
What is Composer in PHP?

If you've been coding in PHP for some time, you'll be aware of how **PHP libraries can help save work** and make code **reusable**. In the past, it was **harder to add libraries to PHP**.

That's where a dependency manager like **Composer comes in**. In fact, before Composer, there was a popular tool called **PEAR which was used to manage PHP extensions and libraries. But it had its own limitations**, which Composer was created to address.

Composer is a **dependency manager for PHP**.

You can simply run **composer update** to get the **latest compatible packages**.



Ex

❖ To ensure that the composer is successfully installed:

\$ composer

❖ To install the package:

\$ composer require phpmailer



```
<?php  
$age = 23;  
  
switch(true){  
    case ($age >= 15 && $age <= 20):  
        echo "You are eligible.";  
        break;  
  
    case ($age >= 21 && $age <= 30):  
        echo "You are not eligible.";  
        break;  
    default:  
        echo "Enter the valid age.";  
        break;  
}  
  
?>
```

You are not eligible.

PHP Switch Example-

```
<?php
$num=20;
switch($num){
case 10:
echo("number is equals to 10");
break;
case 20:
echo("number is equal to 20");
break;
case 30:
echo("number is equal to 30");
break;
default:
echo("number is not equal to 10, 20 or 30");
}
?>
```

```
<?php  
$day = "Monday";  
  
switch ($day) {  
    case "Monday":  
        echo "Today is Monday";  
        break;  
    case "Tuesday":  
        echo "Today is Tuesday";  
        break;  
    case "Wednesday":  
        echo "Today is Wednesday";  
        break;  
    case "Thursday":  
        echo "Today is Thursday";  
        break;  
    case "Friday":  
        echo "Today is Friday";  
        break;  
    case "Saturday":  
        echo "Today is Saturday";  
        break;  
    case "Sunday":  
        echo "Today is Sunday";  
        break;  
    default:  
        echo "Not a valid day";  
}  
?>
```



```

<?php

$score = 85;

switch (true) {
    case ($score >= 90):
        echo "You got an A";
        break;
    case ($score >= 80 && $score < 90):
        echo "You got a B";
        break;
    case ($score >= 70 && $score < 80):
        echo "You got a C";
        break;
    case ($score >= 60 && $score < 70):
        echo "You got a D";
        break;
    default:
        echo "You got an F";
}

?>
180
181 add_action('woocommerce_single_product_summary', 'fn_custom_block', 100);
182 function fn_custom_block() {
183     echo '<div class="custom-block">';
184     echo do_shortcode('[contact-form-7 id="4772" title="Inquiry Form"]');
185     echo '</div>'; | I
186 }

```

The screenshot shows the WordPress dashboard with the UpdraftPlus plugin active. The left sidebar is filled with various theme options and plugin settings. The main content area is titled 'Sidebar Settings' under 'Shop Catalog Sidebar'. It includes sections for 'Sidebar Layout' (with three grid icons), 'Shop Catalog Sidebar' (containing 'Shop Products [shop_products]'), 'Shop Sidebar Width' (set to 'Main Sidebar [sidebar_main-sidebar]'), 'Sticky Sidebar On?' (unchecked), 'Sidebar Side Gap' (set to 'Shop Filter [shop_filter]'), 'Shop Column' (set to '1'), 'Products per page' (set to 'Shop Single [shop_single]'), and 'Use Animation Shop?' (set to 'Off'). At the bottom right are buttons for 'Save Changes', 'Reset Section', and 'Reset All'.



How to code in MVC Framework ?

Controller (controllers/my_controller.php)

```
function total_frogs(){
    $this->load->model("frogs");
    $number_of_frogs= $this->frogs->count_frogs();
    $data['froggies'] = $number_of_frogs;
    $this->load->view("frog_view",$data);
}
```

Model (models/frogs.php)

```
function count_frogs(){
    $this->db->where("type","frog");
    $this->db->from("animals");
    $query = $this->db->get();
    return $query->num_rows();
}
```

View (views/frog_view.php)

```
<html>
  <body>
    <h1>You've <?= $name;?> frogs in list</h1>
  </body>
</html>
```

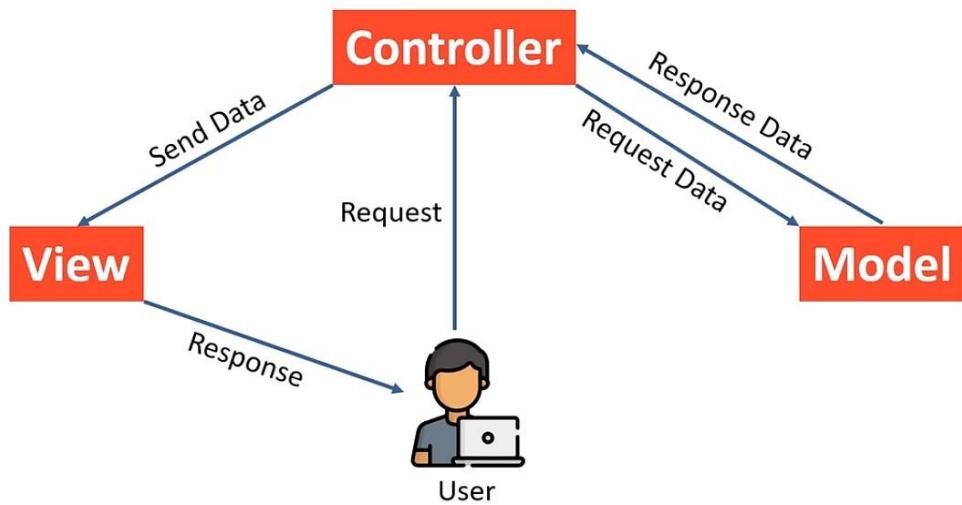


Benefits of MVC Framework

- Organized Code
- Independent Block
- Reduces the complexity of Web Applications
- Easy to maintain
- Easy to modify
- Code reusability
- Improved collaboration
- Platform independence



MVC Pattern Work Flow



Popular MVC Frameworks

PHP MVC Framework

- Laravel
- Symfony
- CodeIgniter
- Yii
- CakePHP
- Zend Framework

Other Programming MVC Framework

- Django & Flask ([Python](#))
- Ruby on Rails ([Ruby](#))
- Express.js ([JavaScript/Node.js](#))
- ASP.NET MVC ([ASP.net Core](#))



What is Framework ?

Programming frameworks are sets of **pre-written code** and **libraries** that provide a foundation for developing software applications.

Pre-written Code & Library

- Tools
- Components
- Modules

Examples

- Database Component
- Caching
- Pagination
- Session management
- Form Handling
- Security mechanisms
- User authentication
- APIs
- Payment Gateways

Benefits

- Code organization
- Reusability
- Standardization
- Testing & debugging support
- Community and support



Benefits of Laravel Framework

- Open source
- Elegant syntax
- MVC architecture
- Database migration and ORM
- Robust routing system
- Command-line Interface (**Composer**)
- Powerful template engine (**Blade Template**)
- Authentication and authorization
- Testing and debugging
- Security (**XSS, CSRF, SQL injection**)
- Scalability and performance(**Redis and Memcached**)
- Robust ecosystem and community



What we will learn in this Laravel Series

- Artisan CLI
- Routing
- Views
- Blade Template
- Controllers
- Model
- Database
- Eloquent ORM
- Migration
- Middleware
- Form Validation
- Authentication
- Handling File Upload
- APIs Validation
- CRUD Project
- News Blog Project

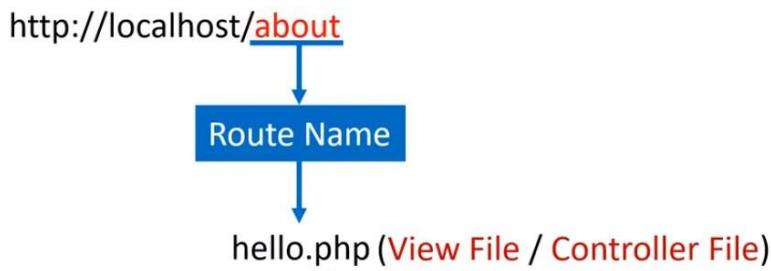
Yahoo Baba

www.yahooomba.net



Laravel Route

routes/web.php



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

Yahoo Baba

www.yahooomba.net

```
web.php  x firstpost.blade.php  post.blade.php  welcome.blade.php
routes > web.php
5 Route::get('/', function () {
6     return view('welcome');
7 });
8
9 Route::get('/post', function () {
10    return view('post');
11 });
12
13 // Route::view('post','/post');
14
15 Route::get('/post/firstpost', function () {
16     return view('post');
17 });

-q, --quiet           Do not output any message
-V, --version          Display this application version
--ansi|--no-ansi       Force (or disable --no-ansi) ANSI output
-n, --no-interaction   Do not ask any interactive question
--env[=ENV]             The environment the command should run under
-v|vv|vvv, --verbose   Increase the verbosity of messages: 1 for normal output, 2 for more ve
g

Available commands:
about                  Display basic information about your application
clear-compiled         Remove the compiled class file
completion             Dump the shell completion script
db                     Start a new database CLI session
docs                  Access the Laravel documentation
down                  Put the application into maintenance / demo mode
env                   Display the current framework environment
help                  Display help for a command
inspire                Display an inspiring quote
list                  List commands
migrate                Run the database migrations
optimize               Cache the framework bootstrap files
serve                  Serve the application on the PHP development server
test                  Run the application tests
tinker                Interact with your application
up                    Bring the application out of maintenance mode
auth
```

```

queue:retry-batch          Retry the failed jobs for a batch
queue:table                Create a migration for the queue jobs database table
queue:work                 Start processing jobs on the queue as a daemon
route
route:cache                Create a route cache file for faster route registration
route:clear                Remove the route cache file
route:list                 List all registered routes
sail
sail:add                  Add a service to an existing Sail installation
sail:install              Install Laravel Sail's default Docker Compose file
sail:publish              Publish the Laravel Sail Docker files
sanctum
sanctum:prune-expired    Prune tokens expired for more than specified number of hours
schedule
schedule:clear-cache     Delete the cached mutex files created by scheduler
schedule:list              List all scheduled tasks
schedule:run               Run the scheduled commands
schedule:test              Run a scheduled command
schedule:work              Start the schedule worker
schema
schema:dump               Dump the given database schema
session
session:table             Create a migration for the session database table
storage
storage:link              Create the symbolic links configured for the application
stub
stub:publish              Publish all stubs that are available for customization
vendor
vendor:publish            Publish any publishable assets from vendor packages
view
view:cache                Compile all of the application's Blade templates

