SAP Mobility 101

Tutorial 12 – Mock data Server

# Objective of Exercise

## Build an example application

The objective of this exercise is to build a Mock Data Server to use in the development stage of this project.

## Note

* We recommend that you use a chrome browser for testing
* Eclipse Juno would be needed for this Tutorial.

# Task 1: Include the Mock server library:

Include the following library in your code, do this in you index.html file:

jQuery.sap.require(“sap.ui.core.util.MockServer”);

# Task 2: Create a metadata.xml file

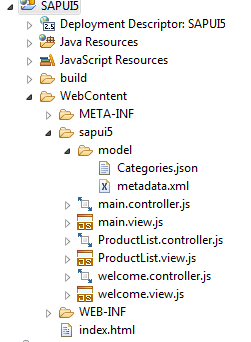
The following files have been given in the folder named model:

Categories.json

metadata.xml

Copy the model directory to the WebContent>sapui5 directory of your current project.

Your project file system should look like this:



# Task 2: Select connectivity setting

We want to have the option of using the original Northwind data source or the mock data, thus create an if statement that would allow this. Set the setting to using the mock data for this tutorial, change you onInit function in main.controller to the following:

**var** mock = **true**;

**if**(mock == **true**){

/\*mock server\*/

}

**else**{

Uri = "proxy/http/services.odata.org/V2/Northwind/Northwind.svc/";

}

**var** oModel = **new** sap.ui.model.odata.v2.ODataModel(

Uri,

{

json: **true**

}

);

sap.ui.getCore().setModel(oModel, 'data1');

# Task 3: Create the mock server

Set the Uri to a mock server location:

Uri ="proxy/http/mymockserver/";

Create the mock server with the Uri selected above:

**var** Uri ="proxy/http/mymockserver/";

**var** oMock = **new** sap.ui.core.util.MockServer({

rootUri: Uri

});

Create a variable with the path to the metadata file, and a variable with path to the directory of the file containing the JSON mock data:

**var** metadataUrl = "sapui5/model/metadata.xml";

**var** mockdatabase = "sapui5/model/";

Simulate the MockServer with the metadata URL created above. Set the base Url to the ‘model’ file.

If there are no data for the application, the Mock server can generate false data for testing purposes, the setting that allows this is the bGenerateMissingMockData boolean setting, in this case set this setting to false.

Start the Mock server.

oMock.simulate(metadataUrl,{

'sMockdataBaseUrl': mockdatabase,

'bGenerateMissingMockData': **false**

});

oMock.start();

## The code after this task should look like this;

onInit: **function**() {

**var** mock = **true**;

**if**(mock == **true**){

**var** Uri ="proxy/http/mymockserver/";

**var** oMock = **new** sap.ui.core.util.MockServer({

rootUri: Uri

});

**var** metadataUrl = "sapui5/model/metadata.xml";

**var** mockdatabase = "sapui5/model/";

oMock.simulate(metadataUrl, {

'sMockdataBaseUrl': mockdatabase,

'bGenerateMissingMockData': **false**

});

oMock.start();

}

**else**{

Uri = "proxy/http/services.odata.org/V2/Northwind/Northwind.svc/";

}

**var** oModel = **new** sap.ui.model.odata.v2.ODataModel(Uri, {json: **true**});

sap.ui.getCore().setModel(oModel, 'data1');

},

# Task 4: How to create the metadata file(optional)

Copy the address of the data source into your browser.

Add an extension /$metadata to the address.

The address would look like this:

<http://services.odata.org/v2/Northwind/Northwind.svc/$metadata>

A page in xml formal would apear:



Copy all the contents of this page into a text file or any other xml editing program:

In this case I used Sublime text. (a setup folder for sublime text are included in the tutorial package, see “Sublime Text Build 3083 x64 Setup”, Sublime text can also be downloaded at the following link: <http://www.sublimetext.com/3> )

Remove the first lines that are not part of the xml document:

Eg. This XML file does not appear to have any style information associated with it. The document tree is shown below.

