HD3C14 – The IconTab Control

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Product and Focus**  HANA Platform/SAPUI5 | **MOTIVATION**  The IconTab control is one of the more useful and iconic SAPUI5 controls. This case also describes how to use components to modularize an application.  **PREREQUISITES**  HD3C06 – The Base Application | | **Target Audience**  Undergrduate/Graduate Beginner to Intermediate | | **Author**  Ross Hightower | | https://bgoerke.files.wordpress.com/2013/05/section1.png | | |  |

# IconTab Bar

The IconTab control is one of the more iconic SAPUI5 controls. The template for the control in XML is shown below:

|  |
| --- |
| <IconTabBar xmlns=*"sap.m"*  busy=*"false"*  busyIndicatorDelay=*"1000"*  visible=*"true"*  showSelection=*"true"*  expandable=*"true"*  expanded=*"true"*  selectedKey=*""*  upperCase=*"false"*  stretchContentHeight=*"false"*  applyContentPadding=*"true"*  backgroundDesign=*"Solid"*  select=*""*  expand=*""*>  <tooltip></tooltip> <!-- sap.ui.core.TooltipBase -->  <dependents></dependents> <!-- sap.ui.core.Control, since 1.19 -->  <items></items> <!-- sap.m.IconTab -->  <content></content> <!-- sap.ui.core.Control -->  </IconTabBar> |

The items aggregation is used with the IconTabFilter objects to create the tabbed content. The template for the IconTabFilter is shown below.

|  |
| --- |
| <IconTabFilter xmlns=*"sap.m"*  text=*""*  enabled=*"true"*  textDirection=*"Inherit"*  key=*""*  count=*""*  showAll=*"false"*  icon=*""*  iconColor=*"Default"*  iconDensityAware=*"true"*  visible=*"true"*  design=*"Vertical"*>  <tooltip></tooltip> <!-- sap.ui.core.TooltipBase -->  <dependents></dependents> <!-- sap.ui.core.Control, since 1.19 -->  <content></content> <!-- sap.ui.core.Control, since 1.15.0 -->  </IconTabFilter> |

# Create the IconTab View

We’ve used the IconTab control in previous cases. We will use it in a slightly different way here. The actual content of the page is a Table control which is inserted below the IconTab control. The IconTab control is used as a mechanism to allow the user to filter the data in the Table in different ways.

### Add the View to the App Navigation

Add a new object for the Lists view to the views.json file. You can find and icon [here](https://openui5.hana.ondemand.com/iconExplorer.html). Also, add a route to the Component.js file.

### IconTab.view.xml

Create a file called **IconTab.view.xml** in the **view** package. Add the code shown below to the file:

|  |
| --- |
| <mvc:View  controllerName="ui5.controller.IconTab"  xmlns:l="sap.ui.layout"  xmlns:mvc="sap.ui.core.mvc"  xmlns="sap.m">  <Page title="Start Coding IconTabs!"  showNavButton = "true"  navButtonPress = "handleNavButtonPress">  <IconTabBar  expanded="false"  id="idIconTabBar"  select="handleIconTabBarSelect">  <items>  <IconTabFilter  icon="sap-icon://begin"  iconColor="Positive"  design="Horizontal"  text="EBI"  key="EBI" />  <IconTabFilter  icon="sap-icon://compare"  iconColor="Critical"  design="Horizontal"  text="TOU"  key="TOU" />  <IconTabFilter  icon="sap-icon://inventory"  iconColor="Negative"  design="Horizontal"  text="TRE"  key="TRE" />  <IconTabFilter  icon="sap-icon://inventory"  iconColor="Negative"  design="Horizontal"  text="All"  key="All" />  </items>  </IconTabBar>  </Page>  </mvc:View> |

Listing

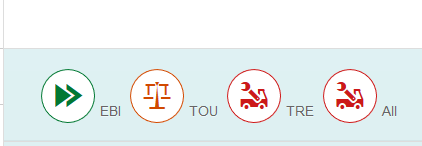
### IconTab.controller.js

Create a file called **IconTab.controller.js** in the **view** package. Add the code shown below to the file:

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller"  ], function(Controller) {  "use strict";  return Controller.extend("ui5.controller.IconTab", {  onInit: function() {  this.router = sap.ui.core.UIComponent.getRouterFor(this);  },  handleNavButtonPress: function() {  this.router.navTo("Master", {  from: "IconTab"  });  }  });  }); |

Listing

Run the application.



Not too exciting but you can see the basic structure.

## Add Some Content

Now let’s add some content to those panes.

