HD2C03 – Data Binding in SAPUI5

|  |  |
| --- | --- |
| **Product and Focus**  HANA Platform/SAPUI5 | **MOTIVATION**  This case uses a simple application to explain data binding in SAPUI5 development.  **PREREQUISITES**  None |
| **Target Audience**  Undergraduate/Graduate Beginner to Intermediate |
| **Author**  Ross Hightower |
| https://bgoerke.files.wordpress.com/2013/05/section1.png | |

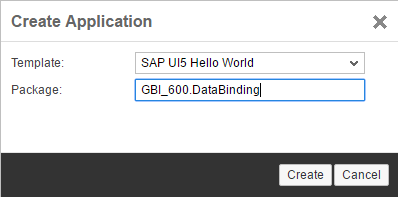
# Data Binding in SAPUI5

The link between the data in the model and controls on the view is accomplished via data binding. This allows the model to manage the flow of data between the view and model. Create the Basic Hello World App

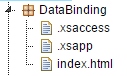
Open a browser and navigate to the URL provided by your UCC to access the WDW.

Open the **Editor**. A package has been created for you with a name equal to GBI\_### where ### is the last three digits of your user id. Right-click your package and select **Create Application.**

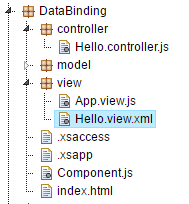
Select the **SAP UI5 Hello World** template and add .**DataBinding** to the package name. This will create a package called HelloWorld and create the application in that package.



When the application is created, three files are automatically created in the **DataBinding** package: **index.html**, **.xsaccess**, and **.xsapp.**



Use the context menu (accessed by right-clicking a package) to create three new packages: **view**, **controller** and **model** and four new files: **Hello.controller.js**, **Hello.view.xml**, **App.view.js** and **Component.js.**



That completes the basic structure.

## Add the Code

Now we will add the code.

### index.html

Replace the code in the index.html file with the code shown below.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="X-UA-Compatible" content="IE=edge" />  <title>SAPUI5 in 20 Seconds</title>  <!-- 1.) Load SAPUI5 (from local server), select theme and control library -->  <script id="sap-ui-bootstrap"  type="text/javascript"  src="/sap/ui5/1/resources/sap-ui-core.js"  data-sap-ui-theme="sap\_bluecrystal"  data-sap-ui-libs="sap.m"  data-sap-ui-xx-bindingSyntax="complex"  data-sap-ui-resourceroots='{ "ui5": "./" }'></script>  <script>  sap.ui.getCore().attachInit(function() {  new sap.m.Shell("shell",{  app : new sap.ui.core.ComponentContainer({  height : "100%",  name : "ui5"  })  }).placeAt("uiArea"); });  </script>  </head>  <body class="sapUiBody">  <!-- This is where you place the UI5 button -->  <div id="uiArea"></div>    </body>  </html> |

Listing

### Component.js

Insert the following code into the Component.js file.

|  |
| --- |
| jQuery.sap.declare("ui5.Component");  sap.ui.core.UIComponent.extend("ui5.Component", {  metadata: {  },    init: function() {    //call createContent  sap.ui.core.UIComponent.prototype.init.apply(this, arguments);    },    createContent : function() {  // create root view  var oView = sap.ui.view({  id : "app",  viewName : "ui5.view.App",  type : "JS",  viewData : { component : this }  });  return oView;  }  }); |

Listing

### App.view.js

Add the code below to the App.view.js file.

|  |
| --- |
| sap.ui.jsview("ui5.view.App", {  createContent: function () {    // create app  this.app = new sap.m.App();    // load the master page using an XML view  var hello = sap.ui.xmlview("Hello", "ui5.view.Hello");  this.app.addPage(hello, true);    return this.app;  }  }); |

Listing

### Hello.view.xml

Insert the code below into Hello.view.xml.

|  |
| --- |
| <mvc:View controllerName="ui5.controller.Hello"  xmlns="sap.m" xmlns:mvc="sap.ui.core.mvc">  <StandardTile id="tileId" press="doIt" icon="sap-icon://world"  title="Hello World"  info="English" />    </mvc:View> |

Listing

This code implements a single StandardTile.

