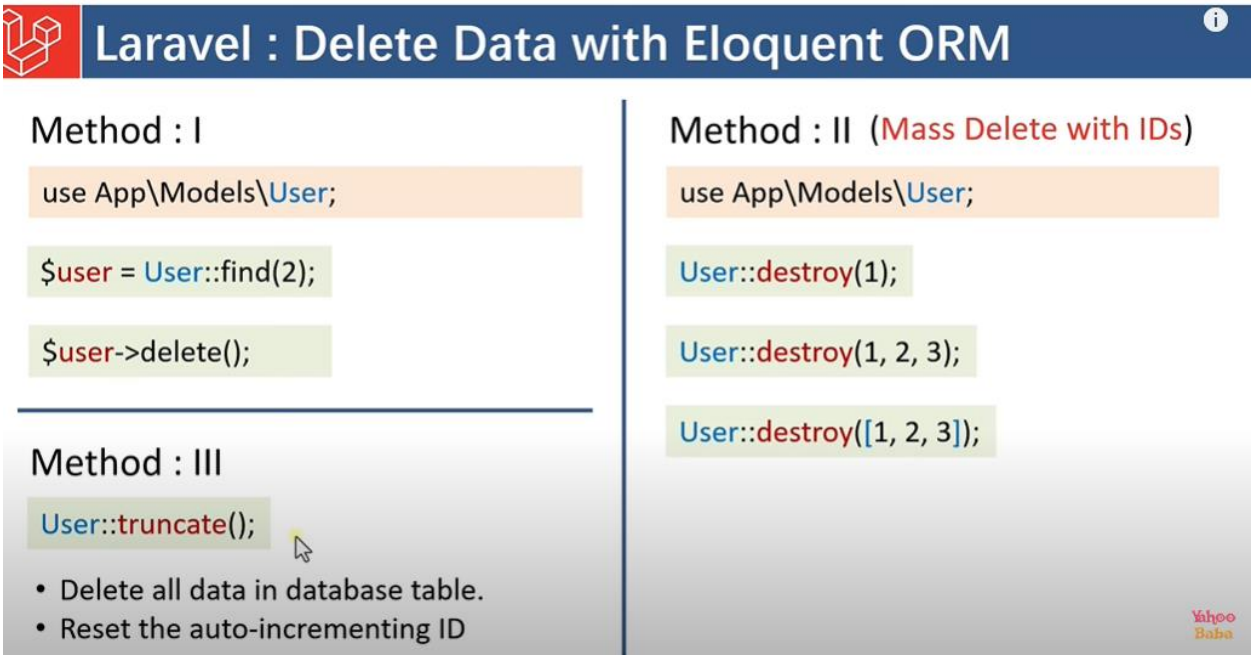
Run Terminal Help • User.php - project-eloquent - Visual Studio Code

web.php | UserController.php | User.php | layout.blade.php | home.blade.php | viewuser.blade.php | adduser.bl

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    use HasFactory;

    public $timestamps = false;

    protected $guarded = [];
}
```

Update

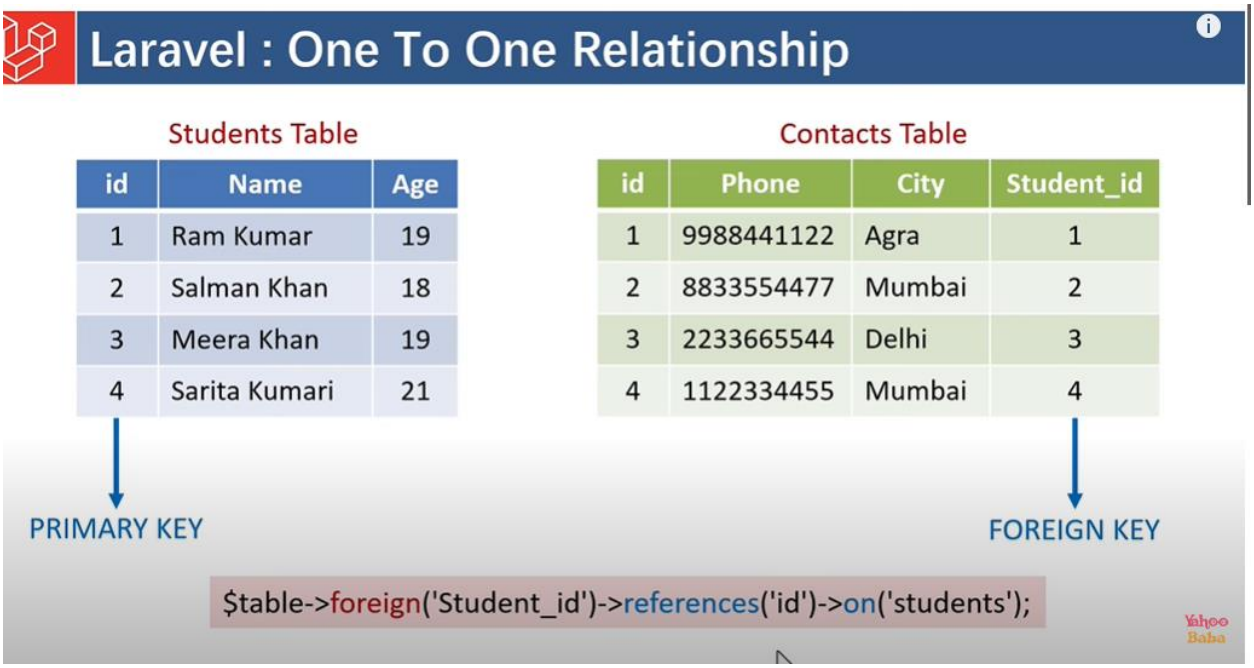




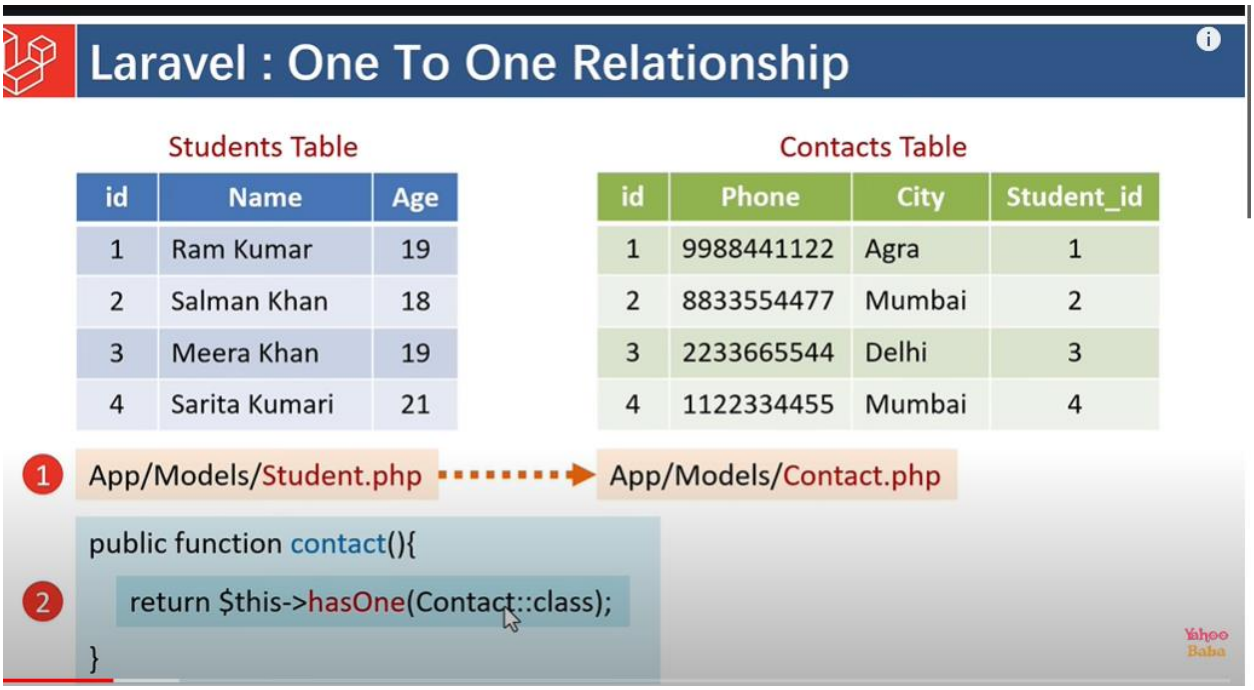

