Laravel

The PHP Framework For Web Artisans

About me

- Name: Manuel de Jesus Toala Perez
- Full Stack Developer
  T4B
- LinkedIn: kapilsharmainfo
- Web: https://manueltoala.wordpress.com/
- Github https://github.com/spaky08/
- email: manuel.toala@t4b.mx

ember

BACKBONE.JS

jQuery
write less, do more.

ANGULARJS
by Google

Bootstrap

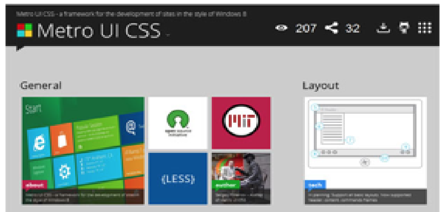
laravel

Microsoft®
.NET
MVC 5

Code Igniter

Smartphone Data User

Lemon drops oat cake oat cake sugar plum sweet chocolate cake sweet pastry gummies fruitcake cotton candy sugar plum chocolate cake cookie cotton candy tiramisu sweet


Metro UI CSS - a framework for the development of sites in the style of Windows 8

HTML

JS

CSS

php

Java
ORACLE

Ruby

node JS

Microsoft

Composer

Dependency Manager for PHP

```
php -r "copy('https://getcomposer.org/
installer', ‘composer-setup.php');"

php composer-setup.php

sudo composer.phar /usr/local/bin/composer
```

Installing Laravel

Laravel installer:

```
composer global require “laravel/installer”
laraval new <project-name>
```

Composer:

```
composer create-project laravel/laravel <project name>
“5.1.*"
```

Initial settings

- Writable permission to following folders:
  - storage/app
  - storage/framework
  - storage/logs
  - bootstrap/cache
- cp .env.example .env
- php artisan key:generate

artisan

- Command line tool to do few common tasks.
- php artisan list

Routing

- Defined in app/Http/routes.php
- Loaded by App\Providers\RouteServiceProvider

Route - closure

```
Route::get(‘/',function() {
    return 'Hello World';
});
Route::post(‘foo/bar',function() {
    return 'Hello World';
});
```

```
Route::put(‘foo/bar',function() {
    //
});
Route::delete(‘foo/bar',function() {
    //
});
```

Route - Multiple verbs

```
Route::match([‘get','post'],'/',function() {
    return 'Hello World';
});

Route::any(‘foo’,function() {
    return 'Hello World';
});
```

```
$url = url('foo');
```

Routes - Parameters

```
Route::get('user/{id}', function($id) {
    return 'User '.$id;
});
                Route::get('posts/{post}/comments/{comment}',
                    function($postId, $commentId) {
                        // code
                    }
                );
```

Note: Route parameters cannot contain the - character. Use an underscore (_) instead.

Optional Parameters

```
Route::get(‘user/{name?}’, function($name=null) {
    return $name;
});

Route::get(‘user/{name?}', function($name=‘John') {
    return $name;
});
```

Parameters Constraints

```
Route::get(‘user/{name}', function($name) {
    //
})->where('name','[A-Za-z]+');

Route::get(‘user/{id}', function($id) {
    //
})->where('id','[0-9]+');

Route::get(‘user/{id}/{name}’, function($id, $name) {
    //
})->where(['id'=>'[0-9]+','name'=>'[a-z]+']);
```

Controllers

```
<?php
namespace App\Http\Controllers;
use App\User;
use App\Http\Controllers\Controller;
class UserController extends Controller
{
 . public function showProfile($id)
   {
        $user = User::findOrFail($id)];
        return ‘Hello ‘ . $user->name;
   }
}
```

```
Route::get('user/{id}','UserController@showProfile');
```

Request

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

class UserController extends Controller
{
     public function store(Request $request)
    {
        $name = $request->input('name');
    }
}
```

Request

```
$uri=$request->path();
if($request->is('admin/*')){
$request->url();
$method=$request->method();
if($request->isMethod('post')){
if($request->has('name')){
$request->cookie('name');
$file=$request->file('photo');
if($request->hasFile('photo')){
```

Response

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

```
use Illuminate\Http\Response;

Route::get(‘home', function() {
    return (new Response($content, $status))

