FIORI 2.0 FOR YOUR UI5 APPLICATION

Exercise / Solutions

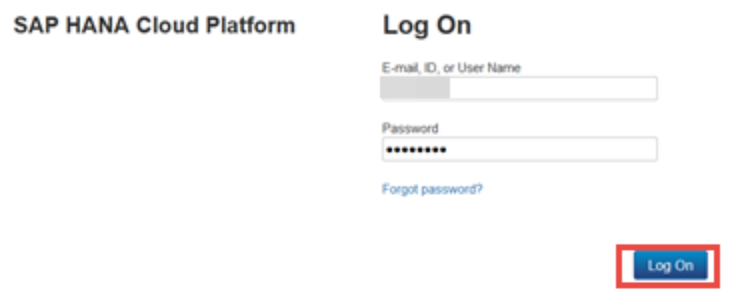
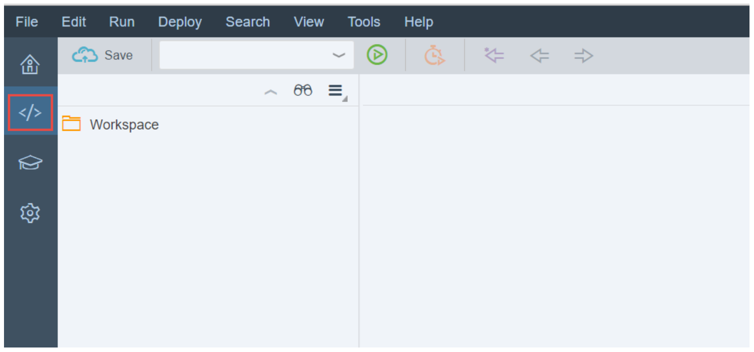
Stanislava Baltova SAP SE
Yavor Ivanov SAP SE

TABLE OF CONTENTS

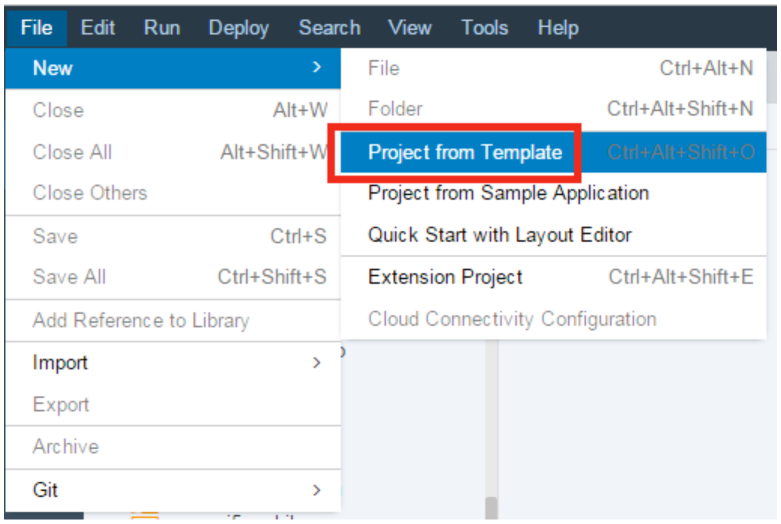
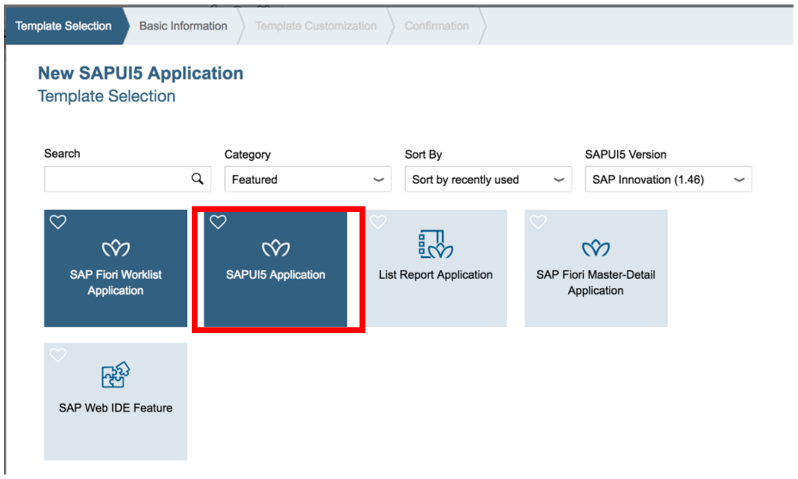
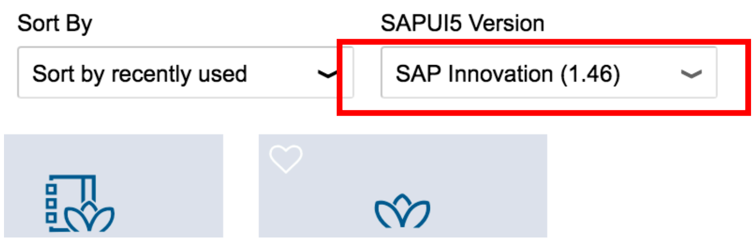
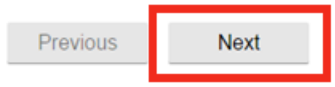
1 SAP Web IDE ... 3
2 Generate Basic UI5 App from Template ... 4
3 Bootstrap ... 7
4 App Descriptor – Manifest json ... 8
5 Model ... 9
6 Master View ... 9
7 Master Controller ... 12
8 View Content and Floating Footer ... 15
9 Create Master-Detail-Detail Pattern with Flexible Column Layout ... 16
10 Detail View ... 17
11 Detail Controller ... 17
12 Detail-Detail View ... 18
13 Detail-Detail Controller ... 18
14 Routing ... 18
15 Master-Detail Finalization – Object Page ... 22

