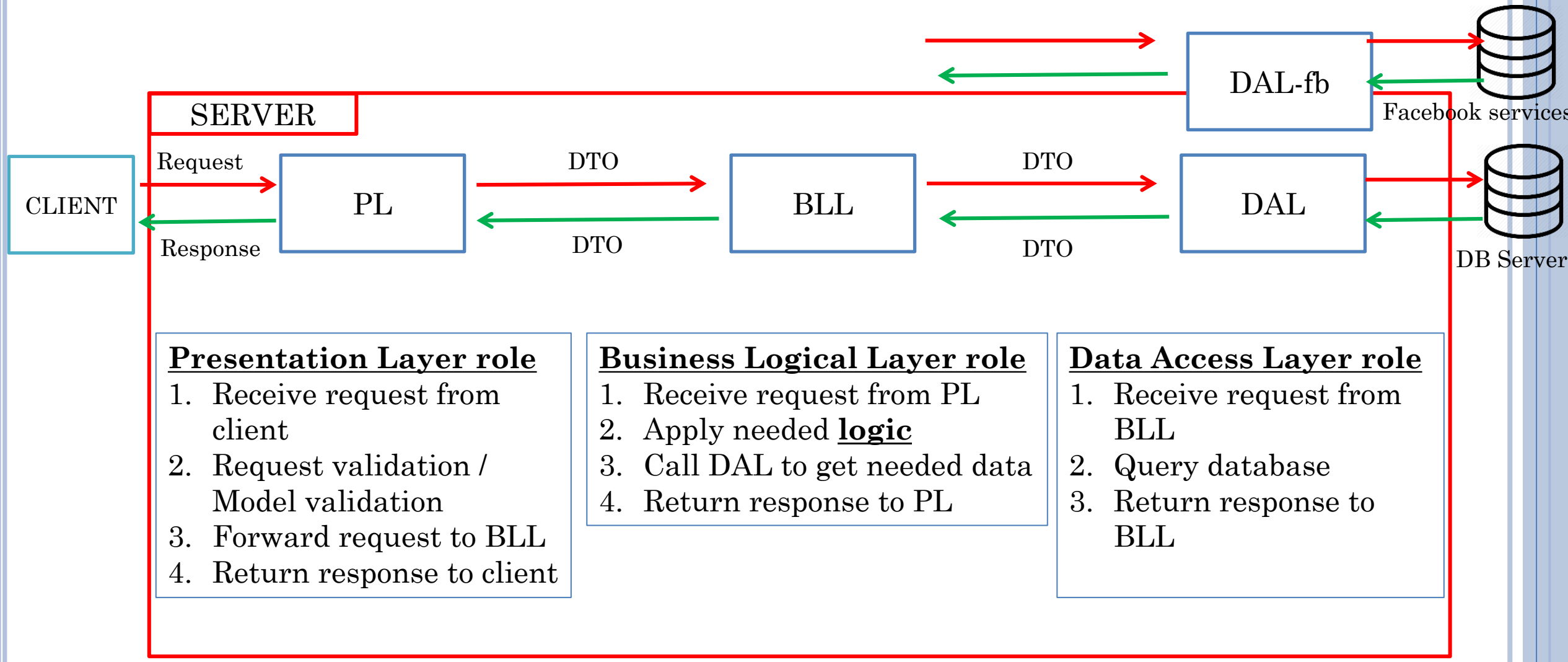
Laravel

Introduction



Presentation Layer role

1. Receive request from client
2. Request validation / Model validation
3. Forward request to BLL
4. Return response to client

Business Logical Layer role

1. Receive request from PL
2. Apply needed **logic**
3. Call DAL to get needed data
4. Return response to PL

Data Access Layer role

1. Receive request from BLL
2. Query database
3. Return response to BLL

Data Transfer Object - DTO

Are only used to pass data between layers
Does not contain any business logic

What is the MVC pattern?

