Mini Project#2-Spring WEB MVC(Consumer)+ JAX-RS with Hibernate(Producer)

----------Full Code for Producer App----------

1.pom.xml file:

Depencencies:

Jersey Container Servlet
Jersey HK2
Jersey Media JSON Jackson
MySQL Connector
Hibernate Core
Project Lombok

```
<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<maven.compiler.source>15</maven.compiler.source>
		<maven.compiler.target>15</maven.compiler.target>
	</properties>

	<dependencies>
		<!-- https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
		<dependency>
			<groupId>com.jslsolucoes</groupId>
			<artifactId>ojdbc6</artifactId>
			<version>11.2.0.1.0</version>
		</dependency>

		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>4.11</version>
			<scope>test</scope>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet -->
```

```
		<dependency>
			<groupId>org.glassfish.jersey.containers</groupId>
			<artifactId>jersey-container-servlet</artifactId>
			<version>2.30</version>
		</dependency>
		<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
		<dependency>
			<groupId>org.glassfish.jersey.inject</groupId>
			<artifactId>jersey-hk2</artifactId>
			<version>2.30</version>
		</dependency>
		<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<version>1.18.18</version>
			<scope>provided</scope>
		</dependency>
```

1. Model class:

```
package in.nit.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;
@Data
@Entity
@Table(name="product")
public class Product {
@Id
@GeneratedValue
@Column(name="pid")
private Integer prodId;

@Column(name="pcode")
```

```
        private String prodCode;

        //@Column(name="pcost")
        private Double prodCost;

        @Column(name="pdisc")
        private Double prodDiscount;

        @Column(name="pgst")
        private Double prodGst;
        }
```

3.Configuration File:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
                "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
                "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
   <session-factory>
       <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</prop
erty>
       <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/webservic
es</property>
       <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">root</property>

       <property
name="hibernate.dialect">org.hibernate.dialect.MySQL55Dialect</prope
rty>
       <property name="hibernate.show_sql">true</property>
       <property name="hibernate.format_sql">true</property>
       <property name="hibernate.hbm2ddl.auto">create</property>
       <mapping class="in.nit.model.Product"></mapping>
   </session-factory>
</hibernate-configuration>
```

4.HibernateUtil

```
package in.nit.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
private static SessionFactory sf;
static {
	try {
		sf=new
Configuration().configure("/in/nit/cfgs/hibernate.cfg.xml").buildSessi
onFactory();
	} catch (Exception e) {
		e.printStackTrace();
	}
}
	public static SessionFactory getSf() {
		return sf;
	}
}
```

5.dao:

```
package in.nit.dao;

import in.nit.model.Product;

public interface IProductDAO {
public Integer saveProduct(Product p);
}
```

6.impl:

```
package in.nit.dao.impl;

import org.hibernate.Session;
import org.hibernate.Transaction;

import in.nit.dao.IProductDAO;
import in.nit.model.Product;
import in.nit.util.HibernateUtil;
```

```
public class ProductDAOImpl implements IProductDAO {

	@Override
	public Integer saveProduct(Product p) {
		Session ses=HibernateUtil.getSf().openSession();
		Transaction tx=null;
		Integer id=null;
		try(ses) {
			tx=ses.beginTransaction();
			id=(Integer)ses.save(p);
			tx.commit();
		} catch (Exception e) {
			if(tx!=null && tx.getStatus().canRollback()) {
			tx.rollback();
			}
			e.printStackTrace();
		}
		return id;
	}

}
```

7.service:

```
package in.nit.service;

import in.nit.model.Product;

public interface IProductService {
public Integer saveProduct(Product p);
}
```

8.impl:

```
package in.nit.service.impl;
import javax.inject.Inject;

import in.nit.dao.IProductDAO;
import in.nit.model.Product;
import in.nit.service.IProductService;

public class ProductServiceImpl implements IProductService {
```

```
   @Inject
        private IProductDAO dao;

        @Override
        public Integer saveProduct(Product p) {
                var cost=p.getProdCost();
                var discount=cost*8/100.0;
                var gst=cost/6*100.0;
                p.setProdGst(gst);
                p.setProdDiscount(discount);

                return dao.saveProduct(p);
        }

}
```

9.AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.dao.IProductDAO;
import in.nit.dao.impl.ProductDAOImpl;
import in.nit.service.IProductService;
import in.nit.service.impl.ProductServiceImpl;


@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {
        public AppConfig(){
        packages("in.nit");
        register(new AbstractBinder() {
                public void configure() {
                        //IProductDAO dao=new ProductDAOImpl();

        bind(ProductDAOImpl.class).to(IProductDAO.class);

        bind(ProductServiceImpl.class).to(IProductService.class);
        }
```