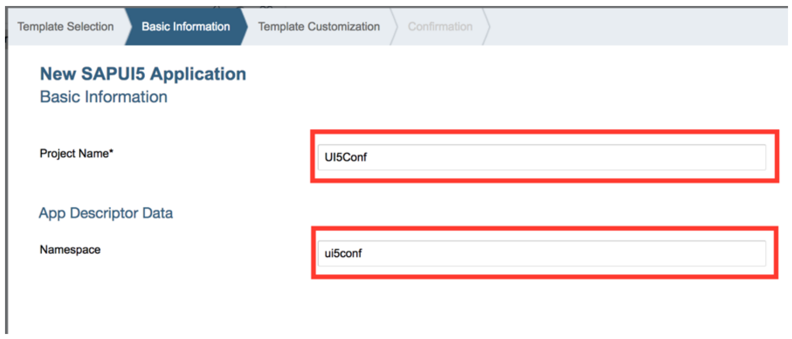
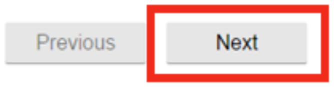
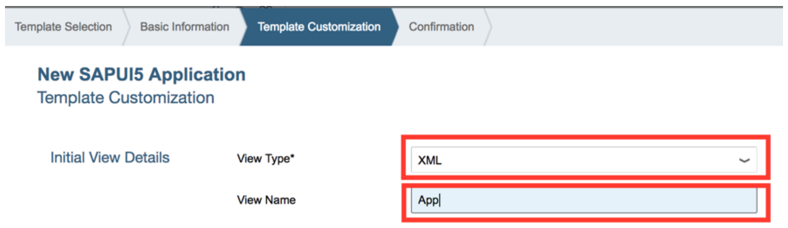
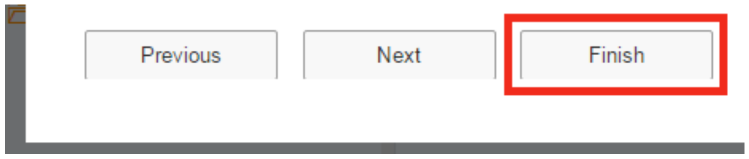
1 SAP WEB IDE

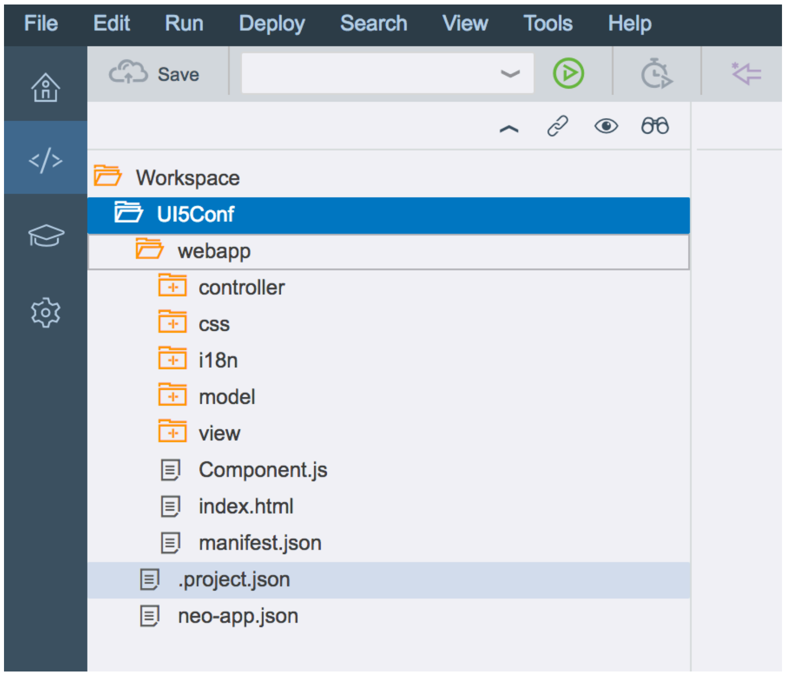
1.1.1 Access Your Development Environment