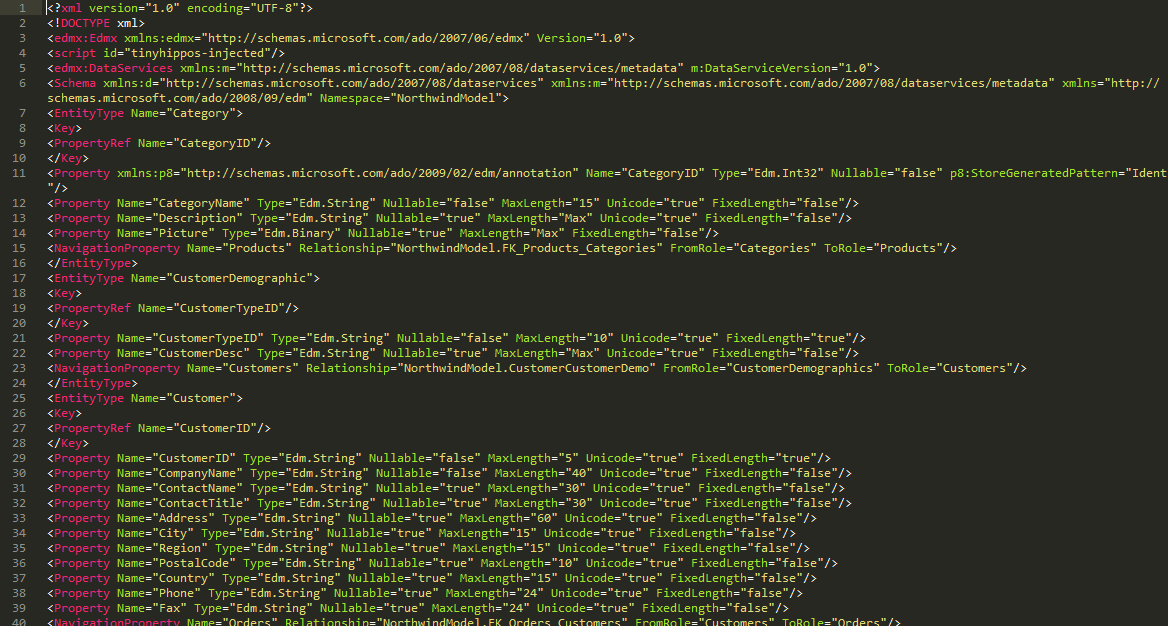
The first line should now start with <edmx:

Add the following two lines to the top of the page:

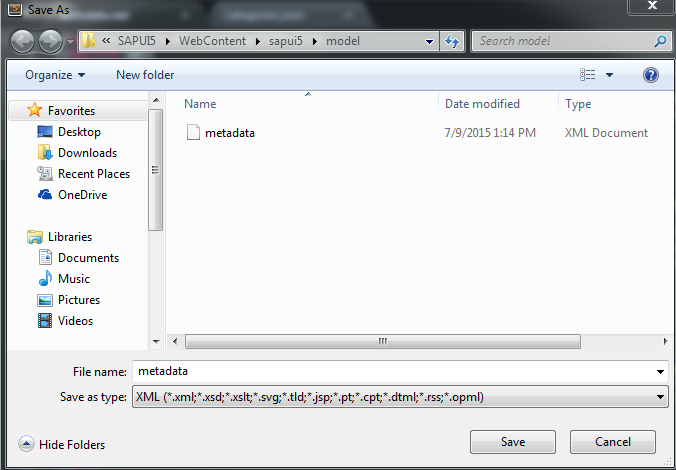
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE xml>

A Sublime text file would now look like this:



Select File>Save As



Select the “Save as type:” as XML, name the file “metadata.xml” under the model/ directory. In this case you already have the metadata file so you don’t have to do this task, this is only for information.

# Task 4: Create a Products.json file

Enter the following address into you browser:

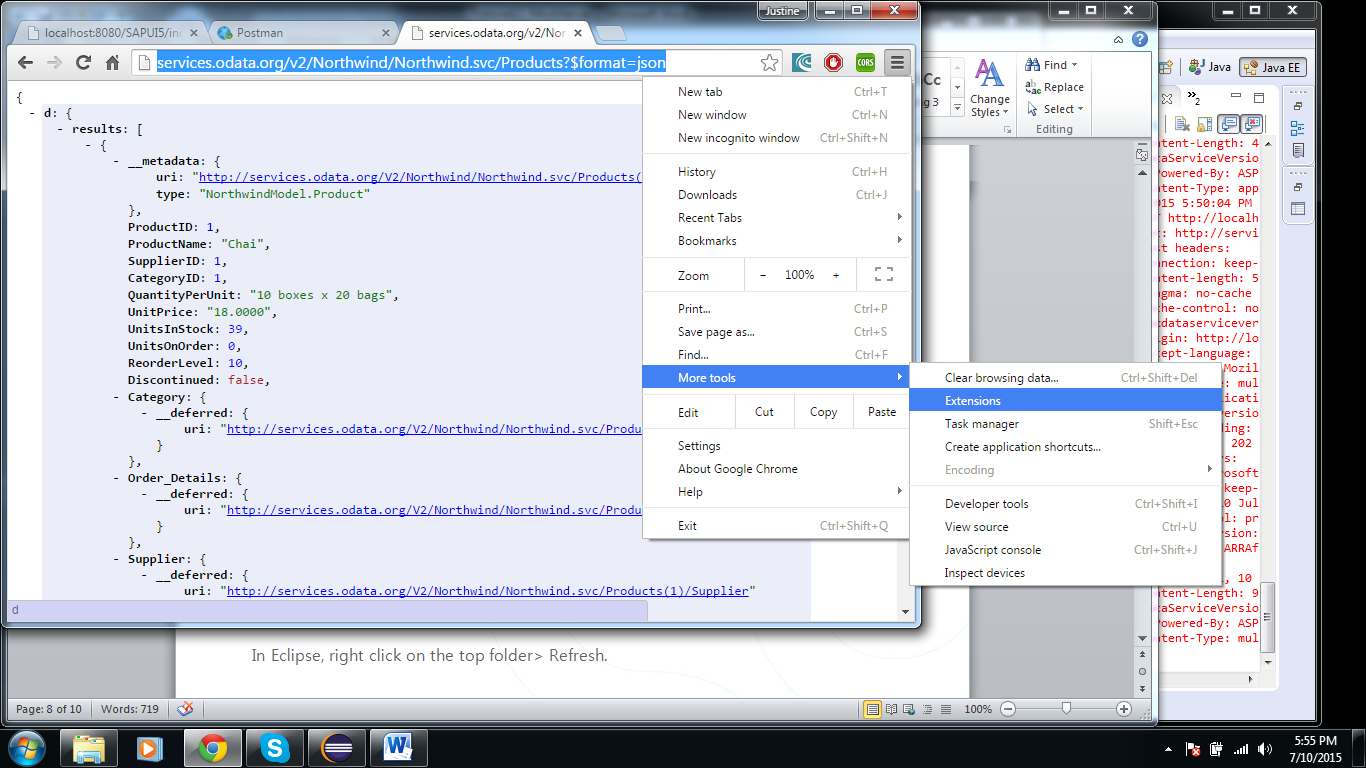
<http://services.odata.org/v2/Northwind/Northwind.svc/Products?$format=json>