```



Laravel Route Parameters

<http://localhost/post/10>

<http://localhost/post/yahoobaba>

<http://localhost/post/news10>

<http://localhost/post/@news10>

```

Route::get('/post/{id}', function (string $id) {
    return 'User ' . $id;
});

```

```
⌚ web.php
<?php

use Illuminate\Support\Facades\Route;

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

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

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

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

⌚ web.php
<?php

use Illuminate\Support\Facades\Route;

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

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



Laravel Route Constraints

http://localhost/post/10 → whereNumber('id')

http://localhost/post/yahoobaba → whereAlpha('name')

http://localhost/post/news10 → whereAlphaNumeric('name')

http://localhost/post/song → whereIn('category', ['movie', 'song'])

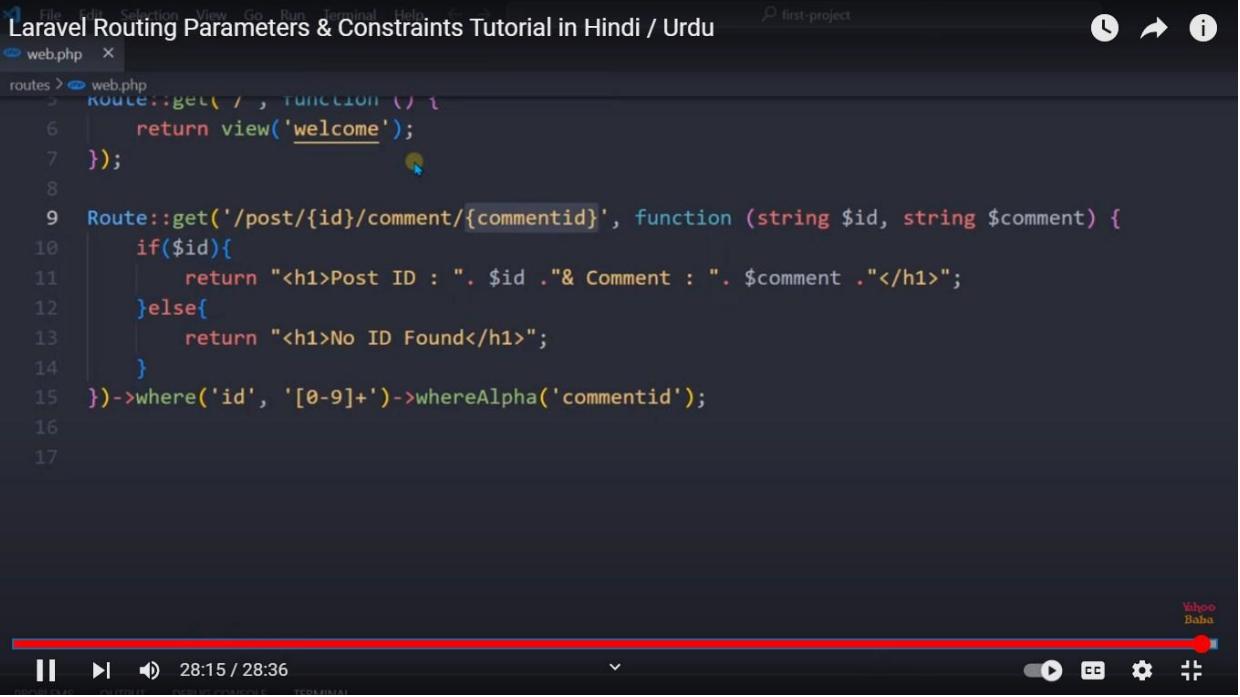
http://localhost/post/@10 → where('id', '[@0-9]+')

Regular Expressions



Laravel Route Constraints

```
Route::get('/post/{id}', function (string $id) {
    return 'User ' . $id;
})->whereNumber('id');
```



The image shows two vertically stacked screenshots of a video player interface, likely from a platform like YouTube. The top screenshot displays a code editor window for a file named 'web.php'. The code defines two routes: one for the root path '/' returning a 'welcome' view, and another for '/post/{id?}/comment/{commentid?}' which checks if an 'id' parameter is provided. If present, it returns an HTML string combining the ID and comment; if not, it returns a message stating 'No ID Found'. The bottom screenshot shows the same code in a browser window titled 'Laravel Routing Parameters & Constraints Tutorial in Hindi / Urdu'. The browser interface includes a navigation bar, a search bar, and various control buttons. A progress bar at the bottom indicates the video is at 28:15 of 28:36. The video content itself is visible in the main area.

```
<?php  
use Illuminate\Support\Facades\Route;  
  
Route::get('/', function () {  
    return view('welcome');  
});  
  
Route::get('/post/{id?}/comment/{commentid?}', function (string $id = null, string $comment = null) {  
    if($id){  
        return "<h1>Post ID : ". $id . "</h1><h2>". $comment . "</h2>";  
    }else{  
        return "<h1>No ID Found</h1>";  
    }  
});
```



Laravel Named Routes

http://localhost/page/about

```
Route::get('/page/about-us', function () {
    return 'About Page';
})->name('about');
```

first.blade.php

```
<a href="{{ route('about') }}>About</a>
```

second.blade.php

```
<a href="{{ route('about') }}>About</a>
```

third.blade.php

```
<a href="{{ route('about') }}>About</a>
```

The screenshot shows a code editor interface for a Laravel project named 'FIRST-PROJECT'. The left sidebar displays the project structure:

- app
- Console
- Exceptions
- Http
- Models
 - User.php
 - Providers
- bootstrap
- config
- database
- public
- resources
 - css
 - js
 - views
 - about.blade.php
 - post.blade.php
 - welcome.blade.php

The main editor area shows the contents of the `routes/web.php` file:

```
<?php
use Illuminate\Support\Facades\Route;
Route::get('/', function () {
    return view('welcome');
});
Route::get('/about', function () {
    return view('about');
});
Route::get('/post', function () {
    return view('post');
})->name('mypost');
```

The browser tab at the top right shows the URL `www.yahooobaba.net`.

The image displays two screenshots of a code editor interface, likely PhpStorm, showing the structure of a Laravel project named "first-project".

Top Screenshot (welcome.blade.php):

- EXPLORER:** Shows the project structure under "FIRST-PROJECT": app, Exceptions, Http, Models (User.php), Providers, bootstrap, config, database, public, resources (css, js), views (about.blade.php, post.blade.php, welcome.blade.php), routes (api.php, channels.php, console.php, web.php), storage, tests.
- EDITOR:** The file "welcome.blade.php" contains the following Blade template code:

```
<h1>Home : First Page</h1>
About
Post
```

Bottom Screenshot (routes.php):

- EXPLORER:** Same project structure as above.
- EDITOR:** The file "routes.php" contains the following Laravel route definitions:

```
<?php
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
})->name('home');

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

Route::get('page/posts', function () {
    return view('post');
})->name('mypost');
```

The image displays two screenshots of a code editor interface, likely PhpStorm, showing the structure and content of a Laravel project named "first-project".

Top Screenshot (about.blade.php):

- EXPLORER:** Shows the project structure under "FIRST-PROJECT".
- Content:** The "views" folder contains "about.blade.php", "post.blade.php", and "welcome.blade.php". The "about.blade.php" file contains the following Blade template code:

```
<h1>About Page</h1>
Home
Post
```

Bottom Screenshot (routes/web.php):

- EXPLORER:** Shows the project structure under "FIRST-PROJECT".
- Content:** The "routes" folder contains "api.php", "channels.php", "console.php", and "web.php". The "web.php" file contains the following Laravel route definitions:

```
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
})->name('home');

Route::get('page/posts', function () {
    return view('post');
})->name('mypost');

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

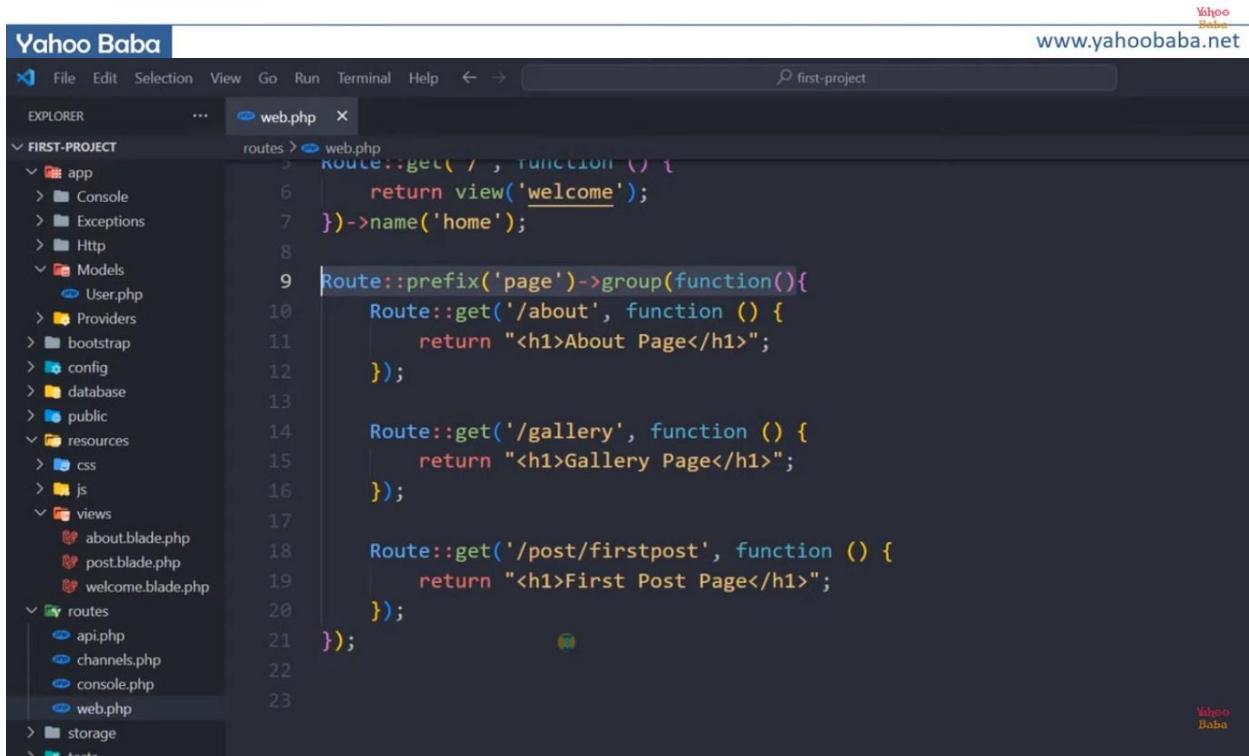
Route::redirect('/about', '/test');
```



Laravel Route Groups

