

# UX410

## Developing SAP Fiori UIs

### EXERCISES AND SOLUTIONS

Course Version: 20

Course Duration: 8 Hours 20 Minutes



# SAP Copyrights, Trademarks and Disclaimers

© 2020 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see <http://global12.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.

National product specifications may vary.

This course may have been machine translated and may contain grammatical errors or inaccuracies.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.

# Typographic Conventions

American English is the standard used in this handbook.

The following typographic conventions are also used.

This information is displayed in the instructor's presentation



Demonstration



Procedure



Warning or Caution



Hint



Related or Additional Information



Facilitated Discussion



User interface control

*Example text*

Window title

*Example text*

# Contents

<b>Unit 1:</b>	<b>Introduction to User Experience (UX)</b>
	No exercises
<b>Unit 2:</b>	<b>SAP UX strategy</b>
	No exercises
<b>Unit 3:</b>	<b>SAP Fiori UX</b>
	No exercises
<b>Unit 4:</b>	<b>User Experience Design</b>
1	Exercise 1: Create a Prototype with SAP BUILD
<b>Unit 5:</b>	<b>SAP Fiori Design Guidelines</b>
	No exercises
<b>Unit 6:</b>	<b>Development Basics SAP Web IDE</b>
12	Exercise 2: Create a Trial Account for SAP Cloud Platform and Start the SAP Web IDE
17	Exercise 3: Create a Simple SAPUI5 Application with the help of the Layout Editor
26	Exercise 4: Set up the SAP Cloud Connector
30	Exercise 5: Deploy an SAPUI5 Application via the SAP Web IDE on an ABAP Server
35	Exercise 6: Import an SAP BUILD Prototype into SAP Web IDE
<b>Unit 7:</b>	<b>SAPUI5 Advanced Topics</b>
39	Exercise 7: Work with Diagrams
<b>Unit 8:</b>	<b>Golden Rules of SAPUI5 Development</b>
	No exercises
<b>Unit 9:</b>	<b>SAP Fiori Launchpad</b>
	No exercises
<b>Unit 10:</b>	<b>SAP Fiori Launchpad Configuration</b>
71	Exercise 8: Navigate in SAP Fiori

<b>Unit 11:</b>	<b>SAP Fiori Layout Decision Guidelines</b>
94	Exercise 9: Create a Dynamic Page App
118	Exercise 10: Create Master-Detail using the Flexible Column Layout
<b>Unit 12:</b>	<b>SAP Fiori Design Guidelines</b>
139	Exercise 11: Implement Value Helps
168	Exercise 12: Implement a List Report
200	Exercise 13: Implement an Object Page
<b>Unit 13:</b>	<b>SAP Fiori Extension Concept</b>
210	Exercise 14: Implement and Extend an Extension Point
<b>Unit 14:</b>	<b>SAP Fiori Elements</b>
224	Exercise 15: Implement List Report using SAP Fiori Elements
240	Exercise 16: Implement Search and Filter
248	Exercise 17: Implement Object Page SAP Fiori Elements
260	Exercise 18: Display Dependent Entities as SAP Fiori Elements
<b>Unit 15:</b>	<b>Lean Development Infrastructure</b>
No exercises	
<b>Unit 16:</b>	<b>Hybrid Application Toolkit</b>
No exercises	

## Unit 4 Exercise 1

# Create a Prototype with SAP BUILD



### Note:

SAP Build is a cloud service. Due to the nature of cloud software, step procedures and names such as *buttons* or *fields* may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business Scenario

In this exercise, you create a prototype with **SAP BUILD** using the **Clone** function.

1. Create a new account at <https://www.build.me>.
2. Log in to your newly created account.

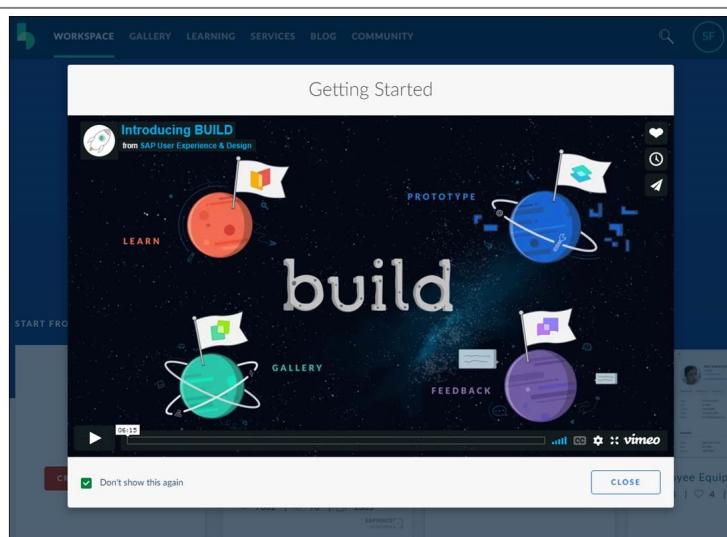


Figure 7: Getting Started

You are now logged into SAP BUILD.

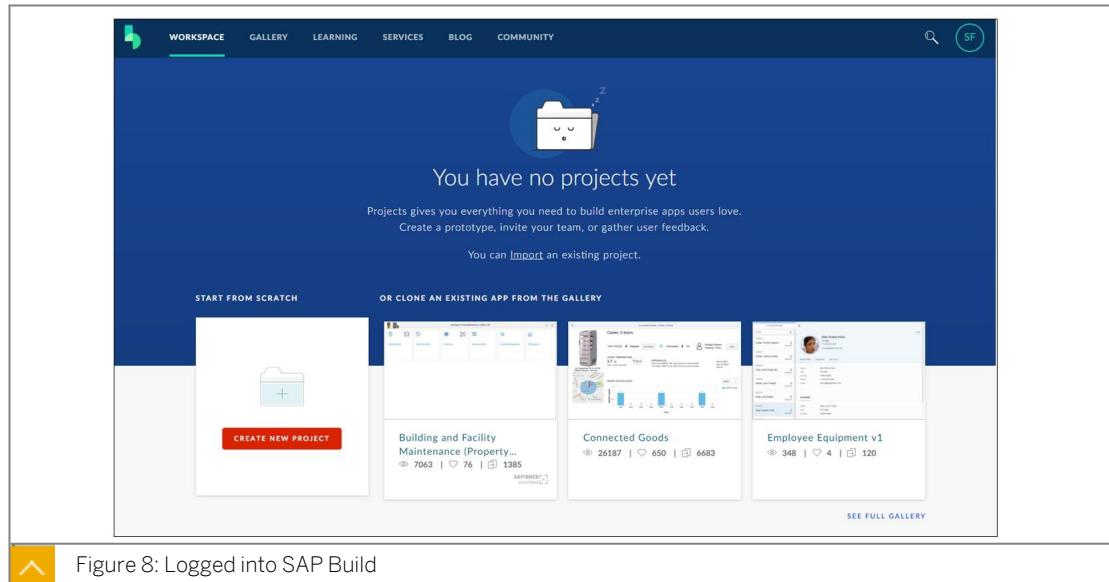


Figure 8: Logged into SAP Build

3. Clone the template project *Manage Products* from the gallery.
4. Open the cloned project to familiarize yourself with the design of the newly-created prototype.

# Unit 4 Solution 1

## Create a Prototype with SAP BUILD



### Note:

SAP Build is a cloud service. Due to the nature of cloud software, step procedures and names such as *buttons* or *fields* may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business Scenario

In this exercise, you create a prototype with **SAP BUILD** using the **Clone** function.

1. Create a new account at <https://www.build.me>.
  - a) Open Google Chrome and insert <https://www.build.me> into the address bar.

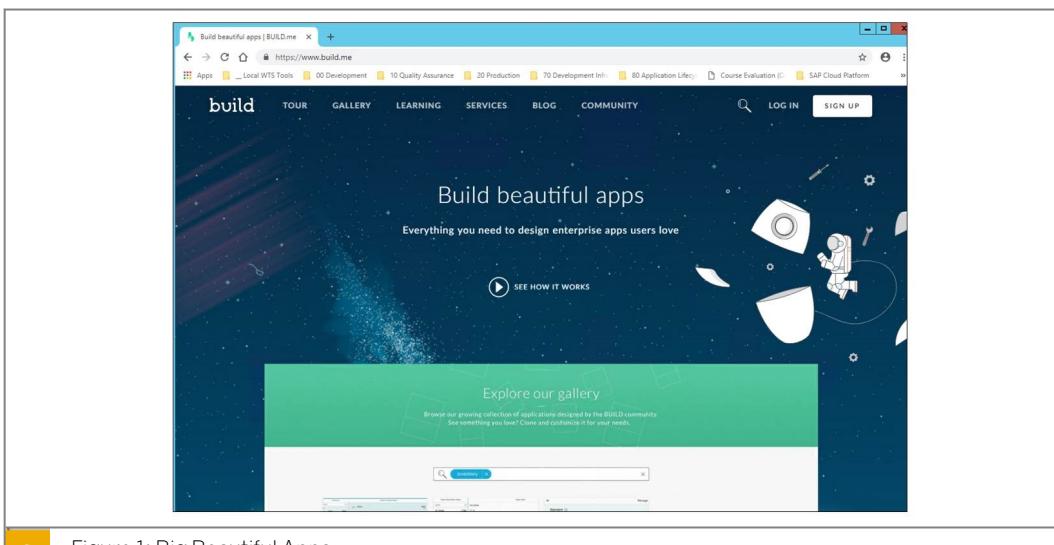


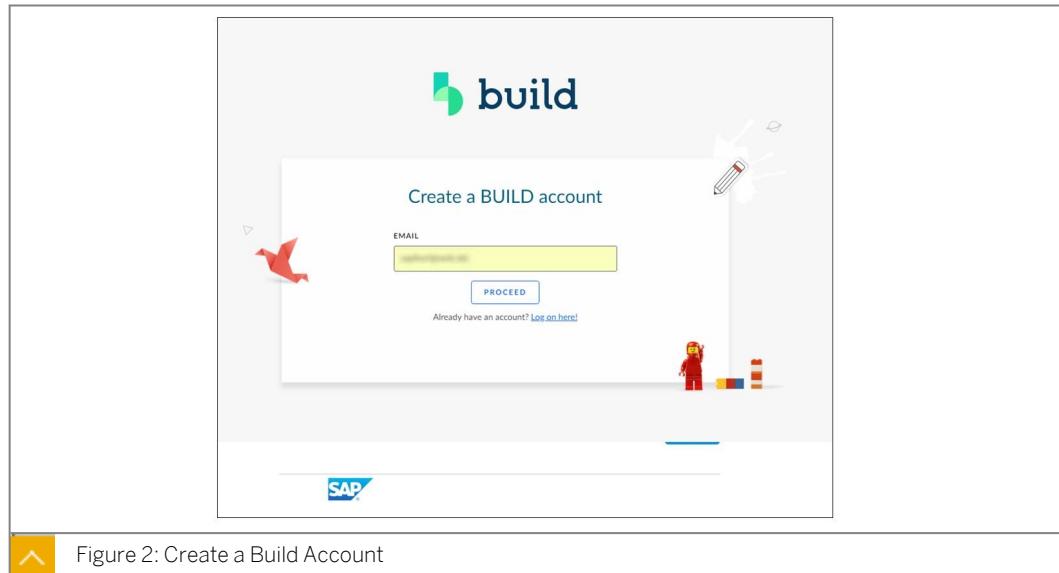
Figure 1: Big Beautiful Apps

- a) Choose the button labelled **SIGN UP**.
- c) Enter an email address in the email address field and choose **Proceed**.



### Note:

To activate the trial account, you need to access the entered email account. If you do not have an email account that you can access during training, you can create one, for example at <https://www.google.com/gmail/>.



- d) Fill in the fields using a valid email address to receive an activation link and choose Register.

The screenshot shows the SAP BUILD registration form. The title 'Registration' is at the top. The 'Tell Us About Yourself' section contains three input fields: 'First Name \*' with placeholder 'John', 'Last Name \*' with placeholder 'Doe', and 'E-Mail \*' with placeholder 'john.doe@example.com'. The 'Set Password' section has two input fields: 'Password \*' and 'Re-Enter Password \*', both with placeholder '\*\*\*\*\*' and a checkmark icon. The 'Terms and Conditions' section contains two checkboxes: 'I have read the [Privacy Statement](#) and consent to this agreement. \*' and 'I have read and understood the Terms and Conditions of SAP BUILD. \*'. A note at the bottom right says '\*Required'. A blue 'Register' button is at the bottom right. The SAP logo is at the bottom left.

Figure 3: Registration SAP Build

- e) Open the activation link you receive to your submitted email address. This activates your account.

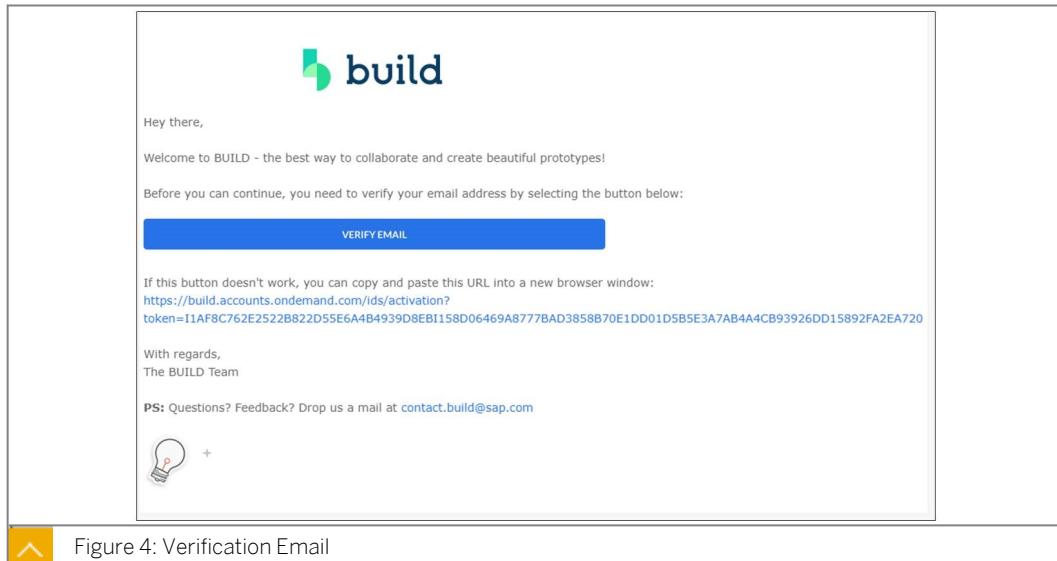


Figure 4: Verification Email

f) Choose *Continue*.

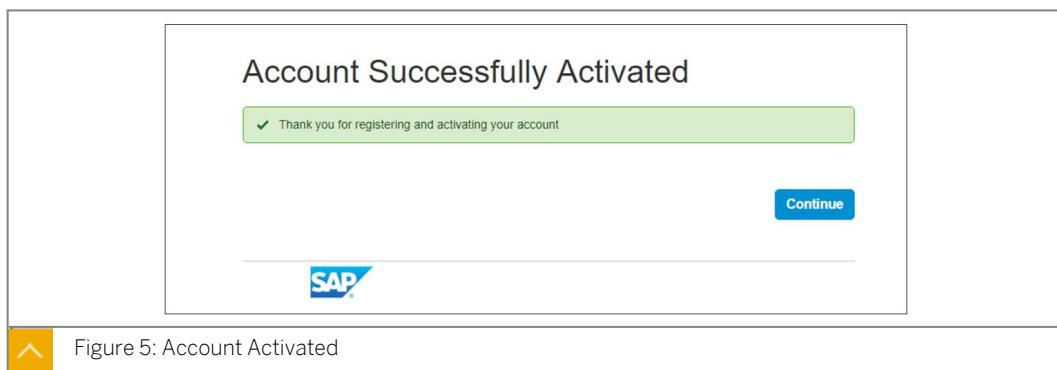


Figure 5: Account Activated

2. Log in to your newly created account.

a) Choose *login*.

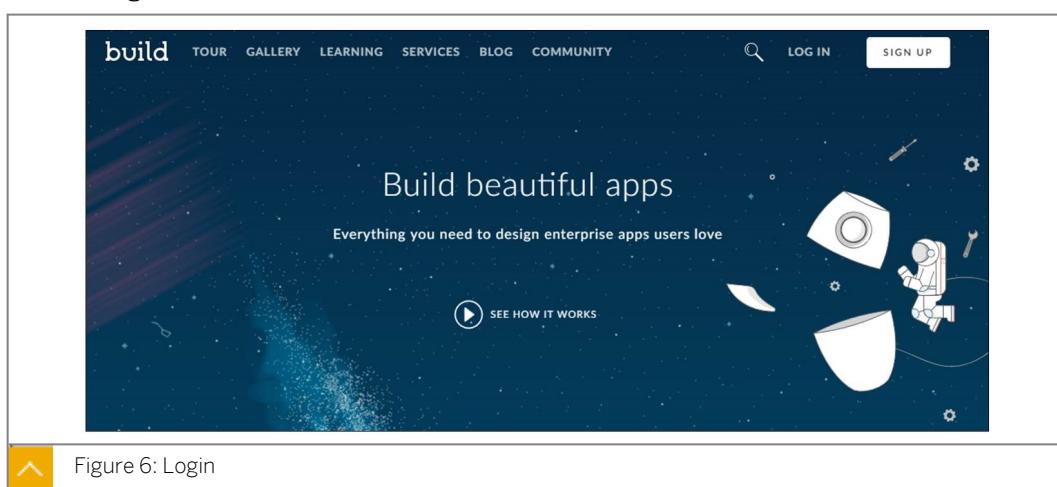


Figure 6: Login

b) Enter your email address and choose *Proceed*.

c) Optionally, select the *Checkbox* to 'Don't show this again' and choose *Close*.

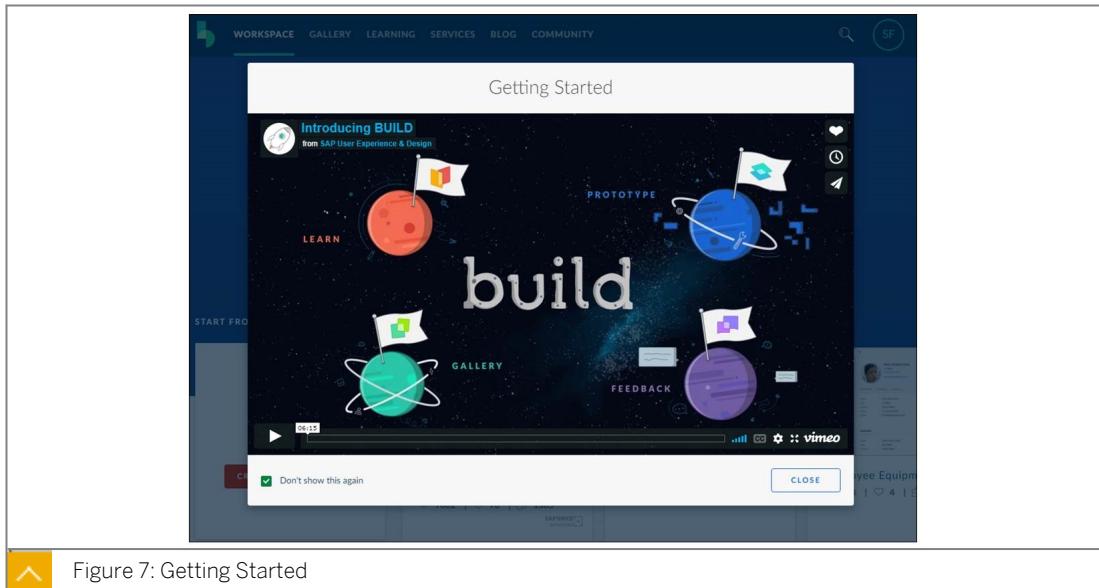


Figure 7: Getting Started

You are now logged into SAP BUILD.

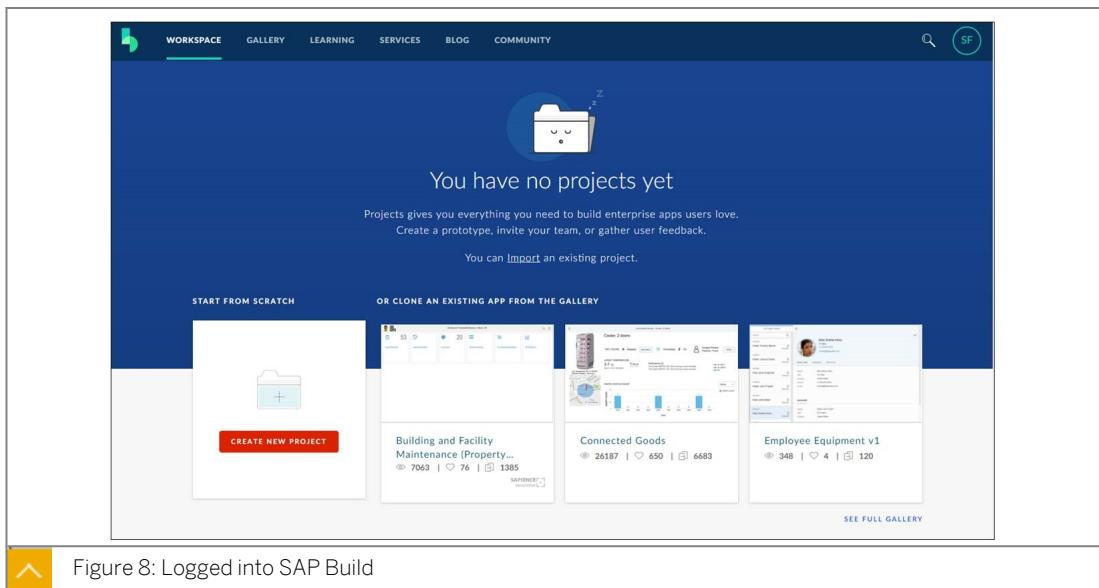


Figure 8: Logged into SAP Build

3. Clone the template project *Manage Products* from the gallery.
  - a) Choose the button *See Full Gallery*.

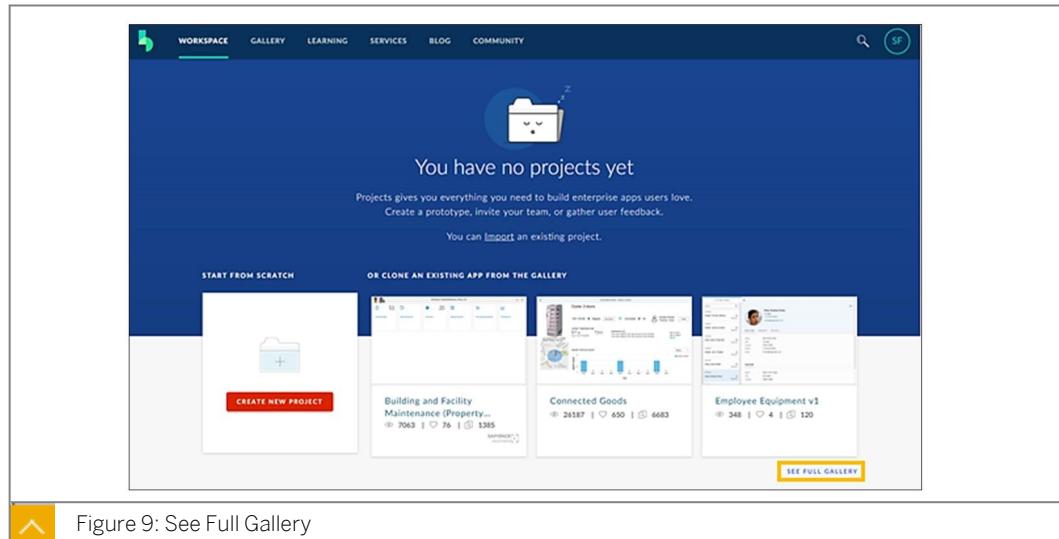


Figure 9: See Full Gallery

b) Browse the Gallery.

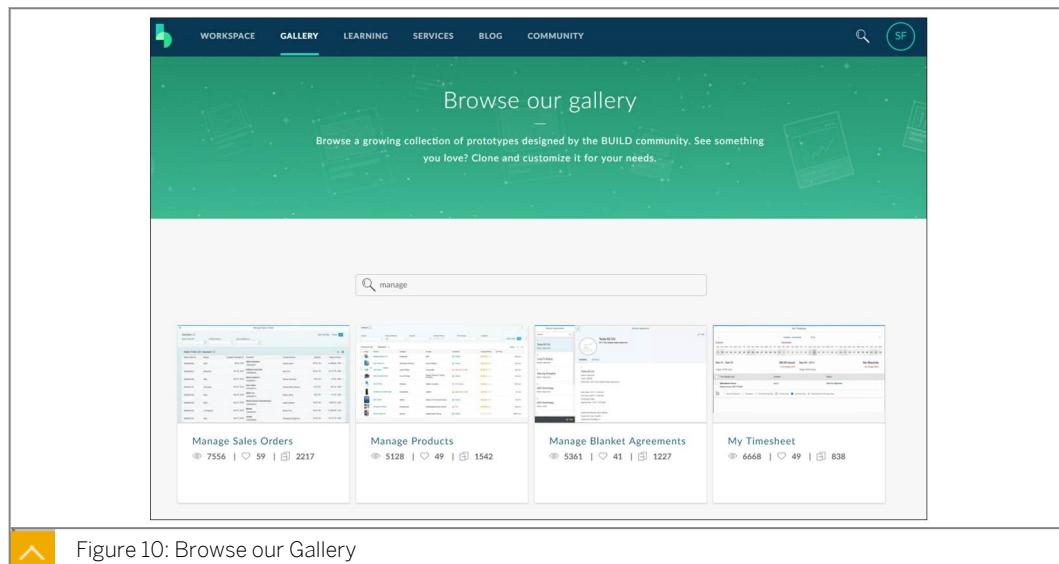


Figure 10: Browse our Gallery

c) Search for the prototype of *Manage products* by entering **Manage** into the search field.

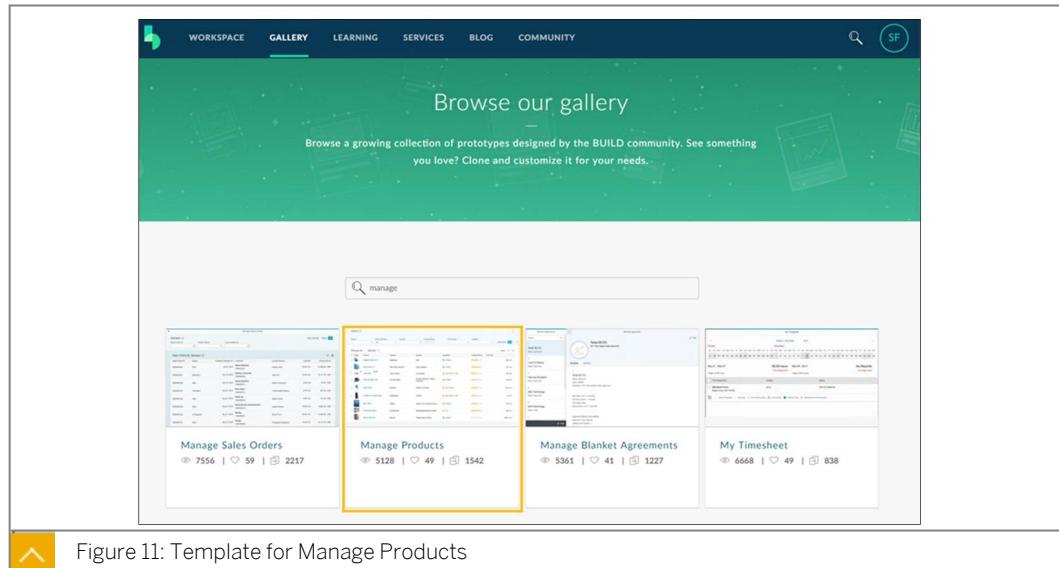


Figure 11: Template for Manage Products

d) Choose the template *Manage Products*.

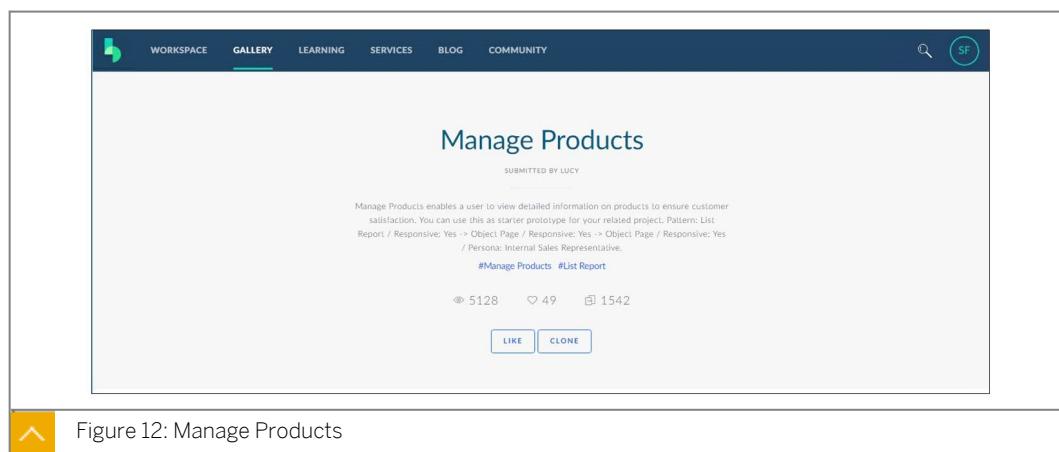


Figure 12: Manage Products

e) Choose the *Clone* button.

f) Choose *Clone to a new project* from the *CLONE TO* options.

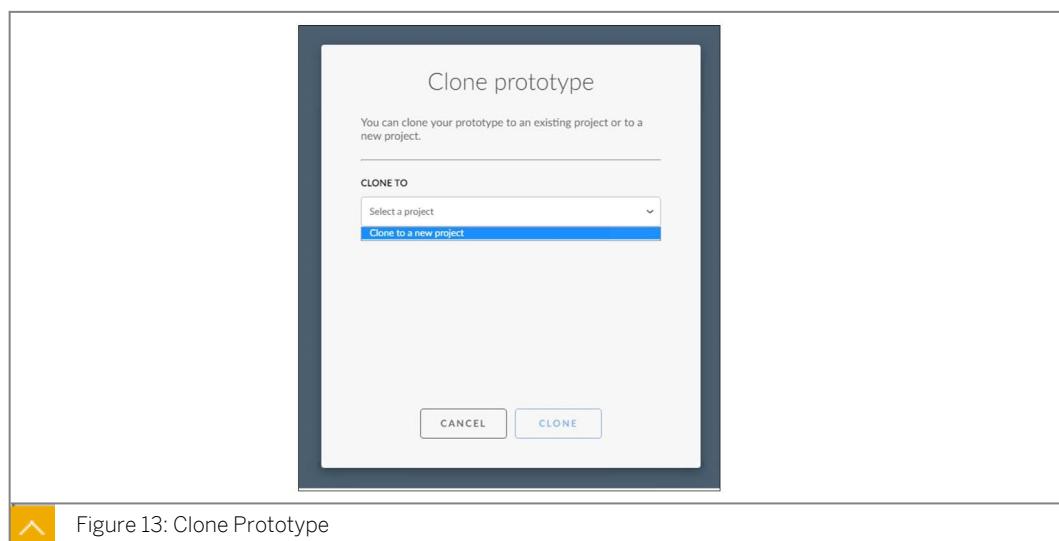
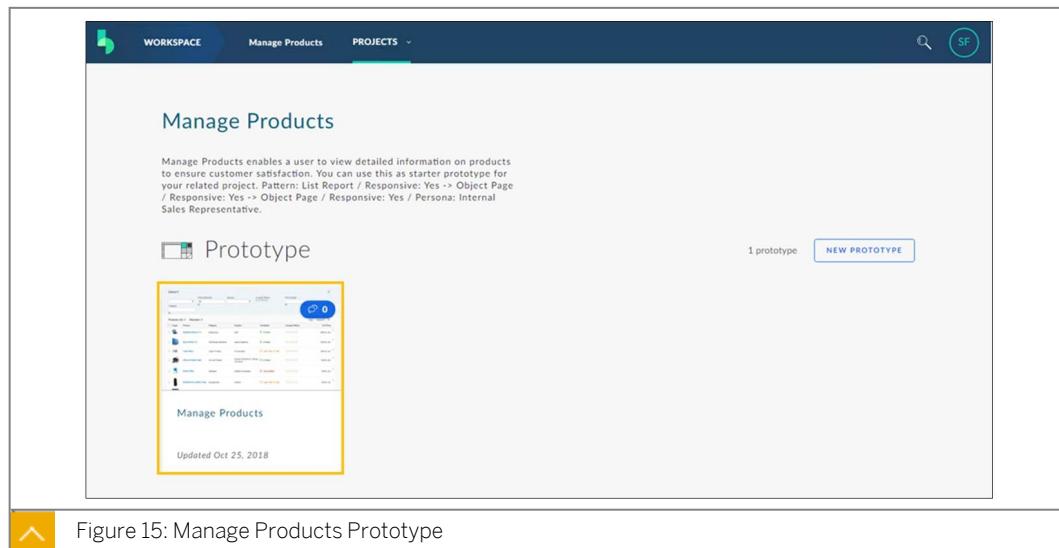


Figure 13: Clone Prototype

g) Choose the *CLONE* button to start the clone process.



- h) After successfully cloning the template, choose *OK*.
4. Open the cloned project to familiarize yourself with the design of the newly-created prototype.
- a) To open the newly-created prototype choose *Title*.



- b) Open *Page 1* in *Edit* mode to investigate the new prototype.

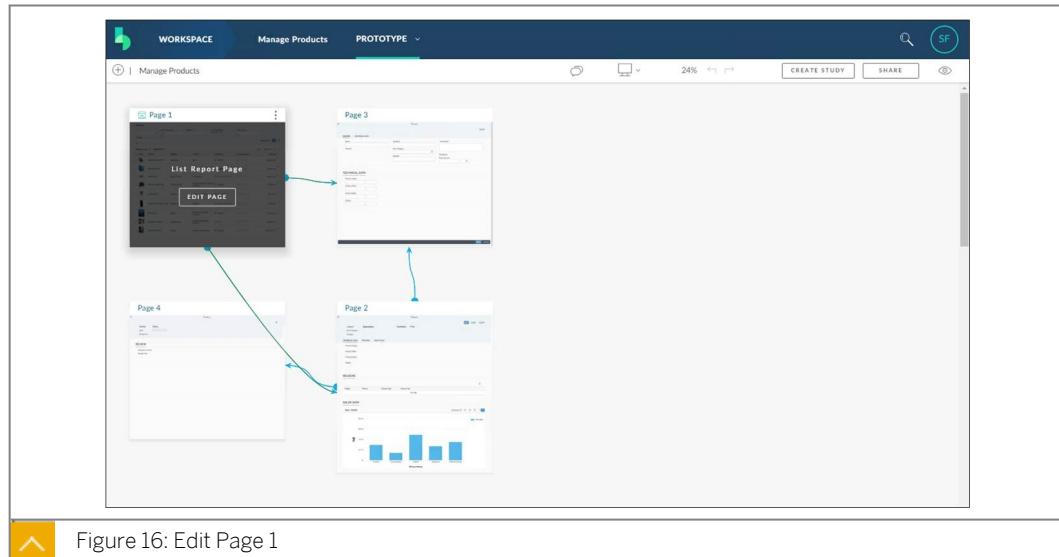


Figure 16: Edit Page 1

c) Open the *Data Editor* to take a look at the implementation.

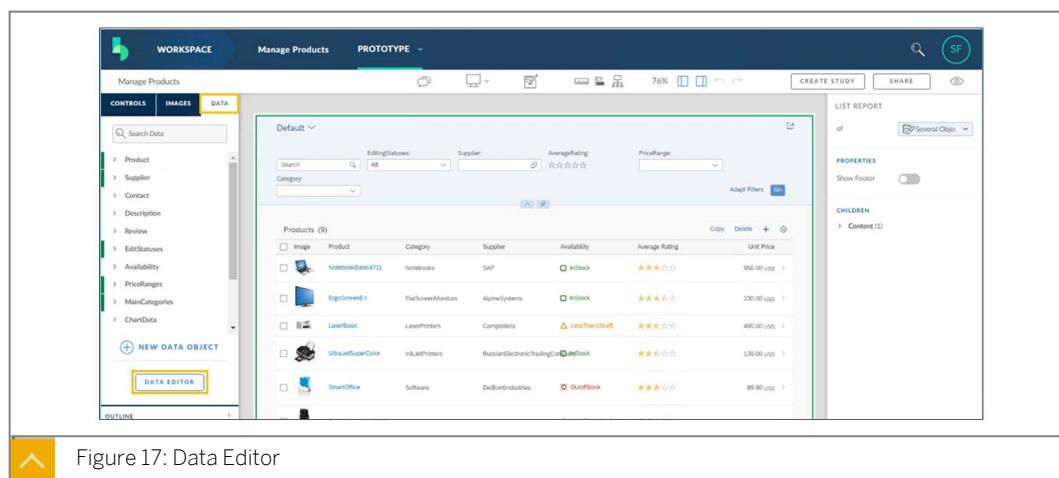


Figure 17: Data Editor

d) After the review, select the *Preview* button in the upper toolbar to start a preview of the cloned project.

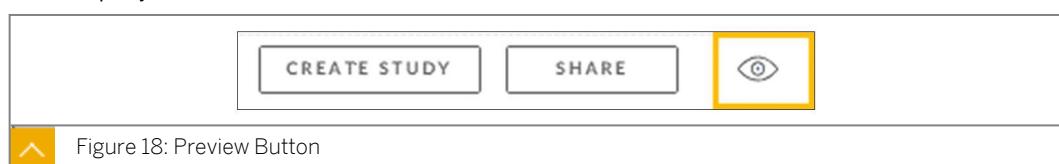


Figure 18: Preview Button

e) Explore the application.

Figure 19: Application

- f) Choose the *UI EDITOR* button in the upper toolbar to exit preview mode.

Figure 20: UI Editor Button

- g) You finished the exercise successfully. Log out from SAP BUILD.

Figure 21: Logout

## Unit 6 Exercise 2

# Create a Trial Account for SAP Cloud Platform and Start the SAP Web IDE



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

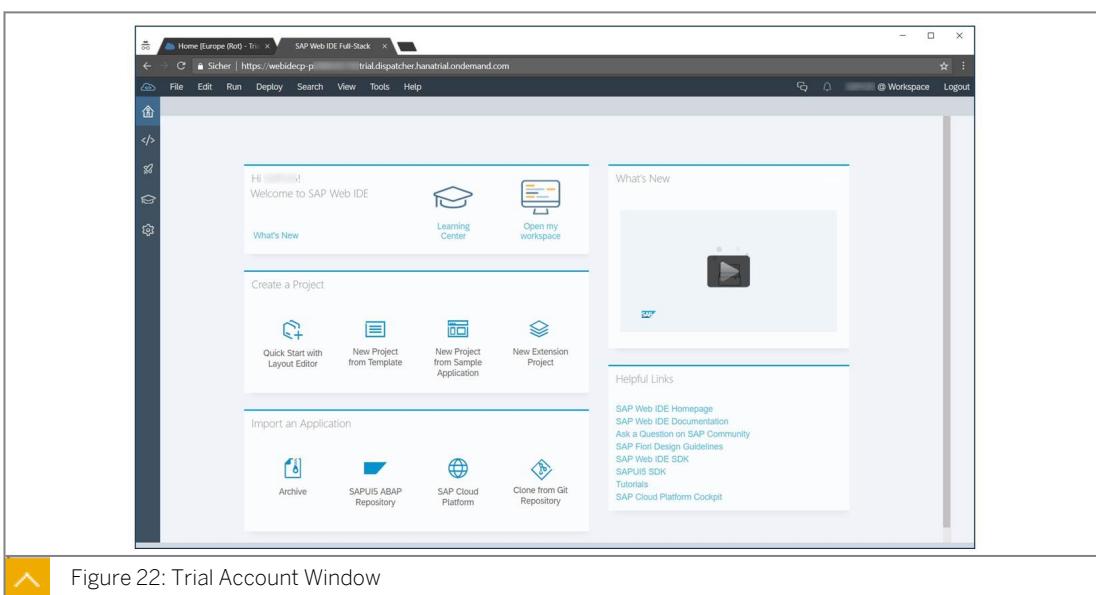
### Objective:

Create a trial account for the SAP Cloud Platform. Start the *SAP Web IDE*.

### Business Example

The *SAP Web IDE* is used for the exercises that follow. In this exercise, you access the *SAP Web IDE* via a trial account on the SAP Cloud Platform:

1. Create a trial account for the SAP Cloud Platform.



2. Use the account created to start the *Full Stack SAP Web IDE* using this account.

## Unit 6 Solution 2

# Create a Trial Account for SAP Cloud Platform and Start the SAP Web IDE



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Objective:

Create a trial account for the SAP Cloud Platform. Start the *SAP Web IDE*.

### Business Example

The *SAP Web IDE* is used for the exercises that follow. In this exercise, you access the *SAP Web IDE* via a trial account on the SAP Cloud Platform:

1. Create a trial account for the SAP Cloud Platform.

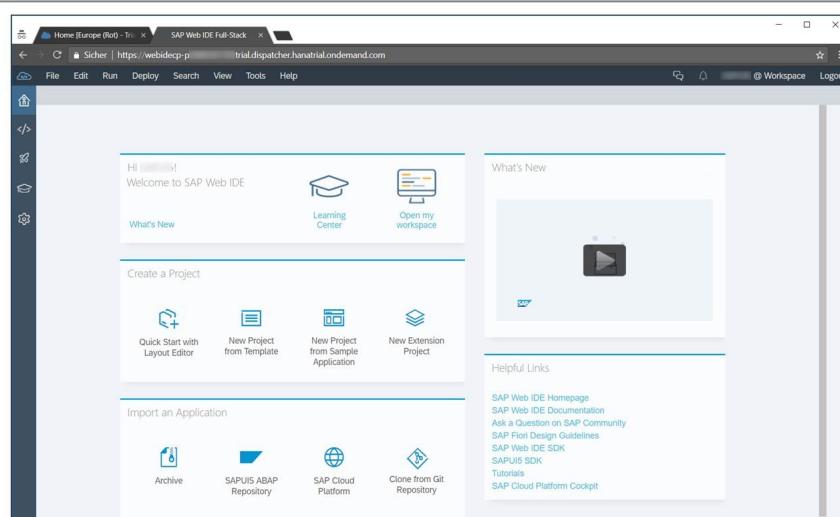


Figure 22: Trial Account Window

- a) Open the browser window and enter the following URL: <https://account.hanatrial.ondemand.com>.
- b) Choose *Register* on the web page to create a new account.
- c) At a minimum, enter your last name and e-mail address. Assign a password for your account.



## Note:

To activate the trial account, you need to access the entered email account. If you do not have an email account that you can access during training, you can create one, for example at <https://www.google.com/gmail/>.

- d) Enter your preferences for contact by SAP and select the two check boxes to accept SAP's *Privacy Statement* and *Terms and Conditions*.

Contact Preferences

In addition to communications that will result from this registration, would you also like to receive news and event notifications from SAP that are specific to your interests?

By e-mail \*

Yes  No

By telephone \*

Yes  No

Terms and Conditions

I acknowledge that I have read [SAP's Privacy Statement](#) \*

I have read and understood the Terms and Conditions of [SAP Cloud Platform](#). \*

\*Required

**Register**

Figure 23: Contact Preferences

- e) Confirm your input with the *Register* button.

- f) Select the correct images from a *captcha* dialog that appears and choose *Verify*.

Select all images with  
**traffic lights**

**VERIFY**

Figure 24: Select Captcha Images

- g) Close the window that confirms your registration.

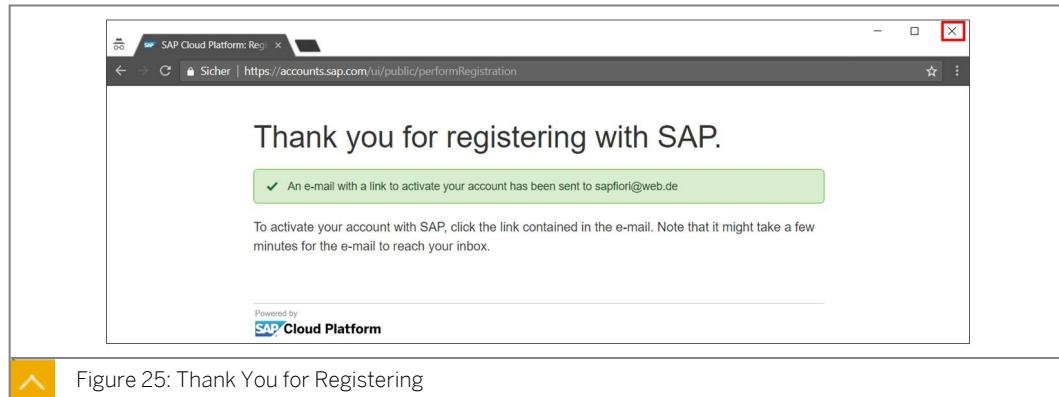


Figure 25: Thank You for Registering

2. Use the account created to start the *Full Stack SAP Web IDE* using this account.
  - a) Check your registered email account and select *Click here to activate your account*. The link opens to activate your free trial.

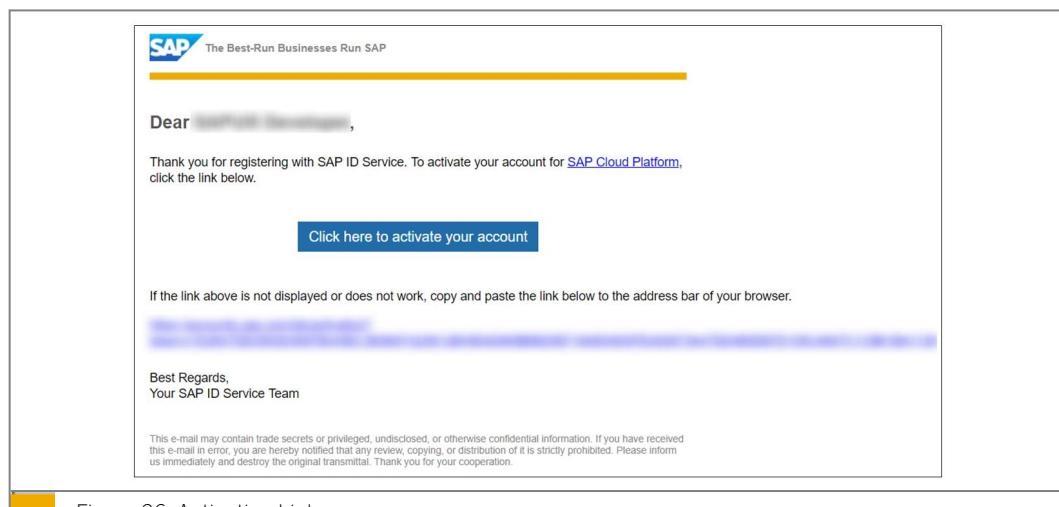


Figure 26: Activation Link

- b) To close the confirmation window, choose *Continue*.

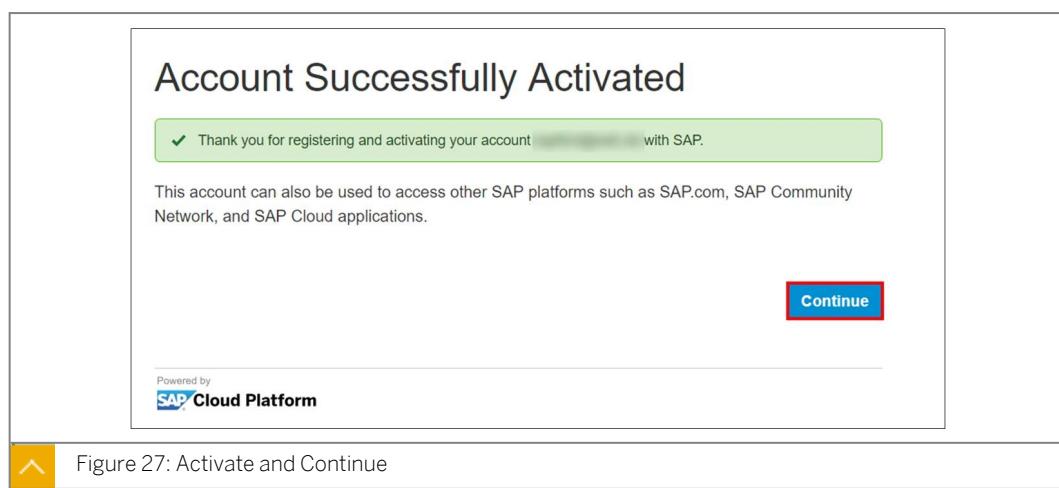


Figure 27: Activate and Continue

- c) Check that you are automatically logged into your new account and confirm any popups.

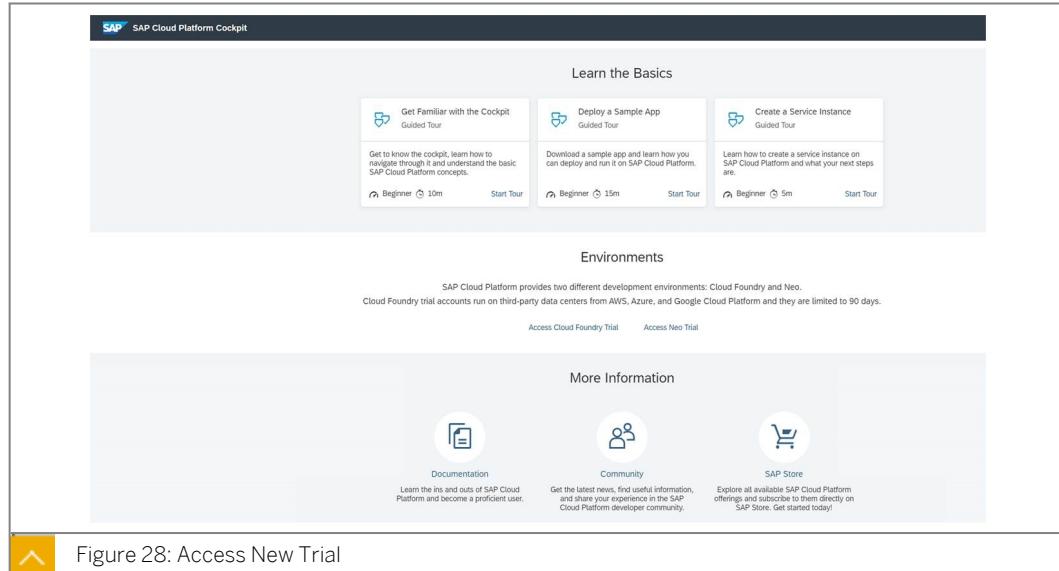


Figure 28: Access New Trial

- d) Scroll down to the environments and choose Access New Trial.
  - e) To start the SAP Web IDE, select the Neo Trial hyperlink.
  - f) Choose Services on the SAP Cloud Platform Cockpit.
  - g) In the text input field, enter **web** and, in the DevOps screen area, choose the SAP Web IDE Full-Stack tile.
  - h) Select the Go to Service link to start the SAP Web IDE and display in the browser.
- Look for your trial account window.

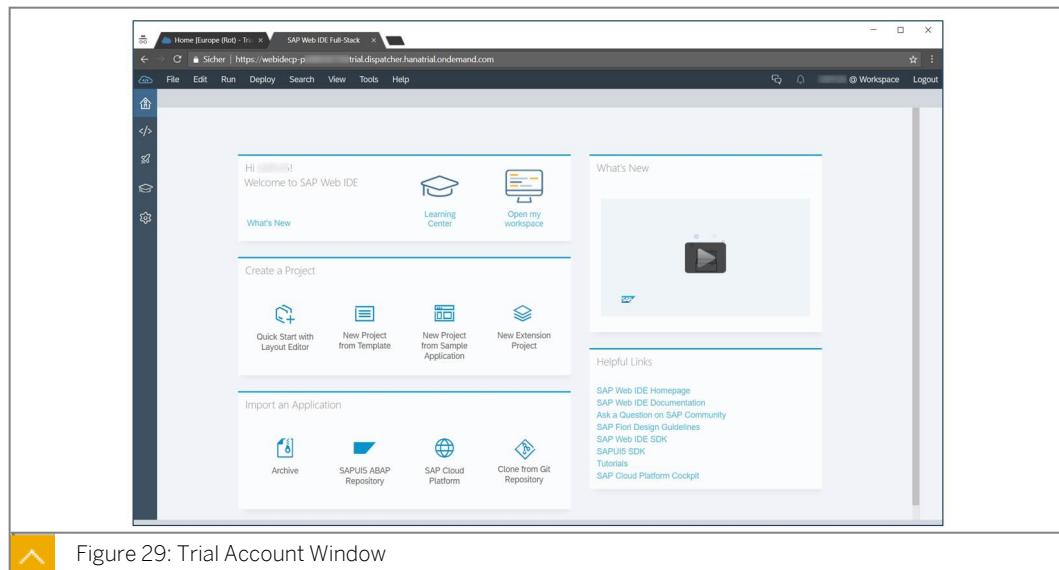


Figure 29: Trial Account Window

## Unit 6

### Exercise 3

# Create a Simple SAPUI5 Application with the help of the Layout Editor



#### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and names such as *buttons* and *fields* may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises. The graphical layout editor of the SAP Web IDE is only available in Google Chrome.

Objective: Create and test a simple SAPUI5 application in the SAP Web IDE. Use the graphical layout editor to edit the application views.

#### Business Scenario:

In this exercise, you create an SAPUI5 project with an XML view in the SAP Web IDE. In the generated view, you enter the text “Hello World!” using a **Text UI** element. You also set the title of the displayed page to **SAP Web IDE**. To do this, you edit the view with the graphical Layout Editor. You then test the application in the browser:

1. Open the development perspective of the SAP Web IDE.
2. In the SAP Web IDE create a new project. Use the following data and replace ## with your student number:

Property	Value
Project Name:	helloworld
Namespace:	student##.sap.training
View type:	XML
View name:	Main
SAPUI5 version:	1.60

3. Open the *Main* view in the graphical Layout Editor generated by the wizard. Add a **Text UI** element to the view and set its **Text** property to the value `Hello World`. Also set the title of the page to **SAP Web IDE**.
4. Adjust the bootstrap script so that **SAPUI5** is loaded from a Content Delivery Network (CDN) externally and no longer loaded from the web server locally. Use the following URL  
<https://sapui5.hana.ondemand.com/resources/sap-ui-core.js>.



Note:

The relative paths for loading the **SAPUI5** are different in the cloud environment than on an **ABAP** server. In the next exercise, you deploy your SAPUI5 application on an ABAP server and test it there. You will load the SAPUI5 externally via a CDN, so that the application can run without path adjustment in both the cloud environment and on an ABAP server.



Note:

The latest SAPUI5 version is now loaded via the URL you defined. This procedure can have an impact on the stability of an application in new releases, so only follow this procedure for test purposes. You can load a specific SAPUI5 version using a version URL, for example <https://sapui5.hana.ondemand.com/1.38.8/resources/sap-ui-core.js>

5. Test your application by starting it from the *SAP Web IDE*.



Note:

In some environments the pop-up blocker of your browser may prevent the start of the application. Please allow pop-ups so the application can run.

## Unit 6 Solution 3

# Create a Simple SAPUI5 Application with the help of the Layout Editor



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and names such as *buttons* and *fields* may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises. The graphical layout editor of the SAP Web IDE is only available in Google Chrome.

Objective: Create and test a simple SAPUI5 application in the SAP Web IDE. Use the graphical layout editor to edit the application views.

### Business Scenario:

In this exercise, you create an SAPUI5 project with an XML view in the SAP Web IDE. In the generated view, you enter the text “**Hello World!**” using a **Text UI** element. You also set the title of the displayed page to **SAP Web IDE**. To do this, you edit the view with the graphical Layout Editor. You then test the application in the browser:

1. Open the development perspective of the SAP Web IDE.
  - a) Start the SAP Web IDE.
  - b) Click on the < > icons and the development perspective will appear.

