Assignment 17: Query Builder in Larave

OSTAD.APP

Name: Md. Raqibul Islam Howlader

Phone: 01681533509

Submit Date: 12- 6- 2013

1.Explain what Laravel's query builder is and how it provides a simple and elegant way to interact with databases.

Answer:

Here are some key features and benefits of Laravel's query builder:

Fluent Interface: The query builder utilizes a fluent interface, allowing you to chain methods together to construct queries. This approach results in more readable and concise code compared to writing raw SQL statements.

Database Agnostic: Laravel's query builder supports multiple database systems, including MySQL, PostgreSQL, SQLite, and SQL Server. This means you can write queries that work across different database platforms without worrying about specific syntax or implementation details.

Parameter Binding: The query builder automatically handles parameter binding, which helps prevent SQL injection attacks. You can safely pass user input as query parameters, and the builder will handle escaping and sanitizing them.

Query Building Methods: The query builder provides a rich set of methods for constructing various types of queries. You can easily perform common operations like selecting columns, filtering rows with conditions, joining tables, sorting results, grouping data, and more.

Eloquent Integration: Laravel's query builder seamlessly integrates with Laravel's Eloquent ORM (Object-Relational Mapping) system. Eloquent provides an active record implementation, allowing you to map database tables to model classes and interact with them using expressive, object-oriented syntax.

Pagination and Eager Loading: The query builder includes convenient methods for paginating query results and eager loading relationships between models. These features enhance performance by retrieving data efficiently and reducing the number of database queries.

Overall, Laravel's query builder simplifies database interactions by offering an elegant and consistent API. It abstracts away the complexities of SQL, promotes code readability, and provides a flexible and database-agnostic solution for querying and manipulating data.

2.Write the code to retrieve the "excerpt" and "description" columns from the "posts" table using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

Answer:

```
use Illuminate\Support\Facades\DB;\

$posts = DB::table('posts')

   ->select('excerpt', 'description')

   ->get();

print_r($posts);
```

3.Describe the purpose of the distinct() method in Laravel's query builder. How is it used in conjunction with the select() method?

Answer:

```
use Illuminate\Support\Facades\DB;

$distinctPosts = DB::table('posts')

   ->select('category')

   ->distinct()

   ->get();

$distinctPosts = DB::table('posts')

   ->select('category')

   ->distinct()

   ->where('published', true)

   ->orderBy('created_at', 'desc')

   ->get();
```

4.Write the code to retrieve the first record from the "posts" table where the "id" is 2 using Laravel's query builder. Store the result in the $posts variable. Print the "description" column of the $posts variable.

Answer:

```
use Illuminate\Support\Facades\DB;

$posts = DB::table('posts')
   ->where('id', 2)
   ->first();
if ($posts) {
   echo $posts->description;
}
```

5.Write the code to retrieve the "description" column from the "posts" table where the "id" is 2 using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

Answer:

```
use Illuminate\Support\Facades\DB;

$posts = DB::table('posts')
   ->where('id', 2)
   ->pluck('description');
print_r($posts);
```

6.Explain the difference between the first() and find() methods in Laravel's query builder. How are they used to retrieve single records?

Answer:

The first() and find() methods in Laravel's query builder are both used to retrieve single records from a database table, but they differ in their approach and usage.

first() method:

The first() method retrieves the first record that matches the query conditions.

It returns a single object representing the first row found.

It is typically used when you need to retrieve the first record that satisfies certain conditions or when you don't need to specify the specific primary key value.

Example usage:

```
$post = DB::table('posts')
    ->where('published', true)
    ->orderBy('created_at')
    ->first();
```

find() method:

- The find() method retrieves a record by its primary key value.
- It assumes that the primary key column in the table is named id (though you can customize it).
- It returns a single object representing the row with the matching primary key value.
- It is commonly used when you want to retrieve a specific record by its primary key value.

Example usage:

```
$post = DB::table('posts')->find(1);
```

Here's a summary of the main differences between first() and find():

- first() is used to retrieve the first record that matches the query conditions, while find() is used to retrieve a record by its primary key value.
- first() requires specifying the query conditions using methods like where(), orderBy(), etc., whereas find() only requires passing the primary key value as an argument.
- first() returns the first matching record as an object, while find() returns the record with the specified primary key value as an object.
- first() is useful when you need to retrieve the first record that satisfies certain conditions or when you want to specify custom query conditions. find() is useful when you know the primary key value of the record you want to retrieve.