```
<td><a href="{{ route('user.show',$user->id) }}" cl
<td>
```

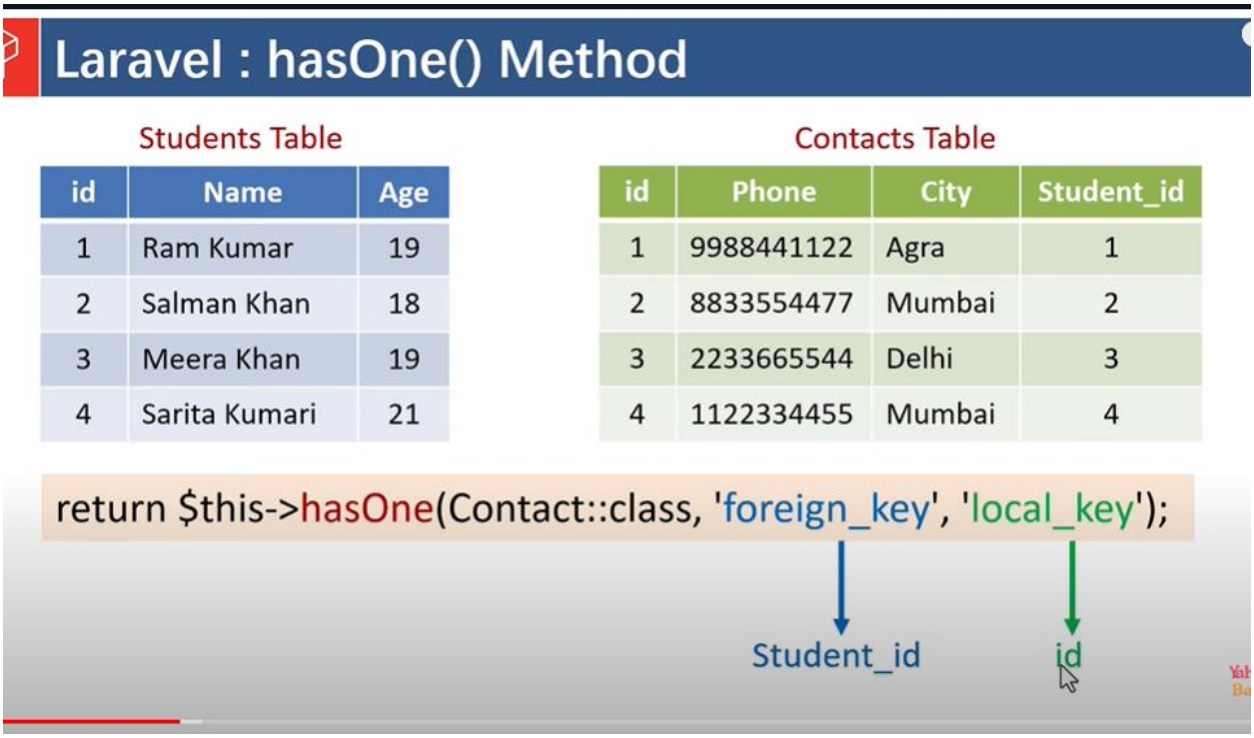
Relationships

Step-1 foreign key mrigate file

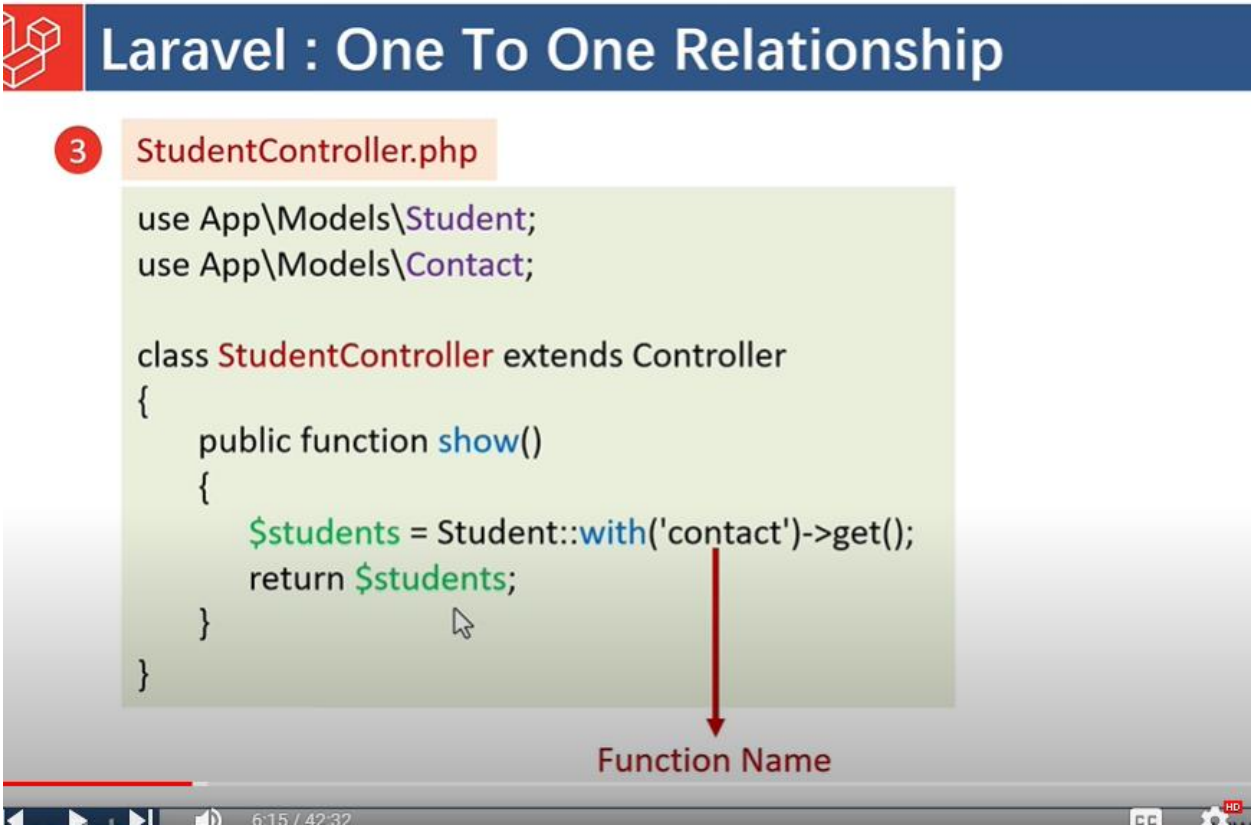


Step-2:model

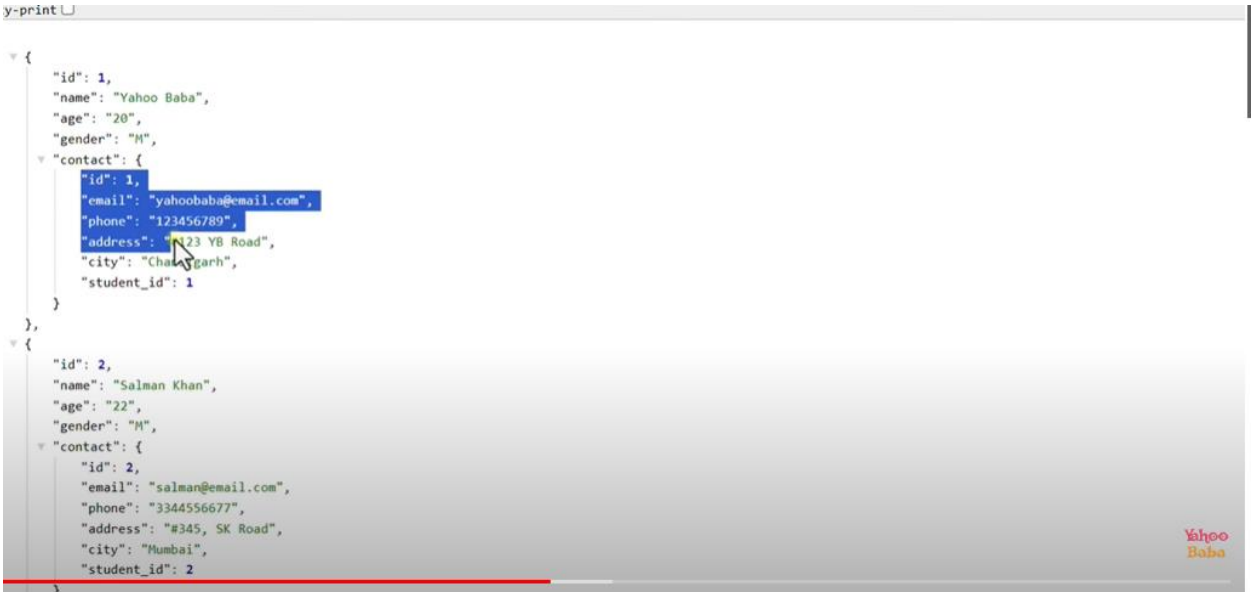




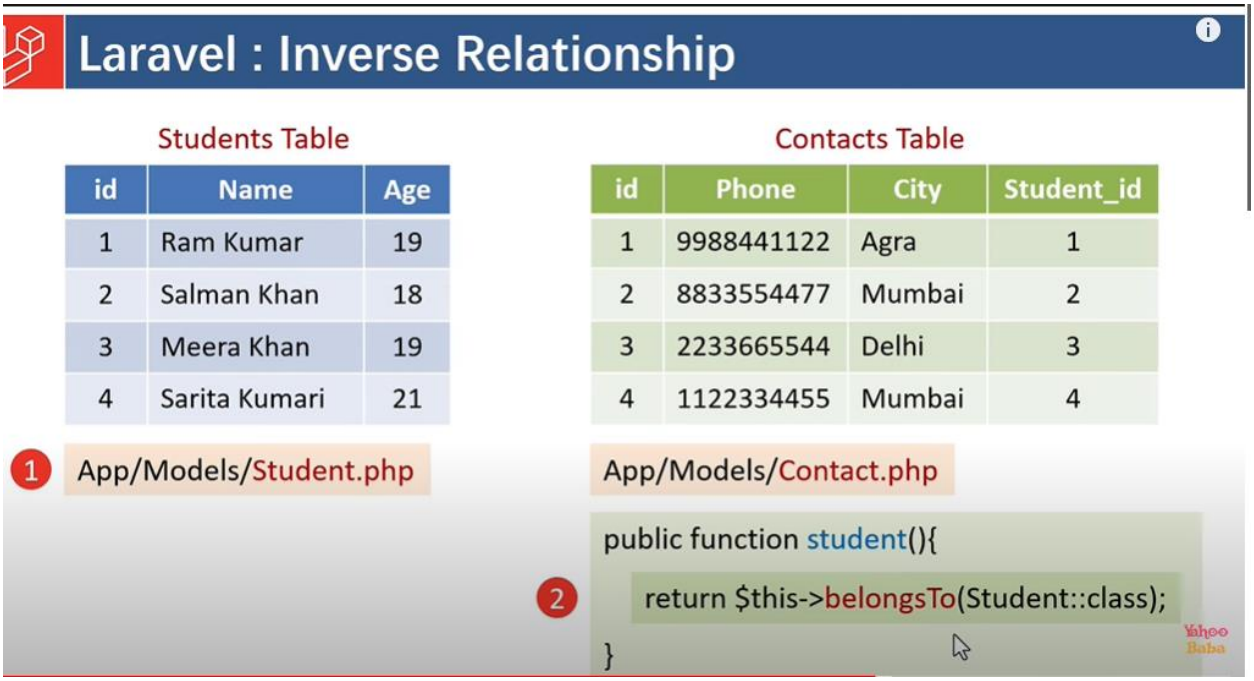
Step-3



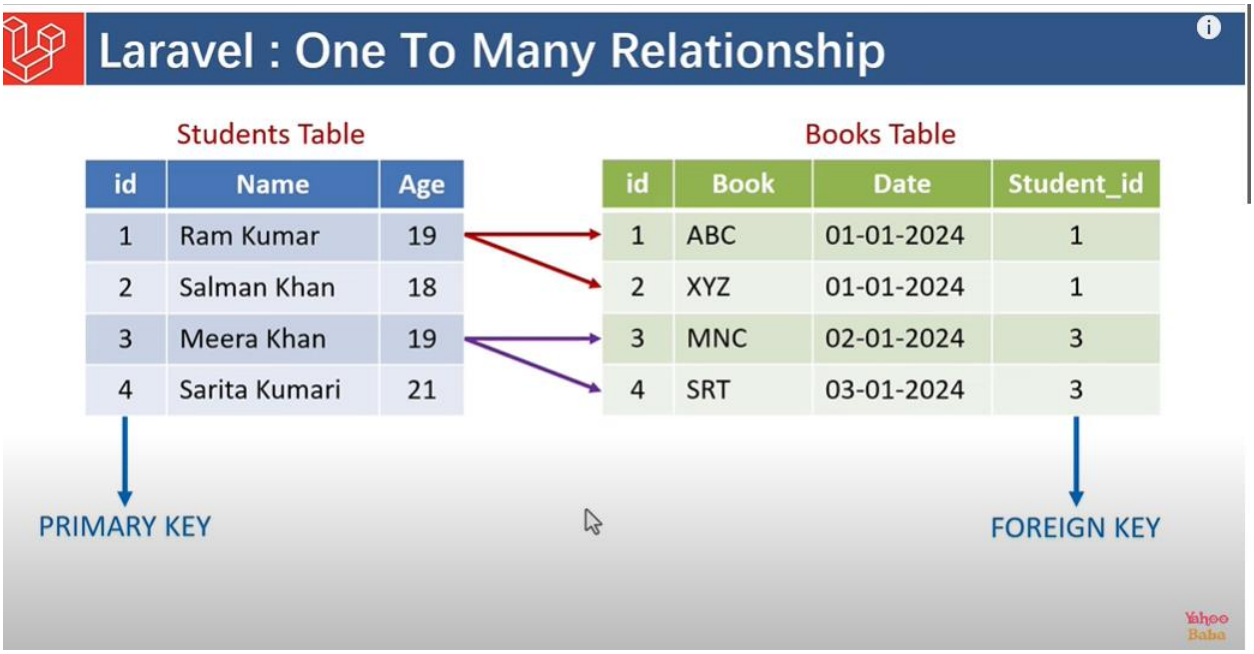
Show the contact table all item



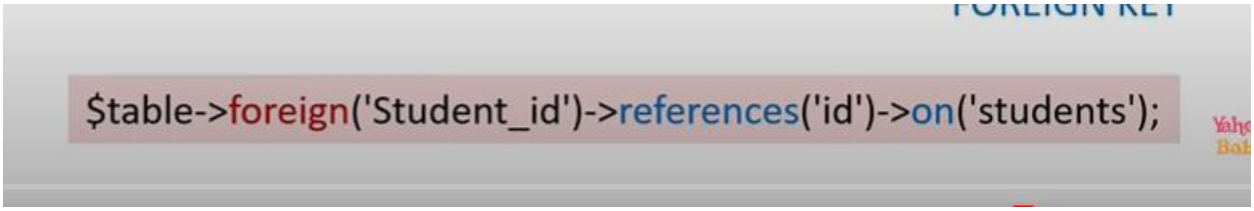
Inverse Relationship



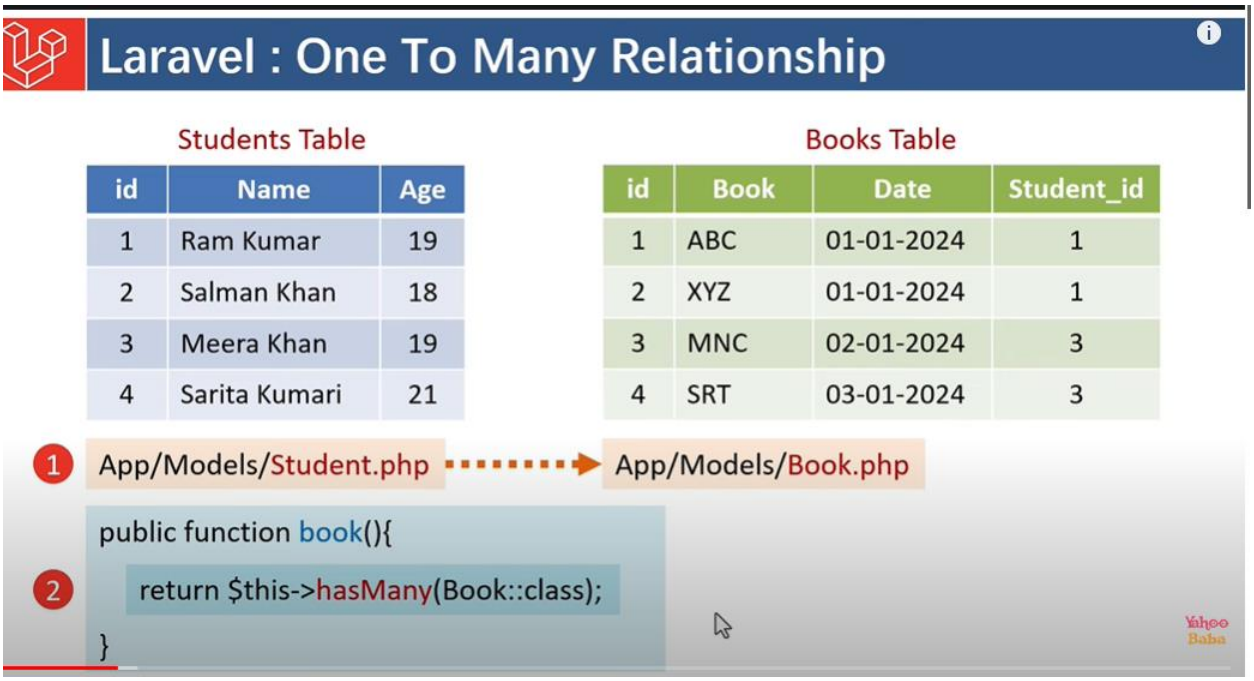
One to Many relationship



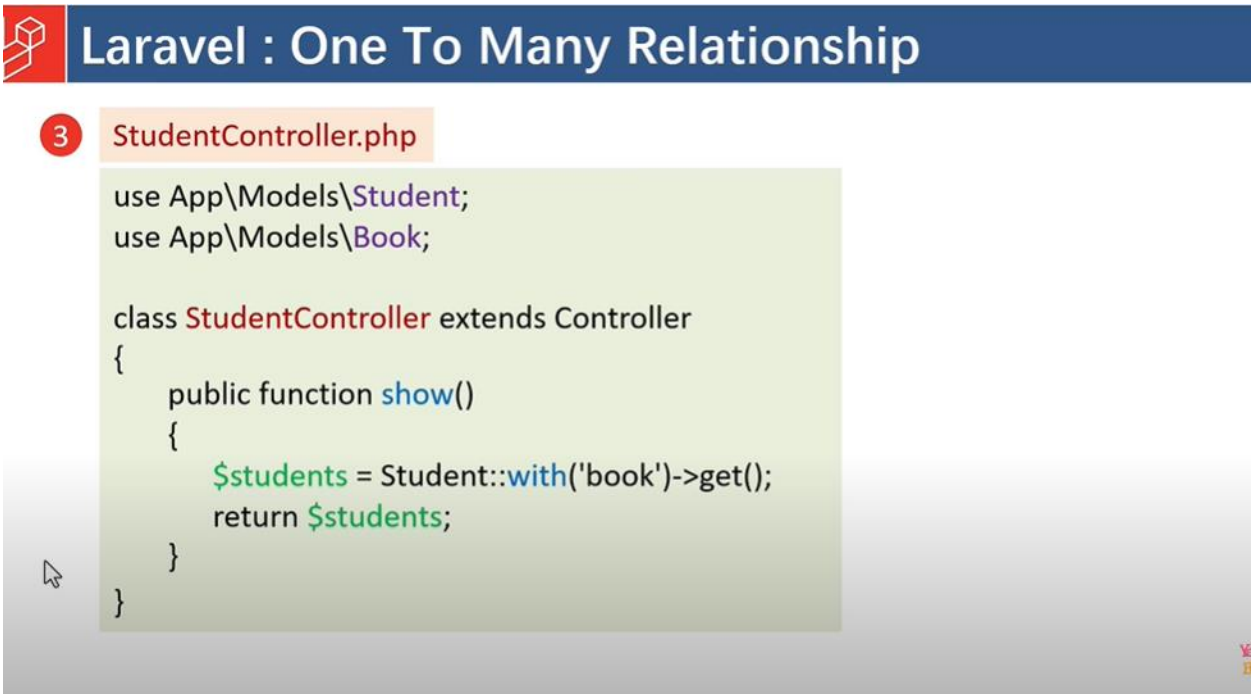
Step-1 foreign key setup



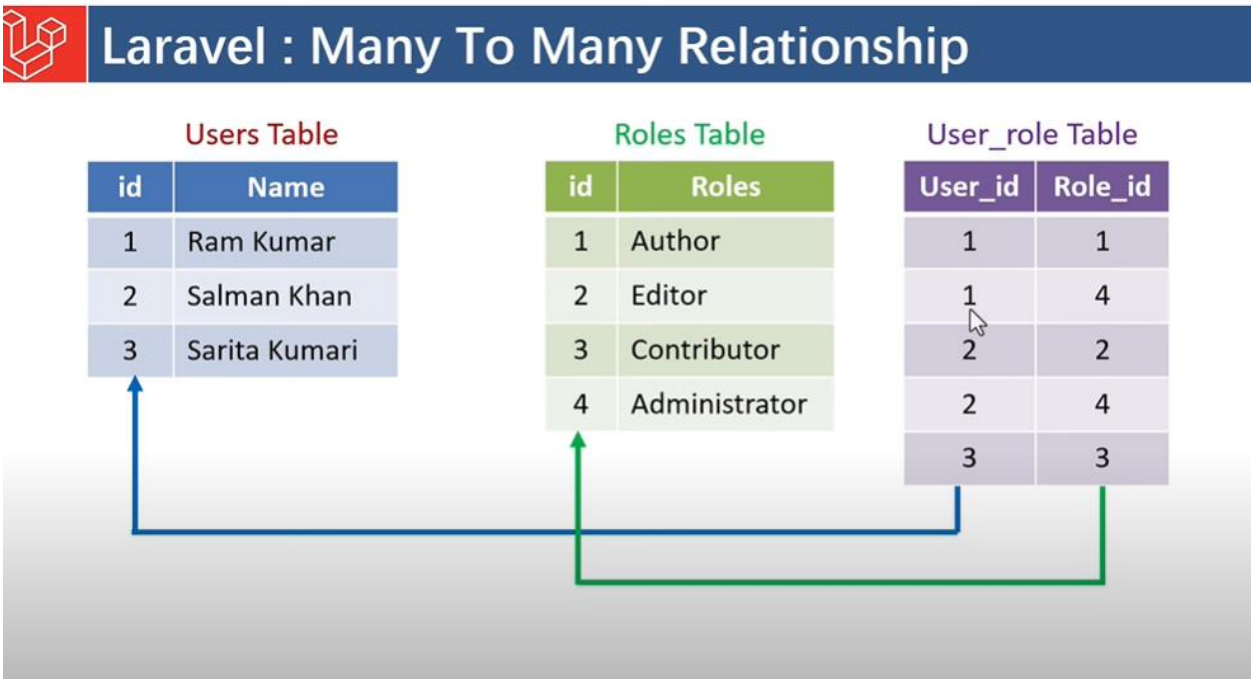
Step 2 :create function in model file



Step-3



Many to Many Relationship



Laravel : Many To Many Relationship

Users Table

id	Name
1	Ram Kumar
2	Salman Khan
3	Sarita Kumari

PRIMARY KEY

Roles Table

id	Roles
1	Author
2	Editor
3	Contributor
4	Administrator

PRIMARY KEY

User_role Table

User_id	Role_id
