SAP Mobility 101

Tutorial 20 – Publish subscribe

# Objective of Exercise

## Build an example application

The objective of this exercise is to make use of the Publish Subscribe design by using the eventbus.

## Note

* We recommend that you use a chrome browser for testing
* Eclipse Luna would be needed for this Tutorial.
* We will be continuing with the SAPUI5\_XML code.

# Task 1: Understanding the model

Instead of having two separate objects like two separate views operating on each other, we can use an event system. In this event system, methods can be subscribed to an event; another method can execute that event and publish data to the subscribed methods.

Each component has its own event bus.

The .publish method can be used to publish an event.

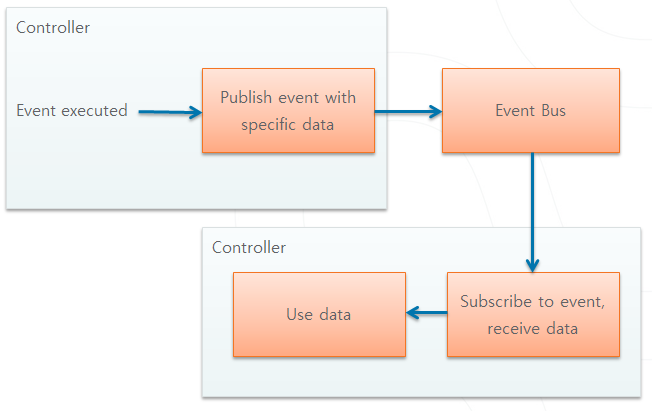
Eg:

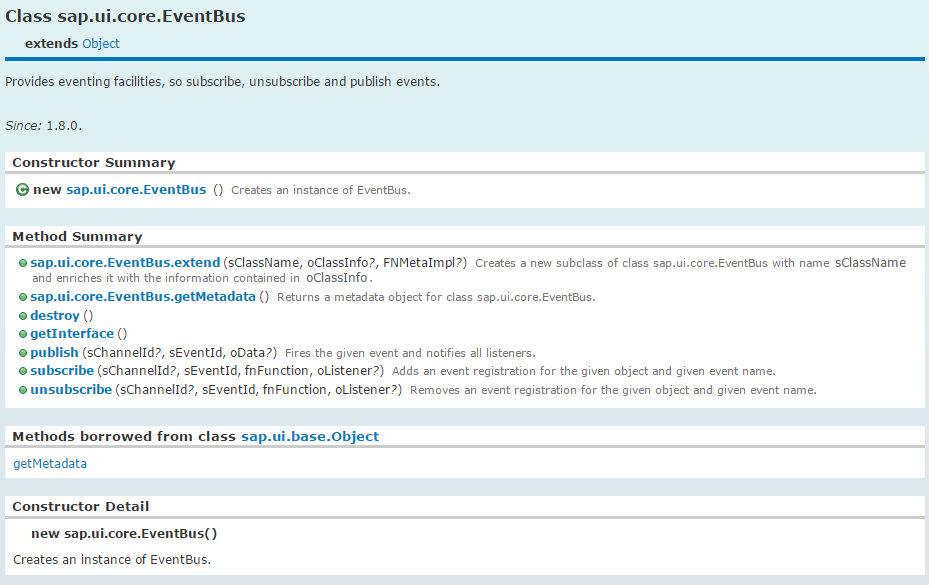
1. **var** oEventBus = sap.ui.core.Component.getOwnerComponentFor(**this**.getView()).getEventBus();
3. oEventBus.publish(
4. “Event",
5. {
6. data1: "Data object that you want to pass to the subscribed method",
7. data2: "Some data"
8. }
9. );

The .subscribe method can be used to subscribe to a specific event.

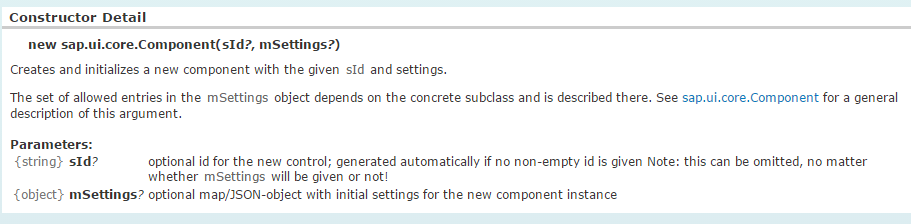
Eg;

1. onInit: **function**() {
3. **var** oEventBus = sap.ui.core.Component.getOwnerComponentFor(**this**.getView()).getEventBus();
4. oEventBus.subscribe(
5. "Event",
6. **this**.onEvent,
7. **this**
8. );
10. },
12. onEvent: **function**(sId, sEventId, oData){
13. console.log(oData.dataKey1);
14. }





As in the examples above, each eventBus belongs to a Component:

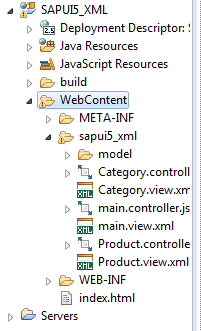


Let us learn how the eventbus and component works by example.