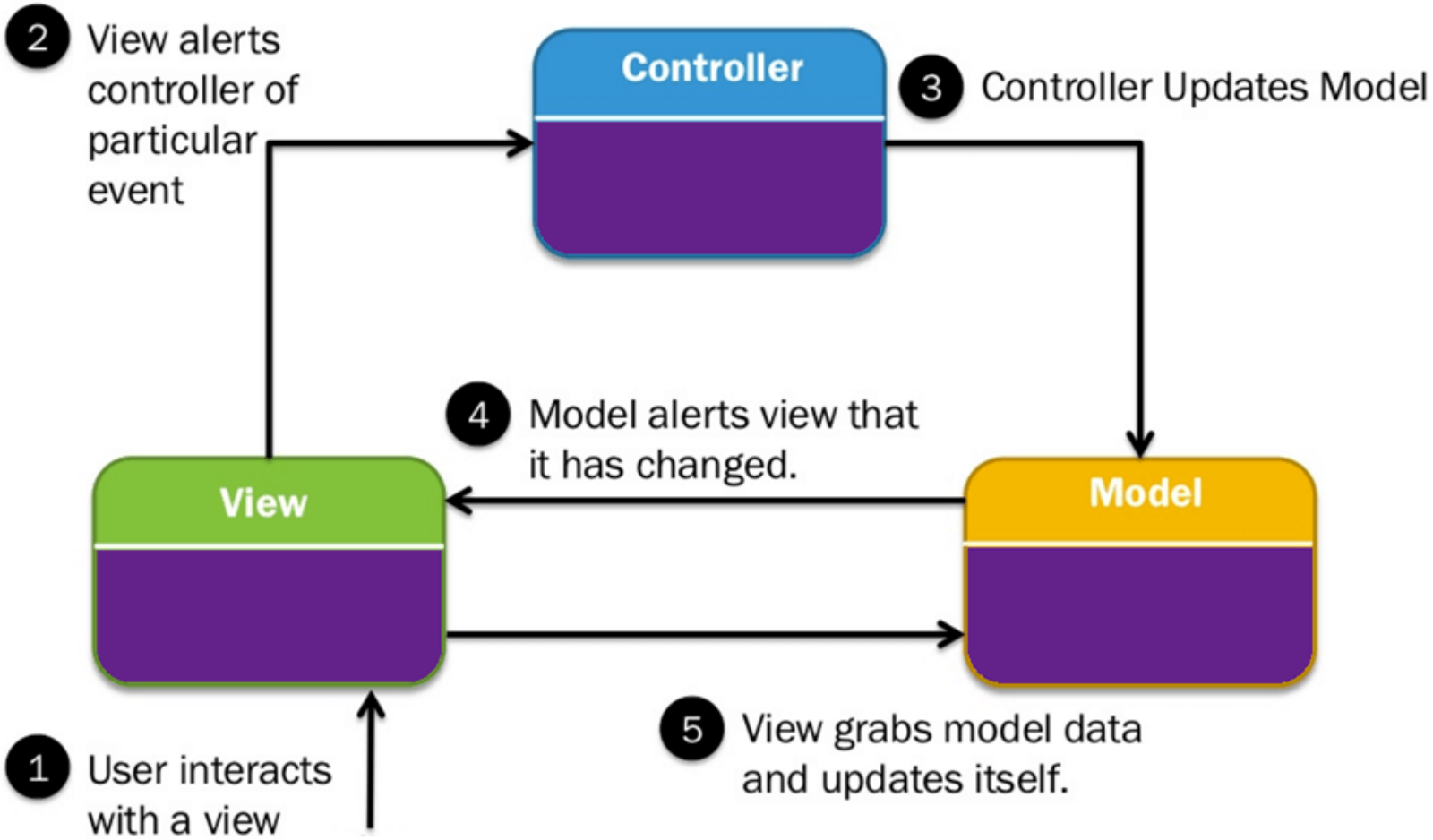
- MVC is an architectural pattern that separates an application into three main groups of components :
  - Model
  - View
  - Controller

  Each of these components are built to handle specific development aspects of an application.
- MVC is frequently used industry-standard web development framework to create scalable and extensible projects
- It helps achieve separation of concerns

What is the MVC pattern?