1	1
1	4
2	2
2	4
3	3

FOREIGN KEY

Laravel : Many To Many Relationship

Users Table

id	Name
1	Ram Kumar
2	Salman Khan
3	Sarita Kumari

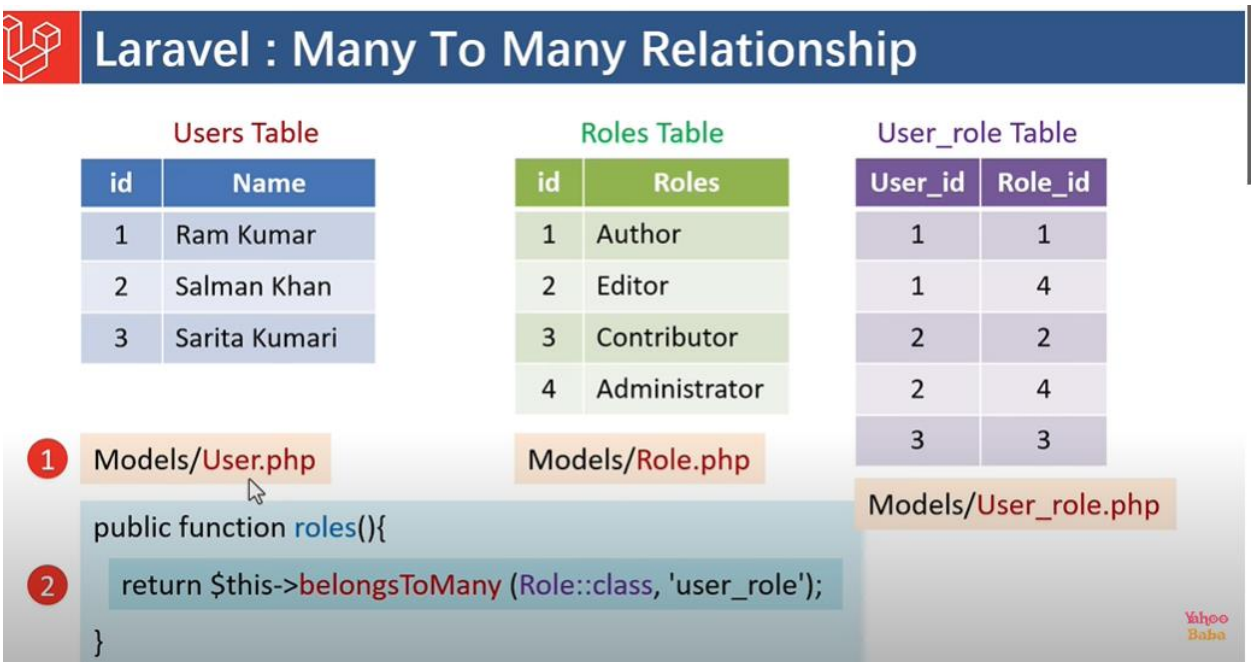
Roles Table

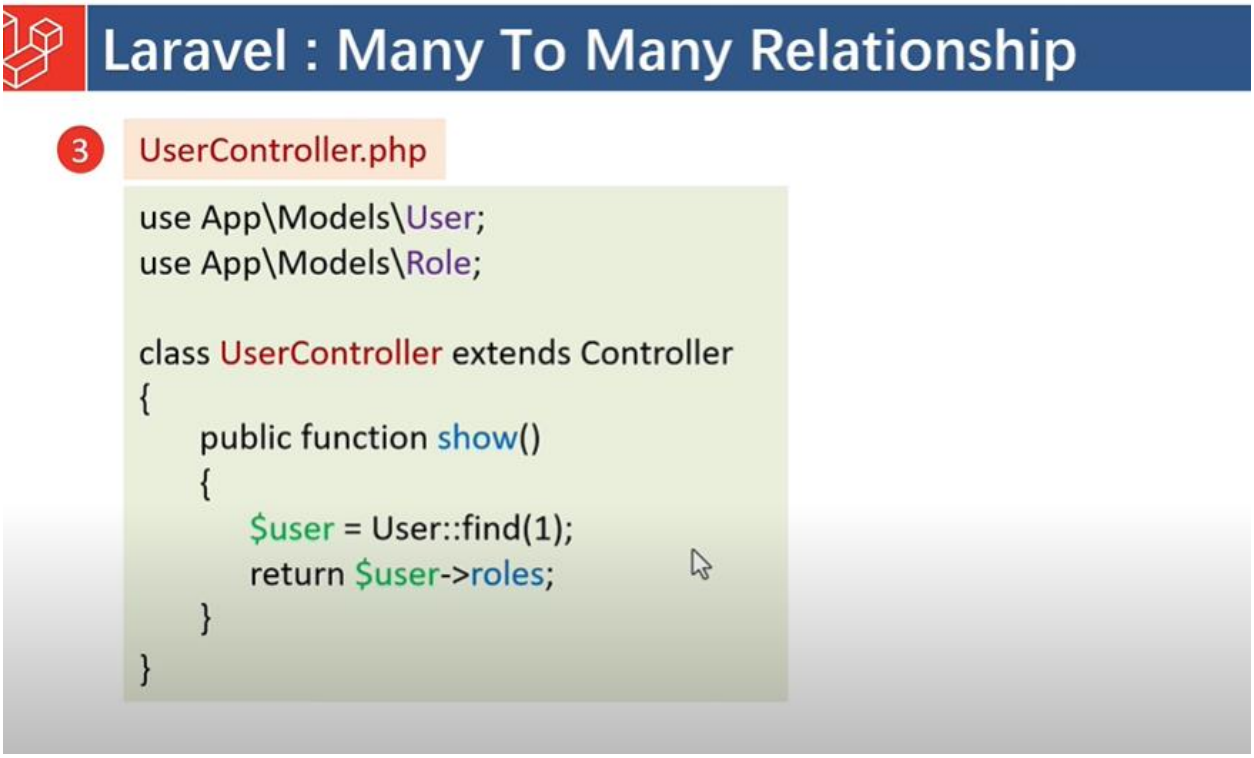
id	Roles
1	Author
2	Editor
3	Contributor
4	Administrator

User_role Table

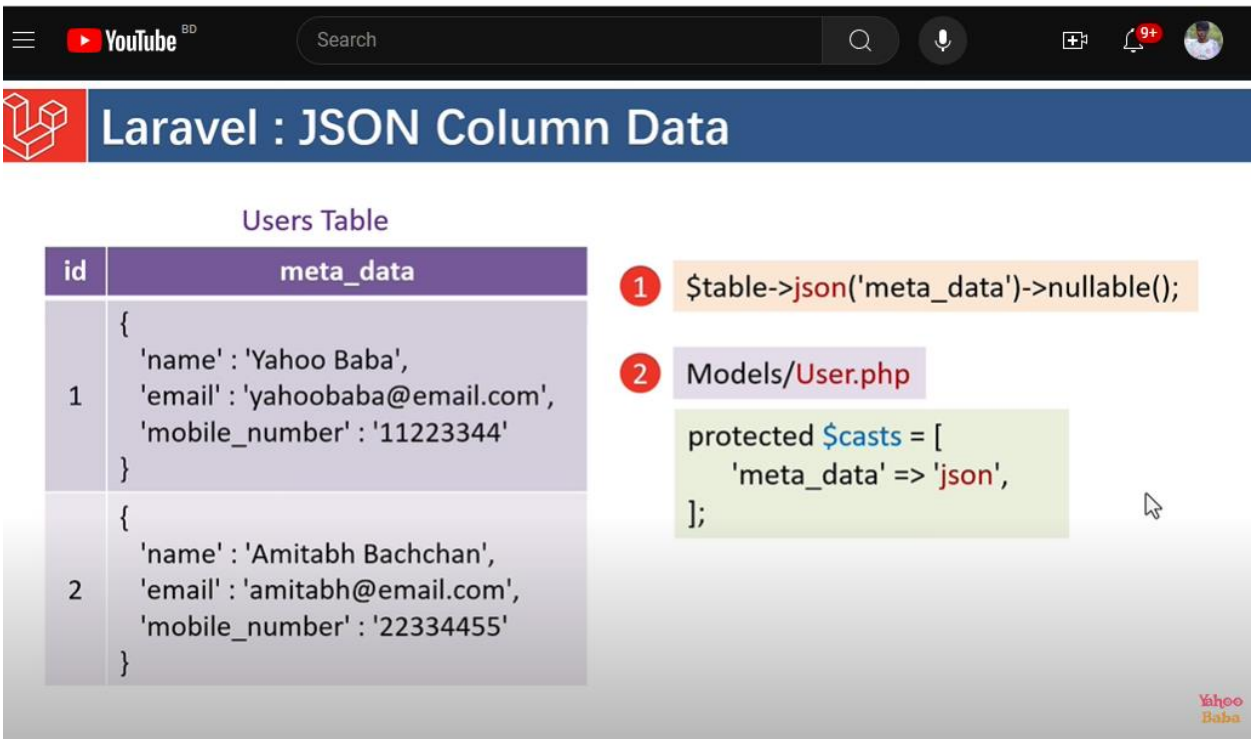
User_id	Role_id
1	1
1	4
2	2
2	4
3	3

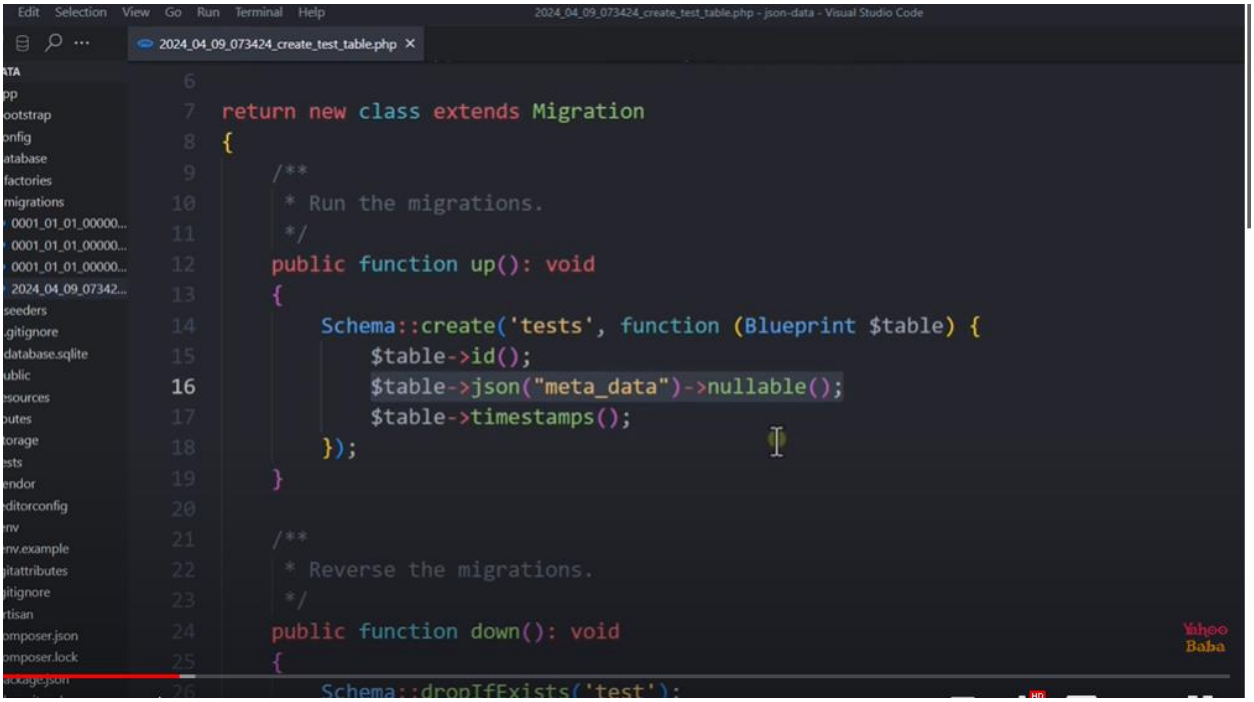
```
$table->foreign('User_id')->references('id')->on('Users');
$table->foreign('Role_id')->references('id')->on('Roles');
```





Laravel Eloquent with json data


YouTube BD
Search
9+
Laravel : JSON Column Data
Users Table
id
meta_data
1
{
'name' : 'Yahoo Baba',
'email' : 'yahoobaba@email.com',
'mobile_number' : '11223344'
}
2
{
'name' : 'Amitabh Bachchan',
'email' : 'amitabh@email.com',
'mobile_number' : '22334455'
}
1 $table->json('meta_data')->nullable();
2 Models/User.php
protected $casts = [
'meta_data' => 'json',
