Routing

¿Qué pasa exactamente cuando hacemos una petición?

Antes de profundizar en los aspectos prácticos de laravel, veamos primero qué sucede exactamente cuando llega una solicitud.
Imaginemos que administra un restaurante de entrega de pizzas.

Cuando una persona visita URL de la página del restaurante antes de que se muestren las páginas, primero llegamos al servidor y cargamos su laravel. Inmediatamente, su aplicación se da cuenta de que, esta es una solicitud para la página de inicio.
Ahora en laravel, es realmente fácil asociar cualquier URI con una respuesta

Para notar en este caso si el usuario realiza una solicitud para la página de inicio, cargaremos un controlador y un controlador es la siguiente pieza del rompecabezas.

El controlador recibe una solicitud y proporciona una respuesta, por lo que en este caso, la solicitud es cargar la página de inicio de Larry's Pizza.

Cómo el controlador genera esa respuesta puede tomar muchas formas y hablaremos un poco sobre esto, pero por ahora digamos que el controlador debe delegar para obtener la información necesaria de su base de datos puede ser todas las pizzas, todos los especiales, todos los temas que podemos utilizar, lo que se conoce como un modelo elocuente.Para esto ahora, no solo el modelo proporciona una buena API para realizar cualquier cantidad de consultas SQL en su base de datos, sino que también es una ubicación para almacenar cualquier lógica de dominio o lógica de negocios específica.

Una vez que el controlador ha delegado en el modelo, el siguiente paso es cargar la vista.

Son las hojas del usuario en última instancia. Es la parte HTML de su código base, define la estructura, carga un archivo CSS, las referencias de JavaScript ahora lo más importante si recibió los datos del controlador.

Ahora lo más importante es que la vista recibe los datos del controlador y luego los procesa para el usuario.

Resumen

El controlador delega y carga toda la información necesaria para proporcionar una respuesta que recibe los datos del controlador y luego genera la estructura HTML para el usuario.

Setup a Database Connection

En la carpeta del proyecto se encuentra el archivo .env Aquí es donde almacena todas las claves y valores de configuración importantes, lo primero que debe entender son todas estas claves de base de datos, todos ellos serán referenciados dentro del archivo de configuración correspondiente.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=blog_alexamith
DB_USERNAME=root
DB_PASSWORD=
```

Cambio los valores a los cuales se adapte con lo que voy a trabajar.

Paso 1

```
Voy a la carpeta config
```

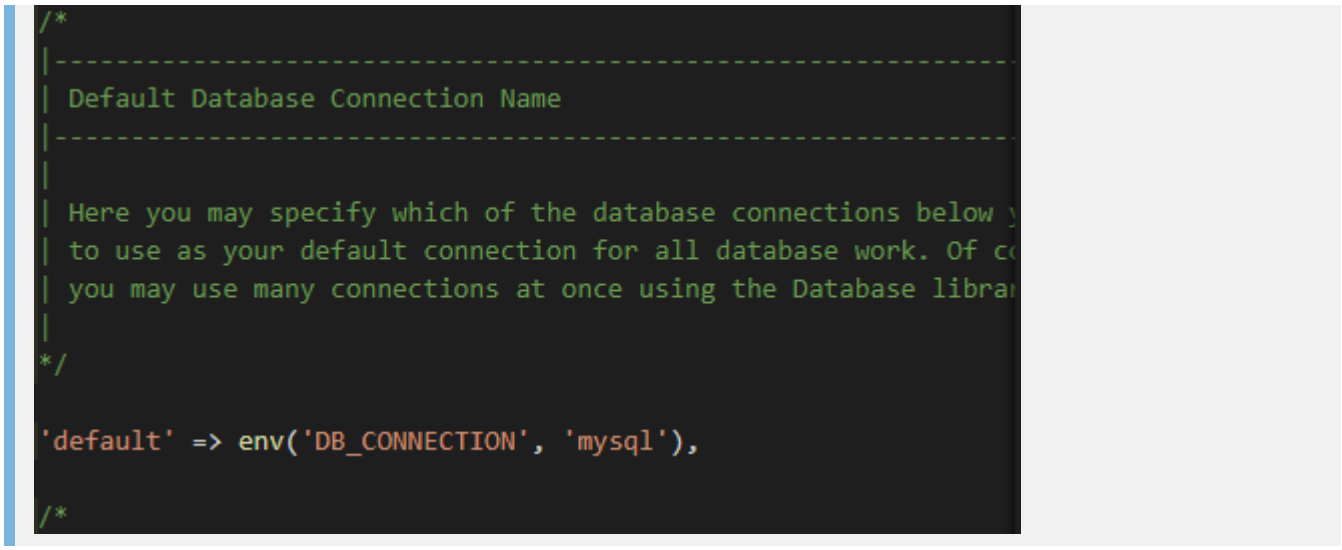
Paso 2

```
Abro el archivo database.php
```

Paso 3

```
Elijo y configuro la base de datos que vaya a usar
```

- Conexión de graves: estamos usando nuestra reconexión usando un adaptador MySQL o sqlite o postgres. Cuando este caso va a leer esa clave del archivo de entorno y si no existe, cambiaremos de forma predeterminada a mySQL.



- Aquí están todas las conexiones entre las que puede elegir, así que imaginemos que desea usar sqlite en su lugar, de acuerdo, volvería a su archivo de entorno y cambiaría la conexión, así que ahora está usando sqlite.

Crear la base de datos mySql

En la terminal lanzo sigo los siguients pasos.

- Me conecto a mysql con el comando