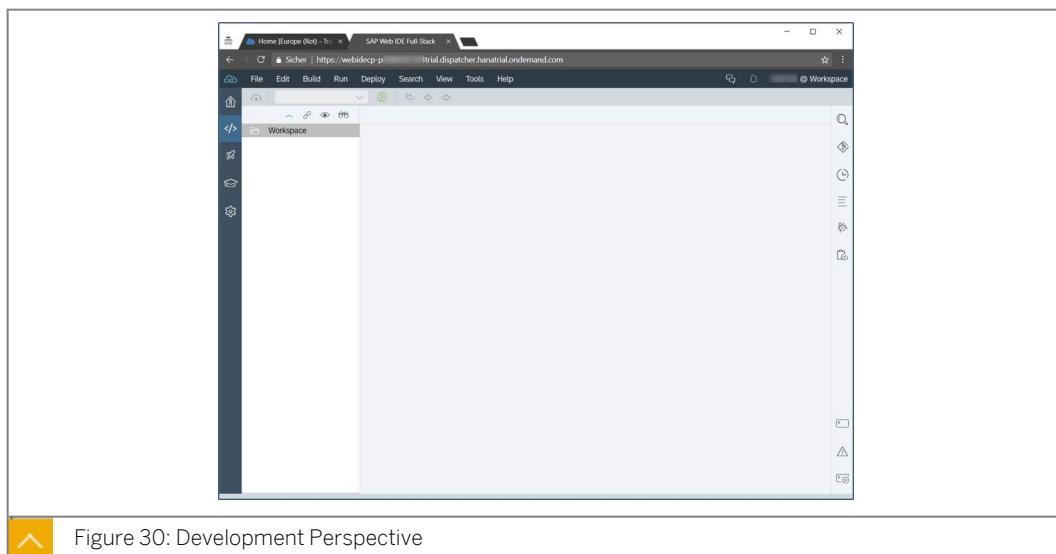


Figure 30: Development Perspective

2. In the SAP Web IDE create a new project. Use the following data and replace ## with your student number:

Property	Value
Project Name:	helloworld
Namespace:	student##.sap.training
View type:	XML
View name:	Main
SAPUI5 version:	1.60

- a) Choose the following path *File* → *New* → *Project from Template* from the SAP Web IDE menu.

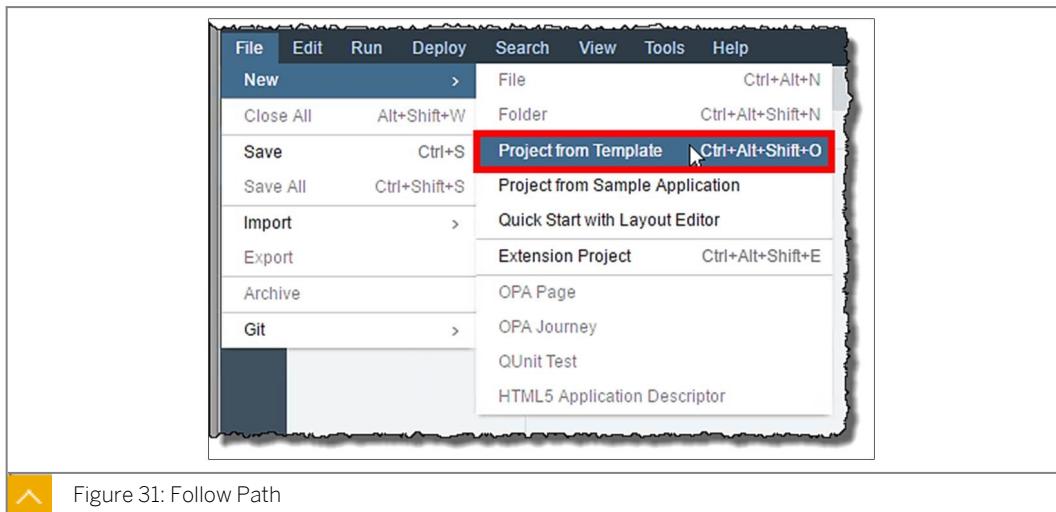


Figure 31: Follow Path

- b) Select the SAPUI5 Application Project template, and choose *Next*.

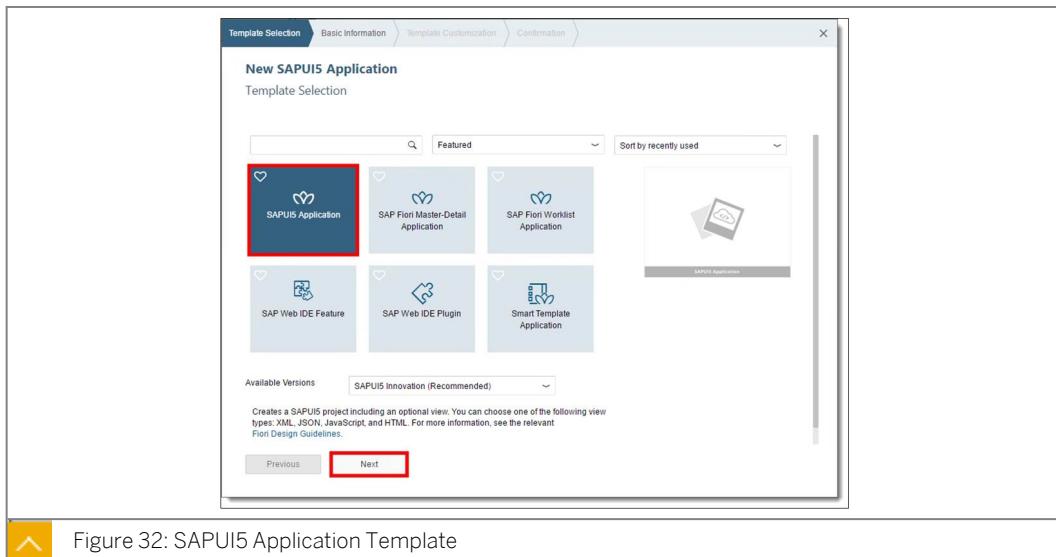
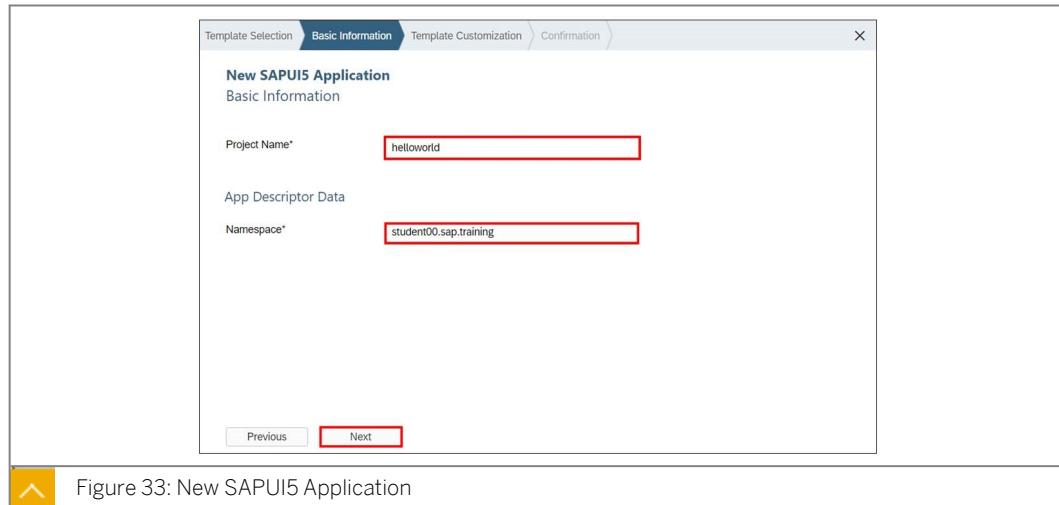
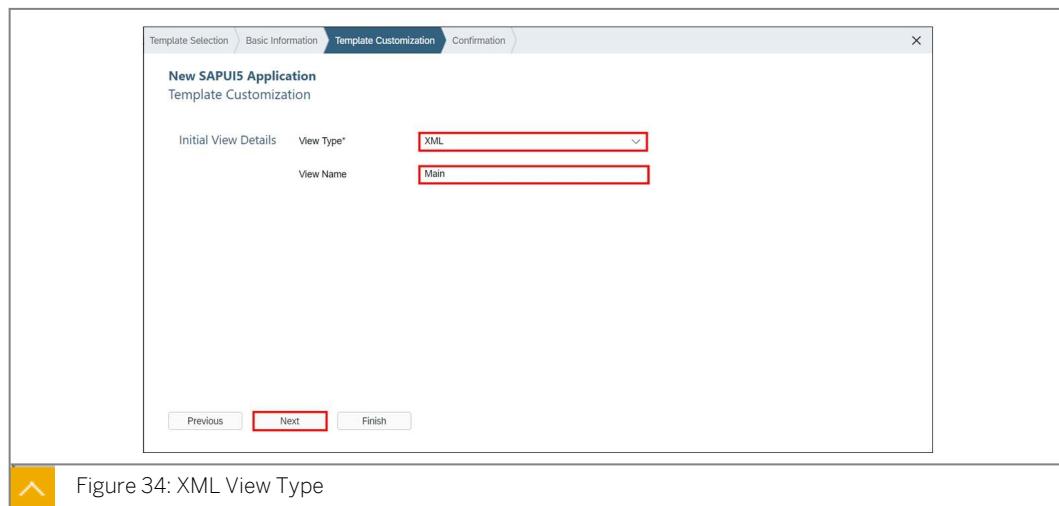


Figure 32: SAPUI5 Application Template

- c) Enter **helloworld** in the *project name* field and **student##.sap.training** in the *namespace* field. Replace the group number **00** with your assigned group number and choose the *next*.



- d) Choose XML as the *View Type* to be generated. Enter **Main** as the name of the view. Choose *next*.



- e) Close the wizard by choosing *Finish*.
3. Open the *Main* view in the graphical Layout Editor generated by the wizard. Add a *Text* UI element to the view and set its **Text** property to the value *Hello World*. Also set the title of the page to *SAP Web IDE*.
- a) Open the *Main.view.xml* in the layout editor by choosing: *Open With → Layout Editor*.

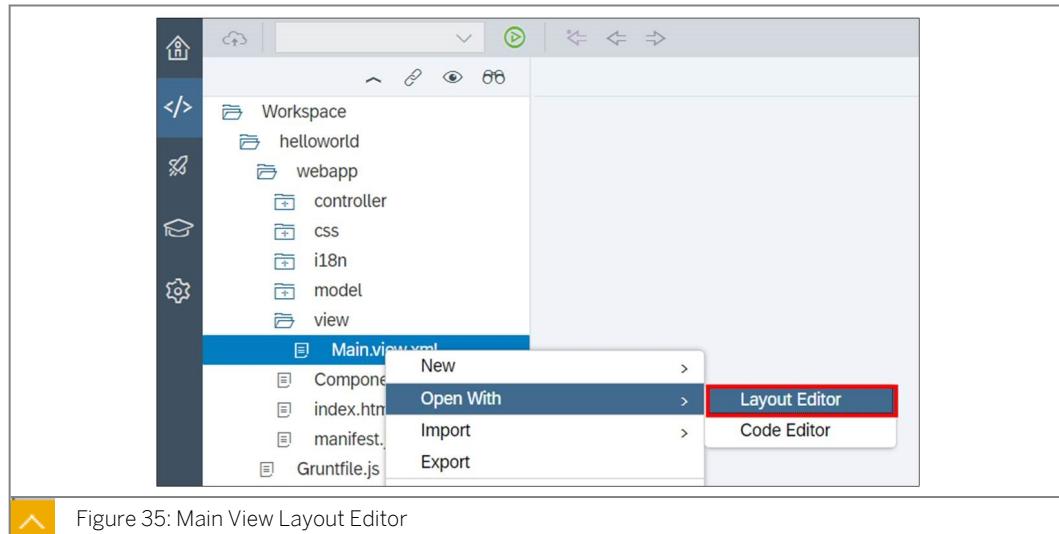


Figure 35: Main View Layout Editor

- b) Expand the display section on the *Control* tab and drag and drop the *Text UI* element from this section to the *Main view*.

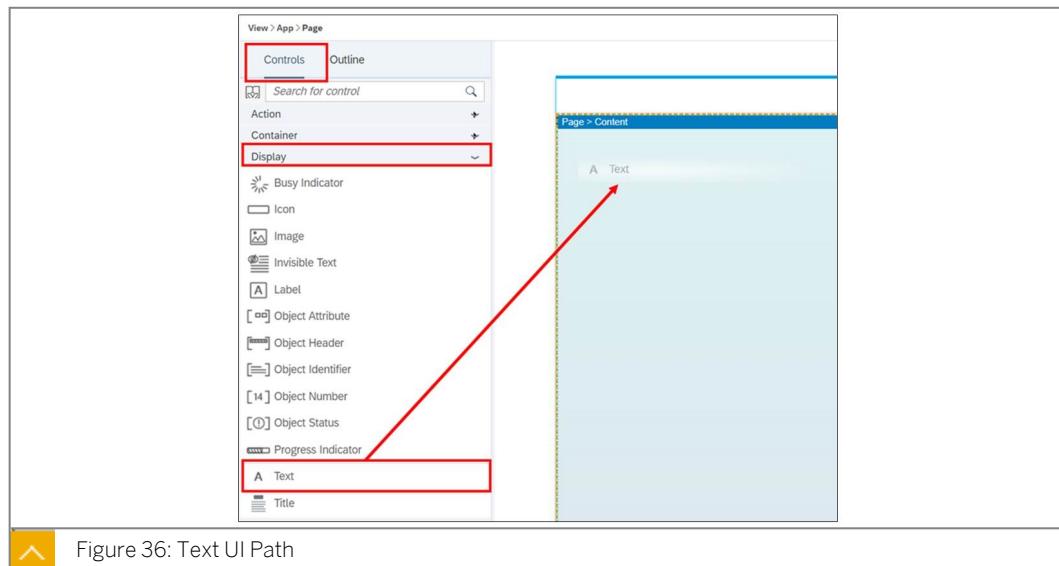
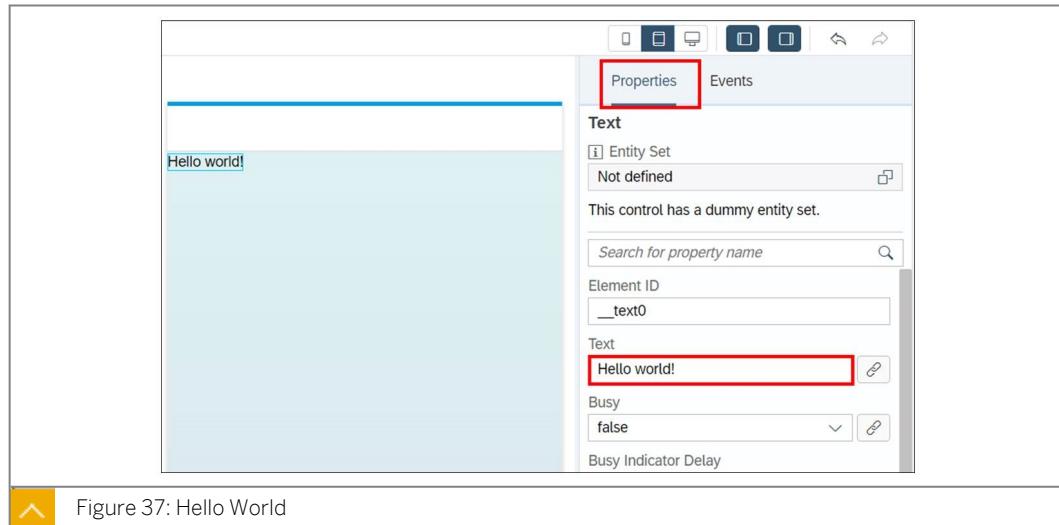
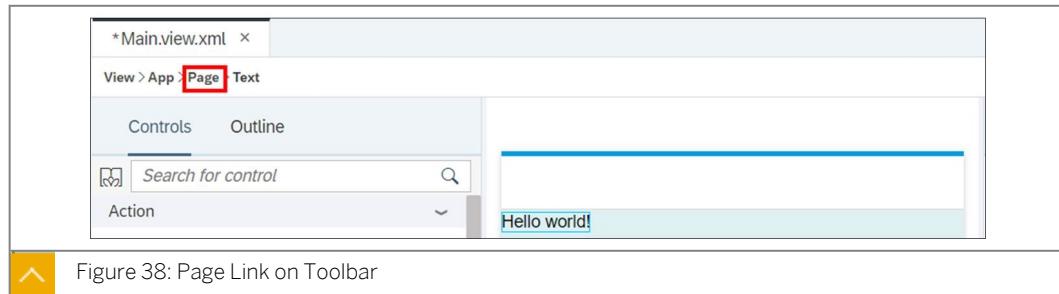


Figure 36: Text UI Path

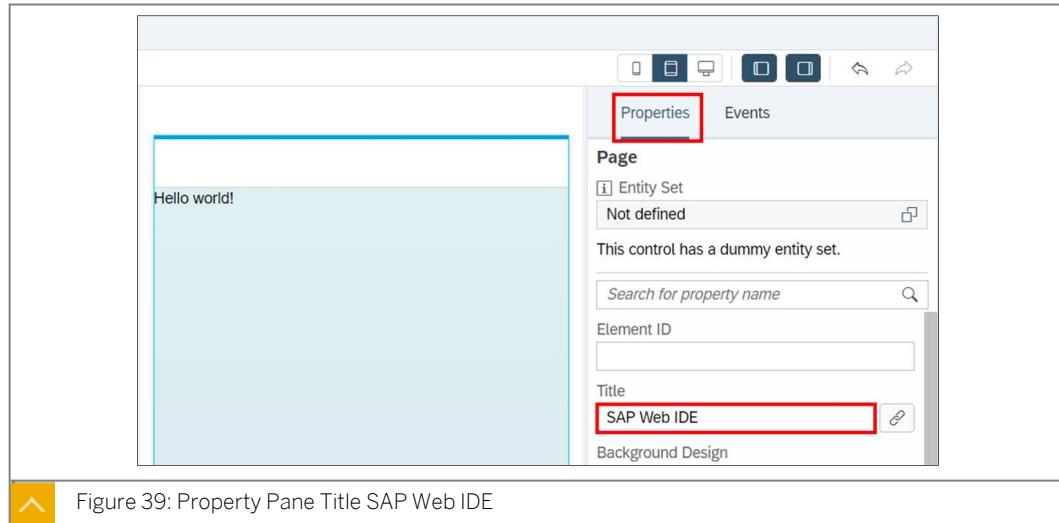
- c) In the *Properties Pane* enter the value **Hello World!** for the text property of the *Text UI* element.



- d) Select the associated *Page UI* element in the *Toolbar* to adjust the title of the page.



- e) In the *Properties Pane*, enter **SAP Web IDE** for the *Title* property of the *Page UI* element and Save your changes.



4. Adjust the bootstrap script so that **SAPUI5** is loaded from a Content Delivery Network (CDN) externally and no longer loaded from the web server locally. Use the following URL <https://sapui5.hana.ondemand.com/resources/sap-ui-core.js>.



## Note:

The relative paths for loading the **SAPUI5** are different in the cloud environment than on an **ABAP** server. In the next exercise, you deploy your SAPUI5 application on an ABAP server and test it there. You will load the SAPUI5 externally via a CDN, so that the application can run without path adjustment in both the cloud environment and on an ABAP server.

- a) Open the *index.html* file. In the bootstrap script, change the value of the *src* attribute to the following URL: <https://sapui5.hana.ondemand.com/resources/sap-ui-core.js>

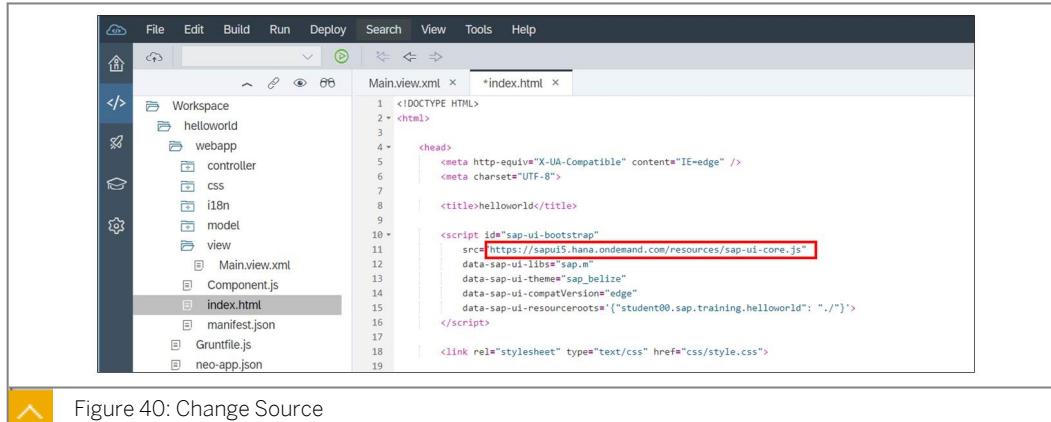


Figure 40: Change Source

- b) Save your changes.



## Note:

The latest SAPUI5 version is now loaded via the URL you defined. This procedure can have an impact on the stability of an application in new releases, so only follow this procedure for test purposes. You can load a specific SAPUI5 version using a version URL, for example <https://sapui5.hana.ondemand.com/1.38.8/resources/sap-ui-core.js>

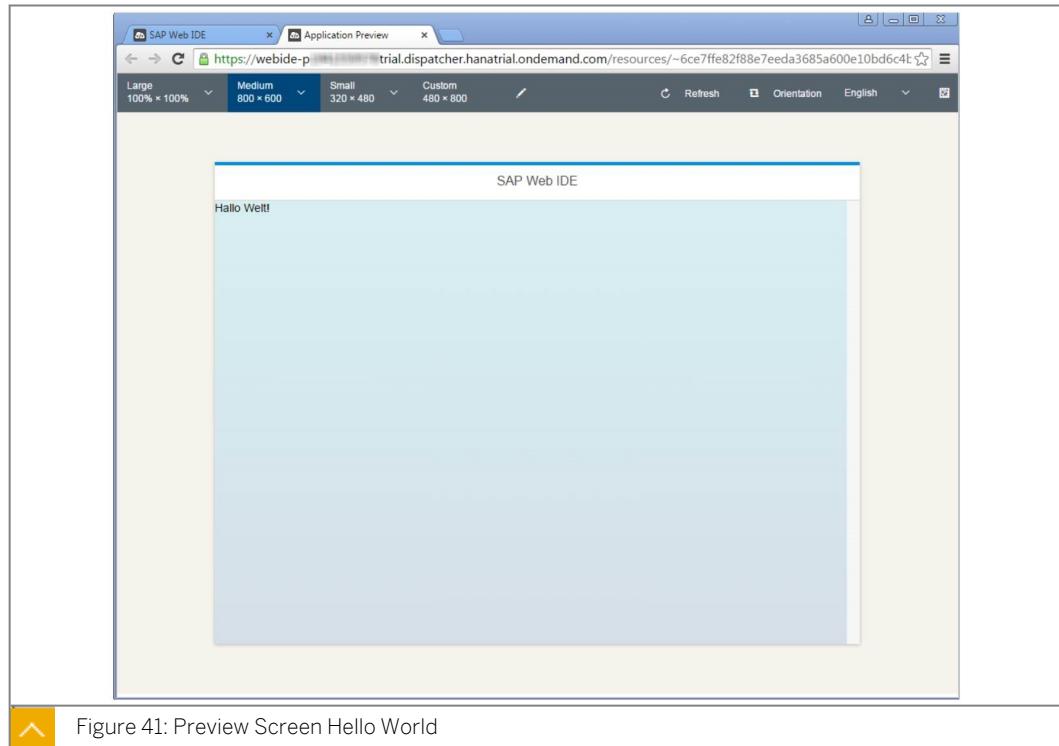
## 5. Test your application by starting it from the SAP Web IDE.



## Note:

In some environments the pop-up blocker of your browser may prevent the start of the application. Please allow pop-ups so the application can run.

- a) Select the *index.html* file and start the application by choosing the *Run* button in the toolbar of the SAP Web IDE.
- b) Check whether the adjustments you made on the *Main View* appear in the preview screen.



## Unit 6 Exercise 4

# Set up the SAP Cloud Connector



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and names of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business Scenario

Set up the SAP Cloud Connector to connect the SAP Cloud Platform to your on-premise SAP System.

#### Task 1: Set up the SAP Cloud Connector.

1. Start the *SAP Cloud Connector Administration* site by browsing in Chrome to the URL <https://localhost:8443> or choose the bookmarked Local WTS Tools folder.
2. Complete the *SAP Cloud Connector* login dialog box.
3. Map the virtual host from the destination to the ABAP training system.
4. Specify the URL paths to be called in the training system.

#### Task 2: Run the Initialize Course script

For several exercises, you are required to download files. Additional resources are also available.

1. To access the files and additional resources for the exercises in this course, run the Initialize Course script.

## Set up the SAP Cloud Connector



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and names of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business Scenario

Set up the SAP Cloud Connector to connect the SAP Cloud Platform to your on-premise SAP System.

#### Task 1: Set up the SAP Cloud Connector.

1. Start the *SAP Cloud Connector Administration* site by browsing in Chrome to the URL <https://localhost:8443> or choose the bookmarked Local WTS Tools folder.
  - a) To start the *SAP Cloud Connector Administration* site, choose the *LocalWTSTools* from the bookmarks bar of Chrome.
  - b) Choose *SAP Cloud Connector*.
  - c) Accept warnings that the site is untrusted. The site is hosted locally without a valid signed certificate and this can be expected. Choose *Advanced* and *Proceed to localhost*.
2. Complete the *SAP Cloud Connector* login dialog box.
  - a) Enter **Administrator** in the *User Name* field.
  - b) Enter **Welcome** in the *Password* field.
  - c) Choose *Login*.

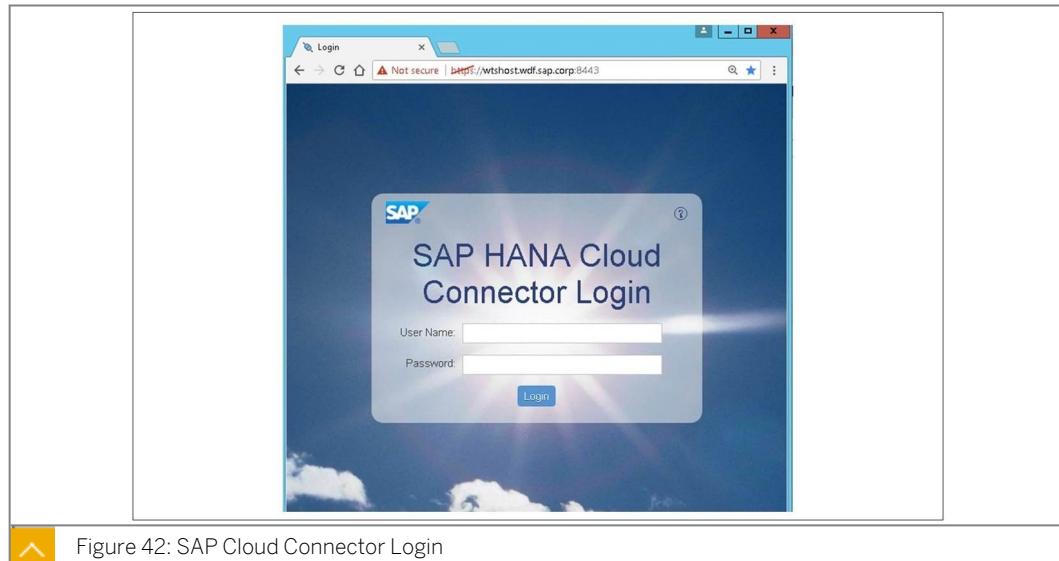


Figure 42: SAP Cloud Connector Login

- d) Choose *Add Subaccount* and the *Add Subaccount* dialog box opens.

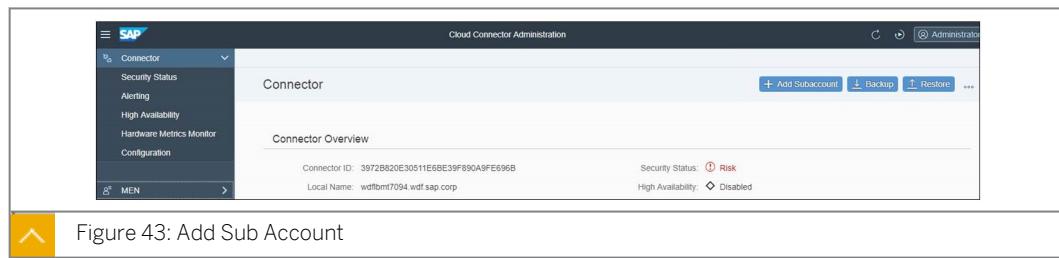


Figure 43: Add Sub Account

- e) Choose **Europe (Rot) -Trial** as the region.  
 f) Enter your SCP account information.  
 g) Choose Save to confirm your values.



Note:  
 The Account Name is shown in the top left corner of the SAP Cloud Platform Cockpit.

3. Map the virtual host from the destination to the ABAP training system.
- Choose the *Access Control* Tab.
  - Choose *Cloud To On-Premise* in the Cloud-Connector front end for your SAP Cloud Platform Account.
  - Choose *Add* on the toolbar in the *Mapping Virtual To Internal System* section of the screen.
  - Select ABAP System in the *Back-end Type* field and choose *Next*.
  - Select HTTPS in the protocol field.
  - Enter **fsdhost.wdf.sap.corp** in the *Internal Host* field and **44310** in the *Internal Port* field and then choose *Next*.
  - Enter **fsd.virtual** in the *Virtual Host* field and **8000** in the *Virtual Port* field.

- h) Choose **None** in the *Principal Type* field, and then *Next*.
- i) Choose *Use Virtual Host* in the *Host In Request Header* field and choose *Next*.



Note:

You can also optionally enter a description.

- j) Choose *Next* and *Finish* and exit the wizard.
4. Specify the URL paths to be called in the training system.
- a) Select the created entry in the *Mapping Virtual To Internal System* section of the screen.
  - b) Choose to *Add* on the *Resource Accessible On fsd.virtual:8000* section of the screen.
  - c) Enter **/sap** in the *URL Path* field, and choose the *Path and all sub-paths* radio button and *Save*.

### Task 2: Run the Initialize Course script

For several exercises, you are required to download files. Additional resources are also available.

1. To access the files and additional resources for the exercises in this course, run the Initialize Course script.
  - a) In your training view, choose *Start* → *More*.
  - b) Choose *Initialize UT4.2*.  
The initialization script starts. It may take several seconds to finish.
  - c) Choose *OK* when the information dialog is displayed.
  - d) Open File Explorer to navigate to the resources and choose: *S:/Courses/ UX410\_20*.

## Unit 6 Exercise 5

# Deploy an SAPUI5 Application via the SAP Web IDE on an ABAP Server



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

In this exercise, you deploy the SAPUI5 application you developed in the last exercise, onto an ABAP server from the SAP Web IDE. To do this, the ABAP system used for this training course must be made accessible for your SAP Cloud Platform account through the SAP HANA Cloud Connector. The connection to your account is established through a destination, which you create in the SAP Cloud Platform Cockpit.

1. In the SAP Cloud Platform cockpit, create a destination with the following properties:

Table 1: Destination Table Configuration

Configuration	Value
Name	FDS_100
Type	HTTP
Description	FSD
URL	http://fsd.virtual:8000
Proxy Type	OnPremise
Authentication	NoAuthentication

Table 2: Add These Properties to the Destination

Properties	Value
WebIDEEEnabled	true
WebIDEUsage	dev_abap,ui5_execute_abap,bsp_execute_abap,odata_abap

2. Deploy the SAPUI5 application you developed in the previous exercise from the SAP Web IDE to the ABAP system used in the training course. To do this, create a new application called **ZUI5\_WEBIDE\_##**, where ## stands for your group number. Assign the application to the existing package **ZTRAIN\_##** (## = group number) and to the transport request that already exists for the training course.

3. Track the deployment process that now starts in the SAP Web IDE console.
4. After deployment, call up the application on the ABAP server using the following URL (replace ## with your group number):  
[http://fsdhost.wdf.sap.corp:8000/sap/bc/ui5\\_ui5/sap/zui5\\_webide\\_##/index.html](http://fsdhost.wdf.sap.corp:8000/sap/bc/ui5_ui5/sap/zui5_webide_##/index.html)



Hint:

You can also start transaction SICF on the frontend server, search for your application. Select the found entry and choose from the context menu Test Service.

# Deploy an SAPUI5 Application via the SAP Web IDE on an ABAP Server



Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

In this exercise, you deploy the SAPUI5 application you developed in the last exercise, onto an ABAP server from the SAP Web IDE. To do this, the ABAP system used for this training course must be made accessible for your SAP Cloud Platform account through the SAP HANA Cloud Connector. The connection to your account is established through a destination, which you create in the SAP Cloud Platform Cockpit.

1. In the SAP Cloud Platform cockpit, create a destination with the following properties:

Table 1: Destination Table Configuration

Configuration	Value
Name	FDS_100
Type	HTTP
Description	FSD
URL	http://fsd.virtual:8000
Proxy Type	OnPremise
Authentication	NoAuthentication

Table 2: Add These Properties to the Destination

Properties	Value
WebIDEEEnabled	true
WebIDEUsage	dev_abap,ui5_execute_abap,bsp_execute_abap,odata_abap

- a) To create a new destination, on the left side of the SAP Cloud Platform cockpit screen, choose *Connectivity* → *Destinations*.
- b) In the screen that appears on the right, choose the *New Destination* button.

- c) In the displayed form, enter the configuration values for the destination from the Destination Table.



Note:

The entered URL defines a virtual host and is therefore freely definable. In the SAP Cloud Connector, this virtual host is mapped to the ABAP system to be used.

- d) Choose the *New Property* button to add further properties to the destination.
- e) In the displayed input fields, enter **WebIDEEnabled** as the property name and **true** as the property value.



Note:

When maintaining the properties, pay attention to upper and lower case.

The individual values for the **WebIDEUsage** property are separated by a comma only (no spaces).

The figure does not show all the values to be entered for the **WebIDEUsage** property.

To deploy an SAPUI5 application on an ABAP server, not all the entered values for the **WebIDEUsage** property are required. To keep things simple, all the values needed during the training course are included. For example, value **odata\_abap** is not needed until you access the OData services of the ABAP system.

- f) Choose the *New Property* button again to add a further property.
- g) In the added input fields, enter **WebIDEUsage** as the property name and **dev\_abap,ui5\_execute\_abap,bsp\_execute\_abap,odata\_abap** as the property value.
- h) Save your destination by choosing *Save*.
2. Deploy the SAPUI5 application you developed in the previous exercise from the SAP Web IDE to the ABAP system used in the training course. To do this, create a new application called **ZUI5\_WEBIDE\_##**, where **##** stands for your group number. Assign the application to the existing package **ZTRAIN\_##** (**##** = group number) and to the transport request that already exists for the training course.
- a) In the context menu of your SAPUI5 project *helloworld*, choose the following path: *Deploy* → *Deploy to SAPUI5 ABAP Repository*.
- b) Ensure the option *Deploy a new application* is selected on the displayed form, and select the system **FSD\_100 - FSD** configured above from the drop-down menu. In the dialog box that is displayed, enter the user name and password for the ABAP system, remembering to replace **##** with your group number. Confirm your entries by choosing *Sign in*.
- c) On successful login, choose the *Next* button.

- d) Enter **ZUI5\_WEBIDE\_##** (where ## is your group number), as the name of the new application and enter **SAPUI5** in SAP Web IDE as the description. Browse to select the existing package **ZTRAIN\_##**, (where ## is your group number), and exit editing of the window by choosing *Next*.
  - e) On the next window, ensure the transport request relating to the training course is selected, and choose *Next*.
  - f) Close the wizard by choosing *Finish*.
3. Track the deployment process that now starts in the SAP Web IDE console.
    - a) Select the following path from the SAP Web IDE menu: *View → Console*.
  4. After deployment, call up the application on the ABAP server using the following URL (replace ## with your group number):  
[http://fsdhost.wdf.sap.corp:8000/sap/bc/ui5\\_ui5/sap/zui5\\_webide\\_##/index.html](http://fsdhost.wdf.sap.corp:8000/sap/bc/ui5_ui5/sap/zui5_webide_##/index.html)



Hint:

You can also start transaction SICF on the frontend server, search for your application. Select the found entry and choose from the context menu Test Service.

## Unit 6

### Exercise 6

# Import an SAP BUILD Prototype into SAP Web IDE



#### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

#### Objective:

Import a prototype created with SAP Build into the SAP Web IDE.

#### Business Scenario

You import a prototype created with SAP Build to the SAP Web IDE to undertake further development.

You require the Prototype created in exercise 1.

#### Download a SAP Build Prototype to the SAP Web IDE.

1. Download the SAP Build prototype created in exercise 1 as a *SAP Web IDE* project.
2. Import the downloaded prototype into the *SAP Web IDE Full-Stack* edition. Use SAPBuildPrototype as the project name for the import.
3. Test the imported project with mock data. Create a runtime configuration with the name SAP Build prototype and use the html file with the name testFLPServiceMockServer.html located in the folder of your test project.

# Import an SAP BUILD Prototype into SAP Web IDE



**Note:**

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

**Objective:**

Import a prototype created with SAP Build into the SAP Web IDE.

**Business Scenario**

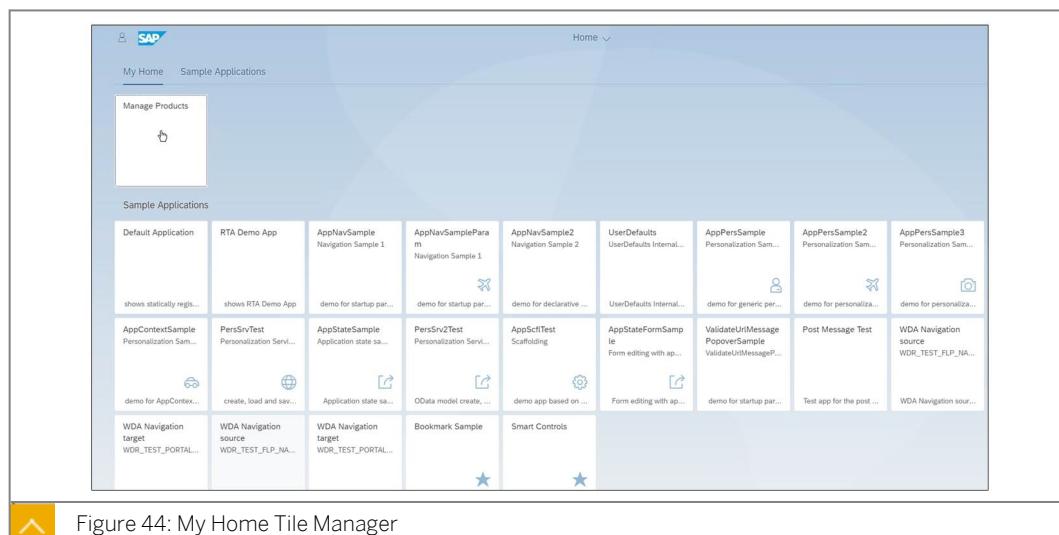
You import a prototype created with SAP Build to the SAP Web IDE to undertake further development.

You require the Prototype created in exercise 1.

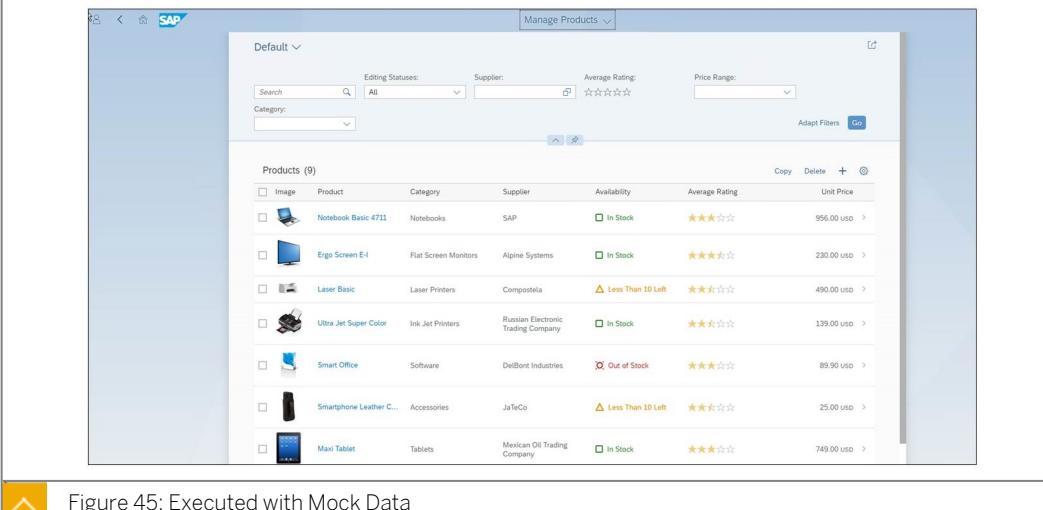
**Download a SAP Build Prototype to the SAP Web IDE.**

1. Download the SAP Build prototype created in exercise 1 as a *SAP Web IDE* project.
  - a) Open <https://www.build.me> in Google Chrome.
  - b) Log in with your email address and choose *Proceed*.
  - c) Enter your email address and password in the corresponding fields and log in.
  - d) On log in, view your current projects in the *SAP Build* workspace.
  - e) Open the *Manage Products* project by double-clicking the project tile or selecting the three dots of the tile, and choosing *Open* to see the content of the prototype.
  - f) Select the *Manage Products* prototype to open it.
  - g) Select the *Share* button in the page overview.
  - h) In the new dialog box, choose *Download Zip of Prototype* in the *DOWNLOAD* tab.
  - i) Close the dialog window when the file has downloaded to your browser window.
2. Import the downloaded prototype into the *SAP Web IDE Full-Stack* edition. Use *SAPBuildPrototype* as the project name for the import.
  - a) Start the SAP Web IDE as shown in the previous exercise.

- b) Choose from the context menu *Import* → *File or Project* from the Workspace folder of your project overview.
  - c) Browse and select the file you downloaded in the previous task and insert the value **SAP Build Prototype** in the *Import to* field.
  - d) After the import finishes successfully, you will see the newly imported project in the *SAP Web IDE*. Investigate the files of the imported project.
3. Test the imported project with mock data. Create a runtime configuration with the name SAP Build prototype and use the html file with the name `testFLPServiceMockServer.html` located in the folder of your test project.
- a) Select the project root folder *SAPBuildPrototype* and choose from the context menu *Run* → *Run Configuration*.
  - b) Select the **+** symbol to add a new run-configuration in the new dialog box.
  - c) Choose the option to *Run as Web Application* and a new default runtime configuration is created in the context menu.
  - d) Change the value in the *Name* field to **SAP Build prototype**.
  - e) Choose to execute the option `testFLPServiceMockServer.html` from the list of available files in the test folder of your project.
  - f) Select the checkbox *Run with mock data*.
  - g) Choose *Save* and *Run* to start the configuration.
  - h) Choose the *Manage Products* tile in the *My Home* section to view the application executed with mock data.



- i) Explore the application. Not all aspects of the prototype are fully implemented.



The screenshot shows the SAP Web IDE interface for managing products. The title bar says "Manage Products". The main area is titled "Default" and contains a table of products. The table has columns for Image, Product, Category, Supplier, Availability, Average Rating, and Unit Price. The products listed are:

Image	Product	Category	Supplier	Availability	Average Rating	Unit Price
	Notebook Basic 4711	Notebooks	SAP	<span>In Stock</span>	<span>★★★★☆☆</span>	956.00 USD
	Ergo Screen E-4	Flat Screen Monitors	Alpine Systems	<span>In Stock</span>	<span>★★★★☆☆</span>	230.00 USD
	Laser Basic	Laser Printers	Compostela	<span>Less Than 10 Left</span>	<span>★★★☆☆☆</span>	490.00 USD
	Ultra Jet Super Color	Ink Jet Printers	Russian Electronic Trading Company	<span>In Stock</span>	<span>★★★★☆☆</span>	139.00 USD
	Smart Office	Software	DeBont Industries	<span>Out of Stock</span>	<span>★★★★☆☆</span>	89.90 USD
	Smartphone Leather C...	Accessories	JaTeCo	<span>Less Than 10 Left</span>	<span>★★☆☆☆☆</span>	25.00 USD
	Maxi Tablet	Tablets	Mexican Oil Trading Company	<span>In Stock</span>	<span>★★★★☆☆</span>	749.00 USD

Below the table are buttons for Copy, Delete, and a plus sign. The footer of the interface shows a navigation bar with icons for Home, SAP, and Help.

Figure 45: Executed with Mock Data

## Unit 7

### Exercise 7

# Work with Diagrams



**Note:**

Please be aware that in some screenshots the code of previous implementation steps is not shown. Please do not delete any code that you have added in previous steps in the following steps.



**Note:**

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

#### **Objective:**

Create a SAPUI5 project to view data with graphs and tables.

#### **Business Scenario**

Implement a simple analytical application to visualize your data with graphs and tables. Use selection criteria to choose the data to display.

Follow the guidelines to create a full-screen application with a *Main* view.

#### **Task 1: Create a SAPUI5 project to view data with graphs and tables**

1. Create a SAPUI5 project with the following settings:

Parameter	Value
Project name	diagram
Namespace	student##.sap.training
Viewtype	XML
View name	Main
SAPUI5 Version	1.60

#### **Task 2: Implement the model**

1. Create a folder with the name localService inside the webapp folder.
2. Add a new DataSource configuration by choosing the `metadata.xml` file located at `s:\Courses\UX410_20\templates\diagram\metadata.xml` as a service document and assigning a model with the name SalesModel to the newly created DataSource.

3. Enter mock data.

### Task 3: Implement a line graph

1. Display data with a line graph by replacing the generated Page control in the *Main.view.xml* with a control of type *sap.f.DynamicPage*. Add the following attributes to the newly created control.

Attribute	Value
id	dynamicPageId
headerExpanded	true
toggleHeaderOnTitleClick	true

2. Add a *sap.f.DynamicPageTitle* control to the title aggregation of the newly added *DynamicPage* control.
3. Add a *sap.m.Title* control to the heading aggregation of the newly added *DynamicPageTitle* control.

Attribute	Value
text	Sales figures



Note:

Add the value as an attribute of your **i18n** property file. Use the value `pageTitle` as the key of your **i18n** file.

4. Add a *sap.suite.ui.commons.ChartContainer* to the content aggregation of the *DynamicPage* control. Use **suite** as a XML-namespace alias for the *sap.suite.ui.commons*-SAPUI5-namespace. Add the table attributes to the *ChartContainer-content* control.

Field	Value
id	idChartContainer
autoAdjustHeight	true
contentChange	onContentChange
showFullScreen	true
showLegend	true
showPersonalization	false
title	Sales figures



Note:

Add the title of the *ChartContainerContent* as an **i18n** property and use the id **diagramTitle**.

5. Add a `sap(suite).ui.commons.ChartContainerContent` control to the content aggregation of the `ChartContainer` control. Use the attributes listed in the table. Add the content aggregation to the newly added `sap(suite).ui.commons.ChartContainerContent` control.

Attribute	Value
icon	sap-icon://line-chart
title	Line Chart



Note:

Assign the title `ChartContainerContent` to the i18n property file and use the id `lineChartTitle`.

6. Add a `sap.ui.viz.Popover` control to the content aggregation of the `ChartContainerContent` control with the id `idLineChartPopover`.
  7. Add a `sap.ui.viz.VizFrame` control after the newly added `Popover` control. Use the attributes in the following table.
- | Attribute | Value                    |
|-----------|--------------------------|
| id        | sap-icon://line-chart    |
| height    | 9.1rem                   |
| width     | 100%                     |
| uiConfig  | {applicationSet:'fiori'} |
8. Implement the `onInit` function in the controller of `Main` view. Declare a member variable with the name `sCurrentVizFrame` and assign the value `idLineChartVizFrame` as a string to the variable. Declare a member variable `sCurrentSelectedDimension` and assign the value `0` as type string.
  9. Add a reference of `sap.viz.ui5.controls.common.feeds.FeedItem` class to the controller implementation.
  10. Implement a function called `_createFeedMap`.

```

    _createFeedMap: function () {
        this.feedMap = {};
        this.feedMap.salesAmount = new FeedItem({
            "uid": "valueAxis",
            "type": "Measure",
            "values": ["SalesAmount"]
        });

        this.feedMap.products = new FeedItem({
            "uid": "categoryAxis",
            "type": "Dimension",
            "values": ["Products"]
        });

        this.feedMap.subregion = new FeedItem({
            "uid": "categoryAxis",
            "type": "Dimension",
            "values": ["Sub_Region_Name"]
        });

        this.feedMap.productsSubregion = new FeedItem({
            "uid": "categoryAxis",
            "type": "Dimension",
            "values": ["Products", "Sub_Region_Name"]
        });
    }
}

```

Figure 61: Add Reference

11. Add a reference to the `sap.viz.ui5.data.MeasureDefinition` and to the class `sap.viz.ui5.data.DimensionDefinition` to your controller implementation and implement the below function `_createDataSetMap` in your `Main` controller.
12. Invoke two new functions `_createDataSetMap` and `_createFeedMap` from inside the `onInit` function of your controller.

```

Main.controller.js ×
1 sap.ui.define([
2     "sap/ui/core/mvc/Controller",
3     "sap/viz/ui5/controls/common/feeds/FeedItem",
4     "sap/viz/ui5/data/DimensionDefinition",
5     "sap/viz/ui5/data/MeasureDefinition"
6 ], function (Controller,FeedItem,DimensionDefinition,MeasureDefinition) {
7     "use strict";
8
9     return Controller.extend("student00.sap.training.diagram.controller.Main", {
10        onInit: function () {
11            this.sCurrentVizFrame = "idLineChartVizFrame";
12            this.sCurrentSelectedDimension = "0";
13            this._createFeedMap();
14            this._createDataSetMap();
15        },
16        _createFeedMap: function () { },
17        _createDataSetMap: function () { }
18    });
19 });

```

Figure 64: Invoke Functions

13. Implement a function `_createDataSet` inside the `Main.controller.js` file. Declare a variable `oDataSet` and assign an object of type `sap.viz.ui5.data.FlattenedDataset`. Bind the Dataset to the `EntitySet` `SalesFigures` of your `SalesModel`. Invoke the newly created function in the `onInit` function of your controller. Assign the newly created dataset to a member variable with the name `dataSet`.
14. Create a function `_handleSelection` to handle the assignment of dimension and measure to the current selected diagram. The function should take a parameter with the name `selectedItem`. The implementation of the function should do the following: store

the reference of the currently visible *VizFrame* control using the stored value from variable *sCurrentVizFrame* and store the reference to the control as a local variable.

15. Create a function with name `_createLineDiagram`.
16. Test the application once the line graph is implemented. Execute using the *MockServer* option.

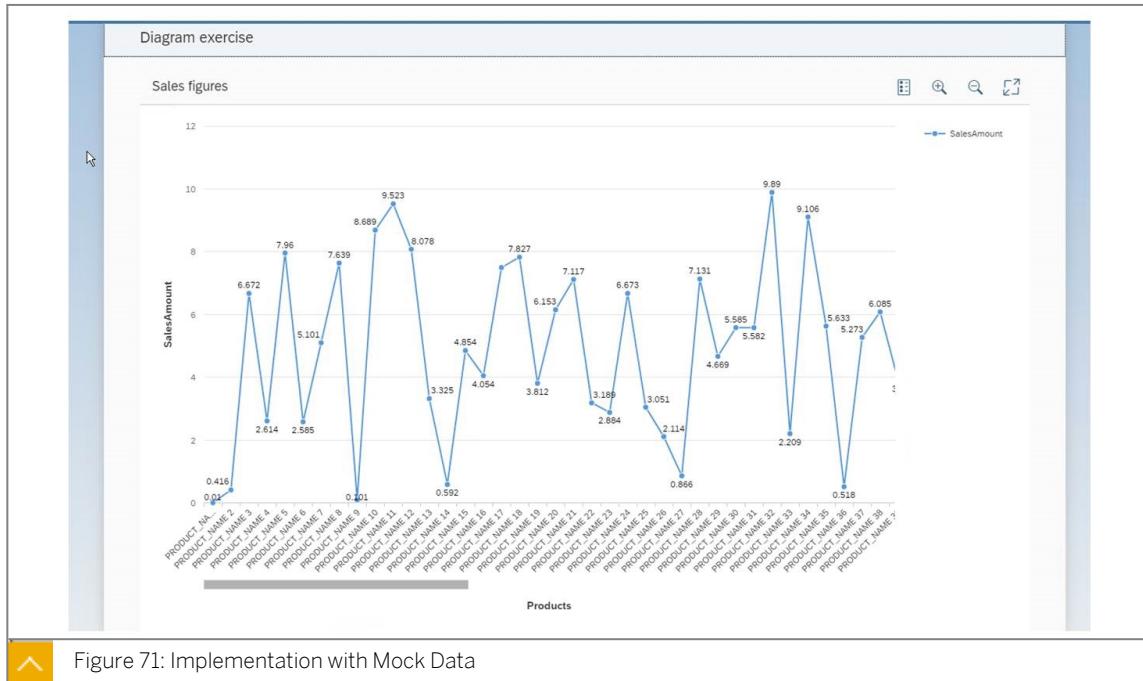


Figure 71: Implementation with Mock Data

#### Task 4: Implement a bar chart

1. Implement a bar chart. The surrounding *ChartContainerContent* control requires an attribute with the title *Column Chart*. Use file *i18n*.

Attribute	Value
id	idBarChartVizFrame
height	9.1rem
width	100%
uiConfig	{applicationSet:'fiori'}

2. Implement the function `_createColumnChart`.
3. Develop the event handler `contentChange` to switch between diagram views with the function `onContentChange`.

#### Task 5: Implement a table view

1. Create a *ChartContainerContent* with the *table-chart* icon and Sales figures as table title using the *i18n* file.
2. Create a function `_createTable`. in the controller of the *Main* view with dependencies.

Diagram exercise			
Sales figures			
Region	Sub Region	Product name	Sales Amount
REGION_NAME 1	SUB_REGION_NAME 1	PRODUCT_NAME 1	0.00985394674611232
REGION_NAME 2	SUB_REGION_NAME 2	PRODUCT_NAME 2	0.4163109025208215
REGION_NAME 3	SUB_REGION_NAME 3	PRODUCT_NAME 3	6.671921353592035
REGION_NAME 4	SUB_REGION_NAME 4	PRODUCT_NAME 4	2.6136080861097093
REGION_NAME 5	SUB_REGION_NAME 5	PRODUCT_NAME 5	7.960439208624051
REGION_NAME 6	SUB_REGION_NAME 6	PRODUCT_NAME 6	2.5848653284436645
REGION_NAME 7	SUB_REGION_NAME 7	PRODUCT_NAME 7	5.101492694600491

Figure 81: Test Application

### Task 6: Implement a dimension selection

1. Implement a dimension selection. Use a control of type sap.m.Select with the following attribute and values:

Parameter	Value
id	idDimSelector
visible	true
change	onChange

2. Create a function `_createSelector`. Use the values in the table below.

The function fills the previously created selection list with values. Extend the implementation of the function `handleSelection` to handle the different visualization facettes.

Item-Key	Item-Text
0	per Product
1	per Region
2	per Product and Region

3. To display more meaningful data in the charts rename the file `SalesFigures.json` located in the `localService` folder to `SalesFigures_Gen.json`. Import the file `SalesFigures.json` located at `s:\courses\UX410_20\templates\diagram\` into the `localService` folder. Change the Mock Data configuration of your project so that the data from the `SalesFigures.json` file is used.

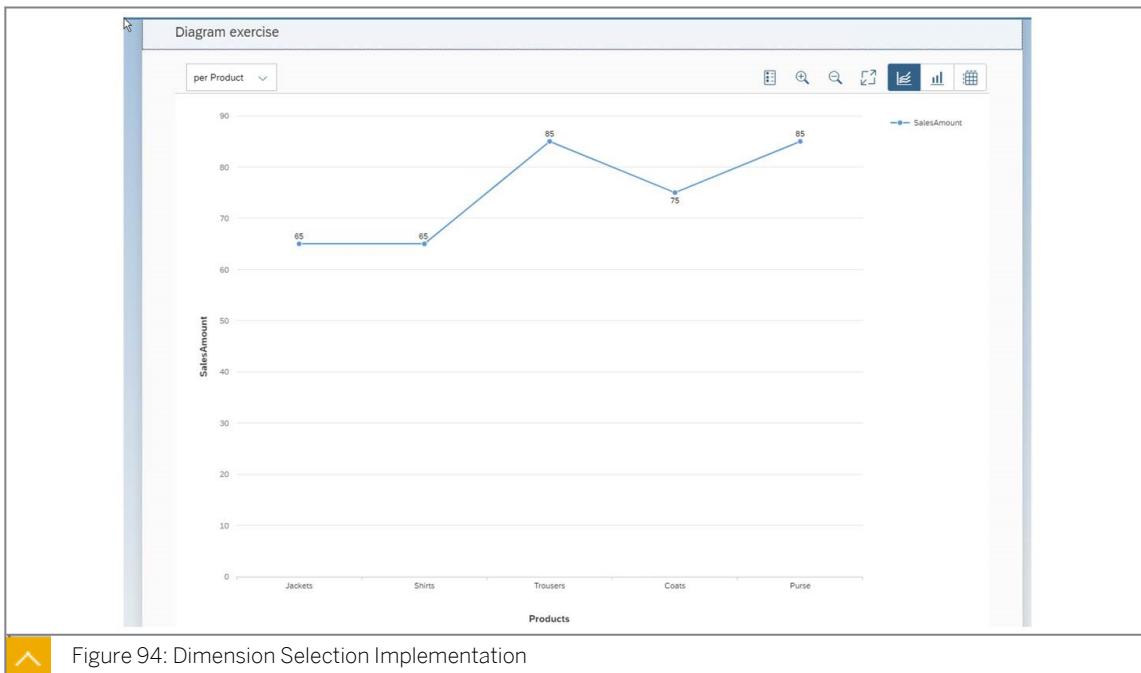


Figure 94: Dimension Selection Implementation

## Work with Diagrams



**Note:**

Please be aware that in some screenshots the code of previous implementation steps is not shown. Please do not delete any code that you have added in previous steps in the following steps.



