Testing Spring

Beginner to Guru

Spring Framework Testing Features

Spring Framework

- This lecture will explore testing features of the Spring Framework
- Specifically Spring Framework, and not Spring Boot
- Spring Boot will be explored fully in an upcoming section of the course
- Spring Boot is very complementary to the testing features in the Spring Framework

Spring Framework Testing Features

- Features for Unit Testing
  - Mock Objects
    - **Environment** - Mock Environment and Properties Source
    - **JNDI** - Mock of JNDI lookup
    - **Servlet API** - For testing of web environment
    - **Spring Web Reactive** - Testing of reactive web environment

Testing Utilities

- ReflectionTestUtils - allows use of refection to modify private fields
  - Spring can autowire private properties - (Considered a poor practice)
  - Can also be used to hook into bean lifecycle events
- AOP Utils - Helps with testing of AOP

Spring MVC Test

- Spring MVC Test - Robust framework for testing controller interactions
    - **MockHttpServletRequest** - Mock implementation of request / response
    - **MockHttpSession** - Mock of Http Session
    - **ModleAndViewAssert** - assertion utilities
- Framework allows testing of web requests without the need of a running container
- Allows true unit tests for controller
    - Tests run much faster when the web context is not started

Spring Integration Testing

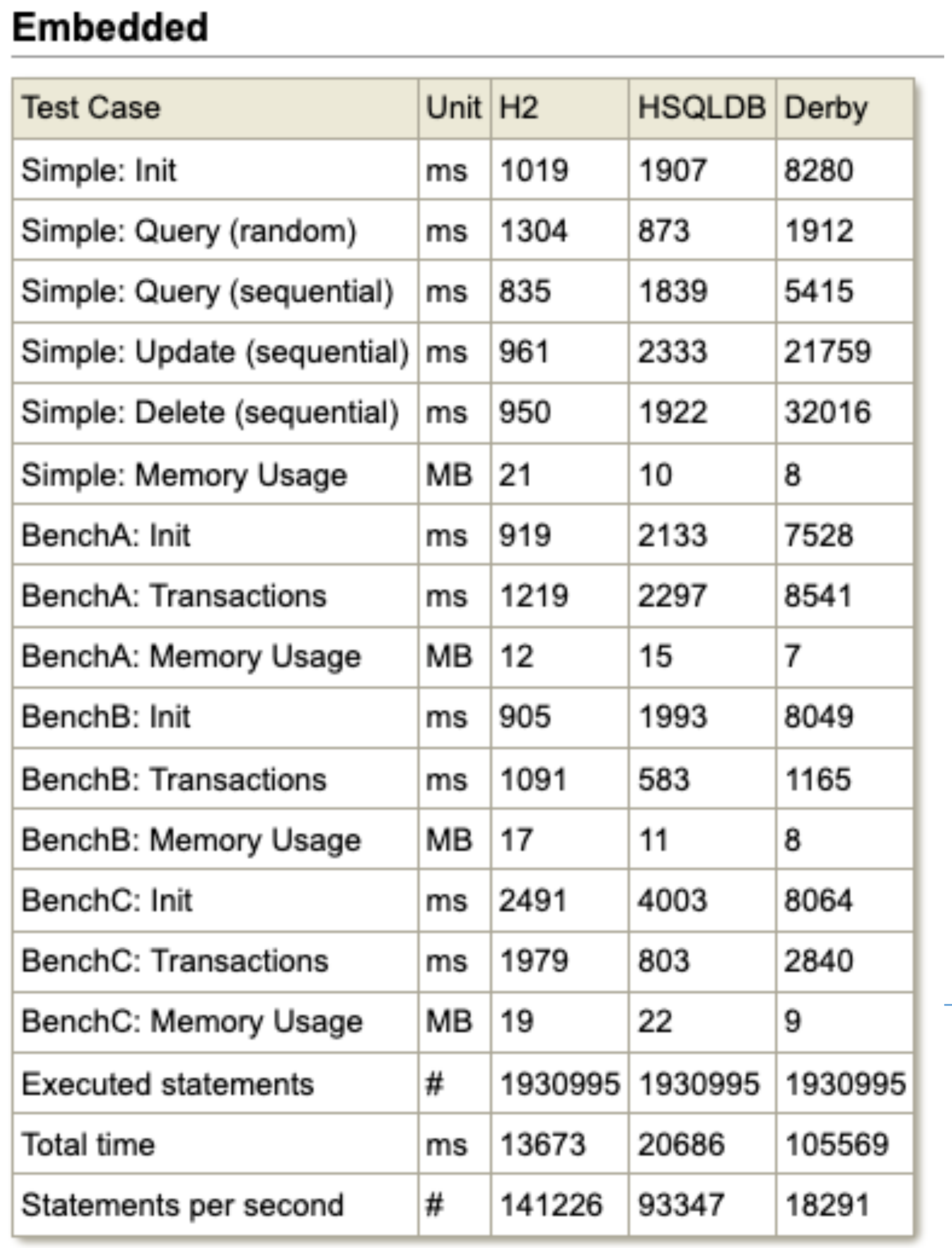
- Integration testing is when the Spring Context is started to support the test
- Loading the Spring Context is considered an expensive operation
  - Can take 10 - 40 seconds to load context
    - Depending on complexity of project and hardware
- Spring will cache the context between tests to improve performance
- Dependency Injection - Spring can be used to inject beans into test classes
- Transaction Management - By default Spring will automatically rollback database transactions

