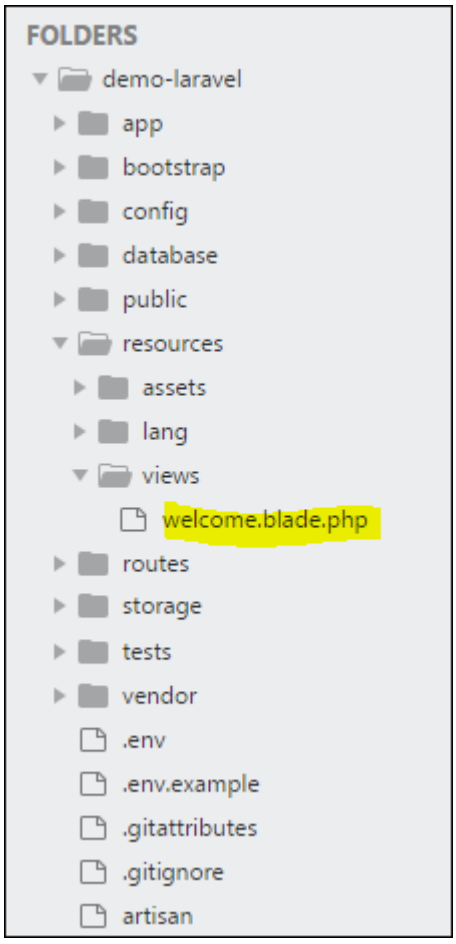
Laravel - Blade Templates

Laravel 5.1 introduces the concept of using **Blade**, a templating engine to design a unique layout. The layout thus designed can be used by other views, and includes a consistent design and structure.

When compared to other templating engines, Blade is unique in the following ways –

- It does not restrict the developer from using plain PHP code in views.
- The blade views thus designed, are compiled and cached until they are modified.



The complete directory structure of Laravel is shown in the screenshot given here.

You can observe that all views are stored in the **resources/views** directory and the default view for Laravel framework is **welcome.blade.php**.

Please note that other blade templates are also created similarly.

Steps for Creating a Blade Template Layout

You will have to use the following steps to create a blade template layout –

Step 1

- Create a layout folder inside the **resources/views** folder. We are going to use this folder to store all layouts together.
- Create a file name **master.blade.php** which will have the following code associated with it –

```
<html>
  <head>
    <title>DemoLaravel - @yield('title')</title>
  </head>
  <body>
    @yield('content')
  </body>
</html>
```

Step 2

In this step, you should extend the layout. Extending a layout involves defining the child elements. Laravel uses the **Blade @extends** directive for defining the child elements.

When you are extending a layout, please note the following points –

- Views defined in the Blade Layout injects the container in a unique way.
- Various sections of view are created as child elements.
- Child elements are stored in layouts folder as **child.blade.php**

An example that shows extending the layout created above is shown here –

```
@extends('layouts.app')

@section('title', 'Page Title')

@section('sidebar')

  @parent

<p>This refers to the master sidebar.</p>

@endsection

@section('content')

<p>This is my body content.</p>

@endsection
```

Step 3

To implement the child elements in views, you should define the layout in the way it is needed.

Laravel – Working With Database

Laravel provides a very simple and elegant way to work with databases. It supports multiple database systems and offers different methods to perform database operations easily and securely.

Databases Supported by Laravel

Laravel currently supports the following databases:

- MySQL
- PostgreSQL
- SQLite
- SQL Server

Laravel allows database queries using:

- Raw SQL queries
- Query Builder (Fluent)
- Eloquent ORM

To understand CRUD (Create, Read, Update, Delete) operations, a simple Student Management System is used as an example.

Connecting to Database in Laravel

To connect Laravel with a database:

1. Configure database details in the config/database.php file *(or using .env file in real projects)*
2. Create a database in MySQL named college

Database Structure

Database Name: College

Table Name: student

Column Name	Data Type	Extra
id	int(11)	Primary Key, Auto Increment
name	varchar(25)	—

CRUD Operations in Laravel Using DB Facade

Laravel provides the DB Facade to interact with the database.

---

1. Insert Records (Create)

Records can be inserted into the database using the insert() method.

Example:

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


DB::insert("insert into student (name) values (?)", ['Rahul']);
```

✅ This query inserts a new student record into the student table.

---

2. Retrieve Records (Read)

Records can be fetched using the select() method.

Example:

```
$students = DB::select("select * from student");


foreach ($students as $student) {
   echo $student->name;
}
```

✅ This retrieves all records from the student table.

---

3. Update Records (Update)

Records can be updated using the update() method.

Example:

```
DB::update("update student set name = ? where id = ?", ['Aman', 1]);
```

✅ This updates the name of the student whose id = 1.

4. Delete Records (Delete)

Records can be removed using the delete() method.

Example:

DB::delete("delete from student where id = ?", [1]);

✅ This deletes the student record with id = 1.