- Model
  - Represents the data transferred between View and Controller
    - Requests
    - Responses
- View
  - UI components
  - information display
- Controller
  - initial entry point for any user request
  - handles user input and interaction
  - works with models
  - selects a view to render

What is the MVC pattern?



https://www.guru99.com/mvc-tutorial.html

What is the MVC pattern?

Benefits of using MVC:

- Maintainability
  - Decoupled components
  - Easy to maintain / change
  - Easy to extend and grow
- Testability
  - Every component can be tested separately
  - Very suitable for test-driven development
- Cleanliness
  - It helps you to avoid complexity by dividing an application into the three units. Model, view, and controller
  - Provides clean separation of concerns(SoC).

Research

What is the exact difference between MVC Models and DB models (or entities) ?

Environment

Laravel - Installation

I. Install any web server having Apache and Mysql
   - Wamp, Xampp, EasyPhp, etc.
II. Install composer :
   - Laravel uses composer to manage dependencies
   - https://getcomposer.org/download/
   - After installation, type in the command prompt *'Composer'*

Laravel - Installation

III. Install Laravel :

- Go to your web server root folder
- Run the command to create your first Laravel project

```
composer create-project laravel/laravel --prefer-dist
```

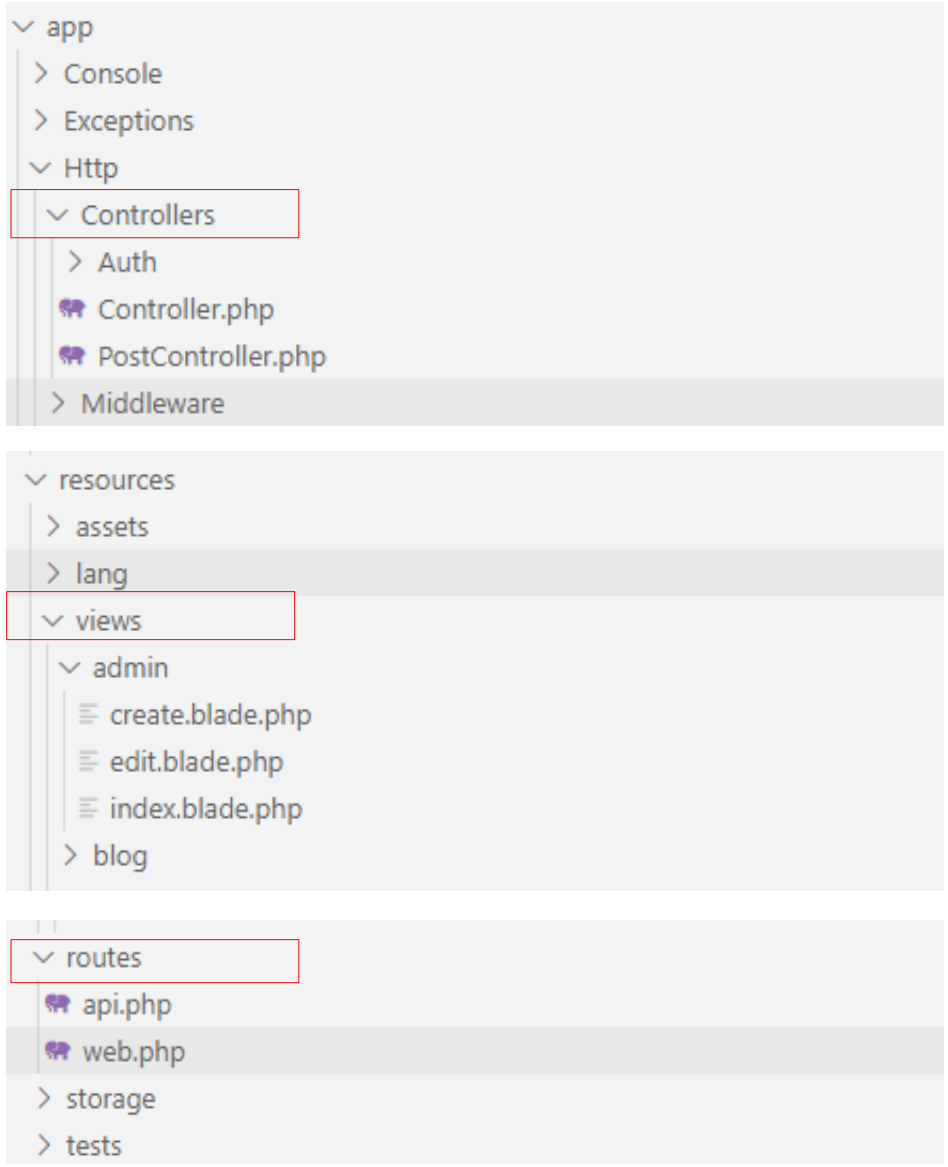
- Go to the newly created project and start Laravel services by executing

```
php artisan serve
```

```
D:\Program%20%Files\Wamp64\www\mylaravel>php artisan serve
Laravel development server started: http://127.0.0.1:8000
[Fri Sep 18 08:59:31 2020] PHP 7.4.0 Development Server (http://127.0.0.1:8000) started
```