Both methods return null if no matching record is found.

7.Write the code to retrieve the "title" column from the "posts" table using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

Answer:

```
use Illuminate\Support\Facades\DB;

$posts = DB::table('posts')
    ->pluck('title');

print_r($posts);
```

8.Write the code to insert a new record into the "posts" table using Laravel's query builder. Set the "title" and "slug" columns to 'X', and the "excerpt" and "description" columns to 'excerpt' and 'description', respectively. Set the "is_published" column to true and the "min_to_read" column to 2. Print the result of the insert operation.

Answer:

```
use Illuminate\Support\Facades\DB;

$result = DB::table('posts')->insert([
    'title' => 'X',
    'slug' => 'X',
    'excerpt' => 'excerpt',
    'description' => 'description',
    'is_published' => true,
    'min_to_read' => 2,
]);

print_r($result);
```

9.Write the code to update the "excerpt" and "description" columns of the record with the "id" of 2 in the "posts" table using Laravel's query builder. Set the new values to 'Laravel 10'. Print the number of affected rows.

Answer:

```
use Illuminate\Support\Facades\DB;

$affectedRows = DB::table('posts')
   ->where('id', 2)
   ->update([
      'excerpt' => 'Laravel 10',
      'description' => 'Laravel 10',
   ]);
echo $affectedRows;
```

10.Write the code to delete the record with the "id" of 3 from the "posts" table using Laravel's query builder. Print the number of affected rows.

Answer:

```
use Illuminate\Support\Facades\DB;

$affectedRows = DB::table('posts')
   ->where('id', 3)
   ->delete();
echo $affectedRows;
```

11.Explain the purpose and usage of the aggregate methods count(), sum(), avg(), max(), and min() in Laravel's query builder. Provide an example of each.

Answer:

Here's a brief explanation of each aggregate method and an example of its usage:

count():

The count() method returns the number of rows that match the query conditions or the total number of rows in a table.

Example usage:

```
use Illuminate\Support\Facades\DB;

$count = DB::table('users')->count();
```

sum():

The sum() method calculates the sum of the values in a column.

Example usage:

```
use Illuminate\Support\Facades\DB;

$totalAmount = DB::table('orders')->sum('amount');
```

avg():

The avg() method calculates the average value of the values in a column.

Example usage:

```
use Illuminate\Support\Facades\DB;

$averagePrice = DB::table('products')->avg('price');
```

max():

The max() method returns the maximum value from a column.

Example usage:

```
use Illuminate\Support\Facades\DB;

$highestScore = DB::table('scores')->max('score');
```

min():

The min() method returns the minimum value from a column.

Example usage:

```
use Illuminate\Support\Facades\DB;
```

```
$lowestPrice = DB::table('products')->min('price');
```

12.Describe how the whereNot() method is used in Laravel's query builder. Provide an example of its usage.

Answer:

whereNot($column, $value)

```
$users = DB::table('users')
        ->whereNot('status', 'inactive')
        ->get();


$users = DB::table('users')
        ->whereNot('status', 'inactive')
        ->whereNot('role', 'admin')
        ->get();
```

13.Explain the difference between the exists() and doesntExist() methods in Laravel's query builder. How are they used to check the existence of records?

Answer:

exists()

```
$hasActiveUsers = DB::table('users')
                ->where('status', 'active')
                ->exists();
```

doesntExist()

```
$noInactiveUsers = DB::table('users')
                ->where('status', 'inactive')
                ->doesntExist();
```

14.Write the code to retrieve records from the "posts" table where the "min_to_read" column is between 1 and 5 using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

Answer:

```
$posts = DB::table('posts')
        ->whereBetween('min_to_read', [1, 5])
        ->get();
print_r($posts);
```

15.Write the code to increment the "min_to_read" column value of the record with the "id" of 3 in the "posts" table by 1 using Laravel's query builder. Print the number of affected rows.

Answer:

```
$affectedRows = DB::table('posts')
            ->where('id', 3)
            ->increment('min_to_read');
echo "Number of affected rows: " . $affectedRows;
```