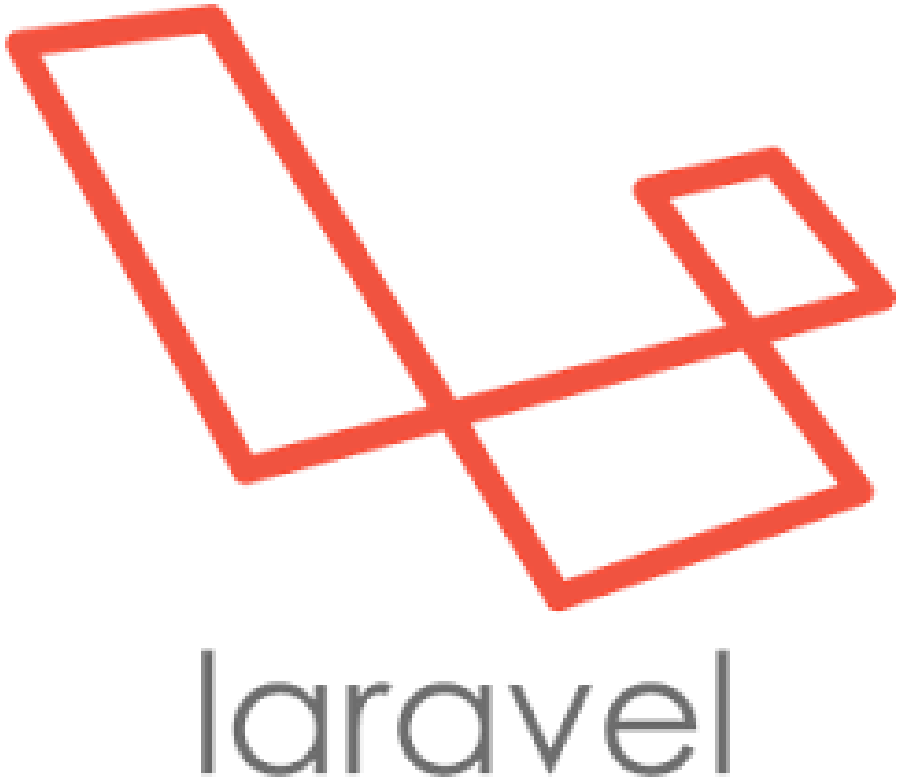
Curso de Laravel



Capítulo 6: Base de datos

Laravel: Base de datos

Configuración de la base de datos

- Laravel soporta distintos tipos de base de datos (MySQL, PostgreSQL, SQLite y SQL Server
- El archivo de configuración que refleja la configuración de la base de datos está en config/database.php
- El archivo de que envía la información a la configuración de laravel es el .env que se encuentra en el directorio raíz del proyecto

Laravel: Base de datos

Crear la base de datos

- Abrir una consola SQL y ejecutar la sentencia siguiente

```
CREATE DATABASE ‘db_name’ /*!40100 DEFAULT CHARACTER SET
utf8 */;
```

- O crear una base de datos con alguna herramienta de interfaz (phpMyAdmin, Workbench, Navicat u otro)

Laravel: Base de datos

Tabla de migraciones

- Laravel provee las migraciones, que sirven para definir y crear tablas mediante código y a la vez mantener un control de versiones de ellas.

- Primero se debe crear la tabla de migraciones con el siguiente comando:

```
php artisan migrate:install
```

Laravel: Base de datos

Nueva migración de creación de tabla

- Para crear una nueva migración se debe usar el comando de Artisan make:migration con el nombre del archivo y el nombre de la tabla:

```
php artisan make:migration create_users_table --create=users
```

- Este comando creará un archivo en database/migrations con el prefijo de timestamp, lo que permite identificar las versiones o el orden de ejecución de las migraciones

Laravel: Base de datos

Nueva migración de modificación de tabla

- Para crear una nueva migración que modifica una tabla también se usa el comando de Artisan make:migration con el nombre del archivo y el nombre de la tabla:

```
php artisan make:migration add_type_to_users_table --table=users
```

- Este comando creará un archivo en database/migrations con el prefijo de timestamp, pensando en añadir un campo sobre la tabla users

