**Steps to configure & create RESTful Services with Spring and JPA/Hibernate**

1. Create a Dynamic Web Project or Maven project in eclipse.
2. Create an interface (RESTful Service) with the required apis and an implementation class which provides the implementations for the apis defined in the interface.
3. In the webapp/WEB-INF folder (in maven project it is src/main/webapp/WEB-INF folder) , open the web.xml and add the below entries. Since I have used apache cxf api for the RESTful service implementation, the corresponding entries are made in the web.xml

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/context.xml</param-value>

</context-param>

<listener>

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

</listener>

<servlet>

<servlet-name>CXFServlet</servlet-name>

<servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>CXFServlet</servlet-name>

<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

1. Then create a spring context file called context.xml under WEB-INF folder and add the below entries.

**<?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:jaxrs="http://cxf.apache.org/jaxrs" xmlns:jpa="http://www.springframework.org/schema/data/jpa" xmlns:context="http://www.springframework.org/schema/context"**

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

**xsi:schemaLocation="**

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

**http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd**

**http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.1.xsd**

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

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

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

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

**default-lazy-init="false">**

**<context:annotation-config />**

**<context:component-scan base-package="com.keerthi.restservice.demo" />**

**<context:component-scan base-package="com.employee.\*" />**

**<jpa:repositories base-package="com.employee.repositoryservice" />**

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

**<import resource="classpath:META-INF/cxf/cxf.xml" />**

**<import resource="classpath:META-INF/cxf/cxf-extension-jaxrs-binding.xml" />**

**<import resource="classpath:META-INF/cxf/cxf-servlet.xml" />**

**<!—Injecting the services which acts as RESTful services-- >**

**<jaxrs:server id="helloRestServiceMgt" address="/">**

**<jaxrs:serviceBeans>**

**<ref bean="helloRestService" />**

**<ref bean="employeeRestService" />**

**</jaxrs:serviceBeans>**

**</jaxrs:server>**

**<!-- 1. Create datasource from JNDI DataSource provided by the container -->**

**<!-- <jee:jndi-lookup id="myDataSource" jndi-name="java:/mydb" expected-type="javax.sql.DataSource" /> -->**

**<!-- HSQL-DB memory database; for testing only -->**

**<bean id="myDataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">**

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

**<property name="url" value="jdbc:h2:mem:test;DB\_CLOSE\_DELAY=-1" />**

**</bean>**

**<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">**

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

**<property name="persistenceUnitName" value="my-jpa" />**

**<!--FOR MySQL -->**

**<!-- <property name="jpaVendorAdapter">**

**<bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">**

**<property name="showSql" value="true" />**

**<property name="generateDdl" value="true" />**

**<property name="database" value="MYSQL" />**

**</bean>**

**</property> -->**

**<!-- FOR IN-MEMORY DB -->**

**<property name="jpaVendorAdapter">**

**<bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">**

**<property name="generateDdl" value="true" />**

**<property name="showSql" value="true" />**

**<property name="databasePlatform" value="org.hibernate.dialect.H2Dialect" />**

**</bean>**

**</property>**

**</bean>**

**<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">**

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

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

**</bean>**

**<bean id="org.h2.tools.Server-WebServer" class="org.h2.tools.Server"**

**factory-method="createWebServer" depends-on="myDataSource"**

**init-method="start" lazy-init="false">**

**<constructor-arg value="-web,-webPort,11111" />**

**</bean>**

**</beans>**

1. Whenever a new rest api is created, corresponding registering of the service bean should be made under the <jaxrs:serviceBeans> tag of the context.xml file.
2. The required jars like Spring jars, hibernate jars, jpa jars should be added to the classpath either through pom.xml by defining the dependencies in the pom.xml file or copying the jars to the lib folder under WEB-INF and manually setting them to the class path.
3. Maven dependency entry for the apache cxf api:

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-frontend-jaxrs</artifactId>

<version>2.2.3</version>

</dependency>

1. Whenever spring’s NoUniqueBeanFoundException is thrown while running the app, check for the version of the spring-beans. Jar . It should be 3.2.1 or higher !