Testing Spring Applications Lab

Purpose

In this lab you will refactor the RewardNetworkTests using Spring's system test support library to simplify and improve the performance of your testing sy You will then use Spring profiles to define multiple tests using different implementations of the AccountRepository, RestaurantRepository and RewardRepository for different environments.

Learning Outcomes

What you will learn:

- 1. The recommended way of system testing an application configured by Spring
- 2. How to write multiple test scenarios

Specific subjects you will gain experience with:

- 1. JUnit 5
- 2. Spring's TestContext framework
- 3. Spring Profiles

You will be using the 24-test project.

Estimated time to complete: 30 minutes.

Prerequisites

The prerequisites are included as part of the Introduction to the Spring Professional Learning Path course Lab Setup lesson.

If you already completed it, you should be ready to do this lab. If not, assuming you already have JDK 11 or 17 installed and Java IDE, you will need to do following:

- 1. Download the lab codebase zip file.
- 2. Once you have downloaded the file, unzip it under a directory of your choice. The unzipped directory core-spring-labfiles/lab contains the lab projects.
- 3. Run ./mvnw clean install (if you plan on using Maven as your build tool).

Use case

In the previous lab, you have written test code in which test target object (RewardNetwork object) gets recreated each time test method gets invoked. This results in inefficient and slow testing performance. In this lab, you are going to use Spring's TestContext framework to address that.

There is also a need to provide different testing configuration data depending on environments. You will leverage Spring's profiles for that purpose.

Quick Instructions

If you are already knowledgeable with the lesson concepts, you may consider jumping right to the code, and execute the lab in form of embedded TODO comments. Instructions on how to view them are at the <u>Using TODO Tasks</u> article.

If you aren't sure, try the TODO instructions first and refer to the lab instructions by TODO number if you need more help.

Instructions

Refactor to use Spring's TestContext framework

In rewards. RewardNetworkTests we setup our test-environment using Spring in the @BeforeEach setUp method. Instead we are going to use Spring's extension.

TODO-01: Use Spring TestContext framework

- Remove setUp() and tearDown() methods in the rewards/RewardNetworkTests class
- Annotate the class with @SpringJUnitConfig(classes=TestInfrastructureConfig.class), which is a composite annotation of @ExtendWith(SpringExtension.class) and @ContextConfiguration(classes=TestInfrastructureConfig.class)

Spring's TestContext framework offers full support for JUnit 5 via the SpringExtension class.

- Remove the attribute 'context' which is not needed anymore
- Run the test. You will get a red bar because the rewardNetwork field is null.
- Use @Autowired to populate the rewardNetwork bean.

When you run your test, the test runner's setup logic will use *auto-wiring* on your test class to set values from the ApplicationContext. This means the rewardNetwork field will be assigned to the RewardNetwork bean from the context automatically.

• Re-run your test and verify the test succeeds.

When you have successful test, you've successfully reconfigured the rewards integration test, and at the same time simplified your system test by leveraging Spring's test support.

In addition, the performance of your system test has potentially improved as the ApplicationContext is now created once per test case run (and c instead of once per test method. This test has only one method so it doesn't make any difference here, however.

Configure Repository Implementations using Profiles

We are now going to modify the test to use different repository implementations - either Stubs or using JDBC.

TODO-02: Annotate all 'Stub*Repository' classes

• First we are going to use the stub repositories in /src/test/java/rewards/internal.

We need to make them Spring beans by annotating them as repository components. Annotate the stub classes with @Repository.

 If you run RewardNetworkTests again, it should fail because you have multiple beans of the same type - the original JDBC implementations and no stubs.

To fix this we will introduce two profiles:

- The stub repositories will belong to the "stub" profile
- The JDBC repositories to the "jdbc" profile.

TODO-03: Assign the 'jdbc' profile to all Jdbc*Repository classes

- Follow all the TODO 03 steps and use the @Profile annotation to put all the repositories in this project into their correct profile there are 6 reposito classes to annotate in total.
- Annotate the RewardNetworkTests class with @ActiveProfiles to make "stub" the active profile.
- Rerun the test it should work now.
- Check the console to see that the stub repository implementations are being used.

Notice that the embedded database is also being created even though we don't use it. We will fix this soon.

TODO-04: Change active-profile to "jdbc"

• Switch the active-profile to "jdbc".

- Rerun the test it should still work.
- Check the console again to see that the JDBC repository implementations are being used.

Switching between Development and Production Profiles

Profiles allow different configurations for different environments such as development, testing, QA (Quality Assurance), UAT (User Acceptance Testing), production and so forth.

In this step, we will introduce two new profiles: "local" and "jndi".

In both cases, we will be using the JDBC implementations of our repositories so two profiles will need to be active at once.

The difference between local and jndi is different infrastructure configuration.

In this case, we are going to swap between an in-memory test database (local profile) and the "real" database defined as a JNDI resource (jndi profile).

TODO-05: Assign beans to the "local" profile

- Modify TestInfrastructureLocalConfig.java so that all the beans defined in this configuration class are members of the profile called "local".
- Does RewardNetworkTests still run OK? Why not?

TODO-06: Use "jdbc" and "local" as active profiles

• Fix the test by adding the "local" profile to the @ActiveProfiles annotation in RewardNetworkTests.

Remember you will need to retain the "jdbc" profile as well.

• Rerun the test - it should work again.

TODO-07: Use "jdbc" and "jndi" as active profiles

- We have already setup the "real" dataSource for you using a JNDI lookup. We use a standalone JNDI implementation in this lab normally JNDI wo provided by your JEE container (such as Tomcat, JBoss or WebLogic).
- Change the active profile of RewardNetworkTests from "local" to "jndi".
- Rerun the test, it should still work.
- To see what has changed, look at the console and you will see logging from an extra bean called SimpleJndiHelper.
- Switch the profile back from "jndi" to "local" and rerun. Check the console and note that the SimpleJndiHelper is no longer used.

Optional Step - Further Refactoring

TODO-08: Use inner class configuration

- When no class is specified, Spring's test framework will look for an inner static class marked with @Configuration. Since the TestInfrastructure(class is so small anyway, copy the entire class definition, including annotations, to an inner static class within the test class.
- Remove the configuration class reference from the @SpringJUnitConfig annotation (no property in the brackets).

This is an example of convention over configuration.

• Run the test.

Does the test still pass?

• When you copy the TestInfrastructureConfig class into RewardNetworkTests, remember to make it static - refer to example in slides if unsu

Summary

In this lab, you have refactored testing code of the Rewards Application.

to use Spring's TestContext framework. Then you have used Spring's profiles to provide different testing configurations.

Summary Instructors

In this lab you gained experience using JUnit 5 with Spring and how to configure the completed reference domain for unit and integration tests.

☐ Give Feedback

Help us improve by sharing your thoughts.

Give Feedback



Guides | Courses | Learning Path | Community | Instructors | About | Give Feedback

Contact | FAQs | Terms | Privacy | Your California Privacy Rights

Unlock your full potential with Spring courses designed by experts.

Copyright © 2005-2024 Broadcom. All Rights Reserved. The term "Broadcom" refers to Broadcom Inc. and/or its subsidiaries.