```
Route::prefix('page')->group(function () {  
    Route::get('/post/1', function () {  
    });  
  
    Route::get('/about/', function () {  
    });  
  
    Route::get('/gallery/', function () {  
    });  
});
```

A diagram illustrating Laravel route grouping. It shows a main route definition with three nested routes. An arrow points from the '/about/' route to the URL <http://localhost/page/about>.



The screenshot shows the `routes/web.php` file in a code editor. The file contains the following code:

```
Route::get('/', function () {  
    return view('welcome');  
})->name('home');  
  
Route::prefix('page')->group(function(){  
    Route::get('/about', function () {  
        return "<h1>About Page</h1>";  
    });  
  
    Route::get('/gallery', function () {  
        return "<h1>Gallery Page</h1>";  
    });  
  
    Route::get('/post/firstpost', function () {  
        return "<h1>First Post Page</h1>";  
    });  
});
```

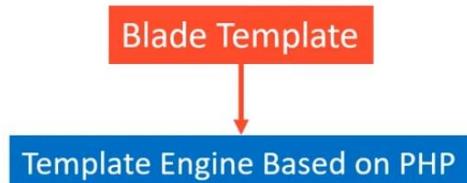
The code editor interface includes a sidebar with project files like `app`, `routes`, and `views`, and a top bar with tabs for `File`, `Edit`, `Selection`, `View`, `Go`, `Run`, `Terminal`, and `Help`. The status bar at the bottom right shows `Yahoo Baba` and `www.yahooomba.net`.

The screenshot shows a code editor interface with a dark theme. The title bar says "first-project". The left sidebar is titled "EXPLORER" and shows a project structure under "FIRST-PROJECT": app (Console, Exceptions, Http, Models, User.php), Providers, bootstrap, config, database, public, resources (css, js), and views (about.blade.php, post.blade.php, welcome.blade.php). Below these are routes (api.php, channels.php, console.php, web.php) and storage. The main editor area is titled "routes > web.php" and contains the following PHP code:

```
17
18     Route::get('/post/firstpost', function () {
19         return "<h1>First Post Page</h1>";
20     });
21 });
22
23 Route::fallback(function(){
24     return "<h1>Page Note Found.</h1>";
25 });
26
```



What is Blade Template ?



Blade provides a clean and convenient way to create views in Laravel

Benefits : Create Dynamic and Reusable Templates

HTML & PHP



Laravel : Blade Template Syntax

```
<?php  
    echo "Hello";  
?>
```

```
 {{ "Hello" }} 
```

Prevent cross-site scripting (XSS) Attacks

```
<?php  
    echo $name;  
?>
```

```
 {{ $name }} 
```

```
<?php  
    echo "<h1>Hello</h1>";  
?>
```

```
 {!! "<h1>Hello</h1>" !!}
```

Yahoo Baba

www.yahooomba.net



Laravel : Blade Template Syntax



```
<?php  
?>
```

```
@php  
@endphp
```

```
<?php  
// Comment  
?>
```

```
-- Comment --}
```

Yahoo Baba

www.yahooomba.net

5:18 / 30:21

Laravel Blade Template | Tutorial in Hindi / Urdu

Blade Control Structures Syntax

Blade Control Structures Syntax

```
<?php
if(condition){
    //Statement
}elseif(condition){
    //Statement
}else{
    //Statement
}
?>
```

```
@if (condition)
    //Statement
@elseif (condition)
    //Statement
@else
    //Statement
@endif
```

Yahoo Baba

6:05 / 30:21

Blade Control Structures Syntax

```
<?php
if(isset($records)){
    //Statement
}
?>
```

```
@isset($records)
// $records is defined and is not null...
@endisset
```

```
<?php
if(empty($records)){
    //Statement
}
?>
```

```
@empty($records)
// $records is "empty"...
@endempty
```

Laravel Blade Template - I Tutorial in Hindi / Urdu

Blade Loop Statement Syntax

The screenshot shows a browser window with several examples of Blade loop statements and a code editor below it.

Blade Loop Statement Syntax Examples:

- @for (\$i = 0; \$i < 10; \$i++)**
The current value is {{ \$i }}
@endfor
- @foreach (\$users as \$user)**
<p>This is user {{ \$user }}</p>
@endforeach
- @while (condition)**
<p>Loop Statement</p>
@endwhile
- @forelse (\$users as \$user)**
{{ \$user->name }}
@empty
<p>No users</p>
@endforelse
- @continue**
- @break**

Code Editor:

```

10:09 / 30:21
File Edit Selection View Go Run ...
first-project
resources > views > welcome.blade.php
3  {{ 5 + 2 }}
4
5  <br><br>
6
7  {{ "Hello World" }}
8
9  <br><br>
10
11 {{ "<h1>Hello World</h1>" }}
12
13{!! "<h1>Yahoo Baba</h1>" !!}
14
15 {{-- {{ "<script>alert('Yahoo Baba')</script>" }} --}}
16
17 {{-- Comment Statement --}}

```

The screenshot shows the Visual Studio Code extension marketplace. The sidebar on the left lists several extensions under 'INSTALLED' and 'RECOMMENDED'. The 'Blade Loop Variable for @foreach' extension is listed under 'INSTALLED' with a status of 22. The main panel displays a snippet of Blade PHP code:

```

17  {{-- Comment Statement --}}
18
19  @php
20      $names = ["Salman Khan", "John Abraham", "Shahid Kapoor"];
21      $user = "YahooBaba";
22  @endphp
23
24  <ul>
25  @foreach ($names as $n )
26      <li>{{ $n }}</li>
27  @endforeach
28  </ul>
29
30  @{{ $user }}
31
32  @@if()

```



Blade Loop Variable for @foreach

| Property | Description |
|-------------------|--|
| \$loop->index | The index of the current loop iteration (starts at 0). |
| \$loop->iteration | The current loop iteration (starts at 1). |
| \$loop->remaining | The iterations remaining in the loop. |
| \$loop->count | The total number of items in the array being iterated. |
| \$loop->first | Whether this is the first iteration through the loop. |
| \$loop->last | Whether this is the last iteration through the loop. |
| \$loop->even | Whether this is an even iteration through the loop. |
| \$loop->odd | Whether this is an odd iteration through the loop. |
| \$loop->depth | The nesting level of the current loop. |
| \$loop->parent | When in a nested loop, the parent's loop variable. |

The screenshot shows a code editor interface with two tabs: 'web.php' and 'welcome.blade.php'. The 'welcome.blade.php' tab is active, displaying the following Blade template code:

