**PHP - Laravel - Using Forms and Gathering Input-Industry**

MODULE : 9-10

**Laravel Assignment**

**Submitted To:-**

**Rajesh Nagar Sir**

**Submitted By:-**

**Ankita Sahu**

MODULE – 9, 10(Forms, Controls)

**1. Explain ORM?**

**Ans:-**

Eloquent is an Object Relational Mapper (ORM) that is included by default within the Laravel framework. An ORM is software that facilitates handling database records by representing data as objects, working as a layer of abstraction on top of the database engine used to store an application's data.

To get started, create an Eloquent model. Models typically live in the app directory, but you are free to place them anywhere that can be auto-loaded according to your composer.json file. All Eloquent models extend Illuminate\Database\Eloquent\Model.

**Defining An Eloquent Model**

class User extends Model {}

You may also generate Eloquent models using the make:model command:

php artisan make:model User

**2. Do Curd using Eloquent Query?**

**Ans:-**

In Laravel, Eloquent is the default ORM (Object-Relational Mapping) that allows you to interact with your database tables using an expressive syntax. To perform CRUD (Create,Update, Delete) operations using Eloquent, you typically work with Eloquent models.

Assuming you have a model named **Post** that represents a "posts" table in your database, here's an example of how you can perform CRUD operations:

**1.** **Create (Insert):**

use App\Models\Post;

// Create a new post

$post = new Post();

$post->title = 'New Post';

$post->content = 'This is the content of the new post.';

$post->save();

**2. Update:**

$post = Post::find(1); // Replace 1 with the actual post ID

$post->title = 'Updated Post Title';

$post->content = 'Updated content for the post.';

$post->save();

Alternatively, you can use the **update** method for mass updates:

Post::where('id', 1)->update([

'title' => 'Updated Post Title',

'content' => 'Updated content for the post.',

]);

**3. Delete:**

$post = Post::find (1); // Replace 1 with the actual post ID

$post->delete();

You can also use the **destroy** method to delete a record by its ID:

Post::destroy(1); // Replace 1 with the actual post ID

**3. Explain - Eloquent Relationships.**

**Ans:-**

In conclusion, Eloquent Relationships are a powerful feature of Laravel that allows developers to easily work with related data. From one-to-one to many-to-many relationships, Eloquent provides a simple and intuitive syntax to define and query these relationships.

As a Laravel developer, mastering Eloquent relationships can greatly enhance your development workflow and make your code more efficient and readable. If you’re interested in learning more about Laravel, Kinsta has various resources available, including a [tutorial](https://kinsta.com/blog/laravel-tutorial/) on getting started with Laravel and an article on [Laravel developer salaries](https://kinsta.com/blog/laravel-developer-salary/).

There are several types of relationships in Laravel's Eloquent:

**1. One to One:**

This relationship is defined using the hasOne and belongsTo methods.

**Example:**

// User model

public function phone()

{

return $this->hasOne('App\Phone');

}

// Phone model

public function user()

{

return $this->belongsTo('App\User');

}

**2. One to Many:**

This relationship is defined using the hasMany and belongsTo methods.

**Example:**

// Post model

public function comments()

{

return $this->hasMany('App\Comment');

}

// Comment model

public function post()

{

return $this->belongsTo('App\Post');

}

**3. Many to Many:**

This relationship is defined using the belongsToMany method.

**Example:**

// User model

public function roles()

{

return $this->belongsToMany('App\Role');

}

// Role model

public function users()

{

return $this->belongsToMany('App\User');

}

**4. What is Eager Loading and lazy loading?**

**Ans:-**

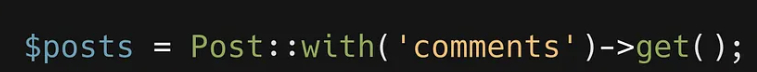
**Eager Loading:**

Eager loading is used to reduce the number of database queries needed to retrieve related data. It allows you to pre-load the corresponding models in a single query.

For example, if you have a Post model with many comments, you can use eager loading to get all the posts and the comments in one query, instead of two queries.

Let’s see an example:

This will get all posts and all the comments related to each post in one query.

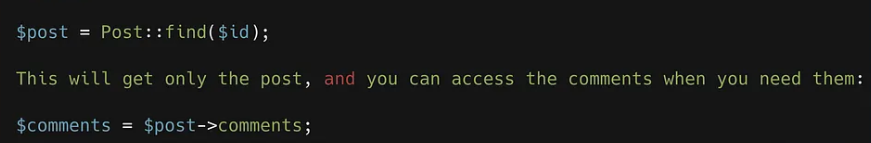
****

# Lazy Loading:

Lazy loading is the opposite of eager loading. It is used to defer loading related models until they are actually needed. It loads related models only when they are accessed by the application.

For example, if you have a Post model with many comments, you can use lazy loading to get only the post and defer loading the comments until they are actually needed.

Let’s see an example:

****

**XOXOXOXOXOXO**