Database connectivity using hibernate and spring mvc

1.Configuring h2

a.the executable jar file inside the bin folder of the setup has to be run,it'll open the brower .

b.In it we have to select h2(server) in place of h2(embedded).

c.Then in place of test as database name , i've entered Sports.

d.Userid was sa, and password was specified as sports.

e.Then connect,it opens the database in H2 Console on browser.

2.Adding spring dependency for using hibernate with H2

a)In pom , add:

<!-- Spring orm dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>4.3.6.RELEASE</version>

</dependency>

b)H2 dependency:

<!-- H2 dependency -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>1.4.190</version>

</dependency>

c)Hibernate core:

<!-- Hibernate-Core dependency -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.5.Final</version>

</dependency>

d)Commons-dbcp2:

<!-- Commons-dbcp2 dependency -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-dbcp2</artifactId>

<version>2.1.1</version>

</dependency>

e)Javac.persistence:

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.0-api</artifactId>

<version>1.0.1.Final</version>

</dependency>

3.Adding JPA to DTO(model) class

a)In Product.js ,modify the following code:

@Entity

**public** **class** Product {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** productId;

**private** String productName;

**private** String productCategory;

**private** String productDescription;

**private** String productKeyword;

**private** String productImage;

**private** **int** productPrice;

**private** **int** productQuantity;

4.Creation of applicationContext.xml file inside WEB-INF folder,and add following:

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

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

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

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

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

xsi:schemaLocation=*"*

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

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

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

*http://www.springframework.org/schema/context/spring-context-3.0.xsd*

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

*http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*

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

*http://www.springframework.org/schema/tx/spring-tx-4.0.xsd"*> </beans>

5.Loading applicationContext in web.xml file.

a)Add following in the web.xml:

<!-- if somebody does not want to follow the convention of naming applicationContext then add : -->

<context-param>

<param-name>ContextConfigLocation</param-name>

<param-value>

/WEB-INF/applicationContext.xml

/WEB-INF/dispatcher-Servlet.xml

</param-value>

</context-param>

<!-- Loading applicationContext.xml -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

6.Entire code apart from dispatcher-servlet.xml apart from outer bean and viewresolver should be cut and pasted to the applicationContext file.

7.Add beans in the applicationContext file

a)Add dataSource bean :

*<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSourcedestroy-method="close">*

*<*property name=*"driverName"* value=*"org.h2.Driver"*/>

<property name=*"url"* value=*"jdbc:h2:tcp://localhost/~/Sports"*/>

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

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

</bean>

b)Add sessionFactory bean:

<bean id=*"sessionFactory"* class=*"org.springframework.orm.hibernate5.LocalSessionFactoryBean"*>

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

<property name=*"packagesToScan"*>

<list>

<value>com.sports.yensports.model</value>

</list>

</property>

<property name=*"hibernateProperties"*>

<props>

<prop key=*"hibernate.dialect"*>org.hibernate.dialect.H2Dialect</prop>

<prop key=*"hibernate.hbm2ddl.auto"*>update</prop>

<prop key=*"hibernate.show\_sql"*>true</prop>

<prop key=*"hibernate.format\_sql"*>true</prop>

</props>

</property>

</bean>

c)Add annotation:

*<tx:annotation-driven transaction-manager="txManager" />*

d)Add the transaction manager bean:

<bean id=*"txManager"* class=*"org.springframework.orm.hibernate5.HibernateTransactionManager"*>

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

</bean>

8.Configure the ProductDAOImpl class

a)Goto the ProductDAOImpl class , and add @Transactional before class.

b)Delete the list and static block from the class.

c)Inside the class , add the following:

@Autowired

**private** SessionFactory sessionFactory;

**public** List<Product> getProducts()

{

**return** sessionFactory.getCurrentSession().createQuery("FROM Product",Product.**class**).getResultList();

}

**public** Product getProduct(**int** id)

{

List<Product> products = **new** ArrayList<Product>();

products = getProducts();

**return** products.get(id);

}

}

d)

9.