- Open the URL in the browser

Laravel - Folders



App

- It is the application folder and includes the entire source code of the project.

Console

- Console includes the artisan commands necessary for Laravel.

Http

The Http folder has sub-folders for

- controllers
- middleware
- application requests

Laravel - Controller

- To create a new controller

```
php artisan make:controller <controller-name> --plain
```

- Example : create user controller
  - Command

```
php artisan make:controller UserController --plain
```

  - cmd

```
C:\laravel-master\laravel>php artisan make:controller UserController --plain
Controller created successfully.
```

  - it will be created at **app/Http/Controller/UserController.php**

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

    class UserController extends Controller {
        //
    }
?>
```

Laravel - Request

- We have 2 ways to read request's input values
  1. Using the input() method: it takes one argument, the name of the field in form.
  2. Using the properties of Request instance
- For example, if the form contains username field then we can access it by the following ways:
  1. input() method

```
$name = $request->input('username');
```

  2. properties method

```
$request->username
```

Laravel - Response

- We have different ways to return response
  - Basic : string, int, etc.

```
Route::get('/basic_response', function () {
   return 'Hello World';
});
```

  - Json response

```
Route::get('json',function() {
   return response()->json(['name' => 'Virat Gandhi', 'state' => 'Gujarat']);
});
```

  - Attaching Headers
  - etc.

Laravel - View

- In MVC framework, the letter **“V”** stands for **Views**
  - They contain the HTML which will be served by the application.
  - They are stored in **resources/views** directory

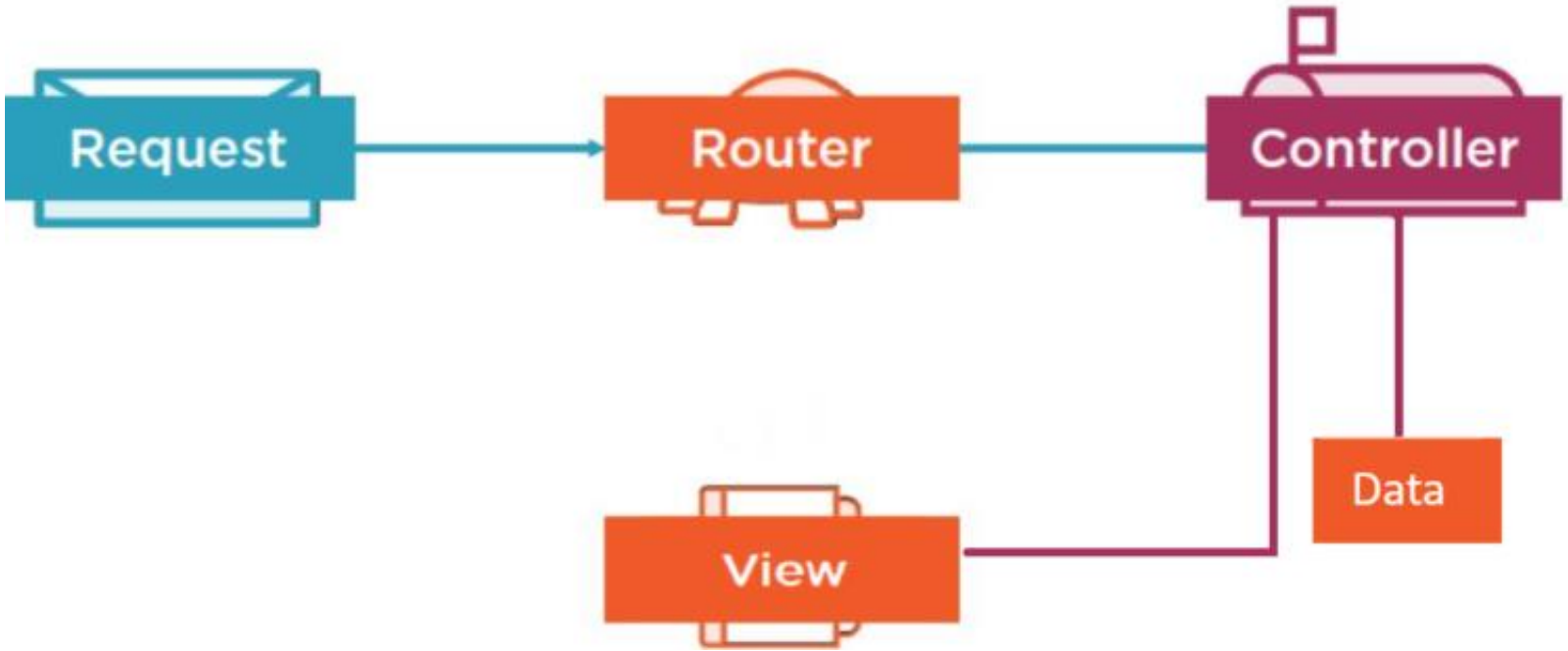
- Laravel introduced the concept of using Blade, a templating engine to design a unique layout. We will work with Blade throughout the course.

Routing

Routing

- In Laravel, all requests are mapped with the help of routes.
- Basic routing routes the request to the associated controller.
- All routes for a web app are registered in *app/routes/Web.php*
  - This file tells Laravel where to map every request

Routing



Routing - Basic

http://localhost/

