**User Manual on**

**Visualization using**

**NVD3 Charts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Revision and Approval** **History**  **Title:** User Manual on NVD3 charts  **ID :**  **Effective:** 06-Jun-2014 | | | | |
| Edition No. | Release Date | Author’s Name | Software Version | Reason for Change / Description of Changes |
| 1 | 05/10/2014 | Harish |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**About:** In this tutorial you will get an hands on understanding on how you can create dynamic visualization using NVD3. Initially you will execute existing charts to get a high level understanding. As the next step, you will actually create the charts in a step by step manner. This will include taking data[JSON] from a Big Data system and creating a chart.

**Table of Contents**

1. End-End IoT Flow

2. NVD3 chart flow Diagram

3. Objective

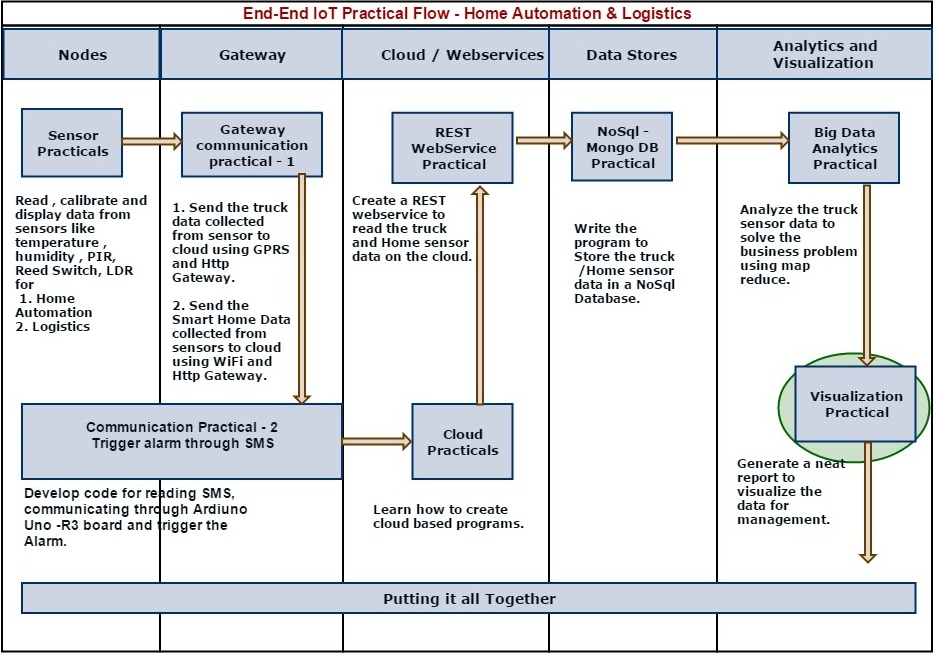
4. Pre Requisites

5. Executing Existing Charts

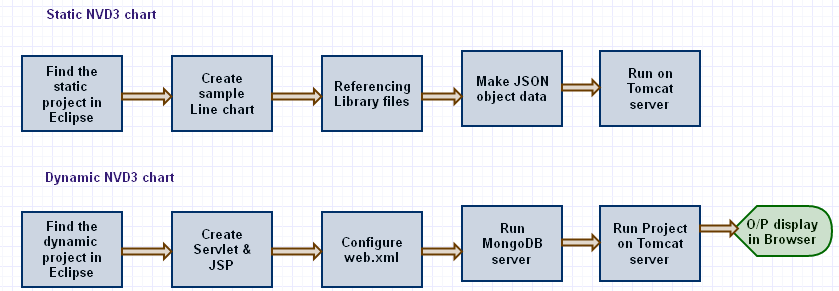
6. Create Sample Line Chart – NVD3

7. Create Dynamic Stacked Multi Bar Chart (mongoDB - MapReduce)

1. **Complete End-End IoT Flow diagram:**

****

**2) NVD3 chart flow Diagram:**

****

**3) Objective:**

In this user manual we are going to learn

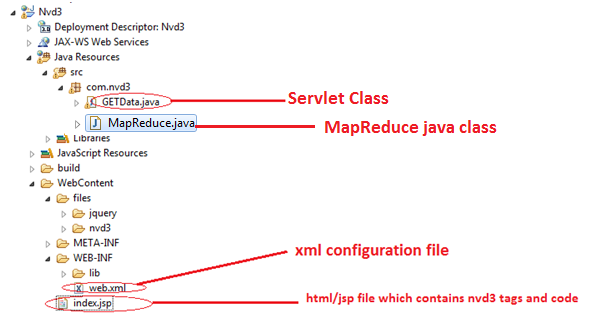
1. Create a Sample chart using static data
2. Connect with MongoDB and get information from collections
3. Create a Dynamic Stacked Multi Bar Chart of number of trips of material fishes getting spoiled in each week using the map reduce from MongoDB

