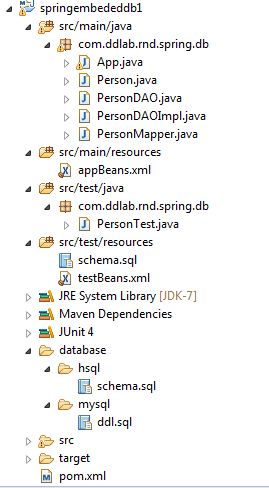
Spring JDBC Embedded database for Unit Testing

## Project Structure



## Maven pom.xml

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>springembededdb1</groupId>

<artifactId>springembededdb1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>springembededdb1</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>3.2.8.RELEASE</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.14</version>

</dependency>

**<dependency>**

**<groupId>commons-dbcp</groupId>**

**<artifactId>commons-dbcp</artifactId>**

**<version>1.2.2</version>**

**</dependency>**

<!-- For Unit Testing, Use HSQLDB -->

**<dependency>**

**<groupId>hsqldb</groupId>**

**<artifactId>hsqldb</artifactId>**

**<version>1.8.0.10</version>**

**</dependency>**

<!-- For actual application, we use MYSQL -->

**<dependency>**

**<groupId>mysql</groupId>**

**<artifactId>mysql-connector-java</artifactId>**

**<version>5.1.30</version>**

**</dependency>**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

## SQL File

## Hsql (schema.sql)

**drop** **table** Person if **exists**;

**create** **table** Person (id **int** **primary** **key**, name **varchar**(50) **not** **null**);

### mysql (ddl.sql)

**drop** **table** Person ;

**create** **table** Person (id **int** **primary** **key**, name **varchar**(50) **not** **null**);

**select** \* **from** Person;

## Configuration for unit testing (src/test/resources)

### Schema.sql

**drop** **table** Person if **exists**;

**create** **table** Person (id **int** **primary** **key**, name **varchar**(50) **not** **null**);

### testBeans.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop.xsd*

*http://www.springframework.org/schema/jdbc*

*http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd"*>

**<jdbc:embedded-database id=*"dataSource"*>**

**<jdbc:script location=*"classpath:schema.sql"* />**

**</jdbc:embedded-database>**

<bean id=*"personDao"* class=*"com.ddlab.rnd.spring.db.PersonDAOImpl"*>

<property name=*"dataSource"* ref=*"dataSource"* />

</bean>

</beans>

## Java code for Junit (src/test/java)

#### PersonTest

package com.ddlab.rnd.spring.db;

import static org.junit.Assert.assertEquals;

import java.util.List;

import org.junit.BeforeClass;

import org.junit.FixMethodOrder;

import org.junit.Test;

import org.junit.runners.MethodSorters;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

/\*\*

\* The Class PersonTest.

\*

\* @author Debadatta Mishra (PIKU)

\* Running test cases in order of method names in ascending order

\* Use @FixMethodOrder(MethodSorters.NAME\_ASCENDING).

\* But it s always discouraged because all test cases should be independent.

\*/

@FixMethodOrder(MethodSorters.NAME\_ASCENDING)

public class PersonTest {

private static ApplicationContext context = null;

private static PersonDAO personDao = null;

private static int id = 0;

@BeforeClass

public static void init() {

new PersonTest().doSetup();

}

public void doSetup() {

context = new ClassPathXmlApplicationContext("testBeans.xml");

personDao = (PersonDAO) context.getBean("personDao");

}

@Test

public void testCreatePerson() {

createPerson("Deb");

Person p1 = personDao.getPersonById(1);

assertEquals(1,p1.getId());

}

@Test

public void testGetAllPersons() {

List<Person> personList = personDao.getAllPersons();

if(personList.size() == 0 ) {

createPerson("Deb");

personList = personDao.getAllPersons();

}

assertEquals(1, personList.size());

}

@Test

public void testGetPersonById() {

createPerson("Deb3");

Person p1 = personDao.getPersonById(1);

assertEquals(1,p1.getId());

}

private void createPerson(String personName) {

Person p1 = new Person();

p1.setId((++id));

p1.setName(personName);

personDao.createPerson(p1);

}

}

## Main java code (src/main/java)

#### App.java

package com.ddlab.rnd.spring.db;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) throws Exception {

ApplicationContext context = new ClassPathXmlApplicationContext("appBeans.xml");

PersonDAO personDao = (PersonDAO) context.getBean("personDao");

for( int i = 1 ; i < 6 ; i++ ) {

Person person = new Person();

person.setId(i);

person.setName("Deb"+i);

personDao.createPerson(person);

}

//Get all persons from DB

List<Person> personList = personDao.getAllPersons();

for( Person person : personList ) {

System.out.println("Person Id:::"+person.getId());

System.out.println("Person Name :::"+person.getName());

}

//Get person by Id

Person personById = personDao.getPersonById(3);

System.out.println("Person Id:::"+personById.getId());

System.out.println("Person Name :::"+personById.getName());

}

}

#### Person.java

**package** com.ddlab.rnd.spring.db;

**public** **class** Person {

**private** **int** id;

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

#### PersonDAO.java

**package** com.ddlab.rnd.spring.db;

**import** java.util.List;

**public** **interface** PersonDAO {

**public** List<Person> getAllPersons();

**public** Person getPersonById(**int** id);

**public** **void** createPerson(Person person);

}

#### PersonDAOImpl.java

package com.ddlab.rnd.spring.db;

import java.util.List;

import javax.sql.DataSource;

import org.springframework.jdbc.core.JdbcTemplate;

public class PersonDAOImpl implements PersonDAO {

private DataSource dataSource;

private JdbcTemplate jdbcTemplate;

public List<Person> getAllPersons() {

String sqlQuery = "select \* from Person";

List<Person> persons = jdbcTemplate.query(sqlQuery, new PersonMapper());

return persons;

}

public Person getPersonById(int id) {

String sqlQuery = "select \* from Person where id = ?";

Person person = jdbcTemplate.queryForObject(sqlQuery,

new Object[] { id }, new PersonMapper());

return person;

}

public void createPerson(Person person) {

String sqlQuery = "insert into Person values (?,?)";

jdbcTemplate.update(sqlQuery,

new Object[] { person.getId(), person.getName() });

}

public DataSource getDataSource() {

return dataSource;

}

public void setDataSource(DataSource dataSource) {

this.dataSource = dataSource;

this.jdbcTemplate = new JdbcTemplate(this.dataSource);

}

}

#### PersonMapper.java

package com.ddlab.rnd.spring.db;

import java.sql.ResultSet;

import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class PersonMapper implements RowMapper<Person> {

public Person mapRow(ResultSet rs, int rowNum) throws SQLException {

Person person = new Person();

person.setId(rs.getInt("id"));

person.setName(rs.getString("name"));

return person;

}

}

### Spring configuration (src/main/resources)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop.xsd*

*http://www.springframework.org/schema/jdbc*

*http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd"*>

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"url"* value=*"jdbc:mysql://localhost:3306/test"* />

<property name=*"username"* value=*"deba"* />

<property name=*"password"* value=*"deba"* />

</bean>

<bean id=*"personDao"* class=*"com.ddlab.rnd.spring.db.PersonDAOImpl"*>

<property name=*"dataSource"* ref=*"dataSource"* />

</bean>

</beans>