SAP Mobility 101

Tutorial 7 – OData Model: data binding

# Objective of Exercise

## Build an example application

The objective of this exercise is to build an HTML page that uses JavaScript and output values in a table.

## Note

* We recommend that you use a chrome browser for testing
* Eclipse Juno will be needed for this tutorial

Tutorial 7

OData Model: data binding

# Task 1: Create the connection

The following code can be used to create an instance to the data model;

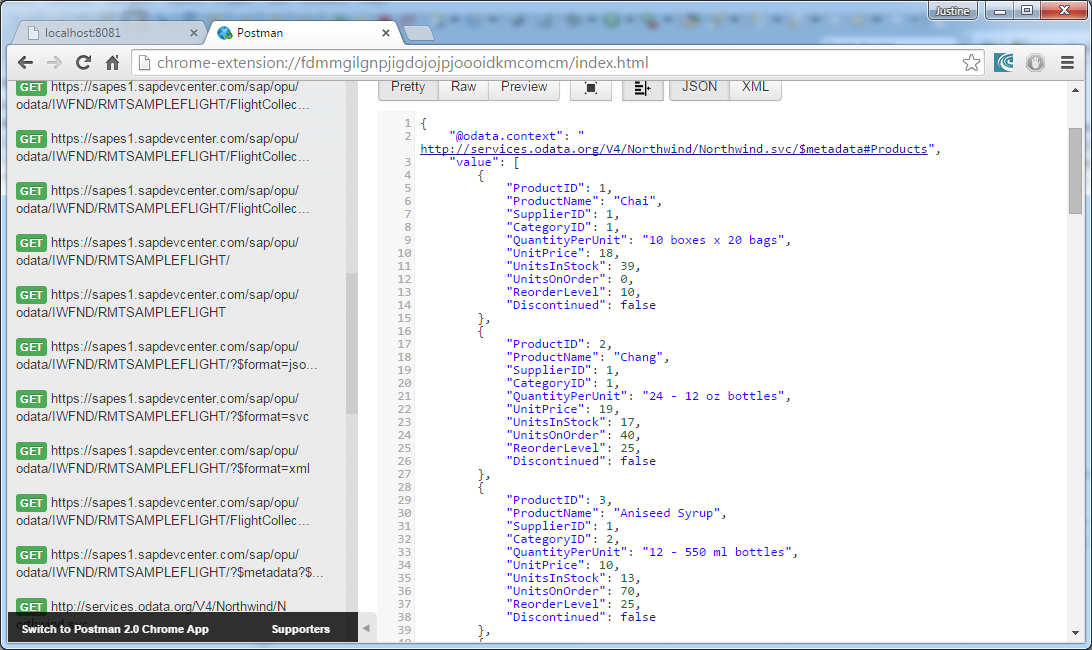
Var oModel = new sap.ui.model.odata.ODataModel(<URL>)

The URL that we are using for this tutorial is:

<http://services.odata.org/V2/Northwind/Northwind.svc>

# Task 2: Create the data Model

* In the main.controller.js file, create an odata model in the onInit function that takes the data in json format since the northwind data looks like this:



Since we are working with sap.m, the new way of defining the url is to add ‘proxy/http/’ as a prefix to the web address

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

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

{

json: **true**

}

);

Set the model with id = ‘data1’.

The final function will look like this:

onInit: **function**() {

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

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

{

json: **true**

}

);

