DevOpsSchool
Lets Learn, Share & Practice DevOps

Laravel Advance Training

Curriculum 6 Days

Day - 1

- **Introduction**
  - What is Laravel?
  - MVC vs Non-MVC
  - MVC and the Structure of This Course
  - Setting up a Laravel Project
  - Understanding the Folder Structure
  - A Quick Intro to the Artisan Command Line Interface
- **Views & Blade Templating Engine**
  - Introduction
  - Views and Templating Engines
  - Why Use Templating Engines?
  - Blade - Displaying Data and Using Layouts
  - Demo: Using Layouts
  - Demo: Displaying Data
  - Blade - Partials
  - Demo: Partials
  - Blade - Control Structures
  - Demo: Control Structures
  - Blade - XSS Protection
  - Demo: XSS Protection
- **Working with Routes**
  - Introduction
  - The Laravel Request Flow
  - What's the Job of a Router?
  - Creating Routes
  - Demo: Project Routes and Views
  - Navigation and Route Names
  - Demo: Adding Navigation to the Project
  - Passing Route Parameters
  - Creating and Using POST Routes
  - Structuring Routes with Route Groups
- **Handling Requests & Responses**
  - Introduction
  - Handling Requests and Sending Responses
  - Demo: Extracting Data from a GET Request
  - Demo: Sending Data to a View
  - Handling POST Requests with Dependency Injection
  - Dependency Injection vs Facades
  - Demo: Using Dependency Injection
  - Protecting Against CSRF Attacks
  - Demo:CSRF Protection
  - Using Sessions to Show Temporary Data
  - A Case for User Input Validation
  - Demo: Implementing Input Validation
  - Demo: Showing Validation Errors
  - Demo: Outsourcing Error Messages into Partials

Day - 2

- **Configuration & Environments**
  - Configuration and Environments Overview
  - Configuring the Database Connection
- **Working with Routes**
  - Introduction
  - Controllers and Models - Overview
  - Demo: Creating the Post Model
  - Demo: Adding Methods to the Model
  - Demo: Creating the Post Controller
  - Connecting Controller Actions and Routes
  - How Laravel Finds the Controller
  - Alternative Route Syntax
  - Demo: Finishing the Post Model
  - Demo: Finishing the Controller
  - Using Validation in Controller
  - Finishing the Course Project

Day - 3

- **Databases - Introduction & Migration**
  - Databases Bird's Eye Overview
  - Migrations Overview
  - Creating and Running Migrations
- **Databases - Eloquent ORM**
  - Understanding Eloquent ORM Models
  - Models- Creating and Updating, Introducing Tinker
  - Models - Retrieving Single Model
  - Models- Retrieving Multiple Models and Collections Overview
  - Models- Using the Query Builder
  - Practical - List of Blog Posts and Single Blog Post
- **Forms (Markup, CSRF, Validation, Errors, Flash Messages, Mass Assignment)**
  - Forms Markup
  - Cross Site Request Forgery Explained
  - Forms - Storing Submitted Data
  - Forms - Input Validation
  - Forms - Displaying Validation Errors
  - Forms - Form Request Classes
  - Session Flash Messages
  - Forms - Old Input Helper
  - Forms/Models - Model Mass Assignment
- **CRUD(Editing, Updating and Deleting)**
  - CRUD - Edit Form
  - CRUD - Update Action
  - CRUD - Deleting Using Forms
  - CRUD - Deleting Models
- **Assets & Styling(Javascript,CSS, Laravel Mix, Bootstrap)**
  - Introduction to Laravel Mix
  - Installing Bootstrap
  - Using NPM and Compiling Assets with Mix/Webpack
  - Including Assets in Views
  - Versioned Assets (Cache Improvements)
  - Introduction to Bootstrap CSS
  - Understanding CSS Flexbox in General and in Bootstrap
  - Layout Grid and Styling Header Bar
  - Styling Forms
  - Styling Post List Page
  - Styling Single Post Page
  - Styling Flash Messages and Error Messages
- **Testing Basics**
  - Testing
  - Testing configuration and environment
  - Writing first function test
  - Testing database interactions
  - Testing store () action (model creation)
  - Testing for failure
  - Testing update () action (model updates)
  - Testing delete () action (model deletion)
- **One to One Eloquent Relations**
  - One to One relation with migration
  - One to One assigning relationship
  - One to One querying relationship
- **One to Many Eloquent Relations**
  - One to Many relation with migration
  - One to Many assigning relationship
  - One to Many querying relationship
- **Querying Basics**
  - Lazy Loading vs Eager Loading
  - Querying Relationship existence
  - Querying relationship absense
  - Counting related models
  - Using withCount() in practice (fetching count of comments) with test

