Testing Code

Terminology

- Test Fixture - A fixed state of a set of objects used as a baseline for running tests. There should be a well known and fixed environment in which tests are run so that the results are repeatable.
- Unit Tests / Unit Testing - Code written to test code uder test
  - Designed to test specific sections of code
  - Ideally code coverage is maybe 70-80%
  - Should be small and fast
  - No external dependencies
    - No databases or spring components
- Integration Tests - Designed to test behaviors between objecrts and parts of the overall system
  - Much larger scope
  - Can include the Spring Context, databases, and message brokers
  - Much slower than unit tests
- Functional Tests - Testing a running application
  - Likely deployed in a test env
  - Functional touch points are tested
- Test Driven Development - Write Tests first, which of course fail, then code to 'fix' it
- BDD - 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
- 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.
  - Generally Mocking is more popular than Spying

Testing Goals

- Generally you want the majority of tests to be unit tests.
- Bringing up the Spring Context makes your test 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, top is functional tests

Test Scope Dependencies

- Using spring-boot-starter-test will load the following:
  - JUnit - the de-facto standard for unit testing
  - Spring Test and Spring Boot Test - Utilites and integration test support for Spring Boot applications
  - AssertJ - 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 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
@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
@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.
@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

JUnit 4	JUnit 5
@Test(expected = Foo.class)	Assertions.assertThrows(Foo.class…
@Test(timeout = 1)	Assertions.assertTimeout(Duration…
@RunWith(SpringJUnit4ClassRunner.class)	@ExtendWith(SpringExtension.class)

Revision #2
Created 17 April 2022 00:33:50 by Elkip
Updated 17 April 2022 01:36:11 by Elkip