```
resources > views > welcome.blade.php > ul > li
^
17 {{-- Comment Statement --}}
18
19 @php
20     $names = ["Salman Khan", "John Abraham", "Shahid Kapoor"];
21     $user = "YahooBaba";
22 @endphp
23
24 <ul>
25     @foreach ($names as $n )
26         <li>{{ $loop->index }} -| {{ $n }}</li>
27     @endforeach
28 </ul>
```

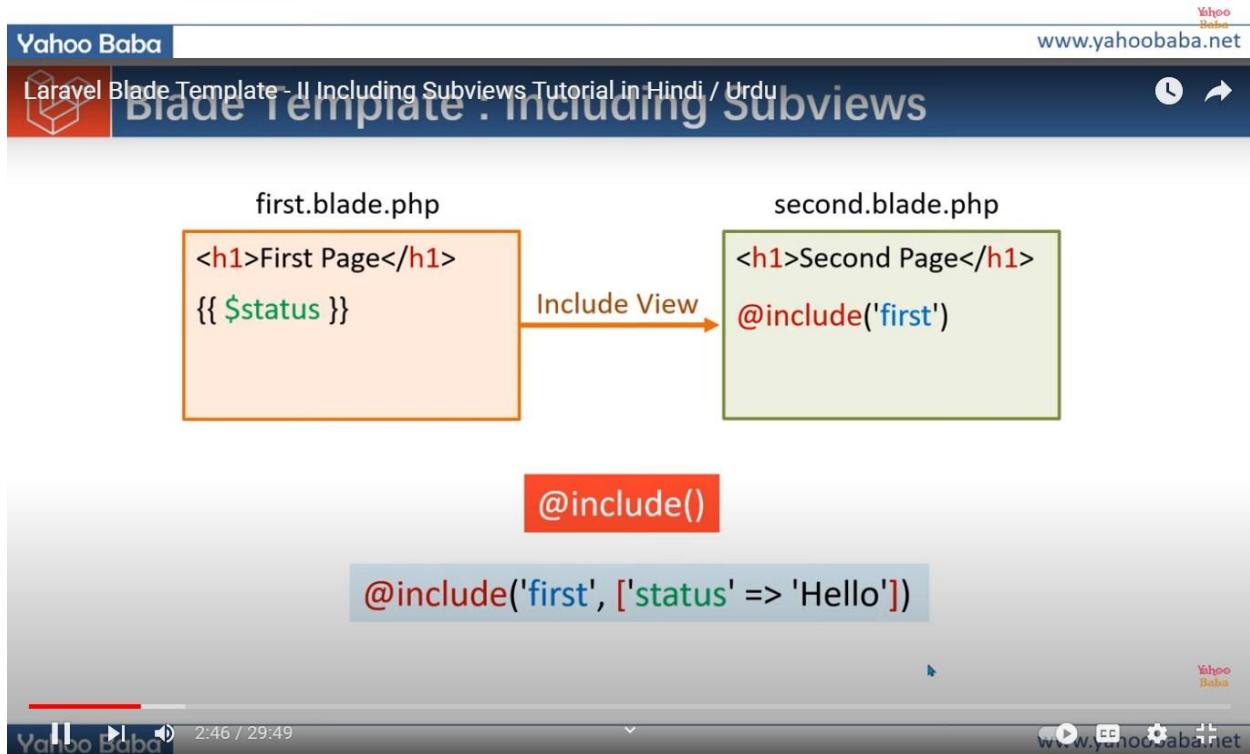
A context menu is open over the '@foreach' loop, showing options: 'b:if-else' (highlighted), 'b:can-elsecan', and 'If-else-bl can-else'. The status bar at the bottom right shows 'Yahoo Baba'.



Blade Template Main Directives

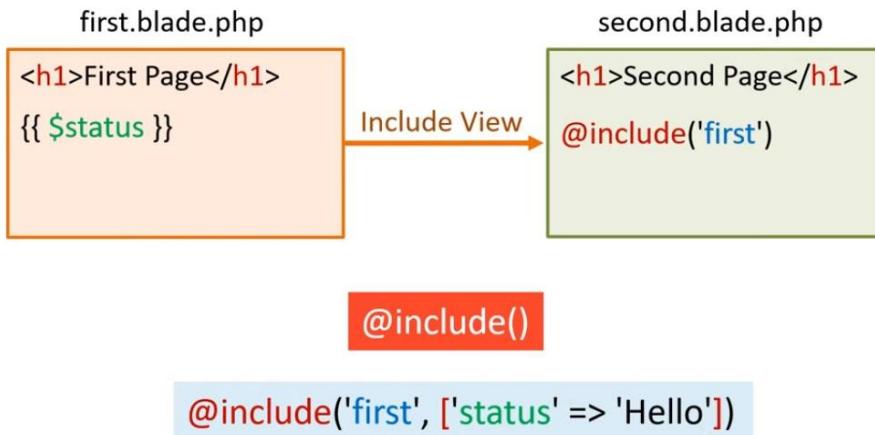
- @include
- @section
- @extend
- @yield

Reusable Templates





Blade Template : Including Subviews



Yahoo Baba

www.yahooomba.net

File Edit Selection View Go Run ...

first-project

resources > views > welcome.blade.php > ...

```
1 @include('pages.header')
2
3 <h1>Home Page</h1>
4
5 @include('pages.footer')
```

FIRST-PROJECT

- app
- bootstrap
- config
- database
- public
- resources
 - css
 - js
 - views
 - pages
 - header.blade.php
 - footer.blade.php
 - welcome.blade.php
- routes
 - api.php
 - channels.php
 - console.php
 - web.php
- storage
- tests
- vendor
- .editorconfig
- .env

The image displays two screenshots of a code editor interface, likely PhpStorm, showing parts of a Laravel application named "first-project".

Top Screenshot:

- Explorer:** Shows the project structure under "FIRST-PROJECT": app, bootstrap, config, database, public, resources (css, js, views), routes (api.php, channels.php, console.php, web.php), storage, tests, vendor, .editorconfig, .env.
- Editor:** The current file is "welcome.blade.php". The code includes:

```
1 @include('pages.header', [ 'name' => 'Yahoo Baba' ])
2
3 <h1>Home Page</h1>
4
5 @include('pages.footer')
```
- Preview:** A small preview window shows the text "Yahoo Baba".

Bottom Screenshot:

- Explorer:** Same project structure as the top screenshot.
- Editor:** The current file is "header.blade.php". The code includes:

```
1 <h1>Header Page</h1>
2
3 <p>{{ $name }}</p>
```
- Preview:** A small preview window shows the text "Yahoo Baba".

The image shows two screenshots of a code editor interface, likely PhpStorm, displaying Laravel blade files.

Screenshot 1: welcome.blade.php

File structure:

- FIRST-PROJECT
- resources > views > welcome.blade.php > ...
- app
- bootstrap
- config
- database
- public
- resources

 - css
 - js
 - views

 - pages
 - header.blade.php
 - footer.blade.php
 - welcome.blade.php

- routes

- api.php
- channels.php
- console.php
- web.php

- storage
- tests
- vendor
- .editorconfig
- .env

Code:

```
1 @php
2     $fruits = ["one" => "Apple", "two" => "Banana", "three" => "Orange"];
3 @endphp
4
5 @include('pages.header', ['names' => $fruits])
6
7 <h1>Home Page</h1>
8
9 @include('pages.footer')
10
11
12
13
```

Screenshot 2: header.blade.php

File structure:

- FIRST-PROJECT
- resources > views > pages > header.blade.php > p
- app
- bootstrap
- config
- database
- public
- resources

 - css
 - js
 - views

 - pages
 - header.blade.php
 - footer.blade.php
 - welcome.blade.php

- routes

- api.php
- channels.php
- console.php
- web.php

- storage
- tests
- vendor
- .editorconfig
- .env

Code:

```
1 <h1>Header Page</h1>
2
3 @foreach ($names as $key => $value)
4     <p>{{ $key }} - {{ $value }}</p>
5 @endforeach
```



IncludingSubviews with Conditional Check

```
@includeWhen (Condition Value,'viewfile', ['status' => 'Hello'])
```

TRUE / FALSE

```
@includeUnless (Condition Value,'viewfile', ['status' => 'Hello'])
```

The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with File, Edit, Selection, View, Go, Run, and a search bar for 'first-project'. Below the navigation is a tab bar with 'web.php', 'header.blade.php', 'welcome.blade.php X', 'style.css', and 'footer.blade.php'. The main area is the 'EXPLORER' sidebar, which lists the project structure:

- FIRST-PROJ... (selected)
- app
- bootstrap
- config
- database
- public
- resources
 - css
 - js
 - views
 - pages (selected)
 - welcome.blade.php
- routes
 - api.php
 - channels.php
 - console.php
 - web.php
- storage
- tests
- vendor
 - .editorconfig
 - .env
 - .env.example
 - .gitattributes

The 'welcome.blade.php' file is open in the editor, showing the following code:

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Yahoo Baba</title>
9
10     <link rel="stylesheet" href="css/style.css">
11
12
13 <body>
14     <div id="wrapper">
15         <header>
16             <h1>YahooBaba</h1>
17         </header>
18         <nav>
19             <a href="">Home</a> |
```

The browser title bar says 'Yahoo Baba' and the URL is 'www.yahooomba.net'. The bottom right corner of the browser window has a small 'Yahoo Baba' logo.

File Edit Selection View Go Run ... **first-project**

EXPLORER routes > **web.php** header.blade.php welcome.blade.php post.blade.php about.blade.php style.css

FIRST-PROJECT

- > app
- > bootstrap
- > config
- > database
- > public
- > resources
 - > css
 - > js
 - > views
 - > pages
 - about.blade.php
 - post.blade.php
 - welcome.blade.php
- > routes
 - api.php
 - channels.php
 - console.php
 - web.php
- > storage
- > tests
- > vendor
 - .editorconfig
 - .env

```

1 <?php
2
3 use Illuminate\Support\Facades\Route;
4
5 Route::get('/', function () {
6     return view('welcome');
7 });
8
9 Route::get('/about', function () {
10    return view('about');
11 });
12
13 Route::get('/post', function () {
14     return view('post');
15 });
16
17
18
  
```

EXPLORER routes > **welcome.blade.php** post.blade.php about.blade.php header.blade.php

FIRST-PROJECT

- > app
- > bootstrap
- > config
- > database
- > public
- > resources
 - > css
 - > js
 - > views
 - > pages
 - footer.blade.php
 - header.blade.php
 - sidebar.blade.php
 - about.blade.php
 - post.blade.php
 - welcome.blade.php
- > routes
 - api.php
 - channels.php
 - console.php
 - web.php
- > storage
- > tests
- > vendor
 - .editorconfig

```

1 @include('pages.header')
2 <article>
3     <h1>Home Page</h1>
4     <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Deleniti dol
5         accusantium quae tenetur aliquam at laborum ullam assumenda nulla bl
6         id pariatur voluptates cumque minima nemo vitae eveniet sint nobis v
7         debitibus quia! Delectus harum sapiente non mollitia veniam accusamus
8         dolorem dicta nostrum laborum nihil. Temporibus, quas non dolorem it
9         odio impedit dolore? Voluptas commodi explicabo praesentium exercita
10        accusamus explicabo repellendus labore odit, itaque suscipit incidun
11        </article>
12        @include('pages.sidebar')
13        @include('pages.footer')
14
15
16
17
  
```

Laravel Blade Template - III Template Inheritance Tutorial in Hindi / Urdu

resources > views > layouts > masterlayout.blade.php > html > body > div# wrapper > main > article

```
</nav>
<main>
    <article>
        @yield('content')
    </article>
    <aside>
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/about">About</a></li>
            <li><a href="/post">Post</a></li>
        </ul>
    </aside>
</main>
<footer>yahoobaba@copyright 2023.</footer>
</div>
</body>
```

File Edit Selection View Go Run ... 17:00 / 28:04

File Edit Selection View Go Run ...

```
@extends('layouts.masterlayout')

@section('content')
    <h2>Home Page</h2>
    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Dicta cupi
    @endsection

@section('title')
    Home
    @endsection
```

File Edit Selection View Go Run ...

```
@extends('layouts.masterlayout')

@section('content')
    <h2>Home Page</h2>
    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Dicta cupi
    @endsection

@section('title')
    Home
    @endsection
```