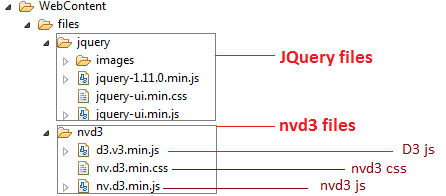
**4) Pre Requisites:**

1. D3 JavaScript library.
2. NVD3 JavaScript library.
3. JQuery JavaScript library.
4. MongoDB.
5. mongo-java-driver-2.10.1.jar
6. json-20090211.jar

**5) Executing Existing Charts – NVD3:**

1. Open Eclipse.
2. Find the Project **Nvd3,** and open it.
3. Project Structure



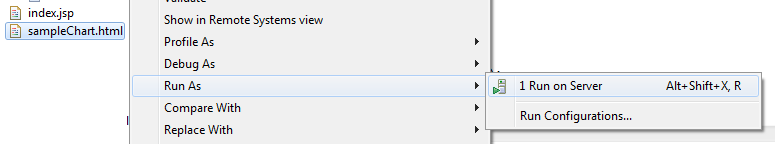




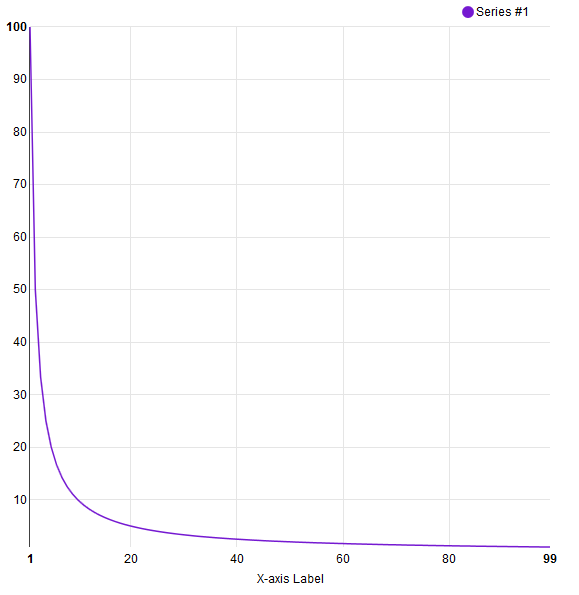
1. Execute sampleChart.html (Line Chart)

Find sampleChart.html in **Nvd3🡪WebContent**, and Open it.

Right Click on **sampleChart.html 🡪 Run As 🡪 Run On Server**



Then you should see output as following figure:

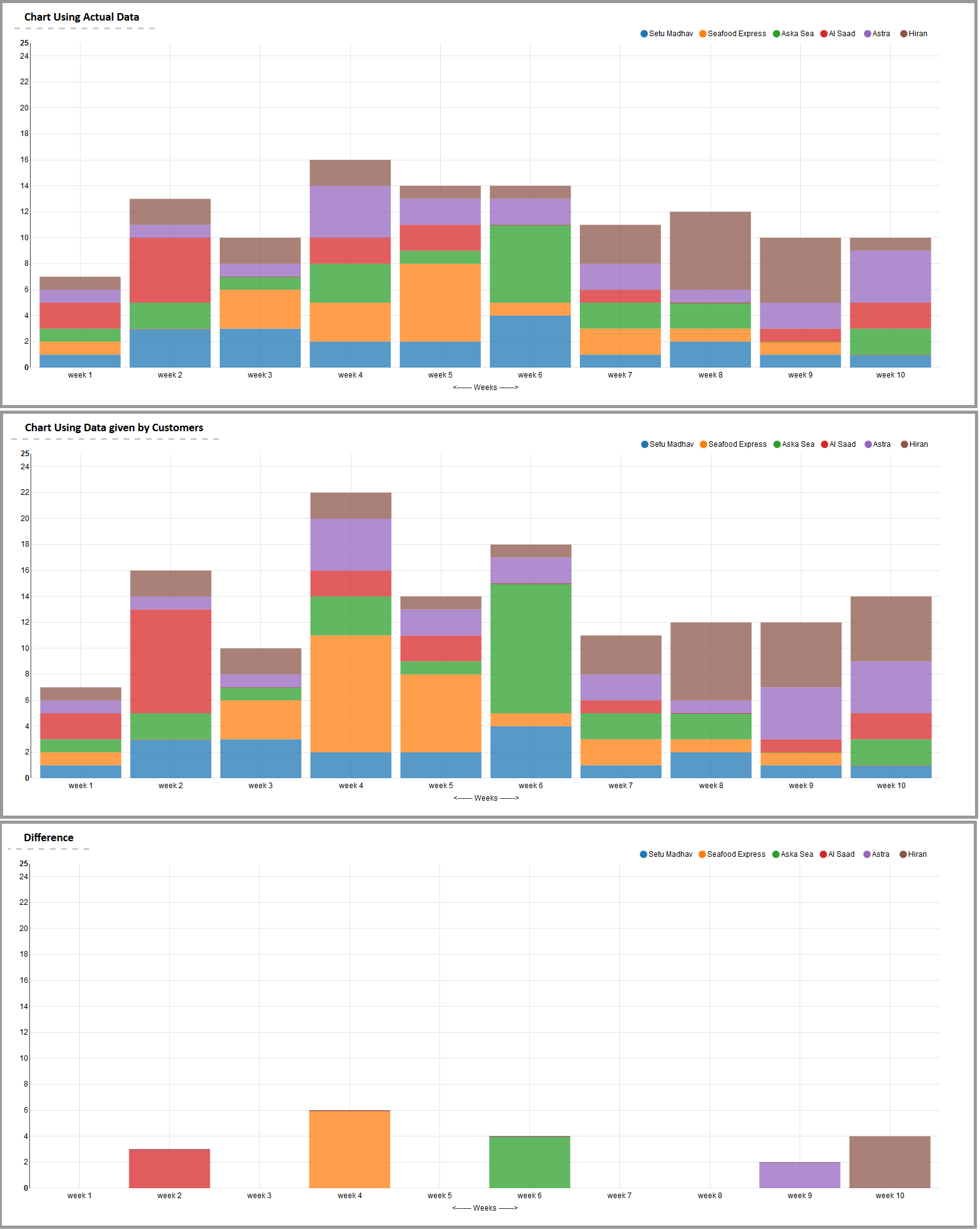
****

1. Execute Dynamic Chart **index.jsp** (Multibar Char)

Find index.jsp in **Nvd3🡪WebContent**, and Open it.

Right Click on **sampleChart.html 🡪 Run As 🡪 Run On Server.**

Then you should see the chart as below figure:

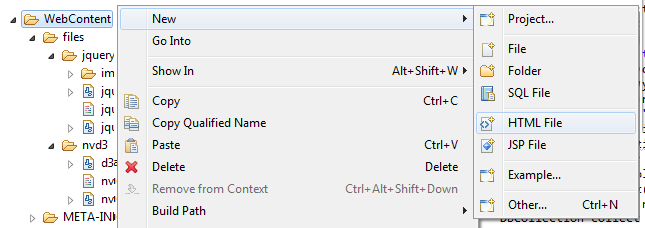


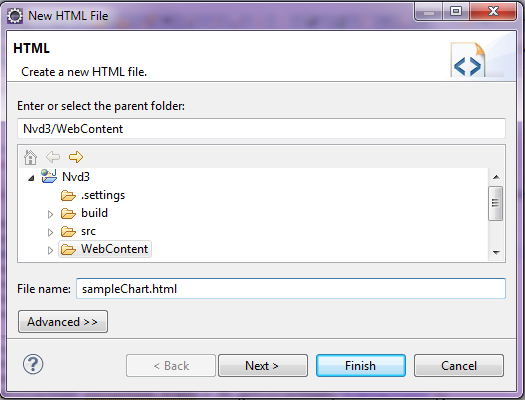
**Detailed Steps for Creating These Charts Above**