```php
<?php
        1   2
Route::get('/', function () {
return view('welcome');
});
                    3

?>
```

This route is defined in *app/routes/Web.php*

It is targeting:

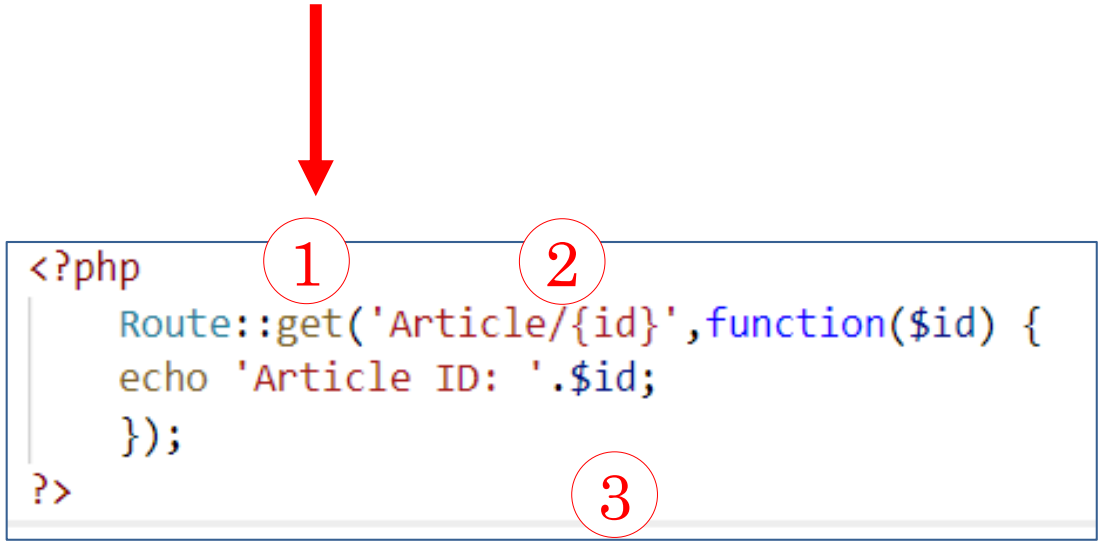
1 – Get request

2 – for the root of the site(hence the /)

3 – as a response , user will be redirected to *welcome* page

Routing - Parameters

http://localhost/Article/9



1 – Get request for Id 9

2 – from Controller *Article*

3 – as a response , *'Article ID: 9'* is echoed

Routing – Using POST

