Order Process System(Restful webservice)

Design notes:

The primary technologies used are spring STS ide(3.7); maven build tool; Stax parser; Oracle 12c; PL/SQL stored proc using custom object types and oracle collections; Spring MVC(spring 4 version); multithreading for server side custom caching. . ['buildship' will replace sts gradle plugin soon.10/14/2016. Read, https://github.com/eclipse/buildship/wiki/Migration-guide-from-STS-Gradle-to-Buildship]

1.

2. **This w/service development consists of 2 services. 1) createorder:** Create order option is to upload the data file (Its an xml file!) to the server. The successful operation produces JSON response. 2) searchOrder: the orderid is consumed and produces the order(JSON) object
3. The xml file input is parsed at the server side using Stax. The parsed data is saved as an "Order" object and the same is used to data persistence to the Oracle DATABASE. Data serialization option is also available as an alternative approach to persist data when the Oracle RDBMS is not setup.
4. The data is looked up using JDBC / PLSQL stored procedure call. The json data is handed over to the client upon successful lookup. All the calls invoked to execute the stored procedures logs the entire database activity on the server side disk. There should be Directory exists for UTL_FILE to save the log files and also sufficient DBA permissions are granted for that user.
5. Caching is implemented at server side for fast accessing the recently searched orders and this improves application performance while offering the fast response to the users. Implemented multithreading option.

6. **How to run the application:**
   1) **Two options to save the data; (a) without Oracle RDBMS system (serialize the Order-data object to the hard disk on server computer). Or (b) save it into Oracle database.**

      I (a). **Serialize the data object to disk:** since 'programing to interface' pattern is followed, it is very simple to switch between I.(a) and I.(b). The xml data is initialized to 'order' object instance. This object is serialized and written to disk. Although this is functional but less responsive unless to test the UI/front end development. Also this feature is useful to test the flow when the Oracle RDBMS is not accessible.

      The buz logic (search and save order) is implemented in '`OrderSerializationHelper.java`'.
      `@Repository("orderDao"):` Uncomment this annotation to enable bean creation for the above java file.
      At the same time comment out the same line in all files, '`OrderDBHelper*.java`'
      II (i). **Oracle RDBMS:** Swap to Oracle mode by reversing as mentioned in the above step, I. That is, uncomment `@Repository("orderDao")` in only one of the files

'`OrderDBHelper*.java`' and comment out the rest of the "`orderDao`" java files. The buz logic (search and save order) is implemented in all '`OrderDBHelper*.java`'.

Prepare database: (a) Create the database tables, custom objects, and Sequence and (b) Create the oracle stored procedures

ii) **import the project** files into workspace. Prepare the project to be eclipse loaded if using java sources. Run, **"`mvn clean compile install`"** to build the war file. Deploy this to tomcat server(8.0.28) and start the server. When the server is up, test the w/service by launching the web browser or some other w/services tool, and enter the url: `http://localhost:9990/OrderProcessRestService/search/27268` (this is as per my config. It may change depending on the respective machine configuration)

iii) **oracle db properties: change accordingly in the file, 'application.properties'**

```
user=venu
pass word=Summer2015
url="jdbc:oracle:thin:@localhost:1521:LOGISTICS"
```

