HD3C05 – Routing

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

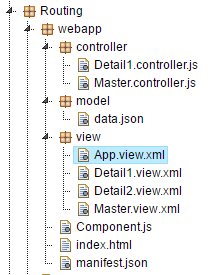
This case borrows heavily from the tutorials available on the SAPUI5 SDK.  
[https://sapui5.netweaver.ondemand.com/sdk/#docs/guide/8b49fc198bf04b2d9800fc37fecbb218.html](https://sapui5.netweaver.ondemand.com/sdk/%23docs/guide/8b49fc198bf04b2d9800fc37fecbb218.html)

# Routing

Routing is the process most web development frameworks use to navigate between views or pages. Routing, while initially it may seem more complex, imposes a structure on navigation that makes managing navigation in large applications much easier.

## Create the Application

Create a package called Routing and then create the following application structure.



### Index.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="X-UA-Compatible" content="IE=edge" />  <title>Routing</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

### manifest.json

|  |
| --- |
| {  "\_version": "1.1.0",  "sap.app": {  "\_version": "1.1.0",  "id": "ui5",  "type": "application",  "i18n": "i18n/i18n.properties",  "title": "{{appTitle}}",  "description": "{{appDescription}}",  "applicationVersion": {  "version": "1.0.0"  },  "ach": "CA-UI5-FST",  "dataSources": {  "employeeRemote": {  "uri": "/here/goes/your/serviceUrl/",  "type": "OData",  "settings": {  "odataVersion": "2.0",  "localUri" : "localService/metadata.xml"  }  }  }  },  "sap.ui": {  "\_version": "1.1.0",  "technology": "UI5",  "deviceTypes": {  "desktop": true,  "tablet": true,  "phone": true  },  "supportedThemes": ["sap\_belize"]  },  "sap.ui5": {  "\_version": "1.1.0",  "rootView": "ui5.view.App",  "dependencies": {  "minUI5Version": "1.30",  "libs": {  "sap.m": { }  }  },  "models": {    }  }  } |

Listing

### Component.js

|  |
| --- |
| 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);  }  });  }); |

Listing

### App.view.xml

|  |
| --- |
| <mvc:View  xmlns="sap.m"  xmlns:mvc="sap.ui.core.mvc"  displayBlock="true">  <App class="myAppDemoWT" id="app"/>  </mvc:View> |

Listing

### Master.view.xml

|  |
| --- |
| <core:View xmlns:core="sap.ui.core"  xmlns:mvc="sap.ui.core.mvc"  xmlns="sap.m" controllerName="ui5.controller.Master"  xmlns:html="http://www.w3.org/1999/xhtml">  <Page title="Master">  <content>    </content>  </Page>  </core:View> |

Listing

### Master.controller.js

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller"  ], function (Controller) {  "use strict";  return Controller.extend("ui5.controller.Master", {    });  }); |

Listing

## Add a Route

If you run the application, you will see a blank page. The reason is that the rootView property in the manifest.json tells UI5 to load the App.view.xml file when the application loads but the App view doesn’t contain any content. What we want to do is load the Master view when the application loads. One way to do that is to change the rootView property but that will not help because we wouldn’t have a way to navigate from the Master view to other views. Instead, we will configure routing information that will tell UI5 to load the Master view into the App view when the application loads. The App view becomes a container used to navigate among views. This is called a default route. Update the manifest.json file as shown below.

|  |
| --- |
| {  "\_version": "1.1.0",  "sap.app": {  "\_version": "1.1.0",  "id": "ui5",  "type": "application",  "i18n": "i18n/i18n.properties",  "title": "Routing",  "description": "Routing Exercise",  "applicationVersion": {  "version": "1.0.0"  }  },  "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": {}  }  },  **"routing": {**  **"config": {**  **"routerClass": "sap.m.routing.Router",**  **"viewType": "XML",**  **"viewPath": "ui5.view",**  **"controlId": "app",**  **"controlAggregation": "pages"**  **},**  **"routes": [**  **{**  **"pattern": "",**  **"name": "Master",**  **"target": "Master"**  **}**  **],**  **"targets": {**  **"Master": {**  **"viewName": "Master"**  **}**  **}**  **}**  }  } |

Listing

The routing configuration consists of the config section which configures default parameters for all routes, the routes section which include the routes and the targets section which contains the names of the view files to load for the various routes.