];
Yahoo Baba


Edit Selection View Go Run Terminal Help
2024_04_09_073424_create_test_table.php - json-data - Visual Studio Code
2024_04_09_073424_create_test_table.php
return new class extends Migration
{
/**
* Run the migrations.
*/
public function up(): void
{
Schema::create('tests', function (Blueprint $table) {
$table->id();
$table->json("meta_data")->nullable();
$table->timestamps();
});
}
/**
* Reverse the migrations.
*/
public function down(): void
{
Yahoo Baba

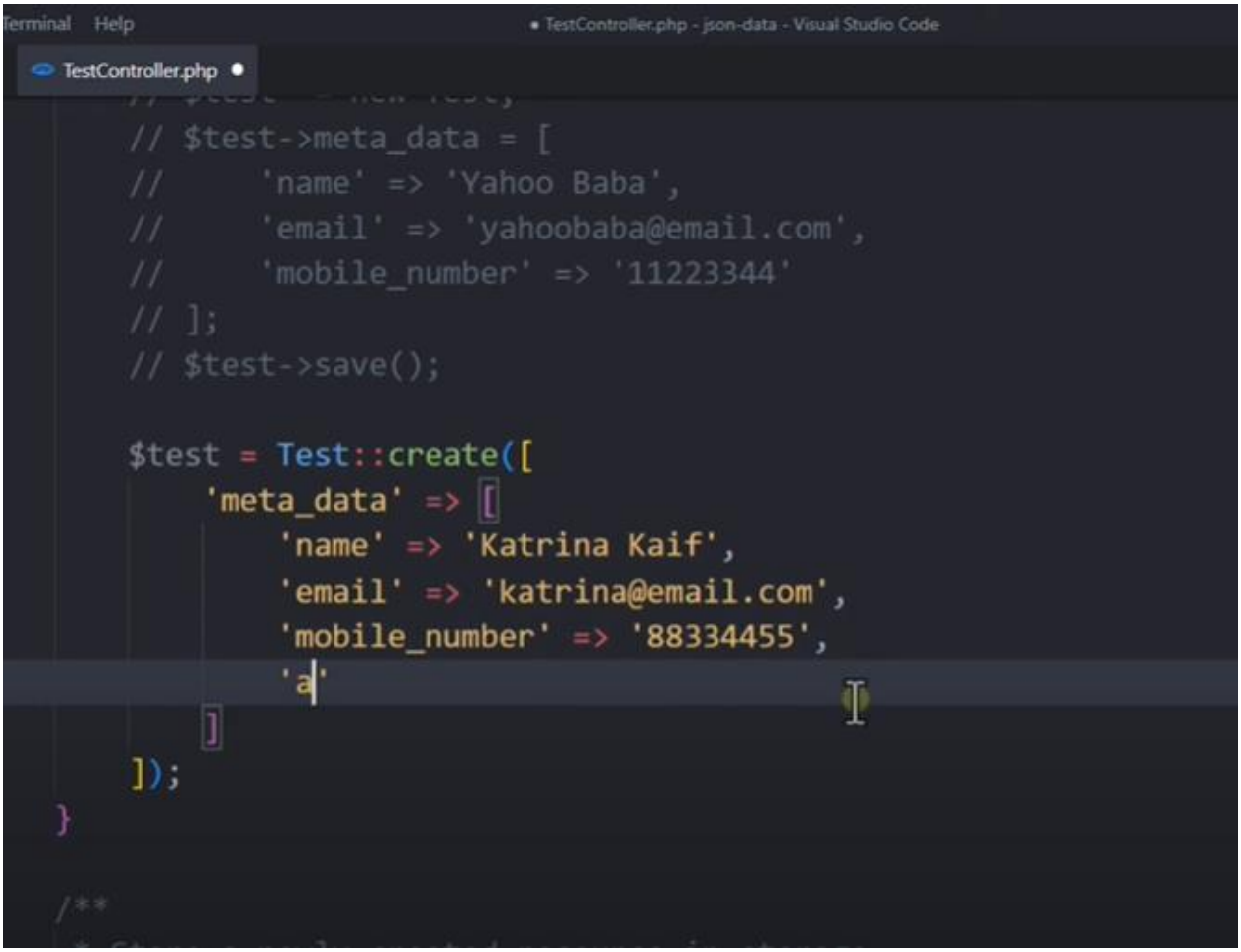
Laravel : JSON Column Data

Users Table

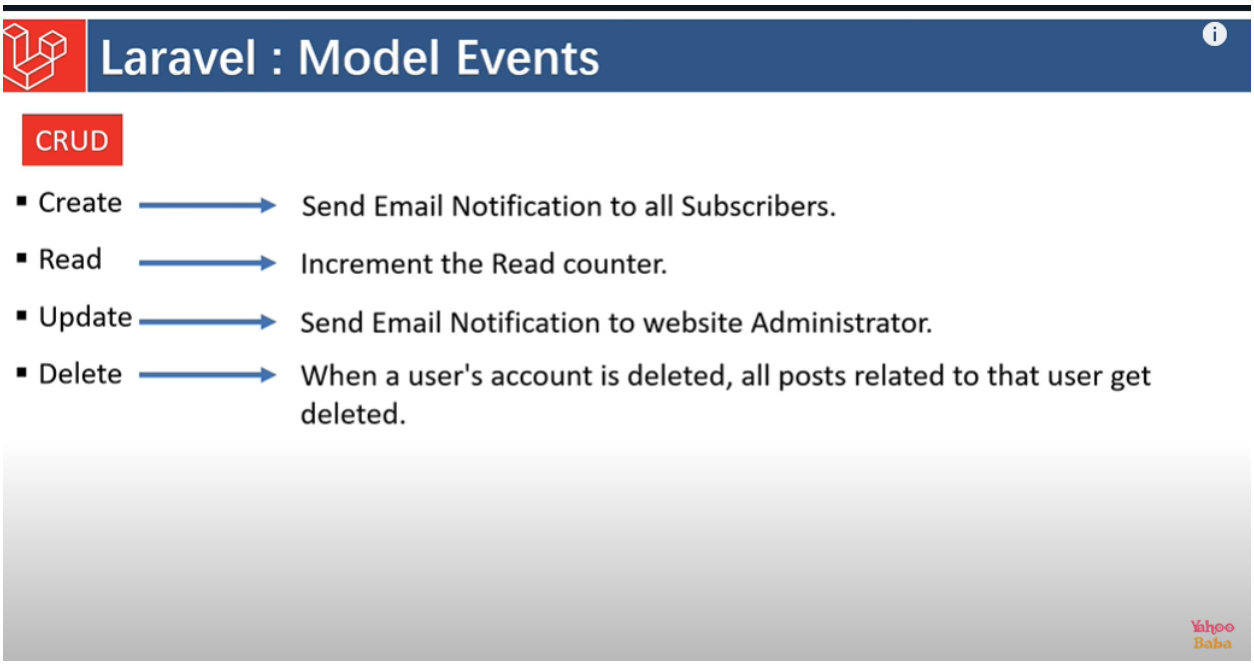
id	meta_data
1	{ 'name' : 'Yahoo Baba', 'email' : 'yahoobaba@email.com', 'mobile_number' : '11223344' }
2	{ 'name' : 'Amitabh Bachchan', 'email' : 'amitabh@email.com', 'mobile_number' : '22334455' }

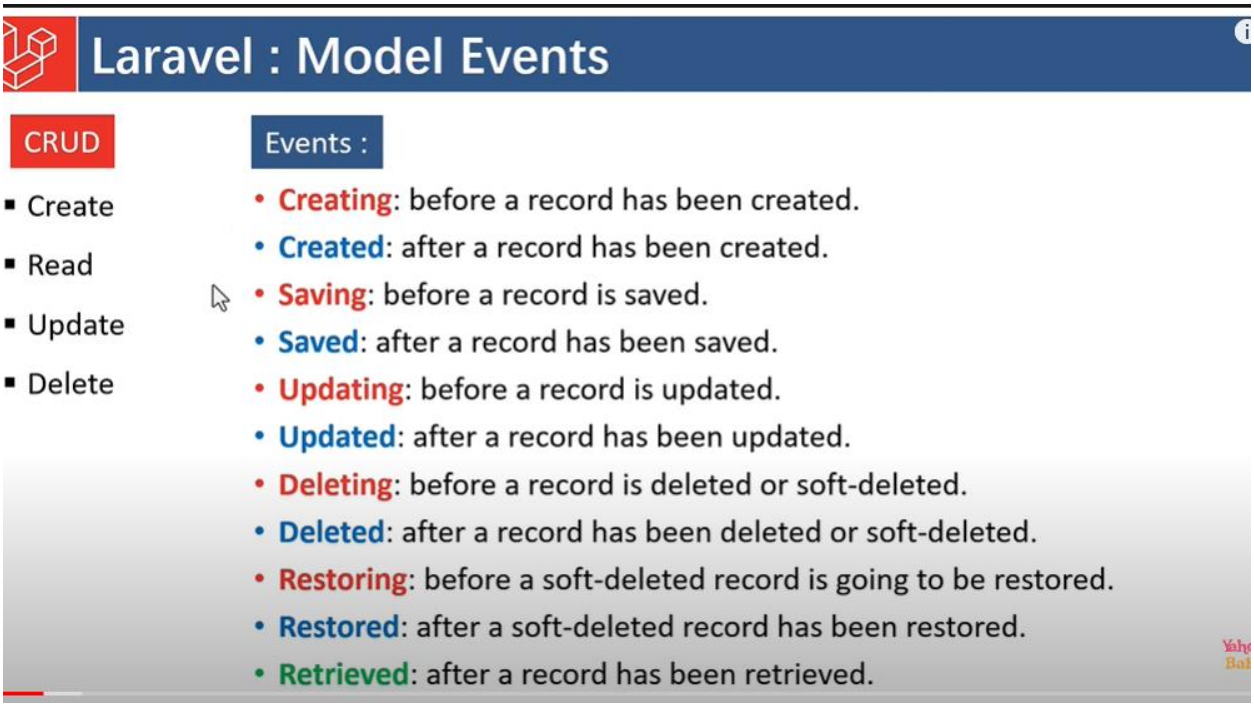
4 UserController.php

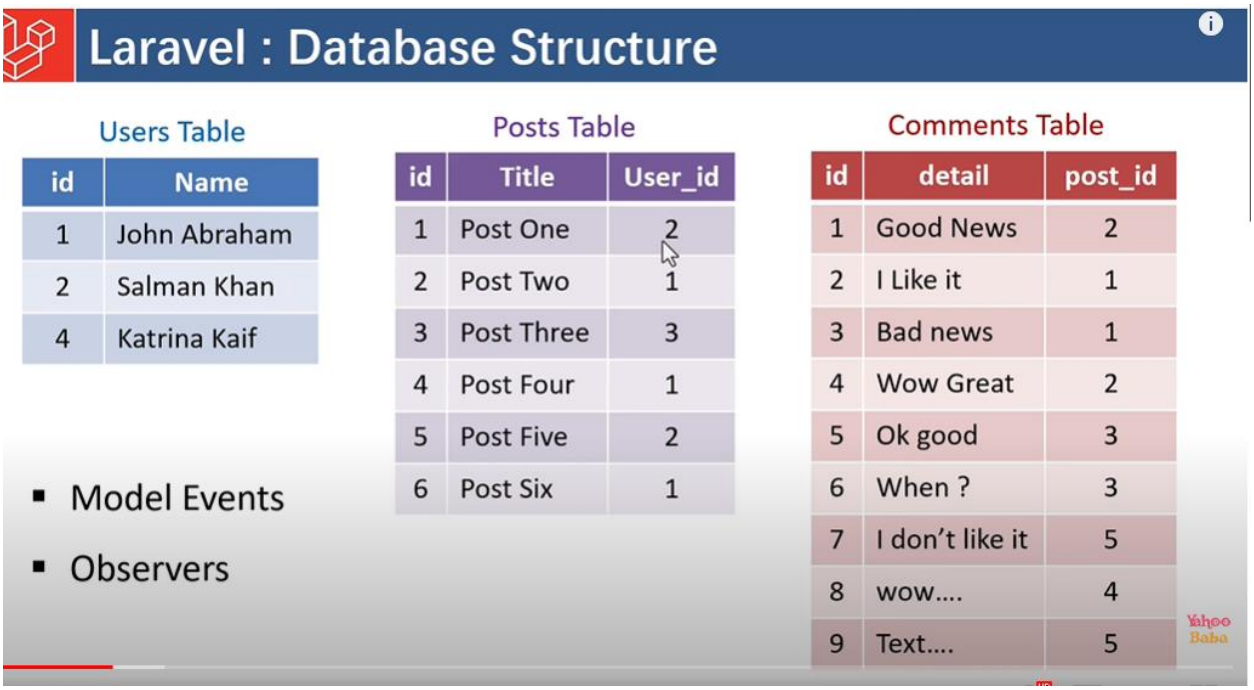
```php
use App\Models\User;

