Department of Information Technology
IT 350 - Web Technologies
Fall - 2021

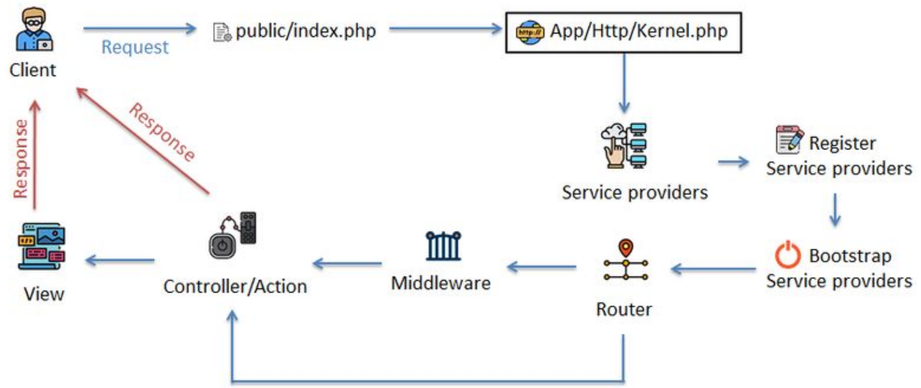
MVC, HTTP Verbs, and REST

Lecture Three

Rebin M. Ahmed
rebin.mohammed@tiu.edu.iq

Previous Lecture

- System Requirements
- Creating a New Laravel Project
- Laravel's Directory Structure
- Configurations
- Architecture Concepts
- Request Lifecycle



Contents

Pass Request Data to Views

- How to pass data in request.
- How to send the data to the Views.
- How to show the data in Views with Blade.

Break Time!

MVC

Most of what we’ll talk about in this lecture references how Model–View–Controller (MVC) applications are structured, and many of the examples we’ll be looking at use REST-ish route names and verbs, so let’s take a quick look at both.

What Is MVC?

In MVC, we have three primary concepts:

- Model : Represents an individual database table (or a record from that table)—think “Company” or “Student.”

- View : Represents the template that outputs your data to the end user—think “the login page template with this given set of HTML and CSS and JavaScript.”

What Is MVC?

- Controller: Like a traffic cop, takes HTTP requests from the browser, gets the right data out of the database and other storage mechanisms, validates user input, and eventually sends a response back to the user.

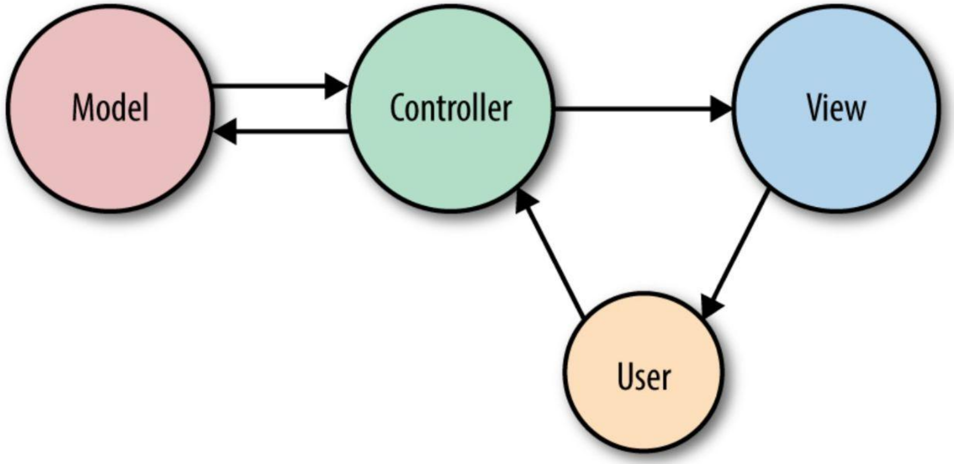


Figure 3-1. A basic illustration of MVC

What Is MVC?