```blade
@section('content')
    @include('partials.errors')
    <div class="row">
        <div class="col-md-12">
            <form action="{{ route('admin.create') }}" method="post">
                <div class="form-group">
                    <label for="title">Title</label>
                    <input type="text" class="form-control" id="title" name="title">
                </div>
                <div class="form-group">
                    <label for="content">Content</label>
                    <input type="text" class="form-control" id="content" name="content">
                </div>
                {{ csrf_field() }}
                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div>
    </div>
@endsection
```

create.blade.php

```php
Route::post('create', [
    'uses' => 'PostController@postAdminCreate',

    'as' => 'admin.create'
]);
```

routes/web.php

Routing – Using POST

PostController.php

```php
public function postAdminCreate(Store $session, Request $request)
{
    $this->validate($request, [
        'title' => 'required|min:5',
        'content' => 'required|min:10'
    ]);
    $post = new Post();
    $post->addPost($session, $request->input('title'), $request->input('content'));
    return redirect()->route('admin.index')->with('info', 'Post created, Title is: ' . $request->input('title'));
}
```

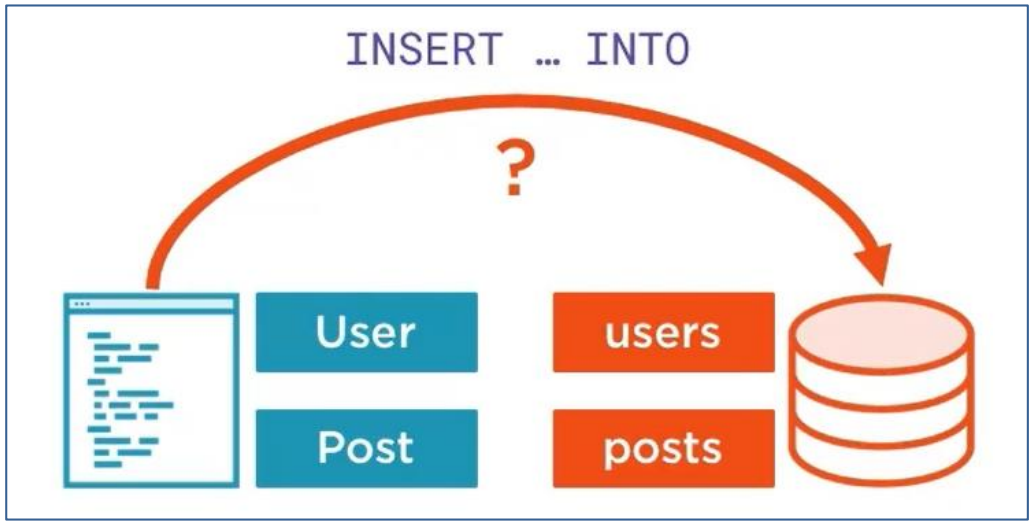
Working with data

Eloquent

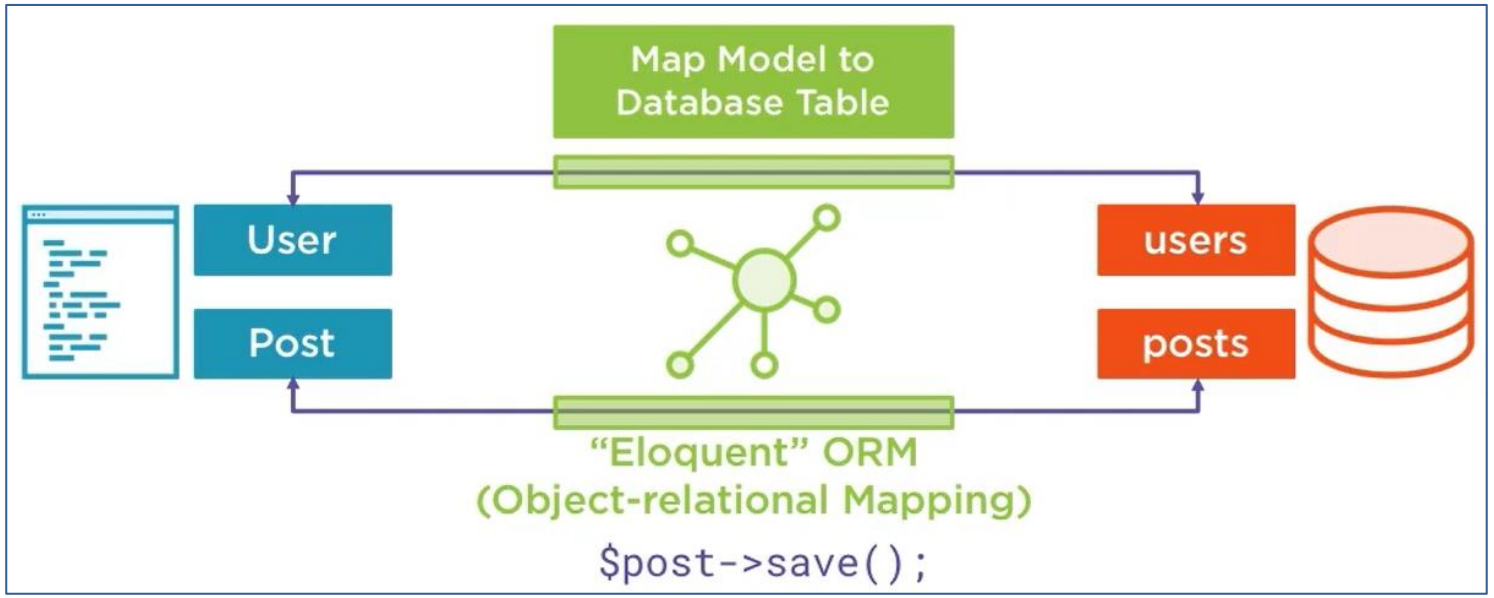
- Is an object-relational mapper (ORM)
- It provides a beautiful, simple ActiveRecord implementation for working with your database.
- Each database table has a corresponding "Model" which is used to interact with that table.
- Models allow you to query for data in your tables, as well as insert new records into the table.

ELOQUENT

Old way



Eloquent way



Eloquent- DB setup

Set up the database connection string in the .env file

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=homestead

DB_USERNAME=root

DB_PASSWORD=

.env