class UserController extends Controller
{
    public function show()
    {
        $user = User::find(1);
        return $user->meta_data;
    }
}
```



Model Event


Laravel : Model Events
CRUD
Create
Send Email Notification to all Subscribers.
Read
Increment the Read counter.
Update
Send Email Notification to website Administrator.
Delete
When a user's account is deleted, all posts related to that user get deleted.
Yahoo Baba


Laravel : Model Events
CRUD
Create
Read
Update
Delete
Events :
Creating: before a record has been created.
Created: after a record has been created.
Saving: before a record is saved.
Saved: after a record has been saved.
Updating: before a record is updated.
Updated: after a record has been updated.
Deleting: before a record is deleted or soft-deleted.
Deleted: after a record has been deleted or soft-deleted.
Restoring: before a soft-deleted record is going to be restored.
Restored: after a soft-deleted record has been restored.
Retrieved: after a record has been retrieved.



Note :users table ekta name delete korle jotogulo post table and comments table er sokol kisu delete hoye jabe

Model event :model event using only small event handling

Laravel : Model Events

1. App/Models/Post.php

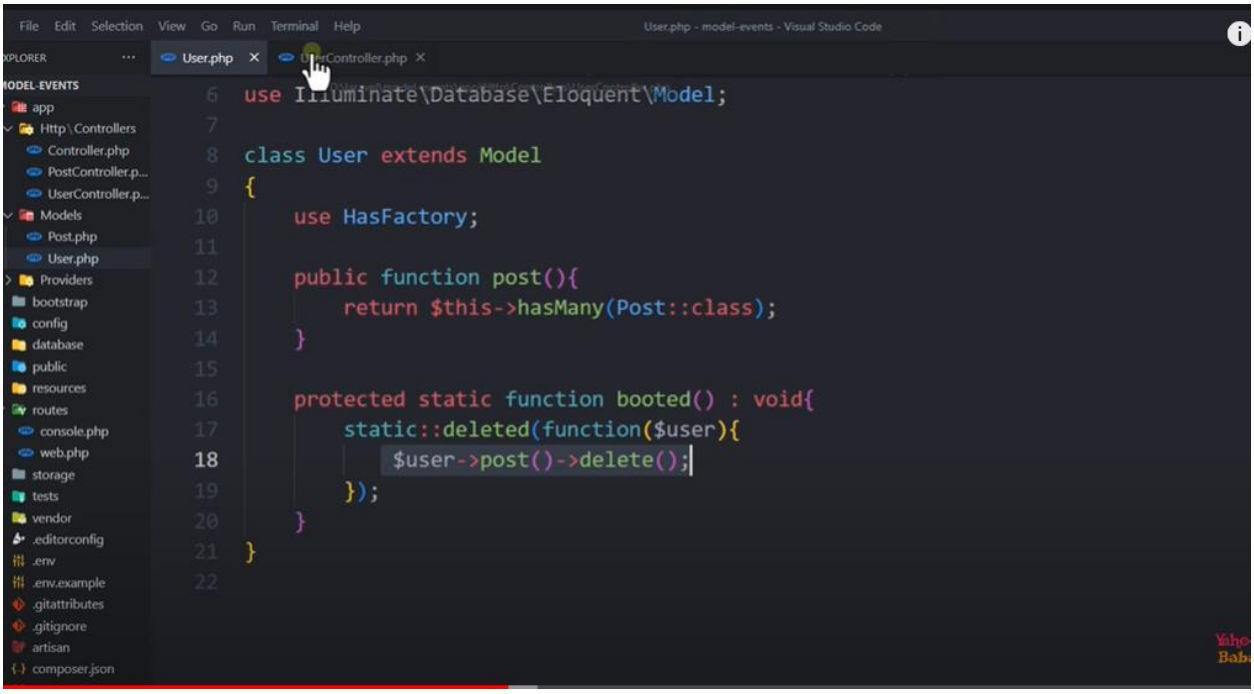
2.
```
protected static function booted(): void {

