SAP Mobility 101

Tutorial 17– XML full Application

# Objective of Exercise

## Build an example application

The objective of this exercise is to build an application with views in XML, we want the index to point to a main xml file that would contain the application. The application file would point to the different views. We also want to edit attributes and add aggregations of elements.

## Note

* We recommend that you use a chrome browser for testing
* Eclipse Luna would be needed for this Tutorial.
* This code will be used for the next tutorial aswell so lets call this code SAPUI5\_XMLfull

# Task 1: index.html

We want to use the MockServer for this application, thus include the MockServer library.

<script>

jQuery.sap.require("sap.ui.core.util.MockServer");

Register the path to the files, the files are stored in the folder named sapui5\_xml the program directory should thus be appended by /sapui5\_xml to get the files. Each program file has the name sapui5\_xml.\*filename\*.

<script>

jQuery.sap.require("sap.ui.core.util.MockServer");

jQuery.sap.registerModulePath("sapui5\_xml", "./sapui5\_xml/");

Place an xml view at the content of the index page, the view you want to place is named sapui5\_xml.main.

<script>

jQuery.sap.require("sap.ui.core.util.MockServer");

jQuery.sap.registerModulePath("sapui5\_xml", "./sapui5\_xml/");

sap.ui.xmlview("components", "sapui5\_xml.main").placeAt("content");

# 

# Task 2: main file: App

Create two new views in XML the same way as above, the only exception is to just create the view and not the entire application, name the two new views products and Categories.

Your index file are pointing to the main file, change the main file from a page to an App with id=”app”

<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"*>

<App id=*"app"*>

</App>

</core:View>

# Task 3: Point to views

Within your app component, create the two views Categories and products with ids category and product in mvc:XMLView

Note: the viewName can be found at the top of the view file.



We want the Categories page to be the first, thus declare it first.

<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"*>

<App id=*"app"*>

<mvc:XMLView viewName=*"sapui5\_xml.Categories"* id=*"category"*/>

<mvc:XMLView viewName=*"sapui5\_xml.products"* id=*"product"*/>

</App>

</core:View>

# Task 4: Categories Page

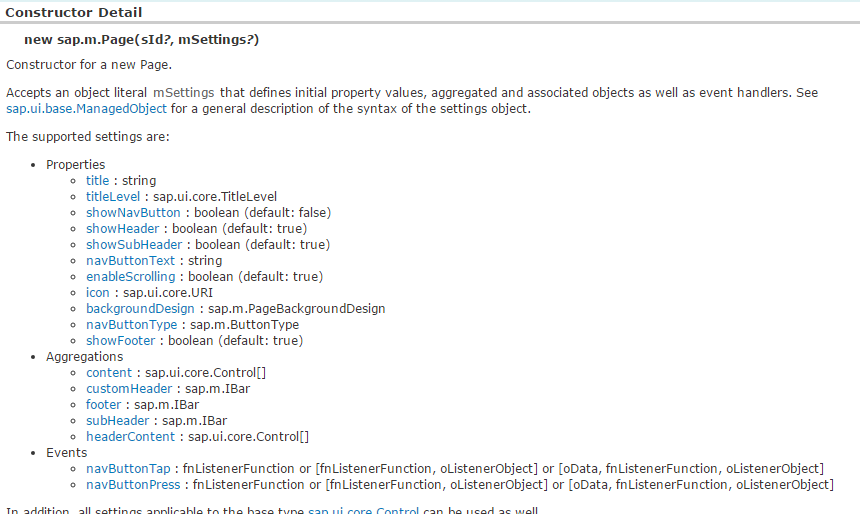
The Categories view currently looks something like this:

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

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

</core:View>

Create a page Object within the View.