**6) Create Sample Line Chart – NVD3:**

* Create a new html file in **WebContent** Folder named **sampleChart.html**

Right Click WebContent🡪New🡪HTML file





* Referencing javascript(d3.v3.js , nv.d3.js) and css(nv.d3.css)

Write the following lines into HTML at the beginning of <head> tag

<link rel=*"stylesheet"* href=*"files/nvd3/nv.d3.min.css"*>

<script type=*"text/javascript"* src=*"files/nvd3/d3.v3.min.js"*></script>

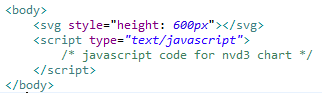
<script type=*"text/javascript"* src=*"files/nvd3/nv.d3.min.js"*></script>

* Configure svg tag for chart

Insert <svg style="height: *600px*"></svg> at the beginning in the <body> of HTML file, which will contain the chart in sampleChart.html (shown below). This is needed for telling the browser that we will be using “Scalable Vector Graphics” in this html file. D3 uses this for charting.

* Write Javascript tag

Insert <script type=*"text/javascript"*></script> after <svg/> element in sampleChart.html



* Code for Line Chart( y = 1/x )

Write the following JavaScript code (*which holds nvd3 chart initialization code*) into sampleChart.html file in <script/> tag of the <body/>. This code creates a simple line(y = 1/x) chart using Nvd3.js library. Code comments in the code below provide more information about this script and how it works.

/\*Data creation. In actual charts this will be coming from the server side or database. \*/

**function** myData() {

**var** series1 = [];

**for** (**var** i = 1; i< 100; i++) { /\*creating data for line of(x)=100/x \*/

series1.push({ /\* creating json Data for Series1\*/

x : i,

y : 100 / i

});

}

**return** [ { /\* return Line chart data \*/

key : "Series #1",

values : series1,

color : "#7519D1"

} ];

}

nv.addGraph(**function**() { /\*This adds the chart to a global rendering queue. \*/

**var** chart = nv.models.lineChart();/\*Create instance of nvd3 lineChart. \*/

chart.xAxis.axisLabel("X-axis Label");/\*Set X-axis attributes\*/

chart.yAxis.axisLabel("Y-axis Label")/\*Set Y-Axis attributes. \*/

.tickFormat(d3.format("d")); /\*Set Y-Axis label formatting. \*/

d3.select("svg")/\*Select the document's <svg> element. \*/

.datum(myData())/\*Attach data to the <svg> element. \*/

.transition()/\*Define transition(Optional). \*/

.duration(500)/\*define Animation speed (Optional). \*/

.call(chart);/\*pass the d3.selection to our lineChart. \*/

nv.utils.windowResize(**function**() {/\*Updates the window resize event callback. \*/

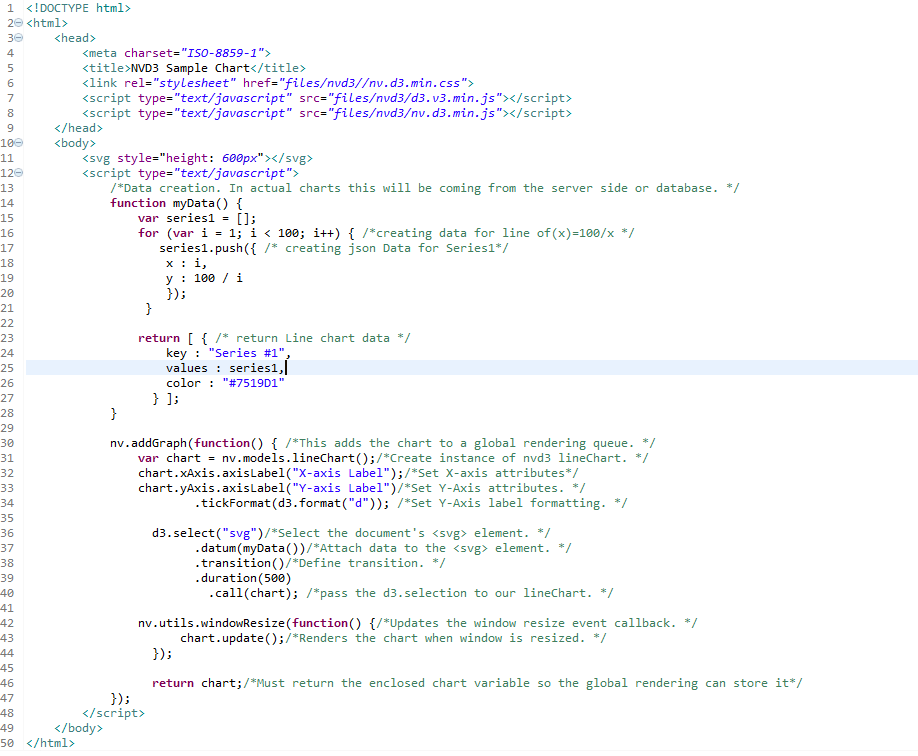
chart.update();/\*Renders the chart when window is resized. \*/