    static::created(function($post) {

    });

    static::deleted(function($post) {

    });

}
```

- Creating
- Created
- Saving
- Saved
- Updating
- Updated
- Deleting
- Deleted
- Restoring
- Restored
- Retrieved

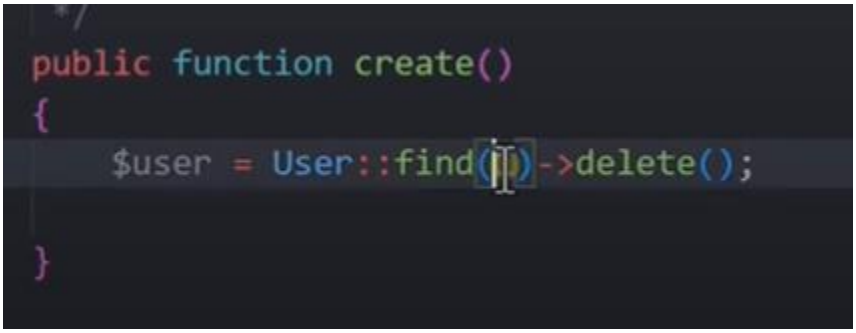
Normal rule : controller working

```
*/
public function create()
{
    $user = User::find(1)->delete();
    Post::where('user_id',1)->delete();
}

/**
```

Step-1

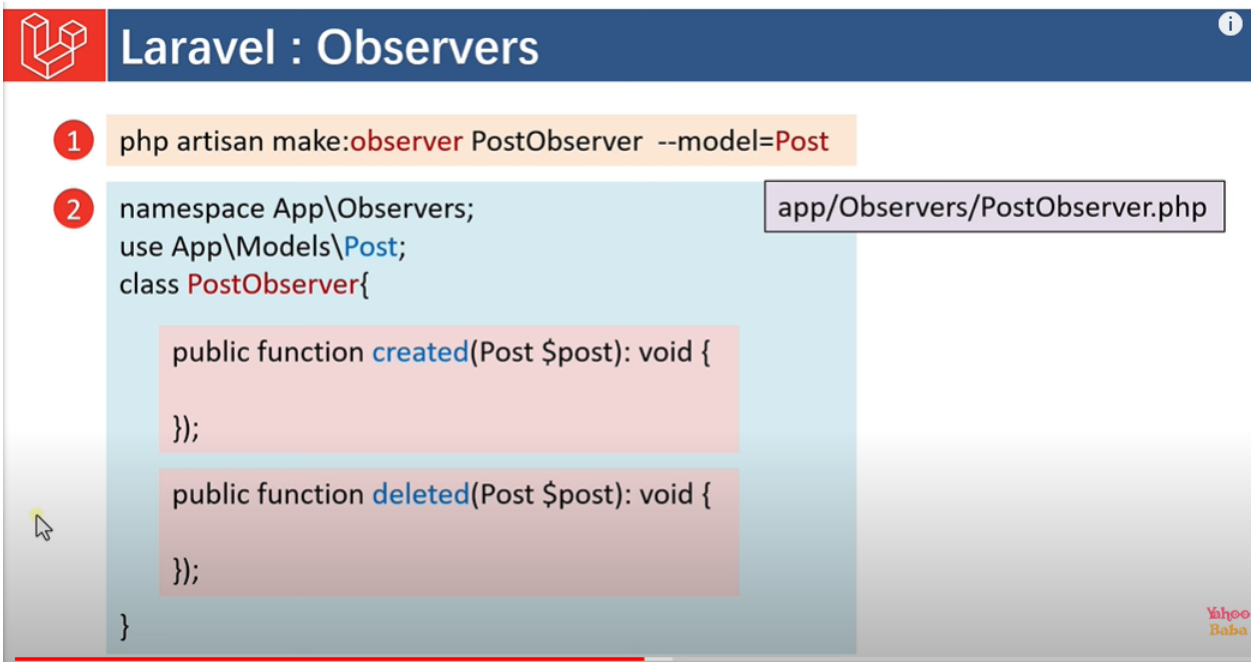


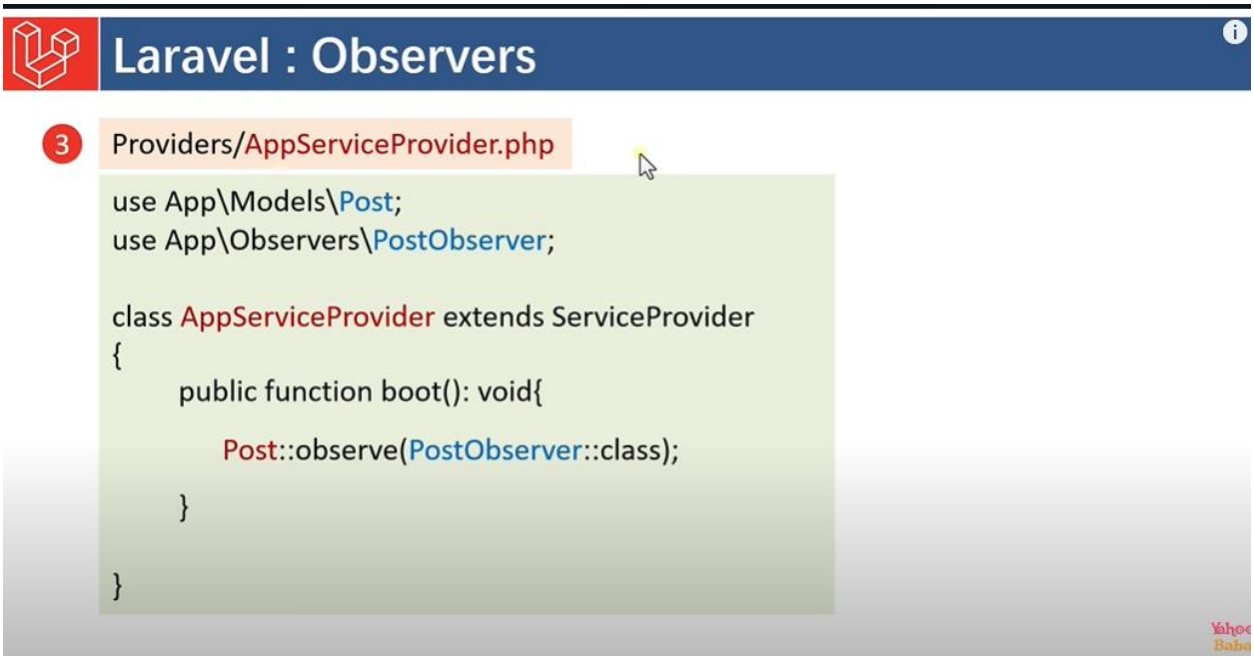
Step-2:



Laravel Observers

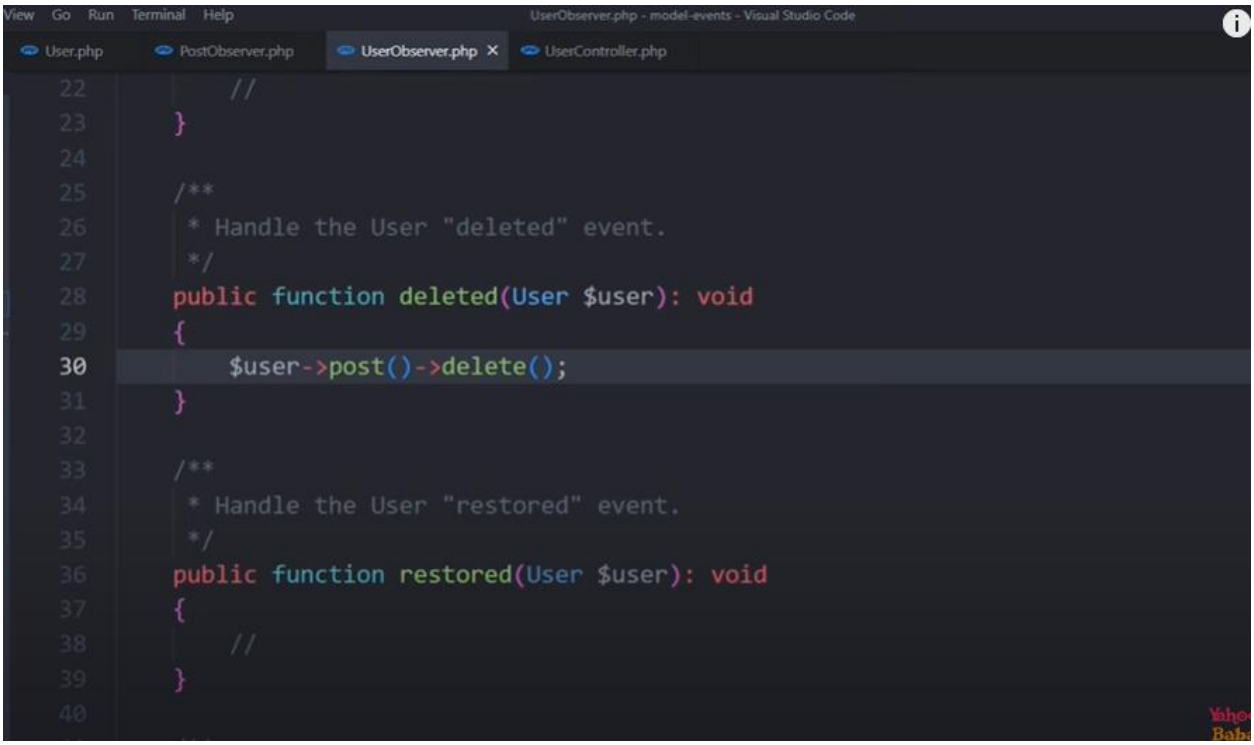
Observer handling using large event handling



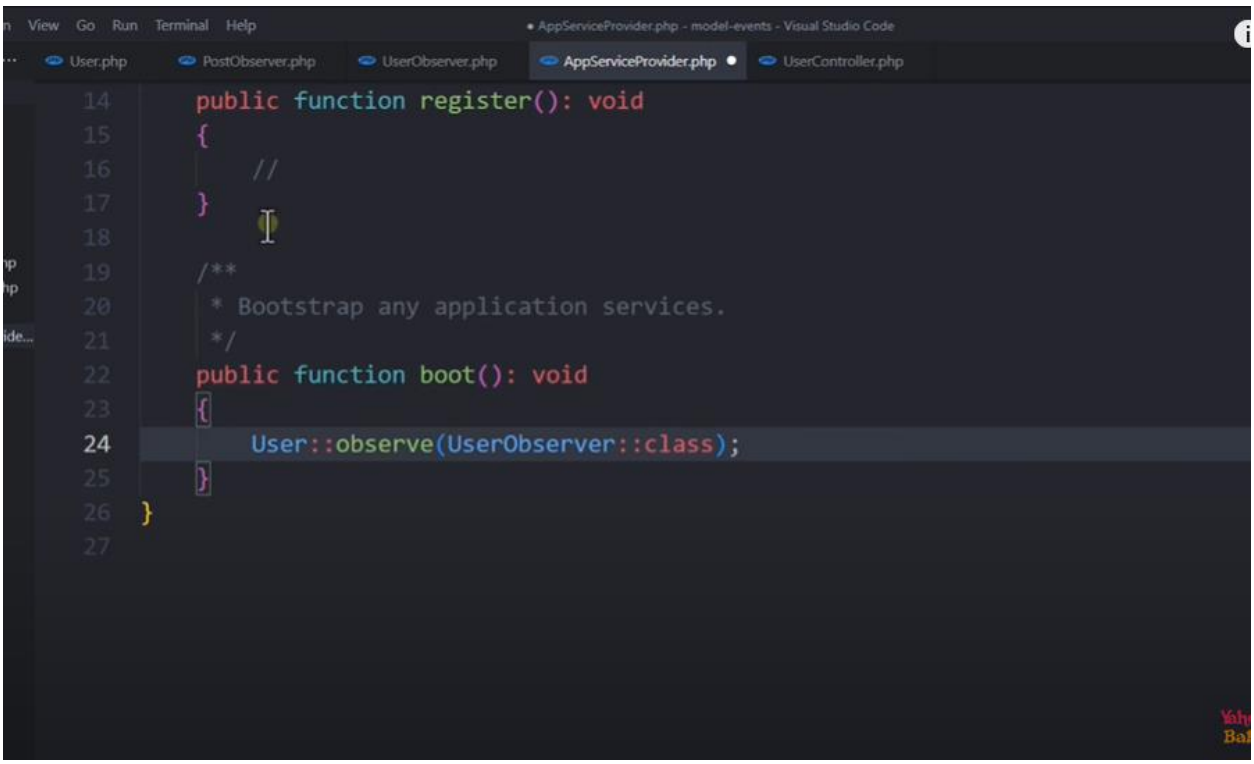


Step 1: create observer file

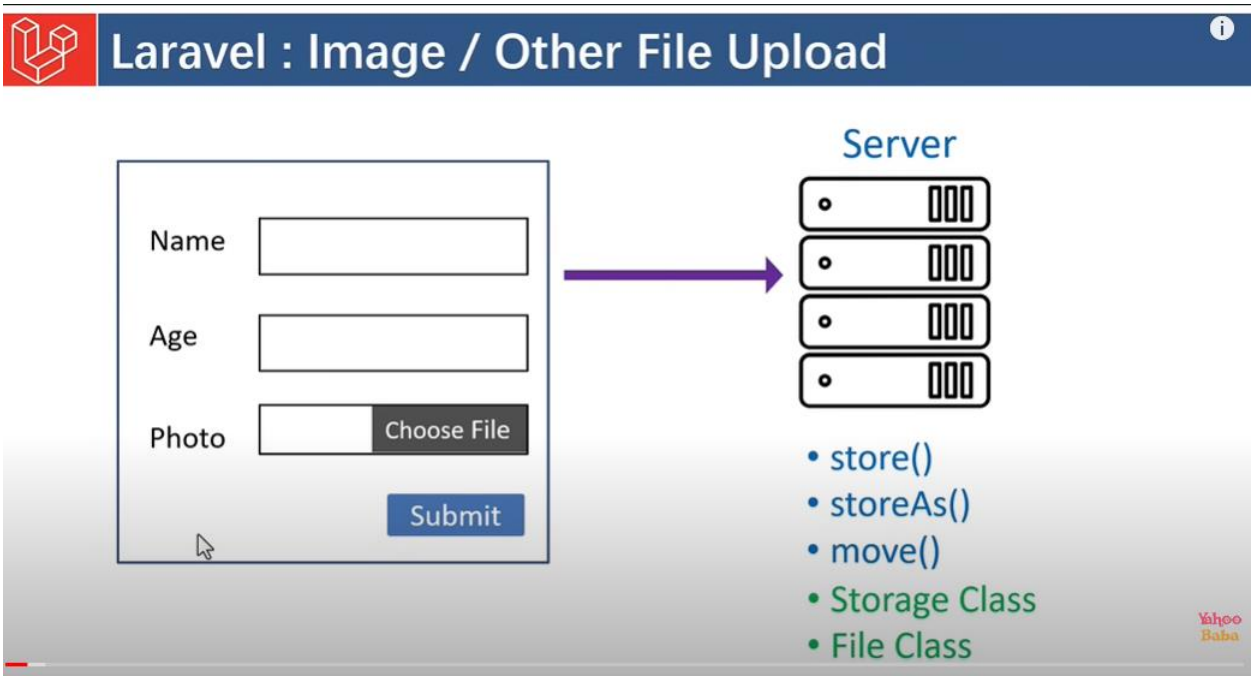
Step 2:

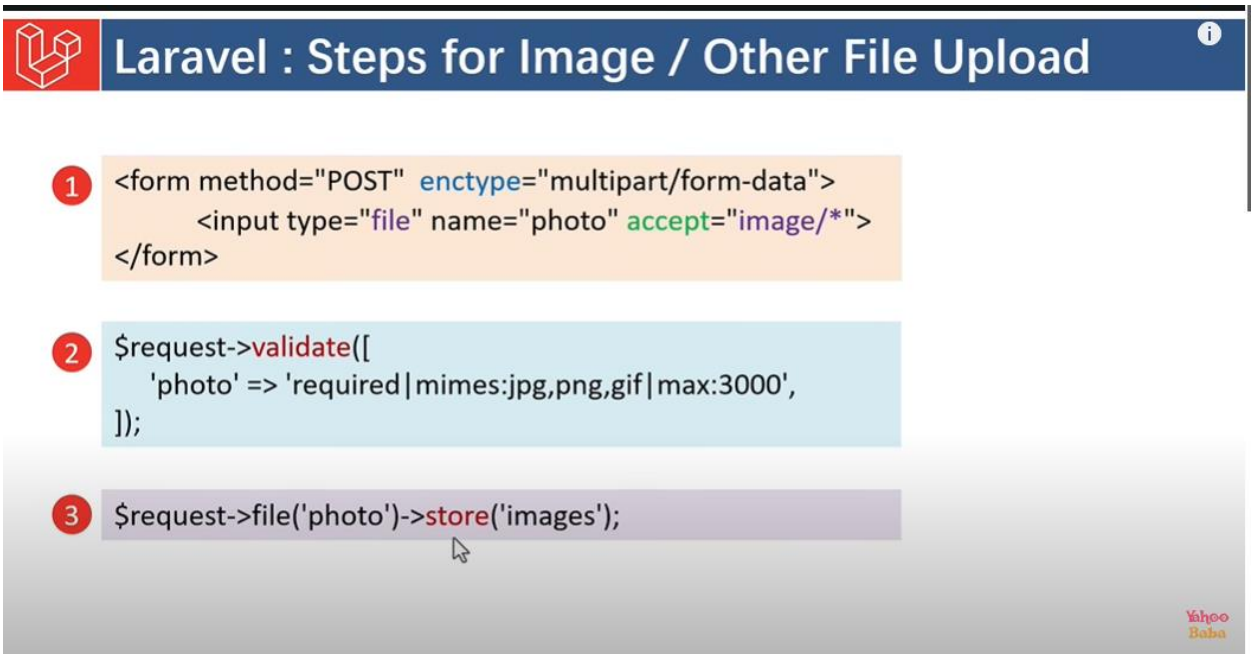

View Go Run Terminal Help
UserObserver.php - model-events - Visual Studio Code
User.php
PostObserver.php
UserObserver.php
UserController.php
//
}
/**
* Handle the User "deleted" event.
*/
public function deleted(User $user): void
{
$user->post()->delete();
}
/**
* Handle the User "restored" event.
*/
public function restored(User $user): void
{
//
}

Step-3:


View Go Run Terminal Help
AppServiceProvider.php - model-events - Visual Studio Code
User.php
PostObserver.php
UserObserver.php
AppServiceProvider.php
UserController.php
public function register(): void
{
//
}
/**
* Bootstrap any application services.
*/
public function boot(): void
{
User::observe(UserObserver::class);
}
}

Laravel Image & File Upload





Controller

Database link add (all time using)

