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 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 at the service layer.
4. The data is looked up using JDBC / PL/SQL stored procedure call. The json data is handed over to the client upon successful lookup. The browser parses the data and displays in table. The pagination feature is incorporate so that large data is split into multiple pages and user is given a handle to browse the pages. The sorting feature is also added enabling the user to sort entire data set in ascending or descending pattern and displays according to pagination design.

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) **Two options to save the data; (a) without Oracle RDBMS system (serialize the data object to the hard disk on server side). Or (b) save it into Oracle database.**

      1. (a). **Serialize the data object to disk:** since programing to interface is followed, it is simple to swap data saving between oracle RDMS and disk at server. The xml data is initialized as 'order' object instance. This object is serialized and written to disk.
         The buz logic is implemented in '`OrderSerializationHelper.java`'.
         `@Repository("orderDao"):` Uncomment the line enabling bean creation for the above java file.
         At the same time comment out this same line in '`OrderDBHelper.java`'
      1. (b). **Oracle RDBMS:** `Swap to Oracle mode by reversing as mentioned above step, 1(a).`
         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: change accordingly in the file, 'application.properties'**

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

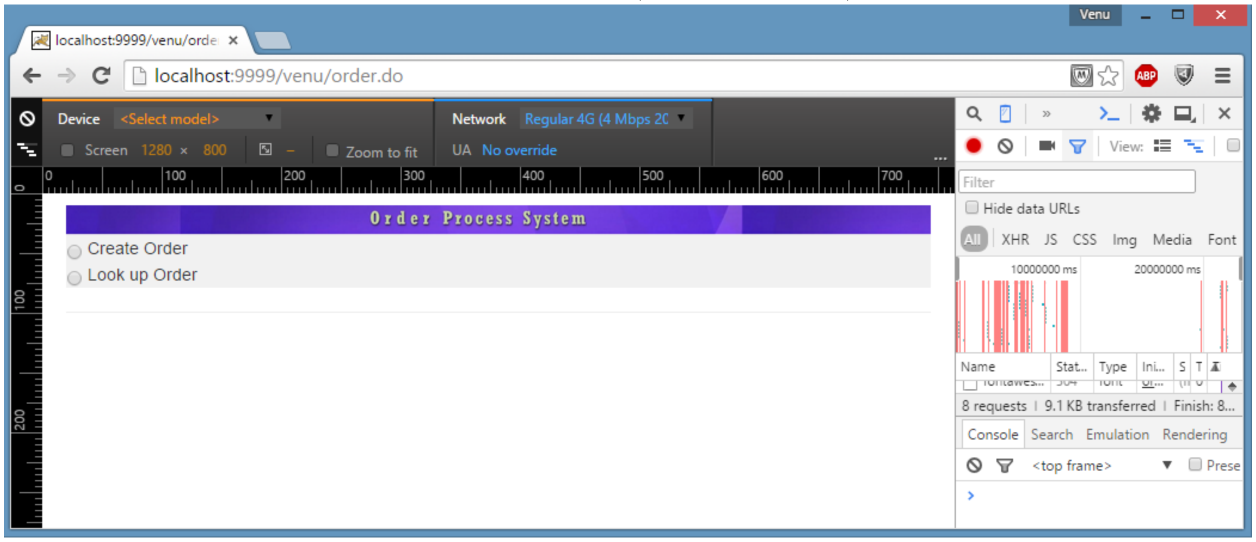
3. Sample xml data file loaded in the system:

```xml
<?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" 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>
```

