Web Development with PHP

Part 4 - Laravel

1

Laravel

- Open-source PHP MVC Framework
- Web Application Framework
- Taylor Otwel

2

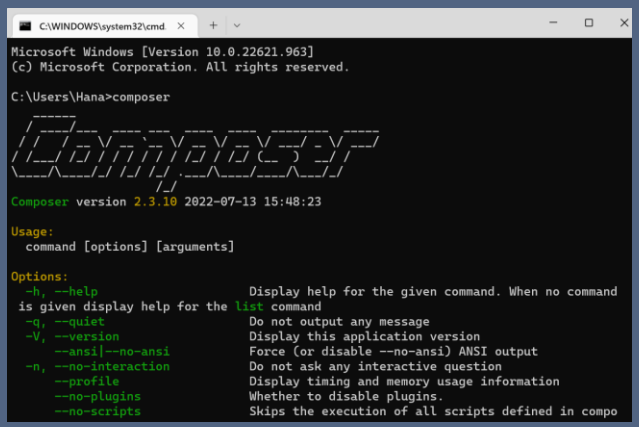
Composer

- Composer is a tool which includes all the dependencies and libraries.
- It allows a user to create a project with respect to the mentioned framework.
- Third party libraries can be installed easily with help of composer.
- All the dependencies are noted in **composer.json** file which is placed in the source folder.

3

Composer Installation

- Visit the following URL and download composer to install it on your system. https://getcomposer.org/download/
- After the Composer is installed, check the installation by typing the Composer command in the command prompt.



4

Your First Laravel Project

- Create a new Laravel project via the Composer create-project command:

```
composer create-project laravel/laravel laravel-app
```

- After the project has been created, start Laravel's local development server using the Laravel's Artisan CLI serve command.

```
cd laravel-app

php artisan serve
```

5

Directory Structure

The Root Directory

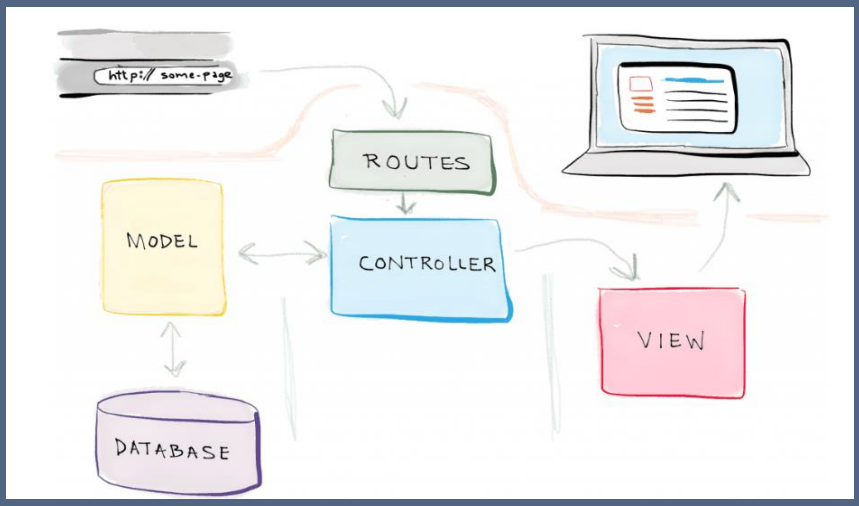
- The app Directory
- The bootstrap Directory
- The config Directory
- The database Directory
- The lang Directory
- The public Directory
- The resources Directory
- The routes Directory
- The storage Directory
- The tests Directory
- The vendor Directory

The App Directory

- The Broadcasting Directory
- The Console Directory
- The Events Directory
- The Exceptions Directory
- The Http Directory
- The Jobs Directory
- The Listeners Directory
- The Mail Directory
- The Models Directory
- The Notifications Directory
- The Policies Directory
- The Providers Directory
- The Rules Directory

6

Request Lifecycle



7

Routing

- Basic Routing

```
Route::get('/hello', [HelloWorldController::class, 'index']);
```

- Route parameters

```
Route::get('/hello/{id}', [HelloWorldController::class, 'index']);
```

- Named Routes

```
Route::get('/user/hello', [HelloWorldController::class, 'index'])->name('profile');
```

8

Basic Controllers and Routing

- Create Basic Controller by artisan command

```
php artisan make:controller HelloWorldController
```

- That controller created in app => Http => Controllers folders.
- Create Basic Routing in routes/web.php.

```
Route::get('/hello', [HelloWorldController::class, 'index']);
```

9

Resource Controllers and Routes

- Create resource Controller by artisan command

```
php artisan make:controller PhotoController --resource
```

- Create resource routes in routes/web.php file.

```
use App\Http\Controllers\PhotoController;

Route::resource('photos', PhotoController::class);
```

10

Route Resource

This route definition will define the following routes:

Verb	URI	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

11

Request

- HTTP request being handled by your application as well as retrieve the input, cookies, and files that were submitted with the request.

```
use Illuminate\Http\Request;

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

12

Generating the request

- Controller method that handles the request, allowing it to automatically validate the incoming form data.
- Use the artisan command to generate new request:

```
php artisan make:request PostRequest
```

- Add the following validation in rules function.

