Full working Spring MVC and Hibernate which commincate to db

CRM-- Customer Relation Project

- track customer

-add, udpate ,delete and list customer

Setup DB environment

- list Consumers

- save new

-update

-delete

Define table

1 create user

2 customer\_tracker

Create DB and SCehame create Table Customer and load with some simple data

Schema **crm**

create table

create table customer(id serial PRIMARY KEY, first\_name VARCHAR(50),last\_name VARCHAR(50), email VARCHAR(50))

insert into customer VALUES

(1,'sachin','hs','sachin.hs@bcits.in'),

(2,'soumya','indi','soumya.indi@bcits.in'),

(3,'darshan','gowda','darshan@bcits.in'),

(4,'manasa','gowda','manasa@bcits.in'),

(5,'diyan','gowda','diyan@bcits.in')

Test DB connection

1 setup eclipse

2 add jdbc driver for postgress also servlet Api if its Servlet program

3 santiy test to make sure we can connect

Set up Web Environment

1 copy starter config files

* Web.xml and spring config

2 copy jstl lib

3 copy spring jars

4 copy hibernate jars (required jars also optional of c3p0 jars for connection pool)

**JAVA 9 and higher HEADS UP- SPRING MVC CRUD**

For Java 9 and higher, you need to additional JAR files.

You need to download the following JAR files:

javax.activation-1.2.0.jar

jaxb-api-2.3.0.jar

jaxb-core-2.3.0.jar

jaxb-impl-2.3.0.jar

Configuration for spring and hibernate

1 define database datasource / connection pool

2 setup hibernate session factory

3 setup hibernate transaction manager

4 enable configuration of transactional manager

Add all these in spring dispatcher servlet

1 Define Database Datasource /connection pool

Hibernate use connection pool for connecting database

<bean id=*"myDataSource"* class=*"com.mchnage.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

Next data base confirmation information

<property name=*"driverClass"* value=*"org.postgresql.Driver"*/>

<property name=*"jdbcUrl"* value=*"jdbc:postgresql://localhost:6412/personal?currentSchema=crm"*/>

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

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

Config connection pool for c3p0

We can specify the minium connection pool

<!-- connection properties of c3p0 -->

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

<property name=*"maxPoolSize"* value=*"25"*/>

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

2 Hibernate SessionFactory

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

Session Factory used to connect DB

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

// refers to datascours / connection pool

<property name=*"packageToScan"* ref=*"com.crm.entity"*/> // scan for Hiberanate @Entity

Add hibernate properties add we can add any extra propeties if required

<property name=*"hibernateProperties"*>

<props>

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

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

</props>

</property>

3 Hibernate TranactionManager

When we write a Hibernate we alwys start a transaction and end a TRnasction

Spring support to minimize it in DAO class : use Spring framwork Hibernate Transaction manager

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

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

</bean>

4 Enable configuration of Transactional annotaions

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

Spring provides spl annotaion @Transactional allows to minimize or eliminate some code fro manualing statring and stoping of transaction

Final Dispatcher-servelt will be like with all spring Propeties and Hibernate Propeties

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

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

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

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/util*

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

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

<!-- Step 3 add Support for componenet Scanning -->

<context:component-scan base-package=*"com.sample.controller"*></context:component-scan>

<!-- Step 4 Add support for Conversion , formatting and validation and support -->

<mvc:annotation-driven/>

<!-- Addind resource mapping like JS ,css and images -->

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

<!-- Step 5 Define Spring MVC view Resolver -->

<bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

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

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

</bean>

<!-- Defining Database datasource , our connection pool -->

<bean id=*"myDataSource"* class=*"com.mchnage.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"org.postgresql.Driver"*/>

<property name=*"jdbcUrl"* value=*"jdbc:postgresql://localhost:6412/personal?currentSchema=crm"*/>

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

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

<!-- connection properties of c3p0 -->

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

<property name=*"maxPoolSize"* value=*"25"*/>

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

</bean>

<!-- Session Factory To Communicate with our DB -->

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

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

<property name=*"packageToScan"* ref=*"com.crm.entity"*/>

<property name=*"hibernateProperties"*>

<props>

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

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

</props>

</property>

</bean>

<!-- Hibernate Trnaction Manager -->

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

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

</bean>

<!-- Enable Congifurtaion Transaction Manager -->

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

<util:properties id=*"cityOptions"* location=*"classpath:../countries.properties"* />

<bean id=*"messageSource"*

class=*"org.springframework.context.support.ReloadableResourceBundleMessageSource"*>

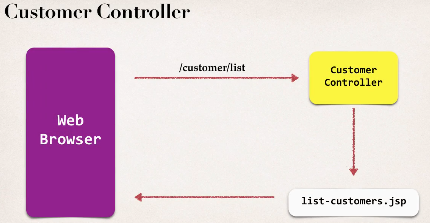
<!-- <property name="basename" value="/WEB-INF/messages" /> -->

