* **Explain ORM**

Models are class based php files

Laravel Includes Eloquent ORM (Object-Relational Mapping) that makes it enjoyable to interact with your database

A Model in Laravel 8 provides **an abstraction for working with a database table with a high-level API**. ... Model events are simply hooks into the important points of a model's life cycle which you can use to easily run code when database records are saved, updated or deleted.

A Model is **basically a way for querying data to and from the table in the database**. Laravel provides a simple way to do that using Eloquent ORM (Object-Relational Mapping). Every table has a Model to interact with the table

Before Use you have to know that :

**Map Database table with Class Name means same name of table with S & model name without S so model automatically connect table**

Exa DB Table(Plural) Model Name(simular)

users user

employees employee

Note : If not use that types of rule means

If table name emp and model name also emp then that types of case you have to configure some manually

Add in model page class :

**public $table="emp";** // if table & model name not same then use this

**public $primarykey="emp\_id";** // if want custom primary key in table

//If don’t want then add : created\_dt and update\_dt

**public $timestamps=false** in user model class

**Make Model**

(Model name is capitalised and Below laravel 7 Model folder not available and user model also)

**=>php artisan make:model User**

**=>php artisan make:model User -- migration // with migration file**

**=>php artisan make:model User -c with controller**

**Import Model** in Controller

**=>use App\Models\User; // add model for use model functionality**

**Fetch Data by model in Mysql**

Function fetchdata()

{

$data= User::all(); // fetch all data from users table

return view('viewusermodel',['data'=>$data]); // load data in views

//return view('viewusermodel',compact(‘data’)); // load data in views convert in array

}

**user.blade.php**

@foreach ($data as $items)

<tr>

<td>{{$items->uid}}</td>

<td>{{$items->unm}}</td>

<td>{{$items->pass}}</td>

<td>{{$items->email}}</td>

<td>{{$items->mobile}}</td>

</tr>

@endforeach

Or

@foreach ($data as $items)

<tr>

<td>{{$items[‘uid’]}}</td>

<td>{{$items[‘unm’]}}</td>

<td>{{$items[‘pass’]}}</td>

<td>{{$items[‘email’]}}</td>

<td>{{$items[‘mobile’]}}</td>

</tr>

@endforeach

* **Do Curd using Eloquent Query**

CRUD operations under the Eloquent object-relational mapper (ORM) make it easier for Laravel developers to work with multiple databases. It performs create, retrieve, update, and delete (CRUD) operations, as well as maps object models to database tables. It handles all the database interaction required for CRUD operations.

**Creating records**

You can use the ::create method to insert a new record in the database.

student\_record::create(array(

'first\_name' => 'John',

'last\_name' => 'Doe',

'student\_rank' => 1

));

Aside from the simple create method shown above, you can also create a new object and assign different attributes to it. Then, you can call the save() function and execute the code. Methods such as firstOrCreate() or firstOrNew() are other options for creating records. These will enable finding a student with certain attributes; if that student is not found, then you will either create it in the database or instantiate a new instance.

**Retrieving records**

Using Eloquent ORM, getting and finding records from the database is manageable and easy. The queries are simply built and offer a smooth flow. For creating ::where statements, you will use get() and first() methods. The first() method will return only one record, while the get() method will return an array of records that you can loop over. Also, the find() method can be used with an array of primary keys, which will return a collection of matching records. Here are a few examples:

$student = Students::all();

This code gets all the students. While the code below, finds a specific student by id:

$student = Students::find(1);

Also, as shown below, the code describes finding a student based on a specific attribute.

$JohnDoe = Students::where('name', '=', 'John Doe')->first();

For the get() method, this code shows how to find a student with a rank level greater than 5.

$rankStudents = Student::where('student\_rank', '>', 5)->get();

**Updating records**

Updating records using Eloquent is as easy. To update a record, just find the record you would like to update, change the attributes, and save. For example, to change the student rank level of John Doe to 5, first find the student and then execute the save method.

