LARAVEL

fecha de apuntes: 23 de marzo de 2020

Instalación

instalar laragon verificar si toma composer en Bash y los comandos de artisan en caso de no tomarlos se instala composer

Inicialización

para instalar laravel corremos el instalador desde la consola y luego para iniciar un proyecto nuevo se corre desde el terminal y dentro del servidor local

> **Inicio en consola**
>
> instalacion
>
> ```
> composer global require laravel/installer
> ```
>
> iniciar proyecto
>
> ```
> laravel new nombreProyecto
> ```

Rutas

las rutas son las url de entrada de la aplicacion, hay 2 tipos principales de rutas web y api.

ruta web

> **Notas**
>
> la carpeta que contiene todas las rutas de la aplicación se llama `routes` está ubicada en el directorio raiz
>
> para generar las rutas se hace a través de la llamada del `metodo`
>
> ```
> Route::get()
> ```

las rutas corresponden a un verbo del HTTP, que es el nombre del método que se invoca sobre la clase `Route`. El metodo recibe dos parámetros:

- *primer parámetro:* el patrón que debe cumplirse para que esa ruta se active `"/"` (el "home" de la aplicación), puede ser desde una simple cadena o más complejo generando partes de la ruta que sean parámetros variables.
- *segundo parámetro:* se indica una función con el código a ejecutar cuando laravel tenga que procesarla

> ✓ **Sintaxis de parametros del metodo**
>
> ('**parametro_patrón**', función anónima o **closure**)

> lo retornado en el closure es lo que se presenta para el usuario

para definir que la ruta responda al ingreso de la raiz se hace:

```
Route::get('/', function)
    //ejemplo
    creaando.com => Route::get('/', function)
    creaando.com/contacto = Route::get ('contacto', function)
```

los tipo de peticiones son:

> Route::get()
> Route::post() = para el envío de formulario // form action=POST
> Route::put()
> Route::patch()
> Route::delete()

verbos HTTP

Sirven para decir el tipo de acción que quieres realizar con un recurso (la URL), siendo posible especificar en el protocolo (la formalidad de la conexión por HTTP entre distintas máquinas) el verbo o acción que deseamos realizar.

Son 8 verbos en el protocolo: Head, Get, Post, Put, Delete, Trace, Options, Connect.

Get es la acción que se usa en el protocolo habitualmente, cuando se consulta un recurso. Sirve para recuperar información. Post por su parte es la acción que se realiza cuando se mandan datos, de un formulario generalmente. Los otros verbos no se usan de una manera muy habitual, pero sí se han dado utilidad en el desarrollo de lo que se conoce como API REST.

Verbos HTTP

significado de los verbos

- **Get**: recuperar información, podemos enviar datos para indicar qué se desea recuperar, pero mediante get en principio no se debería generar nada, ningún tipo de recurso en el servidor o aplicación, porque los datos se verán en la URL y puede ser inseguro.
- **Post**: enviar datos que se indicarán en la propia. Esos datos no se verán en la solicitud, puesto que viajan con la información del protocolo.
- **Put**: esto sirve para enviar un recurso, subir archivos al servidor, por ejemplo. No está activo en muchas configuraciones de servidores web. Con put se supone que los datos que estamos enviando son para que se cree algún tipo de recurso en el servidor.
- **Delete**: borrado de algo.
- ***Trace, patch, link, unlink, options y connect*** no están entre los verbos comunes de Laravel

Para tener en cuenta

observar las rutas posibles de la aplicación

el siguiente comando `artisan` sirve para mostrar todas las rutas disponibles en un proyecto o aplicacion con laravel

```
php artisan route:list
```

Rutas con parámetros

Las rutas en las aplicaciones web corresponden con patrones en los que en algunas ocasiones se encuentran textos fijos y en otras ocasiones textos que van a ser variables. en el presente se refiere a los textos variables con el nombre de parámetros y se pueden producir con una sintaxis de llaves.

para pasar paramentros en las urls, se usan las llaves `{}`

ejemplos de rutas con parámetros:

```
example.com/coloboradores/jose
example.com/coloboradores/luis
example.com/categoria/php
example.com/categoria/php/2
example.com/tienda/prodcutos/28
```

crear rutas con parámetros

en el siguiente ejemplo se crean unas de las rutas anteriores con el sistema de `routing` de laravel.

```
Route:get('colaboradores/{nombre}', function($nombre){
    return "mostrando el colaborador $nombre";
});

Route::get('tienda/productos/{id}', function($id_producto){
    return "Mostrando el producto $id_producto de la tienda";
});
```

Ejemplo Adicional

```
Route::get('saludo/{nombre}', function($nombre){
    return "saludo".$nombre;
});
```

Parámetros Opcionales

al no pasar parametros arroja error 404, si no se quiere parametro obligatorio, se asigna un signo de interrogación `(?)` y se da un valor por defecto en los parametros de la funcion

> Los parámetros opcionales se indican con un símbolo de interrogación `?`

Tomando como base del ejemplo anterior

```
example.com/categoria/php
example.com/categoria/php/2
```

En este código en el patrón de la URI se observa que la página es opcional.

```
Route::get('categoria/{categoria}/{pag?}', function($categoria, $pag = 1){
    return "Viendo categoría $categoria y página $pag";
});
```

Ejemplo Adicional

```
Route::get('saludo/{nombre?}', function($nombre = "invitado"){
    return "saludo".$nombre;
});
```

Precedencia de las rutas

Con dos rutas registradas que tienen patrones distintos, si por un casual una URI puede encajar en el formato definido por ambos patrones, el que se ejecutará será el que primero se haya escrito en el archivo `routes.php`.