File Edit Selection View Go Run ... first-project

EXPLORER resources > views > layouts > masterlayout.blade.php > html > body > div#wrapper > main > aside > ul

```

FIRST-PROJECT
> app
> bootstrap
> config
> database
> public
< views
  > css
  > js
  > layouts
    > masterlayout.blade.php
    > about.blade.php
    > post.blade.php
    > welcome.blade.php
  > routes
    > api.php
    > channels.php
    > console.php
    > web.php
  > storage
  > tests
  > vendor
  .editorconfig

resources > views > layouts > masterlayout.blade.php > html > head > link

```

21 `</nav>

22 `<main>

23 `<article>

24 ` @hasSection('content')

25 ` @yield('content')

26 `@else

27 ` <h2>No Content Found.</h2>

28 `@endif

29 `</article>

30 `<aside>

31 `

32 ` Home

33 ` About

34 ` Post

35 `

36 `</aside>

37 `</main>

38 `<footer>yahoobaba@copyright 2023.</footer>

39 `</div>

File Edit Selection View Go Run ... first-project

EXPLORER resources > views > layouts > masterlayout.blade.php > html > head > link

```

FIRST-PROJECT
> app
> bootstrap
> config
> database
< views
  > css
    > style.css
    > .htaccess
    > favicon.ico
    > index.php
    > robots.txt
  > resources
    > css
    > js
  > layouts
    > masterlayout.blade.php
    > about.blade.php
    > post.blade.php
    > welcome.blade.php
  > routes
    > api.php
    > channels.php

```

1 `<!DOCTYPE html>

2 `<html lang="en">

3

4 `<head>

5 ` <meta charset="UTF-8">

6 ` <meta name="viewport" content="width=device-width, initial-scale=1.0">

7 ` <meta http-equiv="X-UA-Compatible" content="ie=edge">

8 ` <title>Yahoo Baba - @yield('title','Website')</title>

9 ` <link rel="stylesheet" href="{{ asset('css/style.css') }}>

10 `</head>

11

12 `<body>

13 ` <div id="wrapper">

14 ` <header>

15 ` <h1>YahooBaba</h1>

16 ` </header>

17 ` <nav>

18 ` `Home |

19 ` `About |



Passing Data : Route to View



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

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

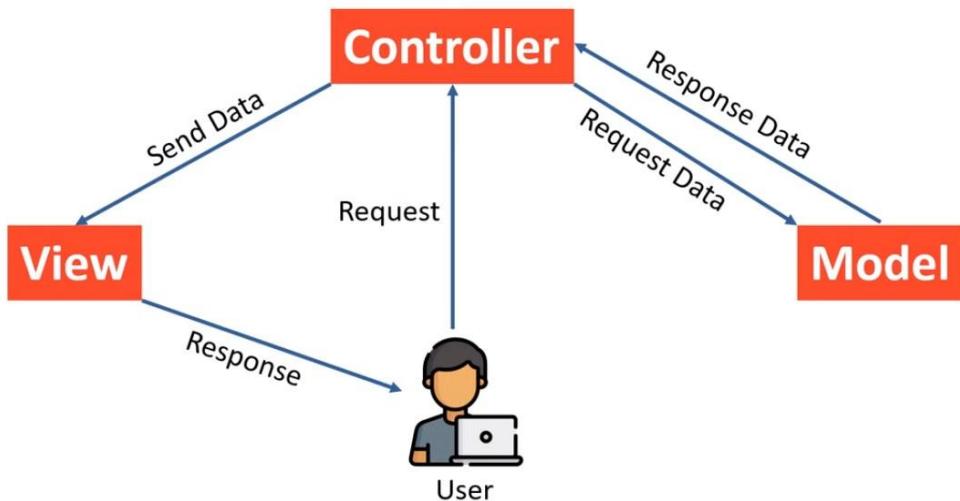
userpage.blade.php

Yahoo Baba

www.yahooomba.net



Controller Role In MVC Framework



Yahoo Baba

www.yahooomba.net



Laravel : Controller

- 1 Create Controller File

```
php artisan make:controller UserController
```

- 2 Create Controller Class

```
class UserController extends Controller
{
    public function show()
    {
        return view('user.profile');
    }
}
```

UserController.php

- 3 Create Route

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

http://localhost/user

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

The screenshot shows the Laravel IDE interface. The left sidebar displays the project structure with files like PageController.php, welcome.blade.php, web.php, routes/web.php, and various configuration and vendor files. The main editor area shows the routes/web.php file with the following code:

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PageController;

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

Route::get('/',[PageController::class,'showHome']);
Route::get(['/user',[PageController::class,'showUser']]);
```

The image shows a code editor interface with two tabs open. The top tab is titled "PageController.php" and the bottom tab is titled "web.php". Both tabs are part of a project named "first-project".

PageController.php:

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
  
class PageController extends Controller  
{  
    public function showHome(){  
        return view('welcome');  
    }  
  
    public function showUser(){  
        return view('user');  
    }  
}
```

web.php:

```
<?php  
  
use Illuminate\Support\Facades\Route;  
use App\Http\Controllers\PageController;  
  
Route::get('/', function () {  
    return view('welcome');  
});  
  
Route::get('/', [PageController::class, 'showHome'])->name('home');  
Route::get('/user', [PageController::class, 'showUser'])->name('users');
```

The image displays two screenshots of a code editor, likely PhpStorm, showing the contents of the routes/web.php file in a Laravel project named 'first-project'.

Top Screenshot:

```
<?php  
use Illuminate\Support\Facades\Route;  
use App\Http\Controllers\PageController;  
  
Route::get('/', [PageController::class, 'showHome'])->name('home');  
Route::get('/blog', [PageController::class, 'showBlog'])->name('blog');  
Route::get('/user/{id}', [PageController::class, 'showUser'])->name('users');
```

Bottom Screenshot:

```
<?php  
use Illuminate\Support\Facades\Route;  
use App\Http\Controllers\PageController;  
  
Route::controller(PageController::class)->group(function(){  
    Route::get('/', 'showHome')->name('home');  
    Route::get('/blog', 'showBlog')->name('blog');  
    Route::get('/user/{id}', 'showUser')->name('users');  
});
```



Laravel : Single Action Controllers

```
class TestController extends Controller
{
    public function __invoke()
    {
        return view('test');
    }
}
```

```
Route::get('/test', TestingController::class);
```



Laravel : Single Action Controllers

```
php artisan make:controller TestingController --invokable
or
php artisan make:controller TestingController --i
```

```
class TestController extends Controller
{
    public function __invoke()
    {
        return view('test');
    }
}
```

```
Route::get('/test', TestingController::class);
```

A screenshot of a terminal window titled "TERMINAL". The command "php artisan route:list --except-vendor" is run, displaying the following routes:

```
PS D:\laravel\first-project> php artisan route:list --except-vendor

GET|HEAD / ..... home > PageController@showHome
GET|HEAD api/user .....
GET|HEAD blog ..... blog > PageController@showBlog
GET|HEAD test ..... TestingController
GET|HEAD user/{id} ..... users > PageController@showUser

Showing [5] routes
```

PS D:\laravel\first-project>



Laravel : Steps to work in Model

- 1 Create Database

CREATE DATABASE laravel-db or Xampp (Phpmyadmin)
Projectfolder/.env

- 2 Create Database Migration
(Create tables in database)

php artisan make:migration create_students_table
Projectfolder/database/migrations/

- 3 Seeding
(Insert Initial Data in Tables)

- 4 Create Model



Laravel : Migration

1 `php artisan make:migration create_students_table`

```
CREATE TABLE table_name (
    id INT(),
    name VARCHAR(30),
    city VARCHAR(15),
    ....
);
```

```
return new class extends Migration
{
    public function up(): void
    {
        Schema::create('students', function (Blueprint $table) {
            $table->id();
            $table->string('name', 30);
            $table->string('city', 15);
        });
    }
}
```

2 `3 php artisan migrate`



Laravel : Artisan Migration Commands

`php artisan make:migration create_students_table`

`php artisan migrate`

`php artisan migrate:rollback`

`php artisan migrate:reset`

`php artisan migrate:refresh`



Laravel : Add to Column with Migration

1 php artisan make:migration update_students_table --table=students

```
return new class extends Migration
{
    public function up(): void
    {
        Schema::table('students', function (Blueprint $table) {
            $table->string('city');

        });
    }
}
```

3 php artisan migrate

Yahoo Baba

www.yahooobaba.net



Laravel : Modify Column with Migration

`$table->renameColumn('from', 'to');`

```
$table->dropColumn('city');
```

```
$table->dropColumn(['city', 'avatar', 'location']);
```

```
$table->string('name', 50)->change();
```

```
$table->integer('votes')->unsigned()->default(1)->comment('my comment')->change();
```

Yahoo Baba

www.yahooobaba.net



Laravel : Constraints with Migration

MySQL

- NOT NULL `$table->string('email')->nullable();`
- UNIQUE `$table->string('email')->unique();`
`$table->unique('email');`
- DEFAULT `$table->string('city')->default('Agra');`
- PRIMARY KEY `$table->primary('user_id');`
- FOREIGN KEY `$table->foreign('user_id')->references('id')->on('users');`
- CHECK