- **Model Factories**
  - Model Factory Introduction
  - Model Factory States
  - Model Factory callbacks (after creating, after Making)
  - Application: Implementing comment list
- **Deleting Models and Soft Deletes**
  - Deleting related model using model events
  - Deleting related models using cascading
  - Soft deletes
  - Soft deletes querying
  - Restoring soft deleted model
  - Testing soft deleted models
- **Authorization, Policies, Gates**
  - Authorization introduction
  - Introduction to Gates
  - Using authorize () helper
  - Verifying permissions of the user
  - Admin users and overriding permissions
  - Policies introduction
  - Policy or Gate?
  - Verifying permissions in Blade templates
  - Using middleware to authorize routes
  - Application: updating tests
- **Authentication**
  - Authentication Overview
  - How user registration works in laravel
  - Guard component and how logging users in works
  - Custom registration from and Auth routes
  - Formatting validation errors
  - RedirectIfAuthenticated middleware
  - Log-in form with "Remember Me" feature
  - Logging out. Guest directive,debugging CSRF token errors
  - Retrieving the currently authenticated user
  - Protecting routes (requiring authentication)
  - Testing routes that requires
- **Laravel Blade Components**
  - Blade Components introduction
  - Component aliases
  - Conditional rendering in Component
  - Practical: creating reusable component for dates
  - Complicated example of conditional rendering
  - Application: Fixing an issue with HAVING clause
- **Database Seeding**
  - Refreshing database, database foreign keys and existing data
  - Problem: SQLite test database NOT NULL problem
  - Database seeding basics
  - Using Model Factory inside Seeder
  - Model relations inside seeder
  - Individual seeder classes
  - Making seeder interactive
- **Query Scopes - Local & Global**
  - Application: setting user_id for the new BlogPost
  - Global Query Scopes introduction
  - Global Query Scopes and potential issues
  - Local Query Scopes introduction
  - Practical: Local Query Scope - most commented posts
  - Practical: Local Query Scope - most active users
  - Practical: Local Query Scope - most active users last month
  - Practical: Global Query Scope - admin can see deleted posts

Day - 4

- **Caching**
  - Caching introduction
  - Laravel Debugbar
  - Storing data in cache
  - Removing from cache
  - Cache facade
  - Practical: using cache as storage
  - Practical: using cache for storage implementation
  - Using and setting up redis as cache storage
  - Cache tags introduction
  - Practical: using cache tags
- **Many to Many Eloquent Relations**
  - ManyToMany introduction
  - ManyToMany migration
  - Defining ManyToMany on models
  - Associating models in ManyToMany
  - Querying the ManyToMany relation and Pivot tables
  - Practical: displaying the list of tags using Blade component
  - Practical: list of blog posts by tag
  - Blade View Composers
- **Reusable Components, query Scopes, Route Model Binding)**
  - Practical: User to Comment OneToMany relation and migration
  - Practical: comments form and reusable errors component
  - Route Model Binding
  - Eager loading nested relationships
  - Converting repeating queries to query scopes
- **File Storage and Uploading**
  - File Storage introduction
  - File upload form
  - Handling file uploads
  - Using Storage facade to store files
  - Getting the URL of stored file
  - Practical: Image model, OneToOne relation and migrations
  - Uploaded image URL
  - Practical: Displaying uploaded image and styling
  - Deleting files
  - Validating uploaded files (size, type, dimensions)
- **Queues and Backgound Processing**
  - Queues and background processing
- **One to One Polymorphic Eloquent Relation**
  - Section intrduction
  - Practical: Scaffolding UserController and UserPolicy, using authorizeResource
  - Practical: Views for showing/editing user profile
  - OneToOne Polymorphic explained
  - OneToOne Polymorphic migration
  - OneToOne Polymorphic defining relation
  - OneToOne Polymorphic associating
  - Practical: OneToOne Polymorphic with BlogPost and Image
- **One to Many Polymorphic Eloquent Relation**
  - OneToMany Polymorphic explained
  - OneToMany Polymorphic migration & relation
  - OneToMany Polymorphic associating
  - Practical: OneToMany Polymorphic views
  - Practical: Running tests on MySQL database
  - OneToMany Polymorphic seeder
- **Localization(Translations)**
  - Localization introduction and demo
  - Configuring locale and translation overview

- View Composer with @include
- ManyToMany seeding

- **Many to many polymorphic Eloquent Relation**
  - ManyToMany Polymorphic explained
  - ManyToMany Polymorphic migration
  - ManyToMany Polymorphic relations
  - Understanding model Traits
  - Creating Taggable model trait

- **Sending E-mails**
  - Development setup for sending emails
  - The Mailable class explained
  - Rendering e-mail content and e-mail sending
  - Attaching files & data to e-mails
  - Embedding an image inside the e-mail
  - Markdown Mailable classes explained
  - Creating the Markdown Mailable class
  - Custom Markdown e-mail component and styling
  - Rendering e-mail previews in browse

- **Testing APIs**
  - Testing API GET methods, verifying JSON structure
  - Testing API GET methods, verifying JSON structure II
  - Test API storing (POST) resources, authentication and validation

introduction
  - Configuring queues, creating and running the first job
  - Optional e-mail queuing and execution delay
  - Dealing with failed jobs
  - Creating and dispatching custom jobs
  - Implementing custom job that dispatches other jobs
  - Rate Limiting queues
  - Named queues and prioritizing