```php
$request->validate([
    'photo' => 'required|mimes:png,jpg,jpeg|max:3000'
]);

$file = $request->file('photo');

$path = $request->photo->store('image','public');
// return $path;

User::create([
    'filename' => $path,
]);

return redirect()->route('user.index')->with('status','User Image Uploded Successfully.');
```

Without database link add

```php
    */
    public function store(Request $request)
    {
        $file = $request->file('photo');

        $request->validate([
            'photo' => 'required|mimes:png,jpg,jpeg|max:3000'
        ]);

        $path = $request->file('photo')->store('image','public');

        return redirect()->route('user.index')->with('status','User Image Uploded Successfully.');
    }

    /**
```

Other away

```php
$file = $request->file('photo');

$request->validate([
    'photo' => 'required|mimes:png,jpg,jpeg|max:3000'
]);

// $path = $request->photo->store('image','public');
$fileName = time(). '_'.$file->getClientOriginalName();
$path = $request->photo->storeAs('image',$fileName,'public');
return $path;

return redirect()->route('user.index')->with('status','User Image Uploded Successfully.');
```

Success status

```
<div class="row">
    <div class="col-6">
        @if (session('status'))
            <div class="alert alert-success">
                {{ session('status') }}
            </div>
        @endif
    </div>
</div>
```

Image read from database

```
@foreach ($users as $user)
    <div class="col-2">
        <img  src="{{ asset('/storage/'. $user->file_name) }}" alt="">
    </div>
@endforeach
```

Delete image:

```
public function destroy(string $id)
{
    $user = User::find($id);

    $user->delete();

    $image_path = public_path("storage/") . $user->file_name;

    if(file_exists($image_path)){
        @unlink($image_path);
    }


}
```

```blade
<form action="{{ route('user.destroy', $user->id) }}" method="POST">
    @csrf
    @method('DELETE')
    <button type="submit" class="btn btn-danger btn-sm mb-3">Delete</button>
</form>
```

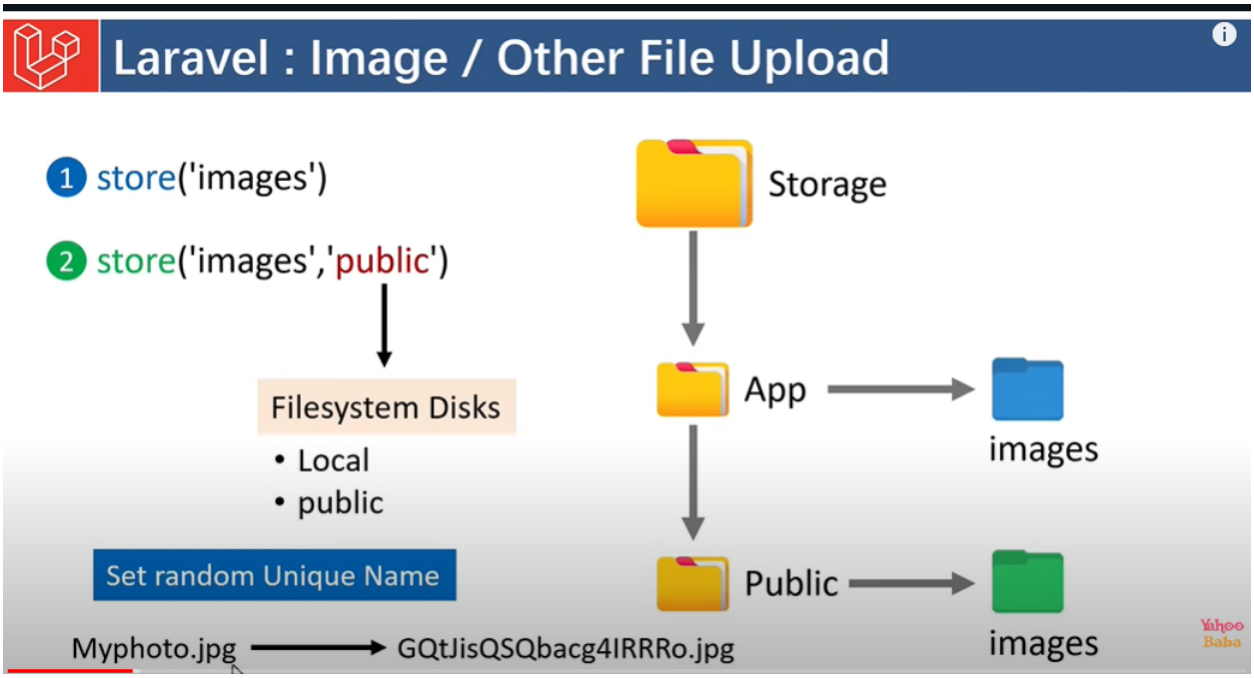
Update :

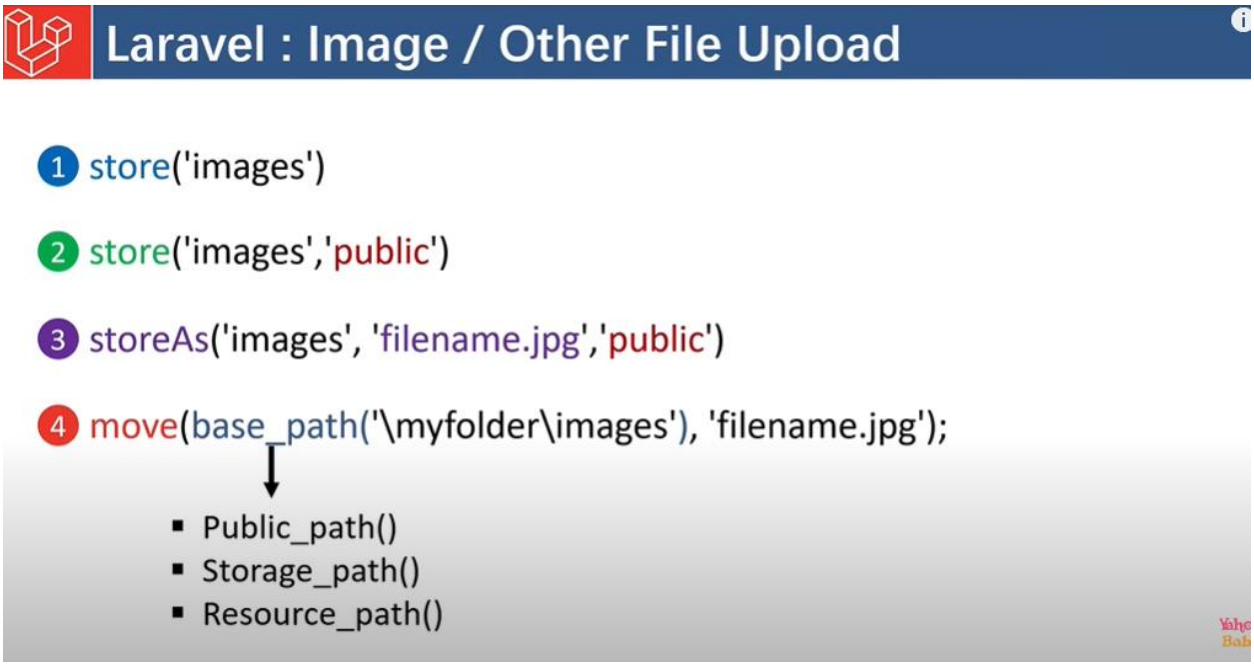
```php
public function update(Request $request, string $id)
{
    $user = User::find($id);

    if($request->hasFile('photo')){

        $path = $request->photo->store('image','public');