                ->header('Content-Type', $value);

});
```

```
Route::get(‘home', function() {
    return response($content, $status)
        ->header('Content-Type', $value);

        ->withCookie('name', 'value');

});
```

Views

```
<!—View stored in resources/views/greeting.php-->
<html>
    <body>
        <h1>Hello, <?php echo $name; ?></h1>
    </body>
</html>
```

```
Route::get('/', function () {
    return view('greeting', ['name' => 'Kapil']);
});
```

Blade

- Simple, yet powerful template engine.
- You can use plain PHP in blade (Unlike most other template engines)
- Parsed into plain php and cached, meaning zero overhead.
- File name: *.blade.php
- Mostly stored in resources/views

Layout

```
<!—Stored in resources/views/layouts/master.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>
```

Template

```
<!—Stored in resources/views/child.blade.php—>
@extends('layouts.master')
@section('title','PageTitle')


@section('sidebar')
    @@parent
    <p>This is appended to the master sidebar.</p>
@endsection

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

Displaying data

```
Route::get(‘greeting', function() {
    return view('welcome', ['name' => ‘Samantha']);
});
```

```
Hello, {{ $name }}.
The current UNIX timestamp is {{ time() }}.
```

Unescaped data

```
Hello, {!! $name !!}.
```

if statement

```
@if (count($records)===1)
    I have one record!
@elseif (count($records)>1)
    I have multiple records!
@else
    I don't have any records!
@endif
```

```
@if (!Auth::check())
    You are not signed in.
@endif
```

```
@unless (Auth::check())
    You are not signed in.
@endunless
```

Loops

```
@for($i=0;$i<10;$i++)
    The current value is {{ $i }}
@endfor
```

```
@foreach($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach
```

```
@forelse($users as $user)
  <li>{{ $user->name }}</li>
@empty
  <p>No users</p>
@endforelse
```

```
@while(true)
    <p>I'm looping forever.</p>
@endwhile
```

Configuration

- DB Configuration in config/database.php

```
'default' => env('DB_CONNECTION', 'mysql'),
'connections' => [
    'mysql' => [
        'driver'    => 'mysql',
        'host'      => env('DB_HOST', 'localhost'),
        'database'  => env('DB_DATABASE', 'forge'),
        'username'  => env('DB_USERNAME', 'forge'),
        'password'  => env('DB_PASSWORD', ''),
        'charset'   => 'utf8',
        'collation' => 'utf8_unicode_ci',
        'prefix'    => '',
        'strict'    => false,
    ],
]
```

Raw queries

```
class UserController extends Controller
{
    /**
     * Show a list of all of the application's users.
     *
     * @return Response
     */
    public function index()
    {
        $users = DB::select('select * from users where
active = ?', [1]);

        return view('user.index', ['users' => $users]);
    }
}
```

```
$results = DB::select('select * from users where
id = :id', ['id' => 1]);
```

Migration & Seeder

- Migrations are like version control for database.
- Seeder are used to add initial data in application, necessary for basic operations.

Eloquent

- Eloquent ORM comes with Laravel by default
- Simple Active Record implementation
- Each DB table have corresponding Model.
- Model used to do CRUD operation on table.
- Follows ‘Convention over configuration’ principle.

Creating Model

```
php artisan make:model User

php artisan make:model User --migration
php artisan make:model User -m
```

Model class

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    //
}
```

Model conventions

- Model - singular, table - plural. user-users
    - protected $table = ‘my_users’;
- Primary key = id
    - protected $id = ‘user_id’;
- Eloquent expect ‘created_at’ & ‘updated_at’ columns
    - public $timestamps = false;

Query

```
$users = User::all();

foreach ($users as $user) {

    echo $user->name;
}

@foreach ($users as $user)

   {{ $user->name }}

@endforeach
```

```
$flights=App\Flight::where('active',1)

                     ->orderBy('name', 'desc')

                     ->take(10)

                     ->get();
```

Insert

```
public function saveFlight(Request $request)
{
    // Validate the request…
    $flight = new Flight;
    $flight->name = $request->name;
    $flight->save();
}
```

Update

```
$flight = App\Flight::find(1);