4. Schema(order.xsd)

```xml
<?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)

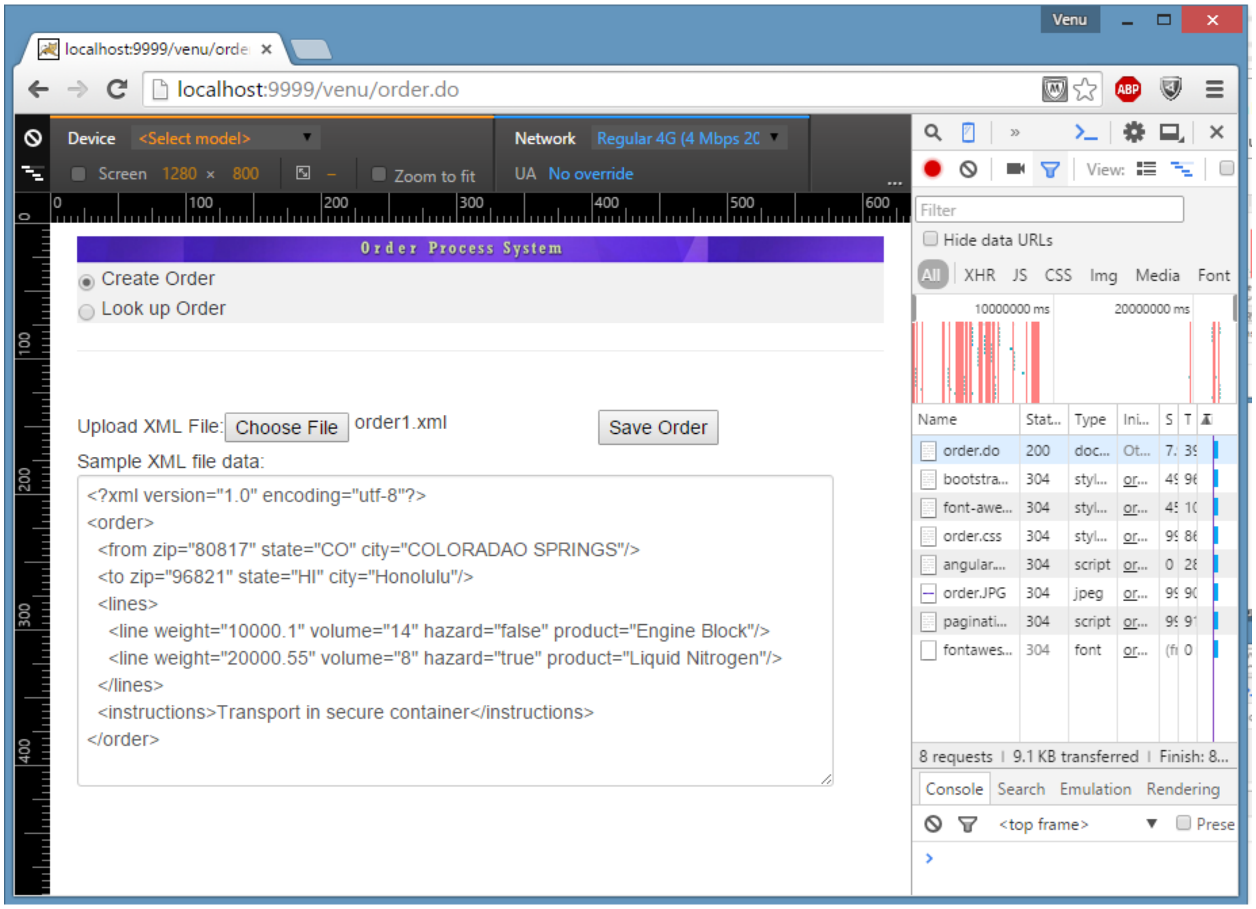

localhost:9999/venu/order.do
Device <Select model>
Screen 1280 × 800
Zoom to fit
Network Regular 4G (4 Mbps 20
UA No override
Filter
Hide data URLs
All XHR JS CSS Img Media Font
8 requests | 9.1 KB transferred | Finish: 8...
Console Search Emulation Rendering
<top frame>

Order Process System

Create Order

Look up Order