        $user->file_name = $path;
        $user->save();

        return redirect()->route('user.index')
    ->with('status','User Image Updated Successfully.');
    }
}
```

```php
public function update(Request $request, string $id)
{
    $user = User::find($id);

    if($request->hasFile('photo')){

        $image_path = public_path("storage/") . $user->file_name;

        if(file_exists($image_path)){
            @unlink($image_path);
        }

        $path = $request->photo->store('image','public');

        $user->file_name = $path;
        $user->save();

        return redirect()->route('user.index')
    ->with('status','User Image Updated Successfully.');
```

```
Edit  Selection  View  Go  Run  Terminal  Help        file-update.blade.php - file-upload - Visual Studio Code
upload.blade.php   UserController.php   file-update.blade.php ×   web.php
<form action="{{ route('user.update', $user->id) }}" method="POST" enctype="multipart/form
    @csrf
    @method('PUT')
    <div class="row">
        <div class="col-3">
            <img id="output" src="{{ asset('/storage/'. $user->file_name) }}" alt="">
        </div>
        <div class="col-9">
          <input type="file" name="photo" accept=".jpg,.png,.jpeg">
            @error('photo')
                <div class="alert alert-danger mb-1 mt-1">{{ $message }}</div>
            @enderror
        </div>
        <div class="col-12">
            <input type="submit" class="btn btn-sm btn-primary">
        </div>
    </div>
```

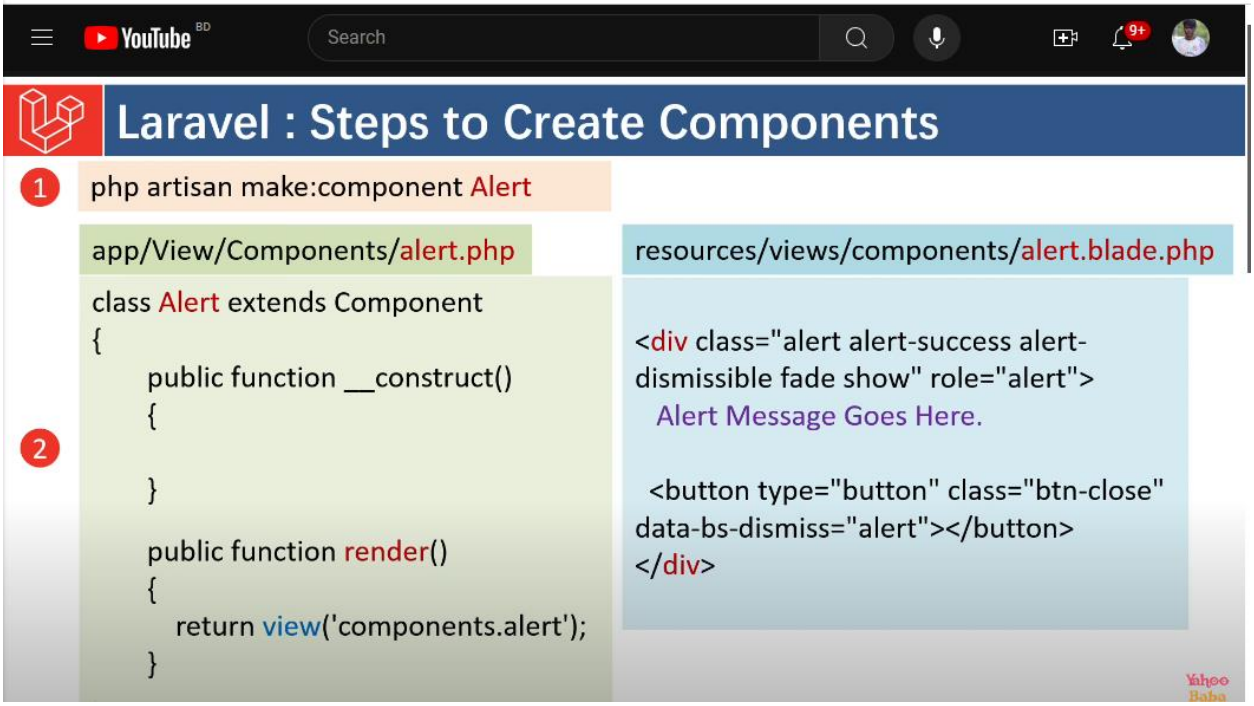
```
class="col-9">
put type="file" onchange="document.querySelector('#output').src=window.URL.createObjectURL(this.files[0])" nam
error('photo')
   <div class="alert alert-danger mb-1 mt-1">{{ $message }}</div>
```

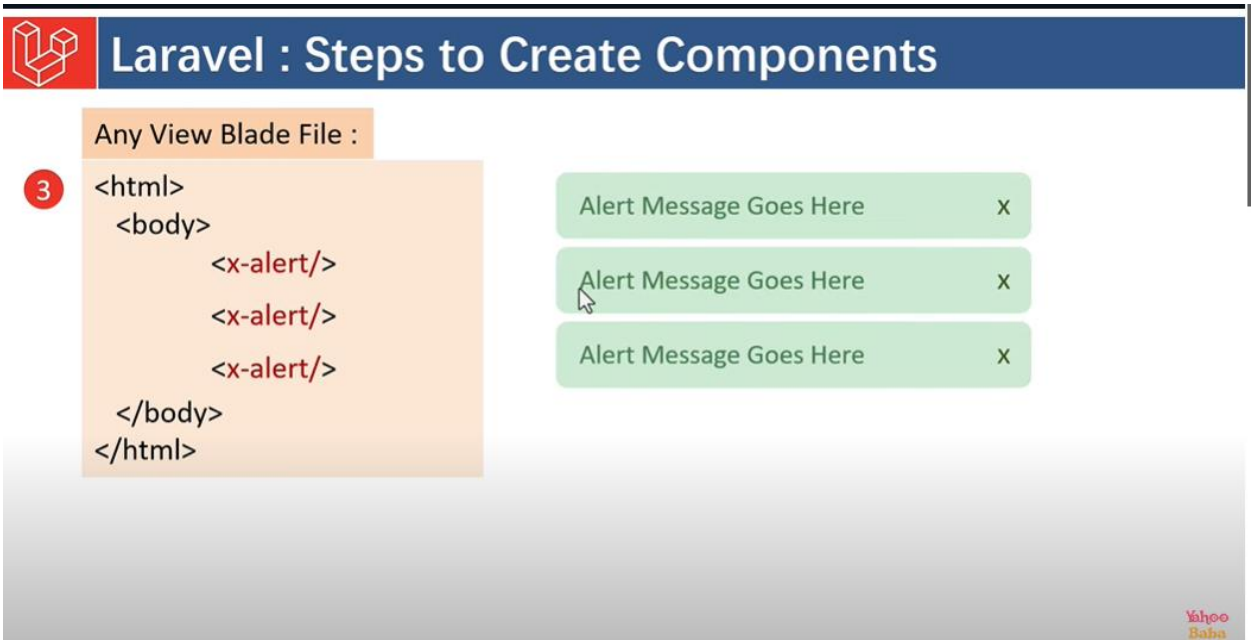




Laravel Components

Static components





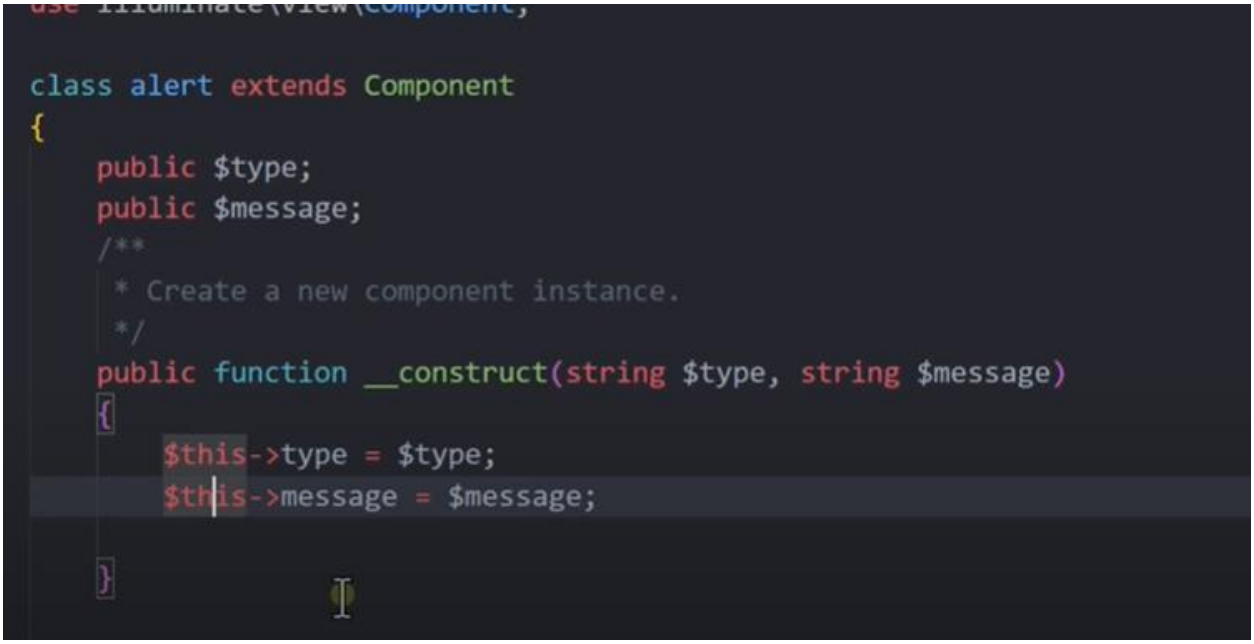
Note: x-component name dite hobe

Dynamic components

Step-1

```
<body>
    <x-alert type="success" message="This is success message alert." />
    <x-alert type="error" message="This is error message alert."/>
    <x-alert type="info" message="This is info message alert." />
</body>
```

Step-2



```php
class alert extends Component
{
    public $type;
    public $message;
    /**
     * Create a new component instance.
     */
    public function __construct(string $type, string $message)
    {
        $this->type = $type;
        $this->message = $message;

    }
```

Step-3

alert.blade.php • | welcome.blade.php | alert.php

```blade
<div class="alert alert-{{ $type }}" role="alert">
    {{ $message }}
</div>
```

Other

Passing data

```blade
<!-- Use the ProductList component and pass the products -->
<x-product-list :products="$products" />
```

View data

blade Copy code

```blade
<ul>
    @foreach ($products as $product)
        <li>{{ $product->name }} - ${{ $product->price }}</li>
    @endforeach
</ul>
```

Class components

```php
class ProductList extends Component
{
    public $products;

    // Accept products as a parameter
    public function __construct($products)
    {
        $this->products = $products;
    }

    public function render()
    {
        // Render the Blade component view
        return view('components.product-list');
    }
}
```

Slots components

Note :html tag passing using slot components

Laravel : Slots

1. php artisan make:component Alert

2.
```
<div class="alert alert-success alert-dismissible fade show" role="alert">
  {{ $slot }}
  <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
</div>
```

3.
```
<x-alert>
    <h4>Heading Goes Here</h4>
    <hr/>
    <p>Alert Message Goes Here</p>
</x-alert>
```

Yahoo Baba