```
1   APP_ENV=local
2   APP_KEY=SomeRandomString
3   APP_DEBUG=true
4   APP_LOG_LEVEL=debug
5   APP_URL=http://localhost
6
7   DB_CONNECTION=mysql
8   DB_HOST=127.0.0.1
9   DB_PORT=3306
10  DB_DATABASE=homestead
11  DB_USERNAME=homestead
12  DB_PASSWORD=secret
13
14  CACHE_DRIVER=file
15  SESSION_DRIVER=file
16  QUEUE_DRIVER=sync
17
18  REDIS_HOST=127.0.0.1
19  REDIS_PASSWORD=null
20  REDIS_PORT=6380
21
22  MAIL_DRIVER=smtp
23  MAIL_HOST=mailtrap.io
24  MAIL_PORT=2525
25  MAIL_USERNAME=null
26  MAIL_PASSWORD=null
27  MAIL_ENCRYPTION=null
```

ELOQUENT- MODEL

```php
<?php
    namespace App;
    use Illuminate\Database\Eloquent\Model;

    class Post extends Model
    {
    }
?>
```

Eloquent helper

extends base model

Eloquent assumes that we have a DB table named posts

```php
// Creating & Inserting Data
$post = new Post();
$post->title = 'A new Post';
$post->save();
// Fetching Data
$oldPost = Post::where('title', 'old')->first();
```

Create and save data

Get data

ELOQUENT- MIGRATION

A migration is a PHP file that we run through Laravel, which will automatically execute all SQL commands required to configure a database