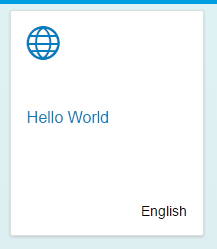
### Hello.controller.js

Finally, insert the code shown below in the Hello.controller.js file. This implements the doIt function. When the button is clicked or tapped the doIt function will show a toast message that includes the Button’s id. The oEvent parameter is an object that contains information about the event that invoked the function. The function oEvent.getSource() returns the control that initiated the event and getId() returns the control’s id.

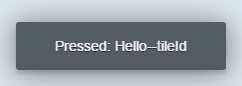
|  |
| --- |
| sap.ui.controller("ui5.controller.Hello", {  doIt : function(oEvent) {    jQuery.sap.require("sap.m.MessageToast");  sap.m.MessageToast.show("Pressed: " + oEvent.getSource().getId());    }  }); |

Listing

Save all your files and then run the application by selecting the index.html file and clicking the run icon.



If you click the tile, the toast message is displayed. The id is constructed from the name of the view and the id assigned to the Button control.



# Add a Model

We’ve demonstrated the overall architecture of SAPUI5 applications and the functions of views and controllers. In this section we’ll demonstrate the use of models.

### HelloModel.json

Create a new file called **HelloModel.json** in the model package and add the following JSON code.

|  |
| --- |
| {    "Language" : "English",  "Greeting" : "Hello World!"    } |

Listing

This is standard JSON (JavaScript Object Notation) code. It has one collection, HelloCollection, which is an array with two objects. JavaScript arrays are zero based so the first object has an index of 0.

### Component.js

Add the model definition to the Component.js file below the definition of oView.

|  |
| --- |
| var oModel = new sap.ui.model.json.JSONModel("model/HelloModel.json");  this.setModel(oModel, "hello"); |

Listing

This creates a JSON model that references the HelloModel.json file. It then adds it to the component and names the model hello.

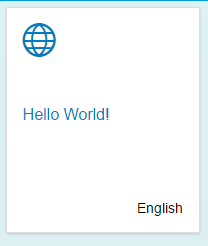
### Hello.view.xml

Replace the tile control with a code shown below.

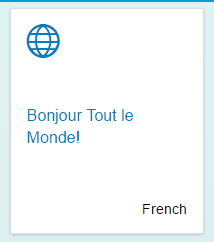
|  |
| --- |
| <StandardTile id="tileId" press="doIt" icon="sap-icon://world"  title="{hello>/Greeting}"  info="{hello>/Language}" /> |

Listing

If you run the application again, you will see that is hasn’t changed. That’s because we used the same values for the properties that we used for the properties before.



However, if you change the values in the HelloModel.json file and refresh the application, the changes are reflected in the tile.



## A More Complex Model

Now, let’s increase the complexity of the data in the model.

### HelloModel.json

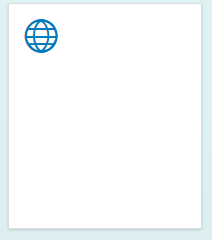
Replace the code with the code shown below.

|  |
| --- |
| {  "HelloCollection": [  {  "Language" : "English",  "Greeting" : "Hello World!"  },  {  "Language" : "français",  "Greeting" : "Bonjour le monde!"  }  ]  } |

Listing 9

Now, there is one property called HelloCollection and its value is an array (enclosed in the [ ]) of objects instead of just one. This is the same format as data returned by a RESTful web service in JSON format. Collections of records are represented by arrays of objects.

If you refresh the application now, you will see this.



The problem is SAPUI5 doesn’t know which object to bind to so we have to tell it which one.