#### config

**viewType**: the code format used to create views XML, JSON, JS or HTML

**viewPath**: the location of view files

**controlId**: the id of the app control used to display views. This is defined in the App view.

**controlAggregation**: the aggregation in the app control to which views are assigned. The App control has one aggregation called pages because it is a full screen application.

#### routes

**pattern:** the pattern shown in the URL when the route is invoked. An empty pattern indicates the default route.

**name:** the name of the route is used to identify the route in the application code.

**target:** the target from the target section associated with the route.

#### targets

The name of the target (e.g. Master) is associated with the target property in the route section. The viewName property is the name of the view file (e.g. Master.view.xml).

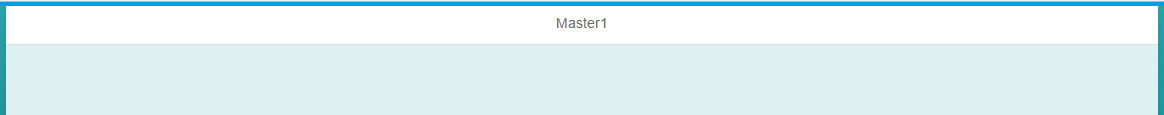
#### Component.js

The router is initialized in the Component.js file by adding the line of code shown below.

|  |
| --- |
| 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);    **this.getRouter().initialize();**  }  });  }); |

Listing

Now when you run the app, the Master view is loaded.



## Add the Detail1 View

Add the code shown below to the Detail1.view.xml file.

### Detail1.view.xml

|  |
| --- |
| <core:View xmlns:core="sap.ui.core"  xmlns:mvc="sap.ui.core.mvc"  xmlns="sap.m" controllerName="ui5.controller.Detail1"  xmlns:html="http://www.w3.org/1999/xhtml">  <Page title="Detail 1">  <content>  </content>  </Page>  </core:View> |

Listing

### Detail1.controller.js

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller"  ], function (Controller) {  "use strict";  return Controller.extend("ui5.controller.Detail1", {    });  }); |

Listing

### Master.view.xml

|  |
| --- |
| <core:View xmlns:core="sap.ui.core"  xmlns:mvc="sap.ui.core.mvc"  xmlns="sap.m" controllerName="ui5.controller.Master"  xmlns:html="http://www.w3.org/1999/xhtml">  <Page title="Master1">  <content>  **<Button text="Go to Detail 1" press="go"/>**  </content>  </Page>  </core:View> |

Listing

### Master.controller.js

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller"  ], function (Controller) {  "use strict";  return Controller.extend("ui5.controller.Master", {  **go: function () {**  **var oRouter = sap.ui.core.UIComponent.getRouterFor(this);**  **oRouter.navTo("Detail1");**  **}**  });  }); |

Listing

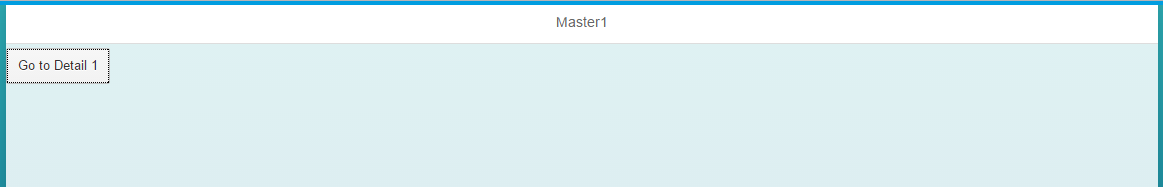
### manifest.json

Update the routes in the manifest.json as shown below.

