TINKER

La herramienta Tinker nos permite manejar nuestra base de datos desde una línea de comandos. Nos permitirá trabajar con los modelos de la aplicación, instanciando objetos y utilizando sus métodos.

Se arranca con artisan mediante el comando:

```
$ php artisan tinker
```

Con esta herramienta podremos

- Insertar o borrar datos en la base de datos
- Probar relaciones
- Mucho más

Lo que vamos a ver en este capítulo, es cómo utilizar Eloquent de Laravel desde la línea de comandos que ofrece la herramienta Tinker.

Podemos, con Tinker, probar nuestros comandos de acceso a datos, antes de incluirlos en nuestra aplicación.

CREANDO DATOS CON TINKER

Insertando un registro

Desde Tinker, vamos a insertar un registro en una tabla de nuestra base de datos.

Terminal

```
$ php artisan Tinker
> $post = App\Models\Post::create(['titulo'=>'Post sobre Tinker',
'contenido'=>'Este es el contenido']);
```

El comando, una vez ejecutado:

1. Insertará un registro en la tabla posts (Estoy utilizando el modelo Post)
2. Mostrará el resultado (guardado en la variable $post)

:

```
= App\Models\Post {#4044
    titulo: "Post sobre Tinker",
    contenido: "Cuento cosas de TInker",
    updated_at: "2023-01-13 09:48:49",
    created_at: "2023-01-13 09:48:49",
    id: 3,
  }
```

Como hemos guardado el resultado de “create” en la variable $post, podemos mostrarla desde Tinker posteriormente, de la forma:

Tinker en el Terminal

```
> $post
```

Y mostrará exactamente la misma información. Si vamos a PHPMyAdmin, veremos el registro, el número 3, con la información indicada en el comando:

id	titulo	contenido	created_at	updated_at	deleted_at
1	Lanzarote	Besitos desde Lanzarote MUA MUA	2023-01-05 20:25:38	2023-01-05 20:25:38	*NULL*
2	Prado	Prado Pinilla en junio!!	2023-01-05 20:25:38	2023-01-05 20:25:38	*NULL*
3	Post sobre Tinker	Cuento cosas de TInker	2023-01-13 09:48:49	2023-01-13 09:48:49	*NULL*

Creando un objeto del modelo para manipular la tabla

Podemos instanciar un objeto del modelo, y a partir de este objeto, gestionar la base de datos.
Vamos a ver un ejemplo:

Tinker en el Terminal

```
> $post = new App\Models\Post;
= App\Models\Post {#4661}

> $post->titulo = 'Vacaciones';
= "Vacaciones"

> $post->contenido = 'El año que viene vamos a Islandia';
= "El año que viene vamos a Islandia"

> $post
= App\Models\Post {#4661
    titulo: "Vacaciones",
    contenido: "El año que viene vamos a Islandia",
  }

> $post->save();
= true

>
```

Los comandos que escribo, en azul, son los siguientes:

1. Instancio un objeto del modelo correspondiente a la tabla posts (Post)
2. Asigno un valor al atributo titulo
3. Asigno un valor al atributo contenido
4. Muestro el objeto
5. Utilizo el método save para insertar en la BD

Si después vamos a PHPMyAdmin, podremos comprobar que el registro ha sido insertado:

id	titulo	contenido	created_at	updated_at	deleted_at
1	Lanzarote	Besitos desde Lanzarote MUA MUA	2023-01-05 20:25:38	2023-01-05 20:25:38	*NULL*
2	Prado	Prado Pinilla en junio!!	2023-01-05 20:25:38	2023-01-05 20:25:38	*NULL*
3	Post sobre Tinker	Cuento cosas de TInker	2023-01-13 09:48:49	2023-01-13 09:48:49	*NULL*
4	Vacaciones	El año que viene vamos a Islandia	2023-01-13 10:02:21	2023-01-13 10:02:21	*NULL*

Localizando registros. Condiciones.

Podemos utilizar nuestros modelos desde Tinker, para recuperar en un objeto los datos de un registro.

NOTA: En Laravel, podemos hacer referencia a la clase Post del modelo de dos formas:

- Indicando el espacio de nombres completo:
  `App\Models\Post`
- Mediante el método estático class:
  `Post::class`

Si os fijáis, al utilizar `::class`, Tinker encuentra la clase App\Models\Post, y le asigna un alias para esa sesión, llamado Post. De esta forma, es sencillo hacer referencia al modelo, sólo por Post, a partir de ese momento.

Tinker en el terminal

```
> $post = Post::class::find(1);
[!] Aliasing 'Post' to 'App\Models\Post' for this Tinker session.
= App\Models\Post {#4670
    id: 1,
    titulo: "Lanzarote",
    contenido: "Besitos desde Lanzarote MUA MUA",
    created_at: "2023-01-05 20:25:38",
    updated_at: "2023-01-05 20:25:38",
    deleted_at: null,
  }

> $post1 = Post::where('titulo','Prado')->first();
= App\Models\Post {#4672
    id: 2,
    titulo: "Prado",
    contenido: "Prado Pinilla en junio!!",
    created_at: "2023-01-05 20:25:38",
    updated_at: "2023-01-05 20:25:38",
    deleted_at: null,
  }
```

Los comandos anteriores realizan lo siguiente:

1. El primero instancia un objeto $post de la clase Post, guardando dentro el registro de la tabla cuya clave primaria (en este caso id) sea igual a 1. Así, $post contendrá los datos del registro 1, el post de Lanzarote.
2. El segundo, instancia $post1, en este caso recupera el primer registro de la tabla donde el título es igual a 'Prado'.