});

**return** chart;/\*Must return the enclosed chart variable so the global rendering can store it\*/

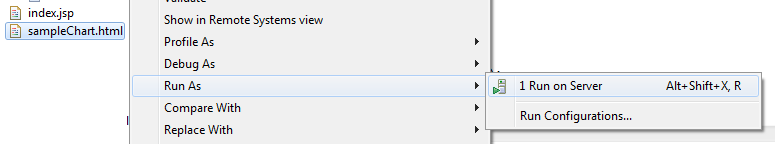
});

* Finally your sampleChart.html should be like:

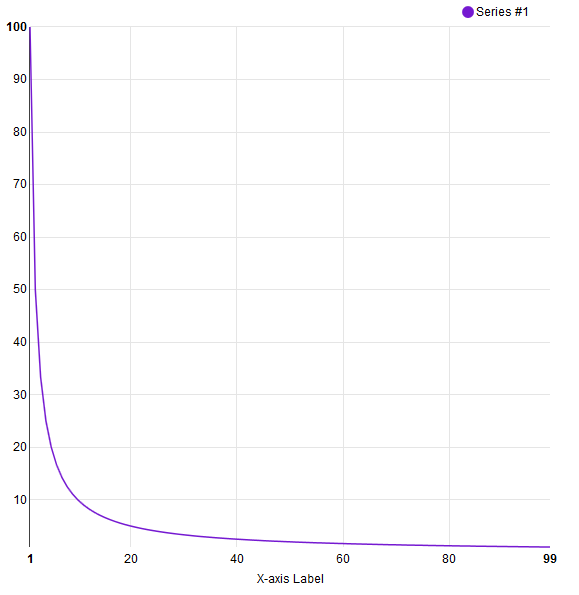


* Viewing your Chart

Right Click on sampleChart.html 🡪 Run As 🡪 Run On Server



Then you should see output as following figure:



* Play with Data / Chart**:** You can change the chart by modifying the data generation. For example, instead of 100/x in the myData () function of JavaScript, you can use x = i2and see its impact on the chart. You can try some other ways as well. Just need to change the code and do a refresh of the browser.

**7) Create Dynamic Stacked MultiBar Chart** (mongoDB - MapReduce)**:**

**Business scenario:**

Best cold chain company is facing huge claims for delivery of spoilt Fish.

Axelta has installed devices on vehicles, these devices sends the data to the Axleta Platform. Write map-reduce to analyze fault claims and display in chart.

For above business scenario, first we need to get actual data from mongodb. Then we need to compare the actual data with customers provided data.

* Open MapReduce.java.

Find the MapReduce.java in src 🡪 com.nvd3 (package) and open it.



* Write following code into MapReduce.java.

MapReduce.java contains code to get actual data from MongoDB using java MapReduce concept.

**package** com.nvd3;

**import** java.net.UnknownHostException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** com.mongodb.DB;

**import** com.mongodb.DBCollection;

**import** com.mongodb.MapReduceOutput;

**import** com.mongodb.Mongo;

**import** com.mongodb.MapReduceCommand.OutputType;

@SuppressWarnings("rawtypes")

**public** **class** MapReduce {

**public** List mapReduce() **throws** UnknownHostException{

List data = **new** ArrayList();

Mongo mongo = **new** Mongo("", 27017);/\*Connecting to local MongoDB server\*/

DB db = mongo.getDB("bcc");/\*Connecting to Database 'bcc'\*/

DBCollection collection = db.getCollection("trip\_trans");/\*select Collection(table) 'trip\_trans' of 'workshop'\*/