Explanation	Screenshot
1. Open Google Chrome	
2. Click on one of the following links to open the SAP Web IDE: https://webide-d85556318.dispatcher.us2.hana.ondemand.com/	
Log in with the username and password combination provided by the speaker.	 SAP HANA Cloud Platform Log On E-mail, ID, or User Name Password ••••••••• Forgot password? Log On
3. Go to the development perspective by clicking on the icon "</>" in the left toolbar. On the left side you will see your workspace, that later contains a folder list with your app projects. **Note:** From now on we will assume that you have SAP Web IDE opened in the development perspective to work on your app projects.	 File Edit Run Deploy Search View Tools Help Save Workspace

2 GENERATE BASIC UI5 APP FROM TEMPLATE

Explanation	Screenshot
1. Open the **File** menu of SAP Web IDE 2. Select **New** 3. Select **Project from Template**	 File Edit Run Deploy Search View Tools Help New Close Alt+W Close All Alt+Shift+W Close Others Save Ctrl+S Save All Ctrl+Shift+S Add Reference to Library Import Export Archive Git File Ctrl+Alt+N Folder Ctrl+Alt+Shift+N Project from Template Ctrl+Alt+Shift+O Project from Sample Application Quick Start with Layout Editor Extension Project Ctrl+Alt+Shift+E Cloud Connectivity Configuration
On the **Template Selection** step: 4. Make sure that **Featured** is selected as filter. 5. Select the **SAP Fiori Worklist Application** (note: it might not be the leftmost template in the list)	 Template Selection Basic Information Template Customization Confirmation New SAPUI5 Application Template Selection Search Category Featured Sort By Sort by recently used SAPUI5 Version SAP Innovation (1.46) SAP Fiori Worklist Application SAPUI5 Application List Report Application SAP Fiori Master-Detail Application SAP Web IDE Feature
6. Observe that the latest SAPUI5 version is automatically selected. **Note:** In this session, you should work with the latest released version which contains all Fiori 2.0 artefacts. (1.46)	 Sort By Sort by recently used SAPUI5 Version SAP Innovation (1.46)
7. Press the **Next** button to continue to the next step	

Explanation	Screenshot
On the **Basic Information** Step 8. Enter **UI5Conf** as the **Project Name** and **ui5conf** as **Namespace**	 Template Selection / Basic Information / Template Customization / Confirmation New SAPUI5 Application Basic Information Project Name* UI5Conf App Descriptor Data Namespace ui5conf
9. Press the **Next** button to continue to the next step	
On the **Template Customization** Step 10. Choose **XML** as **View Type** and enter **App** as **View Name**	 Template Selection / Basic Information / Template Customization / Confirmation New SAPUI5 Application Template Customization Initial View Details View Type* XML View Name App
11. Press the **Finish** button to end the wizard	

Explanation	Screenshot
12. Check that a new folder **UI5Conf** is added to your **Workspace**.	

3 BOOTSTRAP

Switch URL to one of the following to get the latest available version of UI5

- https://openui5.hana.ondemand.com/resources/sap-ui-core.js
- https://sapui5.hana.ondemand.com/resources/sap-ui-core.js

3.1.1 webapp/index.html

```
<!DOCTYPE html>
<html>
	<head>
		<meta http-equiv="X-UA-Compatible" content="IE=edge">
		<meta charset="utf-8">
		<title>Developing Web Apps with SAPUI5</title> //TODO Change it!

		<script
			id="sap-ui-bootstrap"
			src="https://sapui5.hana.ondemand.com/resources/sap-ui-core.js"
			data-sap-ui-theme="sap_belize"
			data-sap-ui-libs="sap.m"
			data-sap-ui-compatVersion="edge"
			data-sap-ui-resourceroots='{"ui5confUI5Conf": ""}'>
		</script>
…

</html>
```

Before we can do anything with SAPUI5, we need to load and initialize it. This process of loading and initializing SAPUI5 is called bootstrapping. In the code above, we load the SAPUI5 framework from the Content Delivery Network (CDN) at https://sapui5.hana.ondemand.com/resources/sap-ui-core.js.

```
		<script>
			sap.ui.getCore().attachInit(function() {
				new sap.m.Shell({
					app: new sap.ui.core.ComponentContainer({
						height : "100%",
						name : "ui5confUI5Conf"
					}),
					appWidthLimited: false //What it is for??
				}).placeAt("content");
			});
		</script>
```