Eliminación y modificación de datos con Tinker

Para eliminar o actualizar un registro de una tabla desde Tinker.

Actualización:

Tinker en un terminal

```
> $post = Post::class::find(3);
[!] Aliasing 'Post' to 'App\Models\Post' for this Tinker session.
= App\Models\Post {#4670
    id: 3,
    titulo: "Post sobre Tinker",
    contenido: "Cuento cosas de TInker",
    created_at: "2023-01-13 09:48:49",
    updated_at: "2023-01-13 09:48:49",
    deleted_at: null,
  }

> $post->titulo = 'Tinker';
= "Tinker"

> $post->contenido='Tinker me deja borrar y actualizar registros!!';
= "Tinker me deja borrar y actualizar registros!!"

> $post->save()
= true

>
```

Los comandos anteriores (en azul) permiten actualizar un registro:

1. Instancio un objeto de la tabla posts y le asignan la información del registro con id=3
2. Modifico el atributo titulo
3. Modifico el atributo contenido
4. Con el método save, actualizo la base de datos. True indica que todo fue bien.

Si miramos la BD podremos comprobar que ha modificado el registro:

id	titulo	contenido	created_at	updated_at	deleted_at
1	Lanzarote	Besitos desde Lanzarote MUA MUA	2023-01-05 20:25:38	2023-01-05 20:25:38	NULL
2	Prado	Prado Pinilla en junio!!	2023-01-05 20:25:38	2023-01-05 20:25:38	NULL
3	Tinker	Tinker me deja borrar y actualizar registros!!	2023-01-13 09:48:49	2023-01-13 10:26:21	NULL
4	Vacaciones	El año que viene vamos a Islandia	2023-01-13 10:02:21	2023-01-13 10:02:21	NULL

Borrado

Para borrar un registro, como en la tabla posts tengo borrado lógico (campo deleted_at).

Siguiendo con el ejemplo, voy a realizar un borrado mediante Eloquent del registro. Como la tabla tiene configurado el borrado lógico, lo que hará Laravel será notificar una fecha de eliminación.

Tinker en Terminal

```
> $post->delete()
true
```

El comando anterior borra (de forma lógica) el registro de la tabla. Podemos comprobarlo posteriormente en PHPMyAdmin:

id	titulo	contenido	created_at	updated_at	deleted_at
1	Lanzarote	Besitos desde Lanzarote MUA MUA	2023-01-05 20:25:38	2023-01-05 20:25:38	NULL
2	Prado	Prado Pinilla en junio!!	2023-01-05 20:25:38	2023-01-05 20:25:38	NULL
3	Tinker	Tinker me deja borrar y actualizar registros!!	2023-01-13 09:48:49	2023-01-13 10:33:36	2023-01-13 10:33:36
4	Vacaciones	El año que viene vamos a Islandia	2023-01-13 10:02:21	2023-01-13 10:02:21	NULL

Borrado definitivo

Si queremos el borrado definitivo del registro, utilizamos la función forceDelete()

Tinker en Terminal

```
> $post->forceDelete()
true
```

En PHPMyAdmin podremos comprobar que ha quedado eliminado definitivamente:

id	titulo	contenido	created_at	updated_at	deleted_at
1	Lanzarote	Besitos desde Lanzarote MUA MUA	2023-01-05 20:25:38	2023-01-05 20:25:38	NULL
2	Prado	Prado Pinilla en junio!!	2023-01-05 20:25:38	2023-01-05 20:25:38	NULL
4	Vacaciones	El año que viene vamos a Islandia	2023-01-13 10:02:21	2023-01-13 10:02:21	NULL

Relaciones con Tinker

Vamos a buscar los roles de un usuario en nuestra base de datos.

Tinker en Terminal

```
> $user=User::class::find(1);
= App\Models\User {#4677
    id: 1,
    name: "Marta",
    email: "marta@eu.es",
    email_verified_at: "2023-01-05 20:23:24",
    #password: "",
    #remember_token: null,
    created_at: "2023-01-05 20:23:24",
    updated_at: "2023-01-05 20:23:24",
    country_id: 1,
  }

> $user->roles;
= Illuminate\Database\Eloquent\Collection {#3702
    all: [
      App\Models\Role {#4667
        id: 1,
        nombre: "Administrador",
        created_at: "2023-01-05 20:28:45",
        updated_at: "2023-01-05 20:28:45",
        pivot: Illuminate\Database\Eloquent\Relations\Pivot {#4659
          user_id: 1,
          role_id: 1,
          created_at: "2023-01-05 20:29:27",
        },
      },
      App\Models\Role {#3713
        id: 2,
        nombre: "Técnico",
        created_at: "2023-01-05 20:28:45",
        updated_at: "2023-01-05 20:28:45",
        pivot: Illuminate\Database\Eloquent\Relations\Pivot {#3711
          user_id: 1,
          role_id: 2,
          created_at: "2023-01-05 20:29:27",
        },
      },
    ],
  }
```

Los comandos anteriores (en azul), realizan lo siguiente:

1. Recupero el usuario con id=1
2. Obtengo los roles. Este atributo es generado por el método roles() del modelo Users.

Como puede verse, se retornan dos roles, en formato JSON (objeto con un array de otros dos objetos, cada uno de los cuales son datos de varias tablas, incluida roles y la tabla de relación n-n (pivote) llamada roles_user.

Podemos probar en Tinker todas nuestras relaciones, antes de incluirlas en nuestra aplicación.