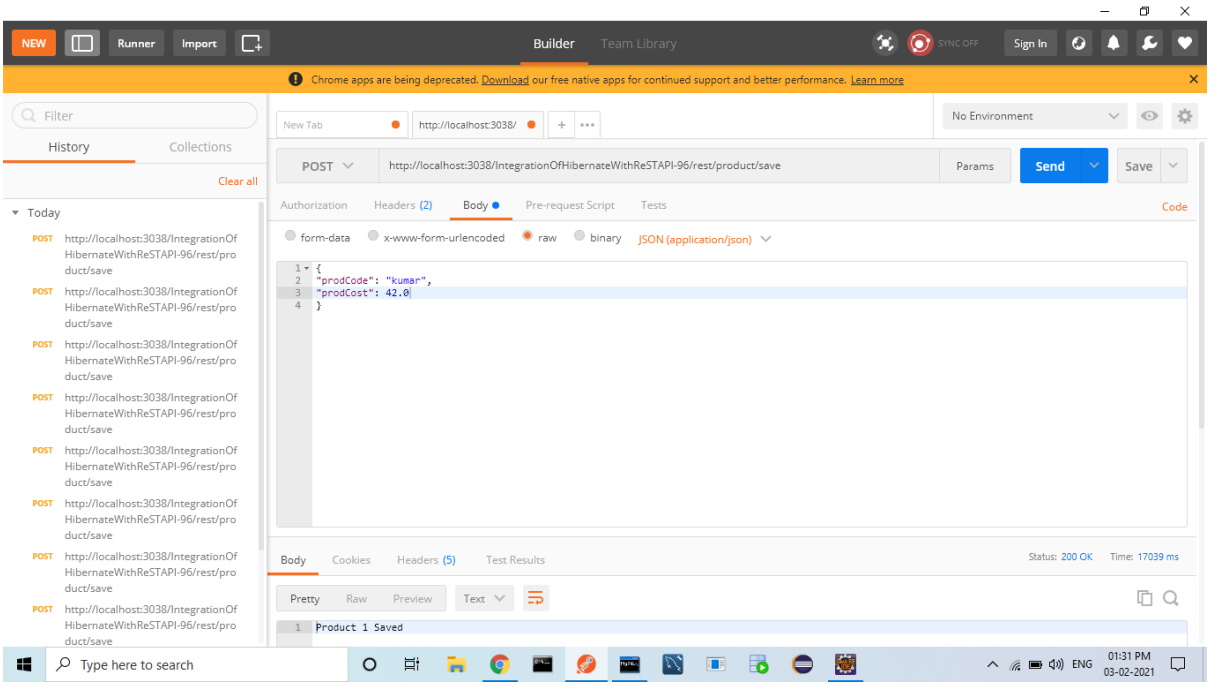
```
	});
	}
}
```

10.RestController:

```
package in.nit.controller;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Product;
import in.nit.service.IProductService;

@Path("/product")
public class ProductRestController {
@Inject
private IProductService service;
//1.Save Products
@POST
@Path("/save")
@Consumes("application/json")
public Response saveProduct(Product p) {
	Response resp=null;
	Integer id=null;
	try {
		id=service.saveProduct(p);
		resp=Response
		.status(Status.OK)
		.entity("Product "+id+" Saved")
		.build();
	} catch (Exception e) {
		resp=Response
		.status(Status.INTERNAL_SERVER_ERROR)
		.entity("Product unable to Save")
		.build();
		e.printStackTrace();
	}
	return resp;
```

}
}

----------Output----------


NEW
Runner
Import
Builder
Team Library
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
Clear all
Today
POST http://localhost:3038/IntegrationOfHibernateWithReSTAPI-96/rest/product/save
New Tab
http://localhost:3038/
No Environment
POST
http://localhost:3038/IntegrationOfHibernateWithReSTAPI-96/rest/product/save
Params
Send
Save
Authorization
Headers (2)
Body
Pre-request Script
Tests
Code
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{
"prodCode": "kumar",
"prodCost": 42.0
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 17039 ms
Pretty
Raw
Preview
Text
Product 1 Saved
Type here to search
ENG
01:31 PM
03-02-2021