In Figure 3-1, you can see that the end user will

- First interact with the controller by sending an HTTP request using their browser.
- The controller, in response to that request, may write data to and/or pull data from the model (database).
- The controller will then likely send data to a view, and then the view will be returned to the end user to display in their browser.

HTTP Verbs

The most common HTTP verbs are GET and POST, followed by PUT and DELETE. There are also HEAD, OPTIONS, and PATCH, and two others that are pretty much never used in normal web development, TRACE and CONNECT.

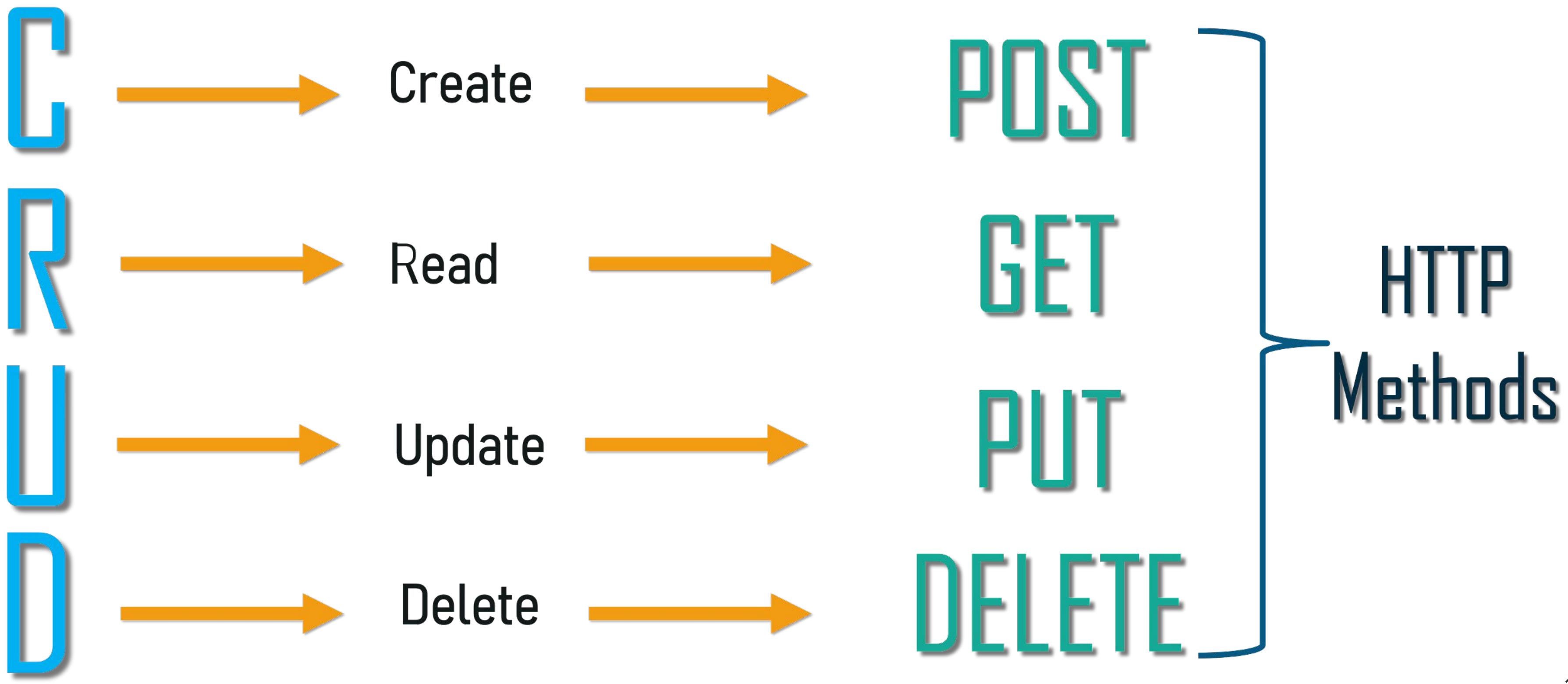
HTTP Verbs

Verb	Description
GET	Request a resource (or a list of resources).
HEAD	Ask for a headers-only version of the GET response.
POST	Create a resource.
PUT	Overwrite a resource.
PATCH	Modify a resource.
DELETE	Delete a resource.
OPTIONS	*Ask the server which verbs are allowed at this URL.*

REST

REST is acronym for **RE**presentational **S**tate **T**ransfer. REST is a software architectural style that defines a set of constraints to be used for creating Web services, Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the internet.