**Note:**

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

**Objective:**

Create a SAPUI5 project to view data with graphs and tables.

**Business Scenario**

Implement a simple analytical application to visualize your data with graphs and tables. Use selection criteria to choose the data to display.

Follow the guidelines to create a full-screen application with a *Main* view.

**Task 1: Create a SAPUI5 project to view data with graphs and tables**

1. Create a SAPUI5 project with the following settings:

Parameter	Value
Project name	diagram
Namespace	student##.sap.training
Viewtype	XML
View name	Main
SAPUI5 Version	1.60

- a) Log on to the SAP Web IDE Full-Stack.
- b) Choose *File* → *New* → *Project from Template* → *SAPUI5 Application Project*, select the *SAPUI5-Version 1.60* and choose *Next*.
- c) Name your project *diagram* and in the *Namespace* field enter **student##.sap.training** and select *Next*.



Note:  
Use your assigned student and group number.

- d) In the View Name field enter **Main**.
- e) Choose *Finish* and the newly created project diagram appears in your workspace.

### Task 2: Implement the model

1. Create a folder with the name localService inside the webapp folder.
  - a) Select the webapp folder of your project and choose *New → Folder* from the context menu.
  - b) Name the folder localService in the dialog box and choose *OK*.
2. Add a new DataSource configuration by choosing the `metadata.xml` file located at `s:\Courses\UX410_20\templates\diagram\metadata.xml` as a service document and assigning a model with the name SalesModel to the newly created DataSource.
  - a) Open the `manifest.json` file in the *Descriptor Editor* view.
  - b) Choose the *Data Sources* tab and with the *plus symbol* select *File System* as a Source.
  - c) Browse and navigate to the `metadata.xml` file located at `s:\Courses\UX410_20\templates\diagram`.
  - d) Select the `metadata.xml` file with a double click and choose *Next* to confirm the data connection step in the data source dialog.
  - e) Choose the option *Create new model* in the model selection and insert **SalesModel** in the *Model Name* field and choose *Next*.
  - f) Choose *Finish* to create the new data source and model.
  - g) Save the changes.
3. Enter mock data.
  - a) Select your project and choose *Project → Project Settings* from the context menu.
  - b) Choose *Mock Data* and *Generated data* and save the selection.
  - c) Select the `metadata.xml` file, and from the context menu choose the option *Edit Mock Data*.
  - d) Select the *Use the data above as my mock data source* checkbox.
  - e) Select the *Generate Random Data* button.
  - f) Confirm the selection.  
The SAP Web IDE generates a file with the name **SalesFigures.json** and adds the newly created file to a folder named `mockdata`.
  - g) Open the file and analyse the content.

### Task 3: Implement a line graph

1. Display data with a line graph by replacing the generated Page control in the `Main.view.xml` with a control of type `sap.f.DynamicPage`. Add the following attributes to the newly created control.

Attribute	Value
<code>id</code>	<code>dynamicPageId</code>
<code>headerExpanded</code>	<code>true</code>
<code>toggleHeaderOnTitleClick</code>	<code>true</code>

- a) Open the file `Main.view.xml`.
- b) Add the **namespace alias** to the view control.

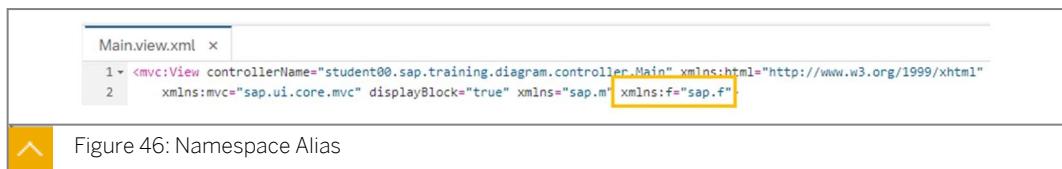


Figure 46: Namespace Alias

```

1+ <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:html="http://www.w3.org/1999/xhtml"
2+   xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m" xmlns:f="sap.f"

```

- c) Remove the generated Page control from the App control.
- d) Add the `DynamicPage` control with the attributes **XML-Namespace alias f** for `sap.f-UI5-namespace` to the app control.



Figure 47: Add Control and Attributes

```

1+ <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2+   xmlns:f="sap.f"
3+   <Shell id="shell">
4+     <App id="app">
5+       <pages>
6+         <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7+           <f:title>
8+             <f:DynamicPageTitle>
9+               <f:label>
10+              ...
11+            </f:label>
12+          </f:DynamicPageTitle>
13+        </f:title>
14+      </f:DynamicPage>
15+    </pages>
16+  </App>
17+ </Shell>
18+ </mvc:View>

```

2. Add a `sap.f.DynamicPageTitle` control to the title aggregation of the newly added `DynamicPage` control.

  - a) Add the title Aggregation to the `DynamicPage` control.
  - b) Add the UI control `sap.f.DynamicPageTitle` to the title aggregation.



Figure 48: Add UI Control

```

1+ <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2+   xmlns:f="sap.f"
3+   <Shell id="shell">
4+     <App id="app">
5+       <pages>
6+         <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7+           <f:title>
8+             <f:DynamicPageTitle>
9+               <f:label>
10+              ...
11+            </f:label>
12+          </f:DynamicPageTitle>
13+        </f:title>
14+      </f:DynamicPage>
15+    </pages>
16+  </App>
17+ </Shell>
18+ </mvc:View>

```

3. Add a `sap.m.Title` control to the heading aggregation of the newly added `DynamicPageTitle` control.

Attribute	Value
<code>text</code>	Sales figures

 Note:

Add the value as an attribute of your `i18n` property file. Use the value `pageTitle` as the key of your `i18n` file.

- a) Add the heading aggregation of the `DynamicPageTitle` control and add a `sap.m.Title` control with the attribute `text` to the newly created `heading` aggregation.



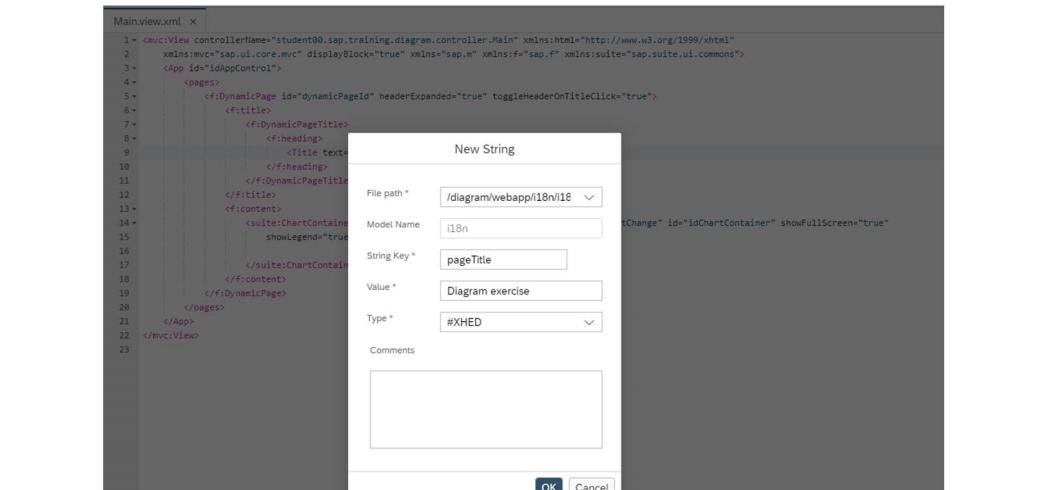
```

*Main.view.xml *
1 <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2   xmlns:f="sap.f"
3   <Shell id="shell">
4     <App id="app">
5       <pages>
6         <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7           <f:title>
8             <f:DynamicPageTitle>
9               <f:heading>
10                 <Title text="">
11               </f:heading>
12             </f:DynamicPageTitle>
13           </f:title>
14         </f:DynamicPage>
15       </pages>
16     </App>
17   </Shell>
18 </mvc:View>

```

 Figure 49: Page Title

- b) Use the following attribute values for the `sap.m.Title` control, type `text` by putting the cursor between the "" of the `text` attribute and choose *Create i18n String from the context menu*.
- c) Add `pageTitle` as the `String` key field and **Diagram Exercise** as the value.
- d) Choose `XHED` as type from the selection box `Type`.



New String

File path: /diagram/webapp/i18n/i18n

Model Name: i18n

String Key: pageTitle

Value: Diagram exercise

Type: #XHED

Comments:

OK Cancel

 Figure 50: Enter Values

- e) Press `OK` to display your implementation.



```

Main.view.xml x
1 <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2   xmlns:f="sap.f"
3   <Shell id="shell">
4     <App id="app">
5       <pages>
6         <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7           <f:title>
8             <f:DynamicPageTitle>
9               <f:heading>
10                 <Title text="{i18n>pageTitle}"/>
11               </f:heading>
12             </f:DynamicPageTitle>
13           </f:title>
14         </f:DynamicPage>
15       </pages>
16     </App>
17   </Shell>
18 </mvc:View>

```

Figure 51: Implementation

4. Add a `sap.suite.ui.commons.ChartContainer` to the content aggregation of the `DynamicPage` control. Use `suite` as a XML-namespace alias for the `sap.suite.ui.commons`-SAPUI5-namespace. Add the table attributes to the `ChartContainer-content` control.

Field	Value
id	idChartContainer
autoAdjustHeight	true
contentChange	onContentChange
showFullScreen	true
showLegend	true
showPersonalization	false
title	Sales figures



Note:

Add the title of the `ChartContainerContent` as an `i18n` property and use the id `diagramTitle`.

- Add the content aggregation to the `DynamicPage` control.
- Add the XML-namespace alias `suite` to view control and assign the UI5-namespace `sap.suite.ui.commons` to the alias `suite`.
- Create a new UI element of the type `ChartContainer` with the table attributes.



```

Main.view.xml x
1 <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2   xmlns:f="sap.f"
3   <Shell id="shell">
4     <App id="app">
5       <pages>
6         <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7           <f:title>
8             <f:DynamicPageTitle>
9               <f:heading>
10                 <Title text="{i18n>pageTitle}"/>
11               </f:heading>
12             </f:DynamicPageTitle>
13           </f:title>
14           <f:content>
15             <suite:ChartContainer autoAdjustHeight="true" contentChange="onContentChange" id="idChartContainer" showFullScreen="true" showLegend="true"
16               showPersonalization="false" title=""/>
17           </f:content>
18         </f:DynamicPage>
19       </pages>
20     </App>
21   </Shell>
22 </mvc:View>

```

Figure 52: Add UI Element

- d) Add the title of the *ChartContainer* control to the i18n properties file of your project by placing the cursor on the *tile* property and choose *Create i18n String* from the context menu.
- e) Insert the table data and choose *OK* to display your implementation.



Figure 53: Implementation

```

*Main.view.xml *
1 <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2   xmlns:f="sap.f"
3   <Shell id="shell">
4     <App id="app">
5       <pages>
6         <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7           <f:title>
8             <f:DynamicPageTitle>
9               <f:heading>
10                 <title text="{i18n>pageTitle}"/>
11               </f:heading>
12             </f:DynamicPageTitle>
13           </f:title>
14           <f:content>
15             <suite:ChartContainer autoAdjustHeight="true" contentChange="onContentChange" id="idChartContainer" showFullScreen="true" showLegend="true"
16               showPersonalization="false" title="{i18n>diagramTitle}"/>
17             </suite:ChartContainer>
18           </f:content>
19         </f:DynamicPage>
20       </pages>
21     </App>
22   </Shell>
23 </mvc:View>

```

- 5. Add a *sap(suite).ui.commons.ChartContainerContent* control to the content aggregation of the *ChartContainer* control. Use the attributes listed in the table. Add the content aggregation to the newly added *sap(suite).ui.commons.ChartContainerContent* control.

Attribute	Value
icon	sap-icon://line-chart
title	Line Chart



Note:

Assign the title **ChartContainerContent** to the i18n property file and use the id `lineChartTitle`.

- a) Add the content aggregation to the *ChartContainer* control.
- b) Add a *ChartContainerContent* control using the table attributes.



Figure 54: Use the Table Attributes

```

*Main.view.xml *
1 <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2   xmlns:f="sap.f"
3   <Shell id="shell">
4     <App id="app">
5       <pages>
6         <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7           <f:title>
8             <f:DynamicPageTitle>
9               <f:heading>
10                 <title text="{i18n>pageTitle}"/>
11               </f:heading>
12             </f:DynamicPageTitle>
13           </f:title>
14           <f:content>
15             <suite:ChartContainer autoAdjustHeight="true" contentChange="onContentChange" id="idChartContainer" showFullScreen="true" showLegend="true"
16               showPersonalization="false" title="{i18n>diagramTitle}"/>
17               <suite:content>
18                 <suite:ChartContainerContent icon="sap-icon://line-chart" title="{i18n>lineChartTitle}"/>
19               </suite:content>
20             </suite:ChartContainerContent>
21           </f:content>
22         </f:DynamicPage>
23       </pages>
24     </App>
25   </Shell>
26 </mvc:View>

```

- c) Add the content aggregation to the newly added *ChartContainerContent* control.
- 6. Add a *sap.ui.viz.Popover* control to the content aggregation of the *ChartContainerContent* control with the id `idLineChartPopover`.

- Add a XML-namespace `viz` pointing to `sap.ui.viz` to your view implementation.
- Add the `sap.ui.viz.Popover` control to the content aggregation of the `ChartContainerContent` control.

 line within the `ChartContainerContent` aggregation of the `ChartContainer` control." data-bbox="230 139 896 334"/>

```

1 <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:suite="sap.suite.ui.commons" xmlns:viz="sap.viz.ui5.controls">
3     <Shell id="shell">
4       <App id="app">
5         <pages>
6           <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7             <f:title></f:title>
8             <f:content>
9               <suite:ChartContainer autoAdjustHeight="true" contentChange="onContentChange" id="idChartContainer" showFullScreen="true" showLegend="true">
10                 <showPersonalization="false" title="{i18n>diagramTitle}">
11                 <suite:content>
12                   <suite:ChartContainerContent icon="sap-icon://line-chart" title="{i18n>lineChartTitle}">
13                     <viz:Popover id="idLineChartPopover"></viz:Popover>
14                   </suite:ChartContainerContent>
15                 </suite:content>
16               </suite:ChartContainer>
17             </f:content>
18           </f:DynamicPage>
19         </pages>
20       </App>
21     </Shell>
22   </mvc:View>

```

Figure 55: Add Control

- Add a `sap.ui.viz.VizFrame` control after the newly added `Popover` control. Use the attributes in the following table.

Attribute	Value
id	sap-icon://line-chart
height	9.1rem
width	100%
uiConfig	{applicationSet:'fiori'}

- Add a UI control of type `sap.ui.viz.VizFrame` to the content aggregation of the `ChartContainerContent` control.



```

1 <mvc:View controllerName="student00.sap.training.diagram.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:suite="sap.suite.ui.commons" xmlns:viz="sap.viz.ui5.controls">
3     <Shell id="shell">
4       <App id="app">
5         <pages>
6           <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
7             <f:title></f:title>
8             <f:content>
9               <suite:ChartContainer autoAdjustHeight="true" contentChange="onContentChange" id="idChartContainer" showFullScreen="true" showLegend="true">
10                 <showPersonalization="false" title="{i18n>diagramTitle}">
11                 <suite:content>
12                   <suite:ChartContainerContent icon="sap-icon://line-chart" title="{i18n>lineChartTitle}">
13                     <viz:Popover id="idLineChartPopover"></viz:Popover>
14                     <viz:VizFrame id="idLineChartVizFrame" height="9.1rem" width="100%" uiConfig="{applicationSet:'fiori'}"></viz:VizFrame>
15                   </suite:ChartContainerContent>
16                 </suite:content>
17               </suite:ChartContainer>
18             </f:content>
19           </f:DynamicPage>
20         </pages>
21       </App>
22     </Shell>
23   </mvc:View>

```

Figure 56: Add UI Control

- Save your changes.

- Implement the `onInit` function in the controller of `Main` view. Declare a member variable with the name `sCurrentVizFrame` and assign the value `idLineChartVizFrame` as a string to the variable. Declare a member variable `sCurrentSelectedDimension` and assign the value `0` as type string.

- Open the file `Main.controller` located in the controller folder of your project.

- b) Add an implementation of the function `onInit` to the `Main.controller.js` implementation.

```
Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.diagram.controller.Main", {
7     |   onInit: function () {
8
9     |   }
10   });
11});
```

Figure 57: Add the Function

- c) Declare a member variable `sCurrentVizFrame` and assign the value `idLineChartVizFrame` as a string.

```
Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.diagram.controller.Main", {
7     |   onInit: function () {
8     |     |   this.sCurrentVizFrame  = "idLineChartVizFrame";
9     |   }
10   });
11});
```

Figure 58: Add String

- d) Declare a member variable `sCurrentSelectedDimension` and assign the value `0` as a string.

```
Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.diagram.controller.Main", {
7     |   onInit: function () {
8     |     |   this.sCurrentVizFrame  = "idLineChartVizFrame";
9     |     |   this.sCurrentSelectedDimension = "0";
10    }
11  });
12});
```

Figure 59: Declare Variable

- e) Save your changes.

9. Add a reference of `sap.viz.ui5.controls.common.feeds.FeedItem` class to the controller implementation.

- a) Add a reference to the `sap.viz.ui5.controls.common.feeds.FeedItem` class.



```
Main.controller.js x
1 sap.ui.define([
2     "sap/ui/core/mvc/Controller",
3     "sap/viz/ui5/controls/common/feeds/FeedItem"
4 ], function (Controller,FeedItem) {
5     "use strict";

```

Figure 60: Add Reference

10. Implement a function called `_createFeedMap`.



```
_createFeedMap: function () {
    this.feedMap = {};
    this.feedMap.salesAmount = new FeedItem({
        "uid": "valueAxis",
        "type": "Measure",
        "values": ["SalesAmount"]
    });

    this.feedMap.products = new FeedItem({
        "uid": "categoryAxis",
        "type": "Dimension",
        "values": ["Products"]
    });

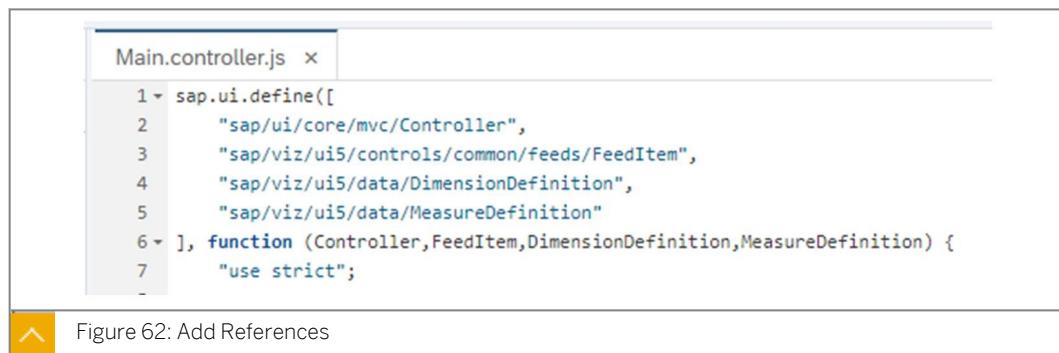
    this.feedMap.subregion = new FeedItem({
        "uid": "categoryAxis",
        "type": "Dimension",
        "values": ["Sub_Region_Name"]
    });

    this.feedMap.productsSubregion = new FeedItem({
        "uid": "categoryAxis",
        "type": "Dimension",
        "values": ["Products", "Sub_Region_Name"]
    });
}
```

Figure 61: Add Reference

11. Add a reference to the `sap.viz.ui5.data.MeasureDefinition` and to the class `sap.viz.ui5.data.DimensionDefinition` to your controller implementation and implement the below function `_createDataSetMap` in your `Main` controller.

- a) Add the reference to `sap.viz.ui5.data.DimensionDefinition` and to `sap.viz.ui5.data.MeasureDefinition` of your controller implementation.



```
Main.controller.js x
1 sap.ui.define([
2     "sap/ui/core/mvc/Controller",
3     "sap/viz/ui5/controls/common/feeds/FeedItem",
4     "sap/viz/ui5/data/DimensionDefinition",
5     "sap/viz/ui5/data/MeasureDefinition"
6 ], function (Controller,FeedItem,DimensionDefinition,MeasureDefinition) {
7     "use strict";

```

Figure 62: Add References

- b) Add the implementation to your `Main` controller implementation.



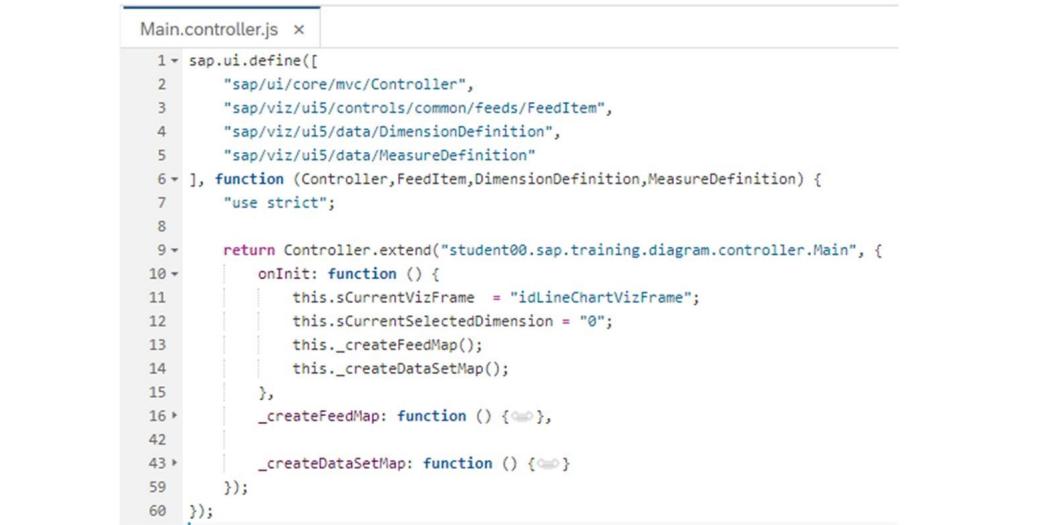
```

Main.controller.js x
1 ~ sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/viz/ui5/controls/common/feeds/FeedItem",
4   "sap/viz/ui5/data/DimensionDefinition",
5   "sap/viz/ui5/data/MeasureDefinition"
6 ~ ], function (Controller,FeedItem,DimensionDefinition,MeasureDefinition) {
7   "use strict";
8
9 ~   return Controller.extend("student00.sap.training.diagram.controller.Main", {
10 ~     /**
11 ~      * @function
12 ~      * @name _createFeedMap
13 ~      */
14 ~     _createFeedMap: function () {
15 ~       /**
16 ~        * @function
17 ~        * @name _createDataSetMap
18 ~        */
19 ~       _createDataSetMap: function () {
20 ~         this.dataSetMap = {};
21 ~         this.dataSetMap.productDim = new DimensionDefinition({
22 ~           name: "Products",
23 ~           value: "(SalesModel)>PRODUCT_NAME"
24 ~         });
25 ~         this.dataSetMap.subregionDim = new DimensionDefinition({
26 ~           name: "Sub_Region_Name",
27 ~           value: "(SalesModel)>SUB_REGION_NAME"
28 ~         });
29 ~         this.dataSetMap.salesAmountMeasure = new MeasureDefinition({
30 ~           name: "SalesAmount",
31 ~           value: "(SalesModel)>SALES_AMOUNT"
32 ~         });
33 ~       }
34 ~     }
35 ~   });
36 ~ });
37 ~ });
38 ~ });
39 ~ });
40 ~ });
41 ~ });
42 ~ });
43 ~ });
44 ~ });
45 ~ });
46 ~ });
47 ~ });
48 ~ });
49 ~ });
50 ~ });
51 ~ });
52 ~ });
53 ~ });
54 ~ });
55 ~ });
56 ~ });
57 ~ });
58 ~ });

```

Figure 63: Implementation of \_createDataSetMap

12. Invoke two new functions `_createDataSetMap` and `_createFeedMap` from inside the `onInit` function of your controller.



```

Main.controller.js x
1 ~ sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/viz/ui5/controls/common/feeds/FeedItem",
4   "sap/viz/ui5/data/DimensionDefinition",
5   "sap/viz/ui5/data/MeasureDefinition"
6 ~ ], function (Controller,FeedItem,DimensionDefinition,MeasureDefinition) {
7   "use strict";
8
9 ~   return Controller.extend("student00.sap.training.diagram.controller.Main", {
10 ~     /**
11 ~      * @function
12 ~      * @name onInit
13 ~      */
14 ~     onInit: function () {
15 ~       this.sCurrentVizFrame = "idLineChartVizFrame";
16 ~       this.sCurrentSelectedDimension = "0";
17 ~       this._createFeedMap();
18 ~       this._createDataSetMap();
19 ~     },
20 ~     /**
21 ~      * @function
22 ~      * @name _createFeedMap
23 ~      */
24 ~     _createFeedMap: function () {
25 ~       /**
26 ~        * @function
27 ~        * @name _createDataSetMap
28 ~        */
29 ~       this._createDataSetMap();
30 ~     }
31 ~   });
32 ~ });
33 ~ });
34 ~ });
35 ~ });
36 ~ });
37 ~ });
38 ~ });
39 ~ });
40 ~ });
41 ~ });
42 ~ });
43 ~ });
44 ~ });
45 ~ });
46 ~ });
47 ~ });
48 ~ });
49 ~ });
50 ~ });
51 ~ });
52 ~ });
53 ~ });
54 ~ });
55 ~ });
56 ~ });
57 ~ });
58 ~ });

```

Figure 64: Invoke Functions

13. Implement a function `_createDataSet` inside the `Main.controller.js` file. Declare a variable `oDataSet` and assign an object of type `sap.viz.ui5.data.FlattenedDataset`. Bind the Dataset to the `EntitySet` `SalesFigures` of your `SalesModel`. Invoke the newly created function in the `onInit` function of your controller. Assign the newly created dataset to a member variable with the name `dataSet`.

- a) Open the file `Main.controller.js` and add a reference to the `sap.viz.ui5.data.FlattenedDataset`.



```
Main.controller.js
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/viz/ui5/controls/common/feeds/FeedItem",
    "sap/viz/ui5/data/DimensionDefinition",
    "sap/viz/ui5/data/MeasureDefinition",
    "sap/viz/ui5/data/FlattenedDataset"
], function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset) {
    "use strict";
})
```

Figure 65: Add a Reference

**b) Implement the `_createDataSet` function.**



```
Main.controller.js
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/viz/ui5/controls/common/feeds/FeedItem",
    "sap/viz/ui5/data/DimensionDefinition",
    "sap/viz/ui5/data/MeasureDefinition",
    "sap/viz/ui5/data/FlattenedDataset"
], function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset) {
    "use strict";

    return Controller.extend("student00.sap.training.diagram.controller.Main", {
        ...
        _createDataSet: function () {
            var oDataset = new FlattenedDataset({
                data: {
                    path: "SalesModel>/SalesFigures"
                }
            });
            return oDataset;
        }
    });
})
```

Figure 66: Implement Function

**c) Invoke the function in the `onInit` function of your controller. Assign the return value to a member variable `_dataSet`.**



```
Main.controller.js
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/viz/ui5/controls/common/feeds/FeedItem",
    "sap/viz/ui5/data/DimensionDefinition",
    "sap/viz/ui5/data/MeasureDefinition",
    "sap/viz/ui5/data/FlattenedDataset"
], function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset) {
    "use strict";

    return Controller.extend("student00.sap.training.diagram.controller.Main", {
        ...
        onInit: function () {
            this.sCurrentVizFrame = "idLineChartVizFrame";
            this.sCurrentSelectedDimension = "0";
            this._createFeedMap();
            this._createDataSetMap();
            this._dataSet = this._createDataSet();
        }
    });
})
```

Figure 67: Assign Return Value

- 14. Create a function `_handleSelection` to handle the assignment of dimension and measure to the current selected diagram. The function should take a parameter with the name `selectedItem`. The implementation of the function should do the following: store the reference of the currently visible `VizFrame` control using the stored value from variable `sCurrentVizFrame` and store the reference to the control as a local variable.**

- a) Remove all the dimensions, measures, and feeds from the dataset object of the currently visible `VizFrame` object.
- b) Implement a switch-statement and check if the value of `selectedItem` is `0` as a string. When `selectedItem` is equal to "0" add the `productDim` and the `salesAmountMeasure` of the `dataSetMap` to the dataset of the current `VizFrame`.
- c) Add the corresponding feeds to the `VizFrame`.
- d) Implement the function `_handleSelection`.



```

*Main.controller.js
6
7  ], function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset) {
8  "use strict";
9
10 return Controller.extend("student00.sap.training.diagram.controller.Main", {
11   onInit: function () {
12     _createFeedMap: function () {
13     _createDataSetMap: function () {
14     _createDataSet: function () {
15     _handleSelection: function (selectedItem) {
16       var oVizFrame = this.getView().byId(
17         this.sCurrentVizFrame);
18       var oDataSet = oVizFrame.getDataset();
19       oDataSet.removeAllDimensions();
20       oDataSet.removeAllMeasures();
21       oVizFrame.removeAllFeeds();
22       switch (selectedItem) {
23         case "0":
24           {
25             oDataSet.addDimension(this.dataSetMap.productDim);
26             oDataSet.addMeasure(this.dataSetMap.salesAmountMeasure);
27             oVizFrame.addFeed(this.feedMap.products);
28             oVizFrame.addFeed(this.feedMap.salesAmount);
29           }
30           break;
31         }
32       }
33     });
34   });
35 });
36 });
37 });
38 });
39 });

```

Figure 68: Implement Function

### 15. Create a function with name `_createLineDiagram`.

- a) The function handles the creation of the line diagram. Read the reference of the `VizFrame` element with the ID `idLineChartVizFrame` and assign it to a variable. Provide a reference on the popover element with the ID `idLineChartPopover`.
- b) Call the function `_createDataSet` and assign the result to the reference of the `VizFrame` object which was read in step a.
- c) Call the function `_handleSelection` and pass the value of the member variable `sCurrentSelectedDimension` created in the `onInit` function.
- d) Call the function `setVizProperties`. Pass an object to define that data shown in the diagram should be labelled. Define the title not to display in the diagram.
- e) Pass a string literal with the value `line` to define the use of a line diagram by means of the function `setVizType` of the `VizFrame` object.
- f) Use the function `getVizUid` to call the function `connect` on the popover object of step a and pass the unique ID of the `VizFrame` object.
- g) Check that the implementation is as expected.



```

Main.controller.js x
1* sap.ui.define([
2  "sap/ui/core/mvc/Controller",
3  "sap/viz/ui5/controls/common/feeds/FeedItem",
4  "sap/viz/ui5/data/DimensionDefinition",
5  "sap/viz/ui5/data/MeasureDefinition",
6  "sap/viz/ui5/data/FlattenedDataset"
7* ], function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset) {
8  "use strict";
9
10* return Controller.extend("student00.sap.training.diagram.controller.Main", {
11*   _onInit: function () { },
12*   _createFeedMap: function () { },
13*   _createDataSetMap: function () { },
14*   _createDataSet: function () { },
15*   _handleSelection: function (selectedItem) { },
16*   _createLineDiagram: function () {
17*     var oVizFrame = this.getView().byId("idLineChartVizFrame");
18*     var oPop = this.getView().byId("idLineChartPopover");
19*     oVizFrame.setDataset(this._createDataSet());
20*     this._handleSelection(this.sCurrentSelectedDimension);
21*     oVizFrame.setVizProperties({ });
22*     oVizFrame.setVizType('line');
23*     oPop.connect(oVizFrame.getVizUid());
24*   },
25* });
26* });
27* });

```

Figure 69: Implementation

- h) Insert the `_createLineDiagram` function invocation at the end of your `onInit` function. Your `onInit` function should have the implementation required.



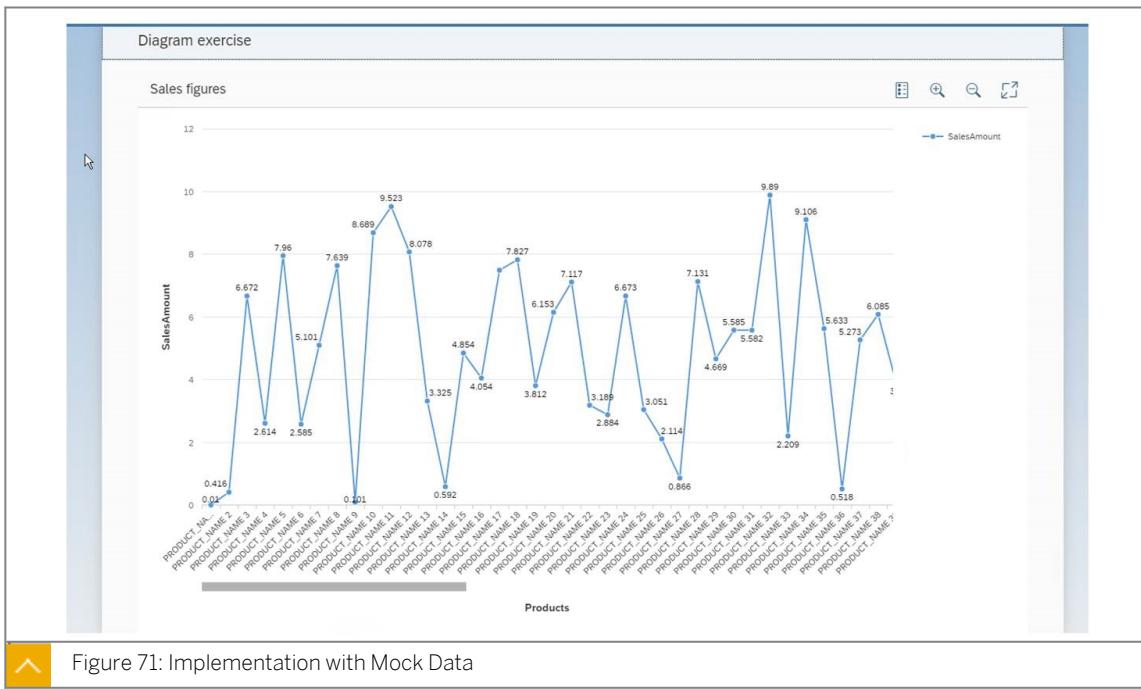
```

*Main.controller.js x
1* sap.ui.define([
2  "sap/ui/core/mvc/Controller",
3  "sap/viz/ui5/controls/common/feeds/FeedItem",
4  "sap/viz/ui5/data/DimensionDefinition",
5  "sap/viz/ui5/data/MeasureDefinition",
6  "sap/viz/ui5/data/FlattenedDataset"
7* ], function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset) {
8  "use strict";
9
10* return Controller.extend("student00.sap.training.diagram.controller.Main", {
11*   _onInit: function () {
12*     this.sCurrentVizFrame = "idLineChartVizFrame";
13*     this.sCurrentSelectedDimension = "0";
14*     this._createFeedMap();
15*     this._createDataSetMap();
16*     this._dataSet = this._createDataSet();
17*     this._createLineDiagram();
18*   },
19*   _createFeedMap: function () { },
20*   _createDataSetMap: function () { },
21*   _createDataSet: function () { },
22*   _handleSelection: function (selectedItem) { },
23*   _createLineDiagram: function () { },
24* });
25* });
26* });

```

Figure 70: Implementation

16. Test the application once the line graph is implemented. Execute using the MockServer option.
- Select `index.html` file and choose from the context menu `Run → Run Configurations`.
  - Press the + Button to add a new run configuration.
  - Choose `Run as Web Application`.
  - Check `Run with mock data` and select `Save and Run`.
  - Select the `index.html` file and from the context menu choose `Run → Run with Mock Data`.



#### Task 4: Implement a bar chart

1. Implement a bar chart. The surrounding `ChartContainerContent` control requires an attribute with the title `Column Chart`. Use file `i18n`.

Attribute	Value
<code>id</code>	<code>idBarChartVizFrame</code>
<code>height</code>	<code>9.1rem</code>
<code>width</code>	<code>100%</code>
<code>uiConfig</code>	<code>{applicationSet:'fiori'}</code>

- a) Create a new `ChartContainerContent` in the `Main.view.xml` file.
- b) The attribute `icon` of the new element, refers to the icon with the identifier `vertical-bar-chart`.
- c) Create an element with the type `VizFrame` in the content aggregation of the `ChartContainerContent` element.
- d) Assign the ID `idBarChartVizFrame` to the element, and assign an object with the variable `applicationSet` to the attribute `uiConfig`. The variable gets the value `fiori`.
- e) Review your implementation.

Figure 72: Review Implementation

## 2. Implement the function `_createColumnChart`.

- Open the view controller `Main.controller.js`.
- Implement the function `_createColumnChart`.
- Create a reference to the `VizFrame` element already created.
- Assign a configuration for `VizProperties` to the `VizFrame` object. To do this specify that the chart title should not be displayed, and define the beams to display with attribute data labels. Use the `setVizType` function to specify a bar chart. You can use a string literal with value **column**.

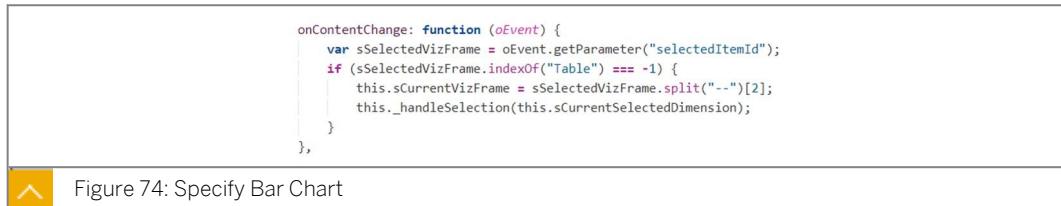
Figure 73: Assign a Configuration

## 3. Develop the event handler `contentChange` to switch between diagram views with the function `onContentChange`.

- Create an event handler function with the name `onContentChange`.
- Read out the parameter `selectedItem`.

- c) Check that the parameter `selectedItemId` contains the string `Table`. If not, split the parameter and assign it to the second value of the resulting array of member variables `sCurrentVizFrame`.

- d) Call the function `_handleSelection` and hand it over to the `sCurrentSelectedDimension` variable.



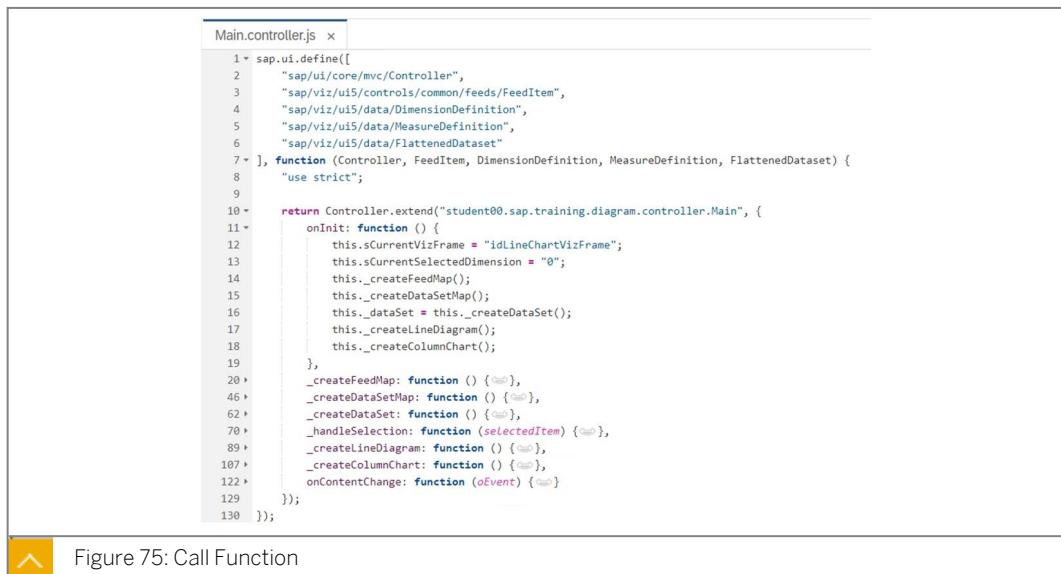
```

onContentChange: function (oEvent) {
    var sSelectedVizFrame = oEvent.getParameter("selectedItemId");
    if (sSelectedVizFrame.indexOf("Table") === -1) {
        this.sCurrentVizFrame = sSelectedVizFrame.split("-")[2];
        this._handleSelection(this.sCurrentSelectedDimension);
    }
},

```

Figure 74: Specify Bar Chart

- e) Add the call up `_createColumnChart` function at the end of your `onInit` implementation.



```

Main.controller.js x
1 * sap.ui.define([
2 *   "sap/ui/core/mvc/Controller",
3 *   "sap/viz/ui5/controls/common/feeds/FeedItem",
4 *   "sap/viz/ui5/data/DimensionDefinition",
5 *   "sap/viz/ui5/data/MeasureDefinition",
6 *   "sap/viz/ui5/data/FlattenedDataset"
7 * ],
8 *   function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset) {
9 *     "use strict";
10 *
11 *     return Controller.extend("student00.sap.training.diagram.controller.Main", {
12 *       onInit: function () {
13 *         this.sCurrentVizFrame = "idLineChartVizFrame";
14 *         this.sCurrentSelectedDimension = "0";
15 *         this._createFeedMap();
16 *         this._createDataSetMap();
17 *         this._dataSet = this._createDataSet();
18 *         this._createLineDiagram();
19 *         this._createColumnChart();
20 *       },
21 *       _createFeedMap: function () { },
22 *       _createDataSetMap: function () { },
23 *       _createDataSet: function () { },
24 *       _handleSelection: function (selectedItem) { },
25 *       _createLineDiagram: function () { },
26 *       _createColumnChart: function () { },
27 *       onContentChange: function (oEvent) { }
28 *     });
29 *   });
30 *

```

Figure 75: Call Function

- f) Save your changes.

- g) Test the applications. Switch between the line graph and bar chart.

### Task 5: Implement a table view

1. Create a `ChartContainerContent` with the `table-chart` icon and Sales figures as table title using the `i18n` file.
  - a) Add an `sap.m.Table` control to the content aggregation and add an `id` attribute to the `Table` control. Assign the value `idTable`.
  - b) Create another `ChartContainerContent` element, assign the value for the `table-chart` icon to the `icon` attribute, and add a `title` attribute.
  - c) Create a table of type `sap.m.Table` in the content aggregation of the `ChartContainerContent` and assign the ID `idTable` to the table control.



Figure 76: Implementation Result

```

32 <suite:ChartContainerContent icon="sap-icon://table-chart" title="{i18n>tableTitle}">
33   <suite:content>
34     <Table id="idTable">
35   </Table>
36 </suite:content>
37 </suite:ChartContainerContent>
38

```

- d) Add the headerToolbar-aggregation to the newly created table. Add a control of type OverflowToolbar and add the title control to the content-aggregation of the OverflowToolbar.



Figure 77: Table with headerToolbar-aggregation and OverflowToolbar control

```

<suite:ChartContainerContent icon="sap-icon://table-chart" title="{i18n>tableTitle}">
  <suite:content>
    <Table id="idTable">
      <headerToolbar>
        <OverflowToolbar>
          <content>
            <Title text="{i18n>tableheader}" level="H2"/>
          </content>
        </OverflowToolbar>
      </headerToolbar>
    </Table>
  </suite:content>
</suite:ChartContainerContent>

```

2. Create a function `_createTable` in the controller of the *Main* view with dependencies.
- a) Add the references as shown below.



Figure 78: Create Function

```

sap.ui.define([
  "sap/ui/core/mvc/Controller",
  "sap/viz/ui5/controls/common/Feeds/FeedItem",
  "sap/viz/ui5/data/DimensionDefinition",
  "sap/viz/ui5/data/MeasureDefinition",
  "sap/viz/ui5/data/FlattenedDataset",
  "sap/m/Column",
  "sap/m/Label",
  "sap/ui/core/Item",
  "sap/m/ColumnListItem",
  ], function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset, Column, Label, Item, ColumnListItem) {
  "use strict";
}

```

- b) Create a function `_createTable` in the view controller of the *Main* view.
- c) Create a reference to the first table in this function.
- d) Add 4 columns to the table to represent the region, area, product name, and sales figures.
- e) Create a template object of the type `sap.m.ColumnListItem` to display the values from the data model for the REGION\_NAME, SUB\_REGION\_NAME, PRODUCT\_NAME and SALES\_AMOUNT fields in the table.
- f) Bind the items in the table to the model element named sales by handing over the `sap.m.ColumnListItem` object already created to the used function `bindItem`.



Figure 79: Bind Items

```

Main.controller.js x
73     _handleSelection: function (selectedItem) {
92     _createLineDiagram: function () {
110     _createColumnChart: function () {
125     onContentChange: function (oEvent) {
132     _createTable: function () {
133         var oTable = this.getView().byId("idTable");
134         oTable.addColumn(new Column({header: new Label({text: "Region", textAlign: "Right"})}));
135         oTable.addColumn(new Column({header: new Label({text: "Sub Region", textAlign: "Left"})}));
136         oTable.addColumn(new Column({header: new Label({text: "Product name"})}));
137         oTable.addColumn(new Column({header: new Label({text: "Sales Amount"})}));
138
139         var oTableTemplate = new ColumnListItem({
140             type: sap.m.ListType.Active,
141             cells: [new Label({
142                 text: "(SalesModel>REGION_NAME)"
143             }), new Label({
144                 text: "(SalesModel>SUB_REGION_NAME)",
145                 textAlign: "Left"
146             }), new Label({
147                 text: "(SalesModel>PRODUCT_NAME)",
148                 textAlign: "Right"
149             }), new Label({
150                 text: "(SalesModel>SALES_AMOUNT)",
151                 textAlign: "Center"
152             })
153         });
154     });
155     oTable.bindItems("SalesModel>/SalesFigures",oTableTemplate, null, null);
156 },
157 });

```

g) Call the function `_createTable` in the `onInit` function.



Figure 80: Call Function to Create Table

```

Main.controller.js x
1 sap.ui.define([
2     "sap/ui/core/mvc/Controller",
3     "sap/viz/u5/controls/common/feeds/FeedItem",
4     "sap/viz/u5/data/DimensionDefinition",
5     "sap/viz/u5/data/MeasureDefinition",
6     "sap/viz/u5/data/FlattenedDataset",
7     "sap/m/Column",
8     "sap/m/Label",
9     "sap/ui/core/Item",
10    "sap/m/ColumnListItem",
11    ],
12    function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset, Column, Label, Item, ColumnListItem) {
13        "use strict";
14        return Controller.extend("student00.sap.training.diagram.controller.Main", {
15            onInit: function () {
16                this.currentVizFrame = "idLineChartVizFrame";
17                this.currentSelectedDimension = "0";
18                this._createFeedMap();
19                this._createDataSetMap();
20                this._createDataSet();
21                this._createLineDiagram();
22                this._createColumnChart();
23                this._createTable();
24            }
25        });
26    }

```

h) Test your application by switching between the line and bar chart and find another button in the toolbar table to show the newly implemented table.

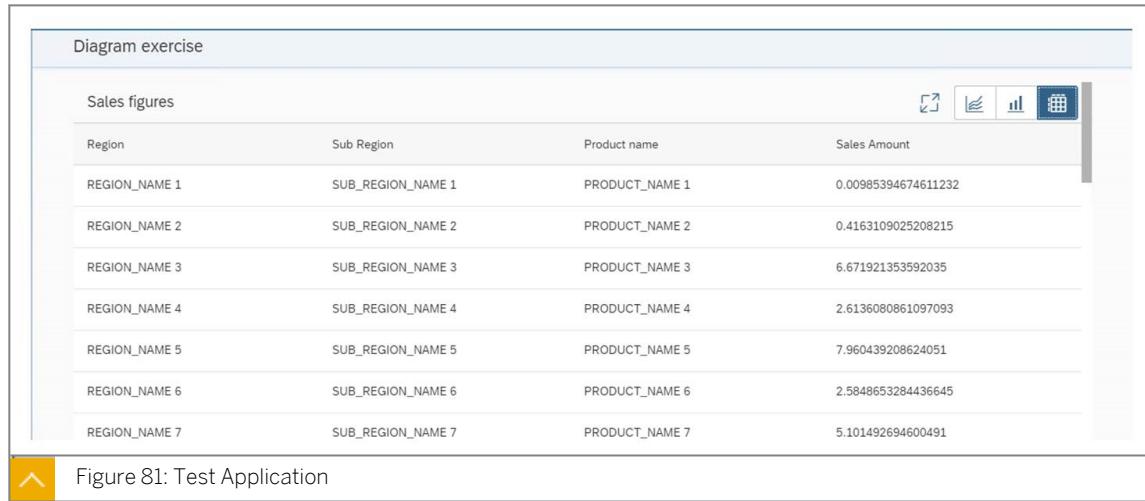


Diagram exercise

Sales figures

Region	Sub Region	Product name	Sales Amount
REGION_NAME 1	SUB_REGION_NAME 1	PRODUCT_NAME 1	0.00985394674611232
REGION_NAME 2	SUB_REGION_NAME 2	PRODUCT_NAME 2	0.4163109025208215
REGION_NAME 3	SUB_REGION_NAME 3	PRODUCT_NAME 3	6.671921353592035
REGION_NAME 4	SUB_REGION_NAME 4	PRODUCT_NAME 4	2.6136080861097093
REGION_NAME 5	SUB_REGION_NAME 5	PRODUCT_NAME 5	7.960439208624051
REGION_NAME 6	SUB_REGION_NAME 6	PRODUCT_NAME 6	2.5848653284436645
REGION_NAME 7	SUB_REGION_NAME 7	PRODUCT_NAME 7	5.101492694600491

Figure 81: Test Application

### Task 6: Implement a dimension selection

1. Implement a dimension selection. Use a control of type sap.m.Select with the following attribute and values:

Parameter	Value
id	idDimSelector
visible	true
change	onChange

- a) Open the `Main.view.xml` file and insert an element of type `dimensionSelector` into `ChartContainer`.
- b) Create an element of type `sap.m.Select`.
- c) Assign the ID `idDimSelector` and to the event handler `onChange` assign the `change` event.



Figure 82: Assign ID

2. Create a function `_createSelector`. Use the values in the table below.

The function fills the previously created selection list with values. Extend the implementation of the function `handleSelection` to handle the different visualization facettes.