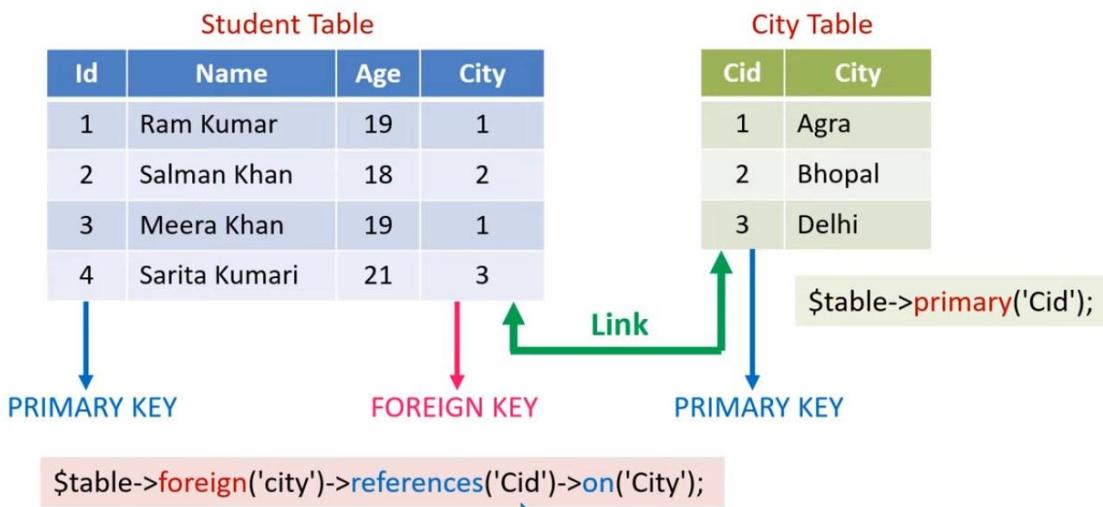


Laravel : Column Modifiers

| Modifier | Description |
|---|--|
| <code>->after('column')</code> | Place the column "after" another column (MySQL). |
| <code>->autoIncrement()</code> | Set INTEGER columns as auto-incrementing (primary key). |
| <code>->comment('my comment')</code> | Add a comment to a column (MySQL/PostgreSQL). |
| <code>->first()</code> | Place the column "first" in the table (MySQL). |
| <code>->from(\$integer)</code> | Set the starting value of an auto-incrementing field (MySQL / PostgreSQL). |
| <code>->invisible()</code> | Make the column "invisible" to SELECT * queries (MySQL). |
| <code>->unsigned()</code> | Set INTEGER columns as UNSIGNED (MySQL). |
| <code>->useCurrent()</code> | Set TIMESTAMP columns to use CURRENT_TIMESTAMP as default value. |
| <code>->useCurrentOnUpdate()</code> | Set TIMESTAMP columns to use CURRENT_TIMESTAMP when a record is updated (MySQL). |



Laravel : Primary Key & Foreign Key Constraints



Yahoo Baba www.yahooomba.net

File Edit Selection View Go Run ... project-db

2023_06_20_073601_create_students_table.php 2023_06_20_074515_create_libraries_table.php ●

database > migrations > 2023_06_20_074515_create_libraries_table.php > class > up > Closure

```
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('libraries', function (Blueprint $table) {
15             $table->id();
16             $table->unsignedBigInteger('stu_id');
17             $table->foreign('stu_id')->references('id')->on('students');
18             $table->string('book');
19             $table->date('due_date')->nullable();
20             $table->boolean(['status']);
21         });
22     }
23
24     /**
25      * Reverse the migrations.
```



Laravel : Steps to work in Seeder

1 php artisan make:model student

2 php artisan make:seeder StudentSeeder

```
use App\Models\student;

class StudentSeeder extends Seeder
{
    public function run(): void
    {
        student::create([
            'name' => 'Yahoo Baba',
            'email' => 'yahoobaba@gmail.com'
        ]);
    }
}
```

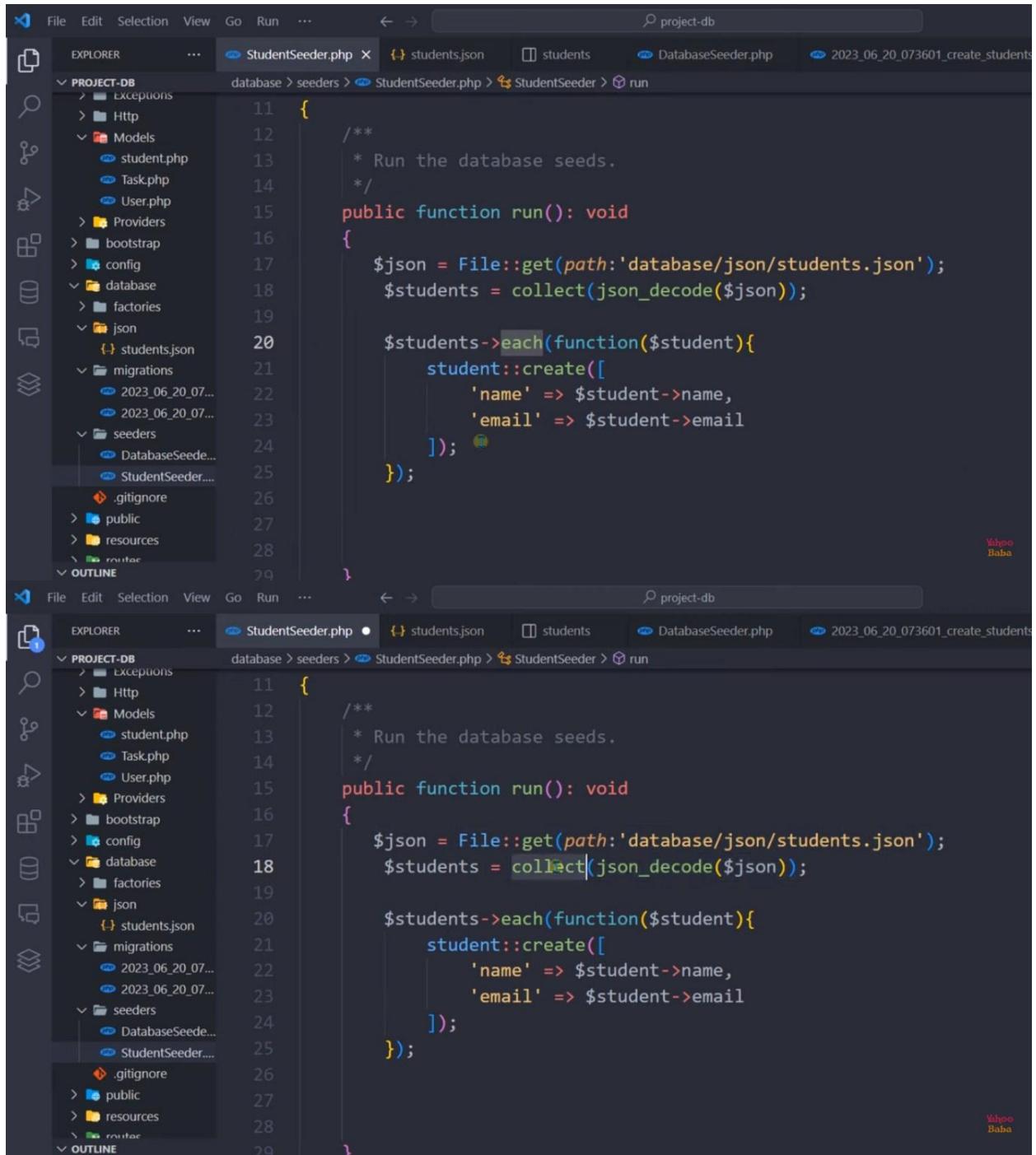
4 Seeders/DatabaseSeeder.php

```
$this->call([
    StudentSeeder::class
]);
```

5 php artisan db:seed

The screenshot shows a code editor interface with the title "Yahoo Baba" at the top. The URL "www.yahoobaba.net" is visible in the address bar. The editor displays the "StudentSeeder.php" file under the "PROJECT-DB" project. The code in the editor matches the steps shown above, including the creation of the model and the seeding logic. The code editor has a dark theme with syntax highlighting for PHP.

```
class StudentSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        student::create([
            'name' => 'Yahoo Baba',
            'email' => 'yahoobaba@gmail.com'
        ]);
    }
}
```



The image shows two side-by-side screenshots of a code editor interface, likely PhpStorm, displaying the same PHP file: `StudentSeeder.php`. The code is a database seeder responsible for creating students from a JSON file.

```
11  {
12      /**
13      * Run the database seeds.
14      */
15     public function run(): void
16     {
17         $json = File::get(path:'database/json/students.json');
18         $students = collect(json_decode($json));
19
20         $students->each(function($student){
21             student::create([
22                 'name' => $student->name,
23                 'email' => $student->email
24             ]);
25         });
26     }
27
28 }
29 }
```

The code editor features a dark theme with syntax highlighting. The left pane shows the project structure under `PROJECT-DB`, including `Exceptions`, `Http`, `Models` (with files `student.php`, `Task.php`, `User.php`), `Providers`, `bootstrap`, `config`, `database` (with `factories` and `json` subfolders containing `students.json`), `migrations` (with files `2023_06_20_07...` and `2023_06_20_07...`), `seeders` (with files `DatabaseSeeder...` and `StudentSeeder...`), `.gitignore`, `public`, `resources`, and `routes`. The right pane shows the file `StudentSeeder.php` with its contents. A status bar at the bottom indicates the file is `project-db`.

```
11  {
12      /**
13      * Run the database seeds.
14      */
15     public function run(): void
16     {
17         for($i = 1 ;$i <= 10; $i++){
18             student::create([
19                 'name' => fake()->name(),
20                 'email' => fake()->unique()->email()
21             ]);
22         }
23     }
24
25     // $json = File::get(path:'database/json/students.json');
26     // $students = collect(json_decode($json));
27     // $students->each(function($student){
28
29     // }
```



Laravel : Artisan Seeders Commands

```
php artisan make:seeder StudentSeeder
```

```
php artisan db:seed
```

```
php artisan db:seed --class=UserSeeder
```

```
php artisan db:seed --force
```

```
php artisan db:seed --class=UserSeeder --force
```



Laravel : Steps to work in Factory

1 `php artisan make:model student`

2 `php artisan make:factory StudentFactory`

```
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();
```

5 `php artisan db:seed`



Laravel : Artisan Factory Commands

```
php artisan make:factory StudentFactory
```

```
php artisan make:factory StudentFactory --model=Student (Factory & Model)
```

```
php artisan make:model student -f (Model & Factory)
```

```
php artisan db:seed
```

```
php artisan db:seed --class=UserSeeder
```

```
php artisan migrate:fresh --seed (Migration & Seeding)
```



Laravel : Steps to work in Query Builder

1 `php artisan make:controller UserController`

```
use Illuminate\Support\Facades\DB;  
  
class UserController extends Controller  
{  
    public function show()  
    {  
        $users = DB::table('users')->get();  
        return $users;  
    }  
}
```

- Read – get()
- Insert – insert()
- Update – update()
- Delete – delete()

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

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

<http://localhost/user>



Laravel : Read Data with Query Builder

```
SELECT * FROM users
```

```
DB::table('users')->get()
```

```
SELECT name, city FROM users
```

```
DB::table('users') ->select('name', 'city')->get()
```

```
SELECT * FROM users WHERE city = 'goa';
```

```
DB::table('users') ->where('city', '=', 'goa')->get()
```

```
DB::table('users') ->where('city', '=', 'goa') ->where('age', '>', 18)->get()
```

```
DB::table('users') ->where('city', '=', 'goa') ->orWhere('age', '>', 18)->get()
```

- whereBetween()
- whereIn()
- whereNull()
- whereMonth()
- whereDay()
- whereYear()
- whereTime()

Yahoo Baba

www.yahooomba.net

```
File Edit Selection View Go Run ... ← → project-query
```

EXPLORER

```
< PROJECT... > app > Http > Controllers > UserController.php > showUsers
```

```
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]);
    }
}
```

www.yahooomba.net

The screenshot displays a dark-themed IDE interface with two code editors.