# Task 2: Add data

Let’s put the exact same mock server in the code as with the Javascript code.

First add the “model” file to the WebContent directory.



Add the library of the MockServer to the index file



In the onInit file of your main.controller file, set the mockserver data.

Note: the controller methods and functions stay the same.

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\_xml/model/metadata.xml";

**var** mockdatabase = "sapui5\_xml/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 3: Create list

Create a list in the Categories view, the list that will be created will be exactly the same as in tutorial 18. We want a list with id “listC” the items can be bound to the mock data by setting the items attribute to the Categories table of the data1 model. Use the DisplayListItem method to display the Categoryname and ID.

<List id=*"listC"* items=*"{data1>/Categories}"*>

<items>

<DisplayListItem label=*"{data1>CategoryName}"* value=*"{data1>CategoryID}"*/>

</items>

</List>

Set the selection mode of the list to SingleSelect mode. Because we are within a sap.m.listmode object, we do not have to add the prefix sap.m.listmode, we can just say that the mode=”SingleSelect”. Remember this for future use: **the controllers are all the same, but the views differ!!**

Create a function, in the Categories.controller file that would handle the event when an item is selected, call it CatSelect.

CatSelect: **function**(){

alert("i am called!!");

}

Call this function when an item is selected.

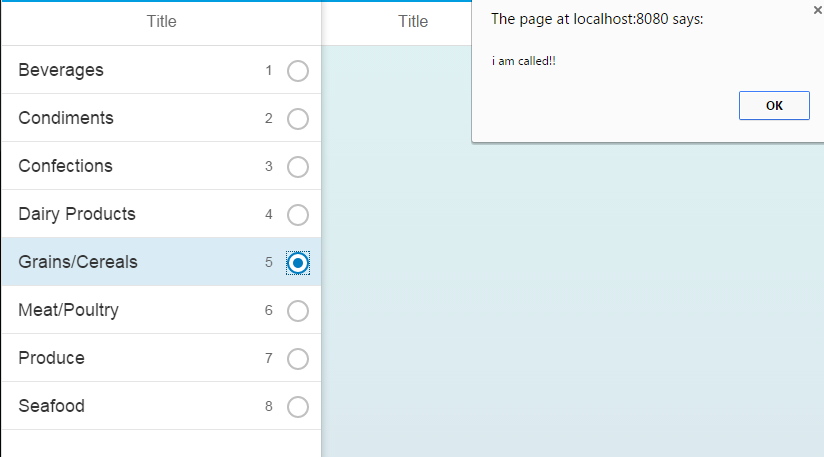
<List id=*"listC"* mode=*"SingleSelect"* select=*"CatSelect"* items=*"{data1>/Categories}"*>

<items>

<DisplayListItem label=*"{data1>CategoryName}"* value=*"{data1>CategoryID}"*/>

</items>

</List>

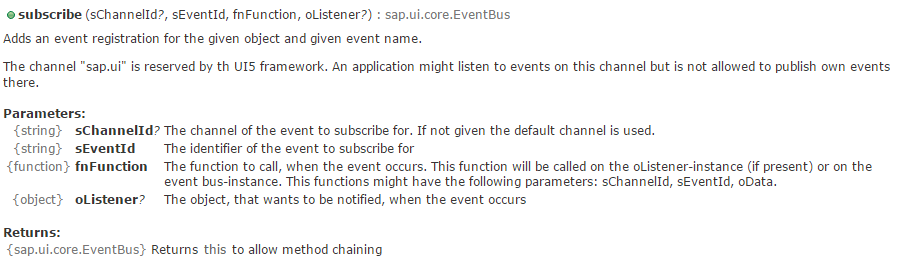


# Task 4: Get eventBus

In your main.controller file in the onInit method, get the event bus, we use this specific file and method since it will be the first to be executed when the initial view is opened.