The “?$format=json” extension indicates that we want the products table in JSON format.

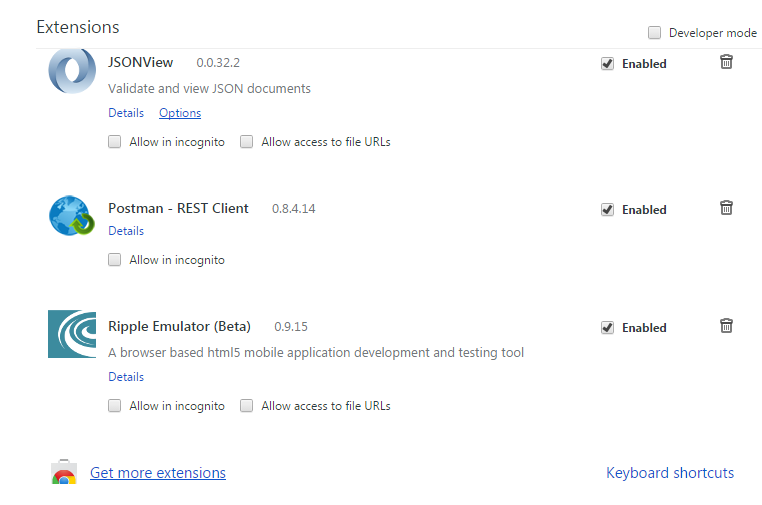


This is not quite the format we are looking for, we want the column headers to be in quotation marks eg. “ProductID”, thus you can either put all the columns in quotes, or you can follow these steps:

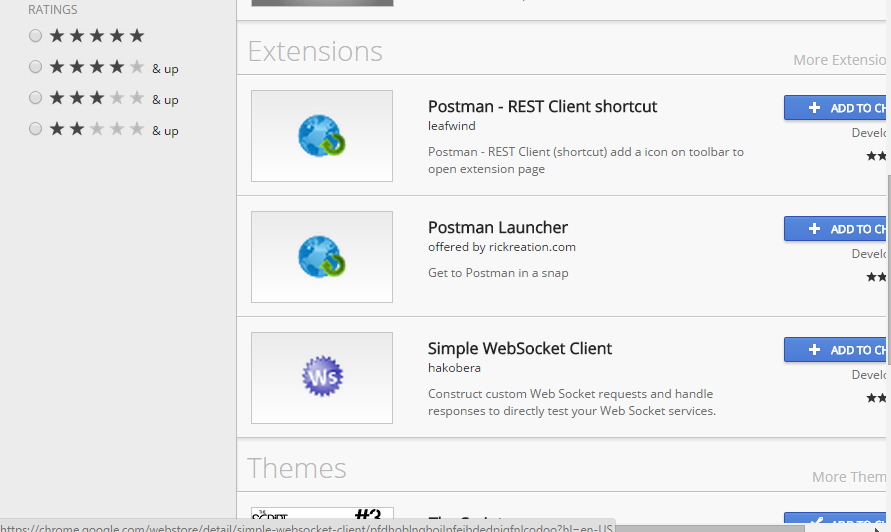
Press the options link in your Chrome browser>More Tools>Extensions



A list of extensions that you have already installed will be displayed. At the bottom of the list, click the link, “Get more extensions”.

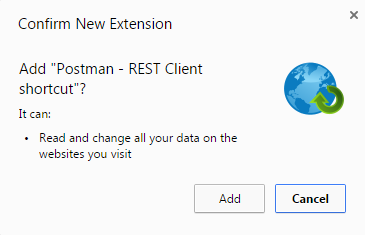


In the right hand corner, type “postman rest client”, in the search field.