Top Editor: Shows a Blade template file (`allusers.blade.php`). The code uses a `@foreach` loop to iterate over user data and display various attributes (name, email, age, city) in an `` list.

```
<h1>All Users List</h1>
@forelse ($data as $id => $user)
    <ul>
        {{ $user->name }} |
        {{ $user->email }} |
        {{ $user->age }} |
        {{ $user->city }} |
    </ul>
@empty
    No users found.
@endforelse
```

Bottom Editor: Shows a PHP file (`UserController.php`) containing a `UserController` class. The `showUsers` method retrieves all users from the database and returns them via a view. The `singleUser` method retrieves a specific user by ID.

```
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;
    }
}
```

A screenshot of the Visual Studio Code interface. The title bar says "project-query". The left sidebar shows a project structure under "PROJECT-QUERY": .vscode, app (Console, Exceptions, Http, Controllers, Middleware, Models, Providers), bootstrap, config, database, public, resources (css, js, views). In the "views" folder, there are "allusers.blade.php" and "user.blade.php". The main editor tab is "user.blade.php", which contains the following code:

```
<h1>User Detail</h1>
@foreach ($data as $id => $user )
    <h3>Name : {{ $user->name }}</h3>
    <h3>Email : {{ $user->email }}</h3>
    <h3>Age : {{ $user->age }}</h3>
    <h3>City : {{ $user->city }}</h3>
@endforeach
```

Laravel : Query Builder

use Illuminate\Support\Facades\DB;

Add New Record

```
DB::table('users')->insert([
    'name' => 'Yahoo Baba',
    'email' => 'yahoobaba@email.com',
]);
```

Delete Record

```
DB::table('users')
    ->where('id', 1)
    ->delete();
```

Update Existing Record

```
DB::table('users')
    ->where('id', 1)
    ->update([
        'city' => 'Agra',
    ]);
```

Yahoo Baba

The image shows two side-by-side code editors, both displaying the same file: UserController.php. The top editor shows code using `insert`, while the bottom editor shows code using `insertOrIgnore`. Both snippets are identical except for the method used in line 25.

```
File Edit Selection View Go Run ... project-query
users web.php UserController.php
app > Http > Controllers > UserController.php > UserController > addUser
public function addUser(){
    $user = DB::table('users')
        ->insert([
            'name' => 'Sanjay Kumar',
            'email' => 'sanjay@gmail.com',
            'age' => 20,
            'city' => 'delhi',
            'created_at' => now(),
            'updated_at' => now()
        ]);
    if($user){
        echo "<h1>Data Successfully Added.</h1>";
    }
}
41

File Edit Selection View Go Run ... project-query
users web.php UserController.php
app > Http > Controllers > UserController.php > UserController > addUser
public function addUser(){
    $user = DB::table('users')
        ->insertOrIgnore([
            [
                'name' => 'Ravi Kumar',
                'email' => 'ravi@gmail.com',
                'age' => 21,
                'city' => 'goa',
            ]
        ]);
    if($user){
        echo "<h1>Data Successfully Added.</h1>";
    }else{
        echo "<h1>Data not Added.</h1>";
    }
}
41
```

The image shows two screenshots of a code editor interface, likely PhpStorm, displaying PHP code related to user management.

Screenshot 1 (Top): UserController.php

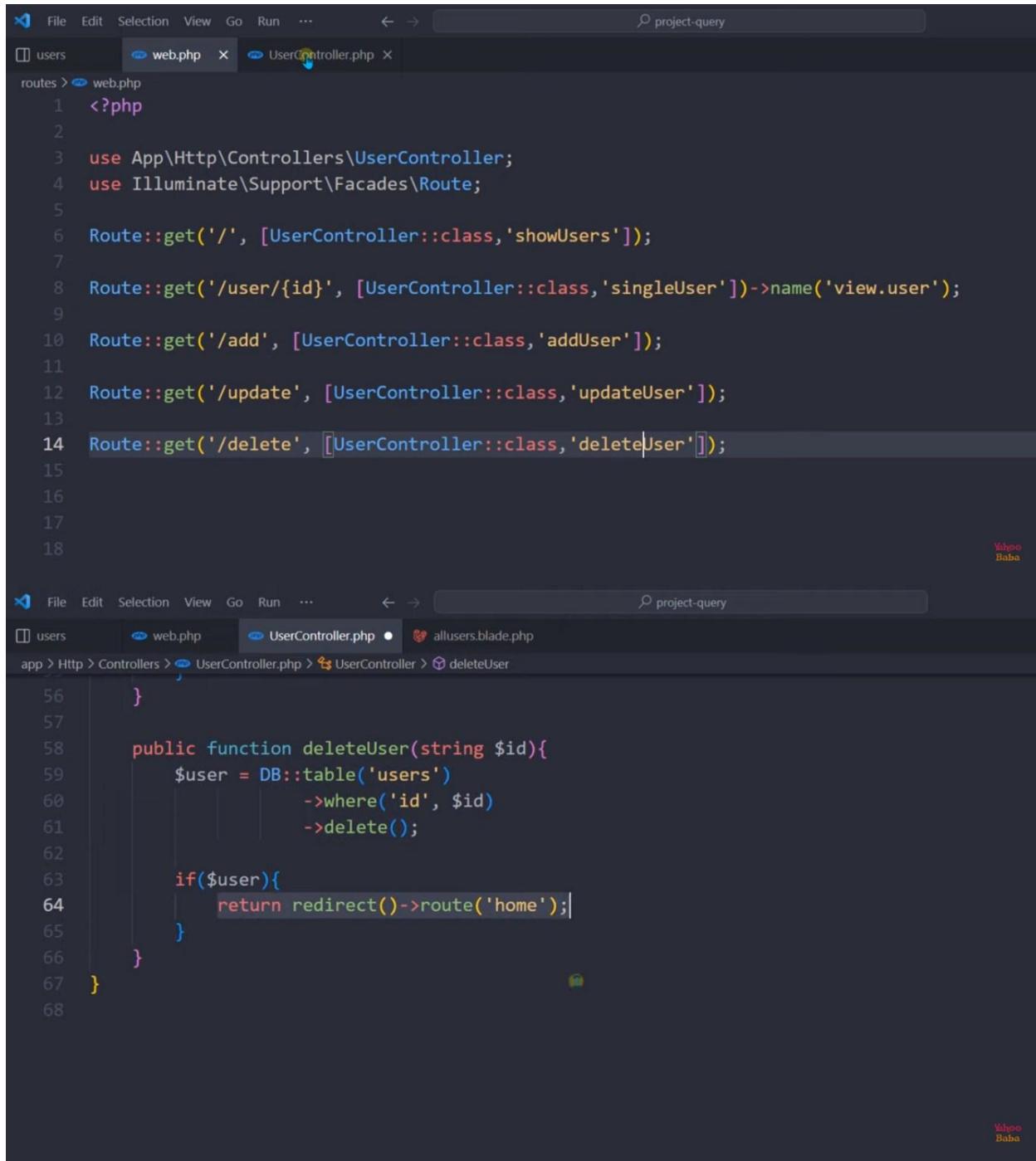
This screenshot shows the `UserController.php` file under the `Http > Controllers` directory. The code implements an `updateUser` method:

```
41 }
42
43 public function updateUser(){
44     $user = DB::table('users')
45         ->where('id', 6)
46         ->update([
47             'city' => 'Pune',
48             'age' => 22,
49         ]);
50
51     if($user){
52         echo "<h1>Data Successfully Updated.</h1>";
53     }else{
54         echo "<h1>Data not Updated.</h1>";
55     }
56 }
57
58 }
```

Screenshot 2 (Bottom): UserController.php

This screenshot shows the `UserController.php` file under the `Http > Controllers` directory. The code implements a `deleteUser` method:

```
56 }
57
58 public function deleteUser(){
59     $user = DB::table('users')
60         ->where('id', 2)
61         ->delete();
62
63 }
64
```



The image shows a code editor interface with two tabs open: "routes/web.php" and "UserController.php".

routes/web.php:

```
1 <?php
2
3 use App\Http\Controllers\UserController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get('/', [UserController::class, 'showUsers']);
7
8 Route::get('/user/{id}', [UserController::class, 'singleUser'])->name('view.user');
9
10 Route::get('/add', [UserController::class, 'addUser']);
11
12 Route::get('/update', [UserController::class, 'updateUser']);
13
14 Route::get('/delete', [UserController::class, 'deleteUser']);
```

UserController.php:

```
56 }
57
58 public function deleteUser(string $id){
59     $user = DB::table('users')
60         ->where('id', $id)
61         ->delete();
62
63     if($user){
64         return redirect()->route('home');
65     }
66 }
67
68 }
```

