Order Process System

Design notes:

1. **The primary technologies used are html5/css/ajax/ angular.js, BootStrap framework, google –gson for UI development.**
   spring STS ide(3.7); gradle build tool; Stax parser; Oracle 12c; PL/SQL; Spring MVC(spring 4 version) for backend

2. UI development: The Screens are developed using angular /ajax to meet the need of 'Single Page Application'. The pagination, Sorting and fetching json data from the server are implemented in the search screen. The entire SPA is developed using angular.js and as of now only 'search orders' is implemented which is a bit more complex

3. The data is looked up using JDBC / PL/SQL stored procedure call. Multiple network calls can be avoided which makes the application more responsive. The PL/SQL is very efficient/fast. The stored procedure is called only once for a given request.
4. For search order task, Data Look up is done using one single sql statement. The returned data is used to make the Order instance. I used this to convert to json data and handed over to the client side. The json object is parsed to show the data as collapsible/expandable list (tree list).
   This is accomplished by just jquery /css. No third party libraries used.

1. **My thoughts**: current project is a model to demonstrate only the working features of the application. The following features can still be added for making the application more responsive and efficient.
   1)
   Screen look and feel: The Order jsp page can be further improved for look and feel. Also, user can be given one more option of entering the order details in the text input. Spring Boot can also be used to develop the java application for rapid application development.

2. **How to run the application:**
   1) **Prepare database.**
      Create the database tables
      Create the Sequence
      Create the oracle stored procedure
   2) **import the project** file into your workspace. Prepare the project to eclipse loaded if using java sources (run command, gradle eclipse). Run, **"gradle jettyRun"**. When the server is up, launch the web browser, and enter the url: http://localhost:9999/venu/order.do (this is as per my config. It may slightly change depending on your configuration)
   3) **oracle db properties:** user=venu
      pass word=Summer2015
      url="jdbc:oracle:thin:@localhost:1521:LOGISTICS"

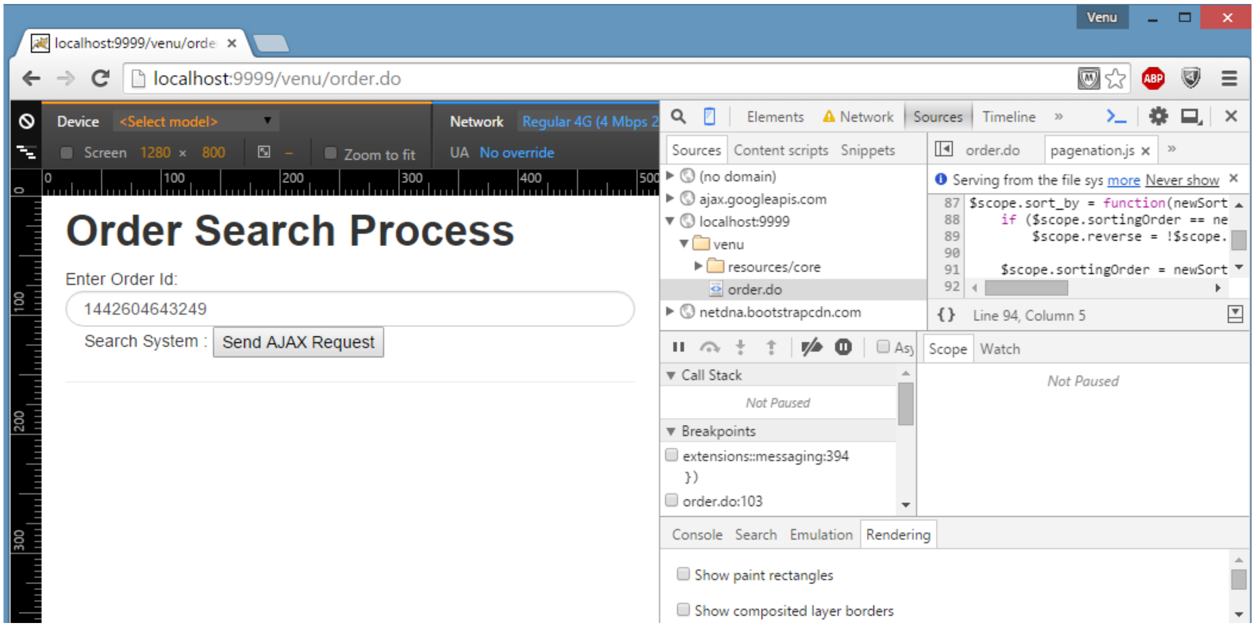
3. Below are the screen shots
4. Sample xml data file loaded in the system:
5. <?xml version="1.0" encoding="utf-8"?>
6. <order>

```
    <from zip="80817" state="CO" city="COLORADAO SPRINGS"/>
    <to zip="96821" state="HI" city="Honolulu"/>
    <lines>
        <line weight="10000.1" volume="14" hazard="false" 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>
```