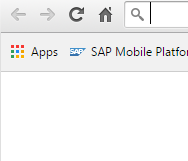


Select the postman rest client extension; select the ‘add to chart’ button next to the extension.

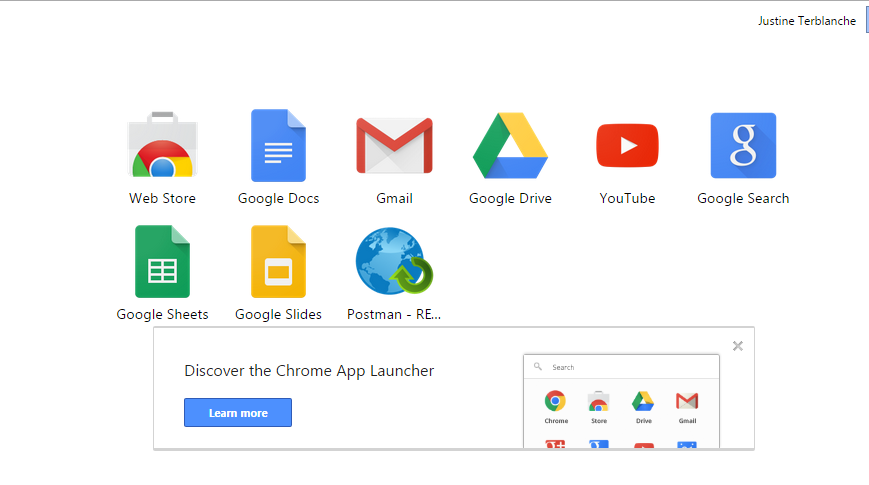
A screen asking if you would like to add the extension would appear; select add.



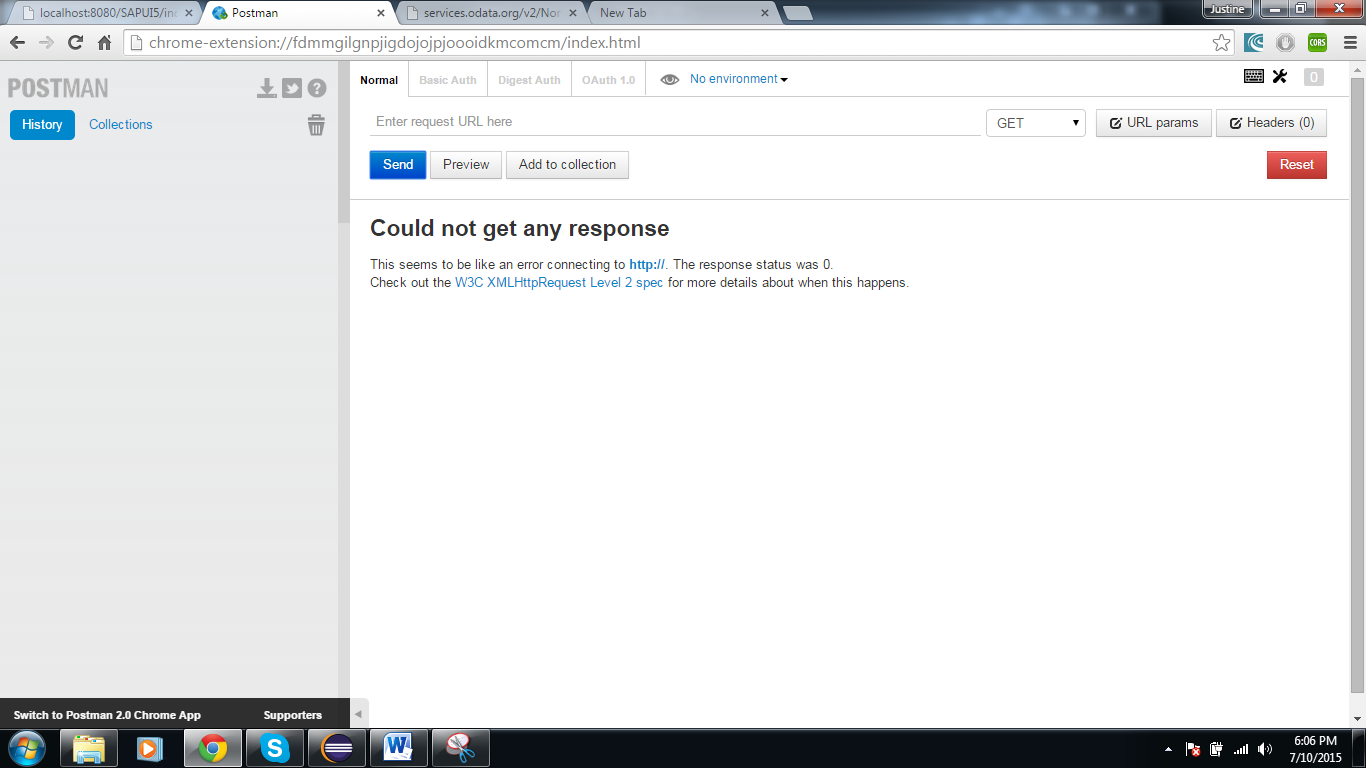
When you open your browser the “Apps” link is in the shortcut bar of your browser:



Select the “Apps” link.