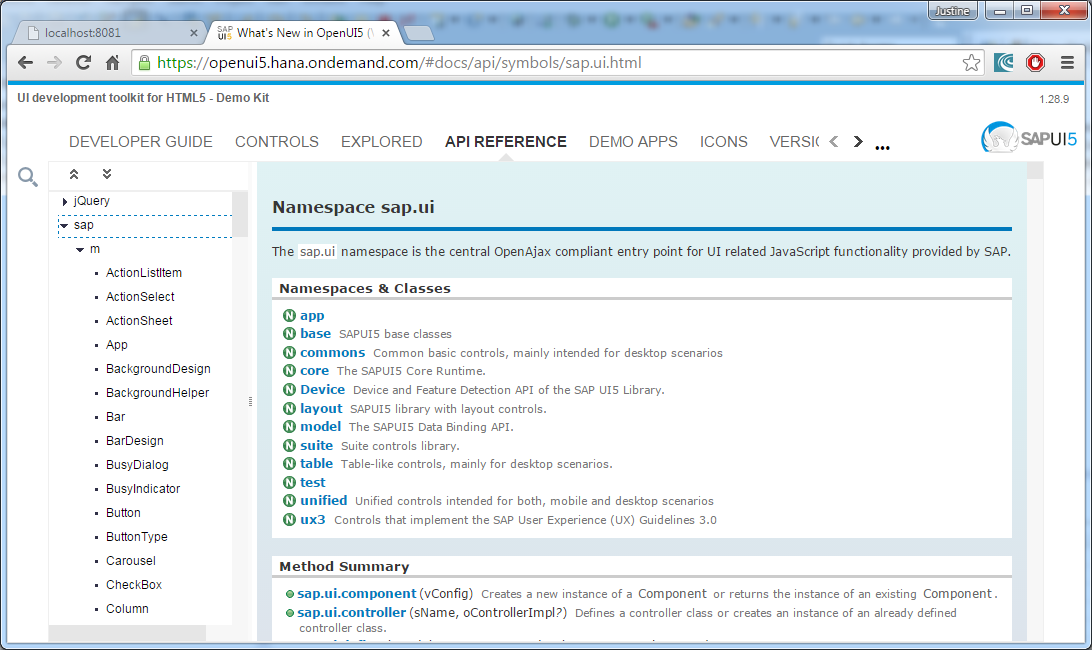
sap.ui.getCore().setModel(oModel, 'data1');

},

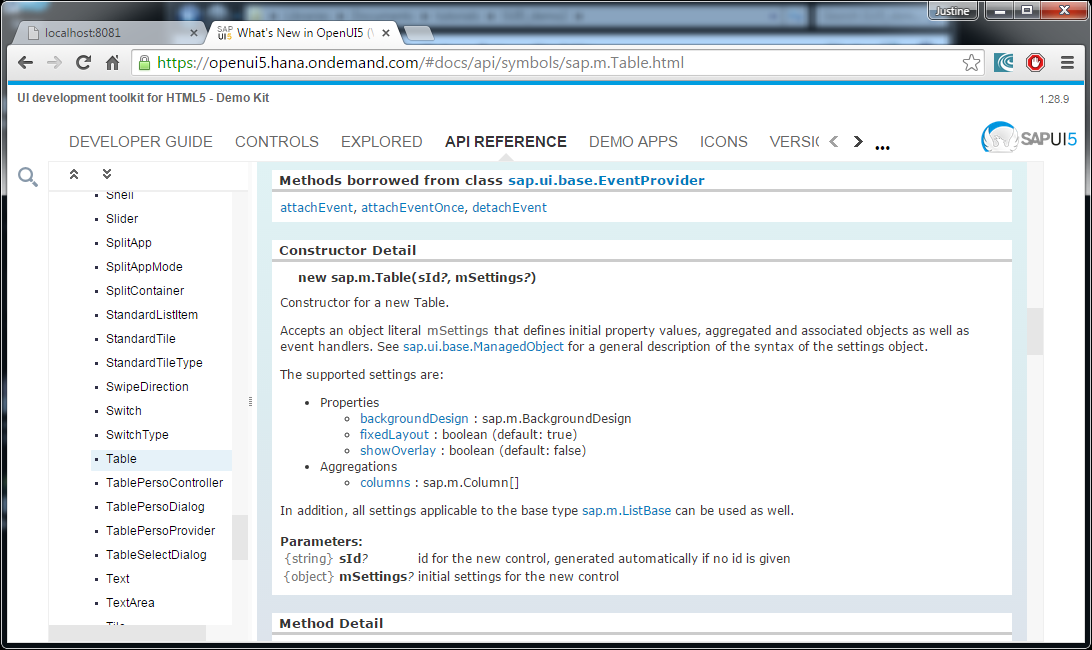
This function will be called when the main view are initialized.

# Task 3: How to use the sap.m library

* Open the following webpage in browser: <https://openui5.hana.ondemand.com>.
* Select the API REFERENCE tab at the top.
* On you left hand side go to SAP>m.