Item-Key	Item-Text
0	per Product
1	per Region
2	per Product and Region

- a) In the view controller create a function `_createSelector`. Use the `byId` function to read the selection list, and assign three objects of type `sap.ui.core.Item`. Use the i18N capabilities of SAPUI5.
- b) These results occur due to the implementation.

```

*Main.controller.js x
4  "sap/viz/u13/data/DimensionInitialization",
5  "sap/viz/u15/data/MeasureDefinition",
6  "sap/viz/u15/data/FlattenedDataset",
7  "sap/m/Column",
8  "sap/m/Label",
9  "sap/ui/core/Item",
10 "sap/m/ColumnListItem"
11 }, function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset, Column, Label, Item, ColumnListItem) {
12 "use strict";
13 return Controller.extend("student00.sap.training.diagram.controller.Main", {
14     "onInit": function () {
15         "createFeedMap: function () {
16             "createDataSet: function () {
17                 "handleSelection: function (selectedItem) {
18                     "createLineDiagram: function () {
19                         "createColumnChart: function () {
20                             "onContentChange: function (oEvent) {
21                             "createTable: function () {
22                             "createSelector: function () {
23                                 "var oViewSelector = this.byId("idDimSelector");
24                                 "var oItemProduct = new Item({key: "0", text: "{i18n>perProduct}"});
25                                 "var oItemRegion = new Item({key: "1", text: "{i18n>perRegion}"});
26                                 "var oItemProductRegion = new Item({key: "2", text: "{i18n>perProdReg}"});
27
28                                 "oViewSelector.addItem(oItemProduct);
29                                 "oViewSelector.addItem(oItemRegion);
30                                 "oViewSelector.addItem(oItemProductRegion);
31
32                             },
33                         });
34                     });
35                 });
36             });
37         });
38     },
39 });
40
41 });
42
43 });
44
45 });
46
47 });
48
49 });
50
51 });
52
53 });
54
55 });
56
57 });
58
59 });
60
61 });
62
63 });
64
65 });
66
67 });
68
69 });
70
71 });
72
73 });
74
75 });
76
77 });
78
79 });
80
81 });
82
83 });
84
85 });
86
87 });
88
89 });
90
91 });
92
93 });
94
95 });
96
97 });
98
99 });
100
101 });
102
103 });
104
105 });
106
107 });
108
109 });
110
111 });
112
113 });
114
115 });
116
117 });
118
119 });
120
121 });
122
123 });
124
125 });
126
127 });
128
129 });
130
131 });
132
133 });
134
135 });
136
137 });
138
139 });
140
141 });
142
143 });
144
145 });
146
147 });
148
149 });
150
151 });
152
153 });
154
155 });
156
157 });
158
159 });
160
161 });
162
163 });
164
165 });
166
167 });
168
169 });
170
171 });
172
173 });
174
175 });
176
177 });
178
179 });
180
181 });
182
183 });
184
185 });
186
187 });
188
189 });
190
191 });
192
193 });
194
195 });
196
197 });
198
199 });
200
201 });
202
203 });
204
205 });
206
207 });
208
209 });
210
211 });
212
213 });
214
215 });
216
217 });
218
219 });
220
221 });
222
223 });
224
225 });
226
227 });
228
229 });
230
231 });
232
233 });
234
235 });
236
237 });
238
239 });
240
241 });
242
243 });
244
245 });
246
247 });
248
249 });
250
251 });
252
253 });
254
255 });
256
257 });
258
259 });
260
261 });
262
263 });
264
265 });
266
267 });
268
269 });
270
271 });
272
273 });
274
275 });
276
277 });
278
279 });
279
280 });
281
282 });
283
284 });
285
286 });
287
288 });
289
290 });
291
292 });
293
294 });
295
296 });
297
298 });
299
299 });
300
301 });
302
303 });
304
305 });
306
307 });
308
309 });
309
310 });
310
311 });
311
312 });
312
313 });
313
314 });
314
315 });
315
316 });
316
317 });
317
318 });
318
319 });
319
320 });
320
321 });
321
322 });
322
323 });
323
324 });
324
325 });
325
326 });
326
327 });
327
328 });
328
329 });
329
330 });
330
331 });
331
332 });
332
333 });
333
334 });
334
335 });
335
336 });
336
337 });
337
338 });
338
339 });
339
340 });
340
341 });
341
342 });
342
343 });
343
344 });
344
345 });
345
346 });
346
347 });
347
348 });
348
349 });
349
350 });
350
351 });
351
352 });
352
353 });
353
354 });
354
355 });
355
356 });
356
357 });
357
358 });
358
359 });
359
360 });
360
361 });
361
362 });
362
363 });
363
364 });
364
365 });
365
366 });
366
367 });
367
368 });
368
369 });
369
370 });
370
371 });
371
372 });
372
373 });
373
374 });
374
375 });
375
376 });
376
377 });
377
378 });
378
379 });
379
380 });
380
381 });
381
382 });
382
383 });
383
384 });
384
385 });
385
386 });
386
387 });
387
388 });
388
389 });
389
390 });
390
391 });
391
392 });
392
393 });
393
394 });
394
395 });
395
396 });
396
397 });
397
398 });
398
399 });
399
400 });
400
401 });
401
402 });
402
403 });
403
404 });
404
405 });
405
406 });
406
407 });
407
408 });
408
409 });
409
410 });
410
411 });
411
412 });
412
413 });
413
414 });
414
415 });
415
416 });
416
417 });
417
418 });
418
419 });
419
420 });
420
421 });
421
422 });
422
423 });
423
424 });
424
425 });
425
426 });
426
427 });
427
428 });
428
429 });
429
430 });
430
431 });
431
432 });
432
433 });
433
434 });
434
435 });
435
436 });
436
437 });
437
438 });
438
439 });
439
440 });
440
441 });
441
442 });
442
443 });
443
444 });
444
445 });
445
446 });
446
447 });
447
448 });
448
449 });
449
450 });
450
451 });
451
452 });
452
453 });
453
454 });
454
455 });
455
456 });
456
457 });
457
458 });
458
459 });
459
460 });
460
461 });
461
462 });
462
463 });
463
464 });
464
465 });
465
466 });
466
467 });
467
468 });
468
469 });
469
470 });
470
471 });
471
472 });
472
473 });
473
474 });
474
475 });
475
476 });
476
477 });
477
478 });
478
479 });
479
480 });
480
481 });
481
482 });
482
483 });
483
484 });
484
485 });
485
486 });
486
487 });
487
488 });
488
489 });
489
490 });
490
491 });
491
492 });
492
493 });
493
494 });
494
495 });
495
496 });
496
497 });
497
498 });
498
499 });
499
500 });
500
501 });
501
502 });
502
503 });
503
504 });
504
505 });
505
506 });
506
507 });
507
508 });
508
509 });
509
510 });
510
511 });
511
512 });
512
513 });
513
514 });
514
515 });
515
516 });
516
517 });
517
518 });
518
519 });
519
520 });
520
521 });
521
522 });
522
523 });
523
524 });
524
525 });
525
526 });
526
527 });
527
528 });
528
529 });
529
530 });
530
531 });
531
532 });
532
533 });
533
534 });
534
535 });
535
536 });
536
537 });
537
538 });
538
539 });
539
540 });
540
541 });
541
542 });
542
543 });
543
544 });
544
545 });
545
546 });
546
547 });
547
548 });
548
549 });
549
550 });
550
551 });
551
552 });
552
553 });
553
554 });
554
555 });
555
556 });
556
557 });
557
558 });
558
559 });
559
560 });
560
561 });
561
562 });
562
563 });
563
564 });
564
565 });
565
566 });
566
567 });
567
568 });
568
569 });
569
570 });
570
571 });
571
572 });
572
573 });
573
574 });
574
575 });
575
576 });
576
577 });
577
578 });
578
579 });
579
580 });
580
581 });
581
582 });
582
583 });
583
584 });
584
585 });
585
586 });
586
587 });
587
588 });
588
589 });
589
590 });
590
591 });
591
592 });
592
593 });
593
594 });
594
595 });
595
596 });
596
597 });
597
598 });
598
599 });
599
600 });
600
601 });
601
602 });
602
603 });
603
604 });
604
605 });
605
606 });
606
607 });
607
608 });
608
609 });
609
610 });
610
611 });
611
612 });
612
613 });
613
614 });
614
615 });
615
616 });
616
617 });
617
618 });
618
619 });
619
620 });
620
621 });
621
622 });
622
623 });
623
624 });
624
625 });
625
626 });
626
627 });
627
628 });
628
629 });
629
630 });
630
631 });
631
632 });
632
633 });
633
634 });
634
635 });
635
636 });
636
637 });
637
638 });
638
639 });
639
640 });
640
641 });
641
642 });
642
643 });
643
644 });
644
645 });
645
646 });
646
647 });
647
648 });
648
649 });
649
650 });
650
651 });
651
652 });
652
653 });
653
654 });
654
655 });
655
656 });
656
657 });
657
658 });
658
659 });
659
660 });
660
661 });
661
662 });
662
663 });
663
664 });
664
665 });
665
666 });
666
667 });
667
668 });
668
669 });
669
670 });
670
671 });
671
672 });
672
673 });
673
674 });
674
675 });
675
676 });
676
677 });
677
678 });
678
679 });
679
680 });
680
681 });
681
682 });
682
683 });
683
684 });
684
685 });
685
686 });
686
687 });
687
688 });
688
689 });
689
690 });
690
691 });
691
692 });
692
693 });
693
694 });
694
695 });
695
696 });
696
697 });
697
698 });
698
699 });
699
700 });
700
701 });
701
702 });
702
703 });
703
704 });
704
705 });
705
706 });
706
707 });
707
708 });
708
709 });
709
710 });
710
711 });
711
712 });
712
713 });
713
714 });
714
715 });
715
716 });
716
717 });
717
718 });
718
719 });
719
720 });
720
721 });
721
722 });
722
723 });
723
724 });
724
725 });
725
726 });
726
727 });
727
728 });
728
729 });
729
730 });
730
731 });
731
732 });
732
733 });
733
734 });
734
735 });
735
736 });
736
737 });
737
738 });
738
739 });
739
740 });
740
741 });
741
742 });
742
743 });
743
744 });
744
745 });
745
746 });
746
747 });
747
748 });
748
749 });
749
750 });
750
751 });
751
752 });
752
753 });
753
754 });
754
755 });
755
756 });
756
757 });
757
758 });
758
759 });
759
760 });
760
761 });
761
762 });
762
763 });
763
764 });
764
765 });
765
766 });
766
767 });
767
768 });
768
769 });
769
770 });
770
771 });
771
772 });
772
773 });
773
774 });
774
775 });
775
776 });
776
777 });
777
778 });
778
779 });
779
780 });
780
781 });
781
782 });
782
783 });
783
784 });
784
785 });
785
786 });
786
787 });
787
788 });
788
789 });
789
790 });
790
791 });
791
792 });
792
793 });
793
794 });
794
795 });
795
796 });
796
797 });
797
798 });
798
799 });
799
800 });
800
801 });
801
802 });
802
803 });
803
804 });
804
805 });
805
806 });
806
807 });
807
808 });
808
809 });
809
810 });
810
811 });
811
812 });
812
813 });
813
814 });
814
815 });
815
816 });
816
817 });
817
818 });
818
819 });
819
820 });
820
821 });
821
822 });
822
823 });
823
824 });
824
825 });
825
826 });
826
827 });
827
828 });
828
829 });
829
830 });
830
831 });
831
832 });
832
833 });
833
834 });
834
835 });
835
836 });
836
837 });
837
838 });
838
839 });
839
840 });
840
841 });
841
842 });
842
843 });
843
844 });
844
845 });
845
846 });
846
847 });
847
848 });
848
849 });
849
850 });
850
851 });
851
852 });
852
853 });
853
854 });
854
855 });
855
856 });
856
857 });
857
858 });
858
859 });
859
860 });
860
861 });
861
862 });
862
863 });
863
864 });
864
865 });
865
866 });
866
867 });
867
868 });
868
869 });
869
870 });
870
871 });
871
872 });
872
873 });
873
874 });
874
875 });
875
876 });
876
877 });
877
878 });
878
879 });
879
880 });
880
881 });
881
882 });
882
883 });
883
884 });
884
885 });
885
886 });
886
887 });
887
888 });
888
889 });
889
890 });
890
891 });
891
892 });
892
893 });
893
894 });
894
895 });
895
896 });
896
897 });
897
898 });
898
899 });
899
900 });
900
901 });
901
902 });
902
903 });
903
904 });
904
905 });
905
906 });
906
907 });
907
908 });
908
909 });
909
910 });
910
911 });
911
912 });
912
913 });
913
914 });
914
915 });
915
916 });
916
917 });
917
918 });
918
919 });
919
920 });
920
921 });
921
922 });
922
923 });
923
924 });
924
925 });
925
926 });
926
927 });
927
928 });
928
929 });
929
930 });
930
931 });
931
932 });
932
933 });
933
934 });
934
935 });
935
936 });
936
937 });
937
938 });
938
939 });
939
940 });
940
941 });
941
942 });
942
943 });
943
944 });
944
945 });
945
946 });
946
947 });
947
948 });
948
949 });
949
950 });
950
951 });
951
952 });
952
953 });
953
954 });
954
955 });
955
956 });
956
957 });
957
958 });
958
959 });
959
960 });
960
961 });
961
962 });
962
963 });
963
964 });
964
965 });
965
966 });
966
967 });
967
968 });
968
969 });
969
970 });
970
971 });
971
972 });
972
973 });
973
974 });
974
975 });
975
976 });
976
977 });
977
978 });
978
979 });
979
980 });
980
981 });
981
982 });
982
983 });
983
984 });
984
985 });
985
986 });
986
987 });
987
988 });
988
989 });
989
990 });
990
991 });
991
992 });
992
993 });
993
994 });
994
995 });
995
996 });
996
997 });
997
998 });
998
999 });
999
1000 });
1000
1001 });
1001
1002 });
1002
1003 });
1003
1004 });
1004
1005 });
1005
1006 });
1006
1007 });
1007
1008 });
1008
1009 });
1009
1010 });
1010
1011 });
1011
1012 });
1012
1013 });
1013
1014 });
1014
1015 });
1015
1016 });
1016
1017 });
1017
1018 });
1018
1019 });
1019
1020 });
1020
1021 });
1021
1022 });
1022
1023 });
1023
1024 });
1024
1025 });
1025
1026 });
1026
1027 });
1027
1028 });
1028
1029 });
1029
1030 });
1030
1031 });
1031
1032 });
1032
1033 });
1033
1034 });
1034
1035 });
1035
1036 });
1036
1037 });
1037
1038 });
1038
1039 });
1039
1040 });
1040
1041 });
1041
1042 });
1042
1043 });
1043
1044 });
1044
1045 });
1045
1046 });
1046
1047 });
1047
1048 });
1048
1049 });
1049
1050 });
1050
1051 });
1051
1052 });
1052
1053 });
1053
1054 });
1054
1055 });
1055
1056 });
1056
1057 });
1057
1058 });
1058
1059 });
1059
1060 });
1060
1061 });
1061
1062 });
1062
1063 });
1063
1064 });
1064
1065 });
1065
1066 });
1066
1067 });
1067
1068 });
1068
1069 });
1069
1070 });
1070
1071 });
1071
1072 });
1072
1073 });
1073
1074 });
1074
1075 });
1075
1076 });
1076
1077 });
1077
1078 });
1078
1079 });
1079
1080 });
1080
1081 });
1081
1082 });
1082
1083 });
1083
1084 });
1084
1085 });
1085
1086 });
1086
1087 });
1087
1088 });
1088
1089 });
1089
1090 });
1090
1091 });
1091
1092 });
1092
1093 });
1093
1094 });
1094
1095 });
1095
1096 });
1096
1097 });
1097
1098 });
1098
1099 });
1099
1100 });
1100
1101 });
1101
1102 });
1102
1103 });
1103
1104 });
1104
1105 });
1105
1106 });
1106
1107 });
1107
1108 });
1108
1109 });
1109
1110 });
1110
1111 });
1111
1112 });
1112
1113 });
1113
1114 });
1114
1115 });
1115
1116 });
1116
1117 });
1117
1118 });
1118
1119 });
1119
1120 });
1120
1121 });
1121
1122 });
1122
1123 });
1123
1124 });
1124
1125 });
1125
1126 });
1126
1127 });
1127
1128 });
1128
1129 });
1129
1130 });
1130
1131 });
1131
1132 });
1132
1133 });
1133
1134 });
1134
1135 });
1135
1136 });
1136
1137 });
1137
1138 });
1138
1139 });
1139
1140 });
1140
1141 });
1141
1142 });
1142
1143 });
1143
1144 });
1144
1145 });
1145
1146 });
1146
1147 });
1147
1148 });
1148
1149 });
1149
1150 });
1150
1151 });
1151
1152 });
1152
1153 });
1153
1154 });
1154
1155 });
1155
1156 });
1156
1157 });
1157
1158 });
1158
1159 });
1159
1160 });
1160
1161 });
1161
1162 });
1162
1163 });
1163
1164 });
1164
1165 });
1165
1166 });
1166
1167 });
1167
1168 });
1168
1169 });
1169
1170 });
1170
1171 });
1171
1172 });
1172
1173 });
1173
1174 });
1174
1175 });
1175
1176 });
1176
1177 });
1177
1178 });
1178
1179 });
1179
1180 });
1180
1181 });
1181
1182 });
1182
1183 });
1183
1184 });
1184
1185 });
1185
1186 });
1186
1187 });
1187
1188 });
1188
1189 });
1189
1190 });
1190
1191 });
1191
1192 });
1192
1193 });
1193
1194 });
1194
1195 });
1195
1196 });
1196
1197 });
1197
1198 });
1198
1199 });
1199
1200 });
1200
1201 });
1201
1202 });
1202
1203 });
1203
1204 });
1204
1205 });
1205
1206 });
1206
1207 });
1207
1208 });
1208
1209 });
1209
1210 });
1210
1211 });
1211
1212 });
1212
1213 });
1213
1214 });
1214
1215 });
1215
1216 });
1216
1217 });
1217
1218 });
1218
1219 });
1219
1220 });
1220
1221 });
1221
1222 });
1222
1223 });
1223
1224 });
1224
1225 });
1225
1226 });
1226
1227 });
1227
1228 });
1228
1229 });
1229
1230 });
1230
1231 });
1231
1232 });
1232
1233 });
1233
1234 });
1234
1235 });
1235
1236 });
1236
1237 });
1237
1238 });
1238
1239 });
1239
1240 });
1240
1241 });
1241
1242 });
1242
1243 });
1243
1244 });
1244
1245 });
1245
1246 });
1246
1247 });
1247
1248 });
1248
1249 });
1249
1250 });
1250
1251 });
1251
1252 });
1252
1253 });
1253
1254 });
1254
1255 });
1255
1256 });
1256
1257 });
1257
1258 });
1258
1259 });
1259
1260 });
1260
1261 });
1261
1262 });
1262
1263 });
1263
1264 });
1264
1265 });
1265
1266 });
1266
1267 });
1267
1268 });
1268
1269 });
1269
1270 });
1270
1271 });
1271
1272 });
1272
1273 });
1273
1274 });
1274
1275 });
1275
1276 });
1276
1277 });
1277
1278 });
1278
1279 });
1279
1280 });
1280
1281 });
1281
1282 });
1282
1283 });
1283
1284 });
1284
1285 });
1285
1286 });
1286
1287 });
1287
1288 });
1288
1289 });
1289
1290 });
1290
1291 });
1291
1292 });
1292
1293 });
1293
1294 });
1294
1295 });
1295
1296 });
1296
1297 });
1297
1298 });
1298
1299 });
1299
1300 });
1300
1301 });
1301
1302 });
1302
1303 });
1303
1304 });
1304
1305 });
1305
1306 });
1306
1307 });
1307
1308 });
1308
1309 });
1309
1310 });
1310
1311 });
1311
1312 });
1312
1313 });
1313
1314 });
1314
1315 });
1315
1316 });
1316
1317 });
1317
1318 });
1318
1319 });
1319
1320 });
1320
1321 });
1321
1322 });
1322
1323 });
1323
1324 });
1324
1325 });
1325
1326 });
1326
1327 });
1327
1328 });
1328
1329 });
1329
1330 });
1330
1331 });
1331
1332 });
1332
1333 });
1333
1334 });
1334
1335 });
1335
1336 });
1336
1337 });
1337
1338 });
1338
1339 });
1339
1340 });
1340
1341 });
1341
1342 });
1342
1343 });
1343
1344 });
1344
1345 });
1345
1346 });
1346
1347 });
1347
1348 });
1348
1349 });
1349
1350 });
1350
1351 });
1351
1352 });
1352
1353 });
1353
1354 });
1354
1355 });
1355
1356 });
1356
1357 });
1357
1358 });
1358
1359 });
1359
1360 });
1360
1361 });
1361
1362 });
1362
1363 });
1363
1364 });
1364
1365 });
1365
1366 });
1366
1367 });
1367
1368 });
1368
1369 });
1369
1370 });
1370
1371 });
1371
1372 });
1372
1373 });
1373
1374 });
1374
1375 });
1375
1376 });
1376
1377 });
1377
1378 });
1378
1379 });
1379
1380 });
1380
1381 });
1381
1382 });
1382
1383 });
1383
1384 });
1384
1385 });
1385
1386 });
1386
1387 });
1387
1388 });
1388
1389 });
1389
1390 });
1390
1391 });
1391
1392 });
1392
1393 });
1393
1394 });
1394
1395 });
1395
1396 });
1396
1397 });
1397
1398 });
1398
1399 });
1399
1400 });
1400
1401 });
1401
1402 });
1402
1403 });
1403
1404 });
1404
1405 });
1405
1406 });
1406
1407 });
1407
1408 });
1408
1409 });
1409
1410 });
1410
1411 });
1411
1412 });
1412
1413 });
1413
1414 });
1
```



```

*Main.controller.js x
1+ sap.ui.define([
2  "sap/ui/core/mvc/Controller",
3  "sap/viz/ui5/controls/common/feeds/FeedItem",
4  "sap/viz/ui5/data/DimensionDefinition",
5  "sap/viz/ui5/data/MeasureDefinition",
6  "sap/viz/ui5/data/FlattenedDataset",
7  "sap/m/Column",
8  "sap/m/Label",
9  "sap/ui/core/Item",
10 "sap/m/ColumnListItem"
11 ], function (Controller, FeedItem, DimensionDefinition, MeasureDefinition, FlattenedDataset, Column, Label, Item, ColumnListItem) {
12 "use strict";
13 return Controller.extend("student00.sap.training.diagram.controller.Main", {
14 	onInit: function () {
15  	this.sCurrentVizFrame = "#idLineChartVizFrame";
16  	this.sCurrentSelectedDimension = "0";
17  	this._createFeedMap();
18  	this._createDataSetMap();
19  	this._dataSet = this._createDataSet();
20  	this._createLineDiagram();
21  	this._createColumnChart();
22  	this._createTable();
23  	this._createSelector();
24  },
25  _createFeedMap: function () { },
26  _createDataSetMap: function () { },
27  _createDataSet: function () { },
28  _handleSelection: function (selectedItem) { },
29  _createLineDiagram: function () { }
30 });
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1390
1391
1392
1393
1394
1394
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1490
1491
1492
1493
1494
1494
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1590
1591
1592
1593
1594
1594
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1690
1691
1692
1693
1694
1694
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1790
1791
1792
1793
1794
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1890
1891
1892
1893
1894
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1990
1991
1992
1993
1994
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2089
2090
2091
2092
2093
2094
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
```

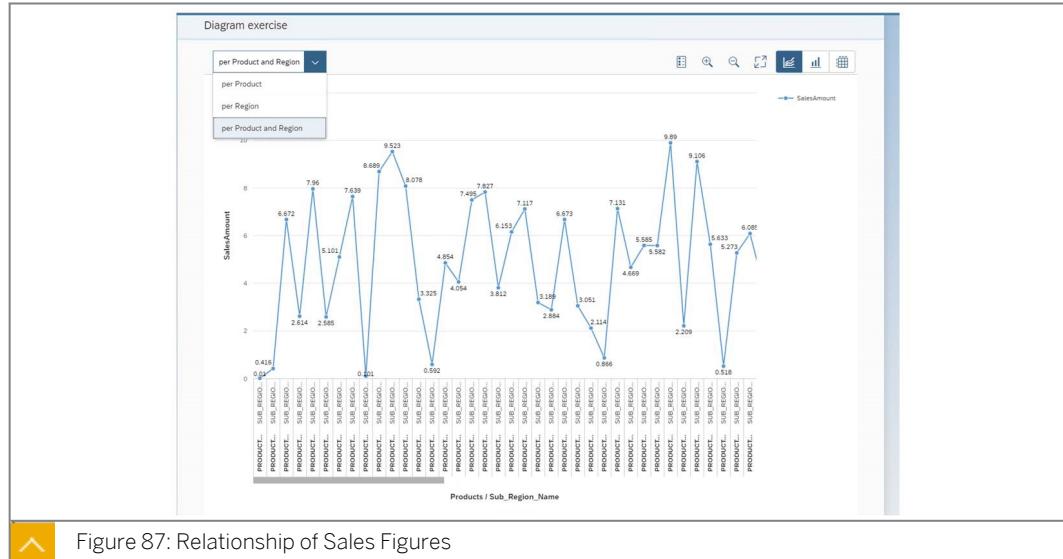


Figure 87: Relationship of Sales Figures

3. To display more meaningful data in the charts rename the file `SalesFigures.json` located in the `localService` folder to `SalesFigures_Gen.json`. Import the file `SalesFigures.json` located at `s:\courses\UX410_20\templates\diagram\` into the `localService` folder. Change the Mock Data configuration of your project so that the data from the `SalesFigures.json` file is used.

- a) Select the `SalesFigures.json` file from the context menu.

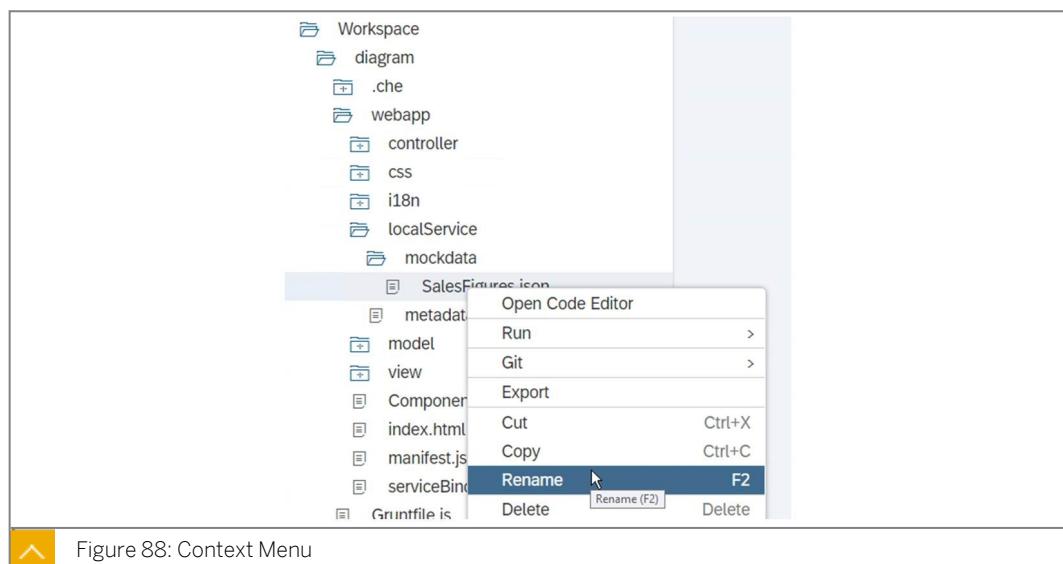
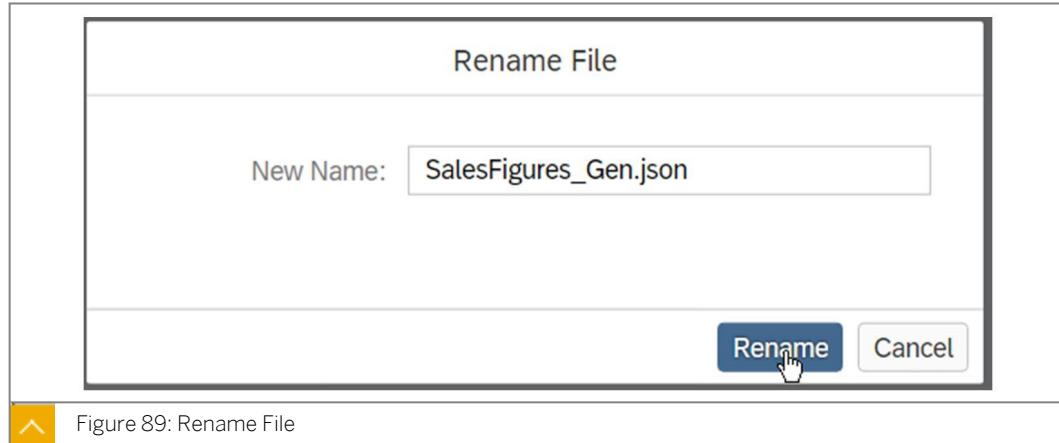
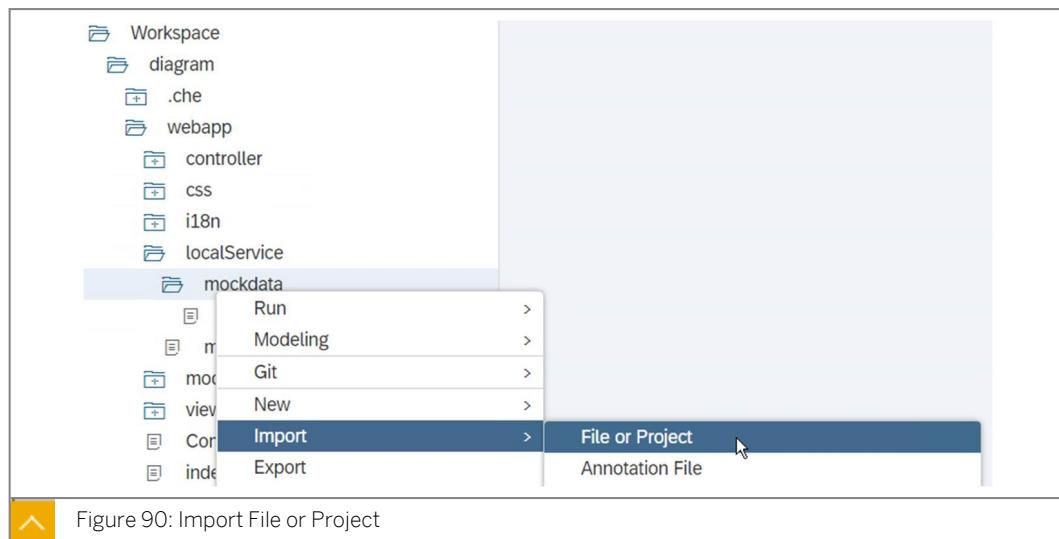


Figure 88: Context Menu

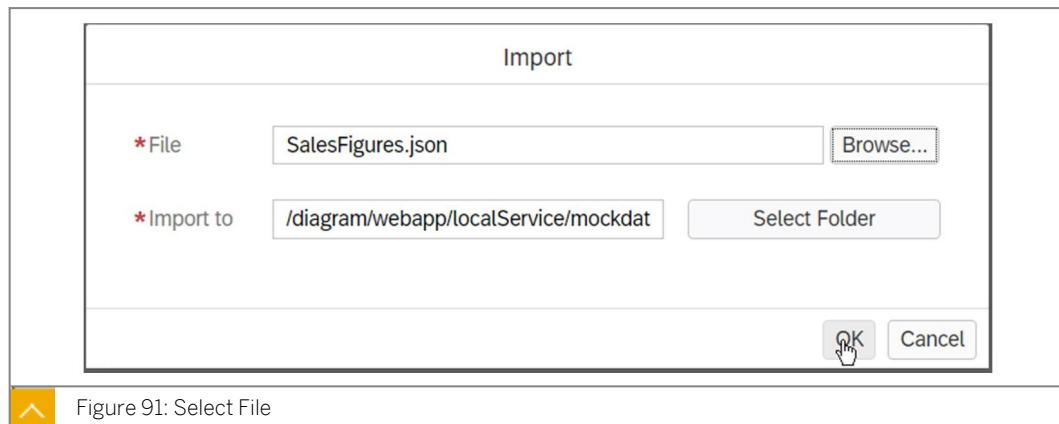
- b) Enter a full name in the dialog and choose *Rename*.



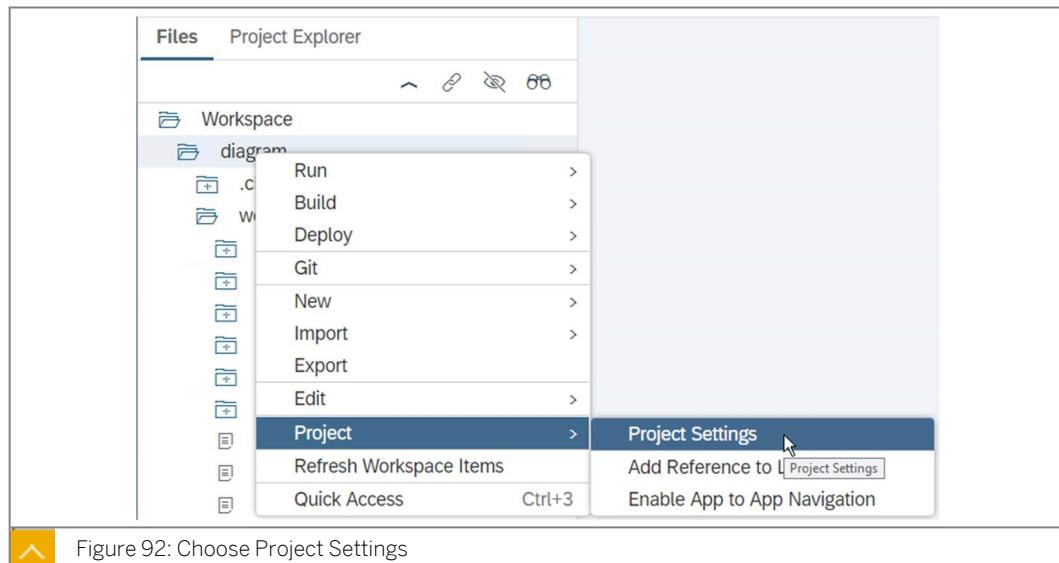
- c) Import File or Project and navigate to `S:\Courses\UX410_20\templates\diagram`.



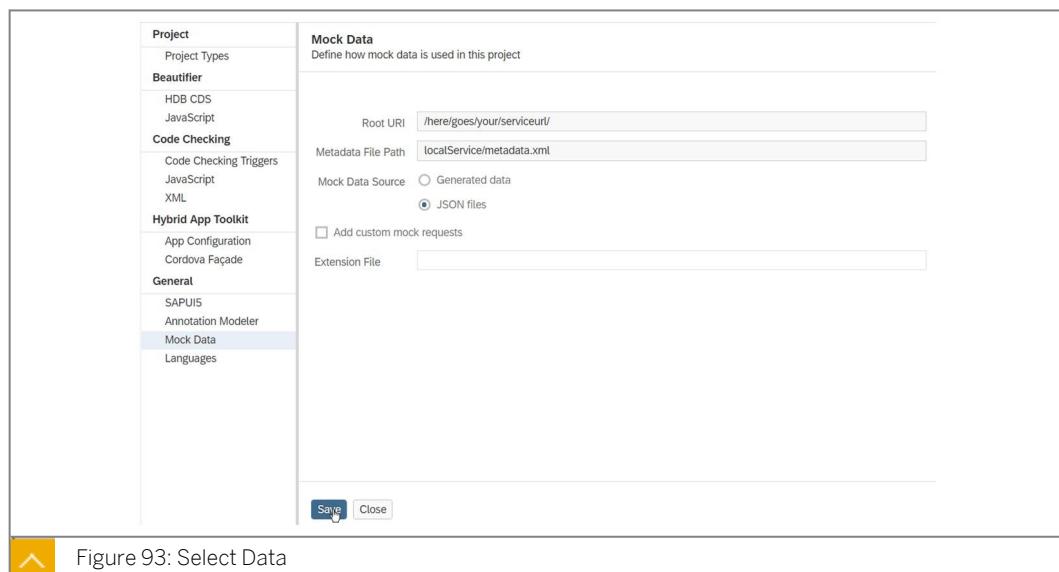
- d) Select the `SalesFigures.json` file and confirm `OK`.



- e) Choose `Project → Project Settings`.



f) Choose *Mock Data* with the option *JSON files* and *Save*.



g) Restart your application.



# Unit 10

## Exercise 8

## Navigate in SAP Fiori



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business Example

You need to implement two small SAPUI5 applications and deploy these applications on the front-end server. Then configure SAP Fiori tiles for both of them and implement a link in the first application that offers to navigate to the second application in the *SAP Fiori launchpad*.

You require an account on the front-end server.

### Task 1: Create SAP Fiori startnavigation and endnavigation Projects

Log on to the SAP Web IDE Fullstack and create two SAP Fiori projects named **startnavigation** and **endnavigation**. Add an *XML Main View* to both projects and insert UI controls.

1. Create an SAPUI5 project with the following settings.

Table 3: Startnavigation Properties

Parameter	Value
Project name	startnavigation
Namespace	student##.sap.training
View Type	XML
View Name	Main
SAPUI5 Version	1.60

2. Implement the basic **startnavigation** project by adding a new *sap.ui.layout.VerticalLayout* control to the page control of the *Main.view.xml* and inserting the following UI controls into the *VerticalLayout*.

Table 4: Insert *sap.m.Link* properties

Attribute	Value
id	idNavLink

Attribute	Value
text	Navigate to Fiori-app with link

Table 5: Insert sap.m.Button properties

Attribute	Value
text	Navigate to Fiori-app using event handler
press	onPress

3. Deploy the **startnavigation** project to the front-end server with the properties listed in the table.

Table 6: Deploy startnavigation Properties

Property	Value
Package	ZTRAIN_##
Application name	ZUX410NavStar##
Description	Navigation starter

4. Create a SAPUI5 project using the following settings.

Table 7: endnavigation Properties

Parameter	Value
Project name	endnavigation
Namespace	student##.sap.training
View Type	XML
View Name	Main
SAPUI5 Version	1.60

5. Implement the basic **endnavigation** project by adding a new `sap.ui.layout.VerticalLayoutcontrol` to the `page` control of `Main.view.xml` and insert the following UI controls to the `VerticalLayout`.

Table 8: sap.m.Label Properties

Attribute	Value
id	idinfo
text	Hello world

6. Deploy the **endnavigation** project from the front-end server. Use the table data.

Table 9: Deploy endnavigation Properties

Property	Value
Package	ZTRAIN_##
Application name	ZUX410NavEnd##
Description	Navigation target

**Task 2: Configure two tiles in the SAP Fiori Launchpad**

1. Create two semantic objects on the front-end server using the following details.

Table 10: Semantic Object, Name and Description

Semantic Object	Semantic Object Name	Semantic Object Description
UX410NavStart##	UX410NavStart##	Semantic Object for start app##
UX410NavEnd##	UX410NavEnd##	Semantic Object for end app##

2. Create a Catalog on the front-end server with the *SAP Fiori launchpad designer* using the attributes from the table: Catalog ID

Table 11: Catalog ID

Attribute	Value
Title	ZUX410_BC_##
ID	ZUX410_BC_##



Note:

You can find the *SAP Fiori launchpad designer* in the user menu.

3. Create a tile mapping for ZUX410NavStar00 using the NavStart table properties.

Table 12: Properties for NavStart Tile Mapping

Property	Value
Semantic Object	UX410NavStart##
Action	Display
Application Type	SAPUI5 Fiori App
Title	UX410 start navigation
URL	/sap/bc/ui5_ui5/sap/zux410navstar##

Property	Value
ID	student##.sap.training.startnavigation

4. Create a tile mapping for ZUX401NavEnd00 with the NavEnd table properties.

Table 13: Properties for NavEnd Tile Mapping

Property	Value
Semantic Object	UX410NavEnd##
Action	display
Application Type	SAPUI5 Fiori App
Title	UX410 endnavigation
URL	/sap/bc/ui5_ui5/sap/zux410navend##
ID	student##.sap.training.endnavigation

5. Create two static tiles with the following attributes.

Table 14: Navigation Start Tile

Attribute	Value
Subtitle	Group ##
Use semantic object navigation:	checked
Semantic Object:	UX410NavStart##

Table 15: Navigation End Tile

Attribute	Value
Subtitle:	Group ##
Use semantic object navigation:	checked
Semantic Object:	UX410NavEnd##

Two tile configurations are displayed.

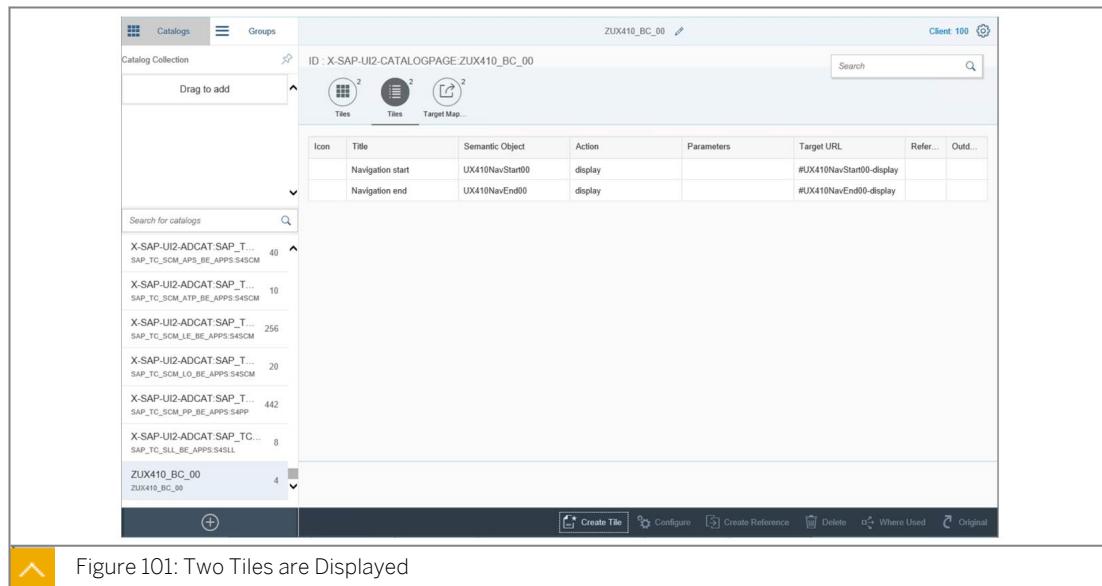


Figure 101: Two Tiles are Displayed

6. Create a new SAP role with the name **zux410\_BR##** of the type **Single Role** on the front-end server and add the new catalog to the role. Assign your user to the role.
7. Log on to the *SAP Fiori launchpad* and add the two new tiles to your homepage.

### Task 3: Use the CrossApplicationNavigation Service

Implement the link and button controls to navigate from the *Navigation start* application tile to the *Navigation end* application tile.

1. Start the *SAP Web IDE Full-Stack* to implement the `onInit` function of the starter application. Check the application is running the *SAP Fiori Launchpad*. Obtain a reference to the `CrossApplicationNavigation` service `ShellContainer` and store the reference in a member variable. Create the navigation link to the intent `UX410NavEnd##display` created in the previous task and assign the navigation link to the `href` attribute of the `sap.m.Link` control that was created in the previous task.
2. Implement the `onPress` event handler in the `Main.controller.js`. Use the reference to the `CrossApplicationNavigation` service and call the function `toExternal`. Pass the `UX410NavEnd##` as a semantic object and `display` as action to the function.
3. Update your deployment of the `startnavigation` project.
4. Log on to the *SAP Fiori launchpad* to test your implementation. Start your application and try the link and button to test the navigation.

### Task 4: Pass parameters to the navigation target

Extend the *start navigation application* by passing a parameter to the *end navigation application* during navigation.

1. Extend your implementation of the navigation in the *start navigation application* and pass a parameter with the name `helloText` during the navigation. Assign the string `Hello UX410` to the `helloText` parameter.
2. Update your deployment of the `startnavigation` project.

### Task 5: Extend the *endnavigation application* to fetch a parameter

Extend your **endnavigation** application so that when the user navigates from the **startnavigation** application to the **endnavigation** application, the passed parameter is fetched from the component and shown inside the UI.

1. Implement the `onInit` function of the `Main.controller.js` of the **endnavigation** project. Read the `helloText` parameter that is passed by the **startnavigation** application and assign the parameter value to the `text` attribute of the label that is part of the `Main.view.xml` implementation.
2. Update the deployed application of the **endnavigation** project.
3. Test your application.

You have now successfully implemented the exercise.

## Navigate in SAP Fiori



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business Example

You need to implement two small SAPUI5 applications and deploy these applications on the front-end server. Then configure SAP Fiori tiles for both of them and implement a link in the first application that offers to navigate to the second application in the *SAP Fiori launchpad*.

You require an account on the front-end server.

### Task 1: Create SAP Fiori startnavigation and endnavigation Projects

Log on to the SAP Web IDE Fullstack and create two SAP Fiori projects named **startnavigation** and **endnavigation**. Add an *XML Main View* to both projects and insert UI controls.

1. Create an SAPUI5 project with the following settings.

Table 3: Startnavigation Properties

Parameter	Value
Project name	startnavigation
Namespace	student##.sap.training
View Type	XML
View Name	Main
SAPUI5 Version	1.60

1. Create an SAPUI5 project with the following settings.
  - a) Select *File* → *New* → *Project from Template* → *SAPUI5 Application Project*, choose the correct SAPUI5 version and *Next*.
  - b) Enter **startnavigation** as the project name and **student##.sap.training** as the *Namespace* field and choose *Next*.
  - c) Choose *XML* as a *View Type*, enter **Main** as *View Name*, and choose *Finish*.
2. Implement the basic **startnavigation** project by adding a new *sap.ui.layout.VerticalLayout* control to the page control of the *Main.view.xml* and inserting the following UI controls into the *VerticalLayout*.

Table 4: Insert sap.m.Link properties

Attribute	Value
id	idNavLink
text	Navigate to Fiori-app with link

Table 5: Insert sap.m.Button properties

Attribute	Value
text	Navigate to Fiori-app using event handler
press	onPress

- Open the file `Main.view.xml`.
- Add an XML-namespace `alias` with the value `layout` and assign the value `sap.ui.layout`.
- Add an `sap.ui.layout.VerticalLayout` to the content aggregation of the generated `sap.m.Page` control.
- Add a control of type `sap.m.Link` to the `VerticalLayout` control and assign the attributes listed in the table, *Insert sap.m.LinkpProperties*.
- Add a control of type `sap.m.Button` and assign the button attributes listed in the table, *Insert sap.m.Button properties*.
- Your view implementation should now look like the following figure:



```

*Main.view.xml *
1 <mvc:View controllerName="student00.sap.training.startnavigation.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true"-
2   xmlns:sap="sap.m" xmlns:layout="sap.ui.layout">
3   <Shell id="shell">
4     <App id="app">
5       <pages>
6         <Page id="page" title="{i18n>title}">
7           <content>
8             <layout:VerticalLayout>
9               <Link id="idNavLink" text="Navigate to Fiori-app with link"/>
10              <Button press="onPress" text="Navigate to Fiori-app using event handler"/>
11            </layout:VerticalLayout>
12          </content>
13        </Page>
14      </pages>
15    </App>
16  </Shell>
17 </mvc:View>

```

Figure 95: View Implementation

- Save your changes.
- Deploy the `startnavigation` project to the front-end server with the properties listed in the table.

Table 6: Deploy startnavigation Properties

Property	Value
Package	ZTRAIN_##
Application name	ZUX410NavStar##

Property	Value
Description	Navigation starter

- a) Select the **startnavigation** project in the project explorer of the SAP Web IDE Full-Stack.
- b) Choose *Deploy* → *Deploy to SAPUI5 ABAP Repository* from the context menu.
- c) Select the system alias *FSD\_100* and enter your credentials for the front-end server.
- d) Select *Deploy a new application* and choose *Next*.
- e) Enter data from the table for the name and description fields using your own student and group number.
- f) Browse and search for the package *ZTRAIN\_##*, where *##* is your own group number.
- g) Choose *OK* and then *Next*.
- h) Check your dialog looks as shown.
- i) Select the transport request that is assigned to your group number from the list and choose *Next*.
- j) Start the deployment process by choosing *Finish*.

4. Create a SAPUI5 project using the following settings.

Table 7: endnavigation Properties

Parameter	Value
Project name	endnavigation
Namespace	student##.sap.training
View Type	XML
View Name	Main
SAPUI5 Version	1.60

- a) Select *File* → *New* → *Project from Template* → *SAPUI5 Application Project* , choose *SAPUI5-version 1.60* and choose *Next*.
- b) Name your project **endnavigation** and in the Namespace field enter **student##.sap.training** and choose *Next*.
- c) Select **XML** as a *View Type* and **Main** in the *View Name* field and choose *Finish*.

5. Implement the basic **endnavigation** project by adding a new `sap.ui.layout.VerticalLayoutcontrol` to the `page` control of `Main.view.xml` and insert the following UI controls to the `VerticalLayout`.

Table 8: sap.m.Label Properties

Attribute	Value
id	idinfo

Attribute	Value
text	Hello world

- Open the file `Main.view.xml`.
- Add XML-namespace alias with the value `layout` and assign the value `sap.ui.layout`.
- Add an `sap.ui.layout.VerticalLayout` to the content aggregation of the generated `sap.m.Page` control.
- Add a control `sap.m.Label` to the `VerticalLayout` control and assign the attributes listed in the table. Your implementation should now look like the following figure:



```

Main.view.xml x
1 <mvc:View controllerName="student00.sap.training.endnavigation.controller.Main" xmlns:html="http://www.w3.org/1999/xhtml"
2   xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m" xmlns:layout="sap.ui.layout">
3   <App id="idAppControl">
4     <pages>
5       <Page title="{i18n>title}">
6         <content>
7           <layout:VerticalLayout>
8             <Label id="idInfo" text="Hello world"/>
9           </layout:VerticalLayout>
10          </content>
11        </Page>
12      </pages>
13    </App>
14  </mvc:View>

```

Figure 96: Implementation

- Save your changes.

## 6. Deploy the `endnavigation` project from the front-end server. Use the table data.

Table 9: Deploy endnavigation Properties

Property	Value
Package	ZTRAIN_##
Application name	ZUX410NavEnd##
Description	Navigation target

- Select the `endnavigation` project in the *project explorer* of the SAP Web IDE Full-Stack.
- From the context menu, choose *Deploy* → *Deploy to SAPUI5 ABAP Repository*.
- Select the system alias `FSD_100` and enter your credentials for the front-end server.
- Select *Deploy a new application* and choose *Next*.
- Enter the name and description with the provided data using your own student and group number.
- Browse and search for the package `ZTRAIN_##`, where `##` is your own number, choose *OK* and then *Next..*.
- Select the transport request that is assigned to your group number from the list and choose *Next*.
- Start the deployment process by selecting *Finish*.

## Task 2: Configure two tiles in the SAP Fiori Launchpad

1. Create two semantic objects on the front-end server using the following details.

Table 10: Semantic Object, Name and Description

Semantic Object	Semantic Object Name	Semantic Object Description
UX410NavStart##	UX410NavStart##	Semantic Object for start app##
UX410NavEnd##	UX410NavEnd##	Semantic Object for end app##

- a) Start SAP GUI and log in to the front-end server using your user *Train-##* login details.
- b) From the user menu of the SAP Fiori folder, choose *Define Semantic Object – Customer* or use the transaction code `/UI2/SEMOBJ`.



Note:

Add the prefix `/n` if you start the transaction using the transaction code.

- c) Choose the edit icon  to switch to edit mode.
- d) Confirm the cross-client message.



Note:

If a message informs you that the table is currently locked by another user, this means that another course participant is currently running the transaction for their exercise. Confirm the message, wait until the other participant has finished, and then try again.

- e) Choose *New Entries* on the toolbar.
- f) Add the semantic object configuration with the data from the Table: Semantic Object, Name and Description.
- g) Save your changes and assign them to your transport request.
- h) Confirm the transport assignment dialog and close the transaction.



Note:

It is important to close the transaction as soon as you have finished your configuration. If you do not leave, you will lock the table.

2. Create a Catalog on the front-end server with the SAP Fiori launchpad designer using the attributes from the table: Catalog ID

Table 11: Catalog ID

Attribute	Value
Title	ZUX410_BC_##
ID	ZUX410_BC_##



## Note:

You can find the *SAP Fiori launchpad designer* in the user menu.

- a) Start the *SAP Fiori launchpad designer* (client-spec) from the start menu or use transaction **/UI2/FLPD\_CUST**.
  - b) When the SAP Fiori launchpad designer has started, assign a customizing request to record your changes by unchecking the checkbox *None* (Local Object), and selecting a customizing request. This is provided by your trainer from the list.
  - c) Choose  to create a new catalog.
  - d) Enter the table values in the dialog and choose *Save*.
3. Create a tile mapping for ZUX410NavStar00 using the NavStart table properties.

Table 12: Properties for NavStart Tile Mapping

Property	Value
Semantic Object	UX410NavStart##
Action	Display
Application Type	SAPUI5 Fiori App
Title	UX410 start navigation
URL	/sap/bc/ui5_ui5/sap/zux410navstar##
ID	student##.sap.training.startnavigation

- a) Select *Target Mapping* inside your new catalog.
  - b) Select *Create Target Mapping* in the footer.
  - c) When prompted for credentials for the back-end system, enter your user name and password and choose *OK*, then *Save*.
- After the successful creation of your target mapping, you will see the newly-created mapping in the list of available target mappings.

Figure 97: Configure Target mapping

4. Create a tile mapping for `zux410NavEnd00` with the `NavEnd` table properties.

Table 13: Properties for NavEnd Tile Mapping

Property	Value
Semantic Object	UX410NavEnd##
Action	display
Application Type	SAPUI5 Fiori App
Title	UX410 endnavigation
URL	/sap/bc/ui5_ui5/sap/zux410navend##
ID	student##.sap.training.endnavigation

- Select *Target Mapping* inside your new catalog.
- Select *Create Target Mapping* in the footer.
- When asked for credentials for the back-end system, enter your user name and password and choose *OK* and *Save*.  
After the successful creation of your target mapping, you will see the newly-created mapping in the list of available target mappings.
- Configure the target mapping.

Figure 98: Configuration of the Target Mapping

Your new target mapping is added to the list of available target mappings.

Figure 99: New Target Mapping

## 5. Create two static tiles with the following attributes.

Table 14: Navigation Start Tile

Attribute	Value
Subtitle	Group ##
Use semantic object navigation:	checked
Semantic Object:	UX410NavStart##

Table 15: Navigation End Tile

Attribute	Value
Subtitle:	Group ##
Use semantic object navigation:	checked
Semantic Object:	UX410NavEnd##

- Click on *Tiles* and choose *Create Tile* in the footer.
- Select *App Launcher – Static*.
- Enter the values as provided and Save.

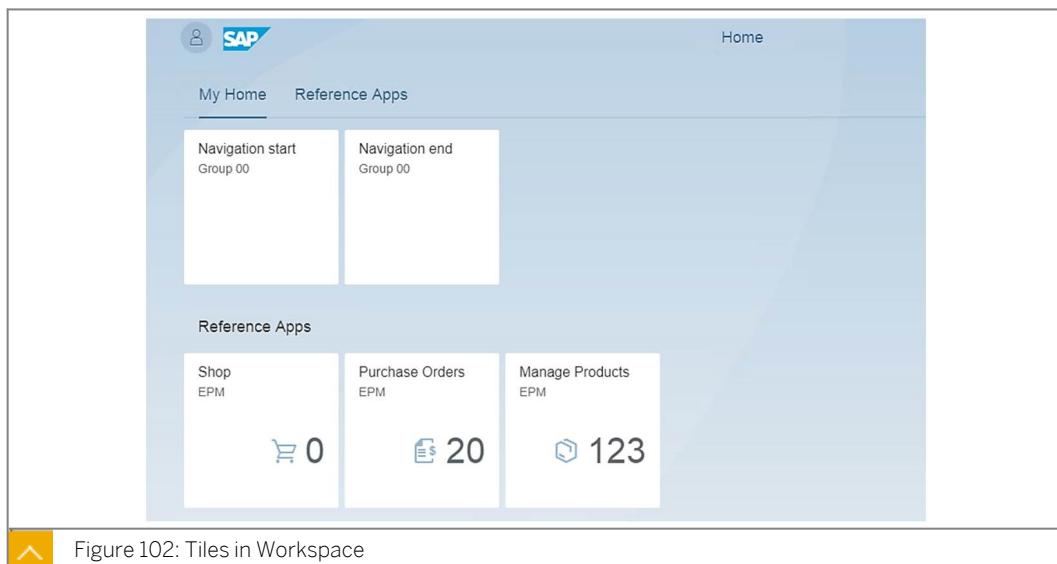
Figure 100: Enter Values to Configure the Tile

- Repeat to create a tile for the second target.

Two tile configurations are displayed.

Catalogs							Groups		ZUX410_BC_00							Client: 100									
Catalog Collection			ID : X-SAP-UI2-CATALOGPAGE.ZUX410_BC_00				Search			Icon		Title		Semantic Object		Action		Parameters		Target URL		Refer...		Outd...	
Drag to add												Navigation start		UX410NavStart00		display		#UX410NavStart00-display							
												Navigation end		UX410NavEnd00		display		#UX410NavEnd00-display							

6. Create a new SAP role with the name **zux410\_BR\_##** of the type **Single Role** on the front-end server and add the new catalog to the role. Assign your user to the role.
  - a) Log on to the front-end server using SAP GUI.
  - b) Start transaction **PFCG**.
  - c) Enter the role name **zux410\_BR\_##**, select the *Single Role* button and then **Save**.
  - d) Choose the *Menu* tab.
  - e) On the menu tab, choose the small arrow on the right of the *Transaction* button to open a drop-down list. Choose *SAP Fiori Tile Catalog*.
  - f) Choose your *SAP Fiori Catalog* from the *Value Help* and confirm the dialog.
  - g) Select the *User* tab and assign your **Train-##** user to the new role.
  - h) Save your changes.
7. Log on to the *SAP Fiori launchpad* and add the two new tiles to your homepage.
  - a) Open Google Chrome and from the bookmarks choose, *00 Development* → *fsdhost* → *FSD Fiori Launchpad (via WDF)*.
  - b) Log on to *SAP Fiori launchpad* using **Train-##**, with your group number and password.
  - c) When asked for credentials for the back-end system, enter your user name and password and choose *OK* and then **Save**.
  - d) Open the *Me* area.
  - e) Start the *App Finder*.
  - f) Choose your new catalog to see the two tiles display.
  - g) Choose the *Pin* icon and then pin to *My Home* and choose *OK*.
  - h) Repeat with the other tile and check that both pins are now highlighted.
  - i) Leave the *App Finder* and see both tiles in your workspace.

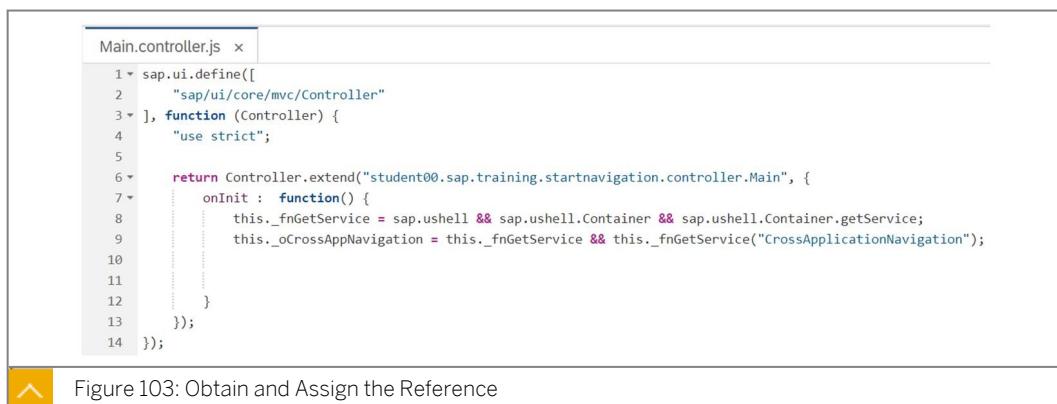


- j) Start each application tile to test your implementation.

### Task 3: Use the CrossApplicationNavigation Service

Implement the link and button controls to navigate from the *Navigation start* application tile to the *Navigation end* application tile.

