**Configurare IntelIiJ**

1. Trebuie sa cream in package un servlet, insa el nu e disponibil cand dam click pe new file, de aceea in pom.xml adaugam:

<dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.1</version>  
</dependency>

1. Cream un servlet in path
2. Mai adaugam in pom.xml:

<dependency>  
 <groupId>com.mchange</groupId>  
 <artifactId>c3p0</artifactId>  
 <version>0.9.5.5</version>  
</dependency>

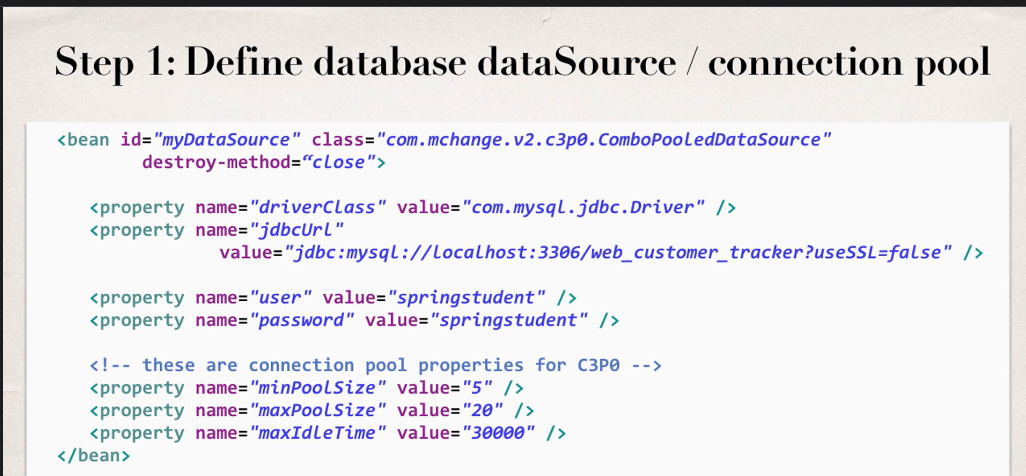
<dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-c3p0</artifactId>  
 <version>6.1.5.Final</version>  
 <type>pom</type>  
</dependency>

C3PO – database connection pool. Foloseste JDBC API pentru a crea conexiuni cu baza de date mai usor.

**Configuram C3PO si BeanFactory**

in applicationContext.xml cream beanuri:

* Cream data source sau connection pool:



Acest Bean ofera datele necesare pentru a se conecta la baza de date si a crea o Sesiune.Anume el controleaza sesiunile generate de sessionFactory, caci acest bean anume si e sesiunea ce va fi creata inca.

* destroy-method=”close” – se va asigura ca fiecare sesiune care va folosi acest Data Source sa fie mereu inchisa automat; Nu putem sa o punem la sessionFactory, asa cum ea se distruge tocmai la final de program, dar sesiunile se distrug cand nu mai avem nevoie de ele.
* minPoolSize – creaza din start conexiunile date si asa nu trebuie sa consumam apoi resurse pentru a le crea la fiecare nevoie, caci ele necesita resurse multe. Mai bine sa fie create cateva la rularea programului,
* maxIdleTime – timpul care arata peste cat timp se va inchide sesiunea automat daca aceasta nu mai e folosita.
* Cream un SessionFactory



Hibernate foloseste Session Factory pentru a crea sesiuni ce se conecteaza la baza de date si comunica cu ea. Cream bean in loc sa tot cream un obiect mereu.

* dataSource – e beanul creat anterior ce contine datele despre baza de date. Nu mai citeste automat fisierul hibernate.cfg.xml Hibernate cand e vorba de bean deja. Acest bean va crea mereu cate o sesiune la baza de date pe baza la source bean.
* packageToScan – va scana tot package pentru a gasi @Entity classes. Asa nu trebuie sa le mai adaugam noi manual de fiecare data, dar vor fi adaugate toate odata.
* Acesta sessionFactory se va asigura sa returneze aceeasi sesiune pentru mai multe DAO dintr-o metoda @Transactional

**<property> <props> si <prop>**

* <property> poate fi folosit nu numai pentru metodele setter. Cand nu oferim nici-un parametru, adica ref=, Hibernate nu va gasi nici-o metoda ce sa inceapa cu set, asa cum ele mereu au parametri. Insa putem seta fielduri din acel bean, adica obiecte ce le are ca membri,de ex

<property name=”field”>

<props>

<prop key=”membru1”>valoare</prop>

<prop key=”membru2”>valoare</prop>

<props>

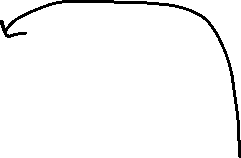
</property>

* Aici <property>e folosit pentru a arata ca beanul nostru are un obiect membru in el, cu numele field
* apoi setam membrii din field cu <props> </props>
* cu <prop key=”membru”> setam valoarea fieldului din field al beanului. Punem ref daca dam o referinta la alt bean, dar aici putem da si un String, ca mai sus.

