**Database connection and environment setup**

**Project: Full working Spring MVC and Hibernate application that connects to a database**

**Customer Relationship Management – CRM (Step-by-Step Process)**:

1. Set up Database Development-Environment
2. List Customers
3. Add a new Customer
4. Update a Customer
5. Delete a Customer

**Create a new database table and load data**:

* Create a new database table: customer
* Load table with sample data

**File: 01-create-user.sql**:

CREATE USER 'springstudent'@'localhost' IDENTIFIED BY 'springstudent';

GRANT ALL PRIVILEGES ON \* . \* TO 'springstudent'@'localhost';

ALTER USER 'springstudent'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'springstudent';

**File: 02-customer-tracker.sql**:

CREATE DATABASE IF NOT EXISTS `web\_customer\_tracker`;

USE `web\_customer\_tracker`;

DROP TABLE IF EXISTS `customer`;

CREATE TABLE `customer` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`first\_name` varchar(45) DEFAULT NULL,

`last\_name` varchar(45) DEFAULT NULL,

`email` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE = InnoDB AUTO\_INCREMENT = 6 DEFAULT CHARSET=latin1;

LOCK TABLES `customer` WRITE;

INSERT INTO `customer` VALUES

  (1,'David','Adams','david@luv2code.com'),

  (2,'John','Doe','john@luv2code.com'),

  (3,'Ajay','Rao','ajay@luv2code.com'),

  (4,'Mary','Public','mary@luv2code.com'),

  (5,'Maxwell','Dixon','max@luv2code.com');

UNLOCK TABLES;

**Database Connection**:

We are creating our new MySQL user for our application

* User Id: **springstudent**
* Password: **springstudent**

**Note**:

In my system there was a problem to create the user “**springstudent**” so I create the table inside another user name “**hbstudent**”.

* User Id: **hbstudent**
* Password: **hbstudent**

**Test Database Connection**:

1. Set up our Eclipse project
2. Add JDBC Driver for MySQL
3. Create “**TestDBServlet**” class and test connection

Add the following Jar file and run the project.

***mysql-connector-java-8.0.15.jar***

**Setup Development Environment**:

1. Copy starter config files
   1. spring-mvc-crud-demo-servlet.xml
   2. web.xml
2. Copy over JSTL libs
   1. Copy “**javax.servlet.jsp.jstl-1.2.1.jar**”
   2. Copy “**javax.servlet.jsp.jstl-api-1.2.1.jar”**
3. Copy latest Spring JAR files
   1. Copy from “**spring-framework-5.0.2.RELEASE\libs**”
4. Copy latest Hibernate JAR files
   1. Copy from “**hibernate-release-5.4.2.Final\hibernate-release-5.4.2.Final\lib\required**”
   2. Copy from “**hibernate-release-5.4.2.Final\hibernate-release-5.4.2.Final\lib\optional\c3p0**” (For Connection Pool)

**Configuration for Spring + Hibernate (Step-By-Step)**:

1. Define database dataSource / connection pool
2. Setup Hibernate session factory
3. Setup Hibernate transaction manager
4. Enable configuration of transactional annotations

**File: spring-mvc-crud-demo-servlet.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:context=*"http://www.springframework.org/schema/context"*

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

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

xsi:schemaLocation=*"*

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

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

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

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

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

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

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

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

<!-- Add support for component scanning -->

<context:component-scan base-package=*"com.odduu.ruhul"* />

<!-- Add support for conversion, formatting and validation support -->

<mvc:annotation-driven />

<!-- Define Spring MVC view resolver -->

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*

<property name=*"prefix"* value=*"/WEB-INF/view/"* />

<property name=*"suffix"* value=*".jsp"* />

</bean>

<!-- Step 1: Define Database DataSource / connection pool -->

<bean id=*"myDataSource"*

class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*

destroy-method=*"close"*>

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

<property name=*"jdbcUrl"*

value=*"jdbc:mysql://localhost:3306/web\_customer\_tracker?*

*useSSL=false&amp;serverTimezone=UTC"* />

<property name=*"user"* value=*"hbstudent"* />

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

<!-- these are connection pool properties for C3P0 -->

<property name=*"initialPoolSize"* value=*"5"* />

<property name=*"minPoolSize"* value=*"5"* />

<property name=*"maxPoolSize"* value=*"20"* />

<property name=*"maxIdleTime"* value=*"30000"* />

</bean>

<!-- Step 2: Setup Hibernate session factory -->

<bean id=*"sessionFactory"*

class=*"org.springframework.orm.hibernate5.LocalSessionFactoryBean"*>

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

<property name=*"packagesToScan"* value=*"com.odduu.ruhul.entity"* />

<property name=*"hibernateProperties"*>

<props>

<prop key=*"hibernate.dialect"*>

org.hibernate.dialect.MySQLDialect

</prop>

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

</props>

</property>

</bean>

<!-- Step 3: Setup Hibernate transaction manager -->

<bean id=*"myTransactionManager"*

class=*"org.springframework.orm.hibernate5.HibernateTransactionManager"*>

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

</bean>

<!-- Step 4: Enable configuration of transactional behavior based on annotations -->

<tx:annotation-driven

transaction-manager=*"myTransactionManager"* />

<!-- Add support for reading web resources: css, images, js, etc ... -->

<mvc:resources mapping=*"/resources/\*\*"*

location=*"/resources/"*>

</mvc:resources>

</beans>

**Step 1:- Define database dataSource / connection pool**:

<!-- Step 1: Define Database DataSource / connection pool -->

<bean id=*"myDataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*

destroy-method=*"close"*>

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

<property name=*"jdbcUrl"*

value=*"jdbc:mysql://localhost:3306/web\_customer\_tracker?*

*useSSL=false&amp;serverTimezone=UTC"* />

<property name=*"user"* value=*"hbstudent"* />

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

<!-- these are connection pool properties for C3P0 -->

<property name=*"initialPoolSize"* value=*"5"* />

<property name=*"minPoolSize"* value=*"5"* />

<property name=*"maxPoolSize"* value=*"20"* />

<property name=*"maxIdleTime"* value=*"30000"* />

</bean>

**Step 2:- Setup Hibernate session factory**:

Hibernate use SessionFactory to connect with the database.

<!-- Step 2: Setup Hibernate session factory -->

<bean id=*"sessionFactory"*

class=*"org.springframework.orm.hibernate5.LocalSessionFactoryBean"*>

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

<property name=*"packagesToScan"* value=*"com.odduu.ruhul.entity"* />

<property name=*"hibernateProperties"*>

<props>

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

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

</props>

</property>

</bean>

**Step 3:- Setup Hibernate transaction manager**:

When we write Hibernate code we always started a transaction in the transactional and so on. Spring has support where we can actually minimize the code in our DAO classes.

<!-- Step 3: Setup Hibernate transaction manager -->

<bean id=*"myTransactionManager"*

class=*"org.springframework.orm.hibernate5.HibernateTransactionManager"*>

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

</bean>

**Step 4:- Enable configuration of transactional annotations**:

Spring provide a special annotation called **@Transactional**, that allows us to minimize or eliminate some of our coding from manually starting and stopping transactions.

<!-- Step 4: Enable configuration of transactional behavior based on annotations -->

<tx:annotation-driven

transaction-manager=*"myTransactionManager"* />

Database connection and environment setup