**var** oBus = sap.ui.getCore().getEventBus();

# Task 5: Subscribe navigation methods



Now that we have the eventBus, we would like to subscribe two events, navigating to the previous page and navigating to the next page. The events should be called “to” and “back”, we are working with a SplitApp so we would have to use the terms “toM”, “toD” and “backD”, the functions to be called when the event occurs will be created later, but lets call them navToM, navToD and navBackD, the object to be notified when the event occurs should be the current view.

Call the channel we are using the “nav” channel.

The “this” term are used again, the “this” term are used to refer to the object currently operated in.

**var** oBus = sap.ui.getCore().getEventBus();

oBus.subscribe("nav", "toD", **this**.navToD, **this**);

oBus.subscribe("nav", "toM", **this**.navToD, **this**);

oBus.subscribe("nav", "backD", **this**.navBackD, **this**);

# Task 6: Create event handlers

navToD : **function**(channelId, eventId, data) { },//to detailed page

navToM : **function**(channelId, eventId, data){ },// to master page

navBackD : **function**(channelId, eventId, data) { }//back to previous page.

# Task 7: Call above Event Handlers

When an item is selected on the category list, the CatSelect function is called.

Retrieve the item selected.

path1=e.getParameter("listItem").oBindingContexts.data1.sPath;

**var** item = sap.ui.getCore().getModel('data1').getProperty(path1);

Publish data to the event bus, the channel to the used is the “nav” channel, the event to be notified will be the “toD” event which in turn are handled by the navToD method created above. The data that should be included should contain the id of the products page as well as the item selected.

sap.ui.getCore().getEventBus().publish("nav", "toD", {

id: 'Products',

context: item,

});

When this code is executed, the navToD function will be called as well.

# Task 8: NavToD

In your main.controll function, change the navToD function so the app will be directed to the products detail page.

The data that can now be used is the data of the event bus; data.id will be the products view id, data.context will be the item selected.

navToD : **function**(channelId, eventId, data) {

**var** oView = sap.ui.getCore().byId(data.id);

**var** oApp = sap.ui.getCore().byId("app");

oApp.toDetail(oView, "slide", data.context);

},

The same can be done for navToM

navToM : **function**(channelId, eventId, data) {

**var** oView = sap.ui.getCore().byId(data.id);

**var** oApp = sap.ui.getCore().byId("app");

oApp.toMaster(oView, "slide", data.context);

},

# Task 9: navBack

To go to the previous details page the .backDetail() method can be used:

navBackD : **function**(channelId, eventId, data) {

**var** oApp = sap.ui.getCore().byId("app");

oApp.backDetail();

}

Let’s add a navigation button to the Products page that would call a navback function.

<Page title=*"Products"* showNavButton=*"true"* navButtonTap=*"navback"*>

<content>

We do not have any data to transport along with the back function, but let’s notify the navBack.D method.

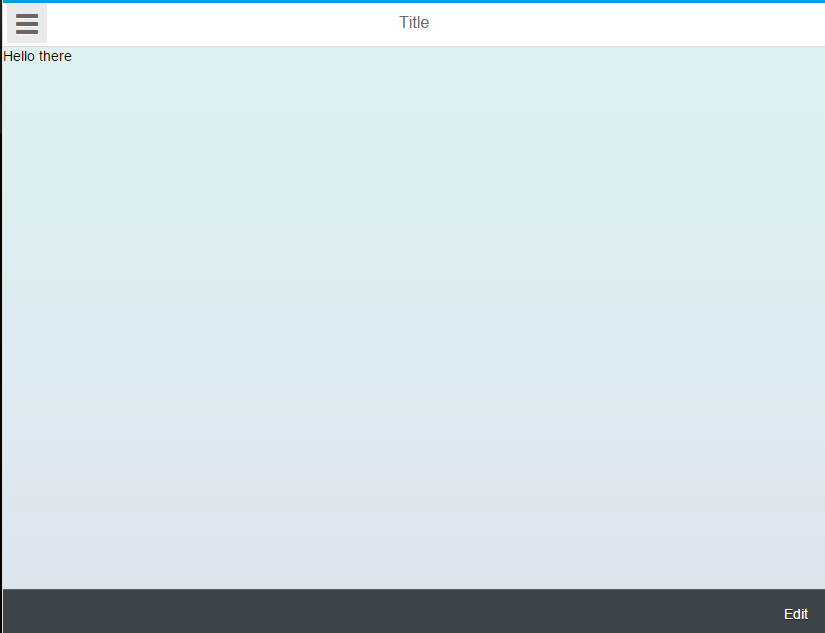
navback : **function**() {

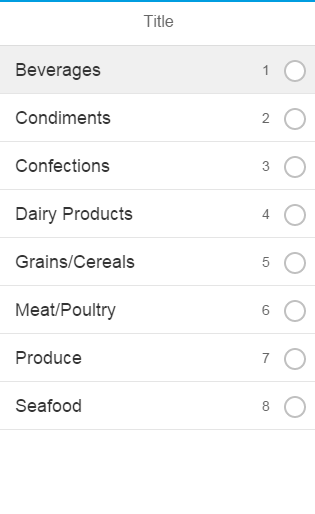
sap.ui.getCore().getEventBus().publish("nav", "backD", { });