Laravel: Base de datos

Estructura de una migración

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up(){

    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down(){

    }
}
```

- La función up() crea o modifica la tabla
- La función down() deshace los cambios definidos en up()

Laravel: Base de datos

Ejecutar migraciones

- Para lanzar las migraciones creadas se debe ejecuar el siguiente comando:

```
php artisan migrate
```

- Este comando ejecutará el método up() de todas las migraciones creadas. Para deshacer los cambios el comando es el siguiente:

```
php artisan migrate:rollback
```

- Para deshacer todas las migraciones:

```
php artisan migrate:reset
```

Laravel: Base de datos

Ejecutar migraciones

- Para deshacer todos los cambios y volver a aplicar las migraciones en la base de datos:

```
php artisan migrate:refresh
```

- Para comprobar el estado de las migraciones

```
php artisan migrate:status
```

Laravel: Base de datos

Schema Builder

- Crear una tabla

```
Schema::create('users', function (Blueprint $table) {
    $table->increments('id')->unique;
   $table->string('name')->default(null);
});
```

Los tipos de datos soportados están disponibles en:
http://laravel.com/docs/5.5/migrations#creating-columns

- En la sección down de la migración se implementa la

```
Schema::drop('users');
Schema::dropIfExists('users');
```

Laravel: Base de datos

Schema Builder

- Añadir índices

```
$table->primary('id'); //Añadir una clave primaria
$table->primary(array('first','last')); //clave primaria
compuesta
$table->unique('email'); //Definir el campo como UNIQUE
$table->index('state'); //Añadir un índice a una columna
```

- Claves foráneas

```
$table->integer('user_id')->unsigned();
$table->foreign('user_id')->references('id')->on('users');
```

Laravel: Base de datos

Schema Builder

- Acciones en claves foráneas

```
$table->foreign('user_id')
      ->references('id')->on('users')
      ->onDelete('cascade');
```

- Eliminar claves ajenas en método down()

```
$table->dropForeign('posts_user_id_foreign');
```

Laravel: Base de datos

Carga de datos de prueba (seeds)

- Para cargar datos de prueba se usa el archivo database/seeds/DatabaseSeeder.php

```php
<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {

    }
}
```

Laravel: Base de datos

Carga de datos de prueba (seeds)

- Implementación del método run()

```php
public function run()
{
    //Borramos los datos de la tabla
    DB::table('users')->delete();
    //Añadimos una entrada a esta tabla
    User::create(array('email'=>'foo@bar.com'));
}
```

- Inicializar los datos

```
php artisan db:seed
```

Laravel: Base de datos

Constructor de consultas (Query Builder)

- Ejemplo de select clásico de Laravel

```php
$users = DB::table('users')->get();
foreach($users as $user)
{
   echo $user->name;
}
```

- Obtener un solo elemento con first()

```php
$user = DB::table('users')->first();
echo $user->name;
```

Laravel: Base de datos

Constructor de consultas (Query Builder)

- Cláusula where

```
$user = DB::table('users')->where('name','Pedro')
                          ->get();
echo $user->name;
```

- Operadores >, =, LIKE

```
$users = DB::table('users')->where('votes','>',100)
                           ->get();
$users = DB::table('users')->where('var','<>','ok')
                           ->get();
$users = DB::table('users')->where('name','like','T%')
                           ->get();
```

Laravel: Base de datos

Constructor de consultas (Query Builder)

- Cláusula and / or where

```
$users = DB::table('users')
->where('votes', '>', 100)
->orWhere('name','Pedro')
->get();
```

- Order By, Group By, Having

```
$users=DB::table('users')
->orderBy('name','desc')
->groupBy('count')
->having('count','>',100)
->get();
```

Laravel: Base de datos

Constructor de consultas (Query Builder)

- Offset / Limit (skip() para el offset y take() para el limit)

```
$users = DB::table('users')->skip(10)->take(5)->get();
```

- Transacciones

```
DB::transaction(function()
{
    DB::table('users')->update(array('votes'=>1));
    DB::table('posts')->delete();
});
```

Laravel: Base de datos

Constructor de consultas (Query Builder)

- Ver Query resultante

```
$users=DB::table('users')
->orderBy('name','desc')
->groupBy('count')
->having('count','>',100)
->toSql();

dd($users);
```

- Más información:
  https://laravel.com/docs/5.5/queries