$JohnDoe = Bear::where('name', '=', 'John Doe')->first();

$JohnDoe->danger\_level = 5;

$JohnDoe->save();

The save method may also be used to update models that already exist in the database.

**Deleting records**

Eloquent boasts its easy process of updating records, but it has the same story with deleting. There are two options: record pull-out and execute delete method, or simply use the destroy method. To find and delete a record, simply execute the following commands:

$student = Students::find(1);

$student->delete();

To delete a record and multiple records, the commands are executed:

Students::destroy(1);

Students::destroy(1, 2, 3);

Please note that the parameters of destroy are primary keys only unlike the delete method which can accept any database column.

To find and delete all students with rank level that is greater than 10.

Students::where('student\_rank', '>', 10)->delete();

Writing a web application in PHP, developers have the option to choose from a rich list of PHP frameworks. The ongoing demand combined with several usage statistics posted from communities suggest that Laravel is currently more popular than other PHP frameworks. However, seasoned web developers never choose a PHP framework based on its popularity or hype. There are pros and cons to consider. A lot of developers tend to downplay the popularity of PHP, but keep in mind that you must choose a PHP framework that fits all the project requirements. Also as a developer, it is important to use a PHP framework, as Laravel helps you reduce web application development cost.

The robust features and tools provided by Eloquent inside Laravel make it easier for developers to build custom web applications following specified business requirements. This article has given you a primer on how to use the basic features of Eloquent ORM. Keep in mind that some features often impact the performance of any Laravel application. Hence you have to implement a number of performance optimization techniques to boost the application’s speed and user experience. But Laravel has been evolving consistently to meet emerging web application development trends. You can always accelerate custom web application development by using the new features and enhancements included in the latest version of Laravel. Learn more about Eloquent and Laravel through its detailed documentation.

In a very competitive world of custom web applications, the health and well-being of the application are synonymous with business. This article discussed how Eloquent efficiently provide an effortless way to communicate with a database. The SQL queries from your application to the database are very critical. For example, a company may have a standard concerning the SQL query execution time. If a query is above 50 ms and according to their standard that is considered a slow query then a developer needs to perform optimization of the query as soon as possible. Time is very critical in this scenario, so a developer needs help in finding where and what causes the slow query. Thus Retrace can truly help developers by answering the where and what questions.

* **Explain - Eloquent Relationships**

Database tables are often related to one another. For example, a blog post may have many comments, or an order could be related to the user who placed it. Eloquent makes managing and working with these relationships easy, and supports several different types of relationships:

[One To One](https://laravel.com/docs/5.4/eloquent-relationships#one-to-one)

[One To Many](https://laravel.com/docs/5.4/eloquent-relationships#one-to-many)

[Many To Many](https://laravel.com/docs/5.4/eloquent-relationships#many-to-many)

[Has Many Through](https://laravel.com/docs/5.4/eloquent-relationships#has-many-through)

[Polymorphic Relations](https://laravel.com/docs/5.4/eloquent-relationships#polymorphic-relations)

[Many To Many Polymorphic Relations](https://laravel.com/docs/5.4/eloquent-relationships#many-to-many-polymorphic-relations)

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

Lazy and Eager are two types of data loading strategies in ORMs such as hibernate and eclipse Link.  These data loading strategies we used when one entity class is having references to other Entities like Employee and Phone (phone in the employee).

**Lazy Loading** − Associated data loads only when we explicitly call getter or size method.

Use Lazy Loading when you are using one-to-many collections.

Use Lazy Loading when you are sure that you are not using related entities.

**Egare Loading** − Data loading happens at the time of their parent is fetched.

 Use Eager Loading when the relations are not too much. Thus, Eager Loading is a good practice to reduce further queries on the Server.

Use Eager Loading when you are sure that you will be using related entities with the main entity everywhere.