- **Observers, Events, Listeners, Subscribers**
  - Model Observers
  - Events and Listeners
  - Practical: Custom Event and Listener
  - Second example of Event and Listener
  - Logging basics in Laravel
  - Handling built-in Laravel events with Subscriber

- **API Resources**
  - API Resources introduction
  - Serializing model relations using a Resource class
  - Limiting serialization only to eager loaded relations
  - Conditional serialization of properties

  - Translating plural forms, passing data
  - Storing translations in JSON
  - Translating the application
  - Storing user preffered language in database
  - Creating custom Locale Middleware
  - Adding language to URL through Apache configuration

- **Service, Service Container, Facades, Contracts**
  - What is a Service and creating a custom one
  - Service Container in practice
  - Basic Dependency Injection
  - Dependency Injection and Contracts
  - Contracts explained
  - Facades explained

- **Model serialization and Postman**
  - Postman - a quite long introduction
  - How Model serialization works
  - Hiding model attributes
  - Serializing model relations

- **API in Laravel**
  - API routes and controllers
  - Practice defining API routes
  - Returning a resource collection and response wrapping
  - Collection pagination

		○ Collection pagination and custom parameters ○ Storing a new resource ○ API Tokens explained and implemented ○ Returning one, updating and deleting resources ○ Handling 404 (resource not found) ○ API Authorization

Day - 5

- **Laravel socialite - social media login integration**
  - Introduction
  - Setup user registration and login
  - Important Laravel resources
  - Download socialite
  - Setup Laravel socialite
- **Facebook**
  - Fix curl error
  - Fixing Facebook https error with ngrok
  - Signup as a developer on Facebook
  - Curl error: unable to find local issuer certificate
  - Retrieve data from Facebook
  - Login with Facebook
- **Twitter**
  - Integrate twitter
  - Test twitter
- **GitHub**
  - Integrate GitHub
  - Test GitHub
- **GitHub**
  - Integrate GitHub
  - Test GitHub
- **Google**
  - Integrate google signup and test
- **Laravel Payment Processing Using the Best Payment Platforms**
  - Obtaining and preparing a Laravel project to process payments
  - Creating the essential models and tables for the payment platform
  - Preparing the payment platform and filling the database from Laravel
- **Preparing some visual components in Laravel to process payments**
  - Adding Laravel/ui to build some visual components
  - Generating visual components with Laravel/ui
  - Improving and using the components generated with Laravel UI
  - Adding components to display the messages from the payment platform
- **Building a payment generator to process from Laravel**
  - Creating a form with random amounts using Bootstrap in Laravel
  - Showing the types of currency available to process a payment
  - Showing the available payment platforms with Laravel and Bootstrap
  - Allowing to customize the form for each payment platform
  - Adding the actions to process a payment on the platform with Laravel
  - **Allowing to consume the API of any payment platform from Laravel**
  - Adding GuzzleHTTP to Laravel to consume HTTP APIs
  - Creating a component to use any API of the payment platforms
  - **Allowing to consume the API of any payment platform from Laravel**
  - Creating PayPal test accounts to send and receive payments
  - Creating a PayPal application to consume your API from Laravel
  - Configuring the payment platform in Laravel to use the PayPal API
  - Adding a service in Laravel responsible for consuming the PayPal API

Day - 6

- **Implementing the necessary actions in Laravel to use the PayPal**
  - Decoding and authenticating requests to the PayPal API
  - Creating an order in the PayPal API for a given amount and currency
  - Capturing a payment of an order given in Laravel with the PayPal API
  - Controlling from Laravel the creation of an order using PayPal
  - Controlling from Laravel the capture of a payment with PayPal
- **Improving and preparing Laravel to use any other payment platform**
  - Implementing the payment cancellation flow for any platform
  - Considering zero-decimal currencies
  - Creating a component that solves a payment platform dynamically
  - Resolving the payment platform service according to the user's choice
- **Preparing the project to use Stripe as an additional payment platform**
  - Creating an account in Stripe for use with the payment platform
  - Obtaining access credentials to use the Stripe API from Laravel
  - Configuring the payment platform with Laravel to use Stripe
  - Creating the service for Laravel that will use the Stripe API
- **Adding the essential elements in Laravel to start Chaining with Stripe**
  - Decoding and authenticating Stripe API requests
  - Requesting a payment method to make payments with Stripe from Laravel
  - Obtaining a token that represents the payment method in Stripe
- **Implementing the components to process payments with the Stripe API**
  - Decoding and authenticating requests to the PayPal API
  - Creating an order in the PayPal API for a given amount and currency
  - Capturing a payment of an order given in Laravel with the PayPal API
  - Controlling from Laravel the creation of an order using PayPal
- **Handling Strong Customer Authentication (SCA) with Stripe and Laravel**
  - Considering Strong Customer Authentication (SCA) with Stripe
  - Preparing the Stripe service to process payments with 3D Secure (SCA)
  - Performing 3D secure validation directly with Stripe
- **Conclusions and recommendations on payment processing from Laravel**
  - There is still much that can be done to process payments with Laravel
  - Considerations when publishing your payment platform to production
- **Laravel OTP Based Login Two Factor Authentication**