Ejemplo de Precedencia

```
Route::get('categoria/{categoria}', function($categoria){
return "Ruta 1- Viendo categoría $categoria y no recibo página";
});

Route::get('categoria/{categoria}/{pagina?}', function($categoria, $pagina=1
    return "Ruta 2 - Viendo categoría $categoria y página $pagina";
});
```

En esas dos rutas tenemos patrones diferentes de URI, pero si alguien escribe:

```
example.com/categoria/laravel/
```

Esa URL podría casar con ambos patrones de URI. En este caso, el mensaje que obtendremos será:

```
Ruta 1 - Viendo categoría laravel y no recibo página
```

Aceptar determinados valores de parámetros

Hay una posibilidad muy útil con los parámetros de las rutas que consiste en definir expresiones regulares para especificar qué tipo de valores se acepta en los parámetros.

Por ejemplo, un identificador de producto debe ser un valor numérico o un nombre de un colaborador debe aceptar solamente caracteres alfabéticos.

Si se ha entendido esta situación se observará que hasta el momento, tal como hemos registrado las rutas, se aceptarían todo tipo de valores en los parámetros, generando rutas que muchas veces no deberían devolver un valor de página encontrada. Por ejemplo:

Ejemplo de valores en parámetros

```
example.com/colaboradores/666
example.com/tienda/productos/kkk
```

colaboradores debería ser un valor de tipo alfabético y el identificador de producto solo puede ser numérico.

Así que se debe modificar las rutas para poder agregarle el código que nos permita no aceptar valores que no deseamos.

```
Route::get('colaboradores/{nombre}', function($nombre){
    return "Mostrando el colaborador $nombre";
})->where(array('nombre' => '[a-zA-Z]+'));
```

Como se puede ver, se le coloca en cadena, sobre la ruta generada, un método `where()` que permite especificar en un *array* todas las reglas que se deben aplicar a cada uno de los parámetros a restringir.

Rutas con Nombre // named Routes

para una localizacion mas rapida y posterior edicion se le agregan nombres a las rutas

Ejemplo de Nombre de ruta

```
Route::get('contactanos', function(){
    return "sección de contactos";
})->name('contacto); //nombre de ruta
```

y se llama desde la ruta

```
Route::get('/', function(){
    echo "<a href='".route('contacto')."'>Contactos</a>";
})
```

vistas

Las vistas son una de las capas que tiene el sistema MVC, que trata de la separación del código según sus responsabilidades. En este caso, las vistas mantienen el código de lo que sería la capa de presentación.

Como capa de presentación, las vistas se encargan de realizar la salida de la aplicación que generalmente en el caso de PHP será código HTML. Por tanto, una vista será un archivo PHP que contendrá mayoritariamente

código HTML, que se enviará al navegador para que éste renderice la salida para el usuario.

> **Notas**
>
> las vistas se encuentran en: `"resources/views"`
>
> se usa la extension `".blade.php"` para hacer referencia a las vistas

por lo tanto la forma correcta de mostrar el html al usuario es retornando las vistas

para esto se invoca a la funcion `view()` dentro del `closure` y se le indica el nombre de la vista a mostrar

> **Ejemplo**

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

la funcion asume que las vistas se encuentran en la carpeta `"view"` y con extension `.blade.php`, por lo tanto no es necesario agregar la direccion completa `"resource/views/welcome.blade.php"`

> `view()` es una funcion global, un helper global en laravel, que se encarga de cargar una vista y devolver una salida producida por ella

pasar datos a las vistas

Para pasar datos a las vistas, se hace desde el archivo de rutas `web.php` o su equivalente en el proyecto

> **Ejemplo de datos en las vistas**
>
> por medio del parámetro `with`

```
Route:get('/', function(){
    $nombre = "jose";
    return view('home')->with('nombre', $nombre);
})->name('home');
```

> en forma de `array`

```
Route:get('/', function(){
    $nombre = "jose";
    return view('home')->with(['nombre' => $nombre]);
})->name('home');
```

array como segundo parámetro de la función `view()`

```
Route:get('/', function(){
    $nombre = "jose";
    return view('home', ['nombre' => $nombre]);
})->name('home');
```

con la funcion `compact`

```
Route:get('/', function(){
    $nombre = "jose";
    return view('home', compact('nombre')); //['nombre' => $nombre]
})->name('home');
```

devuelve el mismo array anterior siempre y cuando tenga el mismo nombre de la variable

otra forma: usando el metodo `view`

```
Route:view('/', 'home')->name('home');
```

tambien se le puede pasar la información por medio del array

```
Route:view('/', 'home', ['nombre' => $nombre]);
```