1. Start the SAP Web IDE Full-Stack to implement the `onInit` function of the starter application. Check the application is running the *SAP Fiori Launchpad*. Obtain a reference to the `CrossApplicationNavigation` service `ShellContainer` and store the reference in a member variable. Create the navigation link to the intent `UX410NavEnd##` display created in the previous task and assign the navigation link to the `href` attribute of the `sap.m.Link` control that was created in the previous task.
  - a) Open the `Main.controller.js` file of the `startnavigation` project.
  - b) Create an empty implementation of the `onInit` function.
  - c) Check whether the application is running the *SAP Fiori launchpad*.
  - d) Obtain a reference to the `CrossApplicationNavigation` service and assign the reference to a member variable.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.startnavigation.controller.Main", {
7     onInit : function() {
8       this._fnGetService = sap.ushell && sap.ushell.Container && sap.ushell.Container.getService;
9       this._oCrossAppNavigation = this._fnGetService && this._fnGetService("CrossApplicationNavigation");
10
11     }
12   });
13 });
14 });

```

Figure 103: Obtain and Assign the Reference

- e) Invoke the function `hrefForExternal` and pass the `UX410NavEnd##` as an `semanticObject` and `display` as an action.
- f) Assign the return value of the `hrefForExternal` function to a local variable.



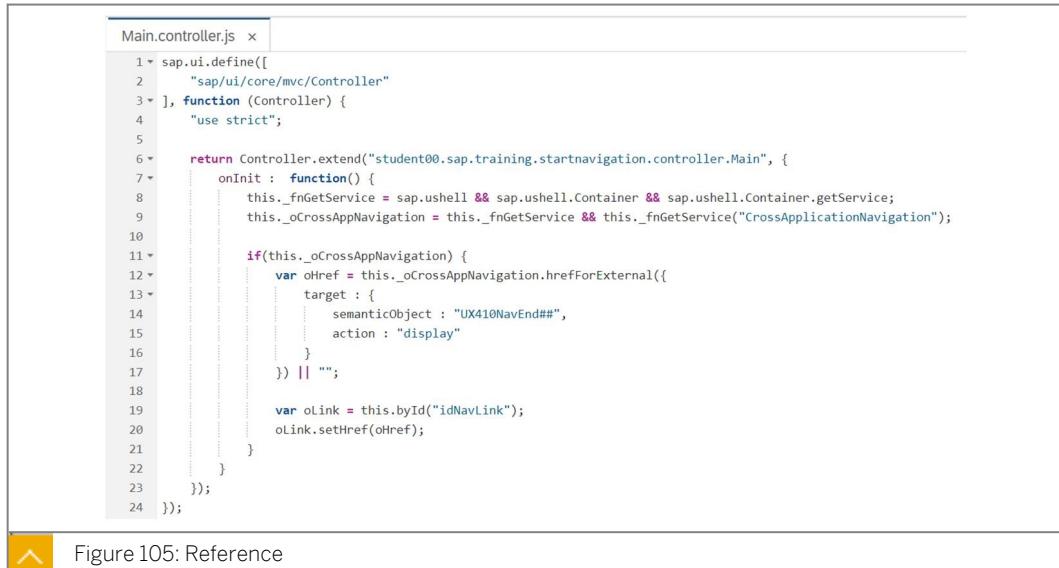
```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.startnavigation.controller.Main", {
7     onInit : function() {
8       this._fnGetService = sap.ushell && sap.ushell.Container && sap.ushell.Container.getService;
9       this._oCrossAppNavigation = this._fnGetService && this._fnGetService("CrossApplicationNavigation");
10
11       if(this._oCrossAppNavigation) {
12         var _href = this._oCrossAppNavigation.hrefForExternal({
13           target : {
14             semanticObject : "UX410NavEnd##",
15             action : "display"
16           }
17         }) || "";
18       }
19     });
20 });
21 });

```

Figure 104: Assign Return Value

- g) Obtain a reference to the `sap.m.Link` control from the `Main.view.xml` and assign the reference to a local variable.
- h) Call the `setHref` function on the obtained reference and pass the intent reference to the function.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.startnavigation.controller.Main", {
7     onInit: function () {
8       this._fnGetService = sap.ushell && sap.ushell.Container && sap.ushell.Container.getService;
9       this._oCrossAppNavigation = this._fnGetService && this._fnGetService("CrossApplicationNavigation");
10
11     if(this._oCrossAppNavigation) {
12       var oHref = this._oCrossAppNavigation.hrefForExternal({
13         target: {
14           semanticObject: "UX410NavEnd##",
15           action: "display"
16         }
17       }) || "";
18
19       var oLink = this.byId("idNavLink");
20       oLink.setHref(oHref);
21     }
22   });
23 });
24 });

```

Figure 105: Reference

2. Implement the `onPress` event handler in the `Main.controller.js`. Use the reference to the `CrossApplicationNavigation` service and call the function `toExternal`. Pass the `UX410NavEnd##` as a semantic object and display as action to the function.

- a) Open the `Main.controller.js` file.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.startnavigation.controller.Main", {
7     onInit: function () {
8       this._fnGetService = sap.ushell && sap.ushell.Container && sap.ushell.Container.getService;
9       this._oCrossAppNavigation = this._fnGetService && this._fnGetService("CrossApplicationNavigation");
10
11     if (this._oCrossAppNavigation) {
12       var oHref = this._oCrossAppNavigation.hrefForExternal({
13         target: {
14           semanticObject: "UX410NavEnd00",
15           action: "display"
16         },
17         params: {
18           "helloText": "Hello UX410"
19         }
20       }) || "";
21
22       var oLink = this.byId("idNavLink");
23       oLink.setHref(oHref);
24     },
25   });

```

Figure 106: Main.controller.js

- b) Implement a function with the name `onPress` as an event handler.

```
Main.controller.js x
1 sap.ui.define([
2     "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4     "use strict";
5
6     return Controller.extend("student00.sap.training.startnavigation.controller.Main", {
7         ...
8         ...
9         ...
10        ...
11        ...
12        ...
13        ...
14        ...
15        ...
16        ...
17        ...
18        ...
19        ...
20        ...
21        ...
22        ...
23        ...
24        ...
25        ...
26        ...
27        ...
28    });
});
```

Figure 107: onPress Function

c) Check the cross-navigation service is available and call the function `toExternal` on the service reference.

```
Main.controller.js x
1 * sap.ui.define([
2 *   "sap/ui/core/mvc/Controller"
3 * ], function (Controller) {
4 *   "use strict";
5 *
6 *   return Controller.extend("student00.sap.training.startnavigation.controller.Main", {
7 *     onInit : function() {
8 *       this._fnGetService = sap.ushell && sap.ushell.Container && sap.ushell.Container.getService;
9 *       this._oCrossAppNavigation = this._fnGetService("CrossApplicationNavigation");
10 *
11 *       if(this._oCrossAppNavigation) {
12 *         var oHref = this._oCrossAppNavigation.hrefForExternal({
13 *           target : {
14 *             semanticObject : "UX410NavEnd#",
15 *             action : "display"
16 *           }
17 *         }) || "";
18 *
19 *         var oLink = this.byId("idNavLink");
20 *         oLink.setHref(oHref);
21 *       }
22 *     },
23 *
24 *     onPress : function(oEvent) {
25 *       if(this._oCrossAppNavigation) {
26 *         this._oCrossAppNavigation.toExternal( {
27 *           target : {
28 *             semanticObject : "UX410NavEnd#",
29 *             action : "display"
30 *           }
31 *         });
32 *       }
33 *     }
34 *   });
35 * });

```

Figure 108: Call the Function toExternal

d) Pass `UX410NavEnd##` as a semantic object and the value `display` as an action.

e) Save your changes.

### 3. Update your deployment of the `startnavigation` project.

a) Select your **startnavigation** project in the project explorer and choose from the context menu *Deploy* → *Deploy to SAPUI5 ABAP Repository*.

b) Enter your user credentials for the front-end server when required.

c) Check the radio button *Update an existing application* and choose *Next*.

d) Check in the new dialog that the correct application is preselected and choose *Next*.

- e) Choose the *Finish* button to deploy the application.
  - f) Confirm the deployment by choosing *OK*.
  - g) Wait until the deployment is finished.
4. Log on to the *SAP Fiori launchpad* to test your implementation. Start your application and try the link and button to test the navigation.
- a) Select the bookmark in Google Chrome to start the *SAP Fiori launchpad*.
  - b) Enter your log on credentials.
  - c) Start the *Starter Application*.
  - d) Select the *Link* to display the target.
  - e) Navigate back to the Navigation start by using the back button of the *SAP Fiori launchpad*.
  - f) Select the *Button* control to display the target.

#### Task 4: Pass parameters to the navigation target

Extend the *start navigation application* by passing a parameter to the *end navigation application* during navigation.

1. Extend your implementation of the navigation in the *start navigation application* and pass a parameter with the name *helloText* during the navigation. Assign the string **Hello UX410** to the *helloText* parameter.
- a) Start the *SAP Web IDE Full-Stack*, if not already started.
  - b) Open the `Main.controller.js` file of the starter application.
  - c) Enhance the `onInit` function of your controller by passing the configuration `object params` to the `hrefForExternal` function and assign the parameter *helloText* with value **Hello UX410** to the parameter.



```

Main.controller.js ×
1+ sap.ui.define([
2+   "sap/ui/core/mvc/Controller"
3+ ], function (Controller) {
4+   "use strict";
5+
6+   return Controller.extend("student00.sap.training.startnavigation.controller.Main", {
7+     onInit: function () {
8+       this._fnGetService = sap.ushell || sap.ushell.Container || sap.ushell.Container.getService;
9+       this._oCrossAppNavigation = this._fnGetService("CrossApplicationNavigation");
10+
11+       if (this._oCrossAppNavigation) {
12+         var oHRef = this._oCrossAppNavigation.hrefForExternal({
13+           target: {
14+             semanticObject: "UX410NavEnd00",
15+             action: "display"
16+           },
17+           params: {
18+             "helloText": "Hello UX410"
19+           }
20+         }) || "";
21+
22+         var oLink = this.byId("idNavLink");
23+         oLink.setHref(oHRef);
24+       }
25+     },
26+
27+     onPress: function (oEvent) {
28+       if (this._oCrossAppNavigation) {
29+         this._oCrossAppNavigation.toExternal({
30+           target: {
31+             semanticObject: "UX410NavEnd00",
32+             action: "display"
33+           }
34+         });
35+       }
36+     }
37+   });
38+ });

```

Figure 109: Enhance the `onInit` Function

- d) Enhance the `onPress` function of your controller by passing the configuration `object params` to the `hrefForExternal` function, and assigning the parameter *helloText* with the value **Hello UX410** to the parameter.



```

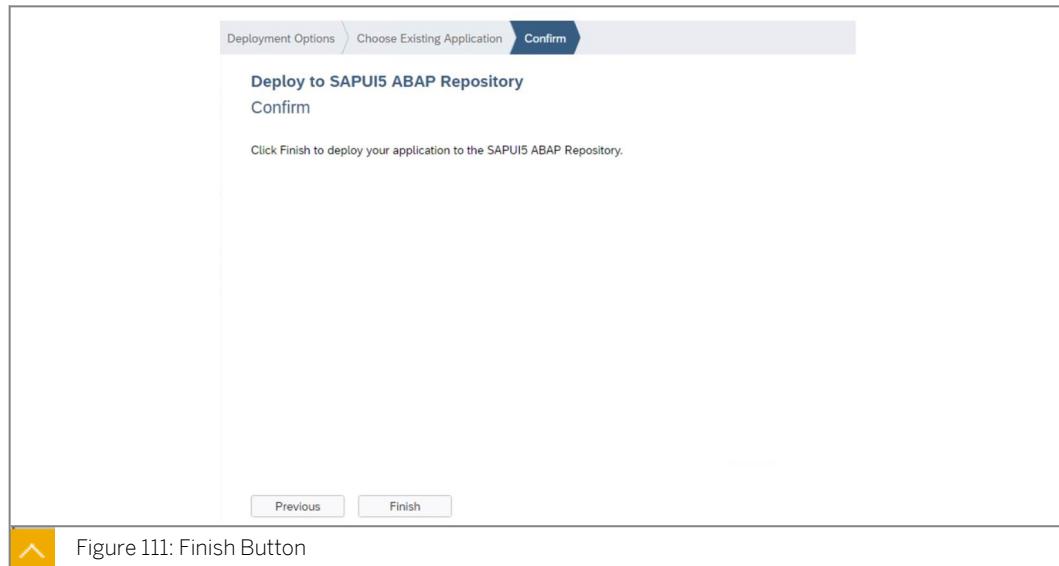
Main.controller.js ×
1> sap.ui.define([
2>   "sap/ui/core/mvc/Controller"
3> ], function (Controller) {
4>   "use strict";
5>
6>   return Controller.extend("student00.sap.training.startnavigation.controller.Main", {
7>     onInit: function () {
8>       this._fnGetService = sap.ushell && sap.ushell.Container && sap.ushell.Container.getService;
9>       this._oCrossAppNavigation = this._fnGetService("CrossApplicationNavigation");
10>
11>       if (this._oCrossAppNavigation) {
12>         var oLink = this._oCrossAppNavigation.hrefForExternal({
13>           target: {
14>             semanticObject: "UX410NavEnd00",
15>             action: "display"
16>           },
17>           params: {
18>             "HelloText" : "Hello UX410"
19>           }
20>         }) || "";
21>
22>         var oLink = this.byId("idNavLink");
23>         oLink.setHref(oLink);
24>
25>       },
26>
27>       onPress: function (oEvent) {
28>         if (this._oCrossAppNavigation) {
29>           this._oCrossAppNavigation.toExternal({
30>             target: {
31>               semanticObject: "UX410NavEnd00",
32>               action: "display"
33>             },
34>             params: {
35>               "HelloText" : "Hello UX410"
36>             }
37>           });
38>         }
39>       }
40>     });
41>   });

```

Figure 110: Enhance the onPress Function

## 2. Update your deployment of the **startnavigation** project.

- In the project explorer select your **startnavigation** project and select *Deploy* → *Deploy to SAPUI5 ABAP Repository* from the context menu.
- Enter your user credentials for the front-end server when required.
- Check the radio button *Update an existing application* and choose *Next*.
- Check the correct application is preselected in the dialog and choose *Next*.
- Choose to deploy the application with the *Finish* button.



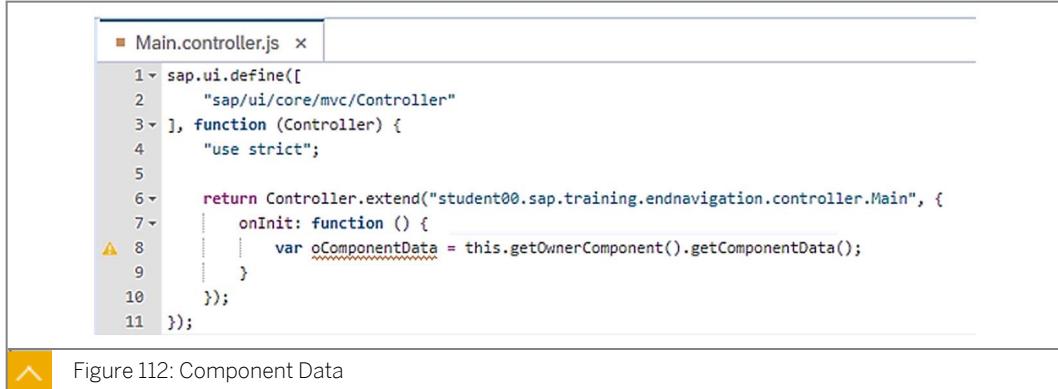
- Confirm the deployment by choosing *OK*.

- Wait for the deployment to finish.

### Task 5: Extend the endnavigation application to fetch a parameter

Extend your **endnavigation** application so that when the user navigates from the **startnavigation** application to the **endnavigation** application, the passed parameter is fetched from the component and shown inside the UI.

1. Implement the `onInit` function of the `Main.controller.js` of the `endnavigation` project. Read the `helloText` parameter that is passed by the `startnavigation` application and assign the parameter value to the `text` attribute of the label that is part of the `Main.view.xml` implementation.
  - a) Open the file `Main.controller.js` of the `endnavigation` project.
  - b) Implement an empty `onInit` function inside the controller implementation.
  - c) Obtain the component data and assign the component data to a local variable with the name `oComponentData`.



```

  Main.controller.js x
  1 sap.ui.define([
  2   "sap/ui/core/mvc/Controller"
  3 ], function (Controller) {
  4   "use strict";
  5
  6   return Controller.extend("student00.sap.training.endnavigation.controller.Main", {
  7     onInit: function () {
  8       var oComponentData = this.getOwnerComponent().getComponentData();
  9     }
 10   });
 11 });

```

Figure 112: Component Data

- a) Check if the `startupParameters` of the component contains the parameter `helloText`.



```

  Main.controller.js x
  1 sap.ui.define([
  2   "sap/ui/core/mvc/Controller"
  3 ], function (Controller) {
  4   "use strict";
  5
  6   return Controller.extend("student00.sap.training.endnavigation.controller.Main", {
  7     onInit: function () {
  8       var oComponentData = this.getOwnerComponent().getComponentData();
  9       if(oComponentData && oComponentData.startupParameters && oComponentData.startupParameters.helloText) {
 10       }
 11     }
 12   });
 13 });
 14 });

```

Figure 113: Check for Parameters

- a) Obtain the value of the parameter `helloText` and assign the value to a local variable.



```

  Main.controller.js x
  1 sap.ui.define([
  2   "sap/ui/core/mvc/Controller"
  3 ], function (Controller) {
  4   "use strict";
  5
  6   return Controller.extend("student00.sap.training.endnavigation.controller.Main", {
  7     onInit: function () {
  8       var oComponentData = this.getOwnerComponent().getComponentData();
  9       if(oComponentData && oComponentData.startupParameters && oComponentData.startupParameters.helloText) {
 10         var sHelloText = oComponentData.startupParameters.helloText[0];
 11       }
 12     }
 13   });
 14 });
 15 });

```

Figure 114: Obtain and Assign Value

- f) Obtain a reference to a UI control of type `sap.m.Label` with ID `idInfo` and assign the reference to a local variable.
- g) Call the `setText` function on the object and pass the `startup` parameter value to the `text` attribute of the UI control.



```

Main.controller.js x
1 ~ sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.endnavigation.controller.Main", {
7     onInit: function () {
8       var oComponentData = this.getOwnerComponent().getComponentData();
9       if(oComponentData && oComponentData.startupParameters && oComponentData.startupParameters.helloText) {
10         var sHelloText = oComponentData.startupParameters.helloText[0];
11         var oLabel = this.byId("idInfo");
12         oLabel.setText(sHelloText);
13       }
14     }
15   });
16 });

```

Figure 115: Call the `setText` Function

- h) Save your changes.
2. Update the deployed application of the endnavigation project.
    - a) In the project explorer select your endnavigation project and from the context menu, choose *Deploy* → *Deploy to SAPUI5 ABAP Repository*.
    - b) Enter your user credentials for the front-end server when required.
    - c) Select the radio button *Update an existing application* and choose *Next*.
    - d) Check in the next dialog that the correct application is preselected and choose *Next*.
    - e) Choose *Finish* to deploy your application.
    - f) Confirm the deployment by selecting *OK*.
    - g) Wait for the deployment to finish.
  3. Test your application.
    - a) Log on to the *SAP Fiori launchpad* or refresh and reload the current session.
    - b) Select the link on the *Navigation Start* tile.
    - c) Look for the **passed `startup`** parameter.
    - d) Navigate back to the *Navigation start* and select your new button to check if the button implementation works correctly.

You have now successfully implemented the exercise.

# Unit 11

## Exercise 9

### Create a Dynamic Page App

Create a Dynamic Page application using SAP Cloud Platform Developer Edition.



#### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The following figures show student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

Credentials to log on to the SAP Cloud Platform Developer Edition.

1. Create an SAPUI5 project with the following settings:

Table 16: Project Properties:

Parameter	Value
Project name	dynamicpage
Namespace	student##.sap.training
View Type	XML
View Name	App
SAPUI5 Version	1.60

2. Add the *UX\_TRAVEL\_SRV* OData service to your project. The service should be referenced by the default model of your application.
3. Remove the generated *sap.m.Page* control in the generated view of *App.view.xml* and remove the pages-aggregation of the generated App-control.
4. Add views of type XML to the project. Locate the *views* in the view folder and the *view controller* in the controller folder of your project

Table 17: View Name Properties

Namespace	View name
student##.sap.training.dynamicpage	Overview
student##.sap.training.dynamicpage	Carrier
student##.sap.training.dynamicpage	NotFound

5. Add the following translations as key value pairs to the default *i18n.properties* file of your project. Add the mentioned documentation note before each section.

Table 18: App Descriptor Properties

Key	Value
appTitle	Exercise:dynamic page app
appDescription	A simple dynamic page app

Table 19: Overview Views Properties

Key	Value
overviewPageTitle	Carriers
columnId	Id
columnName	Name

Table 20: Carrier View Properties

Key	Value
idLabelText	Id
nameLabelText	Name
currLabelText	Currency
urlLabelText	Url

Table 21: NotFound View Properties

Key	Value
notFoundMessageTitle	Not Found
notFoundMessageText	Sorry, but the requested resource is not available
notFoundMessageDescription	Please check the URL and try again.

6. Implement the `Overview.view.xml` file. The Overview page lists all available carriers from the entity set with the name `UX_C_Carrier_TP`. The Overview implements `sap.f.DynamicPage` control. The `DynamicPage` control displays a `sap.m.Table` control in the content aggregation. Use `f` XML namespace alias for the SAPUI5 namespace `sap.f`

Table 22: Attribute and Value pairs for the `sap.f.DynamicPage` control:

Attribute	Value
id	idDynamicPage
headerExpanded	true
toggleHeaderOnTitleClick	true

Table 23: Attribute and Value pairs for sap.m.Table control:

Attribute	Value
items	{UX_C_Carrier_TP}

Table 24: Attribute and Value pairs for sap.m.Title control:

Attribute	Value
text	{i18n>overviewPageTitle}
Level	H2

Table 25: Bind the text attribute to the values listed:

Attribute	Value
text	{i18n>columnId}
text	{i18n>columnName}

Table 26: Attribute and Value pairs for sap.m.ColumnListItem:

Attribute	Value
Press	onPress
Type	Navigation

Table 27: Attribute and Value pairs for a UI control sap.m.ObjectIdentifier:

Attribute	Value
title	{Carrid}

Table 28: Attribute and Value pairs for sap.m.Text:

Attribute	Value
text	{Carrname}

7. Implement the *Carrier* view using an *sap.f.DynamicPage* control. Use the *XML-namespace alias f* for *SAPUI5-namespace sap.f*. Add the listed attributes and values to the *sap.f.DynamicPage* control.

Table 29: Carrier View Properties:

Attribute	Value
id	idCarrierPage
headerExpanded	true

Attribute	Value
toggleHeaderOnTitleClick	true

Table 30: sap.m.Title Attribute and Value:

Attribute	Value
text	{Carrname}

Table 31: sap.m.text Control Attribute and Value:

Attribute	Value
text	{Carrid}

Table 32: Page Header Control Attribute and Value:

Attribute	Value
pinnable	true

Table 33: Flexbox Control Attribute and Value:

Attribute	Value
alignItems	Start
justifyContent	SpaceBetween

Table 34: HorizontalLayout Control Attribute and Value:

Attribute	Value
allowWrapping	true

Table 35: VerticalLayout Control Attribute and Value:

Attribute	Value
class	sapUiMediumMarginEnd

Table 36: ObjectAttribute 1 Control Attribute and Value:

Attribute	Value
title	{i18n>currLabelText}
title	{Currcode}

Table 37: ObjectAttribute 2 Control Attribute and Value:

Attribute	Value
title	{i18n>urlLabel}
title	{Url}

8. Implement `NotFound.view.xml`. Add an `sap.m.MessagePage` control with the following attributes and assign the listed values:

Table 38: Message Page Control Attribute/Value

Attribute	Value
title	{i18n>notFoundMessageTitle}
text	{i18n>notFoundMessageText}
description	{i18n>notFoundMessageDescription}
showNavButton	true
navButtonPress	onNavBack

9. Implement the controller of the carrier view including these functions:



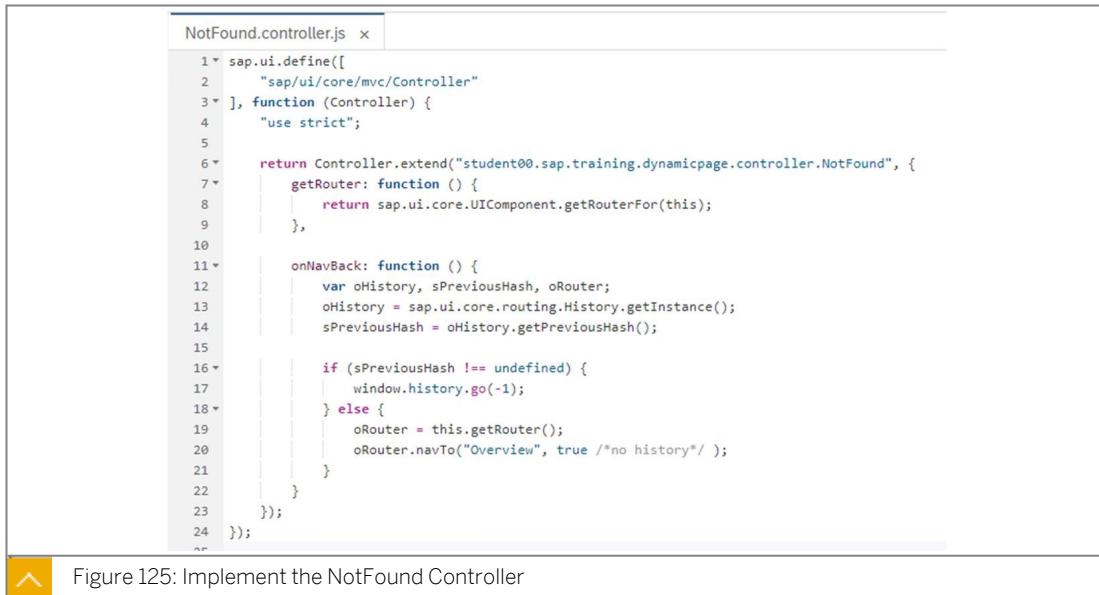
```

Carrier.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Carrier", {
7     ...
8     ...
9     ...
10    ...
11    ...
12    ...
13    ...
14    ...
15    ...
16    ...
17    ...
18    ...
19    ...
20    ...
21    ...
22    ...
23    ...
24    ...
25    ...
26    ...
27    ...
28    ...
29    ...
30    ...
31    ...
32    ...
33    ...
34    ...
35    ...
36    ...
37    ...
38    ...
39    ...
40    ...
41    ...
42    ...
43    ...
44    ...
45    ...
46    ...
47    ...
48    ...
49    ...
50    ...
51    ...
52    ...
53    ...
54    ...
55    ...
56    ...
57    ...
58    ...
59    ...
60    ...
61    ...
62    ...
63  });
}

```

Figure 120: Implement Controller

10. Implement the `NotFound` controller.



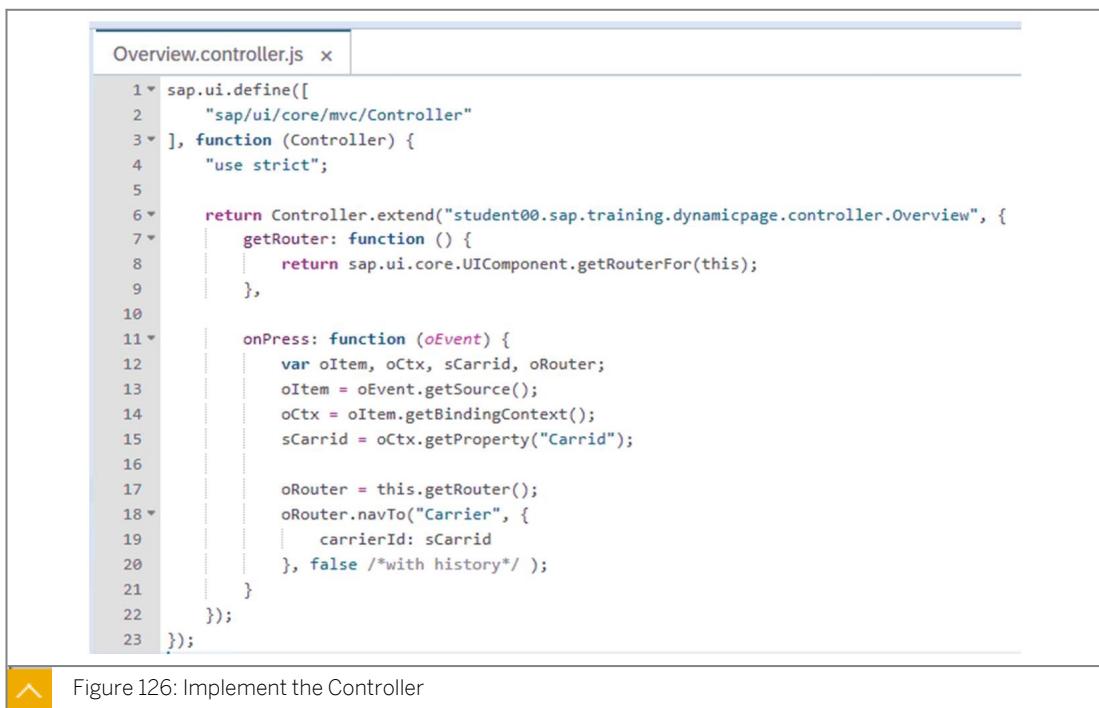
```

1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.NotFound", {
7     getRouter: function () {
8       return sap.ui.core.UIComponent.getRouterFor(this);
9     },
10
11    onNavBack: function () {
12      var oHistory, sPreviousHash, oRouter;
13      oHistory = sap.ui.core.routing.History.getInstance();
14      sPreviousHash = oHistory.getPreviousHash();
15
16      if (sPreviousHash !== undefined) {
17        window.history.go(-1);
18      } else {
19        oRouter = this.getRouter();
20        oRouter.navTo("Overview", true /*no history*/ );
21      }
22    }
23  });
24 });

```

Figure 125: Implement the NotFound Controller

11. Implement the *controller* of the *Overview* view.



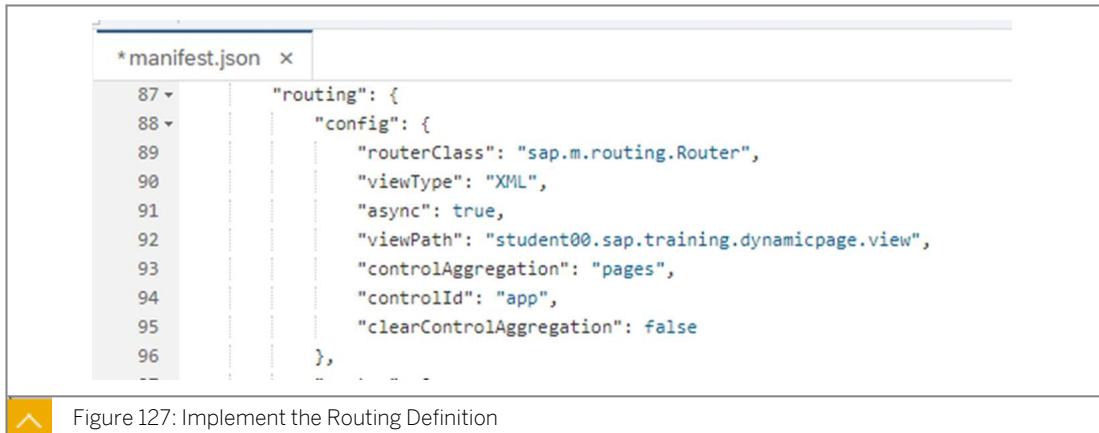
```

1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
7     getRouter: function () {
8       return sap.ui.core.UIComponent.getRouterFor(this);
9     },
10
11    onPress: function (oEvent) {
12      var oItem, oCtx, sCarrid, oRouter;
13      oItem = oEvent.getSource();
14      oCtx = oItem.getBindingContext();
15      sCarrid = oCtx.getProperty("Carrid");
16
17      oRouter = this.getRouter();
18      oRouter.navTo("Carrier", {
19        carrierId: sCarrid
20      }, false /*with history*/ );
21    }
22  });
23 });

```

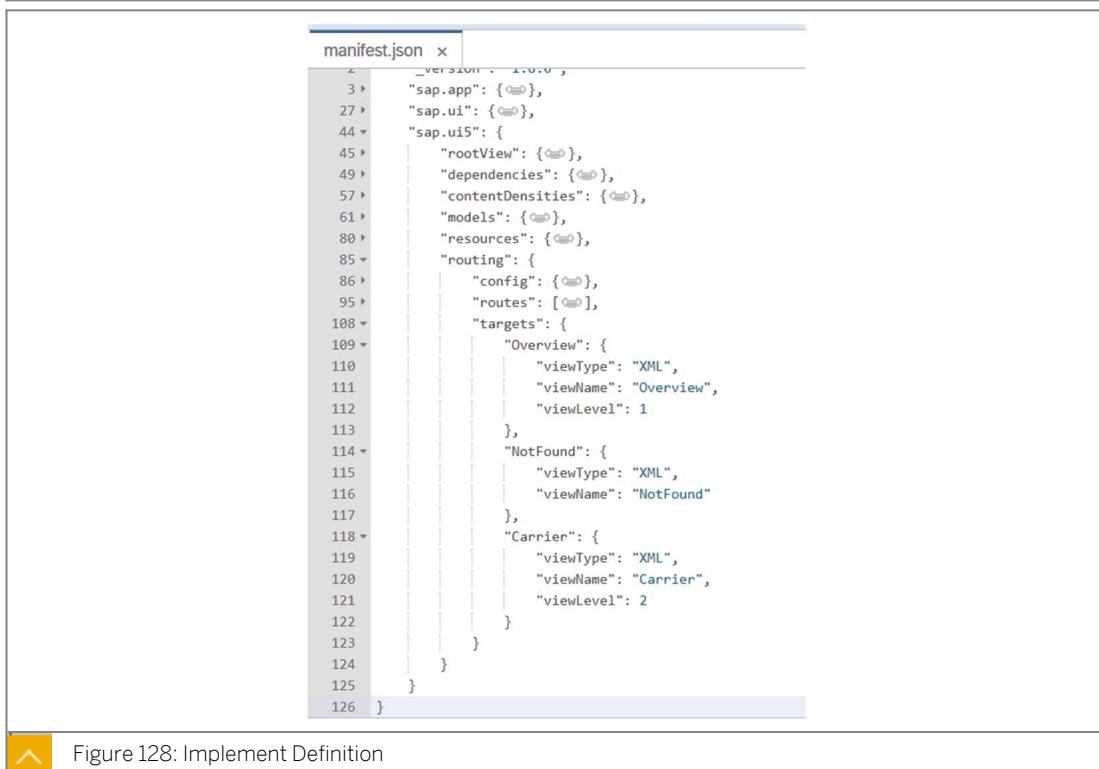
Figure 126: Implement the Controller

12. Implement the routing definition inside the `manifest.json`. Follow the routing and target configuration.



```
* manifest.json x
87  "routing": {
88    "config": {
89      "routerClass": "sap.m.routing.Router",
90      "viewType": "XML",
91      "async": true,
92      "viewPath": "student00.sap.training.dynamicpage.view",
93      "controlAggregation": "pages",
94      "controlId": "app",
95      "clearControlAggregation": false
96    },
97  }
98
```

Figure 127: Implement the Routing Definition



```
manifest.json x
1  "version": "1.0.0",
2  "sap.app": {
3    "rootView": {
4      "viewType": "XML",
5      "viewName": "Overview",
6      "viewLevel": 1
7    },
8    "dependencies": {
9      "minUI5Version": "1.62.0"
10   },
11   "contentDensities": {
12     "Density": "Medium"
13   },
14   "models": {
15     "odata": {
16       "uri": "sap Fiori Launchpad"
17     }
18   },
19   "resources": {
20     "css": {
21       "uri": "sap Fiori Launchpad"
22     }
23   }
24 },
25 "sap.ui": {
26   "deviceChaining": {
27     "order": 1
28   }
29 },
30 "sap.ui5": {
31   "dependencies": {
32     "minUI5Version": "1.62.0"
33   },
34   "resources": {
35     "css": {
36       "uri": "sap Fiori Launchpad"
37     }
38   }
39 },
40   "routing": {
41     "config": {
42       "routerClass": "sap.m.routing.Router",
43       "viewType": "XML",
44       "async": true,
45       "viewPath": "student00.sap.training.dynamicpage.view",
46       "controlAggregation": "pages",
47       "controlId": "app",
48       "clearControlAggregation": false
49     },
50     "routes": [
51       {
52         "name": "Overview",
53         "pattern": "Overview",
54         "target": "Overview"
55       },
56       {
57         "name": "NotFound",
58         "pattern": "NotFound",
59         "target": "NotFound"
60       }
61     ],
62     "targets": {
63       "Overview": {
64         "viewType": "XML",
65         "viewName": "Overview",
66         "viewLevel": 1
67       },
68       "NotFound": {
69         "viewType": "XML",
70         "viewName": "NotFound"
71       },
72       "Carrier": {
73         "viewType": "XML",
74         "viewName": "Carrier",
75         "viewLevel": 2
76       }
77     }
78   }
79 }
```

Figure 128: Implement Definition

13. Test your application.

## Create a Dynamic Page App

Create a Dynamic Page application using SAP Cloud Platform Developer Edition.



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The following figures show student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

Credentials to log on to the SAP Cloud Platform Developer Edition.

1. Create an SAPUI5 project with the following settings:

Table 16: Project Properties:

Parameter	Value
Project name	dynamicpage
Namespace	student##.sap.training
View Type	XML
View Name	App
SAPUI5 Version	1.60

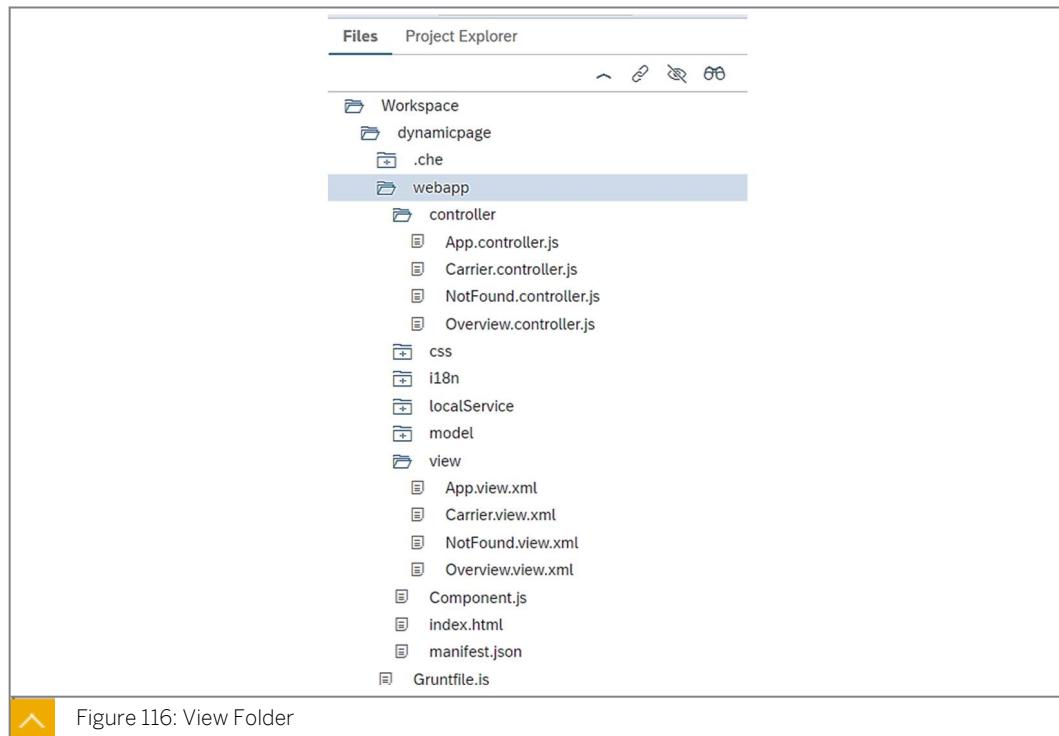
1. Create an SAPUI5 project with the following settings:
  - a) Log on to the SAP Web IDE Full-Stack.
  - b) Choose *File* → *New* → *Project from Template* and select the entry *SAPUI5 Application Project* and *Next*.
  - c) Name your project *dynamicpage* and give the namespace as *student##.sap.training*. Choose *Next*.
  - d) Choose *XML* as a *View Type* and give the value **App** in the field *View Name* and choose *Finish*.
  - e) When your project is created, you can find it in the project explorer of the SAP Web IDE Full-Stack.
2. Add the *UX\_TRAVEL\_SRV OData* service to your project. The service should be referenced by the default model of your application.
  - a) Select your project and from the context menu, choose *New-OData service*.
  - b) From the list of available systems, choose the *FSD\_100 – FSD system*.
  - c) Enter your user name and password for the front-end server if required.

- d) Search for the *UX\_TRAVEL\_SRV* service and choose the service from the list of results and choose *Next*.
  - e) Select the *Radio* button labelled *Use default model* and choose *Next*.
  - f) Complete the data source assignment by choosing *Finish*.
3. Remove the generated *sap.m.Page* control in the generated view of *App.view.xml* and remove the pages-aggregation of the generated App-control.
- a) Open the file *App.view.xml* in the view folder of your project.
  - b) Remove the *pages* aggregation and the *sap.m.Page* control with the content aggregation.
  - c) Shorten the app tag implementation.
  - d) Save your changes.
4. Add views of type *XML* to the project. Locate the *views* in the view folder and the *view controller* in the controller folder of your project

Table 17: View Name Properties

Namespace	View name
student##.sap.training.dynamicpage	Overview
student##.sap.training.dynamicpage	Carrier
student##.sap.training.dynamicpage	NotFound

- a) Select the *webapp* folder of your project and from the context menu, choose *New SAPUI5 View*.
- b) Select *XML* as the *view type*. Enter the name of your project in **namespace** and *overview* in the *View Name* field. Choose *Next* and *Finish* to exit the view creation dialog.
- c) Repeat steps a and b to create the listed views.
- d) Your controller and view folders display with your views created as shown.



5. Add the following translations as key value pairs to the default `i18n.properties` file of your project. Add the mentioned documentation note before each section.

Table 18: App Descriptor Properties

Key	Value
appTitle	Exercise:dynamic page app
appDescription	A simple dynamic page app

Table 19: Overview Views Properties

Key	
overviewPageTitle	Carriers
columnId	Id
columnName	Name

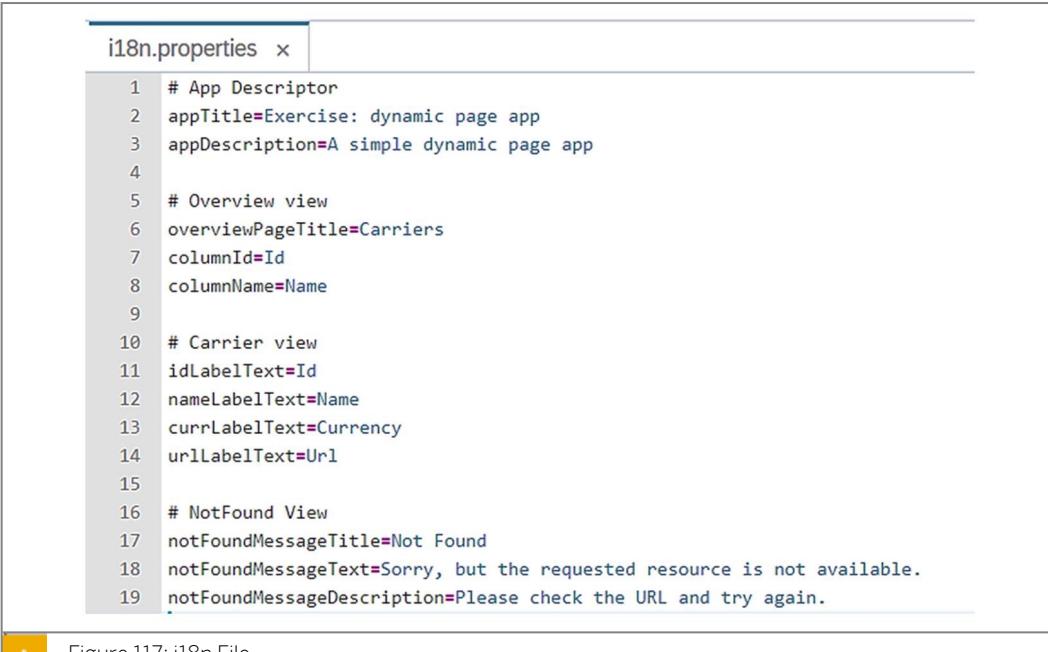
Table 20: Carrier View Properties

Key	Value
idLabelText	Id
nameLabelText	Name
currLabelText	Currency
urlLabelText	Url

Table 21: NotFound View Properties

Key	Value
notFoundMessageTitle	Not Found
notFoundMessageText	Sorry, but the requested resource is not available
notFoundMessageDescription	Please check the URL and try again.

- Open the file `i18n.properties` in the `i18n` folder of your project.
- Add the listed key value pairs from the tables to the files.
- The files appear as shown with the key-value pairs and comments added.



```

i18n.properties x
1 # App Descriptor
2 appTitle=Exercise: dynamic page app
3 appDescription=A simple dynamic page app
4
5 # Overview view
6 overviewPageTitle=Carriers
7 columnId=Id
8 columnName=Name
9
10 # Carrier view
11 idLabelText=Id
12 nameLabelText=Name
13 currLabelText=Currency
14 urlLabelText=Url
15
16 # NotFound View
17 notFoundMessageTitle=Not Found
18 notFoundMessageText=Sorry, but the requested resource is not available.
19 notFoundMessageDescription=Please check the URL and try again.

```

- Save your changes.
6. Implement the `Overview.view.xml` file. The Overview page lists all available carriers from the entity set with the name `UX_C_Carrier_TP`. The Overview implements `sap.f.DynamicPage` control. The DynamicPage control displays a `sap.m.Table` control in the content aggregation. Use `XML namespace alias` for the SAPUI5-namespace `sap.f`

Table 22: Attribute and Value pairs for the `sap.f.DynamicPage` control:

Attribute	Value
id	idDynamicPage
headerExpanded	true
toggleHeaderOnTitleClick	true

Table 23: Attribute and Value pairs for sap.m.Table control:

Attribute	Value
items	{UX_C_Carrier_TP}

Table 24: Attribute and Value pairs for sap.m.Title control:

Attribute	Value
text	{i18n>overviewPageTitle}
Level	H2

Table 25: Bind the text attribute to the values listed:

Attribute	Value
text	{i18n>columnId}
text	{i18n>columnName}

Table 26: Attribute and Value pairs for sap.m.ColumnListItem:

Attribute	Value
Press	onPress
Type	Navigation

Table 27: Attribute and Value pairs for a UI control sap.m.ObjectIdentifier:

Attribute	Value
title	{Carrid}

Table 28: Attribute and Value pairs for sap.m.Text:

Attribute	Value
text	{Carrname}

- Open the file `Overview.view.xml` in the view folder of your project and remove the default implementation of the view.
- Add a *XML-namespace alias* for the *SAPUI5-namespace* `sap.f` to the view.
- Add a UI control of type `sap.f.DynamicPage` with the listed attribute and value pairs.
- Add the *content* aggregation to the `sap.f.DynamicPage` control.
- Add an `sap.m.Table` control with the listed attributes to the content aggregation.
- Add the *headerToolbar* aggregation to the table control.

- g) Add an *sap.m.Toolbar* control to the *headerToolbar* aggregation and insert the *content* aggregation in the *sap.m.Toolbar*.
- h) Add an *sap.m.Title* control to the *content* aggregation of the *sap.m.Toolbar* with the listed attributes.
- i) Add the *columns* aggregation to the *sap.m.Table* control.
- j) Add two *sap.m.Column* controls to the *columns* aggregation and add to each of the *column* controls a control of type *sap.m.Text* with the listed attribute and value pairs.
- k) Add the *items* aggregation to the *sap.m.Table* control.
- l) Add a control of type *sap.m.ColumnListItem* as a template control and add the listed attribute and values to the control.
- m) Add the *cells* aggregation to *sap.m.ColumnListItemcontrol*.
- n) Add the *sap.m.ObjectIdentifier* and the *sap.m.Text* control to the *cells* aggregation and add the listed attribute and value pairs.
- o) Your implementation should like the following figure:



```

* Overview.view.xml *
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2   controllerName="student00.sap.training.dynamicpage.controller.Overview"
3   xmlns:html="http://www.w3.org/1999/xhtml">
4   <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
5     <f:content>
6       <Table items="{/UX_C_Carrier_TP}">
7         <headerToolbar>
8           <Toolbar>
9             <content>
10             <Title text="{i18n>overviewPageTitle}" level="H2"/>
11           </content>
12         </Toolbar>
13       </headerToolbar>
14       <columns>
15         <Column>
16           <Text text="{i18n>columnId}" />
17         </Column>
18         <Column>
19           <Text text="{i18n>columnName}" />
20         </Column>
21       </columns>
22       <items>
23         <ColumnListItem press="onPress" type="Navigation">
24           <cells>
25             <ObjectIdentifier title="{Carrid}" />
26             <Text text="{Carrname}" />
27           </cells>
28         </ColumnListItem>
29       </items>
30     </Table>
31   </f:content>
32 </f:DynamicPage>
33 </mvc:View>

```

Figure 118: Add Controls with Attributes

- p) Save your changes.
7. Implement the *Carrier* view using an *sap.f.DynamicPage* control. Use the *XML-namespace alias f* for SAPUI5-namespace *sap.f*. Add the listed attributes and values to the *sap.f.DynamicPage* control.

Table 29: Carrier View Properties:

Attribute	Value
id	idCarrierPage
headerExpanded	true

Attribute	Value
toggleHeaderOnTitleClick	true

Table 30: sap.m.Title Attribute and Value:

Attribute	Value
text	{Carrname}

Table 31: sap.m.text Control Attribute and Value:

Attribute	Value
text	{Carrid}

Table 32: Page Header Control Attribute and Value:

Attribute	Value
pinnable	true

Table 33: Flexbox Control Attribute and Value:

Attribute	Value
alignItems	Start
justifyContent	SpaceBetween

Table 34: HorizontalLayout Control Attribute and Value:

Attribute	Value
allowWrapping	true

Table 35: VerticalLayout Control Attribute and Value:

Attribute	Value
class	sapUiMediumMarginEnd

Table 36: ObjectAttribute 1 Control Attribute and Value:

Attribute	Value
title	{i18n>currLabelText}
title	{Currcode}

Table 37: ObjectAttribute 2 Control Attribute and Value:

Attribute	Value
title	{i18n>urlLabel}
title	{Url}

- a) Open the file `Carrier.view.xml` in the view folder of your project and remove the default implementation of the view.
- b) Add an *XML-namespace alias* `f` to the view control and assign *SAPUI5-namespace* `sap.f`.
- c) Add the UI control `sap.f.DynamicPage` to the view control and assign the listed attribute and values to the newly added control.
- d) Add the `title` aggregation to the newly added `sap.f.DynamicPage` control.
- e) Add a UI control of type `sap.f.DynamicPageTitle` to the `title` aggregation.
- f) Add the `heading` aggregation to the `sap.f.DynamicPageTitle` control.
- g) Add an `sap.m.Title` control to the `heading` aggregation and add the `text` attribute to the control. Assign the binding to the property `Carrname`.
- h) Add the `expandedContent` aggregation to the `sap.f.DynamicPageTitle` control.
- i) Add an `sap.m.Title` control to the `expandedContent` aggregation and add the `text` attribute to the control. Assign the binding to the property `Carrid`.
- j) Add the `snappedContent` aggregation to the `sap.f.DynamicPageTitle` control.
- k) Add an `sap.m.Title` control to the `snappedContent` aggregation and add the `text` attribute to the control. Assign the binding to the property `Carrid`.
- l) Add the `header` aggregation to the `sap.f.DynamicPage` control.
- m) Add a control of type `sap.f.DynamicPageHeader` to the `header` aggregation. Add the attribute with the name `pinnable` and assign the value `true`.
- n) Add the `content` aggregation to the `sap.f.DynamicPageHeader` control.
- o) Add an `sap.m.FlexBox` control to the `content` aggregation of the `sap.f.DynamicPageHeader` control. Add the attribute `alignItems` to the `sap.m.FlexBox` control and assign the value `Start`. Add the attribute `justifyContent` and assign the value `SpaceBetween`.
- p) Add an *XML-namespace alias* with name `layout` and assign the value `sap.ui.layout`.
- q) Add the `sap.ui.layout.HorizontalLayout` to the `items` aggregation. Add the attribute `allowWrapping` to the `sap.ui.layout.HorizontalLayout` control and assign the value `true`.
- r) Add an `sap.ui.layout.VerticalLayout` control to the `HorizontalLayout` control. Add the attribute `class` to the `sap.ui.layout.VerticalLayout` control and assign the value `sapUiMediumMarginEnd`.

- s) Add an *sap.m.ObjectAttribute* control to the *sap.ui.layout.VerticalLayout* control. Add the attribute *title* and assign the binding to key *currLabelText* of the *ResourceModel* *i18n*. Add the control *text* and bind the attribute to the property *Currcode* of the entity.
- t) Add an *sap.m.ObjectAttribute* control to the *sap.ui.layout.VerticalLayout* control. Add the attribute *title* and assign the binding to key **urlLabel** of the *ResourceModel* *i18n*. Add the control *text* and bind the attribute to the property *Url* of the entity.
- u) Your implementation should look like the following figure:



```

Carrier.view.xml
1  <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2    xmlns:f="sap.f" xmlns:layout="sap.ui.layout"
3    controllerName="student00.sap.training.dynamicpage.controller.Carrier"
4    xmlns:html="http://www.w3.org/1999/xhtml">
5    <f:DynamicPage id="dynamicPage1" headerExpanded="true" toggleHeaderOnTitleClick="true">
6      <f:title>
7        <DynamicPageTitle>
8          <f:heading>
9            <Title text="{Carrname}" />
10         </f:heading>
11         <f:expandedContent>
12           <Text text="{Carrid}" />
13         </f:expandedContent>
14         <f:snappedContent>
15           <Text text="{travel>Carrid}" />
16         </f:snappedContent>
17       </DynamicPageTitle>
18     </f:title>
19     <f:header>
20       <DynamicPageHeader pinnable="true">
21         <f:content>
22           <FlexBox alignItems="Start" justifyContent="SpaceBetween">
23             <items>
24               <layout:HorizontalLayout allowWrapping="true">
25                 <layout:VerticalLayout class="sapUiMediumMarginEnd">
26                   <ObjectAttribute title="{i18n>currLabelText}" text="{Currcode}" />
27                 </layout:VerticalLayout>
28               </layout:HorizontalLayout>
29             </items>
30           </FlexBox>
31         </f:content>
32       </DynamicPageHeader>
33     </f:header>
34   </f:DynamicPage>
35 </DynamicPage>
36 </mvc:View>

```

Figure 119: Added Controls and Attributes

- v) Save your work.

8. Implement *NotFound.view.xml*. Add an *sap.m.MessagePage* control with the following attributes and assign the listed values:

Table 38: Message Page Control Attribute/Value

Attribute	Value
title	{i18n>notFoundMessageTitle}
text	{i18n>notFoundMessageText}
description	{i18n>notFoundMessageDescription}
showNavButton	true
navButtonPress	onNavBack

- a) Open the file *NotFound.view.xml* and remove the existing *sap.m.App* control.
- b) Add an *sap.m.MessagePage* control to the view implementation and use the listed attributes.
- c) Save your changes.

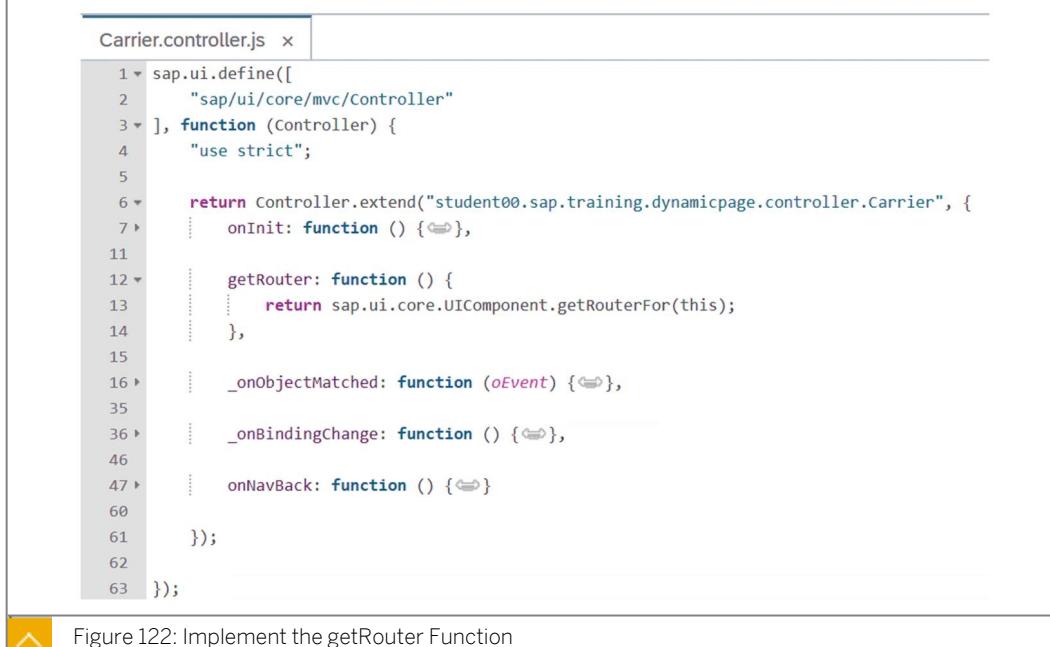
**9. Implement the controller of the carrier view including these functions:**

```
Carrier.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Carrier", {
7     ...
8     ...
9     ...
10    ...
11    ...
12    ...
13    ...
14    ...
15    ...
16    ...
17    ...
18    ...
19    ...
20    ...
21    ...
22    ...
23    ...
24    ...
25    ...
26    ...
27    ...
28    ...
29    ...
30    ...
31    ...
32    ...
33    ...
34    ...
35    ...
36    ...
37    ...
38    ...
39    ...
40    ...
41    ...
42    ...
43    ...
44    ...
45    ...
46    ...
47    ...
48    ...
49    ...
50    ...
51    ...
52    ...
53    ...
54    ...
55    ...
56    ...
57    ...
58    ...
59    ...
60    ...
61    ...
62    ...
63  }));
});
```

 Figure 120: Implement Controller**a) Implement the *onInit* function.**

```
Carrier.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Carrier", {
7     ...
8     ...
9     ...
10    ...
11    ...
12    ...
13    ...
14    ...
15    ...
16    ...
17    ...
18    ...
19    ...
20    ...
21    ...
22    ...
23    ...
24    ...
25    ...
26    ...
27    ...
28    ...
29    ...
30    ...
31    ...
32    ...
33    ...
34    ...
35    ...
36    ...
37    ...
38    ...
39    ...
40    ...
41    ...
42    ...
43    ...
44    ...
45    ...
46    ...
47    ...
48    ...
49    ...
50    ...
51    ...
52    ...
53    ...
54    ...
55    ...
56    ...
57    ...
58    ...
59    ...
60    ...
61    ...
62    ...
63  }));
});
```

 Figure 121: Implement the *onInit* Function**b) Implement the *getRouter* function.**



```

Carrier.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Carrier", {
7     ...
8     ...
9     ...
10    ...
11    ...
12    ...
13    ...
14    ...
15    ...
16    ...
17    ...
18    ...
19    ...
20    ...
21    ...
22    ...
23    ...
24    ...
25    ...
26    ...
27    ...
28    ...
29    ...
30    ...
31    ...
32    ...
33    ...
34    ...
35    ...
36    ...
37    ...
38    ...
39    ...
40    ...
41    ...
42    ...
43    ...
44    ...
45    ...
46    ...
47    ...
48    ...
49    ...
50    ...
51    ...
52    ...
53    ...
54    ...
55    ...
56    ...
57    ...
58    ...
59    ...
60    ...
61    ...
62    ...
63  });
});

```

Figure 122: Implement the getRouter Function

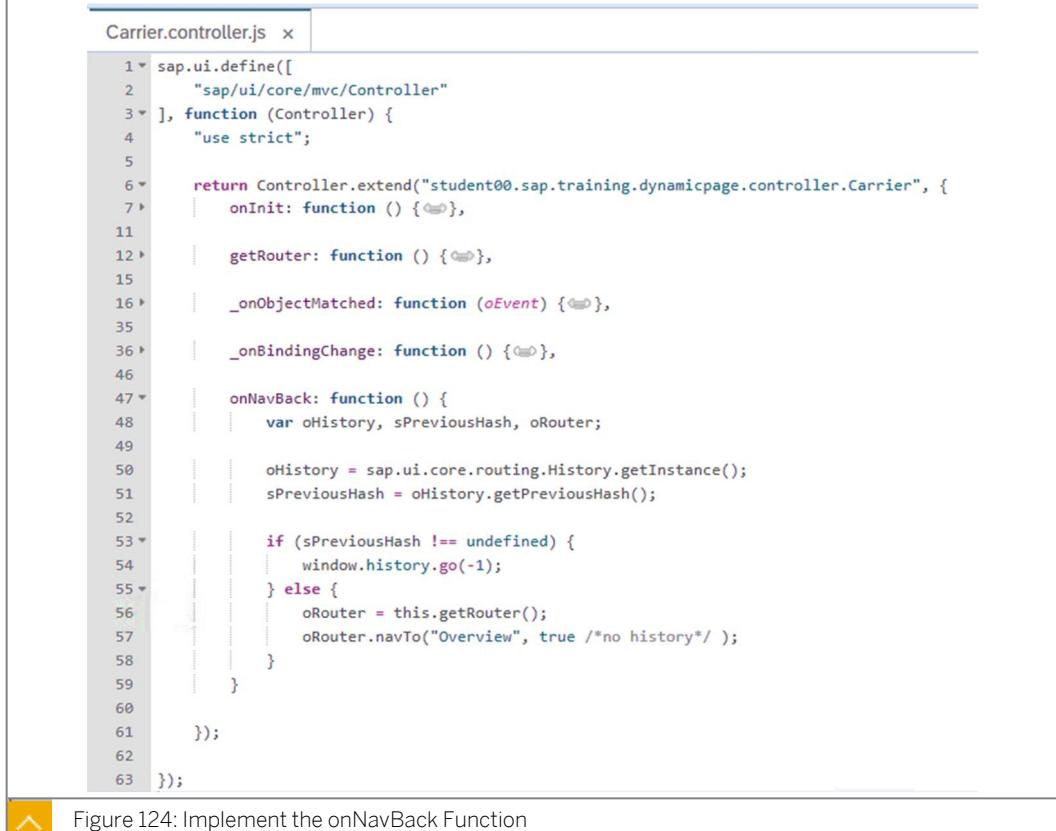
c) Implement the function `_onObjectMatched`.