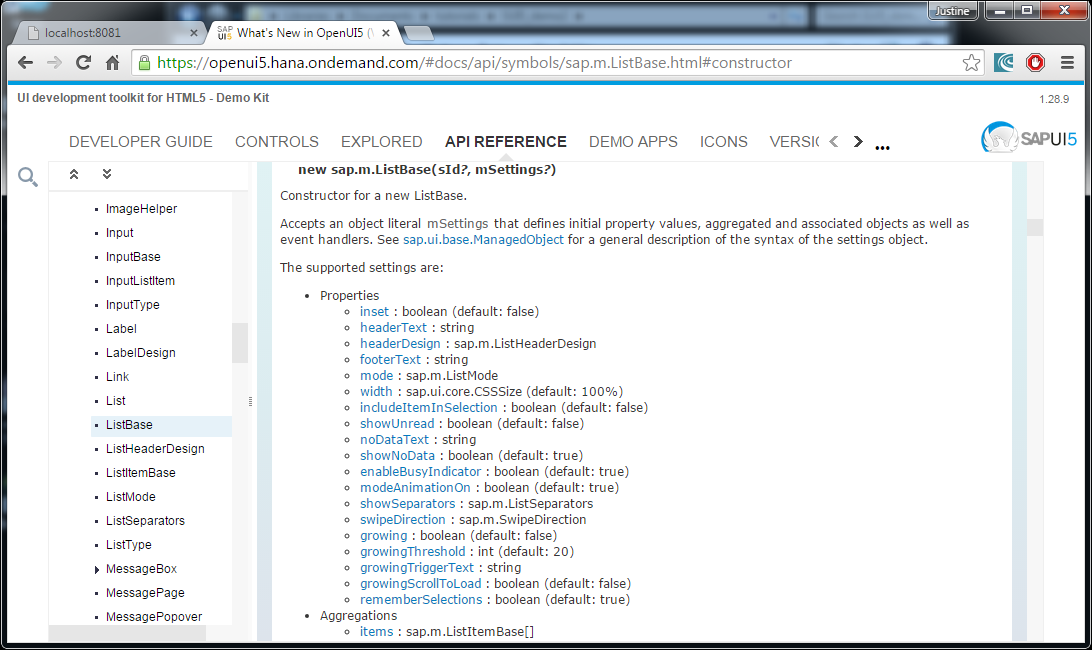


Here you will find a lift of API’s that you can use in your project. For this project, you want to create a table, thus go to the table API:



* Create your table in eclipse:

**oTable = new** sap.m.Table({});

Under the API reference of Table, the properties of Table can be set, as well as Aggregations. Additional settings that can be set is also listed under sap.m.ListBase():

In this list there are a Property called headerText, use this property to create a label for the table:

**return** **new** sap.m.Table(

{

headerText: "Categories",

}

Note: The content settings of the Table body is in JSON format.

# Task 4: add Columns to the Table

* Set aggregation columns:[]
* Create three new Columns, ID, Name and Unit Quantity.

The code should look as follow:

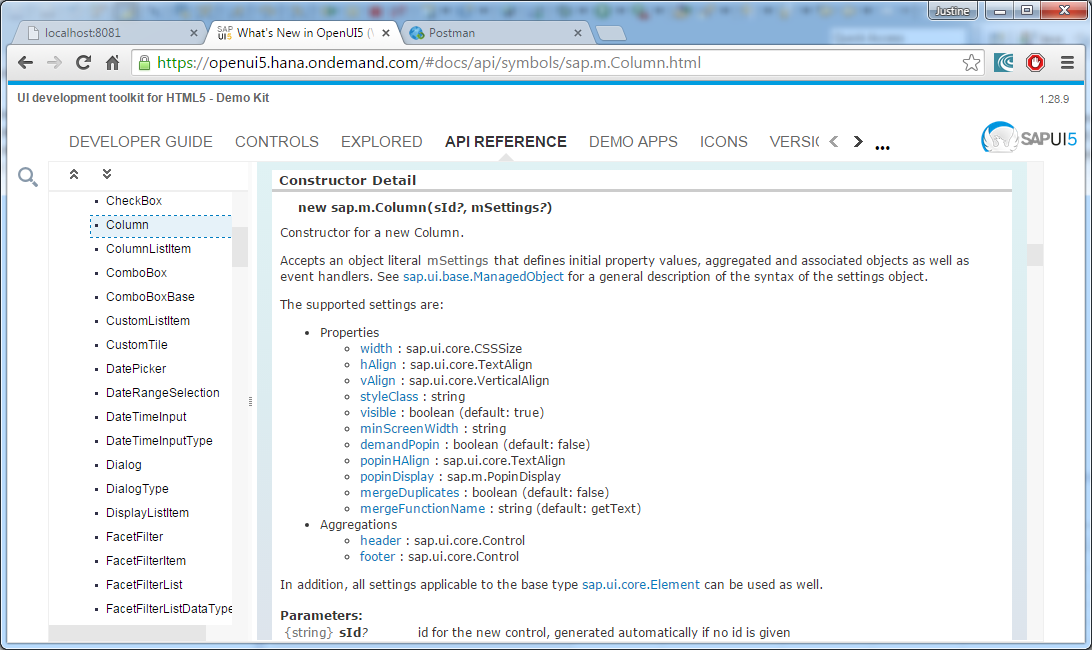
**return** **new** sap.m.Table(

{

headerText: "Categories",

column:[]

}

* Go to the column API for details on how to create a new Column and what properties can be set: 
* Create a Header that is a label with text: ‘ID’

**new** sap.m.Column(

{

header: **new** sap.m.Label(

{

text: "ID"

}

)

}

),

The code would look like this by now:

**oTable =** **new** sap.m.Table(

{

headerText: "My first table",

columns: [

**new** sap.m.Column(

{

header: **new** sap.m.Label(

{

text: "ID"

}

)

}

),

]

}

)

* Add two more columns with text: “name” and “Description”.

# Task 5: Create a template

Under the methods borrowed from other classes, the setModel method can be found

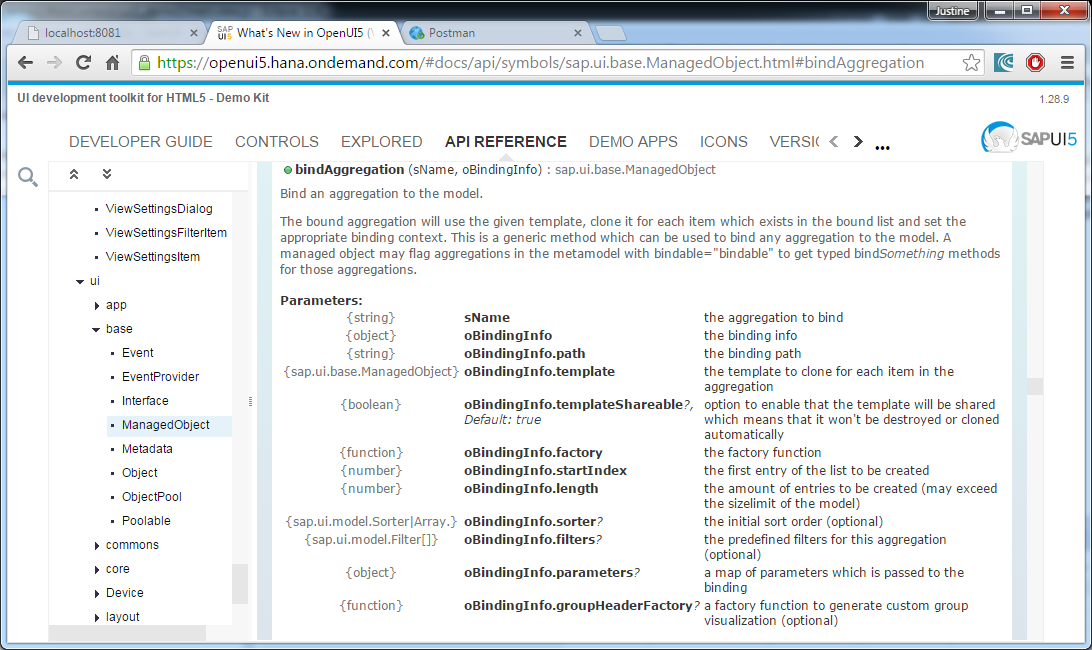
* Create the template for the cells that should be pulled from the data1 source, create a cell for CategoryID, CategoryName, and Description.
* **var** otemplate = **new** sap.m.ColumnListItem({
* cells: [
* **new** sap.m.Text({text: "{data1>CategoryID}"}),
* **new** sap.m.Text({text: "{data1>CategoryName}"}),
* **new** sap.m.Text({text: "{data1>Description}"})
* ]
* });

# Task 6: bindItems

Under the methods borrowed from other classes, the bindItems method can be found under sap.m.ListBase.