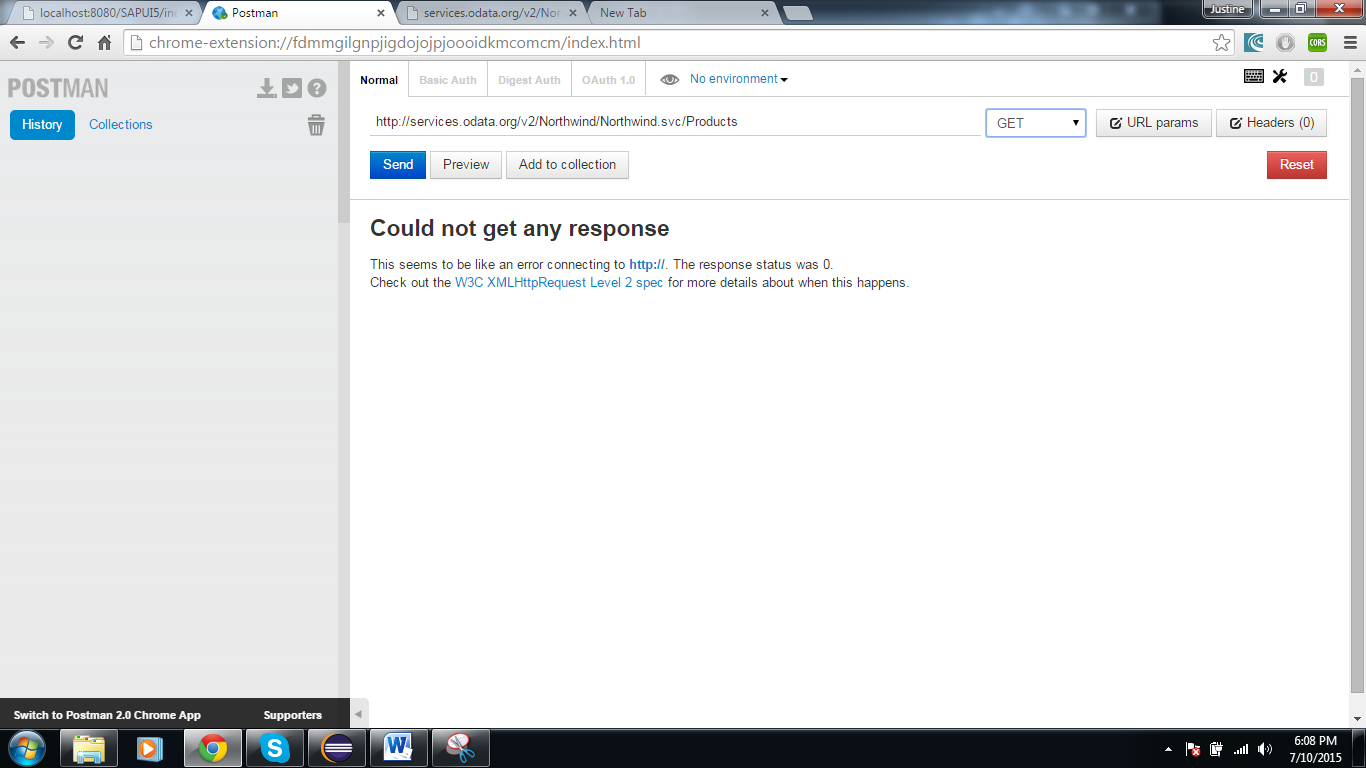


As you can see, Postman – rest client are amongst the extensions in you chrome browser, select the postman application.