```

Carrier.controller.js x
16   ...
17   ...
18   ...
19   ...
20   ...
21   ...
22   ...
23   ...
24   ...
25   ...
26   ...
27   ...
28   ...
29   ...
30   ...
31   ...
32   ...
33   ...
34   ...

```

Figure 123: Implement `_onObjectMatched` Functiond) Implement the `onNavBack` function.



```

Carrier.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Carrier", {
7     /**
8      * @function
9      * @name onInit
10     */
11    /**
12     * @function
13     * @name getRouter
14     */
15    /**
16     * @function
17     * @name _onObjectMatched
18     */
19    /**
20     * @function
21     * @name _onBindingChange
22     */
23    /**
24     * @function
25     * @name onNavBack
26     */
27     /**
28      * @function
29      * @name _onNavBack
30      * @param {sap.ui.core.routing.History} oHistory
31      * @param {string} sPreviousHash
32      * @param {sap.ui.core.routing.Router} oRouter
33      */
34     onNavBack: function () {
35       var oHistory, sPreviousHash, oRouter;
36
37       oHistory = sap.ui.core.routing.History.getInstance();
38       sPreviousHash = oHistory.getPreviousHash();
39
40       if (sPreviousHash !== undefined) {
41         window.history.go(-1);
42       } else {
43         oRouter = this.getRouter();
44         oRouter.navTo("Overview", true /*no history*/ );
45       }
46     }
47   });
48 });
49 });
50 });
51 });
52 });
53 });
54 });
55 });
56 });
57 });
58 });
59 });
60 });
61 });
62 });
63 });

```

Figure 124: Implement the onNavBack Function

## 10. Implement the *NotFound* controller.



```

NotFound.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.NotFound", {
7     /**
8      * @function
9      * @name getRouter
10     */
11    /**
12     * @function
13     * @name onNavBack
14     */
15     /**
16      * @function
17      * @name _onNavBack
18      * @param {sap.ui.core.routing.History} oHistory
19      * @param {string} sPreviousHash
20      * @param {sap.ui.core.routing.Router} oRouter
21      */
22   });
23 });
24 });

```

Figure 125: Implement the NotFound Controller

## 11. Implement the *controller* of the *Overview* view.



```

Overview.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
7     getRouter: function () {
8       return sap.ui.core.UIComponent.getRouterFor(this);
9     },
10
11     onPress: function (oEvent) {
12       var oItem, oCtx, sCarrid, oRouter;
13       oItem = oEvent.getSource();
14       oCtx = oItem.getBindingContext();
15       sCarrid = oCtx.getProperty("Carrid");
16
17       oRouter = this.getRouter();
18       oRouter.navTo("Carrier", {
19         carrierId: sCarrid
20       }, false /*with history*/ );
21     }
22   });
23 });

```

Figure 126: Implement the Controller

12. Implement the routing definition inside the `manifest.json`. Follow the routing and target configuration.

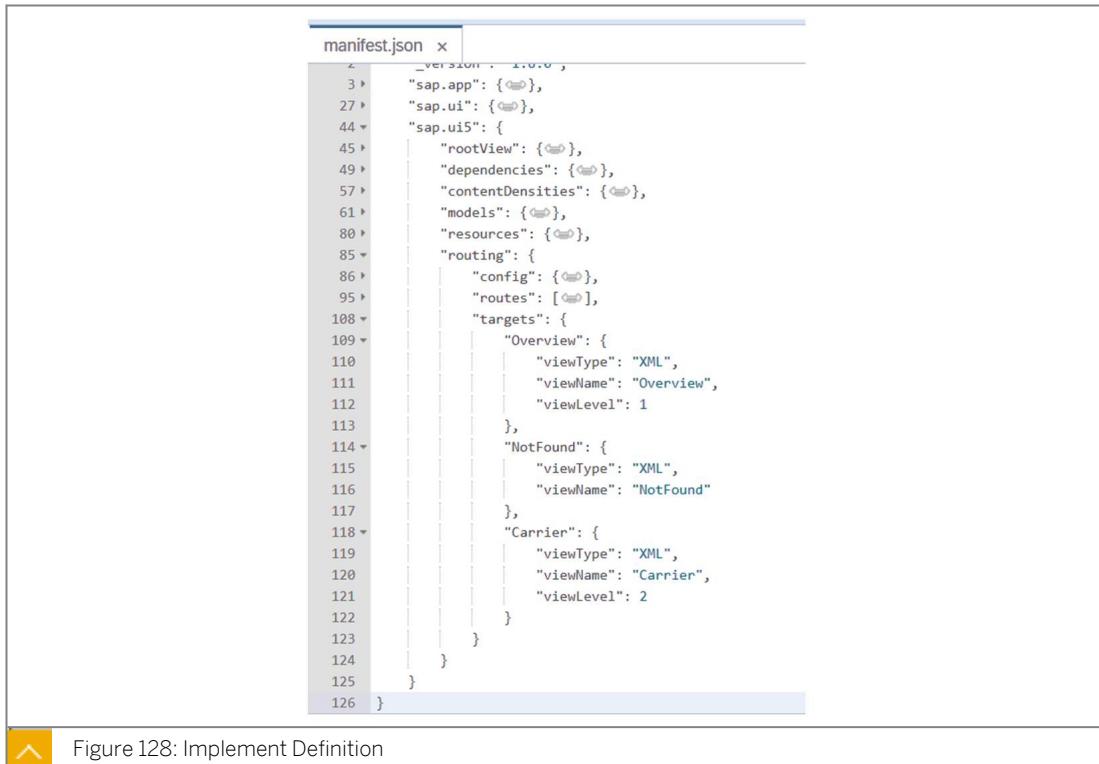


```

*manifest.json x
87 "routing": {
88   "config": {
89     "routerClass": "sap.m.routing.Router",
90     "viewType": "XML",
91     "async": true,
92     "viewPath": "student00.sap.training.dynamicpage.view",
93     "controlAggregation": "pages",
94     "controlId": "app",
95     "clearControlAggregation": false
96   }
97 }

```

Figure 127: Implement the Routing Definition



```

manifest.json x
  3 >   "version": "1.0.0",
  4 >   "sap.app": { },
  5 >   "sap.ui": { },
  6 >   "sap.ui5": {
  7 >     " rootView": { },
  8 >     "dependencies": { },
  9 >     "contentDensities": { },
 10 >     "models": { },
 11 >     "resources": { },
 12 >     "routing": {
 13 >       "config": { },
 14 >       "routes": [ ],
 15 >       "targets": {
 16 >         "Overview": {
 17 >           "viewType": "XML",
 18 >           "viewName": "Overview",
 19 >           "viewLevel": 1
 20 >         },
 21 >         "NotFound": {
 22 >           "viewType": "XML",
 23 >           "viewName": "NotFound"
 24 >         },
 25 >         "Carrier": {
 26 >           "viewType": "XML",
 27 >           "viewName": "Carrier",
 28 >           "viewLevel": 2
 29 >         }
 30 >       }
 31 >     }
 32 >   }
 33 > }
 34 > }
 35 > }
 36 > }
 37 > }
 38 > }
 39 > }
 40 > }
 41 > }
 42 > }
 43 > }
 44 > }
 45 > }
 46 > }
 47 > }
 48 > }
 49 > }
 50 > }
 51 > }
 52 > }
 53 > }
 54 > }
 55 > }
 56 > }
 57 > }
 58 > }
 59 > }
 60 > }
 61 > }
 62 > }
 63 > }
 64 > }
 65 > }
 66 > }
 67 > }
 68 > }
 69 > }
 70 > }
 71 > }
 72 > }
 73 > }
 74 > }
 75 > }
 76 > }
 77 > }
 78 > }
 79 > }
 80 > }
 81 > }
 82 > }
 83 > }
 84 > }
 85 > }
 86 > }
 87 > }
 88 > }
 89 > }
 90 > }
 91 > }
 92 > }
 93 > }
 94 > }
 95 > }
 96 > }
 97 > }
 98 > }
 99 > }
100 > }
101 > }
102 > }
103 > }
104 > }
105 > }
106 > }
107 > }
108 > }
109 > }
110 > }
111 > }
112 > }
113 > }
114 > }
115 > }
116 > }
117 > }
118 > }
119 > }
120 > }
121 > }
122 > }
123 > }
124 > }
125 > }
126 > }

```

Figure 128: Implement Definition

- Open the default route to `manifest.json` with Descriptor Editor and choose *Delete*.
- Go to the target definitions and remove the target with the name `TargetApp`.
- Select the item `Overview` from the list of available route names and press the plus button to create a default route for the target `Overview` and choose the option with the name `Overview` from the list of available targets.

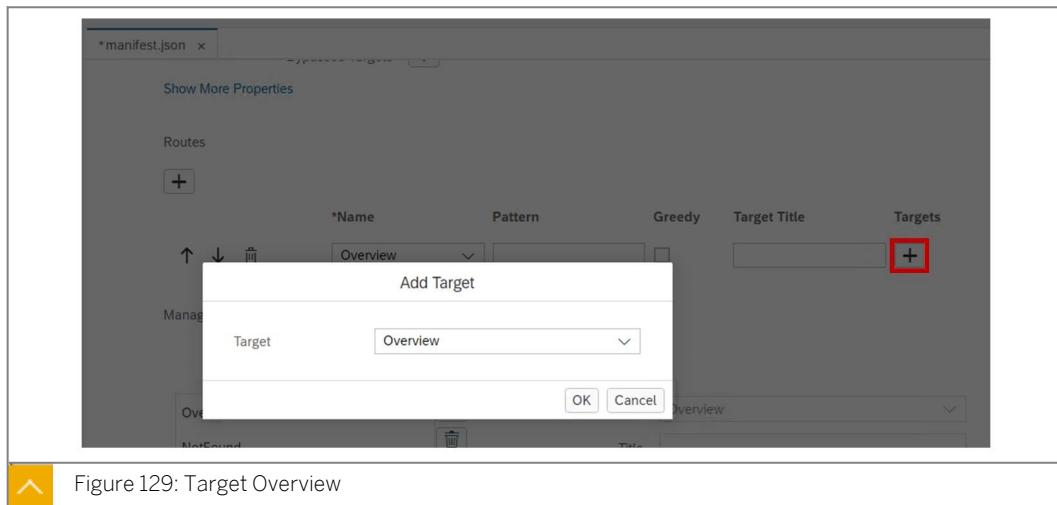
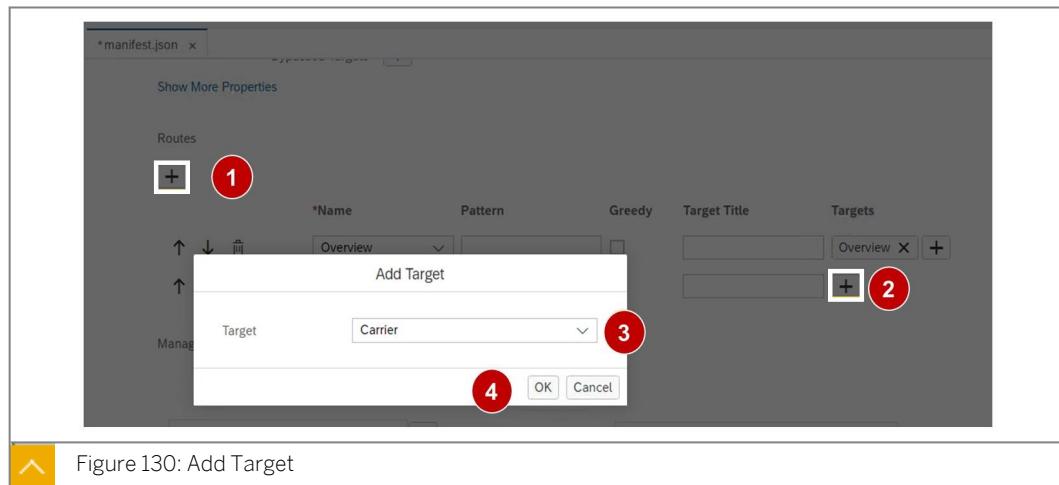
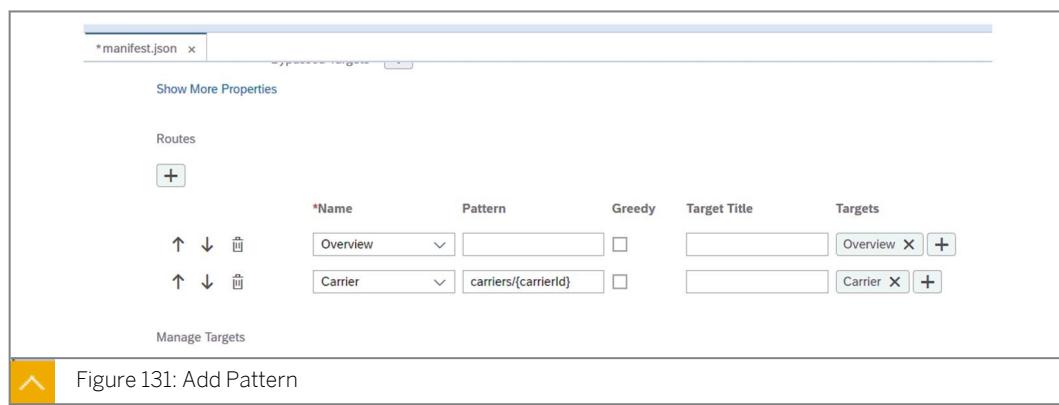


Figure 129: Target Overview

- To create a new route for the pattern carriers `{carrierId}`, select the plus icon on the level of the new route and add carrier as the target reference and choose *OK*.

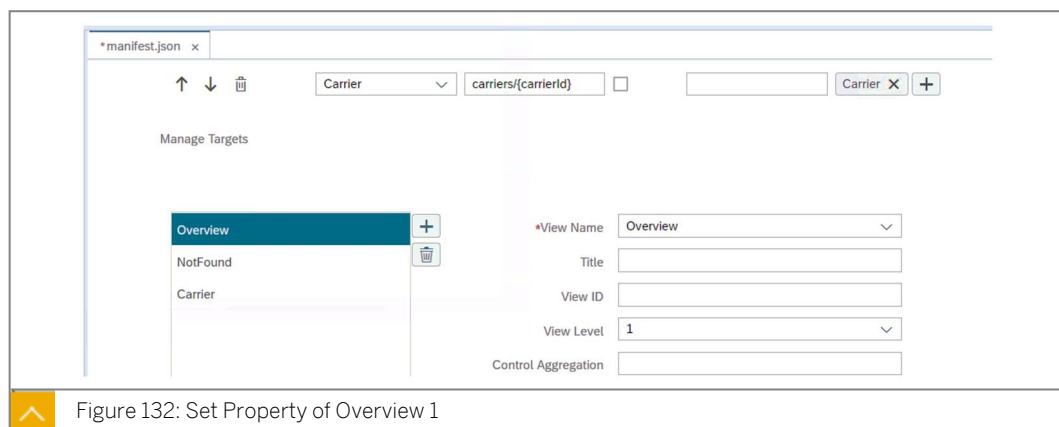


- e) Add the pattern for carriers or carrierId to the *pattern* field.

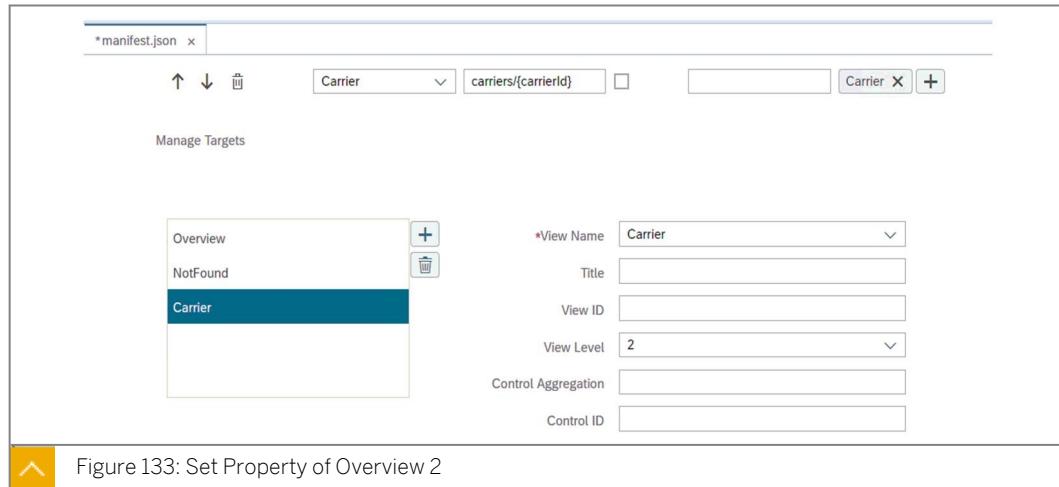


- f) Add the pattern carriers or carrierId.

- g) Set the property *viewLevel* of target *Overview* to 1.



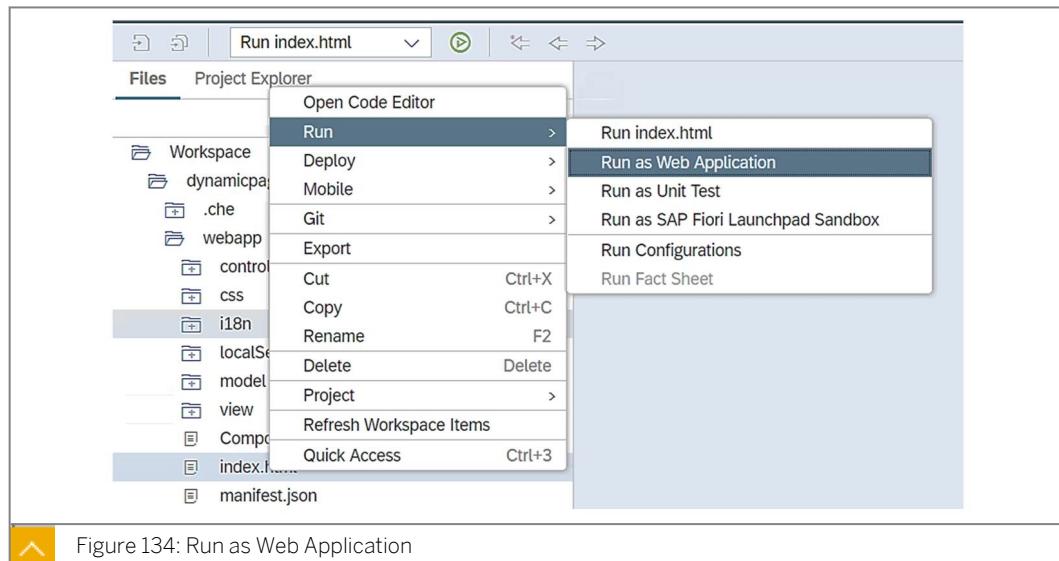
- h) Set the property *viewLevel* of target *Carrier* to 2.



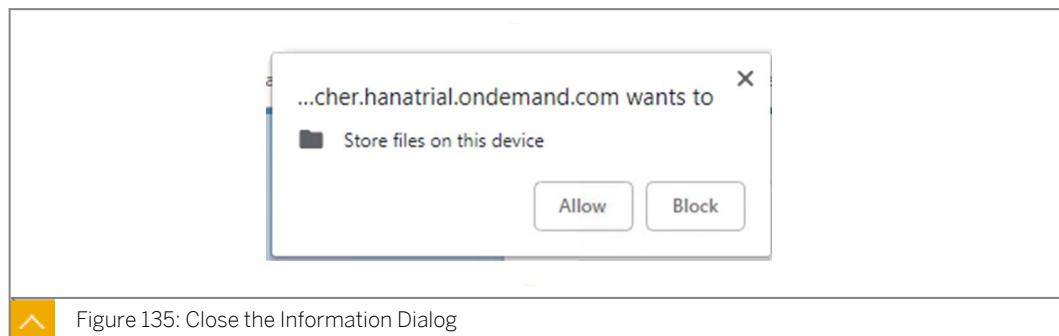
i) Save your changes.

13. Test your application.

- a) Select the file `index.html` in the `webapp` folder of your project and choose from the context menu `Run → Run as Web Application`.



- b) Close the information dialog if one displays.



- c) Sign in with your credentials for the front-end server if required.

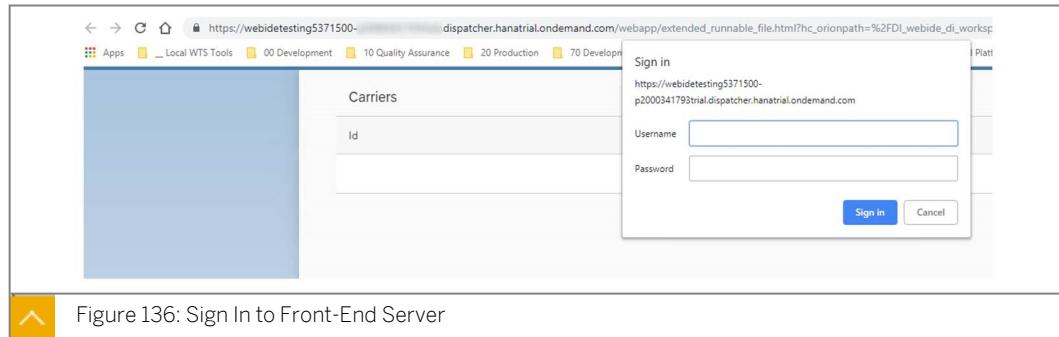


Figure 136: Sign In to Front-End Server

d) Choose a carrier from the list of carriers.

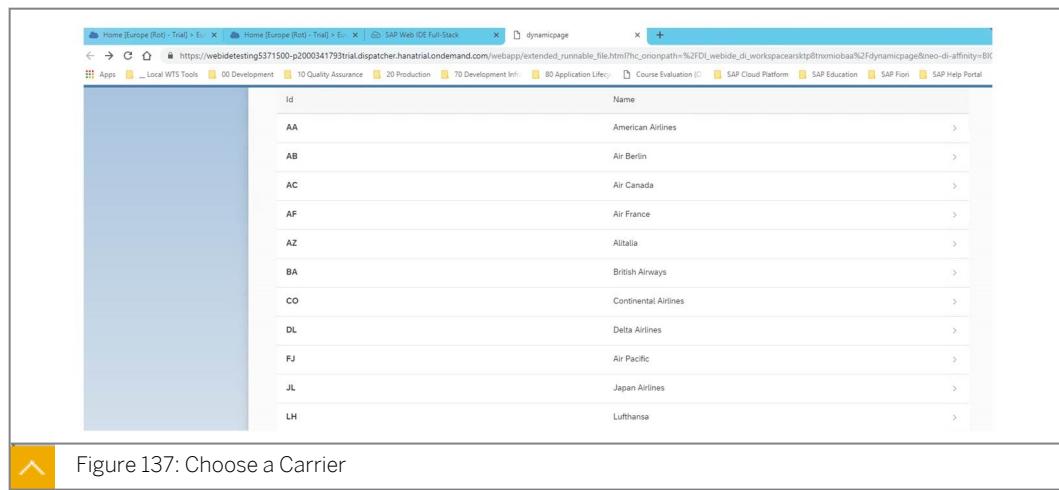
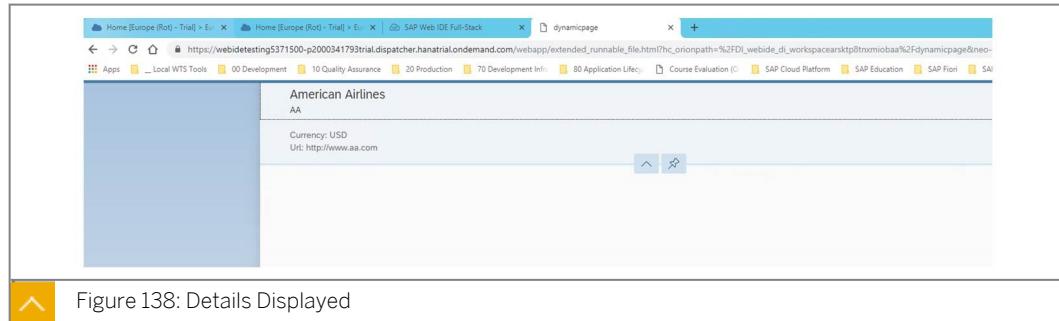


Figure 137: Choose a Carrier

e) The details of the selected carrier are displayed.



Use the back button on your browser to navigate to the carrier-overview page. In a real world scenario the application is embedded in the SAP Fiori launchpad. In the SAP Fiori launchpad the user is able to use the *Back* button in the launchpad.

# Create Master-Detail using the Flexible Column Layout



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and names such as *buttons* and *fields* may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business example

In this exercise you learn how to implement a master-detail application using the flexible column layout.

1. Log on to the SAP Web IDE Full-Stack and export the project *dynamicpage* from the previous exercise and import the project with the name *flexiblecolumn* into the workspace of your SAP Web IDE Full-Stack.
2. Import the content of the file *dynamicpage.zip* as a project with the name *flexiblecolumnlayout*.
3. Add a control of the type *sap.f.FlexibleColumnLayout* to the *App* control located in the *App.view.xml* file. Add an XML Namespace with alias *f* to your View control to reference the SAPUI5 namespace and assign the value *sap.f*. Add the listed attributes to the control.

Table 39: sap.f.FlexibleColumnLayout Attributes/Values

Attribute	Value
<i>id</i>	<i>idFlexLayout</i>
<i>layout</i>	{appView>/layout}
<i>backgroundDesign</i>	Translucent

4. Add the following implementation of the *onInit* function to the controller of the app view.

```

App.controller.js x
1 sap.ui.define([
2     "sap/ui/core/mvc/Controller",
3     "sap/ui/model/json/JSONModel"
4 ], function (Controller,JSONModel) {
5     "use strict";
6
7     return Controller.extend("student00.sap.training.dynamicpage.controller.App", {
8         ...
9         ...
10        ...
11        ...
12        ...
13        ...
14        ...
15        ...
16        ...
17        ...
18        ...
19        ...
20        ...
21        ...
22        ...
23    });
}

```

Figure 140: Add onInit

5. Update your routing configuration inside the `manifest.json` file. Make sure the attribute `clearControlAggregation` is set to `true`. Change the following attributes at the config-object.

Table 40: Routing Configuration

Attribute	Value
<code>controlAggregation</code>	<code>beginColumnPages</code>
<code>controlId</code>	<code>idFlexLayout</code>
<code>clearControlAggregation</code>	<code>true</code>

6. Add the *Overview* target to the carrier route and add the attribute `controlAggregation` to the *NotFound* target. Assign the value **midColumnPages** to the `controlAggregation` attribute. Add the attribute `controlAggregation` to the carrier target and assign the value **midColumnPages**.
7. Add and implement a file with the name `ListSelector.js` to the controller folder.



## Note:

Ensure you are using the correct namespace related to your implementation and group.



Figure 147: Implement ListSelectore.js

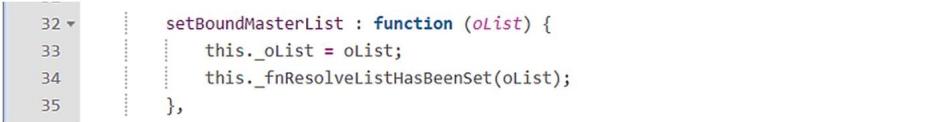


Figure 148: Implement ListSelectore.js

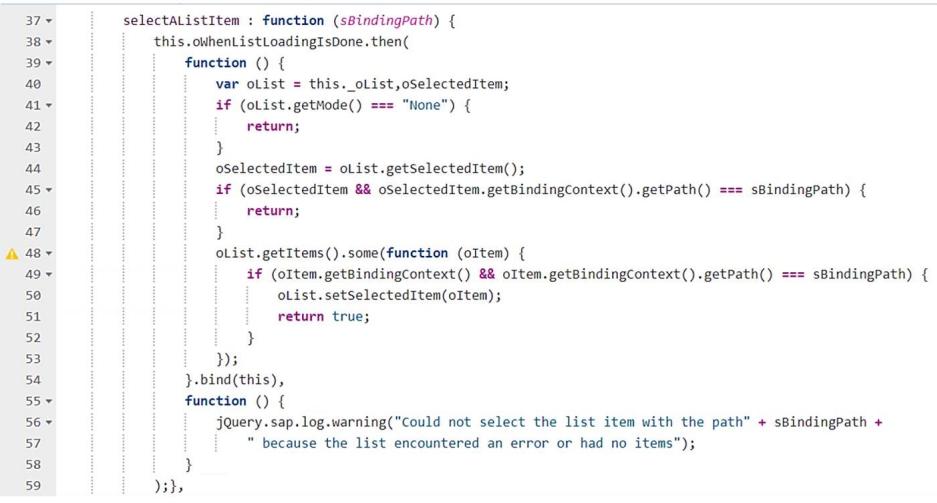


Figure 149: Implement ListSelectore.js

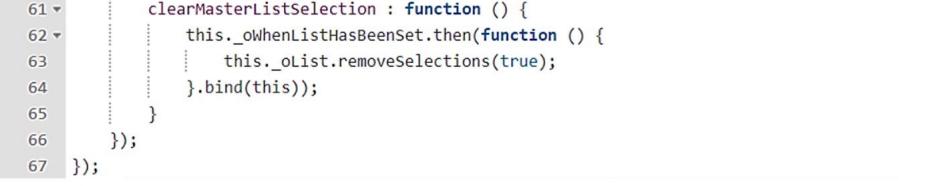


Figure 150: Implement ListSelectore.js

8. Add a reference to the ListSelector.js implementation in your Component.js file and assign an instance of type *ListSelector* to a member variable with the name *oListSelector*.
9. Use the *sap.f.semantic.SemanticPage* in the *Carrier.view.xml* instead of the *sap.f.DynamicPage* control. Show the same data as before and leave the overall layout the same. Replace SAPUI5-namespace *sap.f.semantic* with XML-namespace alias *semantic*. Use the *title* aggregation to display the title *sap.m.Title* control and the *headerContent* aggregation to display the *FlexBox* implementation.
10. Add an *sap.f.semantic.CloseAction* to the *SemanticPage* implementation of *Carrier.view.xml*. Use the following attribute and values.

Table 41: sap.f.semantic.CloseAction Attributes and Values

Attributes	Value
id	idCloseAction
press	onCloseDetailPress

11. Add an sap.f.semantic.FullScreenAction to the SemanticPage implementation of `Carrier.view.xml`. Use the following attributes and values:

Table 42: sap.f.semantic.FullScreenAction Attributes and Values

Attribute	Value
id	idEnterFullScreen
visible	{= !\${device>/system/phone} && !\${appView>/actionButton-Info/midColumn/fullScreen}}
press	onToggleFullScreen

12. Add `sap.f.semantic.ExitFullScreenAction` to the `SemanticPage` implementation of `Carrier.view.xml`.

Table 43: sap.f.semantic.ExitFullScreenAction Attributes and Values

Attribute	Value
id	idExitFullScreen
visible	{= !\${device>/system/phone} && \${appView>/actionButton-Info/midColumn/fullScreen}}
press	onToggleFullScreen

13. Add the listed attributes to the `sap.m.Table` control in the file `Overview.view.xml`.

Table 44: sap.m.Table Action Attributes and Values

Attribute	Value
mode	{= \${device>/system/phone} ? 'None' : 'SingleSelectMaster'}
selectionChange	onSelectionChange

14. Change the implementation of the Overview controller.  
 15. Implement the function `_showCarrierDetails`.

```
    _showCarrierDetails : function(sCarrid) {
        var oRouter = this.getRouter();
        oRouter.navTo("Carrier", {
            carrierId: sCarrid
        }, false /*with history*/ );
    }
}
```

Figure 166: Implement showCarrierDetails

16. Change the implementation of the Carrier controller. Add the following code to the Carrier controller implementation.

Figure 167: Implementation

```
16 ▾ ..... _onObjectMatched: function (oEvent) {
17 .....     var oArgs, oView;
18 .....     oArgs = oEvent.getParameter("arguments");
19 .....     this._sCarrierId = oArgs.carrierId;
20 .....     oView = this.getView();
21 ..... 
22 ▾ .....     oView.bindElement({
23 .....         path: "/ZBC_C_Carrier_TP('" + this._sCarrierId + "')",
24 .....         events: {
25 .....             change: this._onBindingChange.bind(this),
26 .....             dataRequested: function () {
27 .....                 oView.setBusy(true);
28 .....             },
29 .....             dataReceived: function () {
30 .....                 oView.setBusy(false);
31 .....             }
32 .....         }
33 .....     });
34 .....     this.getOwnerComponent().oListSelector.selectListItem("/ZBC_C_Carrier_TP('" + this._sCarrierId + "')");
35 .....     var oViewModel = this.getOwnerComponent().getModel("appview");
36 .....     oViewModel.setProperty("/layout", "TwoColumnsMidExpanded");
37 .....     this.getOwnerComponent().setModel(oViewModel, "appView");
38 ..... }
```

Figure 168: Implementation

```
40 ▼ ..... _onBindingChange: function () {
41     var oView = this.getView();
42     var oElementBinding = oView.getElementBinding();
43     // No data for the binding
44     if (oElementBinding && !oElementBinding.getBoundContext()) {
45         this.getOwnerComponent().oListSelector.clearMasterListSelection();
46         this.getRouter().getTargets().display("NotFound");
47         return;
48     }
49     var sPath = oElementBinding.getPath();
50     this.getOwnerComponent().oListSelector.selectAListItem(sPath);
51 }
```

Figure 169: Implementation

```
53 ▼
54 ▼
55
56
57
58
59
60 ▼
61 ▼
62
63 ▼
64
65 ▼
66
67 ▼
68
69
70 ▼
71
72 ▼
73
74
75
onCloseDetailPress: function () {
  this.getOwnerComponent().getModel("appView").setProperty(
    "actionButtonsInfo/midColumn/fullScreen", false);
  this.getOwnerComponent().olistSelector.clearMasterListSelection();
  this.getRouter().navTo("overview");
},
onToggleFullScreen: function () {
  var bFullScreen = this.getOwnerComponent().getModel("appView").getProperty(
    "actionButtonsInfo/midColumn/fullScreen");
  this.getOwnerComponent().getModel("appView").setProperty(
    "actionButtonsInfo/midColumn/fullScreen", !bFullScreen);
  if (!bFullScreen) {
    // store current layout and go full screen
    this.getOwnerComponent().getModel("appView").setProperty("/previousLayout",
      this.getOwnerComponent().getModel("appView").getProperty("/layout"));
    this.getOwnerComponent().getModel("appView").setProperty("/layout", "MidColumnFullScreen");
  } else {
    // reset to previous layout
    this.getOwnerComponent().getModel("appView").setProperty("/layout",
      this.getOwnerComponent().getModel("appView").getProperty("/previousLayout"));
  }
},
```

Figure 170: Implementation

Figure 171: Implementation

## 17. Test your application.

# Create Master-Detail using the Flexible Column Layout



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and names such as *buttons* and *fields* may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business example

In this exercise you learn how to implement a master-detail application using the flexible column layout.

1. Log on to the SAP Web IDE Full-Stack and export the project *dynamicpage* from the previous exercise and import the project with the name *flexiblecolumn* into the workspace of your SAP Web IDE Full-Stack.
  - a) Choose the project *dynamicpage* from the workspace and select *Export* from the context menu.
  - b) Download and open the *dynamicpage.zip* file.
2. Import the content of the file *dynamicpage.zip* as a project with the name *flexiblecolumnlayout*.
  - a) Select the workspace folder and from the context menu, choose *Import → File or Project*.
  - b) Browse for the exported file and change the file name in the Import field to */flexiblecolumnlayout* and confirm with *OK*.
  - c) On successful import you see the newly imported project with the given name.
3. Add a control of the type *sap.f.FlexibleColumnLayout* to the *App* control located in the *App.view.xml* file. Add an *XML Namespace* with *alias f* to your *View* control to reference the SAPUI5 namespace and assign the value *sap.f*. Add the listed attributes to the control.

Table 39: *sap.f.FlexibleColumnLayout* Attributes/Values

Attribute	Value
<i>id</i>	<i>idFlexLayout</i>
<i>layout</i>	<i>{appView&gt;/layout}</i>

Attribute	Value
backgroundDesign	Translucent

- Open the file `App.view.xml` located in the view folder of your project.
- Add the XML namespace alias `f` to the view control and assign SAPUI5 namespace `sap.f`.
- Add a control of type `sap.f.FlexibleColumnLayout` to the `App` control and add the attributes and values listed in the table to the control.



```

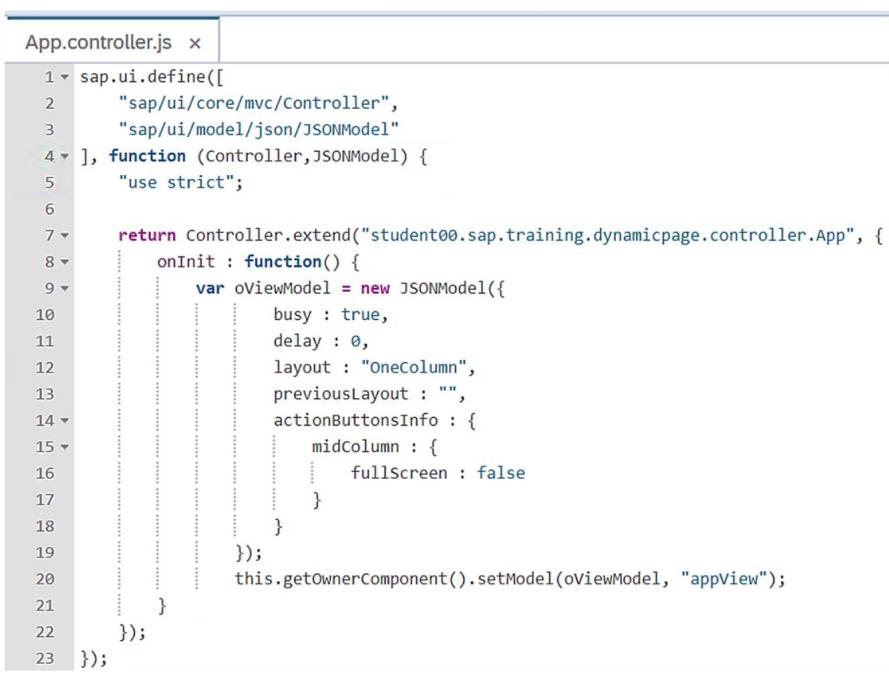
*App.view.xml x
1 <mvc:View controllerName="student00.sap.training.dynamicpage.controller.App" xmlns:mvc="sap.ui.core.mvc"
2   displayBlock="true" xmlns:sap.m" xmlns:f="sap.f">
3 <Shell id="shell">
4   <App id="app">
5     <f:FlexibleColumnLayout id="idFlexLayout" layout="{appView>/layout}" backgroundDesign="Translucent"/>
6   </App>
7 </Shell>
8 </mvc:View>

```

Figure 139: Add a Control

- Save your changes.

#### 4. Add the following implementation of the `onInit` function to the controller of the app view.



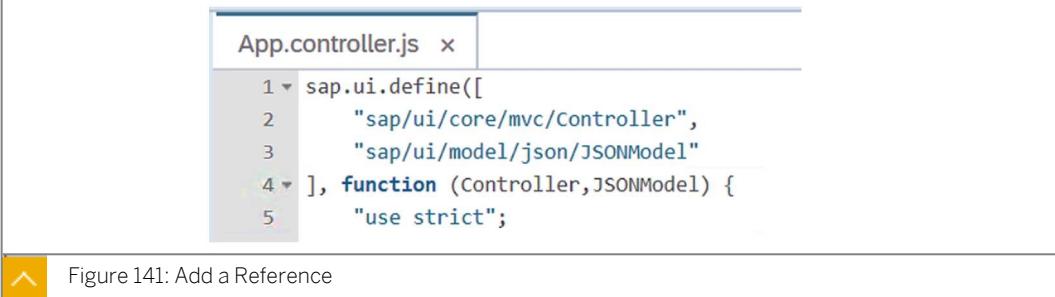
```

App.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/json/JSONModel"
4 ], function (Controller,JSONModel) {
5   "use strict";
6
7   return Controller.extend("student00.sap.training.dynamicpage.controller.App", {
8     onInit : function() {
9       var oViewModel = new JSONModel({
10         busy : true,
11         delay : 0,
12         layout : "OneColumn",
13         previousLayout : "",
14         actionButtonsInfo : {
15           midColumn : {
16             ...fullScreen : false
17           }
18         }
19       });
20       this.getOwnerComponent().setModel(oViewModel, "appView");
21     }
22   });
23 });

```

Figure 140: Add onInit

- Open the file `App.controller.js`. The file is in the controller folder of your project.
- Add a reference to the `sap.ui.model.json.JSONModel` implementation.



```

App.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/json/JSONModel"
4 ], function (Controller, JSONModel) {
5   "use strict";

```

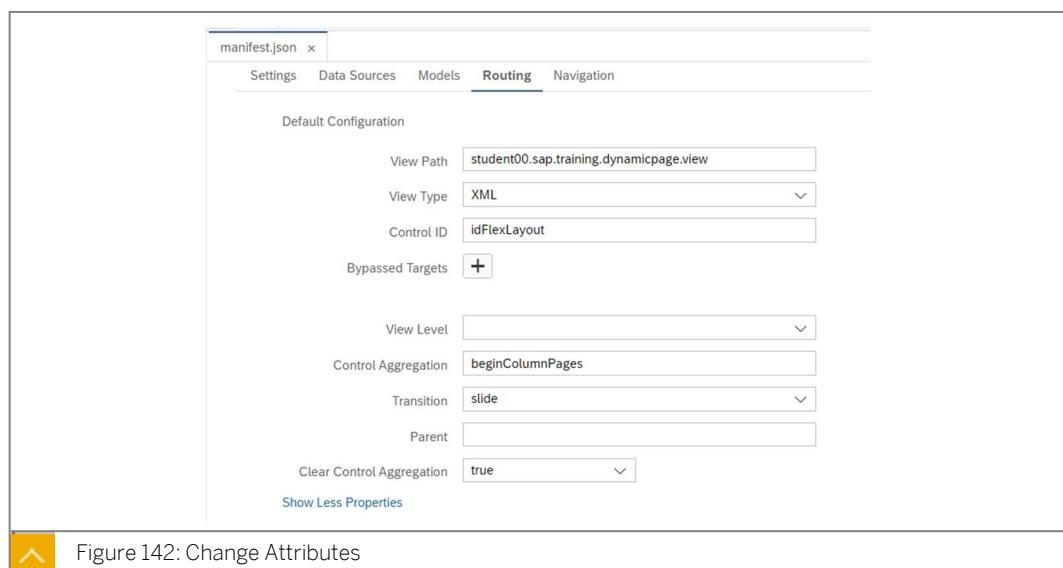
Figure 141: Add a Reference

- c) Implement the `onInit` function.
5. Update your routing configuration inside the `manifest.json` file. Make sure the attribute `clearControlAggregation` is set to `true`. Change the following attributes at the config-object.

Table 40: Routing Configuration

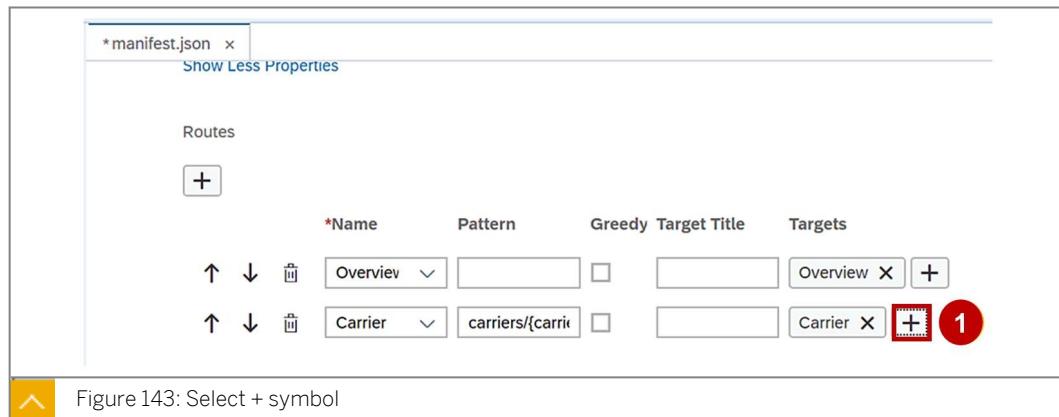
Attribute	Value
<code>controlAggregation</code>	<code>beginColumnPages</code>
<code>controlId</code>	<code>idFlexLayout</code>
<code>clearControlAggregation</code>	<code>true</code>

- a) Open the `manifest.json` file located directly under the `webapp` folder.
- b) Switch to the Routing tab of the Descriptor Editor.
- c) Select the link *Show More Properties*.
- d) Change the attributes to the new attributes.

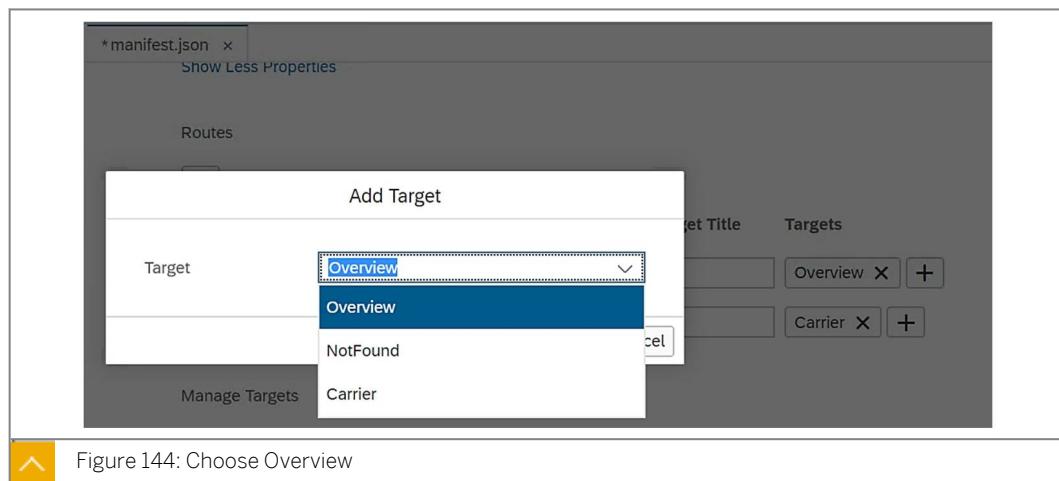


6. Add the *Overview target* to the carrier route and add the attribute `controlAggregation` to the `NotFound` target. Assign the value `midColumnPages` to the `controlAggregation` attribute. Add the attribute `controlAggregation` to the carrier target and assign the value `midColumnPages`.

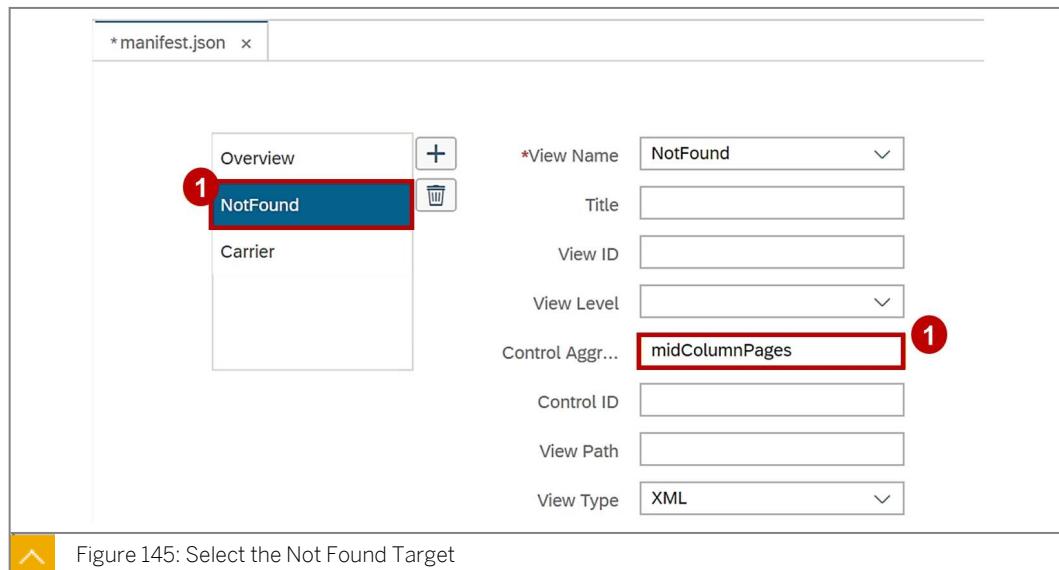
- a) Select the plus symbol at the level of the carrier route to add the overview target.



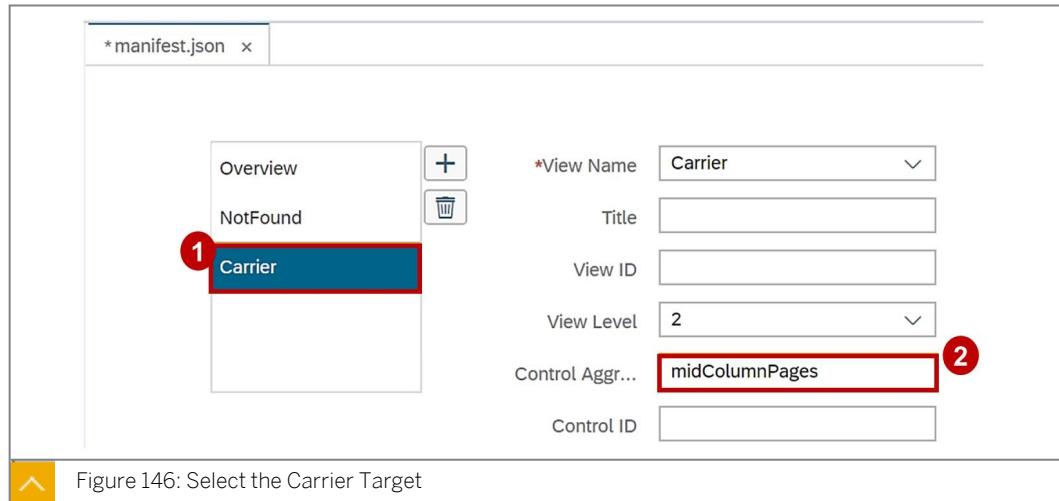
- b) Choose the entry *Overview* from the list of available targets and confirm with *OK*.



- c) Select the *NotFound* target and add the value **midColumnPages** to the *ControlAggregation* attribute.

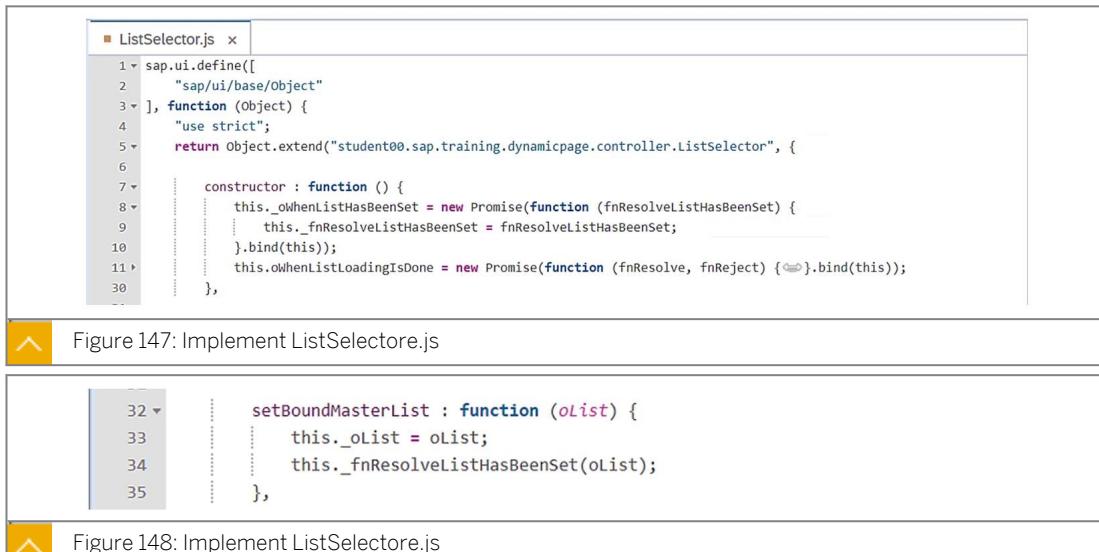
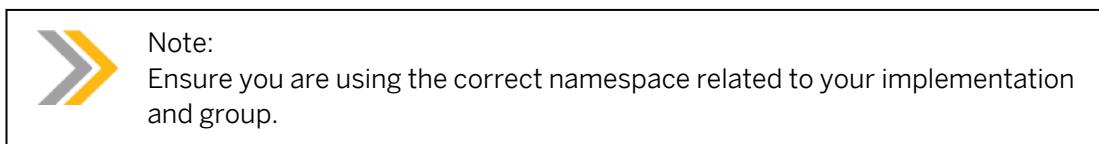


- d) Select the carrier target and add the value *midColumnPages* to the *ControlAggregation* attribute.



e) Save your changes.

7. Add and implement a file with the name ListSelector.js to the controller folder.