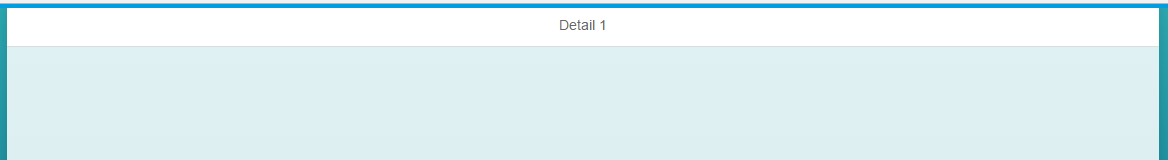
|  |
| --- |
| "routes": [  {  "pattern": "",  "name": "Master",  "target": "Master"  **},**  **{**  **"pattern": "Detail1",**  **"name": "Detail1",**  **"target": "Detail1"**  **}**  ],  "targets": {  "Master": {  "viewName": "Master"  }**,**  **"Detail1": {**  **"viewName": "Detail1"**  **}**  } |

Listing

Now when you run the application there is a button on the Master view.



When you click it, the app navigates to the Detail1 view.



## Add a back Button

### Detail1.view.xml

To add a back button add the highlighted code to Detail.view.xml.

|  |
| --- |
| <core:View xmlns:core="sap.ui.core"  xmlns:mvc="sap.ui.core.mvc"  xmlns="sap.m" controllerName="ui5.controller.Detail1"  xmlns:html="http://www.w3.org/1999/xhtml">  <Page title="Detail 1"  showNavButton = "true"  navButtonPress = "Back">  <content>  </content>  </Page>  </core:View> |

Listing

### Detail1.controller.js

Next, add a function to the Detail1 controller to cause the navigation to the Master view.

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller"  ], function (Controller) {  "use strict";  return Controller.extend("ui5.controller.Detail1", {    Back: function(){  var oRouter = sap.ui.core.UIComponent.getRouterFor(this);  oRouter.navTo('Master');  }  });  }); |