4 APP DESCRIPTOR – MANIFEST JSON

4.1.1 webapp/manifest.json – modify manifest file

In this step we will modify the app descriptor file changing the root view name, removing unnessessary dependencies and adding a default data model for our application.

```
	"sap.ui5": {
		"rootView": {
			"viewName": "ui5confUI5Conf.view.Master",
			"type": "XML"
		},
		"dependencies": {
			"minUI5Version": "1.46.0",
			"libs": {
				"sap.ui.core": {},
				"sap.m": {},
				"sap.f": {},
				"sap.ui.layout": {},
				"sap.uxap": {}
			}
		},
		"contentDensities": {
			"compact": true,
			"cozy": true
		},
		"models": {
			"i18n": {
				"type": "sap.ui.model.resource.ResourceModel",
				"settings": {
					"bundleName": "ui5confUI5Conf.i18n.i18n"
				}
			},
			"": {
				"type": "sap.ui.model.json.JSONModel",
				"uri": "model/model.json"
			}
		},
		"resources": {
			"css": [{
				"uri": "css/style.css"
			}]
		}
	}
```

5 MODEL

5.1.1 Create webapp/model/model.json [NEW]

In this step we will create the json model wich will populate data into our master view.

Copy the json model content from the following url:

https://raw.githubusercontent.com/d3xter666/2017-ui5con-fiori/master/webapp/model/products.json

6 MASTER VIEW

6.1.1 Create webapp/view/Master.view.xml [NEW]

In this step we will create the Master view of our application using the newly introduced in Fiori 2.0 basic layout control – Dynamic Page.

We will create only the Title and Header aggregations of the Dynamic Page control and add content to them.

```
<mvc:View controllerName="ui5confUI5Conf.controller.Master"
xmlns:mvc="sap.ui.core.mvc" xmlns:core="sap.ui.core"
xmlns:layout="sap.ui.layout"
      xmlns="sap.m" xmlns:f="sap.f" displayBlock="true" height="100%">
      <f:DynamicPage id="dynamicPageId" >
            <!-- DynamicPage Title -->
            <f:title>
                  <f:DynamicPageTitle>
                        <f:heading>
                              <Title text="My Products"/>
                        </f:heading>
                        <f:actions>
                              <ToolbarSpacer/>
                              <OverflowToolbarButton icon="sap-icon://add"
text="Add" type="Transparent"/>
                              <OverflowToolbarButton icon="sap-icon://edit"
text="Edit" type="Transparent"/>
                              <OverflowToolbarButton icon="sap-icon://delete"
text="Delete" type="Transparent"/>
                              <Button icon="sap-icon://post" text="Toggle
Footer" type="Transparent"/>
                        </f:actions>
                  </f:DynamicPageTitle>
            </f:title>
            <!-- DynamicPage Header -->
            <f:header>
                  <f:DynamicPageHeader pinnable="true">
                        <f:content>
                              <layout:Grid defaultSpan="XL4 L6 M6 S12">
                                    <layout:VerticalLayout>
                                          <Label text="Name:"/>
                                          <Select id="slName" width="200px"
forceSelection="true" change="onSelectChange" items="{ path: '/ProductNames' }">
                                                <core:Item key="{key}"
text="{name}"/>
                                          </Select>
                                    </layout:VerticalLayout>
                                    <layout:VerticalLayout>
                                          <Label text="Category:"/>
                                          <Select id="slCategory"
width="200px" change="onSelectChange" forceSelection="true" items="{ path:
'/ProductCategories' }">
                                                <core:Item key="{key}"
text="{name}"/>
                                          </Select>
                                    </layout:VerticalLayout>
                                    <layout:VerticalLayout>
                                          <Label text="SupplierName:"/>
                                          <Select id="slSupplierName"
width="200px" change="onSelectChange" forceSelection="true" items="{ path:
'/ProductSuppliers' }">
                                                <core:Item key="{key}"
text="{name}"/>
                                          </Select>
                                    </layout:VerticalLayout>
                              </layout:Grid>
                        </f:content>
                  </f:DynamicPageHeader>
            </f:header>
      </f:DynamicPage>
```

```
</mvc:View>
```

