Unit Test Hibernate – DAO Layer with Spring & HSQLDB

Posted on October 4, 2015 by Arulkumaran Kumaraswamipillai

Step 1: The pom.xml file with test jars, Spring jars, and hibernate jars

```
1  <project xmlns="http://maven.apache.org/POM/4.0
2      xsi:schemaLocation="http://maven.apache.org
3      <modelVersion>4.0.0</modelVersion>
4
5      <groupId>com.mytutorial</groupId>
6      <artifactId>simpleSpringRestWeb</artifactId
7      <version>1.0-SNAPSHOT</version>
8      <packaging>war</packaging>
9
10     <name>simpleSpringRestWeb</name>
11     <url>http://maven.apache.org</url>
12
13     <properties>
14         <project.build.sourceEncoding>UTF-8</pr
15         <slf4j.version>1.7.10</slf4j.version>
16         <commons-logging.version>1.1.1</commons
17         <spring.version>4.0.9.RELEASE</spring.v
18     </properties>
19
20     <dependencies>
21         <!-- Unit Testing-->
22         <dependency>
23             <groupId>junit</groupId>
```

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

open all | close all

- Ice Breaker Interview
- Core Java Interview C
- JEE Interview Q&A (3
- Pressed for time? Jav
- SQL, XML, UML, JSC
- Hadoop & BigData Int
- Java Architecture Inte
- Scala Interview Q&As
- Spring, Hibernate, & M
  - Spring (18)
  - Hibernate (13)
  - AngularJS (2)
  - Git & SVN (6)
  - JMeter (2)
  - JSF (2)
  - Maven (3)
- Testing & Profiling/Sa
  - Automation Testing
  - Code Coverage (2)
  - Code Quality (2)
  - jvisualvm profiling (
  - Performance Testir

```
24              <artifactId>junit</artifactId>
25              <version>4.11</version>
26              <scope>test</scope>
27          </dependency>
28          <dependency>
29              <groupId>org.mockito</groupId>
30              <artifactId>mockito-all</artifactId
31              <version>1.9.0</version>
32              <scope>test</scope>
33          </dependency>
34          <dependency>
35              <groupId>org.hamcrest</groupId>
36              <artifactId>hamcrest-all</artifactI
37              <version>1.3</version>
38              <scope>test</scope>
39          </dependency>
40          <dependency>
41              <groupId>org.springframework</group
42              <artifactId>spring-test</artifactId
43              <version>${spring.version}</version
44              <scope>test</scope>
45          </dependency>
46          <dependency>
47              <groupId>com.jayway.jsonpath</group
48              <artifactId>json-path</artifactId>
49              <version>0.8.1</version>
50              <scope>test</scope>
51          </dependency>
52          <dependency>
53              <groupId>com.jayway.jsonpath</group
54              <artifactId>json-path-assert</artif
55              <version>0.8.1</version>
56              <scope>test</scope>
57          </dependency>
58
59          <!-- Hibernate -->
60          <dependency>
61              <groupId>org.hibernate.common</grou
62              <artifactId>hibernate-commons-annot
63              <version>4.0.1.Final</version>
64              <scope>compile</scope>
65          </dependency>
66          <dependency>
67              <groupId>org.hibernate.javax.persis
68              <artifactId>hibernate-jpa-2.0-api</
69              <version>1.0.1.Final</version>
70              <scope>compile</scope>
71          </dependency>
72          <dependency>
73              <groupId>org.hibernate</groupId>
74              <artifactId>hibernate-core</artifac
75              <version>4.2.0.SP1</version>
76              <scope>compile</scope>
77          </dependency>
78          <dependency>
79              <groupId>org.hibernate</groupId>
80              <artifactId>hibernate-entitymanager
81              <version>4.2.0.SP1</version>
82              <scope>compile</scope>
83          </dependency>
84          <dependency>
85              <groupId>org.hibernate</groupId>
86              <artifactId>hibernate-ehcache</arti
87              <version>4.2.0.SP1</version>
88              <scope>compile</scope>
89          </dependency>
```


16 Technical Key Areas

80+ step by step Java

```
90          <dependency>
91              <groupId>org.jboss.jbossts</groupId
92              <artifactId>jbossjta</artifactId>
93              <version>4.16.4.Final</version>
94          </dependency>
95
96          <!-- HSQLDB -->
97          <dependency>
98              <groupId>org.hsqldb</groupId>
99              <artifactId>hsqldb</artifactId>
100             <version>2.3.2</version>
101         </dependency>
102
103         <!-- Spring -->
104         <dependency>
105             <groupId>org.springframework.data</
106             <artifactId>spring-data-jpa</artifa
107             <version>1.5.1.RELEASE</version>
108             <exclusions>
109                 <exclusion>
110                     <groupId> org.springframewo
111                     <artifactId>spring-jdbc</ar
112                 </exclusion>
113                 <exclusion>
114                     <groupId> org.springframewo
115                     <artifactId>spring-core</ar
116                 </exclusion>
117             </exclusions>
118         </dependency>
119
120         <dependency>
121             <groupId>org.springframework</group
122             <artifactId>spring-core</artifactId
123             <version>${spring.version}</version
124         </dependency>
125         <dependency>
126             <groupId>org.springframework</group
127             <artifactId>spring-beans</artifactI
128             <version>${spring.version}</version
129         </dependency>
130
131         <dependency>
132             <groupId>org.springframework</group
133             <artifactId>spring-expression</arti
134             <version>${spring.version}</version
135         </dependency>
136         <dependency>
137             <groupId>org.springframework</group
138             <artifactId>spring-context</artifac
139             <version>${spring.version}</version
140         </dependency>
141
142         <dependency>
143             <groupId>org.springframework</group
144             <artifactId>spring-jdbc</artifactId
145             <version>${spring.version}</version
146         </dependency>
147
148         <!-- Other -->
149     </dependencies>
150 </project>
```