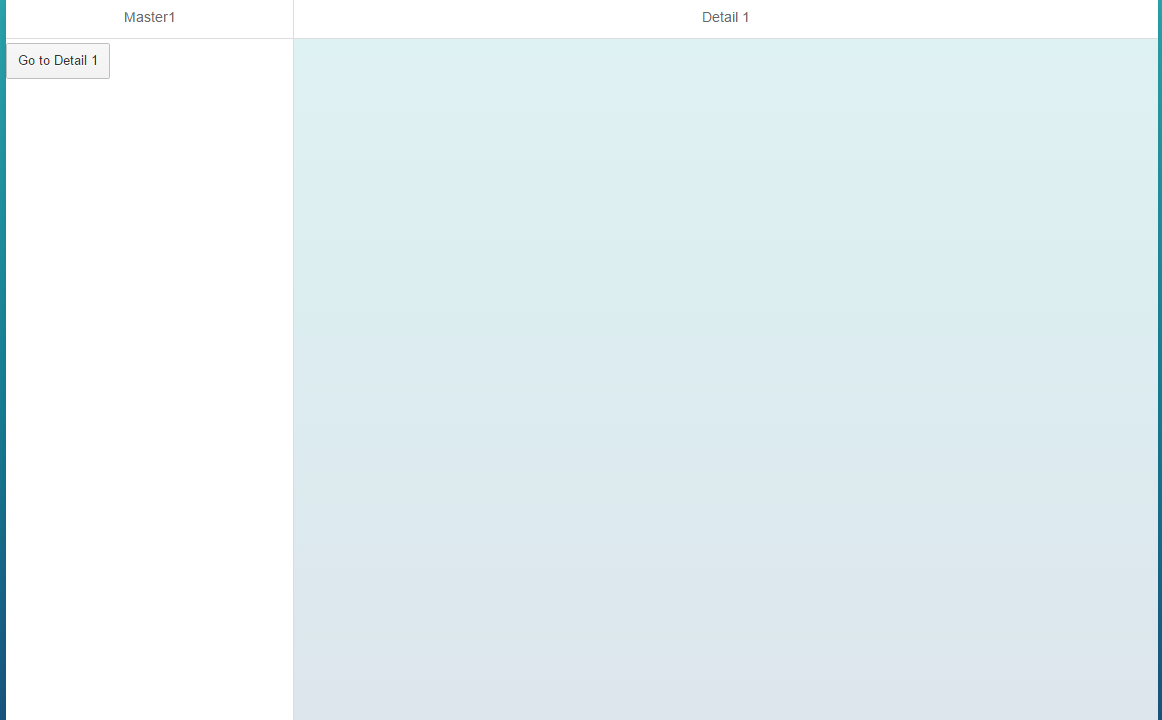
Listing

Run the application now and there is a back button on the Detail1 view that allows you to navigate back to the Master view.

## Make a Master-Detail App

Many business applications use a master-detail structure (e.g. Customers is the master and the customer’s sales orders are the details). In UI5 the master-detail structure is represented as shown below. The Master view is on the left and the Details are shown on the right. In this section, we’ll reorganize the app into a master-detail structure.

First undo the changes you made to add a back button.



### App.view.xml

Change the App control in the App view as shown.

|  |
| --- |
| <mvc:View  xmlns="sap.m"  xmlns:mvc="sap.ui.core.mvc"  displayBlock="true">  <**SplitApp** class="myAppDemoWT" id="app"/>  </mvc:View> |

Listing

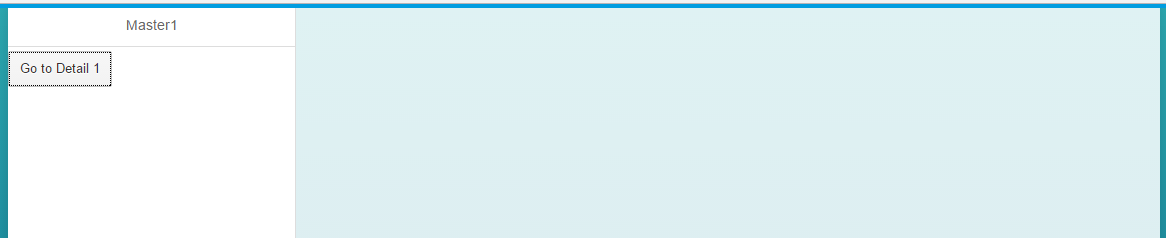
### manifest.json

Update the targets as shown below.

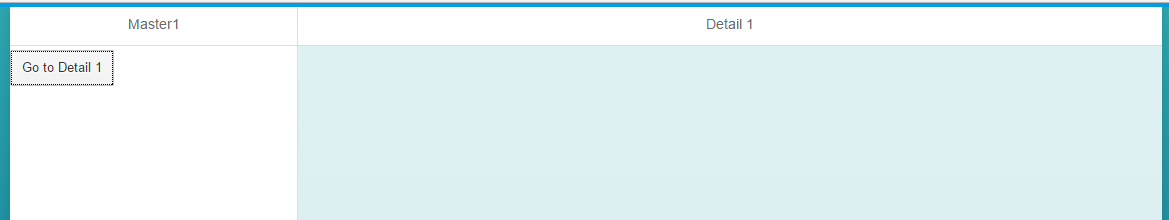
|  |
| --- |
| "routing": {  "config": {  "routerClass": "sap.m.routing.Router",  "viewType": "XML",  "viewPath": "ui5.view",  "controlId": "app"  },  "routes": [  {  "pattern": "",  "name": "Master",  "target": "Master"  },  {  "pattern": "Detail1",  "name": "Detail1",  "target": "Detail1"  }  ],  "targets": {  "Master": {  "viewName": "Master**",**  **"controlAggregation": "masterPages"**  },  "Detail1": {  "viewName": "Detail1**",**  **"controlAggregation": "detailPages"**  }  } |

Listing

The only change is to move the controlAggregation from the config section to the target section and changing the names of the aggregation from pages to masterPages and detailPages. Now, when you run the app the Master view loads in the master section and the detail section is empty.



If you click the button, the Detail1 view loads in the detail section.