19. 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="optional" 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>
```

Screen1 (initial screen)



localhost:9999/venu/order.do

Order Search Process

Enter Order Id: 1442604643249

Search System : Send AJAX Request

Serving from the file sys more Never show

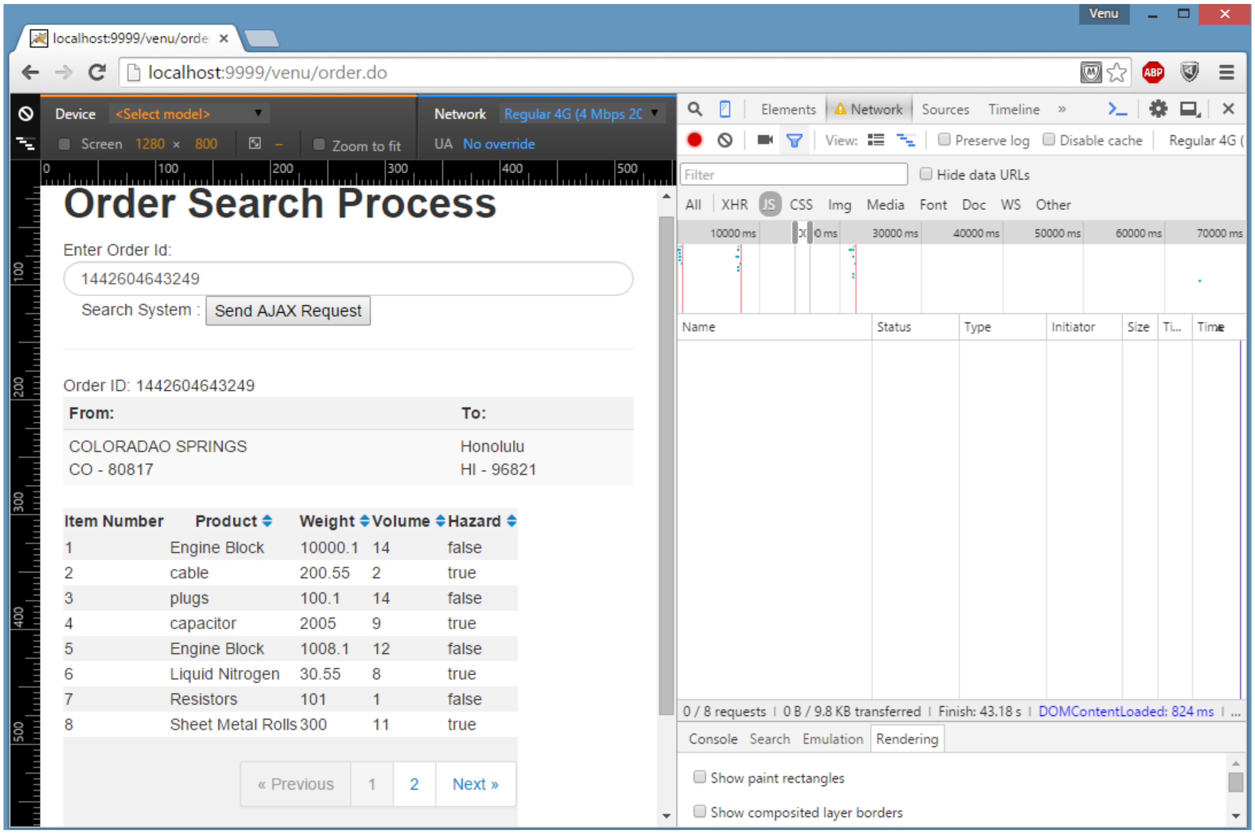
```
$scope.sort_by = function(newSort
    if ($scope.sortingOrder == ne
        $scope.reverse = !$scope.

    $scope.sortingOrder = newSort
```

Line 94, Column 5

Screen:4 (results of search : Table structure)

Results are displayed in a table structure. Sorting is available for all the columns except the first column as this is the row number indicator only



Order Search Process

Enter Order Id:

1442604643249

Search System : Send AJAX Request

Order ID: 1442604643249

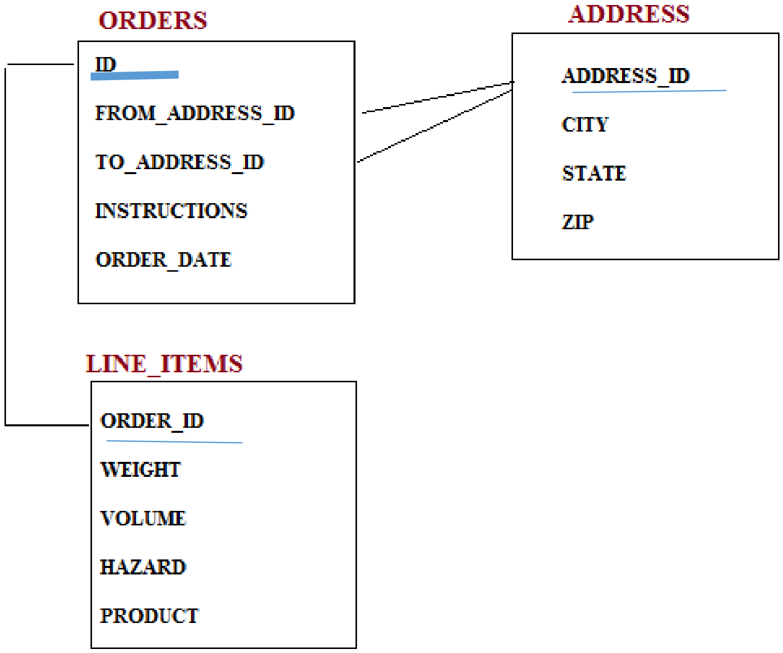
From:	To:
COLORADAO SPRINGS CO - 80817	Honolulu HI - 96821

Item Number	Product ⬍	Weight ⬍	Volume ⬍	Hazard ⬍
1	Engine Block	10000.1	14	false
2	cable	200.55	2	true
3	plugs	100.1	14	false
4	capacitor	2005	9	true
5	Engine Block	1008.1	12	false
6	Liquid Nitrogen	30.55	8	true
7	Resistors	101	1	false
8	Sheet Metal Rolls	300	11	true

« Previous | 1 | 2 | Next »

0 / 8 requests | 0 B / 9.8 KB transferred | Finish: 43.18 s | DOMContentLoaded: 824 ms | ...

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