Screen:2 (Uploading xml file; 'order' data)


localhost:9999/venu/order.do
Device <Select model>
Screen 1280 × 800
Zoom to fit
Network Regular 4G (4 Mbps 20
UA No override
Filter
Hide data URLs
All XHR JS CSS Img Media Font
Choose File
Save Order
Console Search Emulation Rendering
<top frame>

Order Process System

[✓ Create Order]

Look up Order

Upload XML File: Choose File order1.xml Save Order

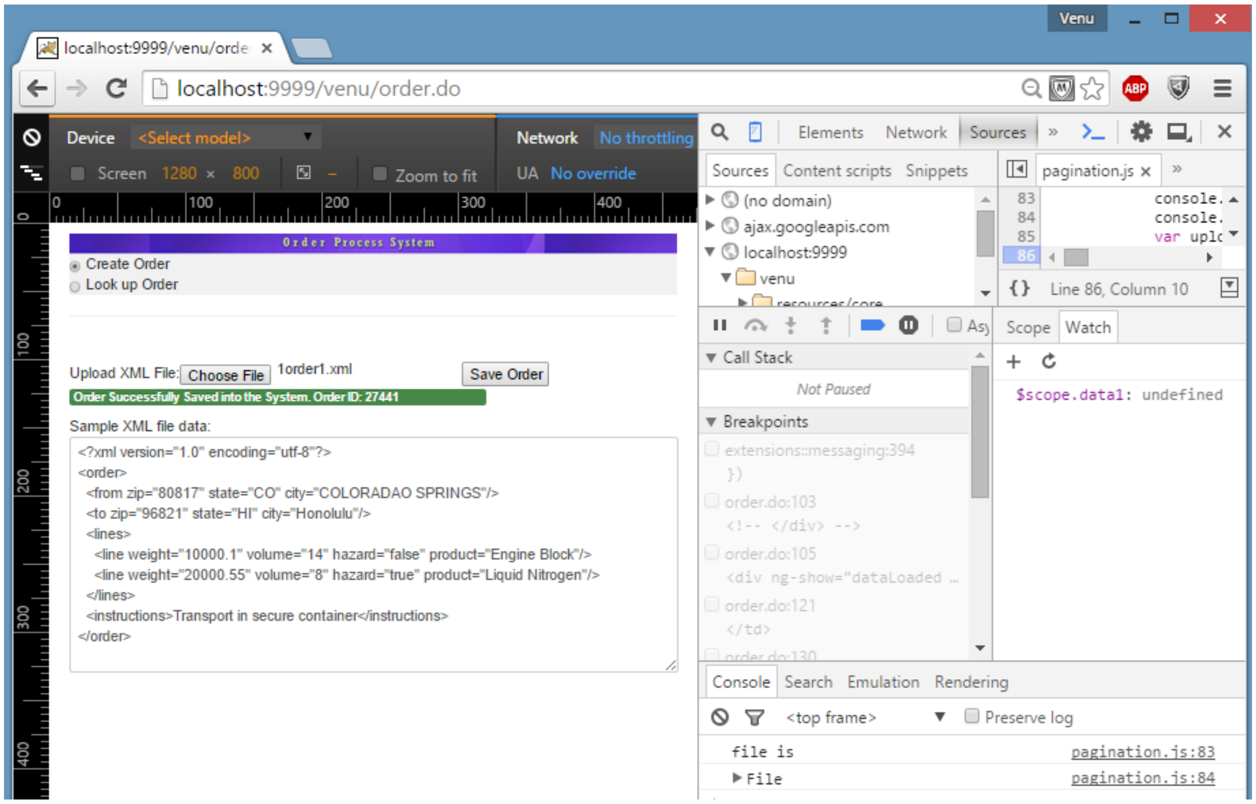
Sample XML file data:

```
<?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" hazard="false" product="Engine Block"/>
    <line weight="20000.55" volume="8" hazard="true" product="Liquid Nitrogen"/>
  </lines>
  <instructions>Transport in secure container</instructions>
</order>
```

Name	Stat...	Type	Ini...
order.do	200	doc...	Ot...
bootstra...	304	styl...	or...
font-awe...	304	styl...	or...
order.css	304	styl...	or...
angular...	304	script	or...
order.JPG	304	jpeg	or...
paginati...	304	script	or...
fontawes...	304	font	or...

8 requests | 9.1 KB transferred | Finish: 8...

Screen: 3 (File Upload completed, saved to DB)



Order Process System

Create Order
Look up Order

Upload XML File: Choose File 1order1.xml Save Order

Order Successfully Saved into the System. Order ID: 27441

Sample XML file data:

```
<?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" hazard="false" product="Engine Block"/>
    <line weight="20000.55" volume="8" hazard="true" product="Liquid Nitrogen"/>
  </lines>
  <instructions>Transport in secure container</instructions>
</order>
```

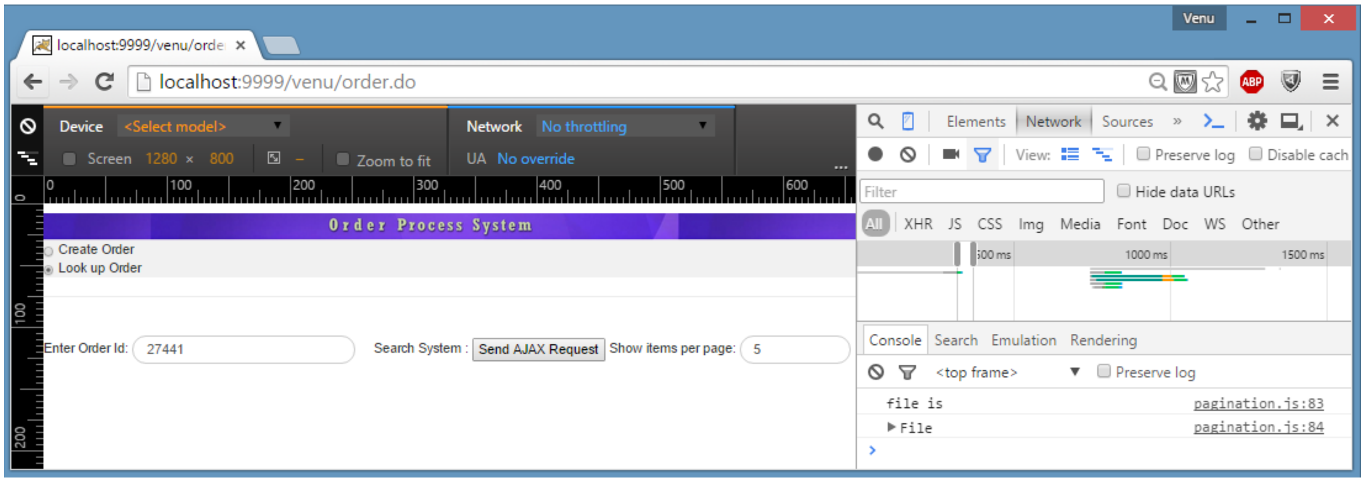
```
extensions::messaging:394
})
order.do:103
<!-- </div> -->
order.do:105
<div ng-show="dataLoaded …
order.do:121
</td>
order.do:130
```

```
$scope.data1: undefined
```

```
file is                         pagination.js:83
▶File                           pagination.js:84
```

Screen: 4.i. (search order id: initial screen)

Initial search screen loaded. Enter Order ID.



Order Process System

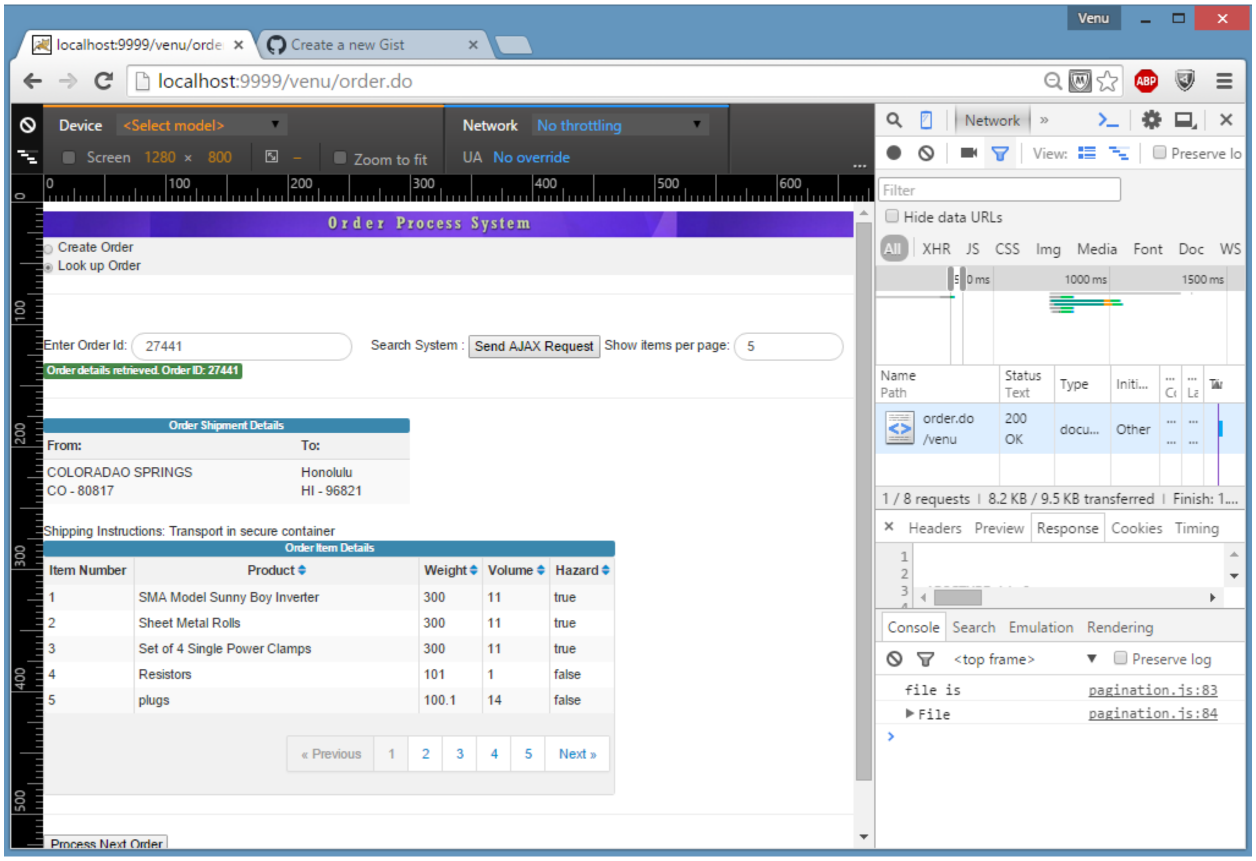
Create Order
Look up Order

Enter Order Id: 27441 Search System : Send AJAX Request Show items per page: 5