REST

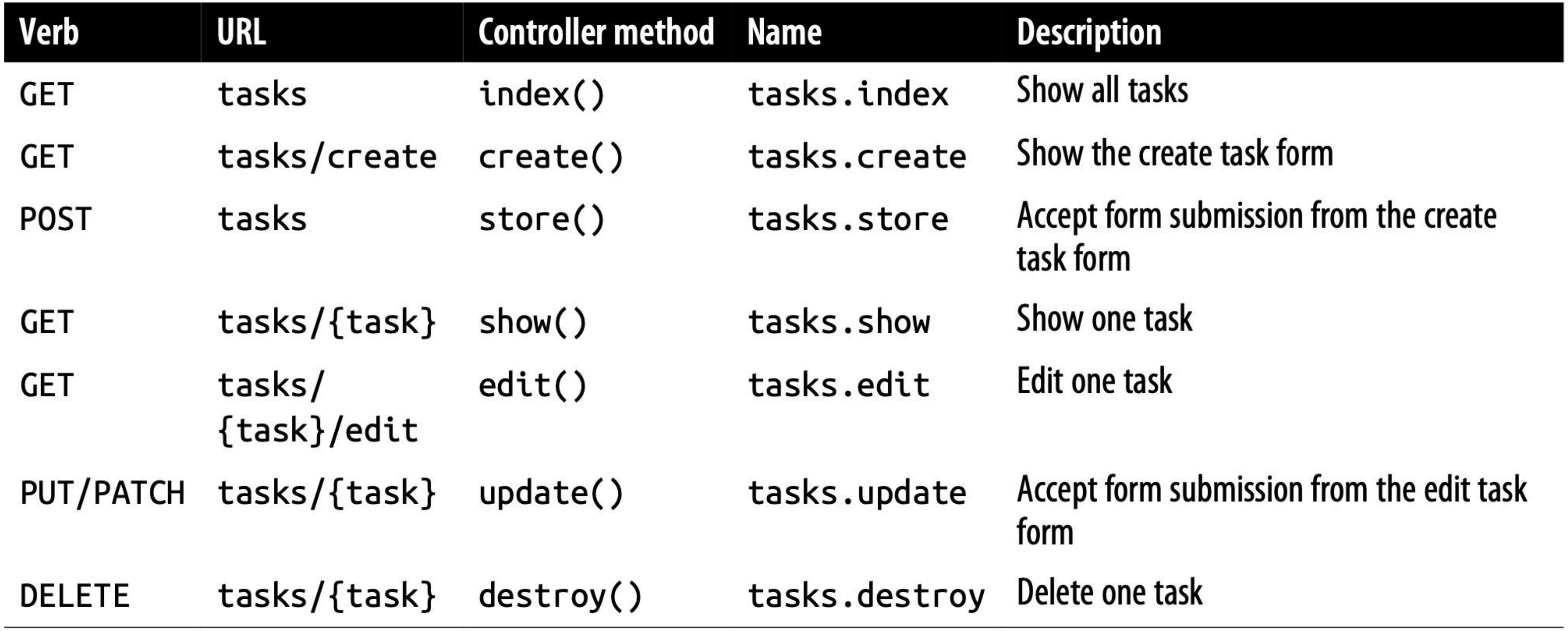

C
R
U
D
Create
Read
Update
Delete
POST
GET
PUT
DELETE
HTTP Methods

REST

There’s more to it, but usually “**RESTful**” as it’ll be used in this course will mean “patterned after these URL-based structures so we can make predictable calls like **GET /tasks/14/edit** for the edit page.”

