Laravel from Scratch

An MVC Perspective

2017.5.10

ComMouse

Outline

- Reasons to try frameworks
- The MVC Pattern
- More Decoupling Techniques
- Example

Reasons to try frameworks

How can we make a website?

- First, a website contains numerous webpages;
- Second, server-end programs need to handle:
  1. Routing: Decide which script (*.php) to invoke by URL
  2. Fetching: Resolve parameters from requests & validate them
  3. Querying: Run queries in a database to look up/edit records
  4. Rendering: Return a HTML page filled with data from the database

How can we make a website? (cont.)

- First, a website contains numerous webpages;
- Second, server-end programs need to handle:
  1. Routing: Decide which script (*.php) to invoke by URL
     - `GET /activity/detail.php?id=13345` invokes `act/activity_detail.php`
  2. Fetching: Resolve parameters from requests & validate them
     - `id=13345` and it's a valid **integer**
  3. Querying: Run queries in a database to look up/edit records
     - Searching an activity which its id = 13345
     - Increasing page views of this activity by 1
  4. Rendering: Return a HTML page filled with data from the database
     - A page is returned with the detail of No. 13345 activity

What's the potential problem here if we use this architecture?

Problems of the traditional approach

1. Hell of numerous similar PHP files
2. Massive code segments with repetitive parts
3. “God-like” scripts: All the stuff in each script file

……

How to solve?

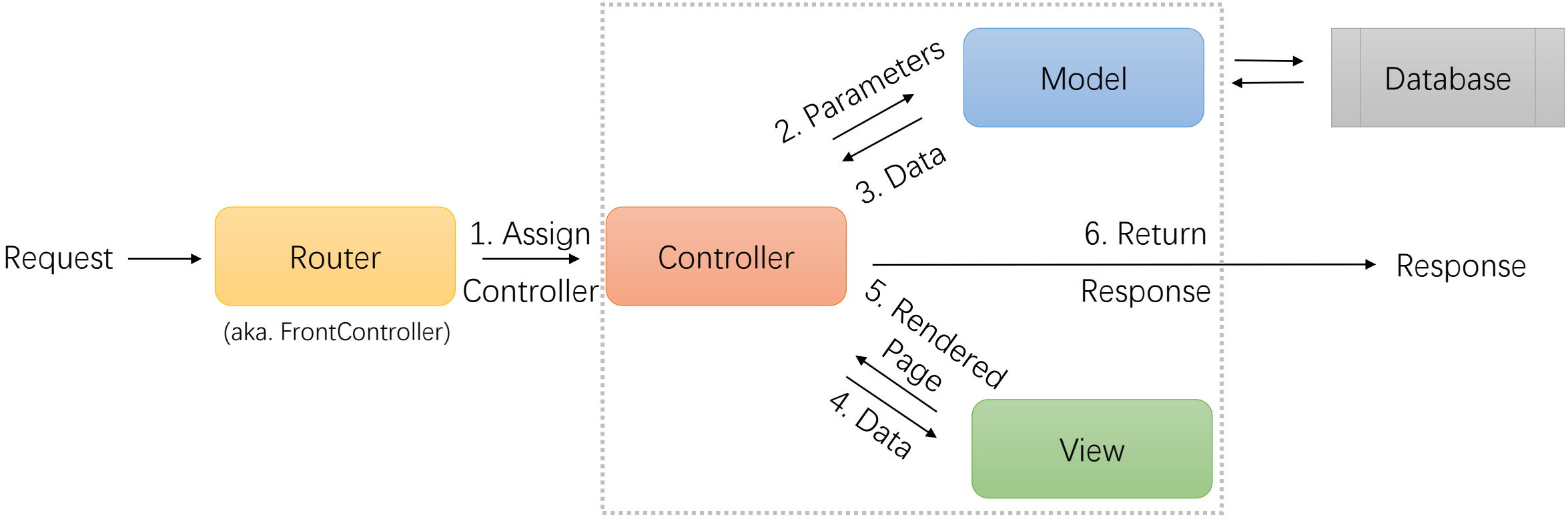
1. Divide code by responsibilities
2. Follow a pre-defined code structure

} What a framework can provide

The MVC Pattern

What's MVC?

- A core concept in nearly all the Web frameworks
- Abbreviation for the **Model-View-Controller** pattern



What's MVC? (Cont.)

- Model
  - Business Logic, Data Handling
- View
  - Data Representation, Page Rendering
- Controller
  - Request Handling, Model & View Invocation

MVC in Laravel

- Model
  - `app/` folder (e.g. `App\User`)
  - Basic **CRUD** approaches are provided through **Eloquent ORM**
- View
  - `resources/views` folder
  - Supports **Blade** template language (`@extends`, `{{ $data }}`, etc.)
- Controller
  - `app/Http/Controllers` folder (e.g. `App\Http\Controllers\Auth\LoginController`)
- Routing
  - `routes/web.php` (maps URL to a specified controller method)

More Decoupling Techniques

Dependency Injection

- Aka. Inversion of Control (IoC)
- Called “Service Container” in Laravel

PHP

```php
<?php
// …
class ArticleController extends Controller
{
    public function getArticle($articleId, Request $request, ArticleRepository $articleRepository)
    {
        $article = $articleRepository->get($articleId);
        if (is_null($article)) {
            abort(404);
        }
        return view('home.pages.article-each', [
            'article' => $article
        ]);
    }
}
```