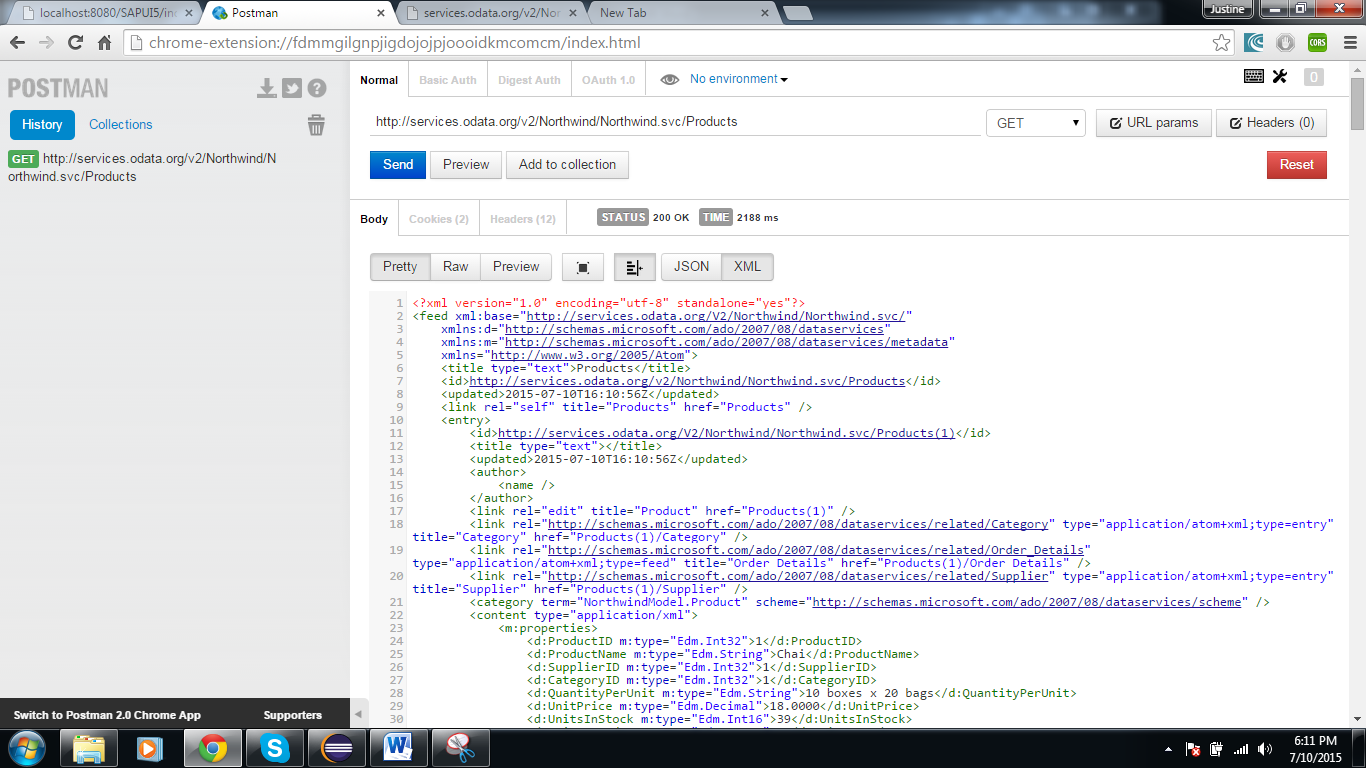
Type the following address into the URL request: <http://services.odata.org/v2/Northwind/Northwind.svc/Products>

select the GET option from the dropdown menu:



Click Send;

The products table will appear in the following format:



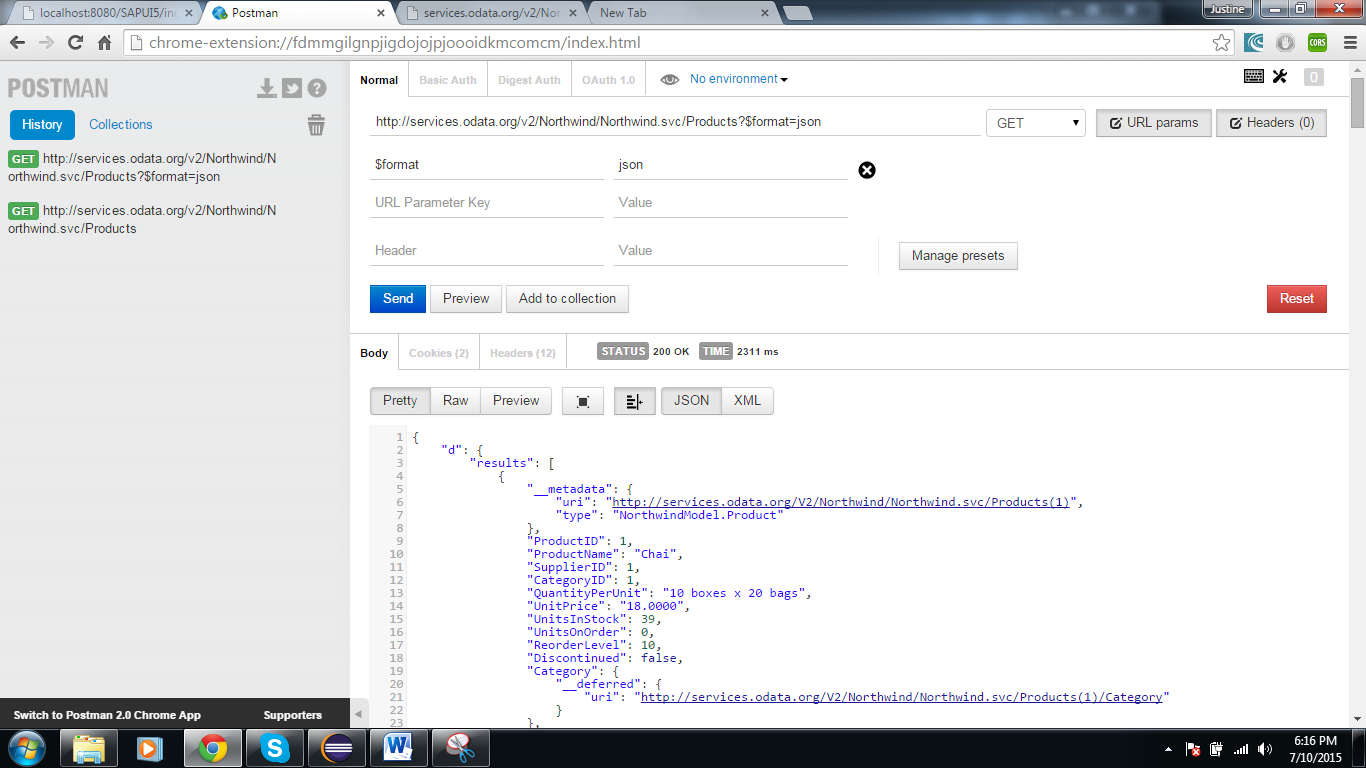
On the right-hand top corner, press ‘URL params’, enter the following into the text areas that appear: URL parameter key: $format

