Laravel Using Forms and Gathering Input

## Introduction

Laravel, one of the most popular PHP frameworks, offers powerful tools for building dynamic web applications. Forms and input handling are essential aspects of any web application, and Laravel simplifies these tasks through its robust features. Additionally, Laravel's ORM (Eloquent) and Query Builder provide developers with elegant ways to interact with databases.

## Creating Forms in Laravel

Forms are an integral part of web applications, allowing users to submit data. Laravel provides a streamlined way to create and manage forms through Blade templates and HTML helpers.

\*\*Example\*\*: Creating a simple registration form in Laravel:  
  
1. \*\*Blade Template\*\*:  
```html  
<form action="/register" method="POST">  
 @csrf  
<label for="name">Name:</label>  
<input type="text" id="name" name="name" required>  
  
<label for="email">Email:</label>  
<input type="email" id="email" name="email" required>  
  
<button type="submit">Register</button>  
</form>  
```  
  
2. \*\*Controller Method\*\*:  
```php  
public function register(Request $request)  
{  
 $validatedData = $request->validate([  
 'name' => 'required|string|max:255',  
 'email' => 'required|email|unique:users',  
 ]);  
  
 User::create($validatedData);  
 return redirect('/')->with('success', 'Registration successful!');  
}  
```  
  
3. \*\*Routes\*\*:  
```php  
Route::post('/register', [UserController::class, 'register']);  
```

## Gathering Input in Laravel

Laravel provides multiple ways to gather input from users. The most common methods include using the `Request` object and retrieving input data via helper functions.

\*\*Methods to Gather Input\*\*:  
1. \*\*Using Request Object\*\*:  
```php  
$name = $request->input('name');  
```  
2. \*\*Using `all()` Method\*\*:  
```php  
$data = $request->all();  
```  
3. \*\*Retrieving Specific Inputs\*\*:  
```php  
$email = $request->get('email');  
```  
4. \*\*Old Input Data\*\*:  
You can retrieve previously submitted data using the `old()` method:  
```php  
{{ old('name') }}  
```

## Validation

Laravel's validation system ensures that data submitted through forms meets the defined requirements. Validation can be performed using the `validate` method or custom form request classes.

\*\*Example\*\*:  
```php  
$validated = $request->validate([  
 'name' => 'required|string|max:255',  
 'email' => 'required|email|unique:users',  
]);  
```

## Laravel ORM (Eloquent)

Eloquent is Laravel's Object-Relational Mapper (ORM), which provides an elegant and intuitive way to interact with databases using models. Each model represents a table in the database, and objects of the model represent rows.

\*\*Features of Eloquent\*\*:  
1. Relationships: Define relationships between tables (e.g., one-to-many, many-to-many).  
2. Easy CRUD Operations: Perform create, read, update, and delete operations effortlessly.  
3. Query Scope: Use global or local scopes to reuse query logic.  
4. Model Events: Hook into events such as `creating`, `updating`, etc.

\*\*Example\*\*:  
```php  
class User extends Model  
{  
 protected $fillable = ['name', 'email'];  
}  
  
// Create a new user  
User::create(['name' => Ravi Mali', 'email' => 'ravimali@example.com']);  
  
// Retrieve all users  
$users = User::all();  
```

## Laravel Query Builder

The Query Builder provides a fluent interface to build and execute database queries. It is less abstract than Eloquent but offers more control and flexibility.

\*\*Features of Query Builder\*\*:  
1. Database-Agnostic: Works with multiple database systems.  
2. Fluent Interface: Chain methods for constructing complex queries.  
3. Raw Queries: Execute raw SQL for advanced use cases.

\*\*Example\*\*:  
```php  
// Retrieve users with a specific condition  
$users = DB::table('users')->where('active', 1)->get();  
  
// Insert a new user  
DB::table('users')->insert([  
 'name' => 'Ravi Mali',  
 'email' => ravimali­@example.com',  
]);  
```

## Comparison between Eloquent and Query Builder

1. \*\*Eloquent\*\*:  
- Provides an object-oriented approach.  
- Suitable for rapid development and when working with well-defined models.  
- Less control over SQL queries.  
  
2. \*\*Query Builder\*\*:  
- Provides a fluent interface for building queries.  
- Suitable for more complex queries or scenarios requiring fine control.  
- Doesn't require models.

## Conclusion

Laravel's tools for handling forms, gathering input, and interacting with databases make it a powerful framework for modern web development. Whether using Blade templates for forms, Eloquent ORM for managing models, or the Query Builder for constructing queries, Laravel simplifies and accelerates the development process. Understanding these features is essential for building efficient and maintainable applications.