```
public function rules()
{
    return [
             'title' => 'required|unique:posts|max:255',
             'body' => 'required',
        ];
}
```

13

Validation

- Laravel provides several different approaches to validate your application's incoming data
- It is most common to use the validate method available on all incoming HTTP requests.
- Provide to validate the incoming values are unique or not in a given database table

```
public function store(Request $request)
{
    $validated = $request->validate([
        'title' => 'required|unique:posts|max:255',
        'body' => 'required',
    ]);

}
```

14

Views

- Save hello.blade.php file at **resources/views/** directory.

```
<html>
   <body>
      <h1>Hello, World</h1>
   </body>
</html>
```

- Add route in routes/web.php file

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

15

Views

- Add the following code to the index function in HelloWorldController.

```
public function index()
{
   return view('hello');
}
```

16

Blade Template & Parameter Passing

- Add the following code in routes/web.php

```
Route::get('/hello/{name}', [HelloWorldController::class, show']);
```

- Add the following function in app/Http/Controllers/HelloWorldController.php.

```
public function show($name)
{
    return view('hello',array('name' => $name));
}
```

- Update the following code in hello.blade.php.

```
<html>
   <body>
      <h1>Hello {{$name}}, welcome to Myanmar</h1>
   </body>
</html>
```

17

Environment Configuration

- Add the following parameters in .env file.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_bookstore
DB_USERNAME=root
DB_PASSWORD=
```

- If we changed configuration, we must be clear config and cache.

```
php artisan config:clear
php artisan config:cache
```

18

Database Configuration

- Configure the database in config/database.php file.
- Create a database “laravel_bookstore” in http://localhost/phpmyadmin.

19

Migrations

- Migrations are like version control for your database.
- Migrations are used to create and modify database schema.

- Create books table using migration

```
php artisan make:migration create_books_table
```

- You will find your newly created migration file in *database/migrations* folder.

20

Migrations

- Add some more fields to table, edit in *up* function as follows.

```
public function up()
  {
      Schema::create('books', function (Blueprint $table) {
          $table->id(); // Primary Key, Auto Increment
          $table->string('title');
          $table->string('code_number');
          $table->string('author');
          $table->string('category');
          $table->string('publishing_house');
          $table->integer('edition');
          $table->timestamps(); // created_at and updated_at
      });
  }
```

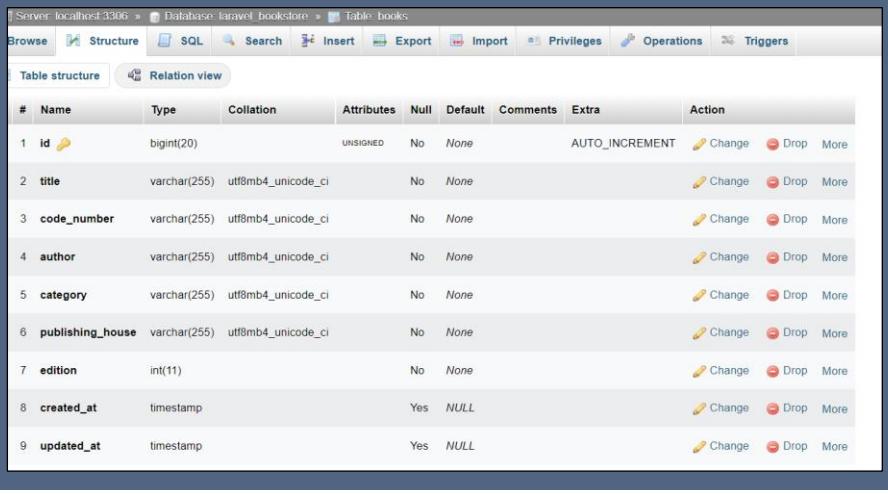
- Running Migrations

```
php artisan migrate
```

21

Migrations

- After that, we will see "books" table in "laravel_bookstore" database.