Value: json



As shown in the figure above, the extensions to your URL entry are added accordingly.

Click send.

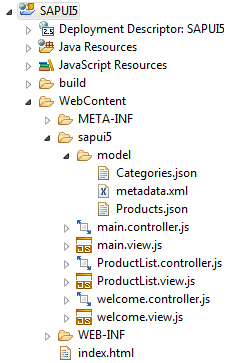


All the column headers are in Quotation marks. Copy all the contents of the file into a text or sublime document.

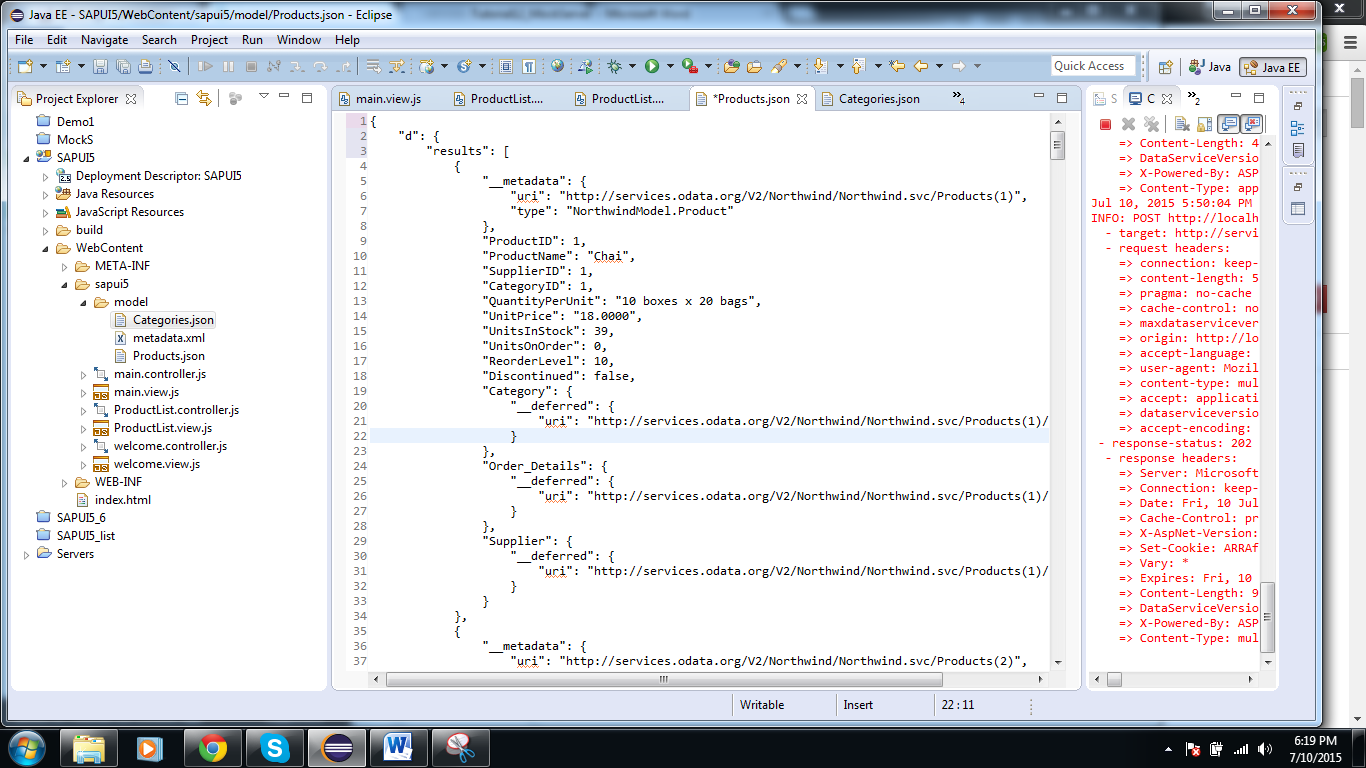
In Sublime text: Select File>Save as

Set the type of file to \*JSON and name it Products.json.