String mapFun = **null**, reduceFun = **null**;

/\*Map Function\*/

mapFun = "function(){"

+"var timestamp = this.timestamp\*1000;"

+"var min\_temp, max\_temp, temp;"

+"temp = this.temperature;"

+"max\_temp = this.maximum\_temperature;"

+"min\_temp = this.minimum\_temperature;"

+"if(this.material == 'Fish' && (temp < min\_temp || temp > max\_temp))"

+"{"

+"emit(this.trip\_id, this);"

+"}"

+"}";

/\*Reduce Function\*/

reduceFun = "function(k, v){"

+"var json1 = {};"

+"var counter = 0;"

+"var previousTimestamp;"

+"var getWeek = function(d){"

+"d.setHours(0,0,0);"

+"d.setDate(d.getDate()+4-(d.getDay()||7));"

+"return Math.ceil((((d-new Date(d.getFullYear(),0,1))/8.64e7)+1)/7);"

+"};"

+"for(var i=0; i < v.length; i++){"

+"if (counter == 0) {"

+"previousTimestamp = v[i].timestamp;"

+"counter++;"

+"continue;"

+"}"

+"if (counter = 3){"

+"json1.customer = v[i].customer;"

+"json1.weekNo = getWeek(new Date(v[i].timestamp\*1000));"

+"break;"

+"}"

+"if ((v[i].timestamp - previousTimestamp)/60 < 16) {"

+"counter++;"

+"previousTimestamp = v[i].timestamp;"

+"} else {"

+"counter = 0;"

+"previousTimestamp = v[i].timestamp;"

+"}"

+"}"

+"if (counter == 3){"

+"return json1;"

+"}"

+"return null;"

+"}";

/\*Execute Map-Reduce Functions \*/

MapReduceOutput mapReduceOutput = collection.mapReduce(mapFun, reduceFun, **null**, OutputType.*INLINE*, **null**);

data = (List) mapReduceOutput.getCommandResult().get("results");/\*get map reduce output main result and assign to data \*/

**return** data;

}

}

* **Configure** web.xml with servlet url patterns

Here we are mapping GETData class with url-pattern /getData.

GETData.java is a servlet Class file. It contains basic servlet doGet() method. It returns json by accessing MapReduce Class.

Write below code in web.xml between <web-app> tags.

<servlet>

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

<servlet-class>com.nvd3.GETData</servlet-class>

</servlet>

<servlet-mapping>

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

<url-pattern>/getData</url-pattern>

</servlet-mapping>

* index.jsp contains multiBarChart javascript and html code.
* JSON Format for Group/Stacked Bar Chart

[

{

key: "<Series\_Name1>",

values: [

{ x : 1 ,y : 33.66} ,

{ x : 2 ,y : 10} ,

{ x : 3 ,y : 35.15} ,

{ x : 4 ,y : 100.55},

...

...

]

},

{

key: "<Series\_Name2>",

values: [

{ x : 1 ,y : 38.78} ,

{ x : 2 ,y : 100} ,

{ x : 3 ,y : 3.1} ,

{ x : 4 ,y : 10.65},

...

...

]

}

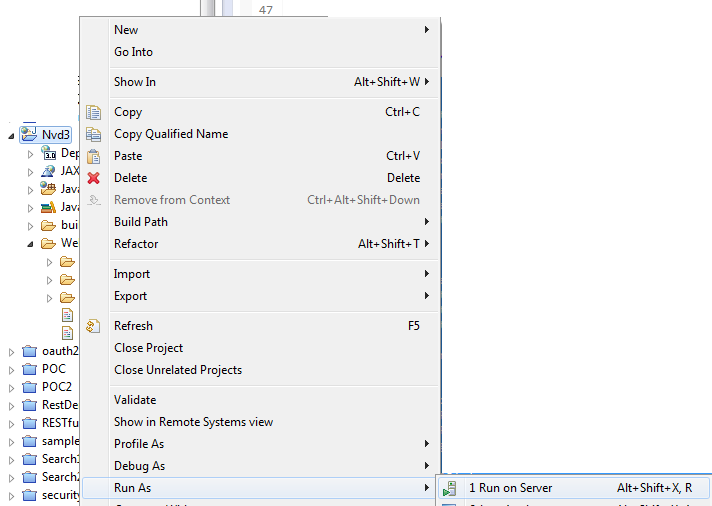
.....