Dependency Injection

- How does it work?
  - Collect all the interfaces/classes that can be injected
    - Request Parameter: `$articleID`
    - Defined by Service Providers: `Request`
    - User-defined classes that can be constructed automatically: `ArticleRepository`
  - Analyze the signature of a method
    - Through PHP Reflection API
  - Instantiate new instances (or fetch the existing ones) and inject into the method when the method is called
    - `$controller->getArticle($parameters->get('articleId'), $container->get(Request::class), new ArticleRepository());`

Object Relational Mapping

- Mapping between RDBMS records and PHP Objects
- A PHP Object is a DB record
- Avoids writing bare SQL statements

PHP

```php
<?php

// …
// class Task extends Model { … }

$task = new Task;
$task->content = 'Follow dyweb\'s slide';
$task->user_id = \Auth::id();
$task->save();

$old_task = Task::find(1);  // Get a task whose id = 1 from DB
$old_task->content = 'Updated content';
$old_task->save();
```

Migrations

- Version Control for Database
- Use through `php artisan migrate` command

PHP

```php
<?php
// …

class CreateSchoolsTable extends Migration
{
    public function up()
    {
        Schema::create('schools', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name', 30)->comment('学校名');
            $table->timestamps();    // create_time & update_time
        });
    }
    public function down()
    {
        Schema::dropIfExists('schools');
    }
}
```

Example

Example

- A simple to-do list
  - A user can view all the tasks that he/she has recorded
  - A user can mark a task as completed and remove it from the task list
  - A user can add a new task into the list

- Three pages to implement
  - Task List Page
  - Task Add Page
  - Task Delete Page

Database Design

- Through migrations (`php artisan make:migration CreateTasksTable`)

PHP

```php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTasksTable extends Migration
{
    public function up()
    {
        Schema::create('tasks', function (Blueprint $table) {
            $table->increments('id');
            $table->mediumText('content');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('tasks');
    }
}
```

Database Design

- The “Task” Model

app/Task.php PHP

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Task extends Model
{
    // the following line can be omitted, since Laravel can guess the table name from the class name
    protected $table = 'tasks';
}
```

Routing

routes/web.php

PHP

```php
<?php

Route::get('/', 'TaskController@getIndex');
Route::post('/add', 'TaskController@postAdd');
Route::post('/{id}/delete', 'TaskController@postDelete');
```

The Task Controller

app/Http/Controllers/TaskController.php PHP

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class TaskController extends Controller
{
    public function getIndex() { /* … */ }

    public function postAdd(Request $request) { /* … */ }

    public function postDelete($id) { /* … */ }
}
```

The Task Controller: Retrieve

app/Http/Controllers/TaskController.php

PHP

```php
// ...

use App\Task;

// ...

class TaskController extends Controller
{
    public function getIndex()
    {
        $tasks = Task::orderBy('id', 'desc')->get();

        // Render the result with “resources/views/task/list.blade.php”
        return view('task.list', ['tasks' => $tasks]);
    }
}
```

The Task Controller: Create

app/Http/Controllers/TaskController.php

PHP

```
// ...

use App\Task;

// ...

class TaskController extends Controller
{
    public function postAdd(Request $request)
    {
        $content = trim($request->input('content', ''));
        if (empty($content)) {
            return view('task.error', ['message' => 'Task content cannot be empty']);
        }

        $task = new Task;
        $task->content = $content;
        $task->save();

        return redirect('/');   // Redirect to task list page
    }
}
```

Task Controller: Delete

app/Http/Controllers/TaskController.php

PHP

```php
// ...

use App\Task;

// ...

class TaskController extends Controller
{
    public function postDelete($id)
    {
        $taskId = intval($id);
        $task = Task::find($taskId);
        if ($task == null) {
            return view('task.error', ['message' => 'Task does not exist']);
        }

        Task::destroy($taskId);

        return redirect('/');   // Redirect to task list page
    }
}
```

View

resources/views/task/list.blade.php

PHP

```
// …

@foreach($tasks as $task)
<p>Task: {!! nl2br($task->content) !!}</p>
<p>Time: {{ $task->create_time }}</p>
<form method="POST" action="{{ url("/{$task->id}/delete") }}">
    <button>Remove</button>
</form>
@endforeach

<form method="POST" action="{{ url("/add") }}">
    <textarea name="content"></textarea>
    <button>Add</button>
</form>
```

Challenge

- How to implement a user system?
  - Laravel has provided us with “Auth” façade class
  - Check the official documentation for more detail

- You may…
  - Add log in & register pages
  - Privilege check to ensure one user cannot delete other user’s task

Further Reading

- Laravel Official Documentation
  - http://laravel.com/docs
- Laravel 5.1 Tutorial
  - http://laravelacademy.org/laravel-tutorial-5_1
- PHP The Right Way
  - http://www.phptherightway.com/

- 《深入PHP：面向对象、模式与实践》
- Symfony Components
  - http://symfony.com/components

THANK YOU