<property name=*"basename"* value=*"classpath:/resources/messages"* />

</bean>

</beans>

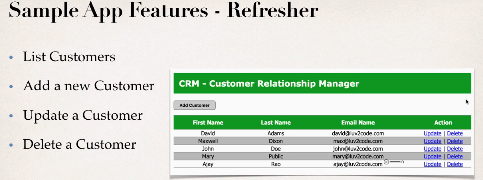
Test Spring MVC Controller

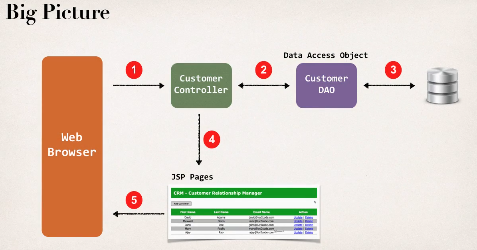


<absolute-ordering /> add these in web.xml after display name?

java.lang.IllegalArgumentException: More than one fragment with the name [spring\_web] was found. This is not legal with relative ordering. See section 8.2.2 2c of the Servlet specification for details. Consider using absolute ordering.

Note : hibernate realase should not be greater then Spiring relase r else it will not build





Customer Data Access Object

- responsible for interfacing with the db

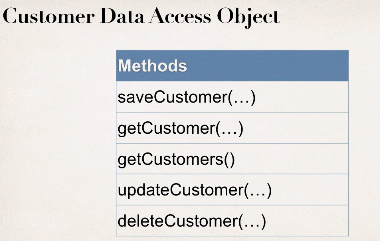
- design pattern DAO (DATA ACCESS OBJECT)



- helper / utility class to db

- uses hibernate API for accessign data

CRUD Operation required



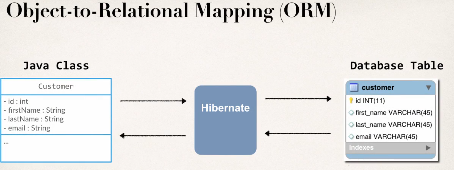
Development Process

1 create Customer class -Entity: java class that is mapped to a db table

2 Create CustomerDAO and CustmoerDAOImpl

3 CustmoerController

1 create Customer class



A : map class to table

B constructor ,getter setter toString

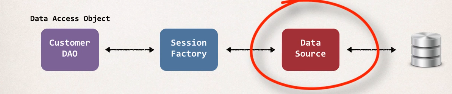
2 Create DAO

For hibernate , our DAO need Hibernate SessionFActory

- the sessionFactory needs a DataSource

-dataSource defines db connection info

These are all Dependnecis we can wire them using DI



CustomerDAo

1 define DAO interface

2 Define DAO impl

- inject sessionFactory

Spring @Tranactionl

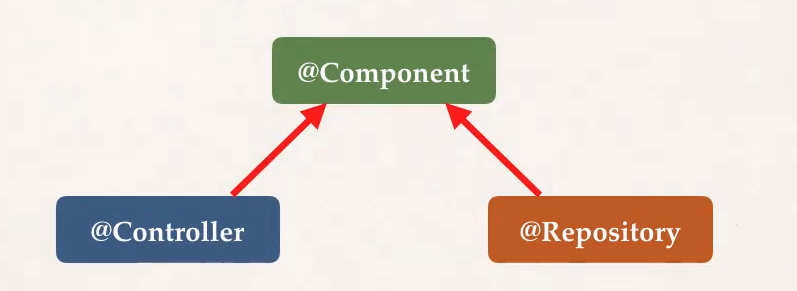
-Spring provides @Transactional annotaion : which automatically begins and end tranaction of hibernate code

- no need of explictly doing in your code spring handle everything in back

Place it begin of your method in DAOimp::l @Tranactionl

Special Annoations for DAO

@Repositry use it at top DaoImpl for DAO impl it inherits from @Component so it is enabled for auto Scanning



@Repositry : applied to DAO impl

Spring will automatically register daoimpl thanks to component scan

Spring also providestrnasaltion any jdbc related exceptions

@Respositry : for specific code for talk to back end data source

Use hibernate 5.2 bcz Query<Customer>, .getResultList() are not availbale in below version r else use .list()

Query<Customer> query=session.createQuery("from Customer");

// get the result list

//List<Customer> customers=query.getResultList();

List<Customer> customers= session.createQuery("from Customer").list();

Inject DAO into Controller

Get Customer data

Add in model

Return it to jsp create JSP

Apply css

Development process

1 place css in resource directory

2 configure Spring to servce up resources directory

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

location=*"/resources/" : physical directory name*

Mapping=*"/resources/\*\*" : url mapping to recurse subdirectory*

3 refer css in jsp

<link type=*"text/css"* rel=*"stylesheet"* href=*"*${pageContext.request.contextPath}*/resources/css/style.css"* />

Head of html/jsp

${pageContext.request.contextPath} : gives correct name of our application

Both applies for javascript ,images,pdfs etc

Same like resource folder config we can make separte config for all other features

Add welcome file

:server will look for wlecome file if not 404 config in web.xml