Set the following attributes:

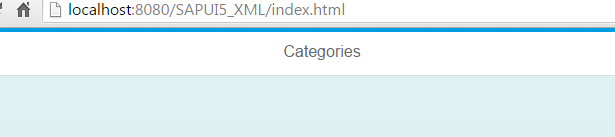
title = “Categories”

<Page title=*"Categories"* shownavButton=*"true"*>

</Page>

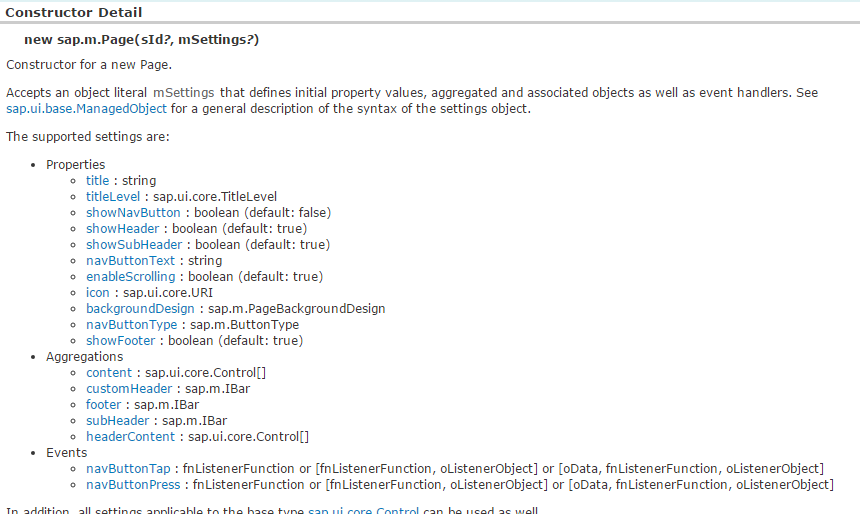
Save and run you application.

A page with the title categories should appear in your browser.



# Task 5: Products page

Create a page Object within the View.



Set the following attributes:

title = “Products”

shownavButton=”true”

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

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

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

</Page>

</core:View>

# Task 6: Content

One of the aggregations of a page is the content; the aggregations of the page can be set within the body of the page, not the same as with attributes.

<Page title=*"Categories"* >

<content></content>

</Page>

The content of the Page are handled exactly the same way as with javascript views, remember the elements and their properties are all exactly the same the only difference is they are now used in xml and not in javascript.

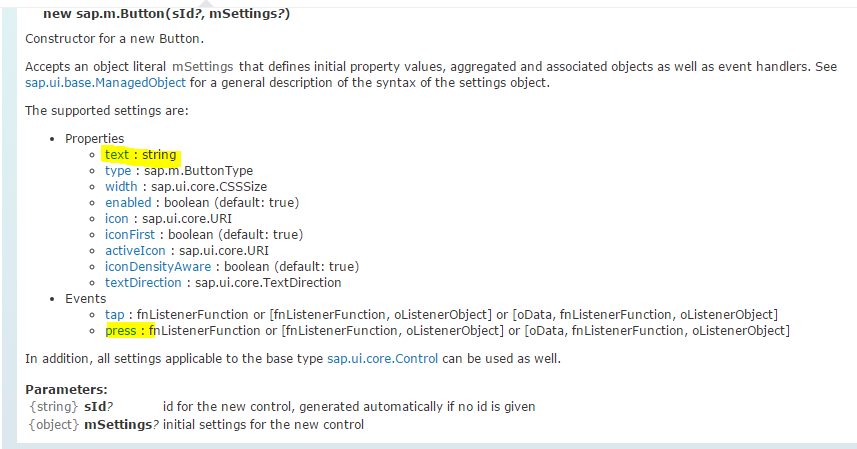
Add a button to the content.

<Page title=*"Categories"* >

<content>

<Button></Button>

</content>

</Page> 

Set the text attribute to “products”.

# Task 7: Events

An event is also incorporated as an attribute, if the button is PRESSsed we want an event handler to be called, the event handling method will be created in the Categories.controller file.

Go to the categories.controller file and create a function in javascript named gotoProducts.

gotoProducts: **function**(){ //this function is in the control file

alert(“button pressed”);

}

When the button is pressed, this function should be called, but note how the reverencing differs from that of a JavaScript view.

In JavaScript we would call the function using reverence oController.gotoProducts, here we just use “gotoProducts”.