## Passing Parameters

We often need to pass parameters in routes. For example, if the user clicks an item on the list to load a view that shows details of the item, we need to pass the identity of the item clicked in the route. This section will illustrate how that is done. We will create a List control on the Master view. When the user clicks and item on the List, the Detail1 view will be loaded showing a List with details of the clicked item.

### data.json

Add the code shown below to the data.json file. This code is in JSON format. It has a single collection (array) called collection that contains two objects. Each object has a property called prop1 which is a string and a property called subcol which is an array.

|  |
| --- |
| {  "collection" : [  {  "prop1" : "A",  "subcol" : [  {  "subProp1" : "One"  },  {  "subProp1" : "Two"  }  ]  },  {  "prop1" : "B",  "subcol" : [  {  "subProp1" : "Three"  },  {  "subProp1" : "Four"  }  ]  }    ]  } |

Listing

### Manifest.json

Now, we have to add the model to the manifest.json file. Update the code using the highlighted portions below.

|  |
| --- |
| {  "\_version": "1.1.0",  "sap.app": {  "\_version": "1.1.0",  "id": "ui5",  "type": "application",  "i18n": "i18n/i18n.properties",  "title": "Routing",  "description": "Routing Exercise",  "applicationVersion": {  "version": "1.0.0"  }  },  "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": {  "routing": {  "type": "sap.ui.model.json.JSONModel",  "uri": "model/data.json"  }  },  "routing": {  "config": {  "routerClass": "sap.m.routing.Router",  "viewType": "XML",  "viewPath": "ui5.view",  "controlId": "app"  },  "routes": [  {  "pattern": "",  "name": "Master",  "target": "Master"  },  {  "pattern": "Detail1",  "name": "Detail1",  "target": "Detail1"  }  ],  "targets": {  "Master": {  "viewName": "Master",  "controlAggregation": "masterPages"  },  "Detail1": {  "viewName": "Detail1",  "controlAggregation": "detailPages"  }  }  }  }  } |

Listing

This code creates a JSON model and names it routing.

### Master.view.xml

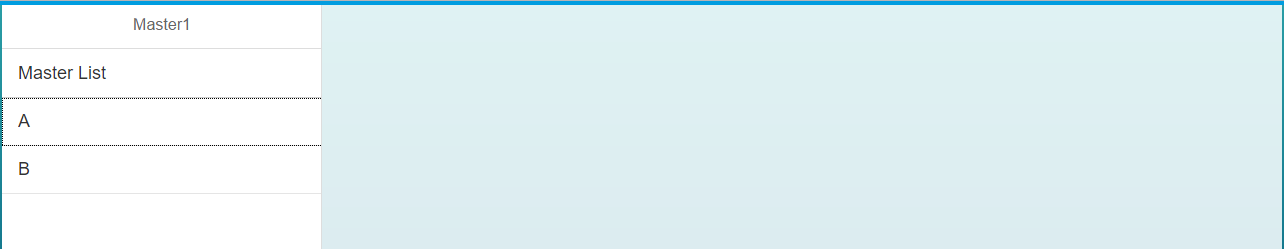
Update the code as shown:

|  |
| --- |
| <core:View xmlns:core="sap.ui.core"  xmlns:mvc="sap.ui.core.mvc"  xmlns="sap.m" controllerName="ui5.controller.Master"  xmlns:html="http://www.w3.org/1999/xhtml">  <Page title="Master1">  <content>  <List  headerText="Master List"  items="{routing>/collection}" >  <StandardListItem  type="Active"  press="go"  title="{routing>prop1}" />  </List>  </content>  </Page>  </core:View> |

