Creating an Entity Class

We'll cover the following

Defining an entity class

Seeing the H2 library in the classpath, Spring has already configured the database. We need to define an entity class that represents data that will be stored in a table in the database.

Kotlin vs. Java

Again, if we were using Java, we'd write the Task entity class like so:

```
//Java code only for comparison purpose
package com.agiledeveloper.todo;
import javax.persistence.*;

@Entity
public class Task {
    @Id @GeneratedValue private Long id;
    private String description;

    public Long getId() { return id; }

    public String getDescription() { return description; }
}
```

Instead, we'll write that code using Kotlin. Create a new file `todo/src/main/kotlin/com/agiledeveloper/todo/Task.kt` and add the following content in it:

```
package com.agiledeveloper.todo

import javax.persistence.*

@Entity
data class Task(@Id @GeneratedValue val id: Long, val description: String)
```

Task.kt

The class Task has been annotated with `@Entity`. Also, the first property id has been annotated with `@Id` and `@GeneratedValue`. These annotations indicate that the property is a primary key and the unique values are generated by the database. The `Task` entity, in addition to an `id`, also has a `description` of type `String`. Since Kotlin generates the getters automatically, we don't have to write anything more. And, since we created `Task` as a data class, we get all the goodies like `equals()`, `hashCode()`, and `toString()` methods for free.

We have the entity class, but we need a way to store instances of that into the database. Spring makes that easy, and we only have to write the minimum code necessary for that step using Kotlin, as we'll see in the next lesson.