6.1.2 Master.view.xml - add Dynamic Page Content and Footer

Add content agregation of the Dynamic Page containing Table control.

```
<f:DynamicPage>
…….
             </f:header>
            <!-- DynamicPage Content -->
            <f:content>
                  <Table id="idProductsTable" inset="false" items="{ path:
'/ProductCollection' }">
                        <columns>
                              <Column minScreenWidth="Tablet"
demandPopin="true">
                                    <Text text="Name"/>
                              </Column>
                              <Column minScreenWidth="Tablet"
demandPopin="true">
                                    <Text text="Category"/>
                              </Column>
                              <Column minScreenWidth="Tablet"
demandPopin="true">
                                    <Text text="SupplierName"/>
                              </Column>
                        </columns>
                        <items>
                              <ColumnListItem vAlign="Middle">
                                    <cells>
                                          <ObjectIdentifier title="{Name}"
text="{ProductId}"/>
                                          <Text text="{Category}"/>
                                          <Text text="{SupplierName}"/>
                                    </cells>
                              </ColumnListItem>
                        </items>
                  </Table>
            </f:content>
            <!-- DynamicPage Footer -->
            <f:footer>
                  <OverflowToolbar>
                        <ToolbarSpacer/>
                        <Button type="Accept" text="Accept"/>
                        <Button type="Reject" text="Reject"/>
                  </OverflowToolbar>
            </f:footer>
</f:DynamicPage>
```

7 MASTER CONTROLLER

7.1.1 Create webapp/controller/Master.controller.js [NEW]

```
sap.ui.define([
      "jquery.sap.global",
      "sap/ui/core/mvc/Controller",
      "sap/ui/model/json/JSONModel"
], function(jQuery, Controller, JSONModel) {
      "use strict";

      return Controller.extend("ui5confUI5Conf.controller.Master", {
            onInit: function() {
                  this.aKeys = ["Name", "Category", "SupplierName"];
                  this.oSelectName = this.getSelect("slName");
                  this.oSelectCategory = this.getSelect("slCategory");
                  this.oSelectSupplierName =
this.getSelect("slSupplierName");
            },

            onExit: function() {
                  this.aKeys = [];
                  this.aFilters = [];
            },

            onSelectChange: function() {
                  var aCurrentFilterValues = [];

      aCurrentFilterValues.push(this.getSelectedItemText(this.oSelectName));

      aCurrentFilterValues.push(this.getSelectedItemText(this.oSelectCategor
y));

      aCurrentFilterValues.push(this.getSelectedItemText(this.oSelectSupplie
rName));

                  this.filterTable(aCurrentFilterValues);
            },

            filterTable: function(aCurrentFilterValues) {

      this.getTableItems().filter(this.getFilters(aCurrentFilterValues));

      this.updateFilterCriterias(this.getFilterCriteria(aCurrentFilterValues
));
            },
            getModel: function() {
                  var oView = this.getView(),
                        oModel = oView.getModel();

                  return oModel;
            },
            updateFilterCriterias: function(aFilterCriterias) {
                  var oModel;

                  if (aFilterCriterias.length > 0) { /* We can`t use a
single label and change only the model data, */
                        oModel = this.getModel();
                        this.removeSnappedLabel(); /* because in case of
label with an empty text, */
                        this.addSnappedLabel(); /* a space for the snapped
content will be allocated and can lead to title misalignment */
                        oModel.setProperty("/Filter/text",
this.getFormattedSummaryText(aFilterCriterias));
                  } else {
                        this.removeSnappedLabel();
                  }
            },

            addSnappedLabel: function() {

      this.getPageTitle().addSnappedContent(this.getSnappedLabel());
            },

            removeSnappedLabel: function() {
                  this.getPageTitle().destroySnappedContent();
            },

            getFilters: function(aCurrentFilterValues) {
                  this.aFilters = [];

                  this.aFilters = this.aKeys.map(function(sCriteria, i) {
                        return new sap.ui.model.Filter(sCriteria,
sap.ui.model.FilterOperator.Contains, aCurrentFilterValues[i]);
                  });

                  return this.aFilters;
            },
            getFilterCriteria: function(aCurrentFilterValues) {
                  return this.aKeys.filter(function(el, i) {
                        if (aCurrentFilterValues[i] !== "") {
                              return el;
                        }
                  });
            },
            getFormattedSummaryText: function(aFilterCriterias) {
                  return "Filtered by: " + aFilterCriterias.join(", ");
            },

            getTable: function() {
                  return this.getView().byId("idProductsTable");
            },
            getTableItems: function() {
                  return this.getTable().getBinding("items");
            },
            getSelect: function(sId) {
                  return this.getView().byId(sId);
            },
            getSelectedItemText: function(oSelect) {
                  return oSelect.getSelectedItem() ?
oSelect.getSelectedItem().getKey() : "";
            },
            getPage: function() {
                  return this.getView().byId("dynamicPageId");
            },
            getPageTitle: function() {
                  return this.getPage().getTitle();
            },
            getSnappedLabel: function() {
                  return new sap.m.Label({
                        text: "{/Filter/text}"
                  });
            }
      });
});
```

