HD3C04 – Data Binding

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

This case borrows heavily from the tutorials available on the SAPUI5 SDK.  
[https://sapui5.netweaver.ondemand.com/sdk/#docs/guide/8b49fc198bf04b2d9800fc37fecbb218.html](https://sapui5.netweaver.ondemand.com/sdk/%23docs/guide/8b49fc198bf04b2d9800fc37fecbb218.html)

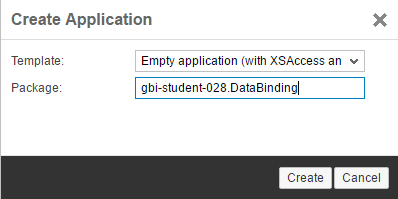
# Data Binding in SAPUI5

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

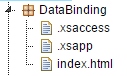
Open a browser and navigate to the URL provided by your UCC to access the WDW.

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.**

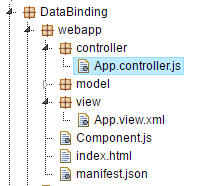
Select the **SAP UI5 Hello World** template and add .**DataBinding** to the package name. This will create a package called HelloWorld and create the application in that package.



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



Create the structure shown below. Remember to cut and paste the index.html file into the webapp package.



### 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=" https://openui5.hana.ondemand.com/1.42.6/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 1

### Component.js

Insert the following code into the Component.js file.

|  |
| --- |
| sap.ui.define([  "sap/ui/core/UIComponent",  ], function (UIComponent) {  "use strict";  return UIComponent.extend("ui5.Component", {  metadata : {  manifest: "json"  },  init : function () {  // call the init function of the parent  UIComponent.prototype.init.apply(this, arguments);  }  });  }); |

### manifest.json

Insert the following code into the manifest.json file.

|  |
| --- |
| {  "\_version": "1.1.0",  "sap.app": {  "\_version": "1.1.0",  "id": "ui5",  "type": "application",  "i18n": "i18n/i18n.properties",  "title": "Data Binding",  "description": "Data Binding Exercise",  "applicationVersion": {  "version": "1.0.0"  },  "dataSources": {  "helloLocal": {  "uri": "model/HelloModel.json",  "type": "JSON"  }  }  },  "sap.ui": {  "\_version": "1.1.0",  "technology": "UI5",  "deviceTypes": {  "desktop": true,  "tablet": true,  "phone": true  },  "supportedThemes": [  "sap\_bluecrystal"  ]  },  "sap.ui5": {  "\_version": "1.1.0",  "rootView": "ui5.view.App",  "dependencies": {  "minUI5Version": "1.30",  "libs": {  "sap.m": {}  }  },  "models": {  }  }  } |

### App.view.xml

Insert the code below into App.view.xml.

|  |
| --- |
| <mvc:View controllerName="ui5.controller.App"  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 4

This code implements a single StandardTile control.

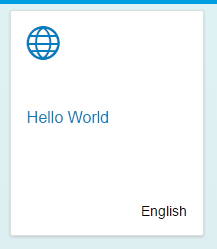
### App.controller.js

Finally, insert the code shown below in the App.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.

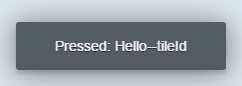
|  |
| --- |
| sap.ui.controller("ui5.controller.App", {  doIt : function(oEvent) {    jQuery.sap.require("sap.m.MessageToast");  sap.m.MessageToast.show("Pressed: " + oEvent.getSource().getId());    }  }); |

Listing 5

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 6

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.

### manifest.json

Add the model to the app by updating the manifest.json file as shown.

|  |
| --- |
| {  "\_version": "1.1.0",  "sap.app": {  "\_version": "1.1.0",  "id": "ui5",  "type": "application",  "i18n": "i18n/i18n.properties",  "title": "Data Binding",  "description": "Data Binding Exercise",  "applicationVersion": {  "version": "1.0.0"  },  "dataSources": {  "helloLocal": {  "uri": "model/HelloModel.json",  "type": "JSON"  }  }  },  "sap.ui": {  "\_version": "1.1.0",  "technology": "UI5",  "deviceTypes": {  "desktop": true,  "tablet": true,  "phone": true  },  "supportedThemes": [  "sap\_bluecrystal"  ]  },  "sap.ui5": {  "\_version": "1.1.0",  "rootView": "ui5.view.App",  "dependencies": {  "minUI5Version": "1.30",  "libs": {  "sap.m": {}  }  },  "models": {  **"hello": {**  **"type": "sap.ui.model.json.JSONModel",**  **"uri": "model/HelloModel.json"**  **}**  }  }  } |

This additional code defines a JSON model named hello that loads the HelloModel.json file

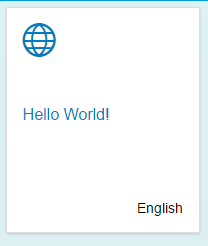
### App.view.xml

Next, we need to bind the attributes of the tile in the App view to the properties in the model. Update the App.view.xml file as shown.