A screenshot of a code editor showing a file named `UserController.php`. The code is written in PHP and contains methods for managing users. The cursor is positioned on the line `->truncate();`. The code editor has a dark theme and shows line numbers from 64 to 73. The top bar includes tabs for `users`, `web.php`, `UserController.php` (which is selected), and `allusers.blade.php`. A search bar at the top right says `project-query`. The bottom right corner of the editor window displays the text `Wahab
Baba`.

```
64     return redirect()->route('home');
65 }
66 }
67
68 public function deleteAllUser(){
69     $user = DB::table('users')
70         ->truncate();
71     }
72 }
73
```



Laravel : Query Builder with HTML Forms

Name

Email

Submit

```
<form action="/addUser" method="POST">
    @csrf
    <input type="text" name="username">
    <input type="text" name="userpassword">
    <input type="submit">
</form>
```

Cross-site Request Forgeries

malicious exploit

Yahoo Baba

www.yahooomba.net



Laravel : Query Builder with HTML Forms

Name

Email

Submit

```
<form action="/addUser" method="POST">
    @csrf
    <input type="text" name="username">
    <input type="text" name="userpassword">
    <input type="submit">
</form>
```

Cross-site Request Forgeries

malicious exploit

Yahoo Baba

www.yahooomba.net

Laravel Query Builder with Forms Tutorial in Hindi / Urdu

```
routes > web.php
1 <?php
2
3 use App\Http\Controllers\UserController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get('/', [UserController::class, 'showUsers'])->name('home');
7
8 Route::get('/user/{id}', [UserController::class, 'singleUser'])->name('view.user');
9
10 Route::post('/add', [UserController::class, 'addUser'])->name('addUser');
11
12 Route::get('/update', [UserController::class, 'updateUser']);
13
14 Route::get('/delete/{id}', [UserController::class, 'deleteUser'])->name('delete.user');
15
16 Route::view('newuser', '/adduser');
17
```

File Edit Selection View Go Run ... ← → project-query

```
resources > views > adduser.blade.php > html > body > div.container > div.row > div.col-4 > form
12 <div class="container">
13   <div class="row">
14     <div class="col-4">
15       <h1>Add New User</h1>
16       <form action="{{ route('addUser') }}" method="POST">
17         <div class="mb-3">
18           <label class="form-label">Name</label>
19           <input type="text" class="form-control" name="username">
20         </div>
21         <div class="mb-3">
22           <label class="form-label">Email</label>
23           <input type="text" class="form-control" name="useremail">
24         </div>
25         <div class="mb-3">
26           <label class="form-label">Age</label>
27           <input type="text" class="form-control" name="userage">
28         </div>
29         <div class="mb-3">
30           <label class="form-label">City</label>
```

The image shows a code editor interface with two tabs open:

- routes/web.php**: This tab displays Laravel route definitions. The code includes imports for `App\Http\Controllers\UserController` and `Illuminate\Support\Facades\Route`. It defines a group for the `UserController` with various GET and POST methods for user management, including `/showUsers`, `/singleUser{id}`, `/addUser`, `/updateUser{id}`, `/updatePage{id}`, `/deleteUser{id}`, and `/newuser`.
- views/updateuser.blade.php**: This tab displays the Blade template for updating user data. The template uses Bootstrap classes for styling. It features a form with three input fields: Name, Email, and Age, each with its value set to the corresponding attribute of the `$data` variable.

```
File Edit Selection View Go Run ... ← → project-query
users web.php UserController.php updateuser.blade.php allusers.blade.php
routes > Closure
3 use App\Http\Controllers\UserController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::controller(UserController::class)->group(function(){
7     Route::get('/', 'showUsers')->name('home');
8
9     Route::get('/user/{id}', 'singleUser')->name('view.user');
10
11    Route::post('/add', 'addUser')->name('addUser');
12
13    Route::get('/update/{id}', 'updateUser')->name('update.user');
14    Route::get('/updatepage/{id}', 'updatePage')->name('update.page');
15
16    Route::get('/delete/{id}', 'deleteUser')->name('delete.user');
17 });
18
19 Route::view('newuser', '/adduser');
20
21
File Edit Selection View Go Run ... ← → project-query
users web.php UserController.php updateuser.blade.php allusers.blade.php
resources > views > updateuser.blade.php > html > body > div.container > div.row > div.col-4 > form
12 <div class="container">
13     <div class="row">
14         <div class="col-4">
15             <h1>Update User Data</h1>
16             <form action="{{ route('update.user', $data->id) }}" method="POST">
17                 @csrf
18                 <div class="mb-3">
19                     <label class="form-label">Name</label>
20                     <input type="text" value="{{ $data->name }}" class="form-control" name="username">
21                 </div>
22                 <div class="mb-3">
23                     <label class="form-label">Email</label>
24                     <input type="text" value="{{ $data->email }}" class="form-control" name="useremail">
25                 </div>
26                 <div class="mb-3">
27                     <label class="form-label">Age</label>
28                     <input type="text" value="{{ $data->age }}" class="form-control" name="userage">
29                 </div>
30             <div class="mb-3">
```

```
45 }
46
47 public function updateUser(Request $req, $id){
48     $user = DB::table('users')
49         ->where('id', $id)
50         ->update([
51             'name' => $req->username,
52             'email' => $req->useremail,
53             'age' => $req->userage,
54             'city' => $req->usercity,
55         ]);
56
57     if($user){
58         echo "<h1>Data Successfully Updated.</h1>";
59     }else{
60         echo "<h1>Data not Updated.</h1>";
61     }
62 }
```



Laravel : Router Methods

Route::get(); → Read
Route::post(); → Add, Update, Delete
Route::put(); → Update
Route::patch(); → Update
Route::delete(); → Delete
Route::options();

```
<form method="POST">
    @csrf
    @method('PUT')
</form>
```

```
Route::match(['get', 'post'], '/', function () {
    // ...
});
```

```
Route::any('/', function () {
    // ...
});
```



Laravel : Pagination Methods

1 Paginate()

```
DB::table('users')->paginate(5)
```

< 1 2 3 4 >

2 simplePaginate()

```
DB::table('users')->simplePaginate(5)
```

Previous

Next

Blade File :

3 cursorPaginate()

```
{{ $data->links() }}
```

```
DB::table('users')->orderBy('id')->cursorPaginate(5)
```



Laravel : Form Validation

| | |
|---------------------------------------|----------------------|
| Name | <input type="text"/> |
| Email | <input type="text"/> |
| <input type="button" value="Submit"/> | |

```
public function addUser(Request $req){  
    $req->validate([  
        'username' => 'required',  
        'useremail' => 'required|email',  
    ]);  
}
```

The image shows two instances of the same PHP file, UserController.php, displayed in separate windows of Visual Studio Code. Both windows have the title "UserController.php - project-validation - Visual Studio Code".

The code in both windows is identical, representing a Laravel controller for adding users. It includes imports for Request and DB, defines a UserController class extending Controller, and contains an addUser method that validates user input using the validate method of the Request object.

```
File Edit Selection View Go Run Terminal Help UserController.php - project-validation - Visual Studio Code
adduser.blade.php validation.php UserController.php ✘ web.php
app > Http > Controllers > UserController.php > UserController > addUser
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class UserController extends Controller
9 {
10     public function addUser(Request $req){
11
12         $req->validate([
13             'username' => 'required',
14             'useremail' => 'required|email',
15             'userage' => 'required|numeric',
16             'usercity' => 'required',
17         ]);
18
19         return $req->all();
20     }
}
34:07 / 54:43
File Edit Selection View Go Run Terminal Help UserController.php - project-validation - Visual Studio Code
adduser.blade.php validation.php UserController.php ● web.php
app > Http > Controllers > UserController.php > UserController > addUser
8 class UserController extends Controller
9 {
10     public function addUser(Request $req){
11
12         $req->validate([
13             'username' => 'required',
14             'useremail' => 'required|email',
15             'userage' => 'required|numeric|between:18,21',
16             'usercity' => 'required',
17         ]);
18
19         return $req->all();
20     }
21
22 }
```

```
File Edit Selection View Go Run Terminal Help
adduser.blade.php validation.php UserController.php ● web.php
app > Http > Controllers > UserController.php > UserController > addUser
8 class UserController extends Controller
9 {
10     public function addUser(Request $req){
11
12         $req->validate([
13             'username' => 'required',
14             'useremail' => 'required|email',
15             'userpass' => 'required|alpha_num|min:6',
16             'userage' => 'required|numeric|between:18,21',
17             'usercity' => 'required',
18         ]);
19
20         return $req->all();
21     }
22
23 }
```

```
File Edit Selection View Go Run Terminal Help
adduser.blade.php validation.php UserController.php ✘ web.php
app > Http > Controllers > UserController.php > UserController > addUser
10     public function addUser(Request $req){
11
12         $req->validate([
13             'username' => 'required',
14             'useremail' => 'required|email',
15             'userpass' => 'required|alpha_num|min:6',
16             'userage' => 'required|numeric|min:18',
17             'usercity' => 'required',
18         ], [
19             "username.required" =>'User Name is required!',
20             "useremail.required" =>'User Email is required!',
21             "useremail.email" =>'Enter the correct email address.',
22             "userage.required" =>'User age is required.',
23             "userage.numeric" =>'User age must be numeric.',
24             "userage.min:18" =>'User age should not less than 18 years old.',
25             "usercity.required" =>'User is required.',
26         ]);
27
28         return $req->all();

```

election View Go Run Terminal Help • adduser.blade.php - project-validation - Visual Studio Code

hp validation.php UserController.php web.php

```
> adduser.blade.php > html > body > div.container > div.row > div.col-4 > form > div.mb-3 > input.form-control @error(username).is-invalid @enderror
<form action="{{ route('addUser') }}" method="POST">
    @csrf
    <div class="mb-3">
        <label class="form-label">Name</label>
        <input type="text" value="{{ old('username') }}" class="form-control @error('username') is-invalid @enderror" name="username">
        <span class="text-danger">
            @error('username')
                {{ $message }}
            @enderror
        </span>
    </div>
    <div class="mb-3">
        <label class="form-label">Email</label>
        <input type="email" class="form-control @error('useremail') is-invalid @enderror" name="useremail">
        <span class="text-danger">
            @error('useremail')
                {{ $message }}
            @enderror
        </span>
    </div>
```