8 VIEW CONTENT AND FLOATING FOOTER

8.1.1 webapp/controller/Master.controller.js

Add new function to the controller that will show and hide the floating footer.

```
            },
            onToggleFooter: function() {

      this.getPage().setShowFooter(!this.getPage().getShowFooter());
            }
```

8.1.2 webapp/controller/Master.view.xml

Attach the function to the view press event so the footer will be displayed on button press.

```
            <Button icon="sap-icon://post" text="Toggle Footer"
type="Transparent" press="onToggleFooter" />
```

9 CREATE MASTER-DETAIL-DETAIL PATTERN WITH FLEXIBLE COLUMN LAYOUT

9.1.1 webapp/view/App.view.xml

Now we will update tha App view so it contains the main Fiori 2.0 layout control for creation of complex patterns such as Master-Detail, Master-Detail-Detail – Flexible Column Layout. Than we will put the Master view in its first column aggregation.

Note: Important to know is that the main Fiori 2.0 controls belong to the new **sap.f** namespace.

```
<mvc:View controllerName="ui5confUI5Conf.controller.App"
      xmlns="sap.f"
      xmlns:mvc="sap.ui.core.mvc"
      displayBlock="true"
      height="100%">

      <FlexibleColumnLayout id="fcl">
            <beginColumnPages>
                  <mvc:XMLView id="beginView"
viewName="ui5confUI5Conf.view.Master"/>
            </beginColumnPages>
      </FlexibleColumnLayout>
</mvc:View>
```

9.1.2 webapp/manifest.json

Change root view back to App view.

```
      "sap.ui5": {
            "rootView": {
                  "viewName": "ui5confUI5Conf.view.App",
                  "type": "XML"
            },
            ….
      }
```

10 DETAIL VIEW

10.1.1 Create webapp/view/Detail.view.xml [NEW]

```
<mvc:View controllerName="ui5confUI5Conf.controller.Detail"
      xmlns:mvc="sap.ui.core.mvc"
      xmlns="sap.m"
      displayBlock="true"
      height="100%">
      <Button text="Test" press="handlePress" />
</mvc:View>
```

11 DETAIL CONTROLLER

11.1.1 Create webapp/view/Detail.controller.js [NEW]

```
sap.ui.define([
      "jquery.sap.global",
      "sap/ui/core/mvc/Controller"
], function(jQuery, Controller) {
      "use strict";

      return Controller.extend("ui5confUI5Conf.controller.Detail", {
            onInit: function() {},
            getRouter : function () {
                  return sap.ui.core.UIComponent.getRouterFor(this);
            },
            handlePress: function () {
                  this.getRouter().navTo("detailDetailName", {"detail-
detail-item": "item-inner1",
                        "detail-item": "item1"});
            }
      });
});
```

12 DETAIL-DETAIL VIEW

12.1.1 Create webapp/view/DetailDetail.view.xml [NEW]

```
<mvc:View controllerName="ui5confUI5Conf.controller.DetailDetail"
      xmlns:mvc="sap.ui.core.mvc"
      xmlns="sap.m"
      displayBlock="true"
      height="100%">
      <Text text="The end"/>
</mvc:View>
```

13 DETAIL-DETAIL CONTROLLER

13.1.1 Create webapp/view/DetailDetail.controller.js [NEW]

```
sap.ui.define([
      "jquery.sap.global",
      "sap/ui/core/mvc/Controller"
], function(jQuery, Controller) {
      "use strict";

      return Controller.extend("ui5confUI5Conf.controller.DetailDetail", {
            onInit: function() {}
      });
});
```

14 ROUTING

In this step, we will define the routing of our application based on Flexible Column Layout routes. In that way we will be able to navigate from Master to Detail and Detail-Detail views.

14.1.1 Modify webapp/manifest.json

```
      "sap.ui5": {
            …
            "resources": {
            …
            },
            "routing": {
                  "config": {
                        "routerClass": "sap.f.routing.Router",
                        "viewType": "XML",
                        "viewPath": "ui5confUI5Conf.view",
                        "controlId": "fcl",
                        "transition": "slide"
                  },
                  "routes": [{
                        "pattern": "",
                        "name": "masterName",
                        "target": [
                              "master"
                        ],
                        "layout": "OneColumn"
                  }, {
                        "pattern": "detail/{detail-item}",
                        "name": "detailName",
                        "target": [
                              "master",
                              "detail"
                        ],
                        "layout": "TwoColumnsMidExpanded"
                  }, {
                        "pattern": "detail/{detail-item}/detail-
detail/{detail-detail-item}",
                        "name": "detailDetailName",
                        "target": [
                              "master",
                              "detail",
                              "detailDetail"
                        ],
                        "layout": "ThreeColumnsEndExpanded"
                  }],
                  "targets": {
                        "master": {
                              "viewName": "Master",
                              "controlAggregation": "beginColumnPages"
                        },
                        "detail": {
                              "viewName": "Detail",
                              "controlAggregation": "midColumnPages"
                        },
                        "detailDetail": {
                              "viewName": "DetailDetail",
                              "controlAggregation": "endColumnPages"
                        }
                  }
            }
      }
```