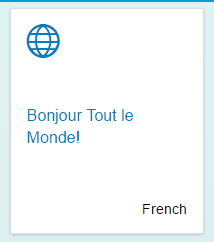
|  |
| --- |
| <mvc:View controllerName="ui5.controller.App"  xmlns="sap.m" xmlns:mvc="sap.ui.core.mvc">  <StandardTile id="tileId" press="doIt" icon="sap-icon://world"  title="**{hello>/Greeting}**"  info="**{hello>/Language}**" />  </mvc:View> |

The curly brackets ({}) indicate to UI5 that the value is bound to a property in a model. In the code hello>/Greeting the hello> indicates the name of the model as defined in the manifest.json file. The /Greeting is the path in the model’s data to the value to bind. The / is the root of the model (data models are hierarchical so the / indicates the top of the hierarchy. This will become clearer later in the case) and Greeting is the property to bind.

If you run the application, it looks the same as before except the values for the title and info attributes are being loaded from the model. 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

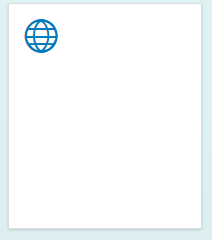
Replace the code with the code shown below.

|  |
| --- |
| {  "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.

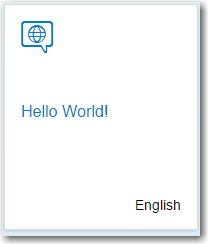
### App.view.xml

Update the highlighted portions of the code as shown below.

|  |
| --- |
| <mvc:View controllerName="ui5.controller.App"  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 10

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:



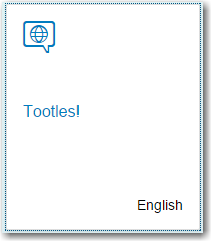
## An Even More Complex Model

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

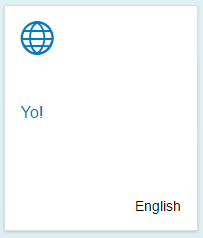
|  |
| --- |
| {  "HelloCollection": [  {  "Language" : "English",  "Greeting" : "Hello World!",  "Variations" : [  {  "Language" : "English",  "Greeting" : "Tootles!"  },  {  "Language" : "English",  "Greeting" : "Yo!"  }  ]  },  {  "Language" : "français",  "Greeting" : "Bonjour le monde!"  }  ]  }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

Listing 11

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.

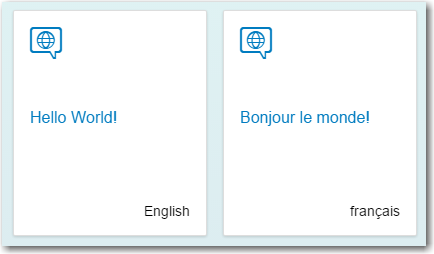
### App.view.xml

Replace the StandardTile control with the code shown below.

|  |
| --- |
| <mvc:View controllerName="ui5.controller.App" **height="100%"**  xmlns="sap.m" xmlns:mvc="sap.ui.core.mvc">  **<TileContainer id="caseTiles" height="50%" tiles="{hello>/HelloCollection}">**  **<StandardTile id="tileId" press="doIt" icon="sap-icon://world"**  **title="{hello>Greeting}"**  **info="{hello>Language}" />**  **</TileContainer>**  </mvc:View> |

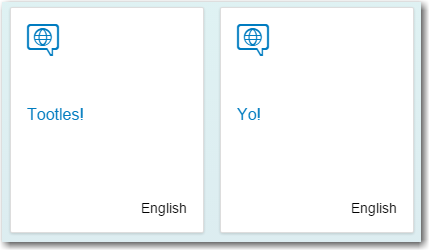
Listing 12

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.



# Exercise

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.