Week 4 Unit 1

Navigation and Routing concepts

Please perform the exercises below in your app project as shown in the video.

Table of Contents

[1 Implement a new “add” route 2](#_Toc449087114)

[2 Implement a new “add” view 4](#_Toc449087115)

[3 Implement a new “add” controller 5](#_Toc449087116)

## Preview

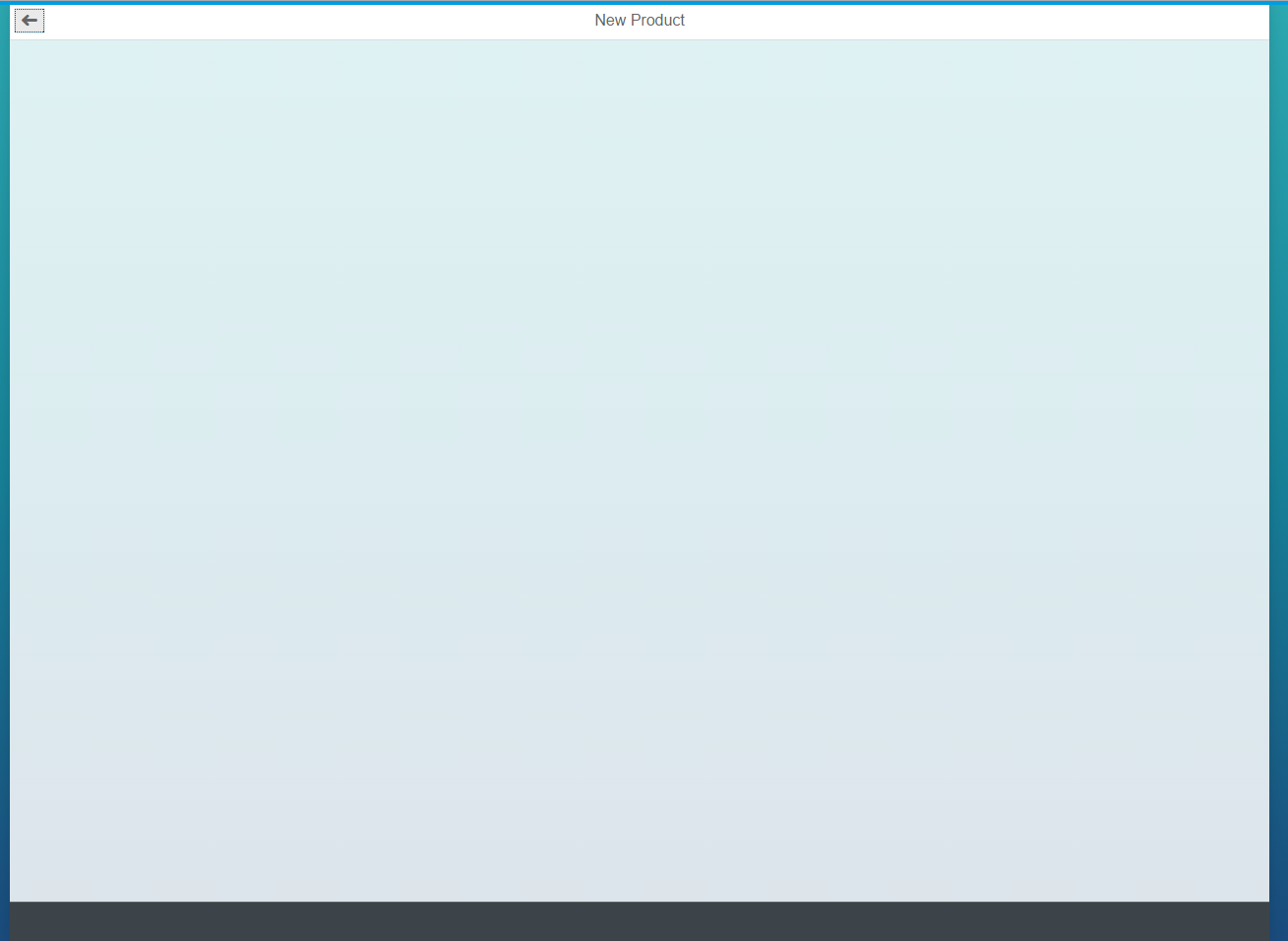


Figure 1 - Preview of the app after doing this unit’s exercises

# Implement a New “add” Route

In this step, we implement an “add” Button, a new “Add” view with a corresponding controller and the route and target to have it within the navigation.

**webapp/view/Worklist.view.xml**

…

<ToolbarSpacer />

<SearchField

id="searchField"

tooltip="{i18n>worklistSearchTooltip}"

search="onSearch"

width="auto">

</SearchField>

**<Button id="addButton" icon="sap-icon://add" press="onAdd" />**

</Toolbar>

</headerToolbar>

…

To achieve this we implement a Button within the Worklist.view.xml within the HeaderToolbar next to the SearchField. As icon, we use a standard “+” icon from the standard SAPUI5 icon font.