<Page title=*"Categories"* >

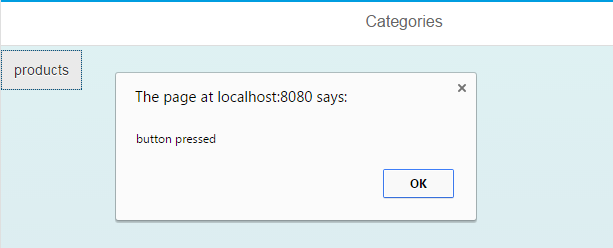
<content>

<Button text=*"products"* press=*"gotoProducts"*></Button>

</content>

</Page>

Save and run you application.



# Task 8: Navigation

When the button is pressed we want to navigate to the products page.

Navigation works a bit differently, with javascript we would have just gone app.to(“products”).

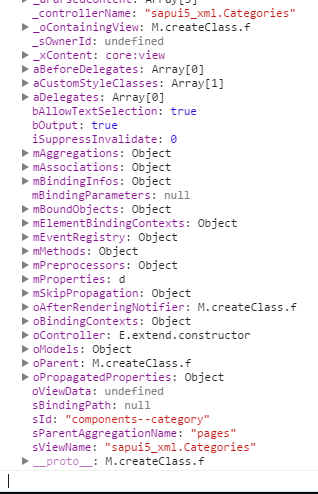
Now we will first find the current view. We are going to use “this”. You would have encountered “this” in java or javascript before even in C. when we use “this”, we are referring the object we are currently in, in this context we are currently in a core:XMLView. Thus if we get the view we use this.getView(). Print out to the console for inspection.

gotoProducts: **function**(){

**var** oView = **this**.getView();

console.log(oView);

}



Can you see the id at the bottom?

Do you notice how the id differs from the actual id we have given?

Now we know how to reference the views within the category.

We want the application that we have created in the xmlview thus we would use the id: “components- -app”, and to get the page we want to navigate to we would use “components- -product”.

gotoProducts: **function**(){

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

app.to("components--product");

}

We want the app to navigate back to the products page when the navigation button is pressed.

Thus create the controller.

goback: **function**(){

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

app.back();

}

and point the attribute to the goback function.

<Page title=*"Products"* showNavButton=*"true"* navButtonPress=*"goback"*>

</Page>

Save and run you application, the app should be able to navigate between the pages.

## Index.html code:

<!DOCTYPE HTML>

<html>

<head>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta http-equiv=*'Content-Type'* content=*'text/html;charset=UTF-8'*/>

<script src=*"resources/sap-ui-core.js"*

id=*"sap-ui-bootstrap"*

data-sap-ui-libs=*"sap.m"*

data-sap-ui-theme=*"sap\_bluecrystal"*>

</script>

<!-- only load the mobile lib "sap.m" and the "sap\_bluecrystal" theme -->

<script>

jQuery.sap.require("sap.ui.core.util.MockServer");

jQuery.sap.registerModulePath("sapui5\_xml", "./sapui5\_xml/");

sap.ui.xmlview("components", "sapui5\_xml.main").placeAt("content");

</script>

</head>

<body class=*"sapUiBody"* role=*"application"*>

<div id=*"content"*></div>

</body>

</html>

## 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"*>

<App id=*"app"*>

<mvc:XMLView id=*"category"* viewName=*"sapui5\_xml.Categories"* />

<mvc:XMLView id=*"product"* viewName=*"sapui5\_xml.products"* />

</App>

</core:View>

## Categories.view.xml

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

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

<Page title=*"Categories"* >

<content>

<Button text=*"products"* press=*"gotoProducts"*></Button>

</content>

</Page>

</core:View>

## Categories.controller.js

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

/\*\*

\* 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.Categories

\*/

// onInit: function() {

//

// },

gotoProducts: **function**(){

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

app.to("components--product");

}

## Products.View.xml

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

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

<Page title=*"Products"* showNavButton=*"true"* navButtonPress=*"goback"*>

</Page>

</core:View>

## Products.controller.js

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

/\*\*

\* 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.products

\*/

// onInit: function() {

//

// },

goback: **function**(){

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

app.back();

}