14.1.2 Update webapp/Component.js

```
sap.ui.define([
	"sap/ui/core/UIComponent",
	"sap/ui/Device",
	"ui5confUI5Conf/model/models"
], function(UIComponent, Device, models) {
	"use strict";

	return UIComponent.extend("ui5confUI5Conf.Component", {

		metadata: {
			manifest: "json"
		},

		/**
		 * The component is initialized by UI5 automatically during the
startup of the app and calls the init method once.
		 * @public
		 * @override
		 */
		init: function() {
			// call the base component's init function
			UIComponent.prototype.init.apply(this, arguments);

			// set the device model
			this.setModel(models.createDeviceModel(), "device");

			//Init router
			this.getRouter().initialize();
		}
	});
});
```

14.1.3 Update webapp/controllers/Master.controller.js

```
…

            },
            handleItemPress: function (oEvent) {
                  var sId = oEvent.getParameter("id"),
                        oSelectedItem = sap.ui.getCore().byId(sId),
                        oModel = oSelectedItem.getModel(),
                        sPath = oSelectedItem.getBindingContextPath(),
                        oData = oModel.getProperty(sPath);

                  this.getRouter().navTo("detailName", {"detail-item":
oData.ProductId});
            },
            getRouter : function () {
                  return sap.ui.core.UIComponent.getRouterFor(this);
            }
      });
});
```

14.1.4 Update webapp/views/Master.view.xml

```
…
<ColumnListItem vAlign="Middle" type="Navigation" press="handleItemPress">
```

15 MASTER-DETAIL FINALIZATION – OBJECT PAGE

15.1.1 webapp/model/model2.json [NEW]

Create new model2.json in /models folder. Copy the json model content from the following url:

https://raw.githubusercontent.com/d3xter666/2017-ui5con-fiori/master/webapp/model/products_less.json

15.1.2 Update webapp/manifest.json

In this step we will add new model for the Object Detail view.

```
"models": {
                    "i18n": {
                          "type": "sap.ui.model.resource.ResourceModel",
                          "settings": {
                                "bundleName": "ui5confUI5Conf.i18n.i18n"
                          }
                    },
                    "": {
                          "type": "sap.ui.model.json.JSONModel",
                          "uri": "model/model.json"
                    },
                    "detailDetail": {
                          "type": "sap.ui.model.json.JSONModel",
                          "uri": "model/model2.json"
                    }
              },
```