This is relevant (even when not building APIs) because Laravel’s routing structures are based around a REST-like structure

REST



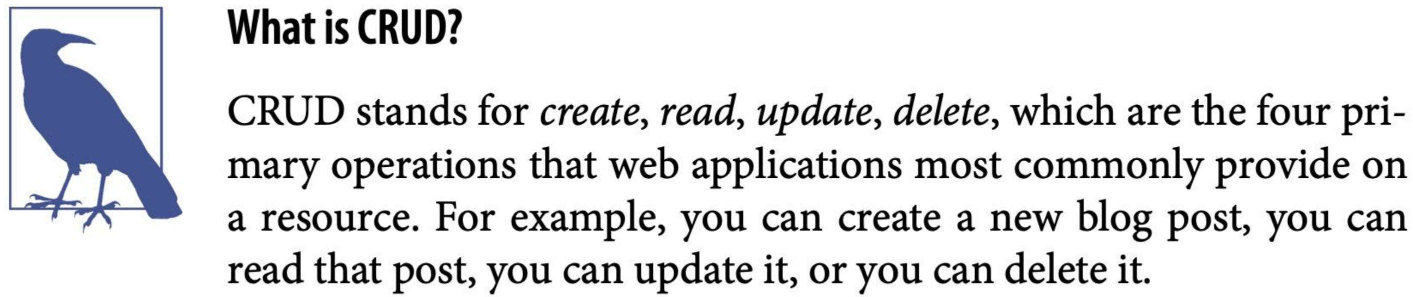
Verb	URL	Controller method	Name	Description
GET	tasks	index()	tasks.index	Show all tasks
GET	tasks/create	create()	tasks.create	Show the create task form
POST	tasks	store()	tasks.store	Accept form submission from the create task form
GET	tasks/{task}	show()	tasks.show	Show one task
GET	tasks/{task}/edit	edit()	tasks.edit	Edit one task
PUT/PATCH	tasks/{task}	update()	tasks.update	Accept form submission from the edit task form
DELETE	tasks/{task}	destroy()	tasks.destroy	Delete one task

For more details on REST
Read Chapter 13

Controllers

Controllers are essentially classes that organize the logic of one or more routes together in one place.

Controllers tend to group similar routes together, especially if your application is structured in a traditionally **CRUD**-like format; in this case, a controller might handle all the actions that can be performed on a particular resource.



What is CRUD?

CRUD stands for *create*, *read*, *update*, *delete*, which are the four primary operations that web applications most commonly provide on a resource. For example, you can create a new blog post, you can read that post, you can update it, or you can delete it.

Controllers

let’s create a controller. One easy way to do this is with an Artisan command, so from the **command line** run the following:

```
php artisan make:controller PostController
```

This will create a new file named PostController.php in app/Http/Controllers.

And this is how you can link it to a route:

```
Route::get('/posts', 'PostController@index');
```

Artisan and Artisan Generators

Laravel comes bundled with a command-line tool called Artisan. Artisan can be used to run migrations, create users and other database records manually, and perform many other manual, one-time tasks.

Under the `make` namespace, Artisan provides tools for generating skeleton files for a variety of system files. That's what allows us to run `php artisan make:controller`.

To learn more about this and other Artisan features, see Chapter 8.

For more details on topics of this lecture: Read Chapter 03

Activities and Next Week Topics

This Week:

- Read Chapter 03 of Laravel: Up & Running, for more information on MVC,HTTP Verbs and REST.
- Practice different route types and passing data through routes.
- Prepare your questions for the practical session in the lab.

Next Week:

- Routing and Controllers.

References / Further Readings

- Laravel.com : Laravel’s official Documentation.
- Matt Stauffer, 2019. Laravel: Up & Running: A Framework for Building Modern PHP Apps. O’Reilly Media.
- Dayle Rees, 2016. Laravel: Code Smart.