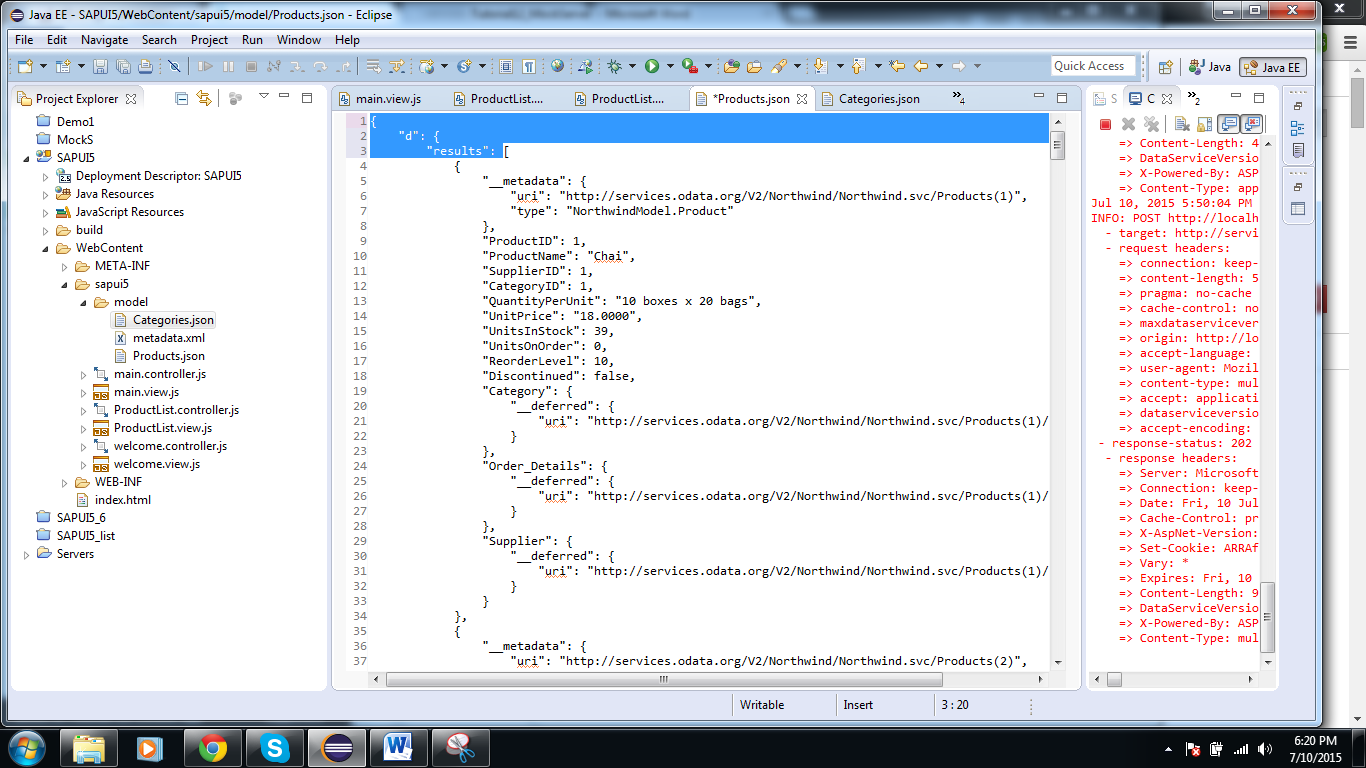
In Eclipse, right click on the top folder> Refresh.

The new file Products.json will appear in the directory. 

Double click on the products.json file.



Delete all the lines before the square bracket ‘[‘ at the beginning of the file.



Delete the lines after the square bracket ‘]’ at the end of the file.

Save the changes.

The file should now be able to load when the mock data are called.

The metadata file indicates that there are more than just these two .json files, but we don’t need those files for this tutorial, thus the console will show some errors when the program is running, these errors can be ignored. In this example, the file for the Shipper table cannot be found:



# Task 5: Use mock data in the Products list

We can use the data model created for the products list as well.

Change the data reference in ProductList.view from data2 to data1:

**var** otemplate = **new** sap.m.StandardListItem({

id: "Plist",

title: "{data1>ProductName}",

description:"{data1>CategoryID}"

});

**var** oFilters = [

**new** sap.ui.model.Filter(

"CategoryID", "EQ", "2"

)];

list.bindAggregation("items", {

path: "data1>/Products",

filters: oFilters,

template: otemplate});

**var** tex = **new** sap.m.Text({text: "{selected>/CategoryID}"});

Comment out the data model data2 in the ProductList.controller file.

onInit: **function**() {

/\*var oModel = new sap.ui.model.odata.v2.ODataModel(

"proxy/http/services.odata.org/V2/Northwind/Northwind.svc/",

{

json: true

}

);

sap.ui.getCore().setModel(oModel, 'data2');

\*/

},