15.1.3 Update webapp/view/Detail.view.xml

In this step we will enhance the Detail veiw so it shows the details of a selected object using Object Page control. Notice that we need to include **sap.uxap** namespace which owns all Object Page assets and controls.

```
<mvc:View controllerName="ui5confUI5Conf.controller.Detail"
	xmlns:mvc="sap.ui.core.mvc"
	xmlns="sap.m"
	xmlns:core="sap.ui.core"
	xmlns:uxap="sap.uxap"
	xmlns:layout="sap.ui.layout"
	displayBlock="true"
	height="100%">

	<uxap:ObjectPageLayout id="ObjectPageLayout"
showTitleInHeaderContent="true">
		<uxap:headerTitle>
			<uxap:ObjectPageHeader objectImageURI="sap-icon://product"
objectImageShape="Square" objectTitle="{detail>/Name}"
objectImageAlt="{detail>/Name}"
				objectSubtitle="{detail>/Category}"
isObjectIconAlwaysVisible="false" isObjectTitleAlwaysVisible="false"
				isObjectSubtitleAlwaysVisible="false">
				<uxap:actions>
					<uxap:ObjectPageHeaderActionButton icon="sap-
icon://pull-down" text="show section" type="Emphasized"/>
					<uxap:ObjectPageHeaderActionButton icon="sap-
icon://show" text="show state" type="Emphasized"/>
					<uxap:ObjectPageHeaderActionButton text="Toggle
Footer" hideIcon="true" hideText="false" type="Emphasized"
press="toggleFooter"/>
				</uxap:actions>
			</uxap:ObjectPageHeader>
		</uxap:headerTitle>
		<uxap:headerContent>
			<layout:VerticalLayout>
				<Text text="{detail>/ProductId}"/>
				<Text text="{detail>/SupplierName}"/>
			</layout:VerticalLayout>
			<layout:VerticalLayout>
				<Label text="Your product has been dispatched to:"/>
				<VBox height="63px">
					<Label text="Supply progress:"/>
					<ProgressIndicator percentValue="30"
displayValue="30%" showValue="true" state="None"/>
				</VBox>
			</layout:VerticalLayout>
			<layout:VerticalLayout>
				<Label text="San Jose, USA"/>
			</layout:VerticalLayout>
		</uxap:headerContent>
		<uxap:sections>
			<uxap:ObjectPageSection title="Title 1">
				<uxap:subSections>
					<uxap:ObjectPageSubSection>
						<uxap:blocks>
							<Text text="Product Description
goes here.."/>
						</uxap:blocks>
					</uxap:ObjectPageSubSection>
				</uxap:subSections>
			</uxap:ObjectPageSection>
			<uxap:ObjectPageSection title="Title 2">
				<uxap:subSections>
```

```
                                <uxap:ObjectPageSubSection>
                                      <uxap:blocks>
                                            <Table id="idProductsTable"
inset="false" items="{path: 'detailDetail>/ProductCollection'}">
                                                  <columns>
                                                        <Column
minScreenWidth="Tablet" demandPopin="true">
                                                              <Text
text="Name"/>
                                                        </Column>
                                                        <Column
minScreenWidth="Tablet" demandPopin="true">
                                                              <Text
text="Category"/>
                                                        </Column>
                                                        <Column
minScreenWidth="Tablet" demandPopin="true">
                                                              <Text
text="SupplierName"/>
                                                        </Column>
                                                  </columns>
                                                  <items>
                                                        <ColumnListItem
vAlign="Middle" type="Navigation" press="handlePress">
                                                              <cells>

      <ObjectIdentifier title="{detailDetail>Name}"
text="{detailDetail>ProductId}"/>
                                                                    <Text
text="{detailDetail>Category}"/>
                                                                    <Text
text="{detailDetail>SupplierName}"/>
                                                              </cells>
                                                        </ColumnListItem>
                                                  </items>
                                            </Table>
                                      </uxap:blocks>
                                </uxap:ObjectPageSubSection>
                          </uxap:subSections>
                    </uxap:ObjectPageSection>
             </uxap:sections>
             <uxap:footer>
                    <OverflowToolbar>
                          <ToolbarSpacer/>
                          <Button type="Accept" text="Accept"/>
                          <Button type="Reject" text="Reject"/>
                    </OverflowToolbar>
             </uxap:footer>
      </uxap:ObjectPageLayout>
</mvc:View>
```

15.1.4 Update webapp/controller/Detail.controller.js

```
sap.ui.define([
	"jquery.sap.global",
	"sap/ui/core/mvc/Controller",
	"sap/ui/model/json/JSONModel"
], function(jQuery, Controller, JSONModel) {
	"use strict";

	return Controller.extend("ui5confUI5Conf.controller.Detail", {
		onInit: function() {

	this.getRouter().attachRouteMatched(this._handleRouteChanged.bind(this));
		},
		_itemDetail: null,
		_handleRouteChanged: function(oEvent) {
			var oView = this.oView,
				oModel = this.oView.getModel(),
				oData = oModel.getData(),
				sProductId = oEvent.getParameter("arguments")["detail-
item"],
				oItemData =
oData["ProductCollection"].filter(function(oCurItem) {
					return oCurItem.ProductId === sProductId;
				})[0];

				this._itemDetail = sProductId;

				if (oItemData) {
					oView.setModel(new JSONModel(oItemData),
"detail");
				}
		},
		getRouter: function() {
			return sap.ui.core.UIComponent.getRouterFor(this);
		},
		toggleFooter: function (oEvent) {
			var oObjectPageLayout =
this.getView().byId("ObjectPageLayout");

	oObjectPageLayout.setShowFooter(!oObjectPageLayout.getShowFooter());
		},
		handlePress: function(oEvent) {
			var sId = oEvent.getParameter("id"),
				oSelectedItem = sap.ui.getCore().byId(sId),
				 // It's important to point that we're getting the
second model, not the default one
				oModel = oSelectedItem.getModel("detailDetail"),
				sPath = oSelectedItem.getBindingContextPath(),
				oData = oModel.getProperty(sPath);

			this.getRouter().navTo("detailDetailName", {
				"detail-detail-item": oData.ProductId,
				"detail-item": this._itemDetail
			});
		}
	});
});
```