```

37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

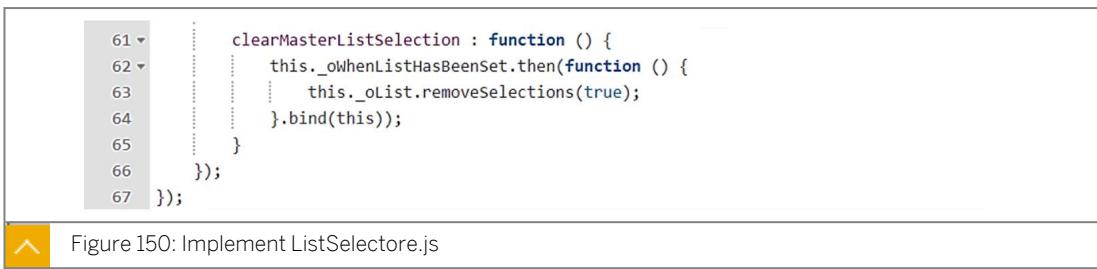
```

```

    selectAListItem : function (sBindingPath) {
        this._oWhenListLoadingIsDone.then(
            function () {
                var oList = this._oList, oSelectedItem;
                if (oList.getMode() === "None") {
                    return;
                }
                oSelectedItem = oList.getSelectedItem();
                if (oSelectedItem && oSelectedItem.getBindingContext().getPath() === sBindingPath) {
                    return;
                }
                oList.getItems().some(function (oItem) {
                    if (oItem.getBindingContext() && oItem.getBindingContext().getPath() === sBindingPath) {
                        oList.setSelectedItem(oItem);
                        return true;
                    }
                });
            }.bind(this),
            function () {
                jQuery.sap.log.warning("Could not select the list item with the path " + sBindingPath +
                    " because the list encountered an error or had no items");
            }
        );
    },

```

Figure 149: Implement ListSelector.js



```

61
62
63
64
65
66
67

```

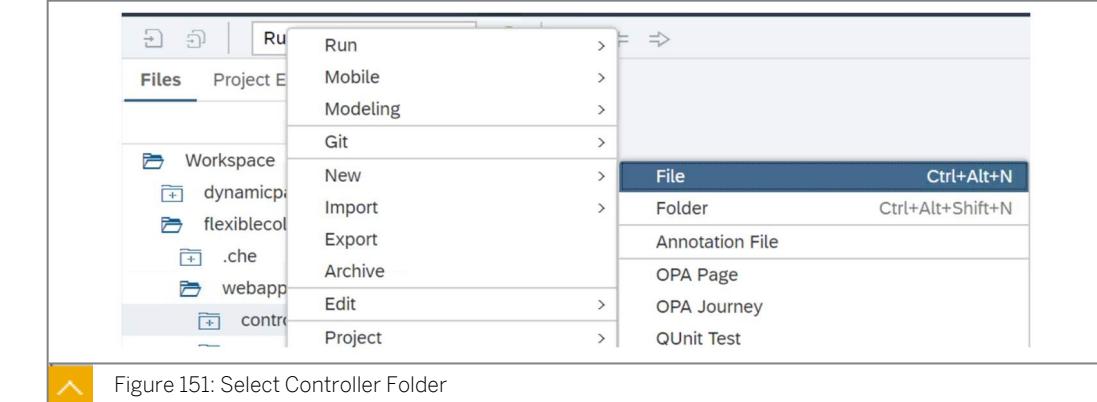
```

    clearMasterListSelection : function () {
        this._oWhenListHasBeenSet.then(function () {
            this._oList.removeSelections(true);
        }.bind(this));
    });
}

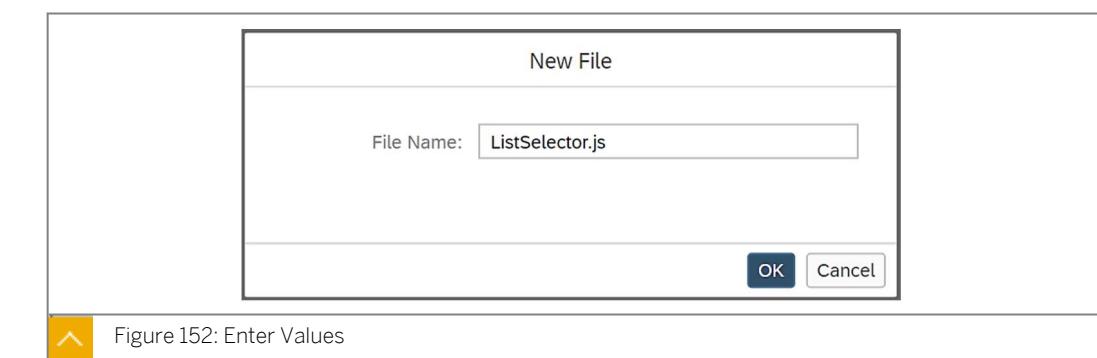
```

Figure 150: Implement ListSelector.js

- a) From the context menu, choose *New* → *File* and then select the controller folder of your project.



- b) Add the value **ListSelector.js** in the file name field and confirm **OK**.



- c) Implement the file and save your changes.

8. Add a reference to the ListSelector.js implementation in your `Component.js` file and assign an instance of type `ListSelector` to a member variable with the name `oListSelector`.
- Open the file `Component.js` located in the `webapp` folder of your project.
  - Add a reference to the `ListSelector` implementation.



The screenshot shows a code editor with a file named `Component.js`. The code defines a component using the `sap.ui.define` function. The first few lines of code are:

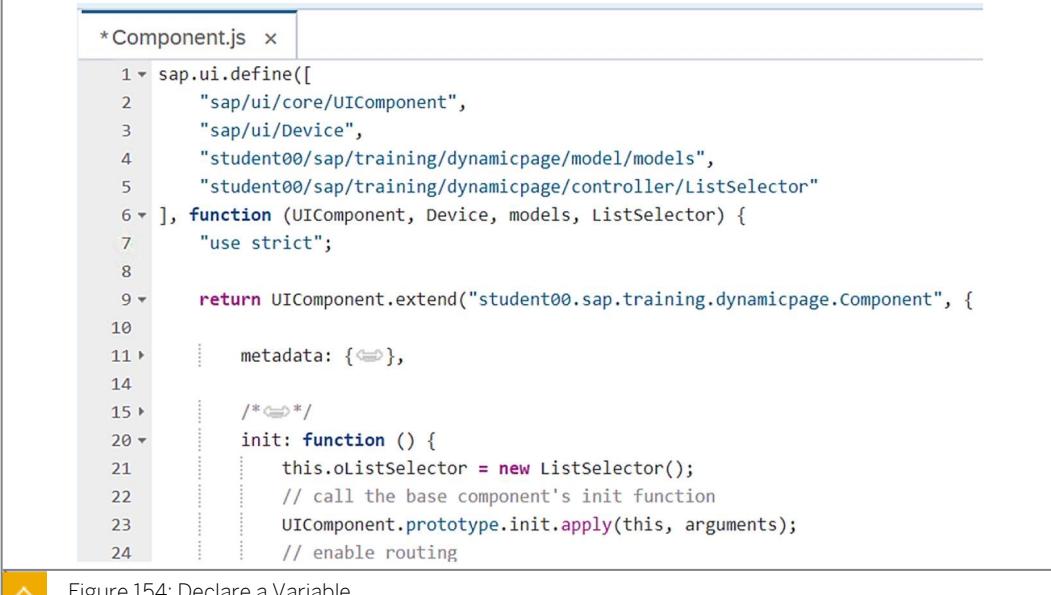
```

1 sap.ui.define([
2   "sap/ui/core/UIComponent",
3   "sap/ui/Device",
4   "student00/sap/training/dynamicpage/model/models",
5   "student00/sap/training/dynamicpage/controller/ListSelector"
6 ], function (UIComponent, Device, models, ListSelector) {
7   "use strict";

```

Figure 153: Add a Reference

- Declare a member variable with the name `oListSelector` and assign a new instance of `ListSelector`.



The screenshot shows the continuation of the `Component.js` code. It includes the declaration of a member variable `oListSelector` and its assignment to a new `ListSelector` instance. The code is as follows:

```

1 sap.ui.define([
2   "sap/ui/core/UIComponent",
3   "sap/ui/Device",
4   "student00/sap/training/dynamicpage/model/models",
5   "student00/sap/training/dynamicpage/controller/ListSelector"
6 ], function (UIComponent, Device, models, ListSelector) {
7   "use strict";
8
9   return UIComponent.extend("student00.sap.training.dynamicpage.Component", {
10
11     metadata: { },
12
13
14
15     init: function () {
16       this.oListSelector = new ListSelector();
17       // call the base component's init function
18       UIComponent.prototype.init.apply(this, arguments);
19       // enable routing

```

Figure 154: Declare a Variable

- Save your changes.

9. Use the `sap.f.semantic.SemanticPage` in the `Carrier.view.xml` instead of the `sap.f.DynamicPage` control. Show the same data as before and leave the overall layout the same. Replace SAPUI5-namespace `sap.f.semantic` with XML-namespace alias `semantic`. Use the `title` aggregation to display the title `sap.m.Title` control and the `headerContent` aggregation to display the `FlexBox` implementation.

- Open the file `Carrier.view.xml` located in the `view` folder of your project.
- Replace SAPUI5-namespace `sap.f.semantic` with XML-namespace alias `semantic`.



Carrier.view.xml

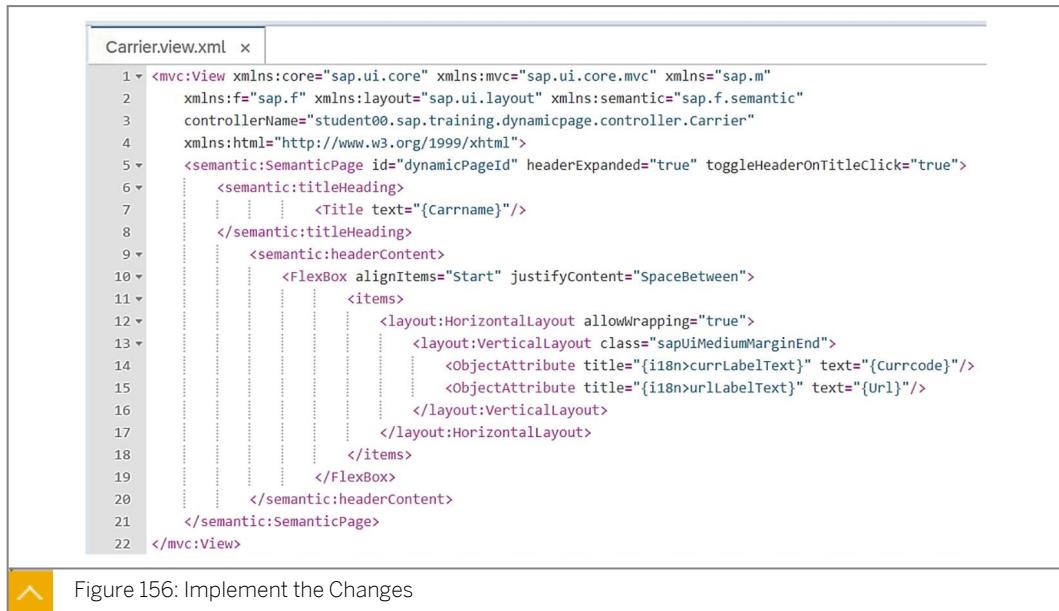
```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:layout="sap.ui.layout" xmlns:semantic="sap.f.semantic"
3   controllerName="student00.sap.training.dynamicpage.controller.Carrier"
4   xmlns:html="http://www.w3.org/1999/xhtml">
5   <f:DynamicPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
6     <f:title>
7       <f:DynamicPageTitle>

```

Figure 155: Replace Namespace

c) Implement the changes.



Carrier.view.xml

```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:layout="sap.ui.layout" xmlns:semantic="sap.f.semantic"
3   controllerName="student00.sap.training.dynamicpage.controller.Carrier"
4   xmlns:html="http://www.w3.org/1999/xhtml">
5   <semantic:SemanticPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
6     <semantic:titleHeading>
7       <Title text="{Carrname}" />
8     </semantic:titleHeading>
9     <semantic:headerContent>
10    <FlexBox alignItems="Start" justifyContent="SpaceBetween">
11      <items>
12        <layout:HorizontalLayout allowWrapping="true">
13          <layout:VerticalLayout class="sapUiMediumMarginEnd">
14            <ObjectAttribute title="{i18n>currLabelText}" text="{CurrCode}" />
15            <ObjectAttribute title="{i18n>urlLabelText}" text="{Url}" />
16          </layout:VerticalLayout>
17        </layout:HorizontalLayout>
18      </items>
19    </FlexBox>
20  </semantic:headerContent>
21 </semantic:SemanticPage>
22 </mvc:View>

```

Figure 156: Implement the Changes

d) Save the changes.

10. Add an `sap.f.semantic.CloseAction` to the `SemanticPage` implementation of `Carrier.view.xml`. Use the following attribute and values.

Table 41: `sap.f.semantic.CloseAction` Attributes and Values

Attributes	Value
<code>id</code>	<code>idCloseAction</code>
<code>press</code>	<code>onCloseDetailPress</code>

- a) Open the file `Carrier.view.xml`.  
 b) Add the `closeAction` aggregation to the `sap.f.semantic.SemanticPage` control.  
 c) Add an `sap.f.semantic.CloseAction` control with the listed attribute and values.



The screenshot shows the SAP Fiori XML editor with the file `*Carrier.view.xml` open. The code is as follows:

```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:layout="sap.ui.layout" xmlns:semantic="sap.f.semantic"
3   controllerName="student00.sap.training.dynamicpage.controller.Carrier"
4   xmlns:html="http://www.w3.org/1999/xhtml">
5   <semantic:SemanticPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleclick="true">
6     <semantic:titleHeading>
7       <title text="{Carrname}" />
8     </semantic:titleHeading>
9     <semantic:closeAction>
10    <semantic:CloseAction
11      id="idCloseAction"
12      press="onCloseDetailPress" />
13  </semantic:closeAction>

```

Figure 157: Add a Control

11. Add an `sap.f.semantic.FullScreenAction` to the `SemanticPage` implementation of `Carrier.view.xml`. Use the following attributes and values:

Table 42: `sap.f.semantic.FullScreenAction` Attributes and Values

Attribute	Value
<code>id</code>	<code>idEnterFullScreen</code>
<code>visible</code>	<code>{= !\${device}/system/phone} &amp;&amp; !\${appView}/actionButtonsInfo/midColumn/fullScreen}}</code>
<code>press</code>	<code>onToggleFullScreen</code>

- a) Add the `fullScreenAction` aggregation to the `SemanticPage` control.
- b) Add a control of type `sap.f.semantic.FullScreenAction` and add the listed attribute and values.



The screenshot shows the SAP Fiori XML editor with the file `*Carrier.view.xml` open. The code is as follows:

```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:layout="sap.ui.layout" xmlns:semantic="sap.f.semantic"
3   controllerName="student00.sap.training.dynamicpage.controller.Carrier"
4   xmlns:html="http://www.w3.org/1999/xhtml">
5   <semantic:SemanticPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleclick="true">
6     <semantic:titleHeading>
7       <title text="{Carrname}" />
8     </semantic:titleHeading>
9     <semantic:closeAction>
10    <semantic:CloseAction
11      id="idCloseAction"
12      press="onCloseDetailPress" />
13  </semantic:closeAction>
14  <semantic:fullScreenAction>
15    <semantic:FullScreenAction
16      id="idEnterFullScreen"
17      visible="{= !${device}/system/phone} && !${appView}/actionButtonsInfo/midColumn/fullScreen}"
18      press="onToggleFullScreen" />
19  </semantic:fullScreenAction>

```

Figure 158: Add a Control

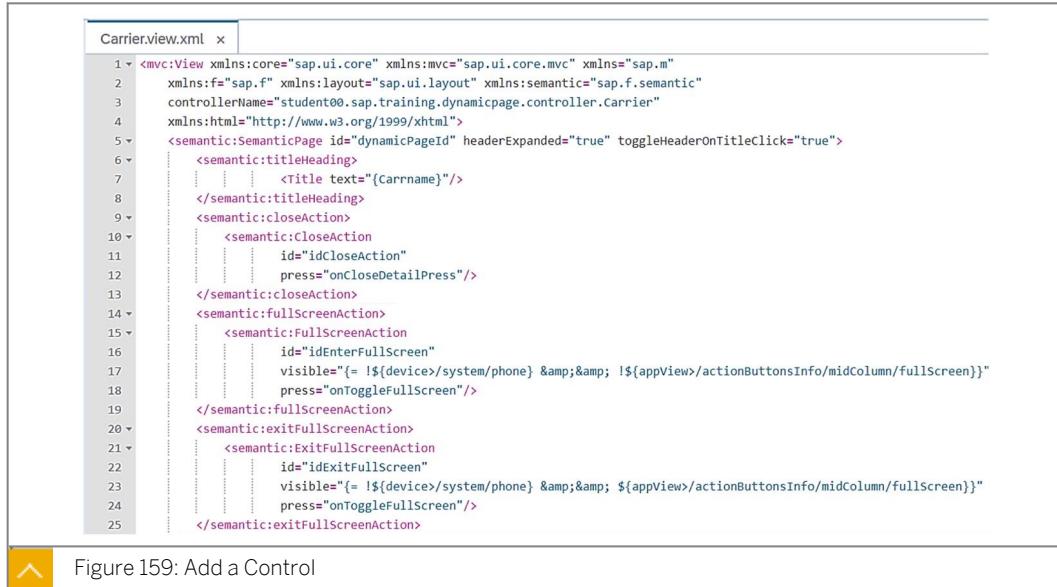
12. Add `sap.f.semantic.ExitFullScreenAction` to the `SemanticPage` implementation of `Carrier.view.xml`.

Table 43: `sap.f.semantic.ExitFullScreenAction` Attributes and Values

Attribute	Value
<code>id</code>	<code>idExitFullScreen</code>

Attribute	Value
visible	<code>{= !\${device}&gt;/system/phone} &amp;&amp; \${appView}&gt;/actionButtonsInfo/midColumn/fullScreen}}</code>
press	onToggleFullScreen

- Add the `exitFullScreenAction` aggregation to the `SemanticPage` control.
- Add a control of type `sap.f.semantic.ExitFullScreenAction` to the aggregation and add the listed attribute and values to the control.



```

Carrier.view.xml x
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:layout="sap.ui.layout" xmlns:semantic="sap.f.semantic"
3   controllerName="student00.sap.training.dynamicpage.controller.Carrier"
4   xmlns:html="http://www.w3.org/1999/xhtml">
5   <semantic:SemanticPage id="dynamicPageId" headerExpanded="true" toggleHeaderOnTitleClick="true">
6     <semantic:titleHeading>
7       <Title text="{Carname}" />
8     </semantic:titleHeading>
9     <semantic:closeAction>
10    <semantic:CloseAction
11      id="idCloseAction"
12      press="onCloseDetailPress" />
13  </semantic:closeAction>
14  <semantic:fullscreenAction>
15    <semantic:FullScreenAction
16      id="idEnterFullScreen"
17      visible="{= !${device}>/system/phone} && ${appView}>/actionButtonsInfo/midColumn/fullScreen}"
18      press="onToggleFullScreen" />
19  </semantic:fullscreenAction>
20  <semantic:exitfullscreenAction>
21    <semantic:ExitFullScreenAction
22      id="idExitFullScreen"
23      visible="{= !${device}>/system/phone} && ${appView}>/actionButtonsInfo/midColumn/fullScreen}"
24      press="onToggleFullScreen" />
25  </semantic:exitfullscreenAction>

```

Figure 159: Add a Control

- Save your changes.

13. Add the listed attributes to the `sap.m.Table` control in the file `Overview.view.xml`.

Table 44: sap.m.Table Action Attributes and Values

Attribute	Value
mode	<code>{= \${device}&gt;/system/phone} ? 'None' : 'SingleSelectMaster'</code>
selectionChange	onSelectionChange

- Open the file `Overview.view.xml` located in the view folder of your project.
- Add the listed attribute and value pairs to the Table control.



Overview.view.xml

```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2   controllerName="student00.sap.training.dynamicpage.controller.Overview"
3   xmlns:html="http://www.w3.org/1999/xhtml">
4   <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
5     <f:content>
6       <Table items="/ZBC_C_Carrier_TP">
7         mode="${device}/system/phone} ? 'None' : 'SingleSelectMaster'" mode="None"
8         selectionChange="onSelectionChange">
9           <headerToolbar>

```

Figure 160: Add Listed Attributes

c) Save your changes.

#### 14. Change the implementation of the Overview controller.

a) Adapt the implementation of the `onInit` function.



Overview.controller.js

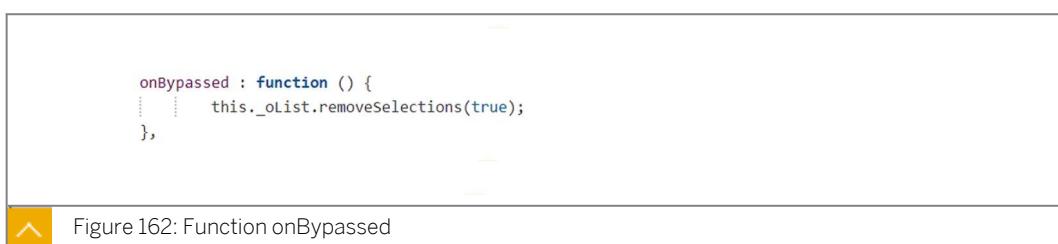
```

1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
7     ...
8     ...
9     ...
10    ...
11    ...
12    ...
13    ...
14    ...
15    ...
16    ...
17    ...
18    ...
19    ...
20    ...
21    ...
22    ...

```

Figure 161: Adapt Implementation

b) Add a function with the name `onBypassed` and implement the function.



```

onBypassed : function () {
  ...
  this._oList.removeSelections(true);
},

```

Figure 162: Function onBypassed

c) Add a function with the name `_onMasterMatched` and implement the function.



```

_onMasterMatched : function() {
  ...
  this.getOwnerComponent().getModel("appView").setProperty("/layout", "OneColumn");
},

```

Figure 163: Function onMasterMatched

d) Add a function with the name `onSelectionChange` and implement the function.

```
onSelectionChange : function(oEvent) {  
    var oItem, oCtx, sCarrid;  
    oItem = oEvent.getParameter("listItem");  
    oCtx = oItem.getBindingContext();  
    sCarrid = oCtx.getProperty("Carrid");  
    this._showCarrierDetails(sCarrid);  
},
```

Figure 164: Function on SelectionChange

```
onPress: function (oEvent) {
    var oItem, oCtx, sCarrid;
    oItem = oEvent.getSource();
    oCtx = oItem.getBindingContext();
    sCarrid = oCtx.getProperty("Carrid");

    this._showCarrierDetails(sCarrid);
},
```

Figure 165: Function onPress

15. Implement the function `showCarrierDetails`.

```
showCarrierDetails : function(sCarrid) {
  var oRouter = this.getRouter();
  oRouter.navTo("Carrier", {
    carrierId: sCarrid
  }, false /*with history*/ );
}
```

Figure 166: Implement showCarrierDetails

a) Open the file `Overview.controller.js` located in the controller folder of your project and change the implementation as instructed.

b) Save your changes.

16. Change the implementation of the Carrier controller. Add the following code to the Carrier controller implementation.

```
Carrier.controller.js x

1 sap.ui.define([
2     "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4     "use strict";
5
6     return Controller.extend("student00.sap.training.dynamicpage.controller.Carrier", {
7         ...onInit: function () {
8             ...var oRouter = this.getRouter();
9             oRouter.getRoute("Carrier").attachMatched(this._onObjectMatched, this);
10        },
11
12        ...getRouter: function () {
13            ...return sap.ui.core.UIComponent.getRouterFor(this);
14        },
15    });
16});
```

Figure 167: Implementation

```

16  .... _onObjectMatched: function (oEvent) {
17      ....     var oArgs, oView;
18      ....     oArgs = oEvent.getParameter("arguments");
19      ....     this._sCarrierId = oArgs.carrierId;
20      ....     oView = this.getView();
21
22  ....     oView.bindElement({
23         ....         path: "/ZBC_C_Carrier_TP('" + this._sCarrierId + "')",
24         ....         events: {
25             ....             change: this._onBindingChange.bind(this),
26             ....             dataRequested: function () {
27                 ....                 oView.setBusy(true);
28             },
29             ....             dataReceived: function () {
30                 ....                 oView.setBusy(false);
31             }
32         }
33     });
34     ....     this.getOwnerComponent().oListSelector.selectAListItem("/ZBC_C_Carrier_TP('" + this._sCarrierId + "')");
35     ....     var oViewModel = this.getOwnerComponent().getModel("appView");
36     ....     oViewModel.setProperty("/layout", "TwoColumnsMidExpanded");
37     ....     this.getOwnerComponent().setModel(oViewModel, "appView");
38 },

```

Figure 168: Implementation

```

40  .... _onBindingChange: function () {
41      ....     var oView = this.getView();
42      ....     var oElementBinding = oView.getElementBinding();
43      ....     // No data for the binding
44      ....     if (oElementBinding && !oElementBinding.getBoundContext()) {
45         ....         this.getOwnerComponent().oListSelector.clearMasterListSelection();
46         ....         this.getRouter().getTargets().display("NotFound");
47         ....         return;
48     }
49     ....     var sPath = oElementBinding.getPath();
50     ....     this.getOwnerComponent().oListSelector.selectAListItem(sPath);
51 },

```

Figure 169: Implementation

```

53  .... onCloseDetailPress: function () {
54      ....     this.getOwnerComponent().getModel("appView").setProperty(
55          ....         "/actionButtonsInfo/midColumn/fullScreen", false);
56      ....     this.getOwnerComponent().oListSelector.clearMasterListSelection();
57      ....     this.getRouter().navTo("Overview");
58 },
59
60  .... onToggleFullScreen: function () {
61      ....     var bFullScreen = this.getOwnerComponent().getModel("appView").getProperty(
62          ....         "/actionButtonsInfo/midColumn/fullScreen");
63      ....     this.getOwnerComponent().getModel("appView").setProperty(
64          ....         "/actionButtonsInfo/midColumn/fullScreen", !bFullScreen);
65      ....     if (!bFullScreen) {
66          ....         // store current layout and go full screen
67          ....         this.getOwnerComponent().getModel("appView").setProperty("/previousLayout",
68          ....             this.getOwnerComponent().getModel("appView").getProperty("/layout"));
69          ....         this.getOwnerComponent().getModel("appView").setProperty("/layout", "MidColumnFullScreen");
70     } else {
71         ....         // reset to previous layout
72         ....         this.getOwnerComponent().getModel("appView").setProperty("/layout",
73             this.getOwnerComponent().getModel("appView").getProperty("/previousLayout"));
74     }
75 },

```

Figure 170: Implementation



```

77  ...
78  ...
79  ...
80  ...
81  ...
82  ...
83  ...
84  ...
85  ...
86  ...
87  ...
88  ...
89  ...
90  ...
91  ...

```

```

onNavBack: function () {
  var oHistory, sPreviousHash, oRouter;
  oHistory = sap.ui.core.routing.History.getInstance();
  sPreviousHash = oHistory.getPreviousHash();

  if (sPreviousHash !== undefined) {
    window.history.go(-1);
  } else {
    oRouter = this.getRouter();
    oRouter.navTo("Overview", true /*no history*/ );
  }
}

```

Figure 171: Implementation

a) Open the file `Carrier.controller.js` located in the controller folder of your project.

b) Implement your changes.

## 17. Test your application.

a) Select the `index.html` file located in the `webapp` folder of your project and choose from the context menu `Run → Run as Web Application`.

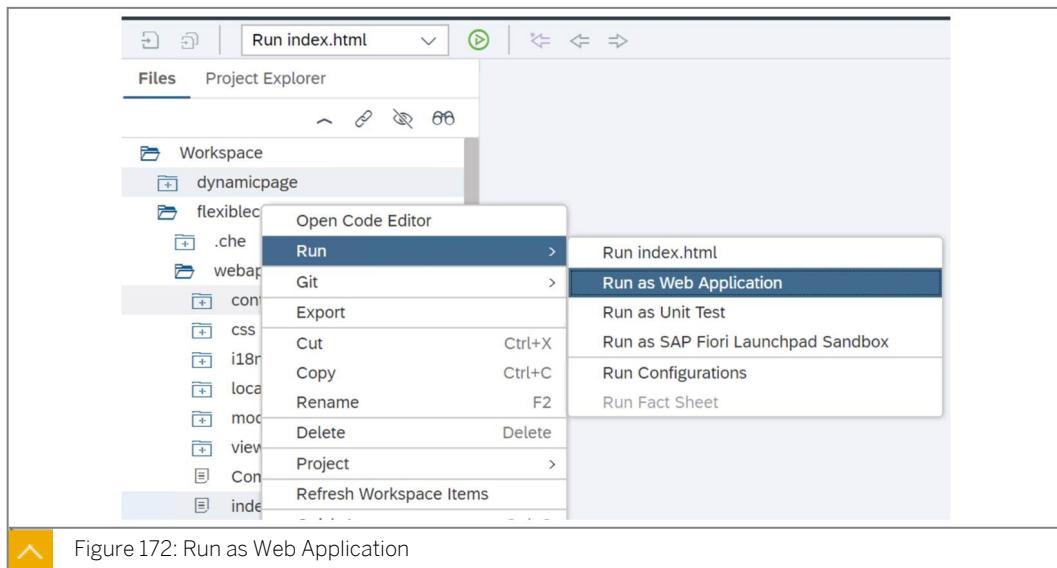


Figure 172: Run as Web Application

b) Log on with your credentials when required.

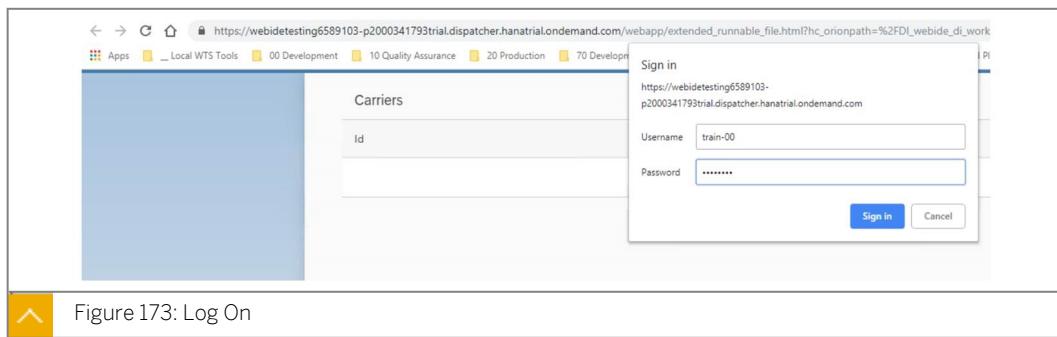
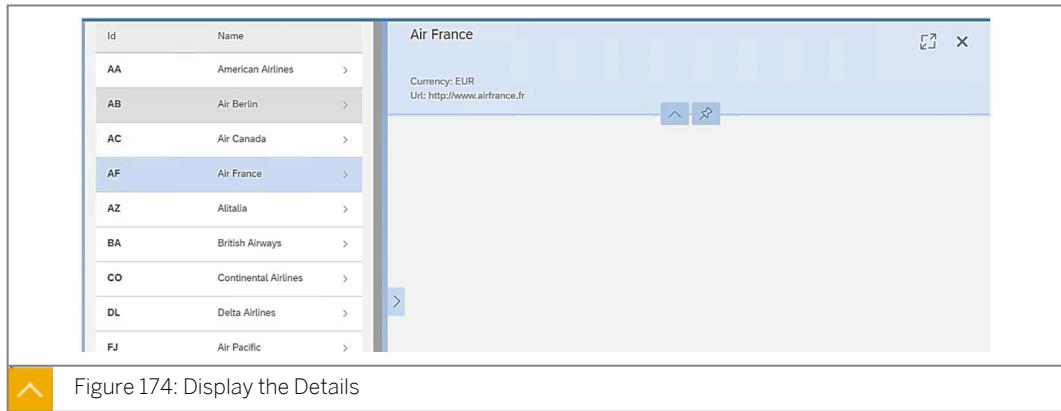


Figure 173: Log On

c) Test your application. Choose a carrier and display the details.



## Unit 12

### Exercise 11

# Implement Value Helps



#### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The exercises show the application of group 00. Use your own student and group number for the exercises as assigned by your instructor.

#### Business Example

In the following exercise you will implement Value Help.

#### Task 1: Log on to the SAP Web IDE Full-Stack and create a SAPUI5 project

1. Log on to the SAP Web IDE Full-Stack and create an SAPUI5 project with the following settings:

Table 45: Project Properties.

Parameter	Value
Project name	valuehelp
Namespace	student##.sap.training
View Type	XML
View Name	Main
SAPUI5 Version	1.60

#### Task 2: Implement value help

Implement the value help floorplan as described in the SAP Fiori Guidelines. This provides the user with a selection dialog to select a carrier. The application makes use of the OData service with the name UX\_TRAVEL\_SRV.

1. Add the UX\_TRAVEL\_SRV OData service to your project. The service is referenced by the default model of your application.
2. Replace the generated `sap.m.Page` control in the `Main.view.xml` with a `sap.f.DynamicPage` control. Assign the following attributes and values to the `sap.f.DynamicPage` control.

Table 46: sap.f.DynamicPage Attributes and Values.

Attribute	Value
id	idDynPage
headerExpanded	true

Attribute	Value
toggleHeaderOnTitleClick	true
preserveHeaderStateOnScroll	true

3. Add an `sap.f.DynamicPageTitle` control to the title aggregation of the `sap.f.DynamicPage` control. Implement a `sap.m.Input` control with the following attributes and values inside the heading aggregation of the `sap.f.DynamicPageTitle` control. Add the aggregation `suggestionItems` to the newly added `sap.m.Input` control and add an `sap.ui.core.Item` control as template. Assign the binding for attribute `Carrid` to the text attribute of the `sap.ui.core.Item` control

Table 47: `sap.m.Input` control Attributes and Values

Attribute	Value
id	idCarInput
placeholder	Enter carrier id
showSuggestion	true
showValueHelp	true
suggestionItems	{/UX_C_Carrier_TP}
type	Text
valueHelpRequest	onValueHelpRequest
submit	onSubmit

4. Create an XML fragment with the name `CarrierSelectionDialog` inside the view folder of your project. Implement selection `Dialog`, using the `sap.m.SelectDialog` control with the listed attributes and values. Add a template of type `sap.m.StandardListItem` to the new `sap.m.SelectDialog` and use the listed attributes and values.

Table 48: `sap.m.SelectDialog` Attributes and Values

Attribute	Value
title	Carriers
class	sapUiPopupWithPadding
item	{/UX_C_Carrier_TP}
search	onValueHelpSearch
confirm	onValueHelpClose
cancel	onValueHelpClose

Table 49: `sap.m.StandardListItem` Attributes and Values

Attribute	Value
iconDensityAware	false

Attribute	Value
iconInsert	false
title	{Carrid}
description	{Carrname}

### Task 3: Task Controller Implementation

Implement the event handler `onValueHelpRequest` inside the `Main.controller.js` file. The implementation obtains the value from the source object of the event and assigns the value to a local variable named `sInputValue`. The ID of the source object is stored inside a member variable with name `_sInputId`.

1. Create an instance of `CarrierSelectionDialog` and store the instance inside a member variable with the name `_oValueHelpDialog`. Get the binding of the items aggregation from the dialog and apply a filter for `Carrid` with `FilterOperator.Contains` and remember the AMD (Asynchronous module definition). Pass the `sInputValue` variable to the filter, then open the dialog.
2. Test the status of your implementation.
3. Implement the function `onValueHelpSearch` to handle events of `SelectDialog`. The `onValueHelpSearch` function is called when the user enters a value in the search field of the dialog and starts the search. The function needs to read the event parameter value and use this value for filtering. Make it possible for the user to search for the carrier name (property `carrname`, `FilterOperator.StartsWith`) and carrier ID (property `Carrid`, `FilterOperator.Contains`).
4. Implement the function `onValueHelpClose` to handle the event of the `SelectDialog`. The `onValueHelpClose` function is called when the dialog is closed. The event object of the event handler will get a parameter `selectedItem`. The parameter contains the item of the list selected by the user. Read the title property from the selected item and display the title in the input field of `Main.view.xml`. Recall that you stored the id of the input field in the member variable `_sInputId`. Reset the filters from the select dialog to complete the implementation. Now test your application.

### Task 4: Show the details and the available flights of the selected carrier.

Add a fragment to the project to show the flights for a selected carrier.

1. Implement a new fragment with the name `Flights.fragment.xml` in the view folder of your project and add XML namespace aliases for `sap.m` as default and `sap.ui.core` with alias `core` to the `sap.ui.core.FragmentDefinition`.
2. Add an `sap.m.Table` control with the listed attributes and values to the newly-created fragment.

Table 50: `sap.m.Table` control Attributes and Values.

Attribute	Value
id	idFlights
items	{ path: 'to_Flight', sorter: { path: 'Carrid' } }
mode	SingleSelectMaster

Attribute	Value
growingThreshold	true
growing	true
visible	true

Add an *sap.m.Table* control to your fragment.



```

Flights.fragment.xml x
1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'Carrid' } }" mode="SingleSelectMaster" growing="true"
3     |   growingThreshold="10" visible="true"
4   </Table>
5 </core:FragmentDefinition>

```

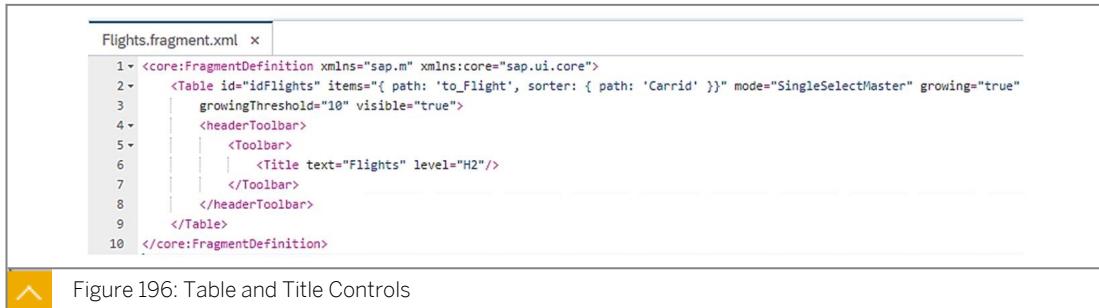
Figure 195: Table Control

3. Add an *sap.m.Title* to the *headerToolbar* aggregation of the *sap.m.Table* with the listed attributes and values.

Table 51: *sap.m.Title* control Attributes and Values.

Attribute	Value
text	Flights
level	H2

Add the *headerToolbar* aggregation to the *sap.m.Table* control and add an *sap.m.Title* control with the listed attributes and values.



```

Flights.fragment.xml x
1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'Carrid' } }" mode="SingleSelectMaster" growing="true"
3     |   growingThreshold="10" visible="true"
4   <headerToolbar>
5     |     <Toolbar>
6     |       |   <Title text="Flights" level="H2"/>
7     |     </Toolbar>
8   </headerToolbar>
9 </Table>
10 </core:FragmentDefinition>

```

Figure 196: Table and Title Controls

4. Add columns to the table with the listed attributes and values.
5. Add one *sap.m.Text* control to each column using the listed attributes.

Table 57: *sap.m.Text* control Attributes and Values

Attribute	Value
text	Carrier
text	Flight no.
text	Date
text	Max seats

Attribute	Value
text	Occ. seats

<pre> Flights.fragment.xml x 1- &lt;core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core"&gt; 2-   &lt;Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'Carrid' }}" mode="SingleSelectMaster" growing="true" 3-     growingThreshold="10" visible="true"&gt; 4-       &lt;headerToolbar&gt; 5-         &lt;Toolbar&gt; 6-           &lt;Title text="Flights" level="H2"/&gt; 7-         &lt;/Toolbar&gt; 8-       &lt;/headerToolbar&gt; 9-       &lt;columns&gt; 10-         &lt;Column width="12em"&gt; 11-           &lt;Text text="Carrier"/&gt; 12-         &lt;/Column&gt; 13-         &lt;Column minScreenWidth="Tablet" demandPopin="true"&gt; 14-           &lt;Text text="Flight no."/&gt; 15-         &lt;/Column&gt; 16-         &lt;Column minScreenWidth="Tablet" demandPopin="true" hAlign="Right"&gt; 17-           &lt;Text text="Date"/&gt; 18-         &lt;/Column&gt; 19-         &lt;Column minScreenWidth="Tablet" demandPopin="true" hAlign="Center"&gt; 20-           &lt;Text text="max. seats"/&gt; 21-         &lt;/Column&gt; 22-         &lt;Column hAlign="Right"&gt; 23-           &lt;Text text="occ. seats"/&gt; 24-         &lt;/Column&gt; 25-       &lt;/columns&gt; 26-     &lt;/Table&gt; 27-   &lt;/core:FragmentDefinition&gt; </pre>	 Figure 198: Text Control
---	--

Add an *sap.m.Text* control to each of the columns and assign the attributes and values.

6. Add an *sap.m.ColumnListItem* to the items aggregation of the table and add UI controls to the cells aggregation of the *sap.m.ColumnListItem*.

**Task 5: Extend the implementation of the Main.view.xml so it displays the details of the selected carrier**

1. Add an *sap.f.DynamicPageHeader* to the header aggregation of the *DynamicPage* control with the listed attributes and values.

Table 58: *sap.f.DynamicPageHeader* Attributes and Values.

Attribute	Value
pinnable	false

2. Add an *sap.m.FlexBox* control to the content aggregation with the listed attributes and values.

Table 59: *sap.m.FlexBox* control Attributes and Values.

Attribute	Value
alignItems	Start
justifyContent	SpaceBetween
id	idHeaderLayout
visible	false

3. Add an `sap.m.Panel` control to the items aggregation of the `sap.m.FlexBox` control and add the listed attributes.

Table 60: `sap.m.Panel` control Attributes and Values

Attribute	
<code>backgroundDesign</code>	Transparent
<code>class</code>	<code>sapUiNoContentPadding sapUiSmallMarginTop</code>

4. Add a `sap.ui.layout.HorizontalLayout` with the following attribute and value to the content aggregation of the `sap.m.Panel` control. Use layout as an XML namespace alias.

Table 61: `sap.ui.layout.HorizontalLayout` Attribute and Value

Attribute	Value
<code>allowWrapping</code>	<code>true</code>

5. Add an `sap.ui.layout.VerticalLayout` with the listed attributes and values to the content aggregation of the `sap.ui.layout.HorizontalLayout` and add the following UI controls. Add an `sap.m.Text` control with attribute `text` and assign the value **Airline** to the control. Add an `sap.m.ObjectAttribute` control with the attributes `title` and `text`. Assign the value **Carrier** to the `title` attribute and bind the `text` attribute to the property `Carname` of the entity. Add an `sap.m.ObjectAttribute` control with the attributes `title` and `text`. Assign the value **Carrier id** to the `title` attribute and bind the `text` attribute to the property `Carrid` of the entity.
6. Add an `sap.ui.layout.VerticalLayout` with the following attributes and values to the content aggregation of the `sap.ui.layout.HorizontalLayout` and add the following UI controls. Add the attribute `class` to the `VerticalLayout` control and assign the value **sapUiMediumMarginBegin** to the attribute. Add an `sap.m.Text` control with attribute `text` and assign the value **Airline** details to the control. Add an `sap.m.ObjectAttribute` control with the attributes `title` and `text`. Assign the value **Currency** to the `title` attribute and bind the `text` attribute to the property `Currcode` of the entity. Add an `sap.m.ObjectAttribute` control with the attributes `title` and `text`. Assign the value **URL** to the `title` attribute and bind the `text` attribute to the property `URL` of the entity.
7. Add an `sap.m.MessagePage` control to the content aggregation of the `sap.f.DynamicPage` control with the listed attributes and values.

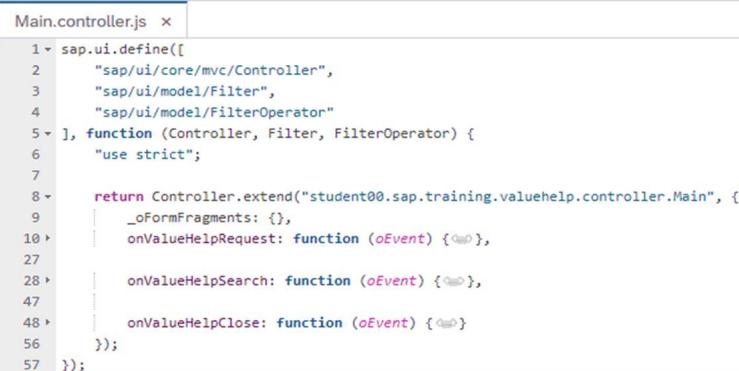
Table 62: `sap.m.MessagePage` control Attributes and Values.

	Value
<code>showHeader</code>	<code>false</code>
<code>icon</code>	<code>sap-icon://search</code>

#### Task 6: Implement controller logic to display the carrier details when the user selects a carrier from the list of carriers

1. Open the file `Main.controller.js` to declare a member variable `_oFormFragments` and add to the implementation by assigning a literal JavaScript object to the variable. This

variable acts as a map for fragments. Fragment instances can be stored by name in the map.



```

1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     _oFormFragments: {},
10    onValueHelpRequest: function (oEvent) {
11    },
12    onValueHelpSearch: function (oEvent) {
13    },
14    onValueHelpClose: function (oEvent) {
15    }
16  });
17 });

```

Figure 202: Member Variable

2. Add and implement the `onInit` function of the controller from the `Main` view. Obtain the UI control `idHeaderLayout` reference and assign the reference to a member variable of the controller. Name the variable `_oDynamicPage` and set the visibility of the layout to `false`.



```

1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     _oFormFragments: {},
10    onInit: function () {
11      this._oDynamicPage = this.byId("idHeaderLayout");
12      this._oDynamicPage.setVisible(false);
13    },
14    onValueHelpRequest: function (oEvent) {
15    },
16    onValueHelpSearch: function (oEvent) {
17    },
18    onValueHelpClose: function (oEvent) {
19    }
20  });
21 });

```

Figure 203: Function

3. Implement the function named `_createFlightsTable`. The function creates an instance of the flights fragment. The function checks if an instance of the fragment was already created. When the instance of the fragment map exists it should be returned. If not the instance must be created. Assign the instance to the map and it should return to the caller. After instantiation of the fragment, obtain a reference to the UI control with ID `idDynPage`. Call on the obtained reference `destroyContent` function and add the new fragment object by calling the `setContent` function on the dynamic page object.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     _oFormFragments: {},
10
11   onInit: function () {},
12
13   onValueHelpRequest: function (oEvent) {},
14
15   onValueHelpSearch: function (oEvent) {},
16
17   onValueHelpClose: function (oEvent) {},
18
19   _createFlightsTable : function() {
20     var oFragment = this._oFormFragments["Flights"];
21     if (oFragment) {
22       return oFragment;
23     }
24     oFragment = sap.ui.xmlfragment(this.getView().getId(), "student00.sap.training.valuehelp.view.Flights");
25     var oDynPage = this.byId("idDynPage");
26     oDynPage.destroyContent();
27     oDynPage.setContent(oFragment);
28     this._oFormFragments["Flights"] = oFragment;
29     return oFragment;
30   }
31 });
32 });
33 });
34 });
35 });
36 });
37 });
38 });
39 });
40 });
41 });
42 });
43 });
44 });
45 });
46 });
47 });
48 });
49 });
50 });
51 });
52 });
53 });
54 });
55 });
56 });
57 });
58 });
59 });
60 });
61 });
62 });
63 });
64 });
65 });
66 });
67 });
68 });
69 });
70 });
71 });
72 });
73 });
74 });
75 });
76 });
77 });
78 });
79 });
80 });
81 });

```

 Figure 204: Function

4. Extend the implementation of the event handler `onValueHelpClose` by setting the visibility of the `DynamicPage` object stored in the member variable `_oDynamicPage` to `true`. Call the function `_createFlightsTable` and assign the return value to a local variable. Bind the fragment to the binding path of the selected carrier. Also bind the UI control with id `idHeaderLayout` to that binding path. Implement the function.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     _oFormFragments: {},
10
11   onInit: function () {},
12
13   onValueHelpRequest: function (oEvent) {},
14
15   onValueHelpSearch: function (oEvent) {},
16
17   onValueHelpClose: function (oEvent) {
18     var oSelectedItem = oEvent.getParameter("selectedItem");
19     if (oSelectedItem) {
20       var oCarridInput = this.getView().byId(this._sInputId);
21       oCarridInput.setValue(oSelectedItem.getTitle());
22     }
23     oEvent.getSource().getBinding("items").filter([{}]);
24     this._oDynamicPage.setVisible(true);
25     var oFlightsFragment = this._createFlightsTable();
26     oFlightsFragment.bindElement(sPath);
27     this.byId("idHeaderLayout").bindElement(sPath);
28   }
29 });
30 });
31 });
32 });
33 });
34 });
35 });
36 });
37 });
38 });
39 });
40 });
41 });
42 });
43 });
44 });
45 });
46 });
47 });
48 });
49 });
40 });
41 });
42 });
43 });
44 });
45 });
46 });
47 });
48 });
49 });
50 });
51 });
52 });
53 });
54 });
55 });
56 });
57 });
58 });
59 });
60 });
61 });
62 });
63 });
64 });
65 });
66 });
67 });
68 });
69 });
70 });
71 });
72 });
73 });
74 });
75 });
76 });
77 });
78 });
79 });
80 });
81 });

```

 Figure 205: Implement Function

5. Test your implementation.

The details of the selected carrier are displayed.

The screenshot shows a SAP Fiori application window titled 'valuehelp'. The URL is <https://webidetesting4308489-p2000341793trial.dispatcher.hanatrial.ondemand.com/webapp/extend...>. The top navigation bar includes 'Apps', 'Local WTS Tools', and several status indicators for development and quality assurance environments. The main content area is titled 'AA' and displays 'Airline details' for American Airlines (Carrier: American Airlines, Carrier Id: AA, Currency: USD, URL: <http://www.aa.com>). Below this is a table titled 'Flights' with the following data:

Carrier	Flight no.	Date	max. seats	occ. seats
AA	0017	May 10, 2018	385	375 >
AA				
AA	0017	Jun 11, 2018	385	370 >
AA				
AA	0017	Jul 13, 2018	385	374 >
AA				
AA	0017	Aug 14, 2018	385	371 >
AA				
AA	0017	Sep 15, 2018	385	373 >
AA				

A blue gear icon with the number '371' is positioned next to the last flight entry. The bottom of the window has a yellow navigation bar with a back arrow and the text 'Figure 207: Carrier'.

## Implement Value Helps



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, naming of fields and buttons as well as steps may differ from the exercise solution. The exercises show the application of group 00. Use your own student and group number for the exercises as assigned by your instructor.

### Business Example

In the following exercise you will implement Value Help.

#### Task 1: Log on to the SAP Web IDE Full-Stack and create a SAPUI5 project

1. Log on to the SAP Web IDE Full-Stack and create an SAPUI5 project with the following settings:

Table 45: Project Properties.

Parameter	Value
Project name	valuehelp
Namespace	student##.sap.training
View Type	XML
View Name	Main
SAPUI5 Version	1.60

- a) Choose *File* → *New* → *Project from Template* and choose the option **SAPUI5 Application Project** and *Next*.
- b) Name your project **valuehelp** and in the *Namespace* field enter the value **student##.sap.training**, select UI5 version 1.60, and choose *Next*.
- c) Choose XML as a View Type and enter the value **Main** in the View Name field and choose *Finish*.
- d) Your created project will appear in the project explorer of the SAP Web IDE Full-Stack.

#### Task 2: Implement value help

Implement the value help floorplan as described in the SAP Fiori Guidelines. This provides the user with a selection dialog to select a carrier. The application makes use of the OData service with the name UX\_TRAVEL\_SRV.

1. Add the UX\_TRAVEL\_SRV OData service to your project. The service is referenced by the default model of your application.

- a) Select your project and from the context menu, choose *New → OData service*.
  - b) Choose *FSD\_100 – FSD system* from the list of available systems.
  - c) Enter your username and password for the front-end server.
  - d) Search for the **UX\_TRAVEL\_SRV service**, then select the service from the list of results and choose *Next*.
  - e) Ensure the radio button labelled *Use default model* is selected and choose *Next*.
  - f) Complete the data source assignment by choosing *Finish*.
2. Replace the generated *sap.m.Page* control in the *Main.view.xml* with a *sap.f.DynamicPage* control. Assign the following attributes and values to the *sap.f.DynamicPage* control.

Table 46: *sap.f.DynamicPage* Attributes and Values.

Attribute	Value
<i>id</i>	<i>idDynPage</i>
<i>headerExpanded</i>	<i>true</i>
<i>toggleHeaderOnTitleClick</i>	<i>true</i>
<i>preserveHeaderStateOnScroll</i>	<i>true</i>

- a) Open the file *Main.view.xml* and remove the generated *sap.m.Page* control.
  - b) Add an XML namespace with alias *f* to your implementation and assign the *sap.f-namespace*.
  - c) Add a control of type *sap.f.DynamicPage* with the listed attribute and value pairs to the pages-aggregation of the *sap.m.App-control* control.
  - d) Save your changes.
3. Add an *sap.f.DynamicPageTitle* control to the title aggregation of the *sap.f.DynamicPage* control. Implement a *sap.m.Input* control with the following attributes and values inside the heading aggregation of the *sap.f.DynamicPageTitle* control. Add the aggregation *suggestionItems* to the newly added *sap.m.Input* control and add an *sap.ui.core.Item* control as template. Assign the binding for attribute *Carrid* to the text attribute of the *sap.ui.core.Item* control

Table 47: *sap.m.Input* control Attributes and Values

Attribute	Value
<i>id</i>	<i>idCarInput</i>
<i>placeholder</i>	Enter carrier id
<i>showSuggestion</i>	<i>true</i>
<i>showValueHelp</i>	<i>true</i>
<i>suggestionItems</i>	<i>{/UX_C_Carrier_TP}</i>
<i>type</i>	<i>Text</i>

Attribute	Value
valueHelpRequest	onValueHelpRequest
submit	onSubmit

- Open the file `Main.view.xml`.
- Add the title aggregation to the `sap.f. DynamicPage` control.
- Add a control of type `sap.f. DynamicPageTitle` inside the title aggregation.
- Add the heading aggregation of the `sap.f. DynamicPageTitle` control.
- Add an `sap.m. Input` control to the heading aggregation using the listed attributes. Add the `suggestionItems` aggregation to the control and insert an `sap.ui.core.Item` control with the described configuration. Remember to add an XML namespace alias for the `sap.ui.core` namespace. At the end your implementation should look like the following:

```

<pages>
  <f:DynamicPage id="idDynPage" headerExpanded="true" toggleHeaderOnTitleClick="true" preserveHeaderStateOnScroll="true">
    <f:title>
      <f:DynamicPageTitle>
        <f:heading>
          <Input id="idCarInput" placeholder="Enter carrier id ..." showSuggestion="true" showValueHelp="true" suggestionItems="/UX_C_Carrier_TP"
            type="Text" valueHelpRequest="onValueHelpRequest" submit="onSubmit">
            <suggestionItems>
              <core:Item text="{Carrid}">
            </suggestionItems>
          </Input>
        </f:heading>
      </f:DynamicPageTitle>
    </f:title>
  </f:DynamicPage>
</pages>

```

Figure 175: Heading Aggregation

- Create an XML fragment with the name `CarrierSelectionDialog` inside the view folder of your project. Implement selection *Dialog*, using the `sap.m.SelectDialog` control with the listed attributes and values. Add a template of type `sap.m.StandardListItem` to the new `sap.m.SelectDialog` and use the listed attributes and values.

Table 48: `sap.m.SelectDialog` Attributes and Values

Attribute	Value
title	Carriers
class	sapUiPopupWithPadding
item	{/UX_C_Carrier_TP}
search	onValueHelpSearch
confirm	onValueHelpClose
cancel	onValueHelpClose

Table 49: `sap.m.StandardListItem` Attributes and Values

Attribute	Value
iconDensityAware	false
iconInsert	false
title	{Carrid}

Attribute	Value
description	{Carrname}

- a) Create a new file with the name **CarrierSelectionDialog.fragment.xml**. Select the view folder of your project and choose from the context menu *New → File*.
- b) Enter **CarrierSelectionDialog.fragment.xml** in the new dialog and confirm *OK*
- c) Add a UI control of type *sap.ui.core.FragmentDefinition* to the new file. Create two XML namespaces for the SAPUI5 namespace *sap.m* and *sap.ui.core*
- d) Add a *sap.m.SelectDialog* to the fragment and assign the listed attribute and value pairs
- e) Add a *sap.m.StandardListItem* control to the added *SelectDialog* control and add the listed attribute and value pairs.
- f) At the end your implementation should look the following figure:



```

1 <core:FragmentDefinition
2   xmlns="sap.m"
3   xmlns:core="sap.ui.core">
4   <SelectDialog
5     title="Carriers"
6     class="sapUiPopupWithPadding"
7     items="{/UX_C_Carrier_TP}"
8     search="onValueHelpSearch"
9     confirm="onValueHelpClose"
10    cancel="onValueHelpClose">
11    <StandardListItem
12      iconDensityAware="false"
13      iconInset="false"
14      title="{Carrid}"
15      description="{Carrname}" />
16  </SelectDialog>
17 </core:FragmentDefinition>

```

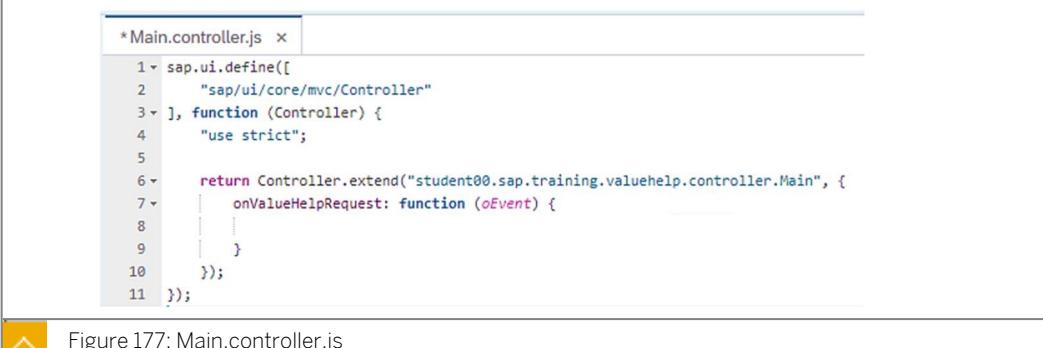
- g) Save your changes and close the file.

### Task 3: Task Controller Implementation

Implement the event handler *onValueHelpRequest* inside the *Main.controller.js* file. The implementation obtains the value from the source object of the event and assigns the value to a local variable named *sInputValue*. The ID of the source object is stored inside a member variable with name *\_sInputId*.

1. Create an instance of *CarrierSelectionDialog* and store the instance inside a member variable with the name *\_oValueHelpDialog*. Get the binding of the items aggregation from the dialog and apply a filter for *Carrid* with *FilterOperator.Contains* and remember the AMD (Asynchronous module definition). Pass the *sInputValue* variable to the filter, then open the dialog.

- a) Open the file `Main.controller.js` and add a function with the name `onValueHelpRequest`.



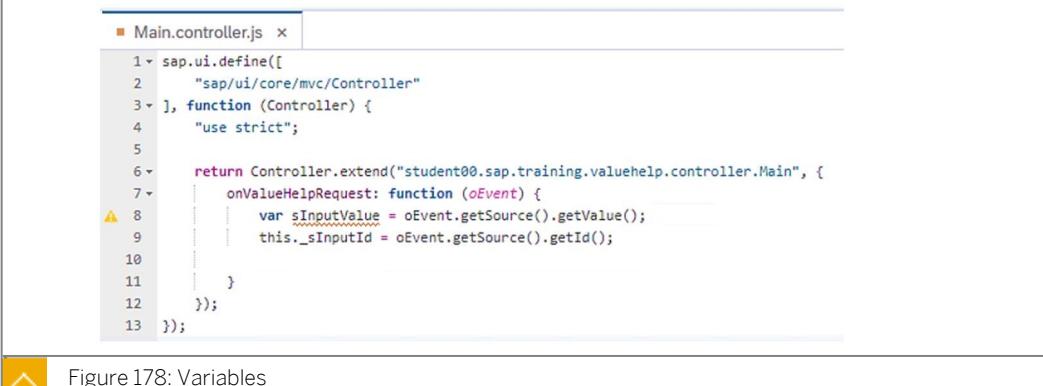
```

* Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
7     onValueHelpRequest: function (oEvent) {
8
9   }
10 });
11 });

```

Figure 177: Main.controller.js

- b) Create the two required variables and assign the appropriate values.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
7     onValueHelpRequest: function (oEvent) {
8       var sInputValue = oEvent.getSource().getValue();
9       this._sInputId = oEvent.getSource().getId();
10
11     }
12 });
13 });

```

Figure 178: Variables

- c) Create an instance of `CarrierSelectionDialog` and assign the instance to the member variable with the name `_oValueHelpDialog`. Set a reference.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
7     onValueHelpRequest: function (oEvent) {
8       var sInputValue = oEvent.getSource().getValue();
9       this._sInputId = oEvent.getSource().getId();
10
11       if (!this._oValueHelpDialog) {
12         this._oValueHelpDialog = sap.ui.xmlfragment(
13           "student00.sap.training.valuehelp.view.CarrierSelectionDialog",
14           this
15         );
16         this.getView().addDependent(this._oValueHelpDialog);
17
18       }
19     }
20 });

```

Figure 179: Dialog

- d) Add a reference to `sap.ui.model.Filter` and `sap.ui.model.FilterOperator` to your controller.