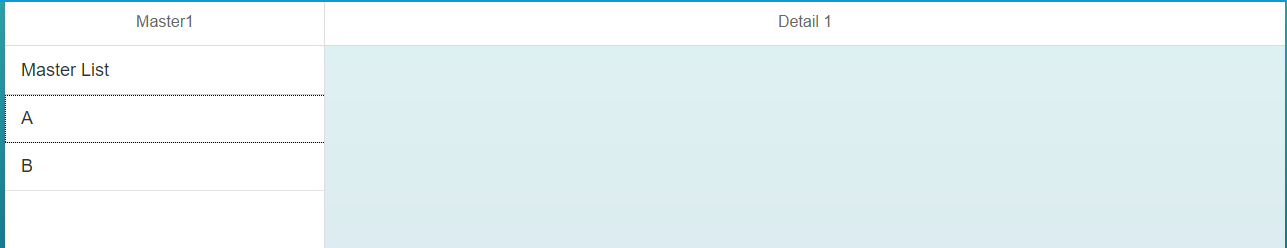
Listing

This code replaces the Button control with a List control. The List control has an aggregation binding called items which is bound the collections collection in the routing model. A StandardListItem control is provided as a template to create each of the items on the list. The title property of the StandardListItem is bound to the prop1 property and the press event is bound to the go function.

If you run the application now, you should see a list control in the Master view with two items.



If you click an item on the list, the Detail1 route is invoked and the Detail1 one view is loaded.



Next, we want to add a list to the Detail1 view that shows the items in the subcol collection that correspond to the item clicked in the Master view list.

### Detail1.view.xml

First, update the Detail1 view as shown.

|  |
| --- |
| <core:View xmlns:core="sap.ui.core"  xmlns:mvc="sap.ui.core.mvc"  xmlns="sap.m" controllerName="ui5.controller.Detail1"  xmlns:html="http://www.w3.org/1999/xhtml">  <Page title="Detail 1">  <content>  <List  headerText="Detail 1 List"  items="{routing>subcol}" >  <StandardListItem  type="Active"  title="{routing>subProp1}" />  </List>  </content>  </Page>  </core:View> |

Listing

### Manifest.json

Next, update the route in the Manifest.json file so that it accommodates a parameter. Update the Detail1 route pattern as shown.

|  |
| --- |
| "routes": [  {  "pattern": "",  "name": "Master",  "target": "Master"  },  {  "pattern": "Detail1/{item}",  "name": "Detail1",  "target": "Detail1"  }  ],  "targets": {  "Master": {  "viewName": "Master",  "controlAggregation": "masterPages"  },  "Detail1": {  "viewName": "Detail1",  "controlAggregation": "detailPages"  }  } |

Listing

### Master.controller.js

Next, update the go function in the Master controller so that it passes the index of the list item that was clicked.

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller"  ], function (Controller) {  "use strict";  return Controller.extend("ui5.controller.Master", {  go: function (oEvent) {  var oItem = oEvent.getSource();  var oRouter = sap.ui.core.UIComponent.getRouterFor(this);  oRouter.navTo("Detail1", {  item: oItem.getBindingContext("routing").getPath().substr(12)  });  }    });  }); |

Listing

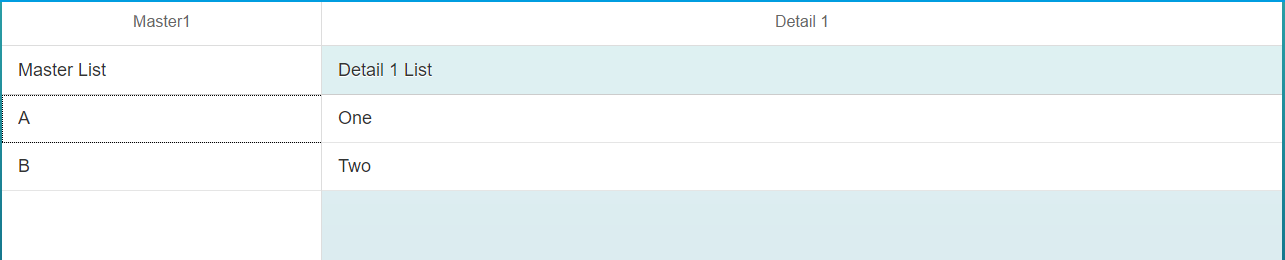
The getSource() function retrieves the list item that was clicked. The getBindingContext() function retrieves the object bound to that item and the getPath() function retrieves the path in the data model to that object. If the first item is clicked, the path will be /collection/0 and if the second item is clicked the path will be /collection/1. The substr() function returns the 12 character in that path which will be the index, 0 or 1.

