Database seeding

The essentials

What is Database Seeding?

Wikipedia defines database seeding as follows:

> ***Database seeding*** *is populating a database with an initial set of data. It's common to load seed data such as initial user accounts or dummy data upon initial setup of an application* (https://en.wikipedia.org/wiki/Database_seeding).

Populating a database with initial data is a common practice in software development for various reasons. Here are three examples that illustrate why it is necessary:

1. **Demonstrations and Presentations:**
    - **Scenario:** Developers often need to showcase the capabilities of their applications to stakeholders, clients, or team members.
    - **Need for Initial Data:** To provide a meaningful and convincing demonstration, the application may need to be preloaded with relevant data that demonstrates its features and functionality.
    - **Example:** For a project management application, having sample projects, tasks, and user data can help present the application's project tracking and collaboration features effectively.
2. **Seed Data for New Installations:**
    - **Scenario:** When a new instance of the software is installed or when a new user account is created, the application needs to provide a basic set of data to ensure a functional starting point.
    - **Need for Initial Data:** Without initial data, the application might be empty or lack essential elements, making it less intuitive for users and hindering their ability to explore and use the system effectively.
    - **Example:** In a content management system (CMS), initializing a new installation with sample articles, categories, and user roles can help users understand the structure and capabilities of the system right from the start.
3. **Application Testing and Quality Assurance:**
    - **Scenario:** Before releasing a software application to the public, thorough testing is essential to ensure its functionality, performance, and reliability.
    - **Need for Initial Data:** During testing, developers and QA engineers need a realistic environment that simulates how the application will behave in real-world usage. This includes having a database with representative data to test the application's features, workflows, and potential edge cases.
    - **Example:** If you're developing an e-commerce application, you might need initial data that includes various products, categories, and user profiles to test the entire purchasing process.

In these examples, populating the database with initial data is crucial for creating realistic testing environments, providing meaningful demonstrations, and ensuring a smooth start for users interacting with the software for the first time.

Seeding Databases in Laravel

Laravel provides a convenient way to define and execute seeders using the Artisan command-line tool. It is common practice to create a different seeder for each table you need to seed. Then you can call each seeder or manage a default seeding procedure with the `DatabaseSeeder` class.

Creating a Seeder

To create a new seeder, you can use the `make:seeder` Artisan command. For example:

```
php artisan make:seeder CitySeeder
```

This command will generate a new seeder class file in the `database/seeders` directory. Note that Laravel naming conventions here: model name followed by 'Seeder' and all in PascalCase.

Defining Seeder Logic:

Open the generated seeder file (e.g., `CitySeeder.php`) and define the logic for populating your database table. You can simply use Eloquent models to interact with the database.

```
use Illuminate\Database\Seeder; use App\Models\City; class CitySeeder extends Seeder { public function run() { $city = new City(); $city->name = 'Nairobi'; $city->save(); } }
```

Running the Seeder:

Once you've defined the seeder logic, you can use the `db:seed` Artisan command to run the seeder and populate the database. You can specify the seeder class to run or let Laravel run all seeders.
or to run all seeders:

```
php artisan db:seed --class=CitySeeder
```

The DatabaseSeeder Class:

Laravel is by default provided with the `DatabaseSeeder` class. This allows you to organize your seeders into one script, that is run by simply use the `db:seed` command without specifying a specific seeder class.

```
php artisan db:seed
```

The `DatabaseSeeder` class, located in the `database/seeders` directory, serves as the main entry point for running seeders. You can put any seeding logic in its `run()` method. The `call()` method helps to just list the specific seeder classes you want to call, for example:

```
use Illuminate\\Database\\Seeder; class DatabaseSeeder extends Seeder { pu
blic function run() { $this->call([ UserSeeder::class, CountrySeeder::clas
s, CitySeeder::class, // Add other seeders here ]); } }
```

Digging deeper

Faking data: factories

Sometimes, especially when testing applications, you need realistic data but without the hassle of making this up and entering it by yourself. You just want 50 blog posts for example with lorem ipsum like content. Most frameworks provide mechanisms to create these fake objects. Laravel has implemented it by means of factories.

Creating factories

To create a factory for the `Country` model, you can use the following command in your terminal:

```
php artisan make:factory CountryFactory --model=Country
```

This command generates a factory class named `CountryFactory` in the `database/factories` directory.

Defining factories

The generated `CountryFactory` is a class. The `definition()` method is used to define random values for each attribute your model requires. To generate this random data, the `$faker` object can be used. It is provided with lots of methods and attributes that return all different sorts of random data. From numbers up to paragraphs of text.

```
return [ 'name' => $this->faker->name, 'content' => $this->faker->paragraphs(3), 'size' => $this->faker->numberBetween(234241, 342234324) ];
```

Laravel uses the Faker library for this. For a full list, check the 'Formatters' section on the Faker documentation website:

FakerPHP / Faker - Available Formatters

Documentation for FakerPHP / Faker

https://fakerphp.github.io/formatters/

Using factories

Now, whenever you need to create dummy `Country` instances for testing or seeding, you can use this factory.

For example, in a database seeder:

```
<?php namespace Database\Seeders; use App\Models\Country; use Illuminate\Database\Seeder; class CountrySeeder extends Seeder { /** * Run the database seeds. */ public function run(): void { // Create 62 country objects in the database Country::factory()->count(62)->create(); } }
```

Further reading

You can read all about factories here:

Laravel - The PHP Framework For Web Artisans

Laravel is a PHP web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you

https://laravel.com/docs/eloquent-factories