**public** **class** ClassRoom {

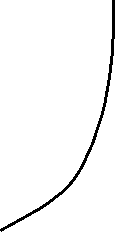
**public** Properties students ;

**public** **void** setStudents(Properties students) {



**this**.students = students;

        }



}

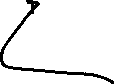
        <bean class=*"com.java4coding.ClassRoom"*>

                <property name=*"students"*>

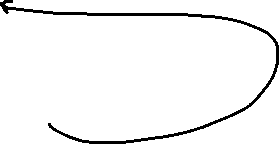
                        <props>



                                <prop key=*"name"*>Manu Manjunatha</prop>



                                <prop key=*"id"*>123</prop>



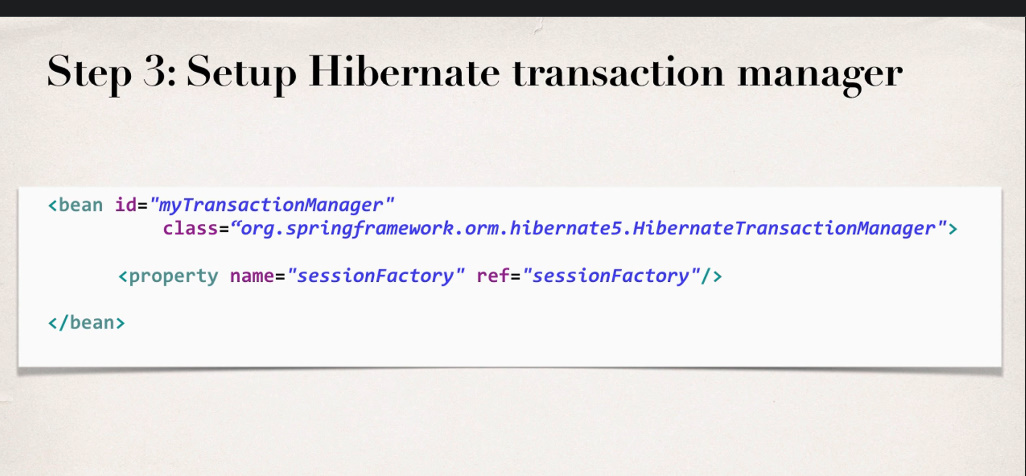
                                <prop key=*"marks"*>100</prop>

                        </props>

                </property>

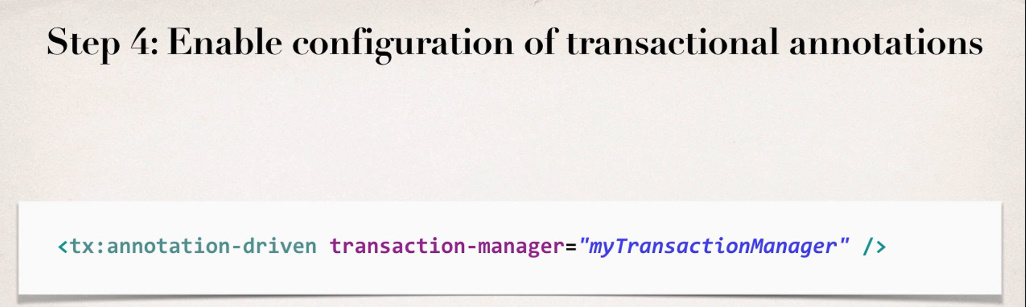
        </bean>

* Setam Hibernate Transaction Manager



Asa nu trebuie mereu sa incepem o tranzactie cand lucram cu o sesiune si nici mereu sa dam commit la tranzactie. Putem elimina codul. Dam referinta la sesionFactory creata. Acest bean gestioneaza fiecare tranzactie din sesiunea creata de sessionFactory.

* Aici permitem folosirea anotatiei @Transactional. Asa nu trebuie mereu sa incepem o tranzactie si sa inchidem sesiunea si nici sa tot dam commit



trasnaction-manager aici punem id la bean Transaction setat. Asa, anotatia @Transaction e va asigura sa foloseasca beanul myTrnsactionManager, care are referinta la SessionFactory, pentru a deschide si inchide automat sesiunile create de noi si a gestiona tranzactiile.

Config final va fi asa:

<?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:tx="http://www.springframework.org/schema/tx"  
 xmlns:context="http://www.springframework.org/schema/context"  
 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/cache http://www.springframework.org/schema/cache/spring-cache.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/mvc https://www.springframework.org/schema/mvc/spring-mvc.xsd">  
 <tx:annotation-driven transaction-manager="TransactionManager"/>  
 <context:component-scan base-package="com.java.MyApp"/>  
 <mvc:annotation-driven/>  
  
 <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">  
 <property name="driverClass" value="com.mysql.jdbc.Driver"/>  
 <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/web\_customer\_tracker?useSSL=false"/>  
 <property name="user" value="testuser"/>  
 <property name="password" value="Frb2eshox!"/>  
  
 <property name="minPoolSize" value="5"/>  
 <property name="maxPoolSize" value="20"/>  
 <property name="maxIdleTime" value="30000"/>  
 </bean>  
  
 <bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">  
 <property name="dataSource" ref="dataSource"/>  
 <property name="packagesToScan" value="com.java.MyApp"/>  
 <property name="hibernateProperties">  
 <props>  
 <prop key="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</prop>  
 <prop key="hibernate.show\_sql">true</prop>  
 </props>  
 </property>  
 </bean>  
 <bean name="TransactionManager" class="org.springframework.orm.hibernate5.HibernateTransactionManager">  
 <property name="sessionFactory" ref="sessionFactory"/>  
  
 </bean>  
  
 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <property name="prefix" value="/WEB-INF/view/"/>  
 <property name="suffix" value=".jsp"/>  
 </bean>  
</beans>