Berner
Fachhochschule

Spring Testing

Stephan Fischli

Introduction

- Dependency injection makes the code easier for testing since mock objects can be used instead of real dependencies
- Integration testing requires an application context but can be performed without deploying the application

Testing Dependencies

- Spring Boot provides utilities and annotations to help testing an application
- The `spring-boot-starter-test` starter contains the following libraries:
  - **JUnit**: de-facto standard for unit testing
  - **AssertJ**: fluent assertion library
  - **Hamcrest**: library of matcher objects
  - **Mockito**: Java mocking framework
  - **JSONassert**: assertion library for JSON
  - **JsonPath**: XPath for JSON

Testing Spring Boot Appplications

- The `@SpringBootTest` annotation creates the application context used in integration tests
- Other annotations such as `@DataJpaTest` are provided for testing specific slices of an application
- The `@Transactional` annotation can be used to execute a test method within a transaction which will be rolled back after the method

```java
@SpringBootTest
public class MovieTests {

    @Test
    @Transactional
    public void testRecommendation() {
        ...
    }
}
```

Test Configuration

- The test annotations search the main configuration starting from the test class up the package hierarchy until the application class is found
- To provide special test properties, the configuration file `application.properties` can be replaced in the test resources
- It is also possible to specify additional property files using the `@TestPropertySource` annotation

```
@SpringBootTest
@TestPropertySource("classpath:test.properties")
public class MovieTests {

    @Test
    public void testRecommendation() {
        ...
    }
}
```

Mocking Beans

- When running integration tests, it is sometimes necessary to mock dependencies
- Mocking can also be useful to increase testing performance or to simulate failures that might be hard to trigger in a real environment
- The `@MockitoBean` annotation can be used to define a Mockito mock for a bean inside the application context

```java
@SpringBootTest
public class MovieTests {

    @Autowired
    private MovieRecommender recommender;
    @MockitoBean
    private MovieCatalog catalog;

    @Test
    public void testRecommendation() {
        Mockito.when(catalog.find()).thenReturn(...);
        Movie movie = recommender.recommend("action");
        assertThat(movie).isEqualTo(...);
        }
}
```