```
file is                         pagination.js:83
▶File                           pagination.js:84
>
```

Screen: 4.ii. (results of search: Tabular display)

Results are displayed in a table structure. Data in the columns are sortable (toggle reverse option). Default page size is ‘5’ and pagination feature is also incorporated.



Order Process System

Create Order
Look up Order

Enter Order Id: 27441 Search System : Send AJAX Request Show items per page: 5

Order details retrieved. Order ID: 27441

Order Shipment Details

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

Shipping Instructions: Transport in secure container

Order Item Details

Item Number	Product	Weight	Volume	Hazard
1	SMA Model Sunny Boy Inverter	300	11	true
2	Sheet Metal Rolls	300	11	true
3	Set of 4 Single Power Clamps	300	11	true
4	Resistors	101	1	false
5	plugs	100.1	14	false

« Previous 1 2 3 4 5 Next »

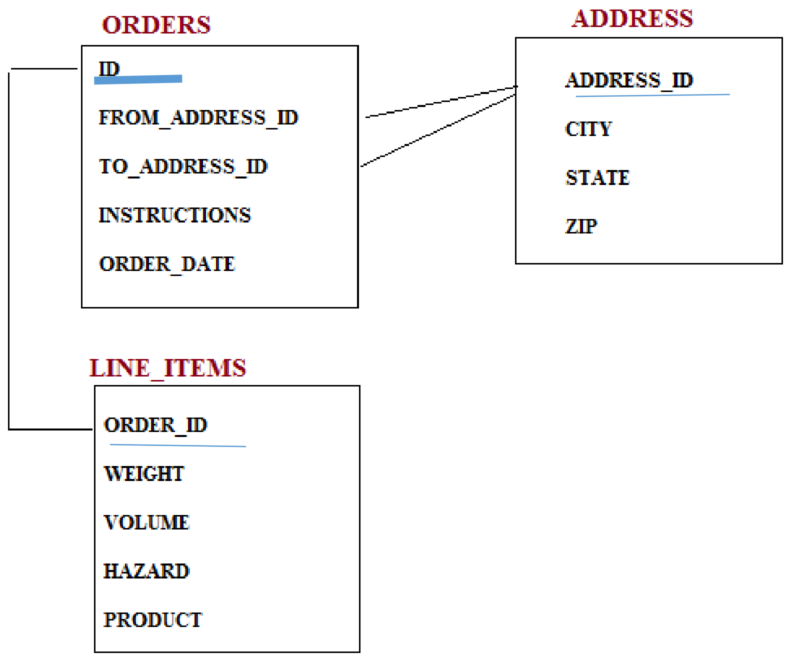
Process Next Order

Name Path	Status Text	Type	Initi...
order.do /venu	200 OK	docu...	Other

1 / 8 requests | 8.2 KB / 9.5 KB transferred | Finish: 1....

```
file is                                   pagination.js:83
▶ File                                    pagination.js:84
```

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