### IconTab.view.xml

Add the code shown below immediately before the </Page> tag.

|  |  |
| --- | --- |
| <Table id="idProductsTable"  inset="false"  items="{gbi>/ProductsCollection}">    <columns>  <Column>  <header>  <Text text="Product" />  </header>  </Column>  <Column  minScreenWidth="Tablet">  <header>  <Text text="Division" />  </header>  </Column>  <Column  minScreenWidth="Tablet"  demandPopin='true'>  <header>  <Text text="Product Category" />  </header>  </Column>  <Column  minScreenWidth="Tablet"  demandPopin='true'>  <header>  <Text text="Price" />  </header>  </Column>  <Column>  <header>  <Text text="Inventory" />  </header>  </Column>    </columns>  <items>  <ColumnListItem>  <cells>  <ObjectIdentifier  title="{gbi>PRODUCT\_NAME}"  text="{gbi>PRODUCT}" />  <Text  text="{gbi>DIVISION}" />  <Text  text="{gbi>PRODUCT\_CATEGORY}" />  <ObjectNumber  number="{gbi>PRICE}"  unit="{gbi>CURRENCY}" />  <ObjectNumber  number="{gbi>INVENTORY}" />  </cells>  </ColumnListItem>  </items>  </Table> |  |

Listing

If you did the case on Tables you will recognize this code.

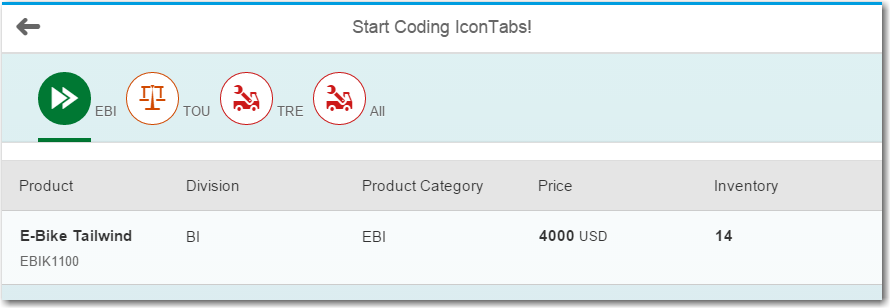
### IconTab.controller.js

Now add the function below to the IconTab.controller.js:

|  |
| --- |
| handleIconTabBarSelect : function (oEvent) {  var oTable = this.byId("idProductsTable");  var oBinding = oTable.getBinding("items");  sKey = oEvent.getParameter("selectedKey");  if (sKey === "EBI") {  oFilter = new sap.ui.model.Filter("PRODUCT\_CATEGORY", "EQ", "EBI");  oBinding.filter([oFilter]);  } else if (sKey === "TOU") {  oFilter = new sap.ui.model.Filter("PRODUCT\_CATEGORY", "EQ", "TOU");  oBinding.filter([oFilter]);  } else if (sKey === "TRE") {  oFilter = new sap.ui.model.Filter("PRODUCT\_CATEGORY", "EQ", "TRE");  oBinding.filter([oFilter]);  } else {  oBinding.filter([]);  }  } |

Listing

This code retrieves a reference to the Table control and then retrieves the Table’s items binding. It then retrieves the key of the IconTab that was clicked and creates a filter based on the PRODUCT\_CATEGORY field using the key. When the binding is updated with the filter, the table shows only rows that match the filter.



## Componentize the Table

Very often fragments of code or even whole views are reused in multiple parts of an application. For example, the Table control used in this case is identical to the Table used in the Tables case. In this section we will create a Table component that is loaded dynamically when required.

Create a new package in the **view** package called **components**. Inside that folder create four files: **Component.js**, **TableComponent.view.xml** and **TableComponent.controller.js.**



### Component.js

The Component.js file is used to describe the component.

|  |
| --- |
| jQuery.sap.declare("ui5.view.components.Component");  sap.ui.core.UIComponent.extend("ui5.view.components.Component", {  metadata : {  publicMethods : [  "getTable"  ],  dependencies : {  libs : [  "sap.m",  "sap.ui.layout"  ]  },  config : {  sample : {  files : [  "TableComponent.view.xml",  "TableComponent.controller.js"  ]  }  }  },    getTable : function () {  return this.\_rootView.getContent()[0];  }  });  ui5.view.components.Component.prototype.createContent = function () {  this.\_rootView = sap.ui.xmlview({ viewName : "ui5.view.components.TableComponent" });  return this.\_rootView;  }; |

Listing

The first line of this code declares the component. To create a component the UIComponent object is extended by specifying metadata that defines this extension. The metadata declares a public method getTable (defined later), declares required libraries and declares the required files. The last line of the code defines the visible content of the component as the TableComponent.view.xml view.

### TableComponent.view.xml

Insert the code shown below. This is the same code used for the table above except that the references the controller is updated to point to the component files.

|  |
| --- |
| <mvc:View controllerName="ui5.view.components.TableComponent"  xmlns:l="sap.ui.layout" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m">  <Table id="idProductsTable" inset="false" items="{gbi>/ProductsCollection}">  <columns>  <Column>  <header>  <Text text="Product" />  </header>  </Column>  <Column minScreenWidth="Tablet">  <header>  <Text text="Division" />  </header>  </Column>  <Column minScreenWidth="Tablet" demandPopin='true'>  <header>  <Text text="Product Category" />  </header>  </Column>  <Column minScreenWidth="Tablet" demandPopin='true'>  <header>  <Text text="Price" />  </header>  </Column>  <Column>  <header>  <Text text="Inventory" />  </header>  </Column>  </columns>  <items>  <ColumnListItem>  <cells>  <ObjectIdentifier title="{gbi>PRODUCT\_NAME}" text="{gbi>PRODUCT}" />  <Text text="{gbi>DIVISION}" />  <Text text="{gbi>PRODUCT\_CATEGORY}" />  <ObjectNumber number="{gbi>PRICE}" unit="{gbi>CURRENCY}" />  <ObjectNumber number="{gbi>INVENTORY}" />  </cells>  </ColumnListItem>  </items>  </Table>  </mvc:View> |