```
sudo mysql -u root
```

- Creo la bd con el nombre que coloqué en el archivo .env

```
create database laravel;
```

- Me conecto a la bd que `larevel`

```
use laravel
```

- Creo una tabla por ejemplo así.

```
CREATE TABLE persona (nombre VARCHAR(30), edad INT);
```

- Comprobar la tabla creada.

```
DESCRIBE persona;
```

- Añadir algunos registros.

```
INSERT INTO persona VALUES ('Jose Sanchez', 22);
```

- Comprobar los datos.

```
SELECT * FROM persona;
```

- Para salir de la consola de mySql

```
ctrl + c
```

Vistas

Las vistas se encuentran en la carpeta

```
resources/views
```

Puedo tener tantas vistas como yo quiera. También puedo tener mi vista dividida en muchas vistas.

Ejemplo

Si tengo que agregar el mismo css a todas las vistas eso quiere decir que tendría que incluír la misma etiqueta en todas las vistas donde quiera usar ese css general. Esto se resuelve así.

- Detro de esa carpeta agregamos un archivo llamado layout.blade.php
- Dentro de este archivo voy a crear la estructura básica de html y sólo le voy agregar las etiquetas de css.

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Laravel</title>

    <!-- Fonts -->
    <link href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=swa

    <style>
        body {
            font-family: 'Nunito';
            color:#679EFA;
        }

    </style>
</head>

<body class="antialiased">
</body>

</html>
```

- En otro archivo llamado welcome.blade.php, ahí sólo voy agregar los divs y demás elementos para la contrucción de una página html, sin agregar la estructura básica de html porque esa estructura ya la agregué en el paso anterior.

```
<div class="container">
    <h1 style="font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;">
    <h1 style="font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;">
    <hr>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo perferendis asperiores quos
    <h2>Soy ingeniero en software y trabajo en ciberseguridad</h2>
    <button class="btn btn-primary">Click here</button>
</div>
```

- ¿Cómo enlazo estos 2 archivos?

  En el archivo welcome.blade en la primera linea agreo la línea de código

  ```
  @extends('layout')
  ```

  Eso le indicará al archivo que pasará a obtener la estructura del archivo layout.blade

- Se debe indicar en el archivo `layout` todo lo que se va agregar queremos que esté dentro de la etiqueta `<body>`

```
<body class="antialiased">
    @yield ('content')
</body>
```

Y de esta manera puedo tener dividida la vista en tantos pedazos de código como yo quiera.

Forms

En el controlador al menos al principio, recomendaría que sigas esta convención, así que comentemos muy rápidamente. cada uno de estos haría la acción de índice como se aprendió claramente. Debería generar una lista de un recurso. La acción de mostrar debería mostrar un solo recurso. La acción de crear muestra de usted para crear un nuevo recurso y así sucesivamente.

Las 7 acciones en el controlador

The Seven Restful Controller Actions

```
class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;

    public function index(){
        // redirecciona una lista de x consulta
    }
    public function show($id){
        // Muestra un único elemento buscado por su id
    }
    public function create(){
        // Muestra una vista para crear un nuevo recurso
    }
    public function store(){
        // Persist the new resource
    }
    public function edit(){
        // Muestra una vista para editar un elemento existente
    }
    public function update(){
        // conservar el recurso editado
    }
    public function destroy(){
        // redirecciona una lista de x consulta
    }


}
```

Del lado del controlador estatal, sin embargo, todavía tenemos que averiguar cuál es la convención para la escritura porque podría pensar que escribo esto es simple si voy a editar un artículo existente, entonces probablemente sea algo como esto y tenga razón, ¿qué pasa con las actualizaciones? ha editado un artículo existente, sabe que desea realizar esa acción de actualización para saber qué debería ser. Me veo así. ¿Es algo como esto? No, no, en cambio, podemos usar el tipo de solicitud o el verbo de solicitud para canalizar estas solicitudes entrantes. Por ejemplo así

```
Route::get('/', function () {
    return view('welcome');
});

Route::update('update', function () {
    return view('welcome');
});
Route::delete('destroy', function () {
    return view('welcome');
});
```

Y así sucesivamente con las rutas que quiera!