**webapp/controller/Worklist.controller.js**

…

/\* =========================================================== \*/

/\* event handlers \*/

/\* =========================================================== \*/

**/\*\***

**\* Event handler when the add button gets pressed**

**\* @public**

**\*/**

**onAdd: function() {**

**this.getRouter().navTo("add");**

**},**

…

The onAdd function we reference within the Worklist.view.xml has to be defined in the Worklist.controller.js. When this function is called, we simply navigate to the “add” route, which still has to be defined.

**webapp/manifest.json**

…

"routing": {

"config": {

"routerClass": "sap.m.routing.Router",

"viewType": "XML",

"viewPath": "opensap.manageproducts.view",

"controlId": "app",

"controlAggregation": "pages",

"bypassed": {

"target": "notFound"

},

"async": true

},

"routes": [

{

"pattern": "",

"name": "worklist",

"target": "worklist"

},

{

"pattern": "ProductSet/{objectId}",

"name": "object",

"target": "object"

}**,**

**{**

**"pattern" : "AddProduct",**

**"name" : "add",**

**"target" : "add"**

**}**

],

"targets": {

"worklist": {

"viewName": "Worklist",

"viewId": "worklist",

"viewLevel": 1

},

"object": {

"viewName": "Object",

"viewId": "object",

"viewLevel": 2

},

**"add": {**

**"viewName": "Add",**

**"viewId": "add",**

**"viewLevel": 3**

**},**

"objectNotFound": {

"viewName": "ObjectNotFound",

"viewId": "objectNotFound"

},

"notFound": {

"viewName": "NotFound",

"viewId": "notFound"

}

}

}

…

Now we have to define the route and corresponding target within the manifest.json. The pattern of the route is “AddProduct” and it is named “add”, like we already referenced it within the Worklist.controller.js. The corresponding target references the Add.view.xml which we still have to implement.

# Implement a New “add” View

**webapp/view/Add.view.xml (NEW)**

**<mvc:View**

**controllerName="opensap.manageproducts.controller.Add"**

**xmlns:mvc="sap.ui.core.mvc"**

**xmlns:semantic="sap.m.semantic"**

**xmlns="sap.m">**

**<semantic:FullscreenPage**

**id="page"**

**title="{i18n>addPageTitle}"**

**showNavButton="true"**

**navButtonPress="onNavBack">**

**</semantic:FullscreenPage>**

**</mvc:View>**

The Add.view.xml is located within the view folder and simply consists of a SemanticPage with a title and a “Back” button.

**webapp/i18n/i18n.properties**

**…**

#YMSG: Send E-Mail message

shareSendEmailObjectMessage=<Email body PLEASE REPLACE ACCORDING TO YOUR USE CASE> {0} (id: {1})\r\n{2}

**#~~~ Add View ~~~~~~~~~~~~~~~~~~~~~~~~~~**

**#XTIT: Add view title**

**addPageTitle=New Product**

#~~~ Not Found View ~~~~~~~~~~~~~~~~~~~~~~~

…

Within the i18n.properties file we add the title for the page title of the Add.view.xml.

# Implement a New “add” Controller

### webapp/controller/Add.controller.js (NEW)

**sap.ui.define([**

**"opensap/manageproducts/controller/BaseController",**

**"sap/ui/core/routing/History"**

**], function(BaseController, History) {**

**"use strict";**

**return BaseController.extend("opensap.manageproducts.controller.Add", {**

**/\* =========================================================== \*/**

**/\* lifecycle methods \*/**

**/\* =========================================================== \*/**

**/\*\***

**\* Called when the add controller is instantiated.**

**\* @public**

**\*/**

**onInit: function() {**

**// Register to the add route matched**

**this.getRouter().getRoute("add").attachPatternMatched(this.\_onRouteMatched, this);**

**},**

**/\* =========================================================== \*/**

**/\* event handlers \*/**

**/\* =========================================================== \*/**

**\_onRouteMatched: function() {**

**//here goes your logic which will be executed when the "add" route is hit**

**//will be done within the next unit**

**},**

**/\*\***

**\* Event handler for navigating back.**

**\* It checks if there is a history entry. If yes, history.go(-1) will happen.**

**\* If not, it will replace the current entry of the browser history with the worklist route.**

**\* @public**

**\*/**

**onNavBack : function() {**

**var oHistory = History.getInstance(),**

**sPreviousHash = oHistory.getPreviousHash();**

**if (sPreviousHash !== undefined) {**

**// The history contains a previous entry**

**history.go(-1);**

**} else {**

**// Otherwise we go backwards with a forward history**

**var bReplace = true;**

**this.getRouter().navTo("worklist", {}, bReplace);**

**}**

**}**

**});**

**});**

Within the Add.controller.js, we register to the patternMatched event of the add route in the onInit method of the controller. The \_onRouteMatched method we reference here will be used and further implemented in the next unit. We also implement an onNavBack function which either goes back in the history, if there is any, or navigates forward to the entry page of the app.

Our app contains now a new view which can be displayed and reached via navigation.

### Conventions

* Define the routing configuration in the descriptor (manifest.json).
* Add a path to go back to the parent page when the history state is unclear.

**Coding Samples**

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.