MODULE – 7(Laravel)

Q1 How to pass Multiple Variable in route?

Laravel you can pass multiple variable in route using route parameter

1 define route with parameter:

You have to go routes folder open web.php in this file you can pass routes

2 create controller

You have to create controller for handle route

Q2 How to pass variable which can be null in Route?

In Laravel, you can define route parameters that can be optional (i.e., nullable) by providing a default value for the parameter in your route definition. Here's how you can pass a variable that can be null in a route:

1 Define Route with Optional Parameter:

In route folder open web.php file and define route with optional parameter by giving default value

2 Create Controller Method:

You have to create controller for handle route

3 Accessing Route Parameter:

Inside your controller method (show in this case), you can access the parameter passed in the URL as a function argument. Check if the parameter is null to handle the case where it's not provided.

Q3 How to remove route caching?

In Laravel, route caching improves the performance of route registration by consolidating all route registration operations into a single file, which is then loaded quickly by the framework. However, sometimes during development or when modifying routes frequently, you may need to remove route caching to reflect the changes immediately. Here's how you can remove route caching

1 Using Artisan Command:

Laravel provides an Artisan command to clear route cache. Open your terminal and run the following command:

php artisan route:clear

This command will remove the cached route file and force Laravel to recompile the routes the next time they are accessed.

2 Manually Deleting Cached Routes File:

If you prefer, you can manually delete the cached routes file. The default location of the cached routes file is bootstrap/cache/routes.php. You can delete this file directly from your filesystem.

rm bootstrap/cache/routes.php

3 Rebuild Cache After Changes:

After removing the route cache, Laravel will automatically rebuild the cache the next time your application is accessed. If you want to rebuild the route cache immediately, you can use the route:cache Artisan command:

php artisan route:cache

This command will re-cache your application's routes, improving the performance of route registration.

MODULE – 8(Migration)

Q1. How to do config database in Laravel

Ans . Laravel uses a configuration file name .env in located in the root directory in your Laravel app to manage some specific setting , including database configuration . make sure you have to set up your .env file with correctly

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=database name

DB\_USERNAME= database username

DB\_PASSWORD= database password

Laravel's database connection configuration is located in the config/database.php file.

Laravel is configured to use the MySQL database. You can find the MySQL configuration under the 'connections' => ['mysql' => [...]] section of this file. If you're using a different database, you'll find configurations for SQLite, PostgreSQL, SQL Server, etc.

Laravel provide migration system for creating table database. You can give commend

**Create new migration:**

Php artisan make:migration create\_table\_name.

**To run migration :**

Php artisan migrate.

**To rollback migrations:**

Php artisan migrate:rollback

**To create a model:**

Php artisan make:model ModelName

Q2. Call MySQLi Store Procedure from Laravel.

Ans . 1 **Create the Stored Procedure in MySQL**: First you have to created your stored procedure in your MySQL database. example

DELIMITER //

CREATE PROCEDURE GetUsers()

BEGIN SELECT \* FROM users;

END //

DELIMITER ;

2 **Call the Stored Procedure in Laravel**: Once your stored procedure is created, you can call it from your Laravel application using the DB facade.

class UserController extends Controller

{

public function getUsers()

{

// Call the stored procedure using DB facade

$users = DB::select('CALL GetUsers()');

// Process the result as needed

return $users;

}

}

In this example, **GetUsers()** is the name of the stored procedure. You call it using **DB::select()** and pass the SQL query as a string, where **CALL GetUsers()** invokes the stored procedure.

**3 Handle the Result**: The result returned from the stored procedure call will be an array of objects or arrays representing the rows returned by the stored procedure. You can process this result as needed in your Laravel application.

**4 Error Handling**: Ensure to handle any errors that might occur during the execution of the stored procedure. You can use Laravel's error handling mechanisms, such as try-catch blocks, to handle exceptions.

MODULE – 9, 10(Forms, Controls)

Q1 Explain ORM

ORM stands for Object-Relational Mapping. It's a programming technique used to convert data between incompatible type systems – specifically, between object-oriented programming languages and relational databases.

Here's a breakdown of what ORM entails:

Object-Oriented Programming (OOP):

In OOP, data and behavior are encapsulated into objects. Objects have attributes (data) and methods (behavior). This paradigm is commonly used in languages like Python, Java, and PHP.

Relational Databases:

Relational databases store data in tables consisting of rows and columns. Each row represents a record, and each column represents an attribute of that record. SQL (Structured Query Language) is typically used to interact with relational databases.

