♥ Unit Testing Data Access Logic in Java

Posted on May 18, 2015 by Arulkumaran Kumaraswamipillai

Most interviewers like Java candidates those who are passionate and experienced about writing unit tests. Any non trivial Java application will be making calls to database tables. So, here are a few questions and answers testing your ability to write unit tests to test data access layer.

Q1. How will you go about unit testing the data access logic?
A1. The main challenge with using a 'real' database" for unit testing is the setup, take down, and isolation of the tests. You don't want to have to spin up an entirely new Oracle database and create tables and data just for unit testing.

Mocking the database is one option when testing the service layer, however for testing the DAO itself you need something behind the DAO has the data and the queries to run properly.

Solution: The solution is to use an in memory database. HyperSQL (i.e. HSQLDB) is an excellent choice for this because it has the ability to emulate the dialect of another database.

Q2. Can you describe the basic steps involved in setting up an in memory database in Java apps?

A2.

Step 1: The pom.xml file that brings in the libraries required for testing — the **hsqldb** and **spring-test** jars.

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version> 3.2.13.RELEASE</version>
</dependency>
<dependency>
    <groupId> org.hsqldb</groupId>
    <artifactId>hsqldb</artifactId>
    <version>2.3.2</version>
</dependency>
```

Step 2: The Spring configuration file to bootstrap HSQLDB. One of the new features introduced in Spring 3 is the **support for embedded Java database engines**. Embedded databases like HSQL, H2, or Derby are very useful during the development phase of the project as they are fast, have small memory footprints and are opensource.
"**EmbeddedDatabaseBuilder**" will make things easier.

```
package com.jpa.tutorial.config;

import javax.sql.DataSource;

import org.springframework.context.annotation.Be
import org.springframework.context.annotation.Co
import org.springframework.context.annotation.Im
import org.springframework.jdbc.datasource.embed
import org.springframework.jdbc.datasource.embed


@Configuration
@Import(MyDomainConfiguration.class)
public class MyAppCommonsDomainTestConfiguration

    @Bean
    public DataSource dataSource() {
        final EmbeddedDatabaseBuilder builder =
        builder.setType(EmbeddedDatabaseType.HSQ
        builder.addScript("classpath:hsqldb-orac
        builder.addDefaultScripts(); // defaults
        return builder.build();
    }
}
```


16 Technical Key Areas

80+ step by step Java

Step 3: The default schema.sql and data.sql along with the hsqldb-oracle.sql need to be in the classpath. Say in src/test/resources

hsqldb-oracle.sql

```
1 SET DATABASE SQL SYNTAX ORA TRUE
2
```

schema.sql

```
1
2  ------------------------------
3  -- Drop the current schema
4  ------------------------------
5  DROP TABLE IF EXISTS account;
6
7  ------------------------------
8  -- Create new schema
9  ------------------------------
10 CREATE TABLE account(
11    account_id            NUMERIC(10,0)    NOT NUL
12    account_code          VARCHAR(32)    NOT NULL,
13 CONSTRAINT account_pk PRIMARY KEY (account_id)
14 );
15
16
```

data.sql dummy data for testing

```
1  ------------------------------
2  -- Truncate Tables
3  ------------------------------
4  TRUNCATE TABLE account;
5
6  ------------------------------
7  -- Populate Schema
8  ------------------------------
9  INSERT INTO account
10   (account_id, account_code)
11 VALUES
12   (1, '456');
13
14 COMMIT;
15
16
```

The above steps would have created the schema and data for the unit testing purpose in memory.

It is also flexible enough to specifically configure the scripts like

```
@Value("classpath:com/myapp/sql/db-schema.sql")
private Resource schemaScript;

@Value("classpath:com/myapp/sql/db-test-data.sql
private Resource dataScript;

@Bean
public DataSourceInitializer dataSourceInitializ
    final DataSourceInitializer initializer = ne
    initializer.setDataSource(dataSource);
    initializer.setDatabasePopulator(databasePop
    return initializer;
}

private DatabasePopulator databasePopulator() {
    final ResourceDatabasePopulator populator =
    populator.addScript(schemaScript);
    populator.addScript(dataScript);
    return populator;
}
```

if you are doing it via XML based spring config, it will be something like

```

    <jdbc:script location="classpath:schema.sql"/
    <jdbc:script location="classpath:data.sql"/>
</jdbc:embedded-database>

```

The above approach can help you identify and fix any object-to-relational mapping issues, spring-jpa-hibernate wiring issues, and other coding issues.

Q3. How the heck to unit test business logic with a database hanging around?
A3. One of the best ways to get the database out of your way is to hide data access behind abstracted interfaces that can be mocked in business logic testing.

There are a few libraries that help you mock database logic.

1) MockRunner: has some JDBC-specific extensions that allow for simulating JDBC ResultSets, as well as for checking whether actual queries are executed.

2) An ordinary Java mocking libraries like jMock, EasyMock, Mockito etc mock the DAO layer. This approach was explained in **Q2.** Mocking DAO Layer

Q4. What are some of the things you don't have to unit test?
A4. You should never write unit test for the sake of writing one. There are scenarios where you don't have to write unit tests. For example,

1) Third-party frameworks and libraries. You should assume that they work as expected.

2) Code that gives non deterministic results.

3) Very complex database logic invoking stored procedures, etc. It is much easier to test basic CRUD operations.

4) Trivial code like getters and setters.

Popular Member Posts

♦ 11 Spring boot interview questions & answers
906 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers
816 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers
427 views

18 Java scenarios based interview Questions and Answers
409 views

♦ 7 Java debugging interview questions & answers
324 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers
312 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints
304 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

♦ Object equals Vs == and pass by reference Vs value

234 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with

this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ jBehave and BDD example

DOS script to execute a spring batch job ›

Posted in Data Access Unit Testing

Tags: Free Content

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer

or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

↑