Step 2: The HSQL DB schema and data scripts in the src/test/resources folder.

sql/schema.sql

```
CREATE TABLE tbl_person
(     person_id int NOT NULL
    , person_gender varchar(6) NOT NULL
    , person_name varchar(100) NOT NULL
    , person_age int NOT NULL
);

CREATE UNIQUE INDEX idx_tbl_person ON tbl_person(
```

sql/data.sql

```
INSERT INTO tbl_person (person_id, person_gender,
        VALUES ( 1, 'MALE', 'John', 23);

INSERT INTO tbl_person (person_id, person_gender,
        VALUES ( 2, 'FEMALE', 'Mary', 23);

```

Step 3: The Spring configuration for the unit tests in the src/test/java folder.

```
package com.mytutorial;

import javax.sql.DataSource;

import org.springframework.context.annotation.Be
import org.springframework.context.annotation.Co
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({ SimpleConfig.class })
@ComponentScan("com.mytutorial")
public class SimpleTestConfig {

    @Bean
    public DataSource dataSource() {
        final EmbeddedDatabaseBuilder builder =
        builder.setType(EmbeddedDatabaseType.HSQ
        builder.addScript("sql/schema.sql");
        builder.addScript("sql/data.sql");
        return builder.build();
    }
}


```

Step 4: The Spring configuration for the entity manager and datasource.

```
1  package com.mytutorial;
2
3  import javax.sql.DataSource;
4
5  import org.hibernate.ejb.HibernatePersistence;
6  import org.springframework.beans.factory.annotat
7  import org.springframework.context.annotation.Be
8  import org.springframework.context.annotation.Co
9  import org.springframework.context.annotation.Co
10 import org.springframework.core.env.Environment;
11 import org.springframework.jdbc.datasource.looku
12 import org.springframework.orm.jpa.LocalContaine
13
14
15 @Configuration
16 @ComponentScan("com.mytutorial")
17 public class SimpleConfig {
18
19     @Autowired
20     Environment environment;
21
22     @Bean
23     public DataSource dataSource() {
24         final JndiDataSourceLookup jndiLookup =
25         final String dataSourceJndi = environmen
26         return jndiLookup.getDataSource(dataSour
27     }
28
29     @Bean
30     public LocalContainerEntityManagerFactoryBea
31         final LocalContainerEntityManagerFactory
32         entityManager.setDataSource(dataSource()
33         entityManager.setPackagesToScan(new Stri
34         entityManager.setPersistenceProviderClas
35         return entityManager;
36     }
37
38 }
```

Step 5: The Hibernate entity class

```
1  package com.mytutorial;
2
3  import javax.persistence.Column;
4  import javax.persistence.Entity;
5  import javax.persistence.Id;
6  import javax.persistence.Table;
7
8
9  @Entity
10 @Table(name = "tbl_person")
11 public class Person {
12
13     @Column(name = "person_id")
14     @Id
15     private Integer id;
```

```
16
17      @Column(name = "person_gender")
18      private String gender;
19
20      @Column(name = "person_name")
21      private String name;
22
23      @Column(name = "person_age")
24      private Integer age;
25
26      public Integer getId() {
27          return id;
28      }
29
30      public void setId(Integer id) {
31          this.id = id;
32      }
33
34      public String getGender() {
35          return gender;
36      }
37
38      public void setGender(String gender) {
39          this.gender = gender;
40      }
41
42      public String getName() {
43          return name;
44      }
45
46      public void setName(String name) {
47          this.name = name;
48      }
49
50      public Integer getAge() {
51          return age;
52      }
53
54      public void setAge(Integer age) {
55          this.age = age;
56      }
57
58      @Override
59      public String toString() {
60          return "Person [id=" + id + ", gender="
61      }
62  }
```

Step 6: The Dao interface and implementation class

```
1  package com.mytutorial;
2
3  public interface PersonDao {
4
5      Person findById(Integer id);
6
7  }
```

```
1  package com.mytutorial;
2
```

```
3  import javax.persistence.EntityManager;
4  import javax.persistence.PersistenceContext;
5
6  import org.springframework.stereotype.Repository
7
8  @Repository
9  public class PersonDaoImpl implements PersonDao
10
11     @PersistenceContext
12     EntityManager entityManager;
13
14     @Override
15     public Person findById(Integer id) {
16         Person person = entityManager.find(Perso
17         return person;
18     }
19
20 }
```

Step 7: Th unit test class to test the DAO layer with the in memory HSQLDB database.

```
1  package com.mytutorial;
2
3  import org.junit.Assert;
4  import org.junit.Test;
5  import org.junit.runner.RunWith;
6  import org.springframework.beans.factory.annotat
7  import org.springframework.test.context.ContextC
8  import org.springframework.test.context.junit4.S
9
10
11 @ContextConfiguration(
12     classes = { SimpleTestConfig.class })
13 @RunWith(SpringJUnit4ClassRunner.class)
14 public class PersonDaoTest {
15
16     @Autowired
17     PersonDao personDao;
18
19     @Test
20     public void testFindById(){
21         Person person = personDao.findById(1);
22         Assert.assertNotNull(person);
23         Assert.assertEquals("John", person.getNa
24         Assert.assertEquals("MALE", person.getGe
25     }
26 }
```

The next post covers JPA, and entitled "Unit Test Hibernate – DAO Layer with JPA, Spring & HSQLDB".

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

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**

‹ Spring Controller Unit Testing

Unit Test Hibernate – DAO Layer with JPA, Spring & HSQLDB ›

Posted in Data Access Unit Testing

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

 ↑