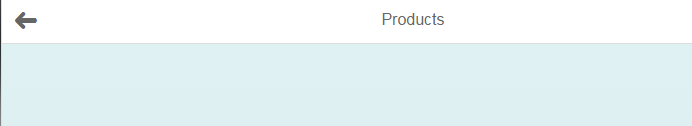
}

Test and run you application.

# Results







## Main.controller:

sap.ui.controller("sapui5\_xml.main", {

ButtonEdit: **function**(){

alert("this was pressed");

},

/\*\*

\* Called when a controller is instantiated and its View controls (if available) are already created.

\* Can be used to modify the View before it is displayed, to bind event handlers and do other one-time initialization.

\* **@memberOf** sapui5\_xml.main

\*/

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\_xml/model/metadata.xml";

**var** mockdatabase = "sapui5\_xml/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');

**var** oBus = sap.ui.getCore().getEventBus();

oBus.subscribe("nav", "toD", **this**.navToD, **this**);

oBus.subscribe("nav", "toM", **this**.navToM, **this**);

oBus.subscribe("nav", "backD", **this**.navBackD, **this**);

},

navToD : **function**(channelId, eventId, data) {

**var** oView = sap.ui.getCore().byId(data.id);

**var** oApp = sap.ui.getCore().byId("app");

oApp.toDetail(oView, "slide", data.context);

},

navToM : **function**(channelId, eventId, data) {

**var** oView = sap.ui.getCore().byId(data.id);

**var** oApp = sap.ui.getCore().byId("app");

oApp.toMaster(oView, "slide", data.context);

},

navBackD : **function**(channelId, eventId, data) {

**var** oApp = sap.ui.getCore().byId("app");

oApp.backDetail();

}

/\*\*

## Main.view.xml

<core:View xmlns:core=*"sap.ui.core"* xmlns:mvc=*"sap.ui.core.mvc"* xmlns=*"sap.m"*

controllerName=*"sapui5\_xml.main"* xmlns:html=*"http://www.w3.org/1999/xhtml"*>

<Page title=*"Title"*>

<Text text=*"Hello there"*/>

<content>

</content>

<footer>

<Bar>

<contentRight>

<Button text=*"Edit"*

press=*"ButtonEdit"*/>

</contentRight>

</Bar>

</footer>

</Page>

</core:View>

## Category.controller.js

sap.ui.controller("sapui5\_xml.Category", {

CatSelect: **function**(e){

path1=e.getParameter("listItem").oBindingContexts.data1.sPath;

**var** item = sap.ui.getCore().getModel('data1').getProperty(path1);

sap.ui.getCore().getEventBus().publish("nav", "toD", {

id: 'Products',

context: item,

});

},

## Category.view.xml

<core:View xmlns:core=*"sap.ui.core"* xmlns:mvc=*"sap.ui.core.mvc"* xmlns=*"sap.m"*

controllerName=*"sapui5\_xml.Category"* xmlns:html=*"http://www.w3.org/1999/xhtml"*>

<Page title=*"Title"* >

<content>

<List id=*"listC"* mode=*"SingleSelect"* select=*"CatSelect"* items=*"{data1>/Categories}"*>

<items>

<DisplayListItem label=*"{data1>CategoryName}"* value=*"{data1>CategoryID}"*/>

</items>

</List>

</content>

</Page>

</core:View>

## Product.controller.js

sap.ui.controller("sapui5\_xml.Product", {

navback : **function**() {

sap.ui.getCore().getEventBus().publish("nav", "backD", { });

}

## Product.view.xml

<core:View xmlns:core=*"sap.ui.core"* xmlns:mvc=*"sap.ui.core.mvc"* xmlns=*"sap.m"*

controllerName=*"sapui5\_xml.Product"* xmlns:html=*"http://www.w3.org/1999/xhtml"*>

<Page title=*"Products"* showNavButton=*"true"* navButtonTap=*"navback"*>

<content>

</content>

</Page>

</core:View>