Listing

### TableComponent.controller.js

Paste the code shown below.

|  |
| --- |
| sap.ui.controller("ui5.view.components.TableComponent", {    }); |

Listing

### Modify the Application to use the Table Component

Now we have to modify the IconTab view to use the table component.

### IconTab.view.xml

**Delete the Table control** which you inserted below the IconTab control before.

### IconTab.controller.js

Add the code below to the IconTab.controller.js file:

|  |
| --- |
| onInit: **function** () {  this.router = sap.ui.core.UIComponent.getRouterFor(this);  // reuse table sample component  **var** oComp = sap.ui.getCore().createComponent({  name : 'ui5.view.components'  });  oComp.setModel(**this**.getView().getModel());  **this**.\_oTable = oComp.getTable();  **this**.getView().byId("idIconTabBar").insertContent(**this**.\_oTable);  // update table  **this**.\_oTable.setHeaderText(**null**);  **this**.\_oTable.setShowSeparators("Inner");  } |

Listing

This code creates component based on the TableComponent. It then retrieves the model (gbi.json) from the view and assigns it to the component. The component is then inserted into the content of the IconTab view.

Replace the handleIconTabBarSelect function with the code below:

|  |
| --- |
| handleIconTabBarSelect : function (oEvent) {  var oTable = oEvent.getSource().getContent()[0];  var oBinding = oTable.getBinding("items");  sKey = oEvent.getParameter("selectedKey");  if (sKey === "EBI") {  oFilter = new sap.ui.model.Filter("PRODUCT\_CATEGORY", "EQ", "EBI");  oBinding.filter([oFilter]);  } else if (sKey === "TOU") {  oFilter = new sap.ui.model.Filter("PRODUCT\_CATEGORY", "EQ", "TOU");  oBinding.filter([oFilter]);  } else if (sKey === "TRE") {  oFilter = new sap.ui.model.Filter("PRODUCT\_CATEGORY", "EQ", "TRE");  oBinding.filter([oFilter]);  } else {  oBinding.filter([]);  }  } |

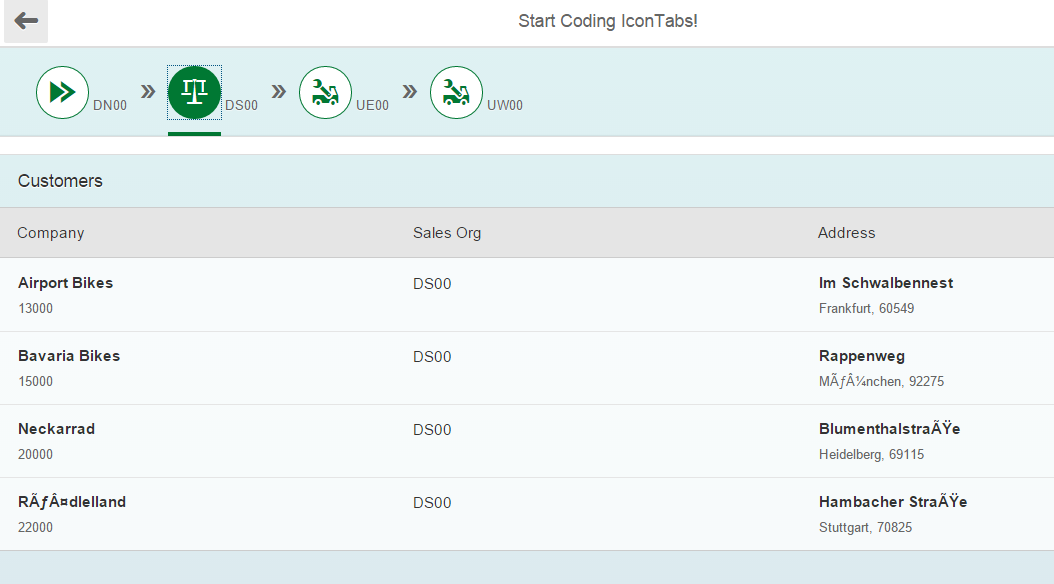
Listing

The change in this code is because the table no longer has an id so we have to reference it using the getContent function.

Now refresh the application. The table should still work and now you have a table that can be reused in multiple places.

# Exercise

Create a view using an IconTab bar used to filter the customer table from the Exercise in the Tables case. Filter on the SALES\_ORG field.



### Adding the View to the Application

To add the View to the application:

1. Add an object to the ExerciseCollection in the views.json file. Remember that the info property specifies the name of the view.
2. Add the route to Component.js.