Laravel Documentation

This documentation provides a guide on various aspects of Laravel, a popular PHP framework for web application development.

Table of Contents

Installation Tutorial

This tutorial provides step-by-step instructions on how to install Laravel on your system.

1. **System Requirements**
2. **Installing Laravel**
   - Run the following command in your terminal or command prompt:

     ```
     composer require global laravel/installer
     laravel new project-name
     ```

   - Replace `project-name` with the desired name for your Laravel project.
3. **Configuring the Environment**
   - Rename the `.env.example` file to `.env`.
   - Open the `.env` file and configure the database connection settings according to your setup.
4. **Generating the Application Key**
   - Run the following command to generate a unique application key:

     ```
     php artisan key:generate
     ```

   - This key is used for encryption and other security purposes.

5. **Running the Development Server**
    - Start the development server by running the following command:

```
php artisan serve
```

    - This will start the server at `http://localhost:8000` by default.
6. **Verifying the Installation**
    - Open your web browser and visit `http://localhost:8000` to see the Laravel welcome page.

Simple CRUD

This tutorial demonstrates how to perform Create, Read, Update, and Delete operations in Laravel.

1. **Setting Up the Database**
    - Configure your database connection in the `.env` file.
2. **Creating a Model, Migration and Database Schema**
    - Run the following command to create a new model:

```
php artisan make:model -mcr Task
```

    - Replace `Task` with the desired name for your model.
3. **Creating Migration and Database Schema**
    - Run the following command to create a migration file for the tasks table:

```
php artisan make:migration create_tasks_table --create=tasks
```

    - Open the generated migration file and define the table structure and columns.
4. **Running Migrations**
    - Run the following command to migrate the database and create the tasks table:

```
php artisan migrate
```

5. **Creating Routes**
    - Open the `routes/web.php` file and define the routes for creating, reading, updating, and deleting tasks.

6. **Building the User Interface**
    - Create the necessary views and forms for creating, listing, updating, and deleting tasks.
7. **Implementing CRUD Functionality**
    - Write the logic in the controllers to handle the CRUD operations for tasks.
8. **Testing the CRUD Operations**
    - Test the create, read, update, and delete operations using the user interface.

Simple Authentication

This tutorial guides you on implementing user authentication functionality in your Laravel application.

1. **Installing Laravel's Authentication Scaffold**
    - Run the following command to install the authentication scaffolding:

      ```
      composer require laravel/breeze
      php artisan breeze:install
      ```

    - Choose between vue, react, api, or blade. For beginner, use blade
2. **User Registration**
    - Laravel's authentication scaffolding includes registration functionality out of the box.
3. **User Login**
    - Laravel's authentication scaffolding also includes login functionality.
4. **User Logout**
    - Implement a logout route and handle the logout functionality.
5. **Protecting Routes**
    - Use the `auth` middleware to protect routes that require authentication.
6. **Customizing the Authentication Logic**
    - Customize the authentication logic by modifying the generated controllers and views.

Simple File Upload and Storage Management

This tutorial explains how to handle file uploads and manage them in Laravel.

1. **Uploading Files**
    - Create a form with an input field of type `file` to allow users to upload files.

2. **Validating Uploaded Files**
    - Validate the uploaded files using Laravel's validation rules and handle validation errors.
3. **Storing Uploaded Files**
    - Use Laravel's `store` method to store the uploaded files in a designated storage location.
4. **Retrieving and Displaying Files**
    - Retrieve and display the stored files in your views using appropriate URLs.
5. **Deleting Files**
    - Implement functionality to delete files from storage when no longer needed.
6. **File Storage Options**
    - Configure and switch between different file storage options, such as local storage or cloud storage services.

Caching

This tutorial demonstrates how to use caching in Laravel to improve performance by storing frequently accessed data.

1. **Configuring the Caching Driver**
    - Set the desired caching driver in the `.env` file.
2. **Storing Data in the Cache**
    - Use Laravel's caching functions to store data in the cache.
3. **Retrieving Data from the Cache**
    - Retrieve and utilize the cached data in your application.
4. **Deleting Data from the Cache**
    - Remove specific data or clear the entire cache when necessary.
5. **Cache Tags**
    - Group related cached data using cache tags for easier management and retrieval.

Advanced Eloquent Query

This tutorial covers advanced techniques for querying and manipulating data using Laravel's Eloquent ORM.

1. **Querying with Conditions**
    - Use Eloquent's query methods to filter and retrieve data based on specific conditions.
2. **Advanced Querying with Joins and Subqueries**

- Perform more complex queries involving joins and subqueries.

3. **Eager Loading Relationships**
    - Optimize performance by eager loading related data to avoid the N+1 query problem.
4. **Working with Collections**
    - Utilize Laravel's collection methods for manipulating and transforming query results.
5. **Inserting, Updating, and Deleting Records**
    - Perform insert, update, and delete operations on records using Eloquent.
6. **Using Model Events and Observers**
    - Implement logic triggered by Eloquent model events and observers.

Logging

This tutorial explains how to log application events and messages using Laravel's built-in logging functionality.

1. **Configuration and Log Levels**
    - Configure the logging options and specify the desired log levels.
2. **Writing Log Messages**
    - Use Laravel's log functions to write messages to the log files.
3. **Logging Contextual Information**
    - Include contextual information in log messages for better troubleshooting.
4. **Creating Custom Log Channels**
    - Set up custom log channels for different parts of your application.
5. **Log File Rotation and Cleanup**
    - Configure log file rotation and cleanup to manage log file sizes.
6. **Viewing and Analyzing Logs**
    - Access and analyze log files to investigate issues and monitor application behavior.

Mail

This tutorial demonstrates how to send emails from your Laravel application using the Mail API.

1. **Configuring Email Drivers**
    - Set up the desired email driver in the `.env` file.

2. **Creating Mailables**
    - Generate mailable classes to define email content and recipients.
3. **Sending Basic Emails**
    - Use the mailables to send simple text-based or HTML-based emails.
4. **Sending Emails with Attachments**
    - Attach files to the emails and send them as attachments.
5. **Customizing Email Templates**
    - Design and customize email templates using Laravel's email components and variables.
6. **Queuing and Background Processing**
    - Queue email sending to improve performance and handle email processing in the background.

Notification

This tutorial guides you on sending notifications to users through various channels, such as email, SMS, and more.

1. **Creating Notification Classes**
    - Generate notification classes to define the content and channels for notifications.
2. **Sending Notifications to Users**
    - Dispatch the notifications to the desired users or recipients.
3. **Configuring Notification Channels**
    - Set up and configure notification channels, such as email, SMS, or database.
4. **Sending Email Notifications**
    - Use notifications to send emails to users.
5. **Sending SMS Notifications**
    - Configure and utilize notification channels to send SMS notifications.
6. **Broadcasting Notifications**
    - Broadcast notifications using real-time broadcasting services, such as Pusher or Redis.

Customizing Error Handling

This tutorial explains how to customize and handle errors in Laravel applications.

1. **Handling Exceptions**

- Create custom exception handlers to handle specific types of exceptions.

2. **Custom Error Pages**
    - Design and implement custom error pages for different HTTP error codes.
3. **Logging Errors**
    - Configure error logging to store error messages and details in log files.
4. **Error Handling for AJAX Requests**
    - Handle errors and send appropriate responses for AJAX requests.
5. **Error Reporting and Debugging**
    - Configure error reporting settings for different environments and enable debugging features.