.....

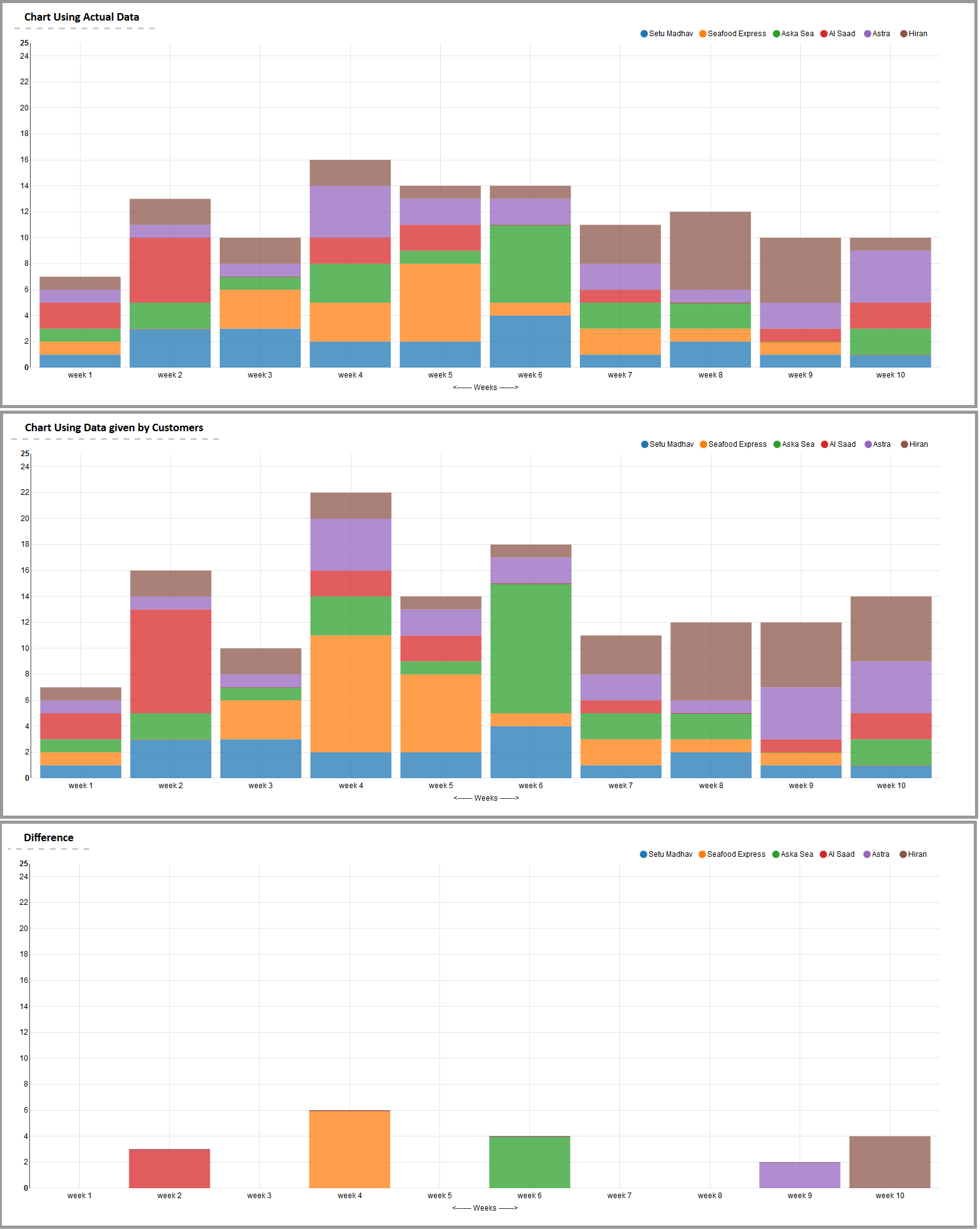
]

* Run MongoDB Server
* Viewing Your Chart

Right Click on Nvd3, click Run as🡪Run On Server🡪Select TomCat Server🡪Finish



Then you should see output as following figure:



References:

<http://nvd3.org/>

<http://d3js.org/>

<https://github.com/novus/nvd3/wiki/API-Documentation> ⃝