7. **i)** **Sample xml data file used for loading into the system:**

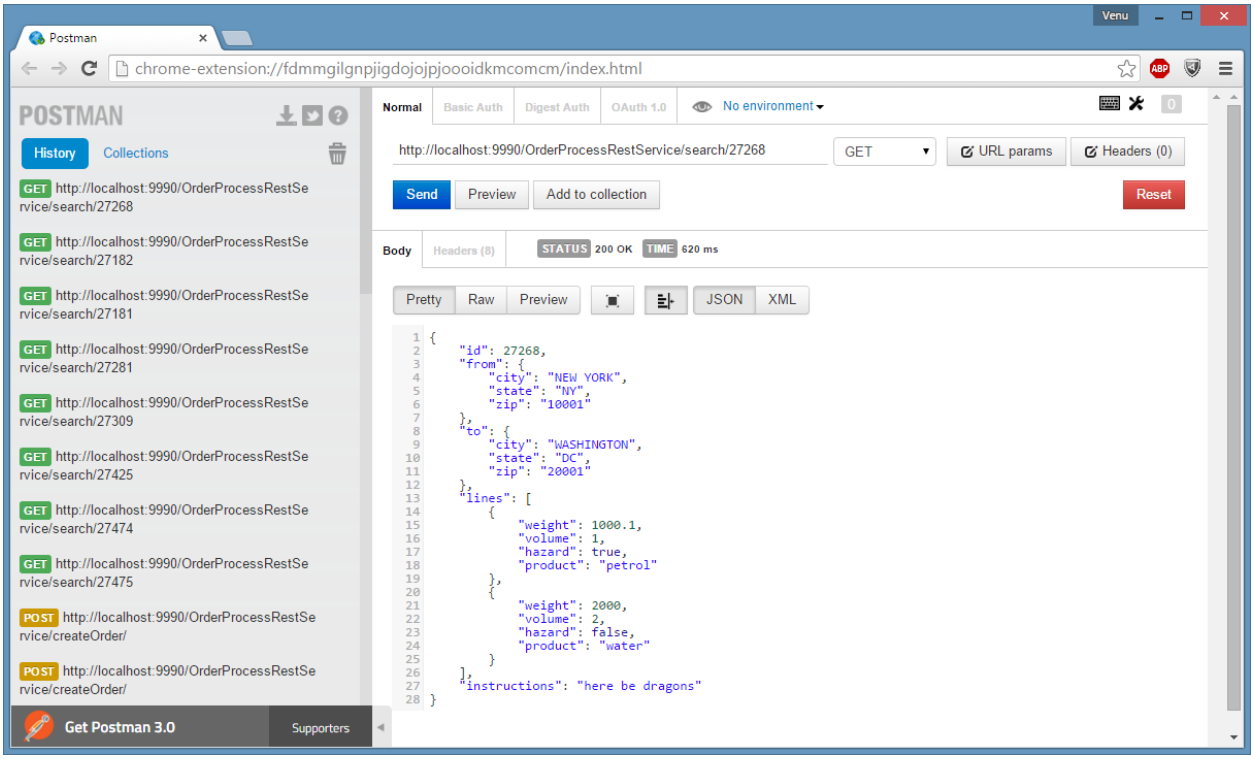
```
<?xml version="1.0" encoding="utf-8"?>
<order>
   <from zip="80817" state="CO" city="COLORADAO SPRINGS"/>
   <to zip="96821" state="HI" city="Honolulu"/>
   <lines>
      <line weight="10000.1" volume="14" product="Engine Block"/>
      <line weight="200.55" volume="8" hazard="true" product="cable"/>
      <line weight="100.1" volume="14" hazard="false" product="plugs"/>
      <line weight="165" volume="8" hazard="false" product="electronic controls"/>
      <line weight="1008.1" volume="14" hazard="false" product="Engine Block"/>
      <line weight="30.55" volume="8" hazard="true" product="Liquid Nitrogen"/>
   </lines>
   <instructions>Transport in secure container</instructions>
</order>
```

ii) **Schema(order.xsd)**

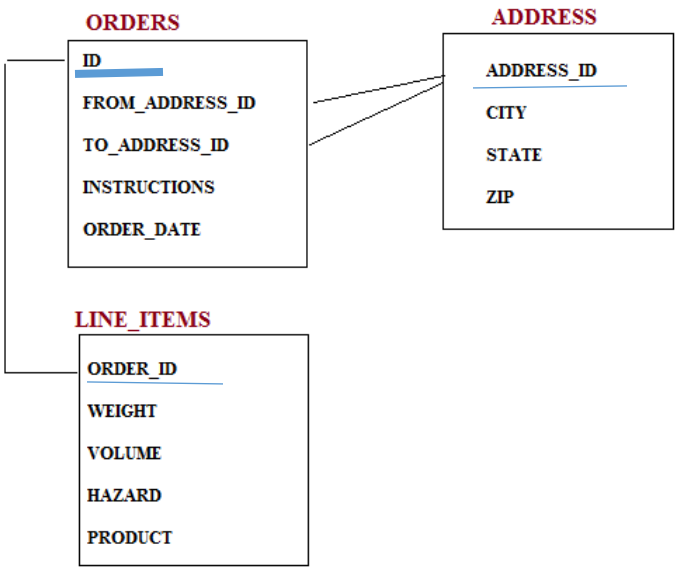
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"   elementFormDefault="qualified">
   <xs:complexType name="order">
      <xs:sequence>
         <xs:element name="from" type="location"/>
         <xs:element name="to" type="location"/>
         <xs:element name="lines" type="lines"/>
         <xs:element name="instructions" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="location">
      <xs:attribute name="city" use="optional" type="xs:string"/>
      <xs:attribute name="state" use="required" type="xs:string"/>
      <xs:attribute name="zip" use="required" type="xs:string"/>
   </xs:complexType>
   <xs:complexType name="lines">
      <xs:sequence>
         <xs:element name="line" maxOccurs="unbounded" type="line"/>
      </xs:sequence>
   </xs:complexType>
```

```
  <xs:complexType name="line">
    <xs:attribute name="hazard" use="optional" default="false" type="xs:boolean"/>
    <xs:attribute name="product" use="required" type="xs:string"/>
    <xs:attribute name="volume" use="required" type="xs:double"/>
    <xs:attribute name="weight" use="required" type="xs:double"/>
  </xs:complexType>
  <xs:element name="order" type="order"/>
</xs:schema>
```

8. **Testing REST service**: Google chrome extension, 'Postman' can be used for testing the restful service. The set up is quick and simple. Below is the screen shot for GET call the JSON is produced. Shown is "GET" URI resource - http://localhost:9990/OrderProcessRestService/search/27268.



9. DATAMODEL: Screen: 6


ORDERS
ID
FROM_ADDRESS_ID
TO_ADDRESS_ID
INSTRUCTIONS
ORDER_DATE
ADDRESS
ADDRESS_ID
CITY
STATE
ZIP
LINE_ITEMS
ORDER_ID
WEIGHT
VOLUME
HAZARD
PRODUCT