
SPRING FRAMEWORK
GURU

Testing in the Spring Framework

Testing Terminology

- **Code Under Test** - This is the code (or application) you are testing
- **Test Fixture** - “A test fixture is a fixed state of a set of objects used as a baseline for running tests. The purpose of a test fixture is to ensure that there is a well known and fixed environment in which tests are run so that results are repeatable.” - JUnit Doc
  - Includes: input data, mock objects, loading database with known data, etc

Testing Terminology

- **Unit Tests / Unit Testing** - Code written to test code under test
  - Designed to test specific sections of code
  - Percentage of lines of code tested is code coverage
    - Ideal coverage is in the 70-80% range
  - Should be ‘unity’ and execute very fast
  - Should have no external dependencies
    - ie no database, no Spring context, etc

Testing Terminology

- **Integration Tests** - Designed to test behaviors between objects and parts of the overall system
  - Much larger scope
  - Can include the Spring Context, database, and message brokers
  - Will run much slower than unit tests

Testing Terminology

- **Functional Tests** - Typically means you are testing the running application
  - Application is live, likely deployed in a known environment
  - Functional touch points are tested
    - i.e. Using a web driver, calling web services, sending / receiving messages, etc

Testing Terminology

- **TDD** - Test Driven Development - Write tests first, which will fail, then code to ‘fix’ test.
- **BDD** - Behavior Driven Development - Builds on TDD and specifies that tests of any unit of software should be specified in terms of desired behavior of the unit.
  - Often implemented with DSLs to create natural language tests
  - JBehave, Cucumber, Spock
  - example: given, when, then

Testing Terminology

- **Mock** - A fake implementation of a class used for testing. Like a test double.
- **Spy** - A partial mock, allowing you to override select methods of a real class.

Testing Goals

- Generally, you will want the majority of your tests to be unit tests
- Bringing up the Spring Context makes your tests exponentially slower
- Try to test specific business logic in unit tests
- Use Integration Tests to test interactions
- Think of a pyramid. Base is unit tests, middle is integration tests, top is functional tests

Test Scope Dependencies

- Using spring-boot-starter-test (default from Spring Initializr will load the following dependencies:
    - JUnit - The de-facto standard for unit testing Java applications
    - Spring Test and Spring Boot Test - Utilities and integration test support for Spring Boot applications
    - AssertJ - A fluent assertion library
    - Hamcrest - A library of matcher objects
    - Mockito - A Java mocking framework
    - JSONassert - An assertion library for JSON
    - JSONPath - XPath for JSON

JUnit 4

- JUnit 4 is the most popular testing framework in the Spring community
- Originally written by Erich Gamma and Kent Beck (creator of extreme programming)
- JUnit 5 is currently in Alpha. Milestone 1 expected in July 2017. GA expected in 2017

JUnit 4 Annotations

Annotation	Description
@Test	Identifies a method as a test method.
@Before	Executed before each test. It is used to prepare the test environment (e.g., read input data, initialize the class).
@After	Executed after each test. It is used to cleanup the test environment. It can also save memory by cleaning up expensive memory structures.
@BeforeClass	Executed once, before the start of all tests. Methods marked with this annotation need to be defined as static to work with JUnit.
@AfterClass	Executed once, after all tests have been finished. Methods annotated with this annotation need to be defined as static to work with JUnit.
@Ignore	Marks that the test should be disabled.
@Test(expected = Exception.class)	Fails if the method does not throw the named exception.
@Test(timeout = 10)	Fails if the method takes longer than 100 millseconds.

Spring Boot Annotations

Annotation	Description
@RunWith(SpringRunner.class)	Run test with Spring Context
@SpringBootTest	Search for Spring Boot Application for configuration
@TestConfiguraiton	Specify a Spring configuration for your test
@MockBean	Injects Mockito Mock
@SpyBean	Injects Mockito Spy
@JsonTest	Creates a Jackson or Gson object mapper via Spring Boot
@WebMvcTest	Used to test web context without a full http server
@DataJpaTest	Used to test data layer with embedded database

Spring Boot Annotations

Annotation	Description
@JdbcTest	Like @DataJpaTest, but does not configure entity manager
@DataMongoTest	Configures an embedded MongoDB for testing
@RestClientTest	Creates a mock server for testing rest clients
@AutoConfigureRestDocks	Allows you to use Spring Rest Docs in tests, creating API documentation
@BootStrapWith	Used to configure how the TestContext is bootstrapped
@ContextConfiguration	Used to direct Spring how to configure the context for the test.
@ContextHierarchy	Allows you to create a context hierarchy with @ContextConfiguration
@ActiveProfiles	Set which Spring Profiles are active for the test

Spring Boot Annotations

Annotation	Description
@TestPropertySource	Configure the property sources for the test.
@DirtiesContext	Resets the Spring Context after the test (expensive to do)
@WebAppConfiguration	Indicates Spring should use a Web Application context
@TestExecutionListeners	Allows you to specify listeners for testing events
@Transactional	Run test in transaction, rollback when complete by default
@BeforeTranasaction	Action to run before starting a transaction.
@AfterTransaction	Action to run after a transaction.
@Commit	Specifies the transaction should be committed after the test.

Spring Boot Annotations

Annotation	Description
@Rollback	Transaction should be rolled back after test. (Default action)
@Sql	Specify SQL scripts to run before
@SqlConfig	Define meta data for SQL scripts
@SqlGroup	Group of @Sql annotations
@Repeat	Repeat test x number of times
@Timed	Similar to JUnit's timeout, but will wait for test to complete, unlike JUnit.
@IfProfileValue	Indicates test is enabled for a specific testing environment
@ProfileValueSourceConfiguration	Specify a profile value source


SPRING FRAMEWORK
GURU