Object-Relational Mapping (ORM):

ORM bridges the gap between OOP and relational databases by allowing developers to work with database data in terms of objects and classes. Instead of writing raw SQL queries to interact with the database, developers can use ORM libraries or frameworks to perform CRUD (Create, Read, Update, Delete) operations using OOP concepts.

Key Concepts of ORM:

Mapping: ORM libraries/frameworks define mappings between database tables and classes/objects. Each table typically corresponds to a class, and each row in the table corresponds to an instance of that class.

Entities: In ORM, objects that are mapped to database records are called entities. Each entity represents a specific record in the database.

Relationships: ORM supports defining relationships between entities, such as one-to-one, one-to-many, and many-to-many relationships. These relationships are defined using associations between classes.

CRUD Operations: ORM libraries provide methods or APIs to perform CRUD operations on database entities without writing SQL queries explicitly. Developers can create, retrieve, update, and delete records using object-oriented methods and syntax.

Querying: ORM libraries often provide query building mechanisms to retrieve data from the database using object-oriented syntax. Developers can write queries using the ORM's query language or methods, which are then translated into SQL queries by the ORM library.

Benefits of ORM:

Productivity: ORM simplifies database interactions by abstracting away the complexities of SQL queries, allowing developers to focus on application logic rather than database management.

Portability: ORM makes it easier to switch between different database systems since the ORM library handles the database-specific details.

Maintainability: By using OOP principles, ORM promotes cleaner and more maintainable code, making it easier to understand and modify database-related code.

Security: ORM libraries often include features like parameterized queries and input validation to prevent SQL injection attacks and other security vulnerabilities.

Q2 Do Curd using Eloquent Query

Certainly! Let's go through each CRUD operation using Eloquent, Laravel's built-in ORM, which allows you to interact with your database using PHP syntax.

1. Create (INSERT):

To create a new record in the database using Eloquent, you typically instantiate a new instance of your model and then set its attributes before saving it to the database.

2 Read (SELECT):

To retrieve records from the database, you can use Eloquent's query builder methods or model methods.

3 Update (UPDATE):

To update records in the database, you retrieve the record, modify its attributes, and then call the save() method.

4 Delete (DELETE):

To delete records from the database using Eloquent, you can call the delete() method on a model instance or use the destroy method to delete records by their IDs.

Q3 Explain - Eloquent Relationships

Eloquent relationships in Laravel are a fundamental feature that allows you to define connections between different database tables/models. These relationships represent how data in one table/model is related to data in another table/model. Eloquent provides several types of relationships:

One-to-One: In a one-to-one relationship, each record in one table is associated with exactly one record in another table, and vice versa. For example, a User may have one Profile, and each Profile belongs to one User.

One-to-Many: In a one-to-many relationship, each record in one table can be associated with one or more records in another table, but each record in the second table is associated with only one record in the first table. For example, a User may have many Posts, but each Post belongs to only one User.

Many-to-One (Inverse of One-to-Many): This is the inverse of a one-to-many relationship. It's essentially a one-to-many relationship viewed from the other direction. For example, many Posts belong to one User, which is the inverse of the one-to-many relationship described above.

Many-to-Many: In a many-to-many relationship, each record in one table can be associated with one or more records in another table, and vice versa. For example, a User can have many Roles, and a Role can be assigned to many Users.

Q4

Eager loading and lazy loading are two strategies for retrieving related data in ORM frameworks like Laravel's Eloquent.

Eager Loading:

Eager loading is a technique used to load the related data along with the primary data in a single database query. This means that when you retrieve a model, its related models are also retrieved from the database in the same query. Eager loading helps reduce the number of database queries executed, improving performance, especially when dealing with relationships between multiple models.

In Laravel's Eloquent, you can eager load relationships using the with method.

Lazy Loading:

Lazy loading is a technique where related data is not loaded until it is explicitly requested. In other words, the related data is fetched from the database only when you try to access it. Lazy loading can lead to an increased number of database queries, especially if you access related data in a loop or multiple times within your code.

In Laravel's Eloquent, relationships are lazily loaded by default.

Comparison:

Eager Loading: Fetches all related data in a single query, reducing the number of database queries and improving performance. Suitable when you know you will need the related data upfront.

Lazy Loading: Fetches related data only when it is requested, which can lead to more database queries being executed, potentially impacting performance, especially in loops or when accessing related data multiple times. Suitable when you want to defer the loading of related data until it's needed.