JDBC Testing Support

- **JdbcTemplate** - Spring will configure an instance of JdbcTemplate for testing support
- **JdbcTestUtils** - Collection of utilities to assist with database testing
  - countRowsInTable
  - countRowsInTableWhere
  - deleteFromTables
  - deleteFromTablesWhere
  - dropTables

Embedded Database Support

- Spring provides support for popular in-memory Java databases
- Very useful for testing database interactions
- Supported natively by Spring:
  - H2
  - HSQL
  - Derby
- H2 has a nice web based DB console



Embedded

Test Case	Unit	H2	HSQLDB	Derby
Simple: Init	ms	1019	1907	8280
Simple: Query (random)	ms	1304	873	1912
Simple: Query (sequential)	ms	835	1839	5415
Simple: Update (sequential)	ms	961	2333	21759
Simple: Delete (sequential)	ms	950	1922	32016
Simple: Memory Usage	MB	21	10	8
BenchA: Init	ms	919	2133	7528
BenchA: Transactions	ms	1219	2297	8541
BenchA: Memory Usage	MB	12	15	7
BenchB: Init	ms	905	1993	8049
BenchB: Transactions	ms	1091	583	1165
BenchB: Memory Usage	MB	17	11	8
BenchC: Init	ms	2491	4003	8064
BenchC: Transactions	ms	1979	803	2840
BenchC: Memory Usage	MB	19	22	9
Executed statements	#	1930995	1930995	1930995
Total time	ms	13673	20686	105569
Statements per second	#	141226	93347	18291

Spring Framework Testing Annotations

- **@BootstrapWith** - Class-level annotation to configure how the test context is bootstrapped
- **@ContextConfiguration** - Class-level annotation to configure the application context
- **@WebAppConfigurtation** - Class-level annotation to configure a web application context
- **@ContextHiearchy** - Class-level annotation to set multiple @ContextConfigurations
- **@ActiveProfiles** - Class-level annotation to set active profiles for test
- **@TestPropertySource** - Class-level annotation to set property sources for test
- **@DirtiesContext** - Class or method level annotation which tells Spring to re-load context after test - (slows down your tests)

Spring Framework Testing Annotations

- **@TestExecutionListeners** - Used to configure test execution listeners
- **@Commit** - Class or method level annotation to commit action of test to database.
- **@Rollback** - Class or method level annotation to rollback action of test from database.
- **@BeforeTransaction** - run a method which returns void before a transaction is started
- **@AfterTransaction** - run a method which returns void after a transaction has completed

Spring Framework Testing Annotations

- **@Sql** - Used to configure SQL scripts to run before a test
- **@SqlConfig** - Configuration for the parsing of SQL scripts
- **@SqlGroup** - Configure a grouping of SQL scripts

JUnit 4 Testing Annotations

- **@IfProvileValue** - Enable test for specific environments
- **@ProfileValueSourceConfiguration** - Class-level annotation to configure how profile values are retrieved
- **@Timed** - Require test to complete within a period of time
- **@Repeat** - Repeat test x number of times

Spring JUnit 5 Testing Annotations

- **@SpringJUnitConfig** - Combines @ContextConfiguration with @ExtendWith(SpringExtension.class) to configure the Spring Context for the test
- **@SpringJUnitWebConfig** - Combines @ContextConfiguration and @WebAppConfiguration with @ExtendWith(SpringExtension.class) to configure the Spring Context for the test
- **@EnabledIf** - Conditional execution of test
- **@DisabledIf** - Conditional execution of test

SPRING FRAMEWORK

GURU