$flight->name = ‘NewFlightName';

$flight->save();
```

Delete

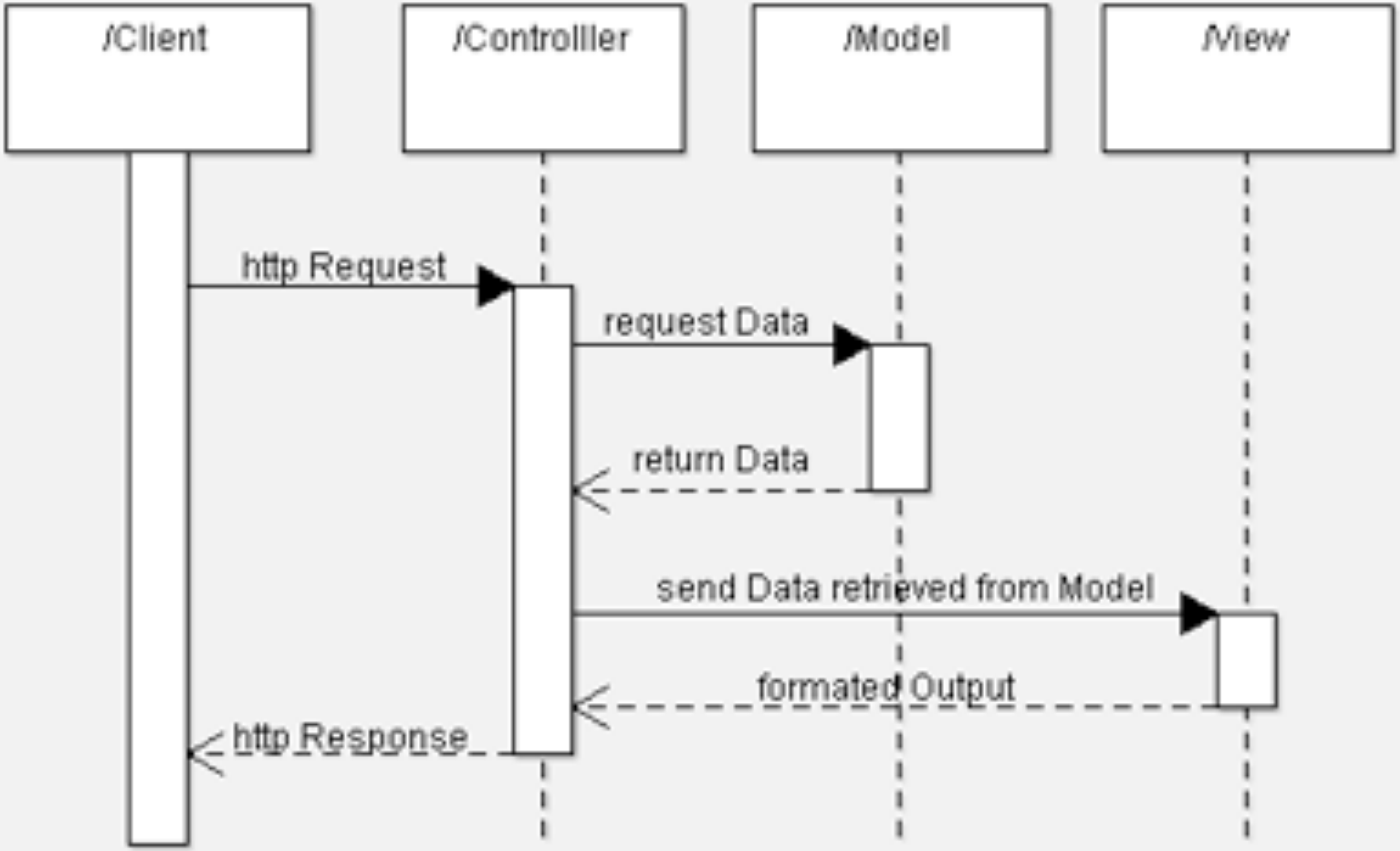
```
$flight = App\Flight::find(1);