* Add bindItems as an extension to the Table()method.

oTable = new sap.m.Table( \*\*\*\*\*) .bindItems();



* Create a path to the Products table of the Nortwind Database:

Path: “/Products”,

* {
* path: "/Categories",
* )
* }

Add the themplate to the bindItemd method:

oTable.bindItems({

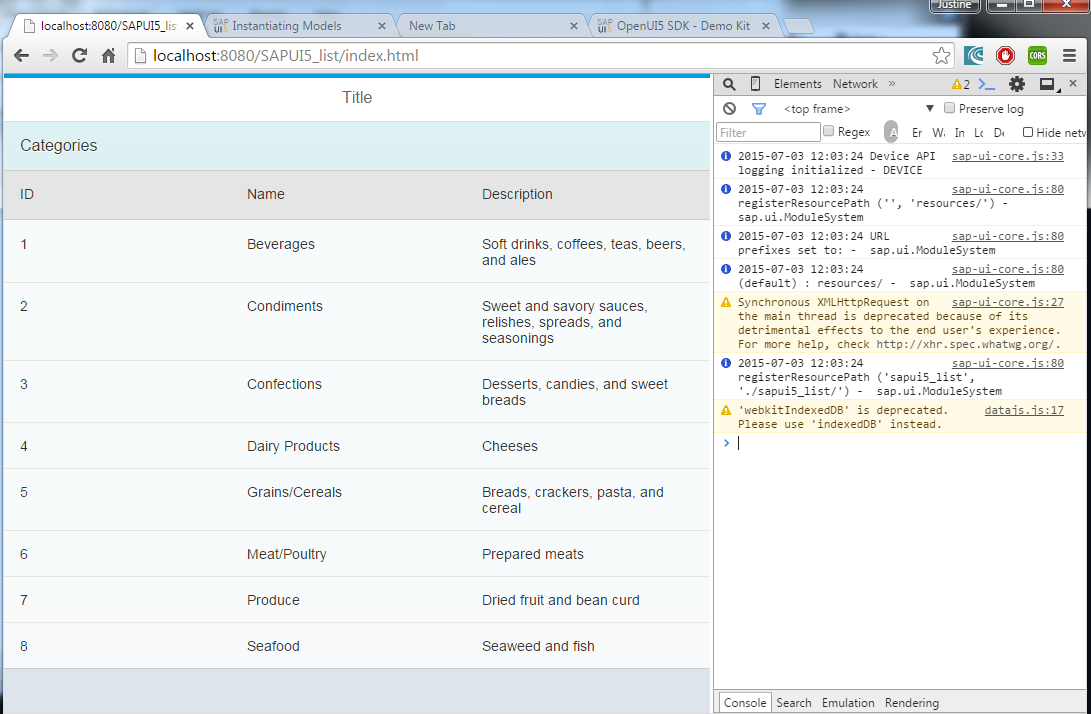
path: "data1>/Categories",

template: otemplate,

});

Return the table to the HTML file.

# Task 6: The result

* Save and Build the project, refresh the localhost:8080 page in your browser, the following page should appear: 

The final code in the main.view.js file;

sap.ui.jsview("sui5\_demo2.main", {

/\*\* Specifies the Controller belonging to this View.

\* In the case that it is not implemented, or that "null" is returned, this View does not have a Controller.

\* **@memberOf** sui5\_demo2.main

\*/

getControllerName : **function**() {

**return** "sui5\_demo2.main";

},

/\*\* Is initially called once after the Controller has been instantiated. It is the place where the UI is constructed.

\* Since the Controller is given to this method, its event handlers can be attached right away.

\* **@memberOf** sui5\_demo2.main

\*/

createContent : **function**(oController) {

oTable = **new** sap.m.Table({

headerText:"Categories",

columns: [

**new** sap.m.Column({header: **new** sap.m.Label({text: "ID"})}),

**new** sap.m.Column({header: **new** sap.m.Label({text: "Name"})}),

**new** sap.m.Column({header: **new** sap.m.Label({text: "Description"})})

]

});

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

cells: [

**new** sap.m.Text({text: "{data1>CategoryID}"}),

**new** sap.m.Text({text: "{data1>CategoryName}"}),

**new** sap.m.Text({text: "{data1>Description}"})

]

});

oTable.bindItems({

path: "data1>/Categories",

template: otemplate,

});

**return** **oTable;**

});