The term item matches the parameter we added to the route in the Manifest.json file.

### Detail1.controller.js

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller"  ], function (Controller) {  "use strict";  return Controller.extend("ui5.controller.Detail1", {  onInit: function () {  var oRouter = sap.ui.core.UIComponent.getRouterFor(this);  oRouter.getRoute("Detail1").attachPatternMatched(this.\_onObjectMatched, this);  },  \_onObjectMatched: function (oEvent) {  this.getView().bindElement({  path: "/collection/" + oEvent.getParameter("arguments").item,  model: "routing"  });  }  });  }); |

Listing

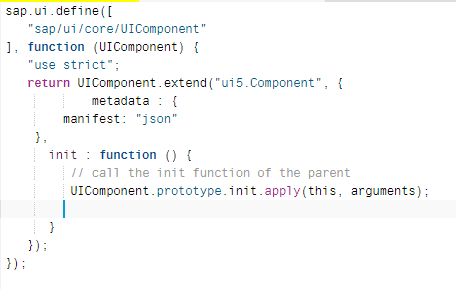


# Debugging Routing

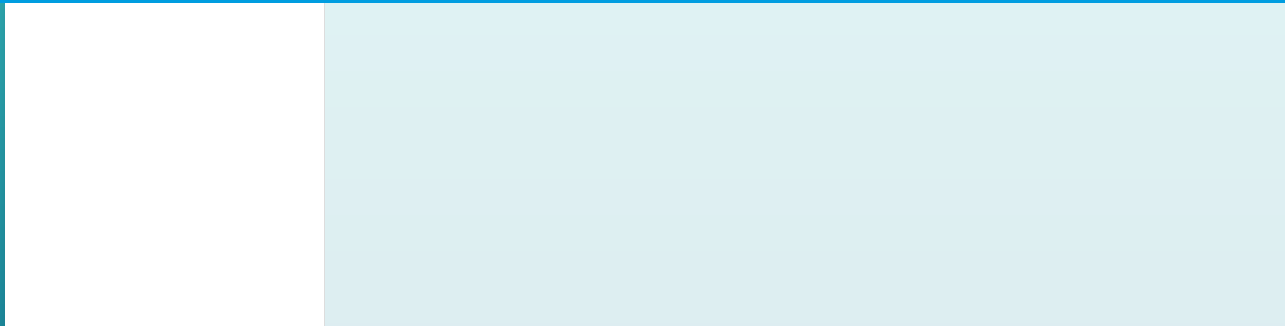
There are a number of things that can go wrong with a routing configuration. Fortunately, most of them are easy to identify if you know what to look for.

### Missing Routing Declaration in Component.js

If you omit the declaration of the router in the Component.js file, you won’t see an error but no views will load.



The application loads but no views are loaded.



### Errors with Targets in Manifest.json

Errors will occur if the targets don’t match in the manifest.json file. In the image below the target specified in the Master route is Master but there is no target defined for Master (there is a target for Master1).



The error you see is self-explanatory.

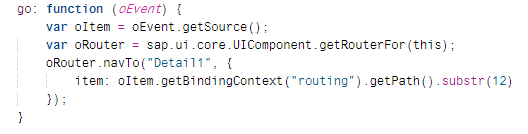
|  |
| --- |
| sap-ui-core.js:160 2017-06-29 13:10:35.163395 The target with the name "Master" does not exist! - EventProvider sap.m.routing.Targets |

### Errors in Route Names

In the code below the Detail1 route is named Detail2...



However, the code that attemps to invoke this route uses the name Detail1.



In the case, the error looks like this.

|  |
| --- |
| Can not navigate to route with name Detail1 because the route does not exist – |

Other errors can be diagnosed in a similar way.

# Exercise

Create a new view called Detail2.view.xml and implement the press event handler to navigate to the Detail2 view in the detailPages aggregation (Detail2 replaces Detail1) as shown below when an item on the **Detail1** list is clicked.

