

PHP - Laravel - Using Forms and

Gathering Input – Industry

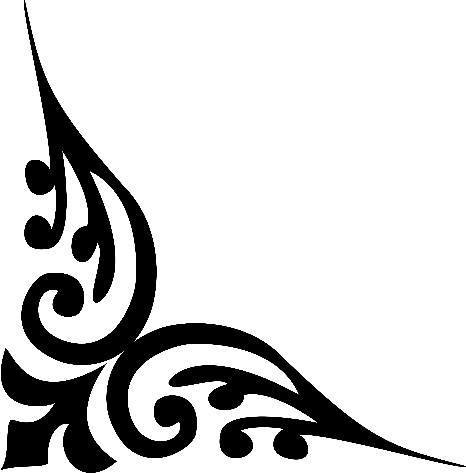
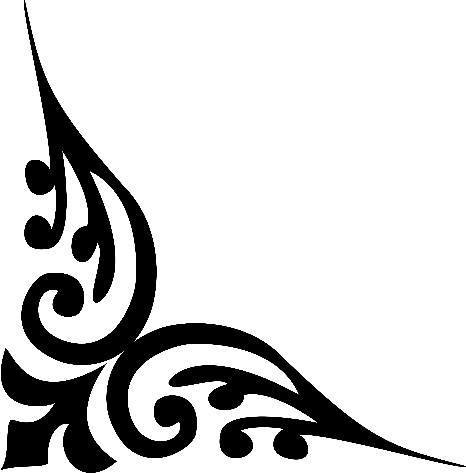
MODULE – 9, 10(Forms, Controls)

**Submitted to :-**

**Mr. Raj Nagar**

**Submitted by :-**

**Janvi Panchal**



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

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, you’ll need to set up the**[**Landing Laravel**](https://github.com/do-community/landing-laravel)**demo project. There are two ways in which you can get the demo application code ready to use with this series:**

1. **The first way is to follow the guide on**[**How To Build a Links Landing Page in Laravel**](https://www.digitalocean.com/community/tutorial_series/how-to-build-a-links-landing-page-in-php-with-laravel-and-docker-compose)**. That series explains how to build the demo application that you’ll use as base for the current series from scratch. If you choose this option, you can move on to the first tutorial in this series**[**How To Create a One-To-Many Relationship in Laravel Eloquent**](https://www.digitalocean.com/community/tutorials/how-to-create-a-one-to-many-relationship-in-laravel-eloquent)**.**
2. **The second option is to download the complete demo application code and use it as the base that you will build on in this series. On the application**[**releases page**](https://github.com/do-community/landing-laravel/releases)**, you’ll find separate application versions for each tutorial in the series. You can choose to start from the first tutorial by downloading version 0.1.1, or you can choose to download one of the**

**elo-tutorial releases that are paired with each individual tutorial in the series.**

1. **Do Curd using Eloquent Query**

**Ans:-**

**In Laravel, Eloquent is the ORM (Object-Relational Mapping) system that allows you to interact with your database tables using PHP syntax. To perform CRUD (Create, Read, Update, Delete) operations using Eloquent, you can follow these basic guidelines:**

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

**To insert data into a table using Eloquent, you can create a new instance of the model representing that table and then save it.**

**use App\Models\User;**

**$user = new User;**

**$user->name = 'John Doe';**

**$user->email = 'john@example.com';**

**$user->save();**

**Or you can use the ‘create’method:**

**User::create([**

**'name' => 'John Doe',**

**'email' => 'john@example.com',**

**]);**

1. **Update Data:**

**To update data, you can retrieve a record, change its attributes, and then save it:**

**$user = User::find(1);**

**$user->name = 'Updated Name';**

**$user->save();**

**You can also use the ‘update’ method:**

**User::where('id', 1)->update(['name' => 'Updated Name']);**

1. **Delete Data:**

**To delete a record, you can use the ‘delete’method:**

**$user = User::find(1);**

**$user->delete();**

**Or you can use the ‘destroy’ method:**

**User::destroy(1);**

1. **Explain - Eloquent Relationships**

**Ans:-**

**In Laravel, Eloquent is the Object-Relational Mapping (ORM) system included with the framework. Eloquent allows you to interact with your database tables using a fluent and expressive syntax in PHP. Eloquent relationships are a powerful feature that allows you to define relationships between different Eloquent models, making it easier to work with related data in your application.**

**There are several types of relationships you can define in Eloquent:**

**One-to-One Relationship:**

**Use the ‘hasOne’ and ‘belongsTo’ methods to define a one-to-one relationship between two models.**

**For example, if you have a ‘User’ model and an ‘Address’ model, you might define a one-to-one relationship where a user has one address.**

**// User model**

**public function address()**

**{**

**return $this->hasOne(Address::class);**

**}**

**// Address model**

**public function user()**

**{**

**return $this->belongsTo(User::class);**

**}**

**One-to-Many Relationship:**

**Use the ‘hasMany’ and ‘belongsTo’ methods to define a one-to-many relationship.**

**For example, if you have a ‘Post’ model and a ‘Comment’ model, you might define a one-to-many relationship where a post has many comments.**

**// Post model**

**public function comments()**

**{**

**return $this->hasMany(Comment::class);**

**}**

**// Comment model**

**public function post()**

**{**

**return $this->belongsTo(Post::class);**

**}**

**Many-to-Many Relationship:**

**Use the ‘belongsToMany’ method to define a many-to-many relationship.**

**For example, if you have a ‘User’ model and a ‘Role’ model, you might define a many-to-many relationship where a user can have multiple roles, and a role can belong to multiple users.**

**// User model**

**public function roles()**

**{**

**return $this->belongsToMany(Role::class);**

**}**

**// Role model**

**public function users()**

**{**

**return $this->belongsToMany(User::class);**

**}**

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

**Ans:-**

**Eager Loading:**

* **Eager Loading is a mechanism by which you can load a model and its relationships in a single query, instead of loading the main model and then querying the database again for each related model.**
* **This is particularly useful when you are dealing with relationships like "one-to-many" or "many-to-many," where loading related models separately for each main model could result in a large number of database queries.**
* **In Laravel, you can use the ‘with’ method to specify the relationships that should be eager loaded. This is commonly used with the ‘Eloquent’ ORM.**

**Example of Eager Loading in Laravel:**

**$posts = Post::with('comments')->get();**

**In this example, the with('comments') method tells Laravel to eager load the comments relationship for each retrieved post.**

**Lazy Loading:**

* **Lazy Loading is the opposite concept. It means that related models are only loaded from the database when you actually access the property representing the relationship.**
* **By default, Eloquent loads relationships lazily. This means that if you have a model with a relationship and you don't access that relationship, the related data won't be loaded until you do.**
* **While lazy loading can be convenient, it can also lead to the N+1 query problem. This occurs when you retrieve a collection of models and then loop through the collection, accessing a relationship for each model. This results in a separate query for each relationship, which can lead to performance issues.**

**Example of Lazy Loading in Laravel:**

**$post = Post::find(1);**

**$comments = $post->comments; // Comments are loaded when accessed**

**In this example, the comments relationship is not loaded until you access the ‘$post->comments’ property.**