$flight->delete();
```

MVC?

Model-View-Controller?

Separation of concerns?

MVC



Where is business logic?

What is Business logic?

request/route: Get all active users

What is active users?

This question is answered by business logic.

Where should we place business logic?

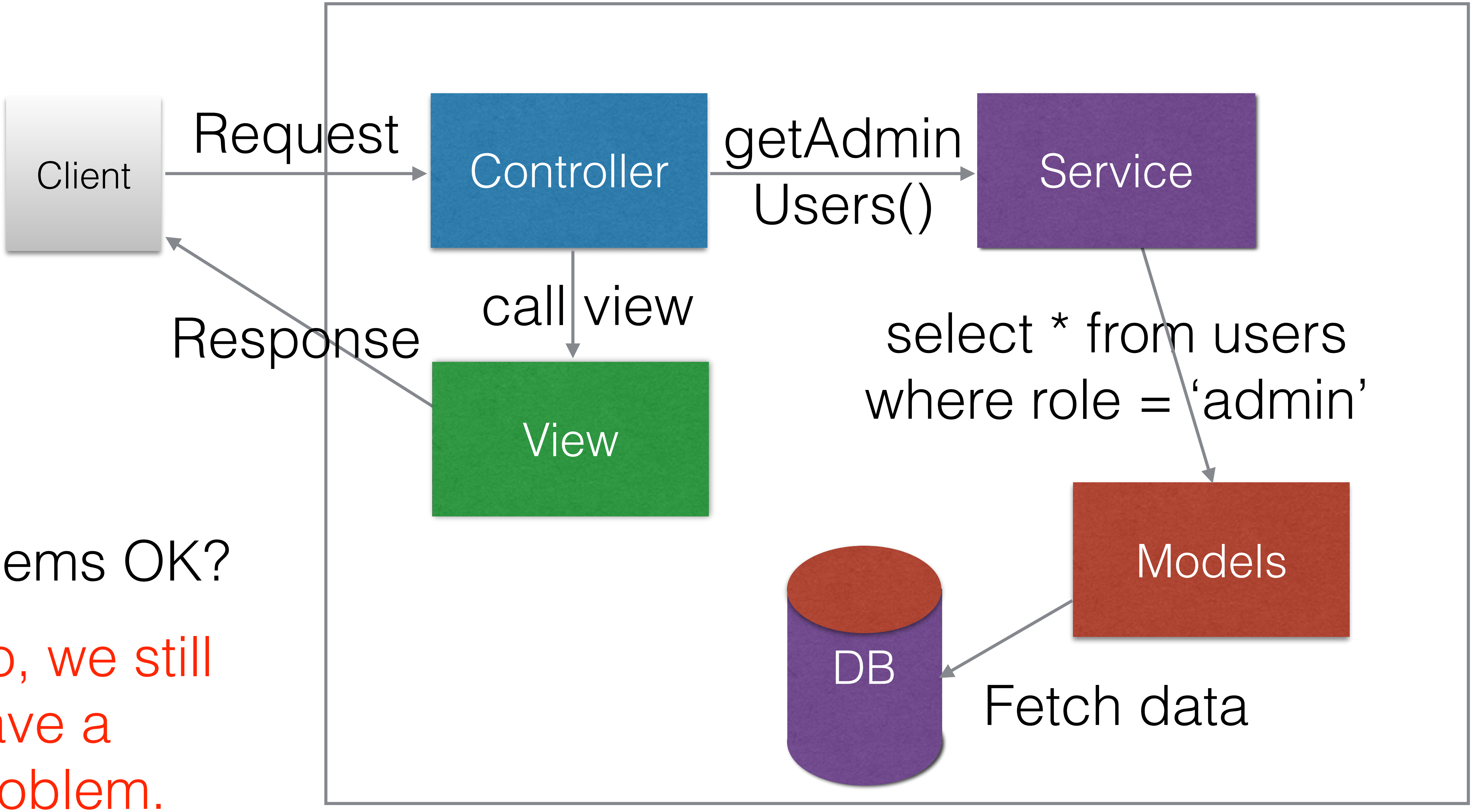
Views? Obviously NO.

Controller? Against DRY. Can duplicate code.

Models? No, model means data. BL is not data.

Service Layer

Service layer



Seems OK?

No, we still have a problem.

Thank you.
Questions?