```
Schema::create('table_name', function (Blueprint $table)
{
    $table->increments('id'); // auto-incrementing integer
    $table->timestamps(); // 'created_at' and 'updated_at'
    $table->string('title'); // 'title' field
}
```

fields

ELOQUENT- MIGRATION

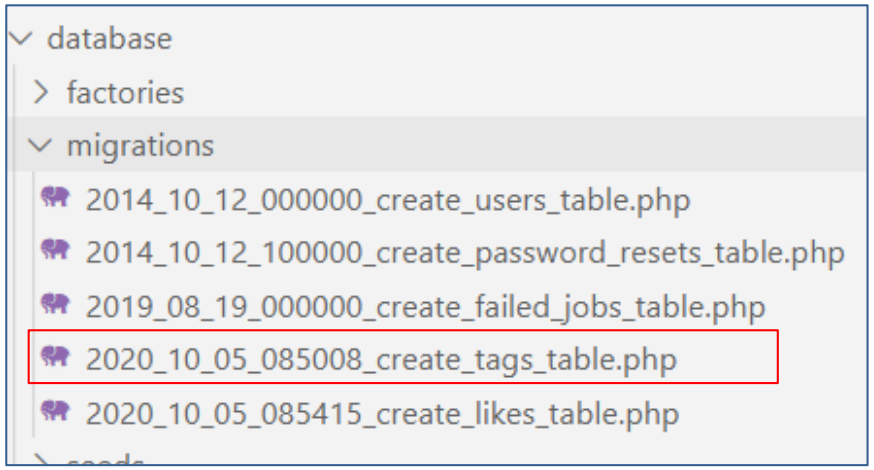
A model can be added

- Using a command
  - *php artisan make:model ModelName –m*
  - this will add the model and the migration file
  - Example : *php artisan make:model Tag -m*

app > Tag.php > ...

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Tag extends Model
{
    //
}

```

Creates Tag.php : empty model



Adds a migration file for the new model

Eloquent- Migration

```php
class CreateTagsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('tags', function (Blueprint $table) {
            $table->increments('id');
            $table->timestamps();
            $table->string('name');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('tags');
    }
```

fields added to Tag table in the database

fields to remove

Eloquent- Migration

Run migration using command :

- *php artisan migrate*

```
D:\Program%20%Files\Wamp64\www\mylaravel>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.27 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (0.17 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.13 seconds)
Migrating: 2020_10_05_085008_create_tags_table
Migrated:  2020_10_05_085008_create_tags_table (0.06 seconds)
Migrating: 2020_10_05_085415_create_likes_table
Migrated:  2020_10_05_085415_create_likes_table (0.05 seconds)
Migrating: 2020_10_07_120821_create_post_table
Migrated:  2020_10_07_120821_create_post_table (0.05 seconds)
```

Eloquent- Migration

Generate	`php artisan make:model Post -m`
	`php artisan make:migration create_posts_table`

Configure / Write	`Schema::create('table_name', function (Blueprint $table) {`
	`  $table->string('title'); // 'title' field`
	`}`

Run	`php artisan migrate`
Roll Back / Refresh	`php artisan migrate:rollback // latest migration`
	`php artisan migrate:reset // all migrations`
	`php artisan migrate:refresh // roll back all & re-run`

Eloquent- Queries

- Insert Data

```php
// Create model instance
$post = new Post();
$post->title = 'The Title';
$post->save();
```

- Fetching Data (SELECT)

```php
// Fetch models where 'title' equals 'Title'
$posts = Post::where('title','=','Title')->get();
// Only get first matching entry
$post = Post::where('title','=','Title')->first();
// Get all models
$posts = Post::all();
// Get by ID shortcut
$post = Post::find(10);
```

Eloquent- Queries

- Updating Data (UPDATE)

```
// Update single model
$post = Post::where('title','=','Title')->first();
$post->title = 'New Title';
$post->save();
// Updating multiple models at once
Post::where('title','=','Title')
->update(['title' => 'New Title']);
```

- Deleting Data (DELETE)

```
// Delete single model
$post = Post::where('title','=','Title')->first();
$post->delete();
// Deleting multiple models at once
Post::where('title','=','Title')
->delete();
```