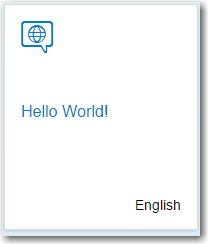
### Hello.view.xml

Update the highlighted portions of the code as shown below.

|  |
| --- |
| <mvc:View controllerName="ui5.controller.Hello"  xmlns="sap.m" xmlns:mvc="sap.ui.core.mvc">  <StandardTile id="tileId" press="doIt" icon="sap-icon://world"  title="{hello>/HelloCollection/0/Greeting}"  info="{hello>/HelloCollection/0/Language}" />  </mvc:View> |

Listing

In binding syntax in SAPUI5 represents array indices with slashes / instead of square brackets as in JavaScript. So the first item in the HelloCollection array in JavaScript would be HelloCollection[0] whereas with SAPUI5 binding it would be HelloCollection/0/. If you run the application now, the tile is bound to the properties in the first object.



Change the binding to **hello>/HelloCollection/1** and you will see this:



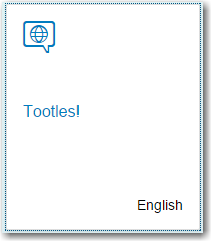
## An Even More Complex Model

Now, replace the code in the JSON file with this code:

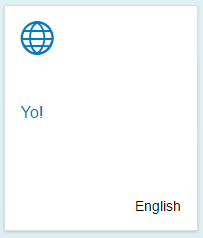
|  |
| --- |
| {  "HelloCollection": [  {  "Language" : "English",  "Greeting" : "Hello World!",  "Variations" : [  {  "Language" : "English",  "Greeting" : "Tootles!"  },  {  "Language" : "English",  "Greeting" : "Yo!"  }  ]  },  {  "Language" : "français",  "Greeting" : "Bonjour le monde!"  }  ]  } |

Listing

We still have the HelloCollection array but now we’ve added a property called Variations and it’s an array as well. In JavaScript, the English variation of the first object in HelloCollection would be HelloCollection[0].Variations[0]. What binding path would you use to bind the tile to the first English variation?



How about the second English variation?



## Aggregation Binding

The previous example bound a single object to a control. Some controls have aggregations that allow you to bind multiple objects to the controls. Examples are the TileContainer, List and Table controls. Let’s see how that works.

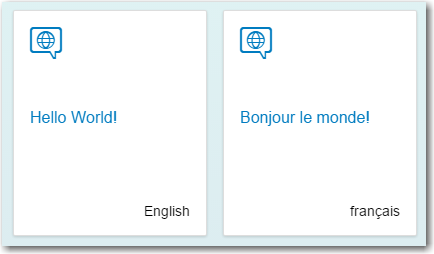
### Hello.view.xml

Replace the StandardTile control with the code shown below.

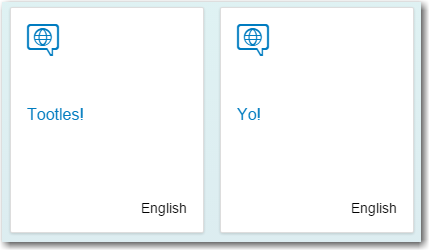
|  |
| --- |
| <TileContainer id="caseTiles" height="50%" tiles="{hello>/HelloCollection}">  <StandardTile id="tileId" press="doIt" icon="sap-icon://world"  title="{hello>Greeting}"  info="{hello>Language}" />  </TileContainer> |

Listing

The TileContainer control has an aggregation called tiles. We can bind the HelloCollection array of the hello model to that aggregation in the view code. Then we supply the StandardTile as a template for depicting each object in the HelloCollection array. SAPUI5 will create a StandardTile for each object in the HelloCollection array.



How would you bind the TileContainer to the English variations?



Managing data binding is one of the more challenging aspects of SAPUI5 to master but hopefully this has given you the beginnings of a deeper understanding.