22

Seedings

- Create your database with data using seed classes
- From *DatabaseSeeder* class, you may use the call method to run other seed classes
- Generate a seeder by artisan command

```
php artisan make:seeder BookSeeder
```

- You will find your newly created file in *database/seeders* folder.

23

Seedings

- Add dummy data in *run* function as follows.

```
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Str;

    public function run()
        {
            DB::table('books')->insert([
                'title' => Str::random(20),
                'code_number' => Str::random(6),
                'author' => Str::random(10),
                'category' => Str::random(10),
                'publishing_house' => Str::random(10),
                'edition' => 1,
            ]);
        }
```

24

Seedings

- Call seeder class in database/seeds/DatabaseSeeder.php.

```
$this->call(BookSeeder::class);
```

- Running Seeders

```
php artisan db:seed
```

- After that, we will see dummy data in "books" table.

id	title	code_number	author	category	publishing_house	edition	created_at	updated_at
1	Z5ZhsfbxLDFgxvlCn6NU	S7iOq3	DSzyE3tA1d	qo6kksTNRL	l8HFfQBVl2	1	NULL	NULL

25

Eloquent Models

- Eloquent is Laravel's ORM, an object-relational mapper (ORM) that makes it enjoyable to interact with you.
- Eloquent Model represents database entities and can be used to query.
- Eloquent Model allow to retrieve records from the database table and then you can insert, update, and delete records to the table as well.

- Create a model for books table using make:model artisan command

```
php artisan make:model Book
```

- You will find your newly created file in *app/Models* folder.

26

Eloquent : Relationships

- Relationships are used to connect tables. Eloquent provides way to connect their models through eloquent relationships.
- Eloquent will bind the models so you will have to use functions.
- Eloquent makes managing and working with these relationships easy, and supports a variety of common relationships:
  - One To One
  - One To Many
  - Many To Many

27

Eloquent : One to One Relationship

- A one-to-one relationship is a very basic type of database relationship.
- For example, a User model might be associated with one Phone model.

```
class User extends Model
{
    public function phone()
    {
        return $this->hasOne(Phone::class);
    }
}
```

```
class Phone extends Model
{
    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

28

Eloquent : One to Many Relationship

- A one-to-many relationship is used to define relationships where a single model is the parent to one or more child models.
- For example, a blog post may have a lot of comments.

```
class Post extends Model
{
    public function comments()
    {
        return $this->hasMany(Comment::class);
    }
}
```

```
class Comment extends Model
{
    public function post()
    {
        return $this->belongsTo(Post::class);
    }
}
```

29

Eloquent : Many to Many Relationship

- Many-to-many relations are slightly more complicated than *hasOne* and *hasMany* relationships.
- An example of a many-to-many relationship is a user that has many roles and those roles are also shared by other users in the application.
- To define this relationship, three database tables are needed: users, roles, and role_user.

```
class User extends Model
{
    public function roles()
    {
      return $this->belongsToMany(Role::class);
    }
}
```

```
class Role extends Model
{
    public function users()
    {
        return $this-> belongsToMany(User::class);
    }
}
```

30

Database : Query Builder

- Laravel's database query builder provides a convenient, fluent interface to creating and running database queries.
- The Laravel query builder uses PDO parameter binding to protect your application against SQL injection attacks.

```
use Illuminate\Support\Facades\DB;
class BookController extends Controller
{
   public function index()
   {
        $books = DB::table(books')->get();

        foreach ($books as $book) {
                echo $book->title;
        }
    }
}
```

31

Eloquent: Collections

- Collections are much more powerful than arrays and expose a variety of *map* / *reduce* operations that may be chained using an intuitive interface.
- For example, we may remove all inactive users and then gather the first name for each remaining user:

```
$names = User::all()->reject(function ($user) {
        return $user->active === false;
})->map(function ($user) {
        return $user->name;
});
```

32

Session

- Sessions are used to store information about the user across multiple requests.
- Your application's session configuration file is stored at config/session.php.
- Laravel provides various drivers like **file, cookie, array, Redis,** and **database** to handle session data.
- Storing Session Data

```
$request->session()->put('key', 'value');
```

- Accessing Session Data

```
$value = $request->session()->get('key');
```

33

Middleware

- Middleware acts as a bridge between a request and a response.
- Middleware that verifies whether the user of the application is authenticated or not. If the user is authenticated, it redirects to the home page otherwise, if not, it redirects to the login page.

```
php artisan make:middleware RoleMiddleware
```

- The middleware that you create can be seen at **app/Http/Middleware** directory.

34

Authentication

- Authentication is the process of identifying the user credentials. In web applications, authentication is managed by sessions which take the input parameters such as email or username and password, for user identification. If these parameters match, the user is said to be authenticated.

35