Creating a Repository Interface

We'll cover the following

Defining an interface

Spring's `CrudRepository` interface provides all the methods like `findById()` and `save()` to fetch from the database or to update the database. All we need to do is define a specialized interface to extend the `CrudRepository`. Spring Data will generate implementations for interfaces that extend `CrudRepository`.

Kotlin vs. Java

If we were writing in Java, we'd write something like:

```java
//Java code only for comparison purpose
package com.agiledeveloper.todo;

import org.springframework.data.repository.CrudRepository;

interface TaskRepository extends CrudRepository<Task, Long> {
}
```

The Kotlin code isn't too different from that: there's no need for `;` or `{}`, and we replaced `extends` with `:`. Let's write the Kotlin equivalent of the interface by creating a file, `todo/src/main/kotlin/com/agiledeveloper/todo/TaskRepository.kt`, with the following content:

```kotlin
package com.agiledeveloper.todo

import org.springframework.data.repository.CrudRepository

interface TaskRepository : CrudRepository<Task, Long>
```

TaskRepository.kt

The interface `TaskRepository` extends from `CrudRepository<Task, Long>`, where the first parametric type `Task` specifies the type of the entity, and the second parametric type `Long` specifies the type of the primary key.

We’re all set to perform CRUD operations, but we need a service to make the calls from. Let’s write that in the next lesson.