```

1 * Main.controller.js x
2
3 sap.ui.define([
4   "sap/ui/core/mvc/Controller",
5   "sap/ui/model/Filter",
6   "sap/ui/model/FilterOperator"
7 ], function (Controller, Filter, FilterOperator) {
8   "use strict";
9
10  return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
11    onValueHelpRequest: function (oEvent) {
12      var sInputValue = oEvent.getSource().getValue();
13      this._sInputId = oEvent.getSource().getId();
14      if (!this._oValueHelpDialog) {
15        this._oValueHelpDialog = sap.ui.xmlfragment(
16          "student00.sap.training.valuehelp.view.CarrierSelectionDialog",
17          this
18        );
19        this.getView().addDependent(this._oValueHelpDialog);
20      }
21    });
22 });

```

Figure 180: Reference

## e) Implement the filter functionality.



```

1 * Main.controller.js x
2
3 sap.ui.define([
4   "sap/ui/core/mvc/Controller",
5   "sap/ui/model/Filter",
6   "sap/ui/model/FilterOperator"
7 ], function (Controller, Filter, FilterOperator) {
8   "use strict";
9
10  return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
11    onValueHelpRequest: function (oEvent) {
12      var sInputValue = oEvent.getSource().getValue();
13      this._sInputId = oEvent.getSource().getId();
14      if (!this._oValueHelpDialog) {
15        this._oValueHelpDialog = sap.ui.xmlfragment(
16          "student00.sap.training.valuehelp.view.CarrierSelectionDialog",
17          this
18        );
19        this.getView().addDependent(this._oValueHelpDialog);
20      }
21      this._oValueHelpDialog.getBinding("items").filter([new Filter(
22        "Carrid",
23        FilterOperator.Contains, sInputValue
24      )]);
25    }
26  });
27 });

```

Figure 181: Filter

## f) Call the open-function on the dialog instance to show the requested value help dialog.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     onValueHelpRequest: function (oEvent) {
10       var sInputValue = oEvent.getSource().getValue();
11       this._sInputId = oEvent.getSource().getId();
12       if (this._oValueHelpDialog) {
13         this._oValueHelpDialog = sap.ui.xmlfragment(
14           "student00.sap.training.valuehelp..view.CarrierSelectionDialog",
15           this
16         );
17         this.getView().addDependent(this._oValueHelpDialog);
18       }
19       this._oValueHelpDialog.getBinding("items").filter([new Filter(
20         "Carrid",
21         FilterOperator.Contains, sInputValue
22       )]);
23
24       this._oValueHelpDialog.open(sInputValue);
25     }
26   });
27 });

```

Figure 182: Dialog

2. Test the status of your implementation.

- a) Select the file `index.html` and from the context menu, choose *Run → Run as Web Application*.

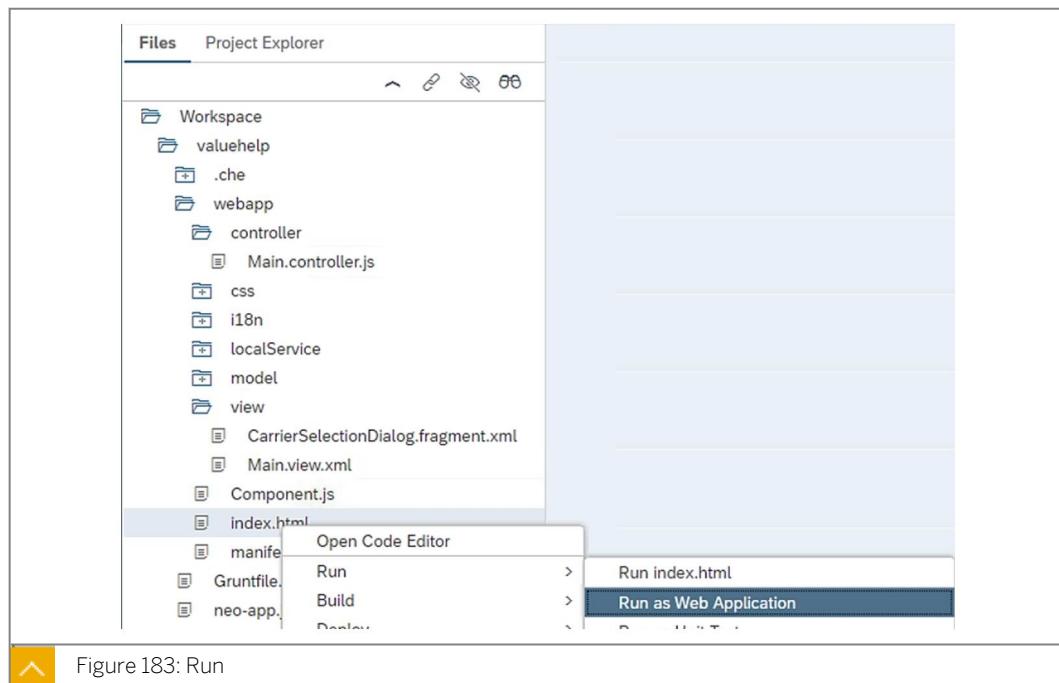


Figure 183: Run

- b) Enter your front-end server credentials when required.

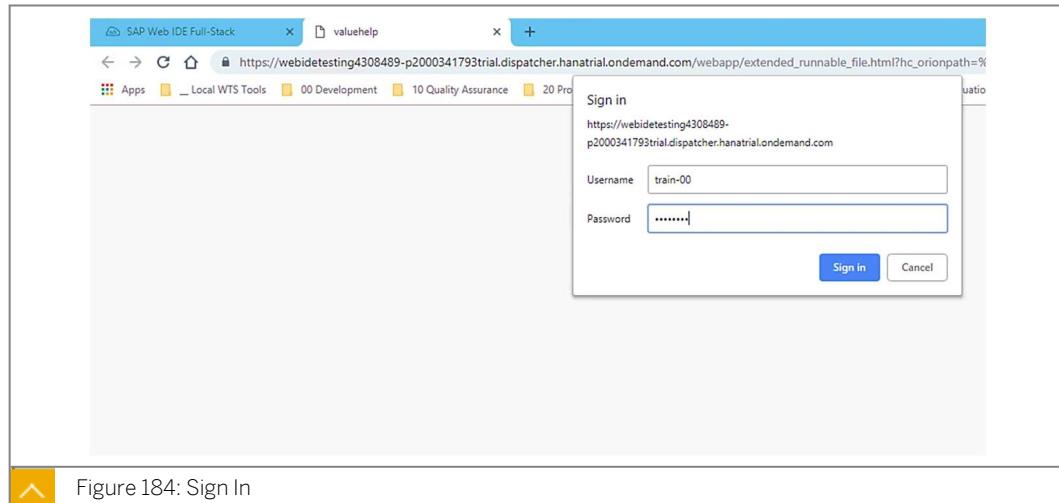


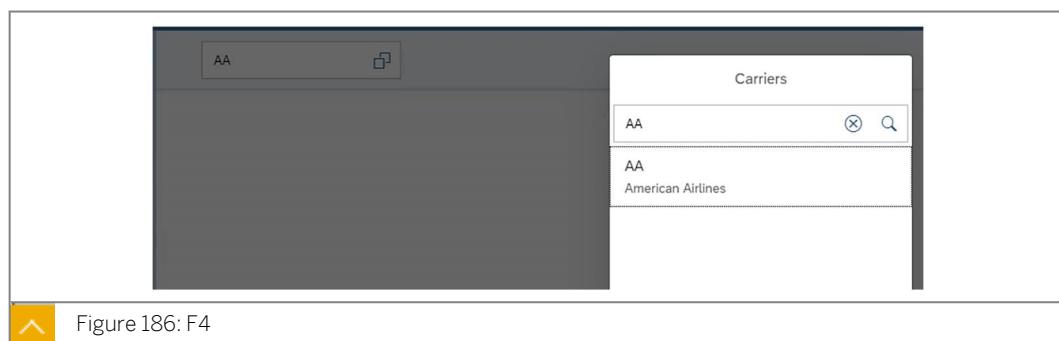
Figure 184: Sign In

c) Enter a carrier ID in the *value help* field.



Figure 185: CarrierID

d) Press *F4*.

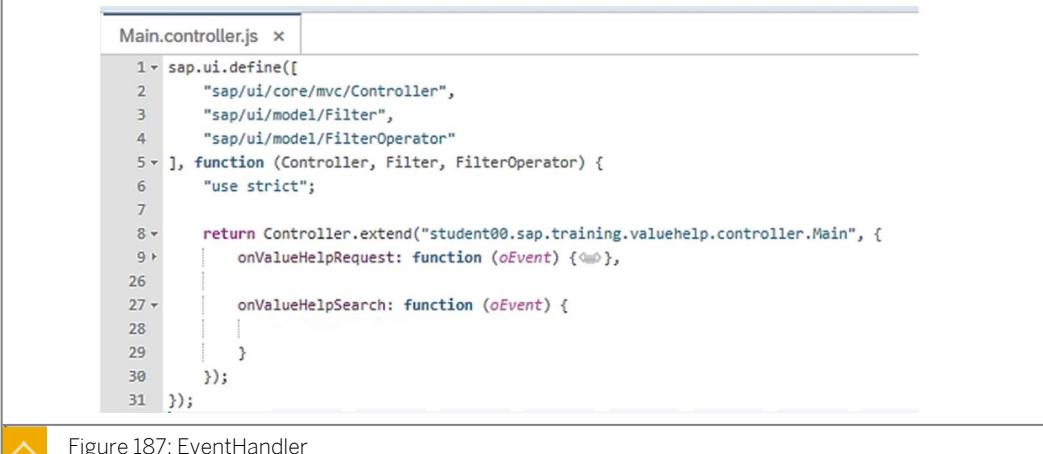


e) Close the dialog. Currently, there is no functionality implemented when closing or selecting a carrier from the list.

3. Implement the function `onValueHelpSearch` to handle events of `SelectDialog`. The `onValueHelpSearch` function is called when the user enters a value in the search field of the dialog and starts the search. The function needs to read the event parameter `value` and use this value for filtering. Make it possible for the user to search for the carrier name (property `carriename`, `FilterOperator.StartsWith`) and carrier ID (property `Carrid`, `FilterOperator.Contains`).

a) Open the file `Main.controller.js`.

b) Add an event handler function `onValueHelpSearch` to the `Main` controller.



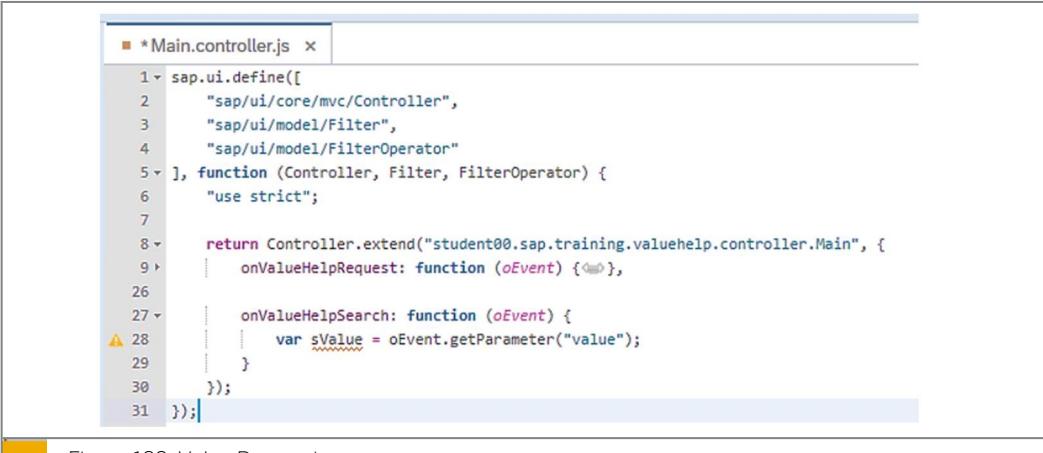
```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     onValueHelpRequest: function (oEvent) {
10    },
11
12     onValueHelpSearch: function (oEvent) {
13       ...
14     }
15   });
16 });
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31 });

```

Figure 187: EventHandler

- c) Obtain the value parameter from the event object and assign it to a local variable.



```

* Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     onValueHelpRequest: function (oEvent) {
10    },
11
12     onValueHelpSearch: function (oEvent) {
13       ...
14       var sValue = oEvent.getParameter("value");
15     }
16   });
17 });
18
19
20
21
22
23
24
25
26
27
28
29
30
31 });

```

Figure 188: Value Parameter

- d) Create the necessary filter objects and start the filter process.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     onValueHelpRequest: function (oEvent) {
10    },
11
12     onValueHelpSearch: function (oEvent) {
13       ...
14       var sValue = oEvent.getParameter("value");
15       var oFilter = new Filter(
16         "Carrid",
17         FilterOperator.Contains, sValue
18       );
19
20       var oFilter2 = new Filter(
21         "Carrname",
22         FilterOperator.StartsWith, sValue
23       );
24
25       var oFilter3 = new Filter({
26         filters: [oFilter, oFilter2],
27         and: false
28       });
29
30       oEvent.getSource().getBinding("items").filter([oFilter3]);
31     }
32   });
33 });
34
35
36
37
38
39
40
41
42
43
44
45
46
47 });

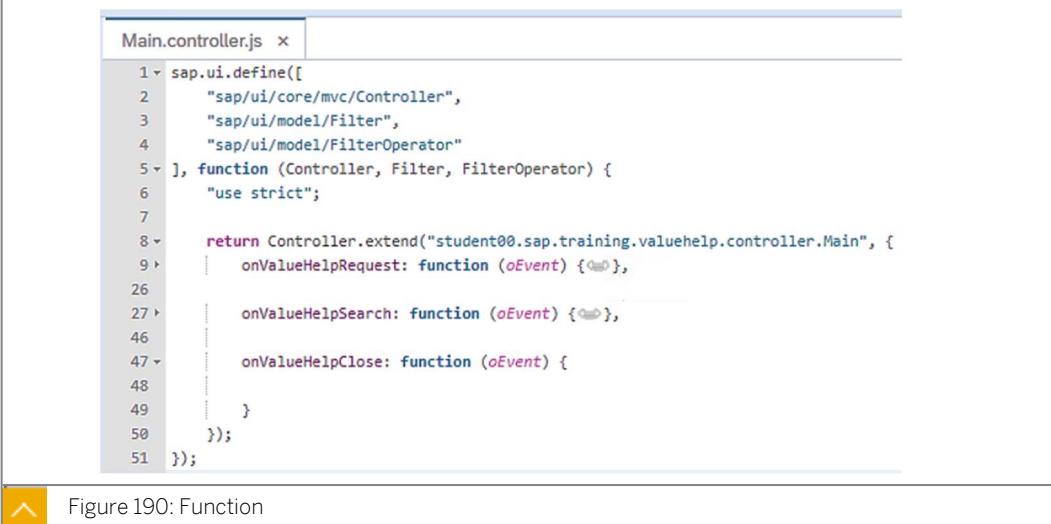
```

Figure 189: Filter

4. Implement the function `onValueHelpClose` to handle the event of the `SelectDialog`.

The `onValueHelpClose` function is called when the dialog is closed. The event object of the event handler will get a parameter `selectedItem`. The parameter contains the item of the list selected by the user. Read the title property from the selected item and display the title in the input field of `Main.view.xml`. Recall that you stored the id of the input field in the member variable `_sInputId`. Reset the filters from the select dialog to complete the implementation. Now test your application.

- a) Add a function `onValueHelpClose` to the implementation of `Main.controller.js`.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     onValueHelpRequest: function (oEvent) {},
10    onValueHelpSearch: function (oEvent) {},
11    onValueHelpClose: function (oEvent) {
12    }
13  });
14 });

```

Figure 190: Function

- b) Implement the function.



```

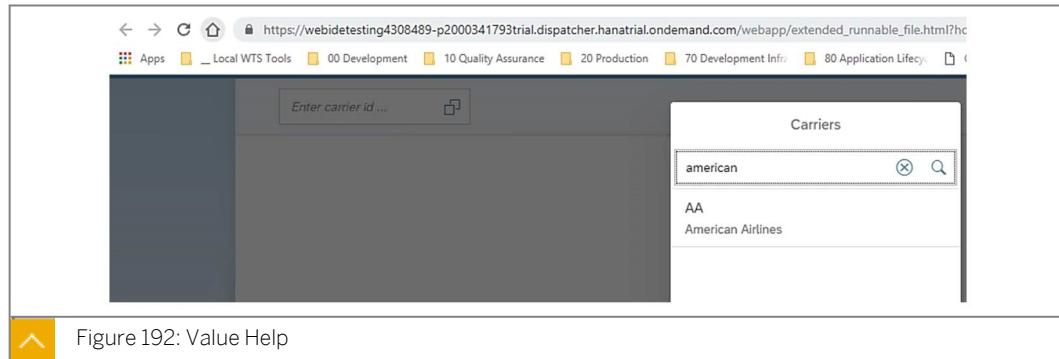
Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     onValueHelpRequest: function (oEvent) {},
10    onValueHelpSearch: function (oEvent) {},
11    onValueHelpClose: function (oEvent) {
12      var oSelectedItem = oEvent.getParameter("selectedItem");
13      if (oSelectedItem) {
14        var oCarridInput = this.getView().byId(this._sInputId);
15        oCarridInput.setValue(oSelectedItem.getTitle());
16      }
17      oEvent.getSource().getBinding("items").filter([ ]);
18    }
19  });
20 });

```

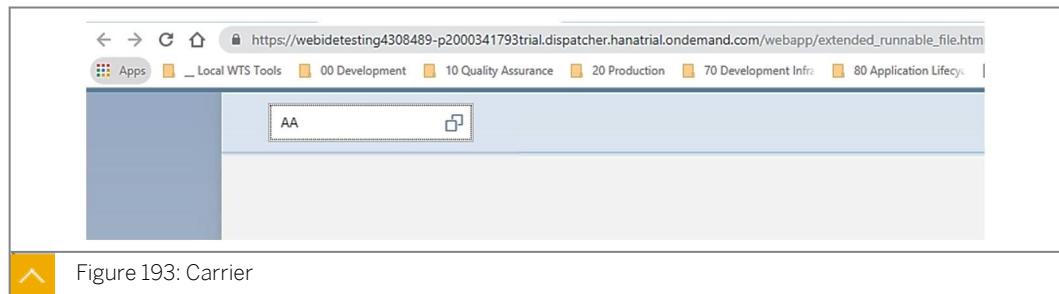
Figure 191: Implement Function

- c) Restart your application and open the value help dialog.

- d) Enter a carrier ID or carrier name in the search field of the value help dialog and start the search.



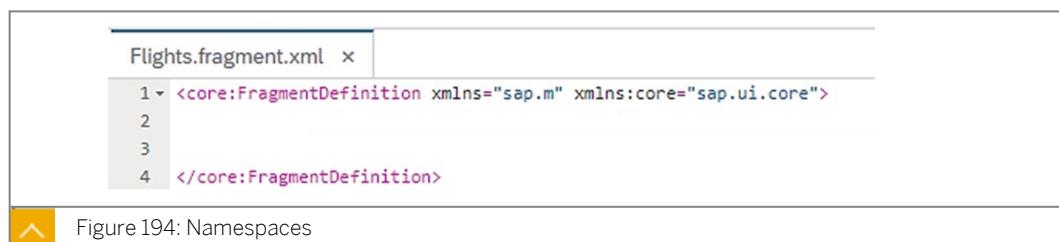
- e) Select one carrier from the list. The dialog closes and the carrier ID is shown in the input field.



#### Task 4: Show the details and the available flights of the selected carrier.

Add a fragment to the project to show the flights for a selected carrier.

1. Implement a new fragment with the name **Flights.fragment.xml** in the view folder of your project and add XML namespace aliases for `sap.m` as default and `sap.ui.core` with alias `core` to the `sap.ui.core.FragmentDefinition`.
  - a) Select the view folder and choose from the context menu.
  - b) Enter the name for the file you are creating in the input field and confirm **OK**.
  - c) Implement the FragmentDefinition and add the required namespaces.



2. Add an `sap.m.Table` control with the listed attributes and values to the newly-created fragment.

Table 50: sap.m.Table control Attributes and Values.

Attribute	Value
id	idFlights
items	{ path: 'to_Flight', sorter: { path: 'Carrid' } }
mode	SingleSelectMaster
growingThreshold	true

Attribute	Value
growing	true
visible	true

Add an *sap.m.Table* control to your fragment.



```

Flights.fragment.xml x
1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'Carrid' } }" mode="SingleSelectMaster" growing="true"
3   |   growingThreshold="10" visible="true"
4   </Table>
5 </core:FragmentDefinition>

```

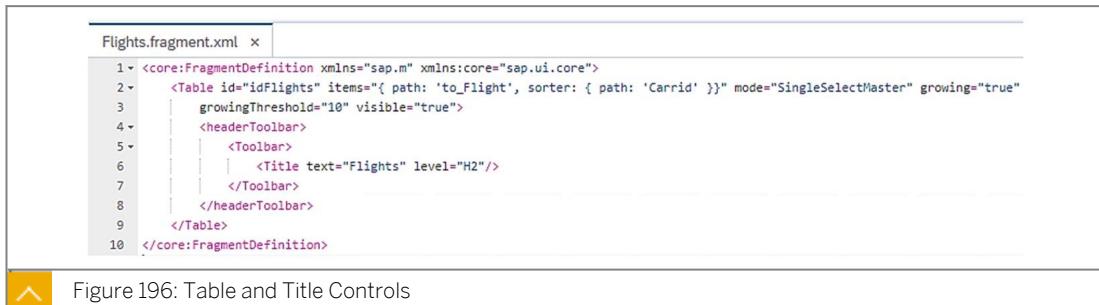
Figure 195: Table Control

3. Add an *sap.m.Title* to the *headerToolbar* aggregation of the *sap.m.Table* with the listed attributes and values.

Table 51: *sap.m.Title* control Attributes and Values.

Attribute	Value
text	Flights
level	H2

Add the *headerToolbar* aggregation to the *sap.m.Table* control and add an *sap.m.Title* control with the listed attributes and values.



```

Flights.fragment.xml x
1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'Carrid' } }" mode="SingleSelectMaster" growing="true"
3   |   growingThreshold="10" visible="true"
4   |   <headerToolbar>
5   |   |   <Toolbar>
6   |   |   |   <Title text="Flights" level="H2"/>
7   |   |   </Toolbar>
8   |   </headerToolbar>
9   </Table>
10 </core:FragmentDefinition>

```

Figure 196: Table and Title Controls

4. Add columns to the table with the listed attributes and values.

- a) First column-control:

Table 52: First column-control Attribute and Value

Attribute	Value
width	12em

- b) Second column-control:

Table 53: Second column-control Attributes and Values

Attribute	Value
minScreenWidth	Tablet
demandPopin	true

## c) Third column-control:

Table 54: Third column-control Attributes and Values

minScreenWidth	Tablet
demandPopin	true
hAlign	Right

## d) Fourth column-control:

Table 55: Fourth column-control Attributes and Values

Attribute	Value
minScreenWidth	Tablet
demandPopin	true
hAlign	Center



```

Flights.fragment.xml x
1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'Carrid' }}" mode="SingleSelectMaster" growing="true"
3   | growingThreshold="10" visible="true">
4     <headerToolbar>
5       <Toolbar>
6         <Title text="Flights" level="H2"/>
7       </Toolbar>
8     </headerToolbar>
9     <columns>
10    <Column width="12em">
11    </Column>
12    <Column minScreenWidth="Tablet" demandPopin="true">
13    </Column>
14    <Column minScreenWidth="Tablet" demandPopin="true" hAlign="Right">
15    </Column>
16    <Column minScreenWidth="Tablet" demandPopin="true" hAlign="Center">
17    </Column>
18    <Column hAlign="Right">
19    </Column>
20  </columns>
21 </Table>
22 </core:FragmentDefinition>

```

Figure 197: Controls

## e) Fifth column-control:

Table 56: Fifth column-control Attribute and Value.

Attribute	Value
hAlign	Right

5. Add one *sap.m.Text* control to each column using the listed attributes.

Table 57: sap.m.Text control Attributes and Values

Attribute	Value
text	Carrier
text	Flight no.
text	Date
text	Max seats
text	Occ. seats



```

Flights.fragment.xml x
1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'Carrid' } }" mode="SingleSelectMaster" growing="true"
3     growingThreshold="10" visible="true"
4     <headerToolbar>
5       <Toolbar>
6         <Title text="Flights" level="H2"/>
7       </Toolbar>
8     </headerToolbar>
9     <columns>
10    <Column width="12em">
11      <Text text="Carrier"/>
12    </Column>
13    <Column minScreenWidth="Tablet" demandPopin="true">
14      <Text text="Flight no."/>
15    </Column>
16    <Column minScreenWidth="Tablet" demandPopin="true" hAlign="Right">
17      <Text text="Date"/>
18    </Column>
19    <Column minScreenWidth="Tablet" demandPopin="true" hAlign="Center">
20      <Text text="max. seats"/>
21    </Column>
22    <Column hAlign="Right">
23      <Text text="occ. seats"/>
24    </Column>
25  </columns>
26 </Table>
27 </core:FragmentDefinition>

```

Figure 198: Text Control

Add an *sap.m.Text* control to each of the columns and assign the attributes and values.

6. Add an *sap.m.ColumnListItem* to the items aggregation of the table and add UI controls to the cells aggregation of the *sap.m.ColumnListItem*.
  - a) In the first cell, insert a UI control of type *sap.m.ObjectIdentifier* and bind the attribute *title* to the *Carrid* property of the entity.
  - b) In the second cell, insert a UI control of type *sap.m.Text* and bind the attribute *text* to the property *Connid* of the entity.
  - c) In the third cell, insert a UI control of type *sap.m.Text*. Bind the attribute *text* to the property *F1date* and declare that the property is of type *sap.ui.model.type.Date* of the entity.
  - d) In the fourth cell, insert a UI control of type *sap.m.Text* and bind the attribute *text* to the property *Seatsmaxx* of the entity.
  - e) In the fifth cell, insert a UI control of type *sap.m.Text* and bind the attribute *text* to the property *Seatsocc* of the entity.
  - f) Add the *items* aggregation to the table.
  - g) Add the *sap.m.ColumnListItem* control to the *items* aggregation and add the controls and binding.

**Task 5: Extend the implementation of the `Main.view.xml` so it displays the details of the selected carrier**

1. Add an `sap.f.DynamicPageHeader` to the header aggregation of the `DynamicPage` control with the listed attributes and values.

Table 58: `sap.f.DynamicPageHeader` Attributes and Values.

Attribute	Value
<code>pinnable</code>	<code>false</code>

2. Add an `sap.m.FlexBox` control to the content aggregation with the listed attributes and values.

Table 59: `sap.m.FlexBox` control Attributes and Values.

Attribute	Value
<code>alignItems</code>	<code>Start</code>
<code>justifyContent</code>	<code>SpaceBetween</code>
<code>id</code>	<code>idHeaderLayout</code>
<code>visible</code>	<code>false</code>

- a) Add the control aggregation to the `DynamicPageHeader` control.
- b) Add an `sap.m.FlexBox` control to the content aggregation and assign the defined attributes and values.



3. Add an `sap.m.Panel` control to the items aggregation of the `sap.m.FlexBox` control and add the listed attributes.

Table 60: `sap.m.Panel` control Attributes and Values

Attribute	
<code>backgroundDesign</code>	<code>Transparent</code>

Attribute	
class	sapUiNoContentPadding sapUiSmallMarginTop

- a) Add the items aggregation to the FlexBox control.
- b) Add a control of type *sap.m.Panel* and add the listed attributes.
4. Add a *sap.ui.layout.HorizontalLayout* with the following attribute and value to the content aggregation of the *sap.m.Panel* control. Use layout as an XML namespace alias.

Table 61: *sap.ui.layout.HorizontalLayout* Attribute and Value

Attribute	Value
allowWrapping	true

- a) Add the XML namespace alias layout pointing to the *UI5-namespace sap.ui.layout*.
- b) Add the content aggregation to the panel control.
- c) Add a control of type *sap.ui.layout.HorizontalLayout* to the content aggregation.
5. Add an *sap.ui.layout.VerticalLayout* with the listed attributes and values to the content aggregation of the *sap.ui.layout.HorizontalLayout* and add the following UI controls. Add an *sap.m.Text* control with attribute text and assign the value **Airline** to the control. Add an *sap.m.ObjectAttribute* control with the attributes title and text. Assign the value **Carrier** to the title attribute and bind the text attribute to the property *Carname* of the entity. Add an *sap.m.ObjectAttribute* control with the attributes title and text. Assign the value **Carrier id** to the title attribute and bind the text attribute to the property *Carrid* of the entity.
- a) Add a control of type *sap.ui.layout.VerticalLayout*.
- b) Add the UI controls to the *VerticalLayout* control and bind the attributes as defined.
6. Add an *sap.ui.layout.VerticalLayout* with the following attributes and values to the content aggregation of the *sap.ui.layout.HorizontalLayout* and add the following UI controls. Add the attribute class to the *VerticalLayout* control and assign the value **sapUiMediumMarginBegin** to the attribute. Add an *sap.m.Text* control with attribute text and assign the value **Airline** details to the control. Add an *sap.m.ObjectAttribute* control with the attributes title and text. Assign the value **Currency** to the title attribute and bind the text attribute to the property *Currcode* of the entity. Add an *sap.m.ObjectAttribute* control with the attributes title and text. Assign the value **URL** to the title attribute and bind the text attribute to the property *URL* of the entity.
- a) Add another *sap.ui.layout.VerticalLayout* control to the content aggregation.
- b) Add the UI controls and the specified binding.
- c) At the end your implementation should look like the following figure:



```

<f:header>
  <f:DynamicPageHeader pinnable="false">
    <f:content>
      <FlexBox alignItems="Start" justifyContent="SpaceBetween" id="idHeaderLayout" visible="false">
        <items>
          <Panel backgroundDesign="Transparent" class="sapUiNoContentPadding sapUiSmallMarginTop">
            <content>
              <layout:HorizontalLayout allowWrapping="true">
                <layout:VerticalLayout>
                  <Title text="Airline"/>
                  <ObjectAttribute title="Carrier" text="{Carrname}"/>
                  <ObjectAttribute title="Carrier Id" text="{Carrid}"/>
                </layout:VerticalLayout>
                <layout:VerticalLayout class="sapUiMediumMarginBegin">
                  <Title text="Airline details"/>
                  <ObjectAttribute title="Currency" text="{Currcode}"/>
                  <ObjectAttribute title="URL" text="{Url}"/>
                </layout:VerticalLayout>
              </layout:HorizontalLayout>
            </content>
          </Panel>
        </items>
      </FlexBox>
    </f:content>
  </f:DynamicPageHeader>
</f:header>

```

Figure 200: Implementation

7. Add an *sap.m.MessagePage* control to the content aggregation of the *sap.f.DynamicPage* control with the listed attributes and values.

Table 62: *sap.m.MessagePage* control Attributes and Values.

	Value
showHeader	false
icon	sap-icon://search

- a) Add the content aggregation to the *sap.f.DynamicPage* control.
- b) Add a *sap.m.MessagePage* control to the content aggregation and add the listed attributes to the control.
- c) At the end your implementation should look like the following figure:



```

<f:header>
  <f:DynamicPageHeader pinnable="false">
    <f:content>
      <FlexBox alignItems="Start" justifyContent="SpaceBetween" id="idHeaderLayout" visible="false">
        <items>
          <Panel backgroundDesign="Transparent" class="sapUiNoContentPadding sapUiSmallMarginTop">
            <content>
              <layout:HorizontalLayout allowWrapping="true">
                <layout:VerticalLayout>
                  <Title text="Airline"/>
                  <ObjectAttribute title="Carrier" text="{Carrname}"/>
                  <ObjectAttribute title="Carrier Id" text="{Carrid}"/>
                </layout:VerticalLayout>
                <layout:VerticalLayout class="sapUiMediumMarginBegin">
                  <Title text="Airline details"/>
                  <ObjectAttribute title="Currency" text="{Currcode}"/>
                  <ObjectAttribute title="URL" text="{Url}"/>
                </layout:VerticalLayout>
              </layout:HorizontalLayout>
            </content>
          </Panel>
        </items>
      </FlexBox>
    </f:content>
  </f:DynamicPageHeader>
</f:header>
<f:content>
  <MessagePage showHeader="false" icon="sap-icon://search"/>
</f:content>
</f:DynamicPage>
</pages>

```

Figure 201: Implementation

**Task 6: Implement controller logic to display the carrier details when the user selects a carrier from the list of carriers**

1. Open the file `Main.controller.js` to declare a member variable `_oFormFragments` and add to the implementation by assigning a literal JavaScript object to the variable. This variable acts as a map for fragments. Fragment instances can be stored by name in the map.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     _oFormFragments: {},
10    onValueHelpRequest: function (oEvent) {
11    },
12    onValueHelpSearch: function (oEvent) {
13    },
14    onValueHelpClose: function (oEvent) {
15    }
16  });
17
18 });

```

Figure 202: Member Variable

2. Add and implement the `onInit` function of the controller from the *Main* view. Obtain the UI control `idHeaderLayout` reference and assign the reference to a member variable of the controller. Name the variable `_oDynamicPage` and set the visibility of the layout to `false`.



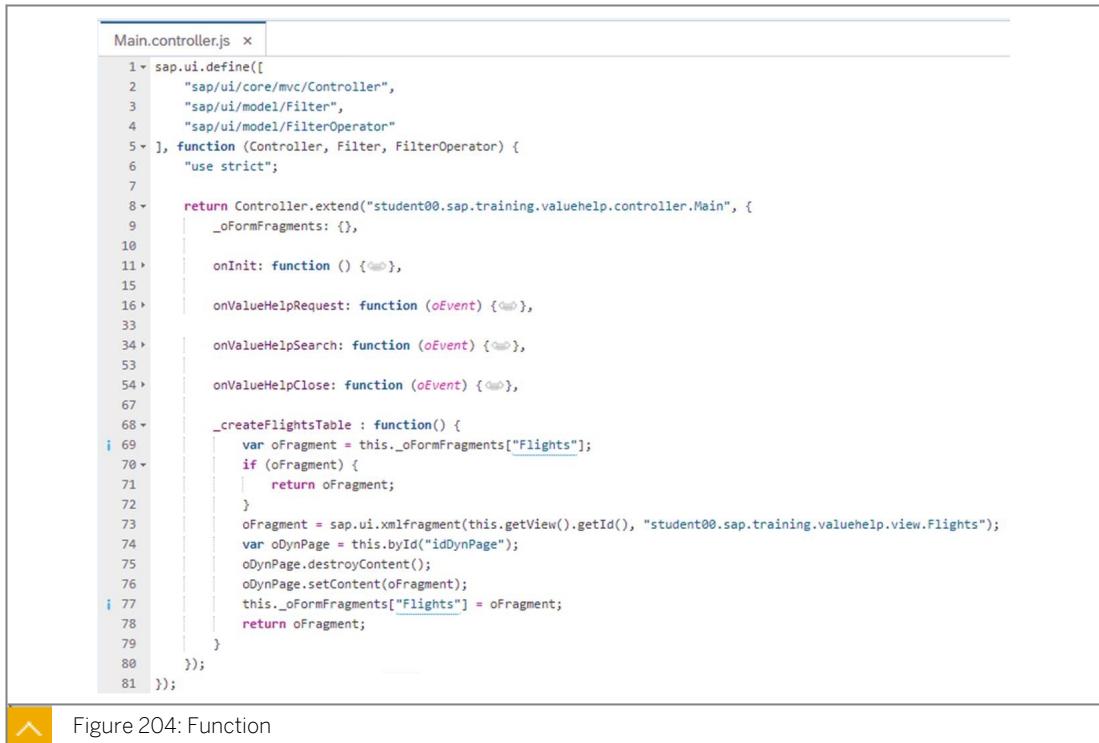
```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     _oFormFragments: {},
10    onInit: function () {
11      this._oDynamicPage = this.byId("idHeaderLayout");
12      this._oDynamicPage.setVisible(false);
13    },
14    onValueHelpRequest: function (oEvent) {
15    },
16    onValueHelpSearch: function (oEvent) {
17    },
18    onValueHelpClose: function (oEvent) {
19    }
20  });
21
22 });

```

Figure 203: Function

3. Implement the function named `_createFlightsTable`. The function creates an instance of the flights fragment. The function checks if an instance of the fragment was already created. When the instance of the fragment map exists it should be returned. If not the instance must be created. Assign the instance to the map and it should return to the caller. After instantiation of the fragment, obtain a reference to the UI control with ID `idDynPage`. Call on the obtained reference `destroyContent` function and add the new fragment object by calling the `setContent` function on the dynamic page object.



```

Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     _oFormFragments: {},
10
11   /**
12    * @function
13    * @param {sap.ui.event.Event} oEvent
14    */
15   onValueHelpRequest: function (oEvent) {
16
17     /**
18      * @function
19      * @param {sap.ui.event.Event} oEvent
20      */
21   onValueHelpSearch: function (oEvent) {
22
23     /**
24      * @function
25      * @param {sap.ui.event.Event} oEvent
26      */
27   onValueHelpClose: function (oEvent) {
28
29     /**
30      * @function
31      * @param {sap.ui.event.Event} oEvent
32      */
33     _createFlightsTable : function() {
34       var oFragment = this._oFormFragments["Flights"];
35       if (oFragment) {
36         return oFragment;
37       }
38       oFragment = sap.ui.xmlfragment(this.getView().getId(), "student00.sap.training.valuehelp.view.Flights");
39       var oDynPage = this.byId("idDynPage");
40       oDynPage.destroyContent();
41       oDynPage.setContent(oFragment);
42       this._oFormFragments["Flights"] = oFragment;
43       return oFragment;
44     }
45   });
46 });
47 });
48 });
49 });
50 });
51 });
52 });
53 });
54 });
55 });
56 });
57 });
58 });
59 });
60 });
61 });
62 });
63 });
64 });
65 });
66 });
67 });
68 });
69 });
70 });
71 });
72 });
73 });
74 });
75 });
76 });
77 });
78 });
79 });
80 });
81 });

```

 Figure 204: Function

4. Extend the implementation of the event handler `onValueHelpClose` by setting the visibility of the `DynamicPage` object stored in the member variable `_oDynamicPage` to `true`. Call the function `_createFlightsTable` and assign the return value to a local variable. Bind the fragment to the binding path of the selected carrier. Also bind the UI control with id `idHeaderLayout` to that binding path. Implement the function.



```

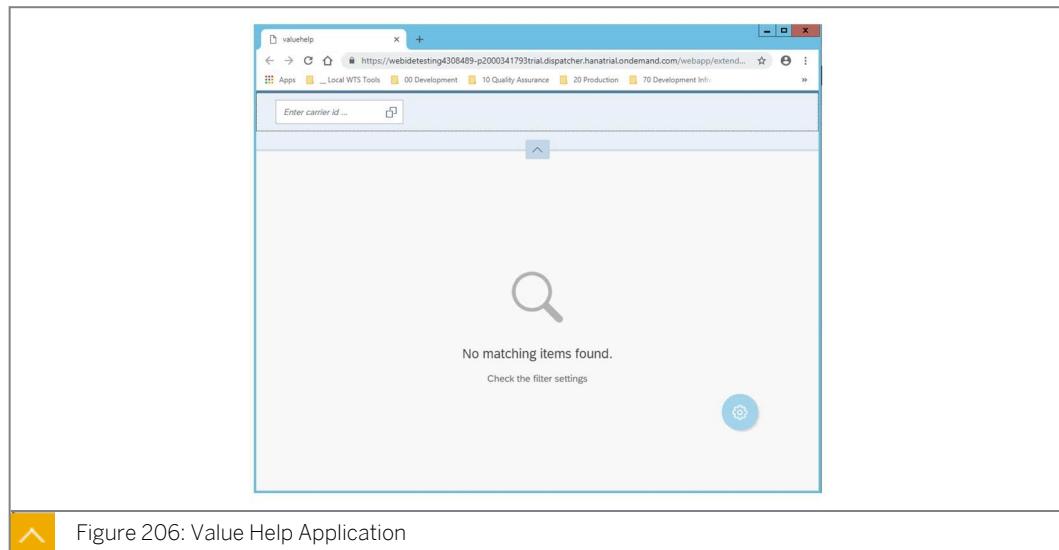
Main.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.valuehelp.controller.Main", {
9     _oFormFragments: {},
10
11   /**
12    * @function
13    * @param {sap.ui.event.Event} oEvent
14    */
15   onValueHelpRequest: function (oEvent) {
16
17     /**
18      * @function
19      * @param {sap.ui.event.Event} oEvent
20      */
21   onValueHelpSearch: function (oEvent) {
22
23     /**
24      * @function
25      * @param {sap.ui.event.Event} oEvent
26      */
27   onValueHelpClose: function (oEvent) {
28     var oSelectedItem = oEvent.getParameter("selectedItem");
29     if (oSelectedItem) {
30       var oCarridInput = this.getView().byId(this._sInputId);
31       oCarridInput.setValue(oSelectedItem.getTitle());
32     }
33     oEvent.getSource().getBinding("items").filter([{}]);
34     this._oDynamicPage.setVisible(true);
35     var oFlightsFragment = this._createFlightsTable();
36     oFlightsFragment.bindElement(sPath);
37     this.byId("idHeaderLayout").bindElement(sPath);
38   },
39
40   _createFlightsTable : function() {
41   });
42 });
43 });
44 });
45 });
46 });
47 });
48 });
49 });
50 });
51 });
52 });
53 });
54 });
55 });
56 });
57 });
58 });
59 });
60 });
61 });
62 });
63 });
64 });
65 });
66 });
67 });
68 });
69 });
70 });
71 });
72 });
73 });
74 });
75 });
76 });
77 });
78 });
79 });
80 });
81 });

```

 Figure 205: Implement Function

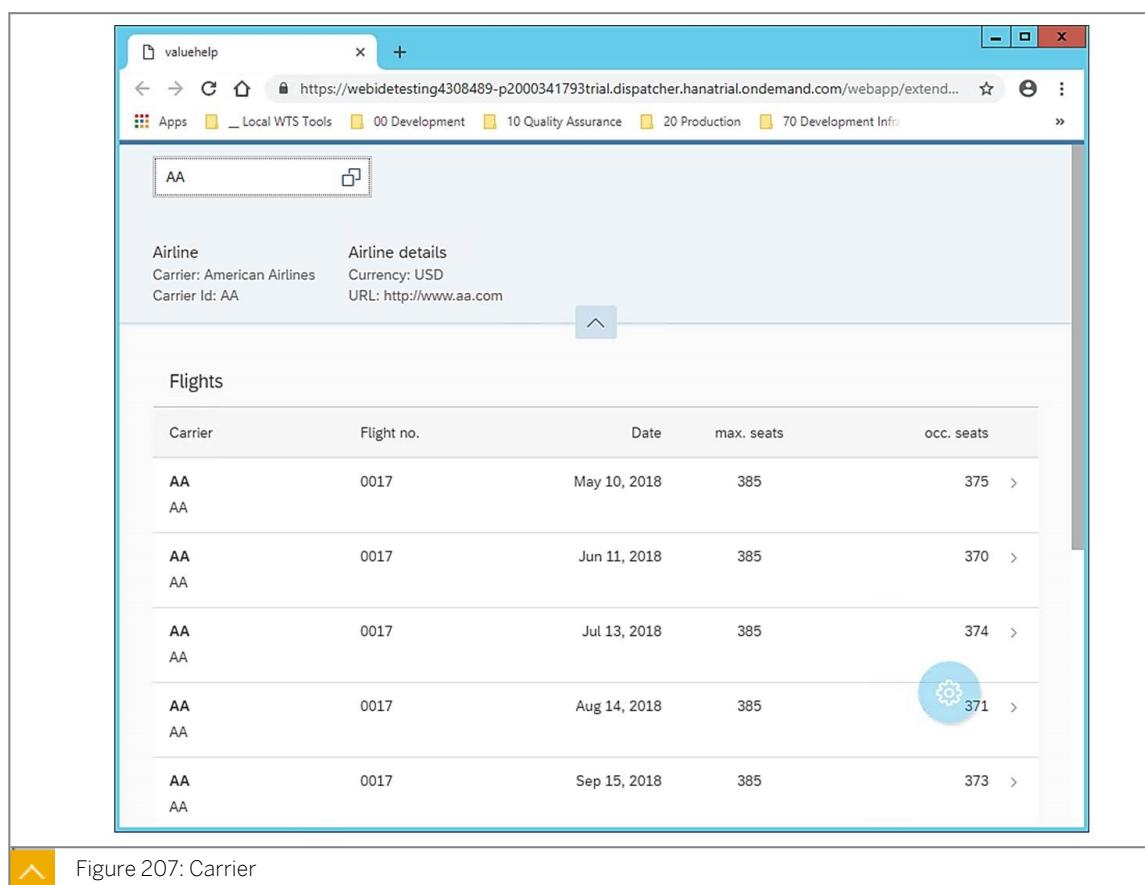
5. Test your implementation.

- Restart your application.
- The application is shown without a preselected carrier.



- Select a carrier from *F4 value help* as described.

The details of the selected carrier are displayed.



## Unit 12

### Exercise 12

# Implement a List Report



#### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

#### Business Example

In this exercise you learn how to extend a standard SAP Fiori application to create a list report floorplan.

This exercise uses the `flexiblecolumnlayout` project created in the previous exercise..

#### Task 1: Create a New Project

Export the project `flexiblecolumnlayout` that you created in the previous project and import the project with the name `listreport` into the workspace of your *SAP Web IDE Full-Stack*.

1. Log on to the SAP Web IDE Full-Stack, and export the project `flexiblecolumnlayout` that you created in one of the previous exercises.
2. Import the content of the file `flexiblecolumnlayout.zip` as a project with the name `listreport`.
3. Add the attribute `id` to the table control in the Overview-view and assign the value `idCarrierList` to the newly added attribute. Implement a search functionality to search the table of carriers, as described in the SAP Fiori guidelines. Add an `sap.m.SearchField` control and place the control in the `sap.m.Toolbar`. Add an `sap.m.ToolbarSpacer` control between the existing `sap.m.Title` control and the newly added `sap.m.SearchField` control. Implement the `onSearch` function. Use the listed properties as attributes for the `SearchField` control. The search functionality returns a list of carriers where the `carrid` or the `carrname` matches the specified value.

Table 63: SearchField Control Properties

Attribute	Value
<code>placeholder</code>	<code>Filter</code>
<code>search</code>	<code>onSearch</code>
<code>width</code>	<code>15rem</code>



Note:

Implement the search functionality in a reusable way as we use the same search and filter aspects in a later step.



Note:

Use the i18n model for the placeholder text. Add the value **Filter** with the key **sfPlaceholder** and type **#XTXT** into the i18n.property file.

4. Test the search implementation.
5. Add a UI control of type `sap.f.DynamicPageHeader` to the header aggregation of the existing `sap.f.DynamicPage` control located in the `Overview.view.xml` file. Assign the following attribute and value to the newly added `sap.f.DynamicPageHeader` control.

Table 64: `sap.f.DynamicPageHeader` attributes and values

Attribute	Value
<code>pinnable</code>	<code>false</code>

6. Add a UI control of type `sap.ui.comp.filterbar.FilterBar` to the content aggregation of the `sap.f.DynamicPageHeader` control. Use an XML namespace with alias **fb** to assign the SAPUI5 `sap.ui.comp.filterbar`. Add the listed attribute and values to the newly added `sap.ui.comp.filterbar.FilterBar` control.

Table 65: `sap.ui.comp.filterbar.FilterBar` control attribute and values

Attribute	Value
<code>id</code>	<code>idFilterBar</code>
<code>reset</code>	<code>onReset</code>
<code>search</code>	<code>onSearch</code>
<code>clear</code>	<code>onClear</code>
<code>cancel</code>	<code>onCancel</code>
<code>filtersDialogClose</code>	<code>onFiltersDialogClose</code>
<code>useToolbar</code>	<code>true</code>
<code>showGoOnFB</code>	<code>true</code>
<code>showRestoreButton</code>	<code>true</code>
<code>showClearButton</code>	<code>true</code>

7. Add two `sap.ui.comp.filterbar.FilterGroupItem` controls to the `filterItems` aggregation of the `sap.ui.comp.filterbar.FilterBar` with the following attributes and values.

Table 66: sap.ui.comp.filterbar.FilterGroupItem Control Attributes and Values

Attribute	Value
groupName	G1
visibleInFilterBar	true
groupTitle	carridGroup
name	A
label	Carrier



## Note:

Use the *i18n ResourceModel* and add the key *filterCarId* to the *i18n.properties* file. Assign the value **Carrier** to the key and use this key instead of the hardwired text.

Table 67: sap.ui.comp.filterbar.FilterGroupItem Control Attributes and Values

Attribute	Value
groupName	G2
visibleInFilterBar	true
groupTitle	carrnameGroup
name	B
label	Carrier name

8. Add a control of type `sap.m.Input` to each control aggregation of the `sap.ui.comp.filter.FilterGroupItem` controls. Assign the listed attributes and values to the controls.

Table 68: sap.m.Input Control 1 Attributes and Values

Attribute	Value
placeholder	{i18n>filterCarId}
liveChange	onFilterChange
id	idFilterCarrid

Table 69: sap.m.Input Control 2 Attributes and Values

Attribute	Value
placeholder	{i18n>filterCarrname}
liveChange	onFilterChange

Attribute	Value
ID	idFilterCarrname

Add a `sap.m.Input` control to each of the `sap.ui.comp.filter.FilterGroupItem` controls and assign the listed attribute and values.

```
1 <mvvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:fb="sap.ui.comp.filterbar"
2   controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3   <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4     <f:header>
5       <f:DynamicPageHeader pinnable="false">
6         <f:content>
7           <fb:FilterBar reset="onReset" search="onSearch" clear="onClear" cancel="onCancel" filtersDialogClosed="onFiltersDialogClosed"
8             useToolBar="true" showGoOnFB="true" showRestoreButton="true" showClearButton="true" id="idFilterBar">
9             <fb:filterGroupItems>
10               <fb:FilterGroupItem groupName="G1" visibleInFilterBar="true" groupTitle="carridGroup" name="A">
11                 <label>{i18n>filterCarId}</label>
12                 <fb:control>
13                   <Input id="idFilterCarrid" placeholder="{i18n>filterCarId}" liveChange="onFilterChange"/>
14                 </fb:control>
15               </fb:FilterGroupItem>
16               <fb:FilterGroupItem groupName="G2" visibleInFilterBar="true" groupTitle="carrnameGroup" name="B">
17                 <label>{i18n>filterCarname}</label>
18                 <fb:control>
19                   <Input id="idFilterCarnname" placeholder="{i18n>filterCarname}" liveChange="onFilterChange"/>
20                 </fb:control>
21               </fb:FilterGroupItem>
22             </fb:filterGroupItems>
23           </fb:FilterBar>
24         </f:content>
25       </f:DynamicPageHeader>
26     </f:header>
```

Figure 231: Add an Input Control

9. Declare a member variable with the name **oFilterBar** in the `onInit` function of the Overview controller and assign a reference to the `sap.ui.comp.filter.FilterBar` control. Declare another member with the name **oCarIdIF** and assign the reference to the UI control with ID `idFilterCarrid`. Declare a member variable with the name **oCarNameIF** and assign a reference to the UI control with the ID `idFilterCarrname`.
  10. Implement the function `onClear` in the controller of the `Overview.controller.js` as shown.

Figure 233: Function onClear

**11. Implement a function `onFilterChange` to fire the `onFilterChange` event.**



The screenshot shows a code editor with the file `Overview.controller.js` open. The code is a JavaScript file that extends a base controller. The `onFilterChange` function is implemented as follows:

```

1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
9     ...
112     onFilterChange: function (oEvent) {
113       this.oFilterBar.fireFilterChange(oEvent);
114     }
115   });
116 });
117 });

```

Figure 234: Implement Function `onFilterChange`

**12. Implement the function `onReset` and reset the applied filters on the table.**



The screenshot shows a code editor with the file `Overview.controller.js` open. The code is a JavaScript file that extends a base controller. The `onReset` function is implemented as follows:

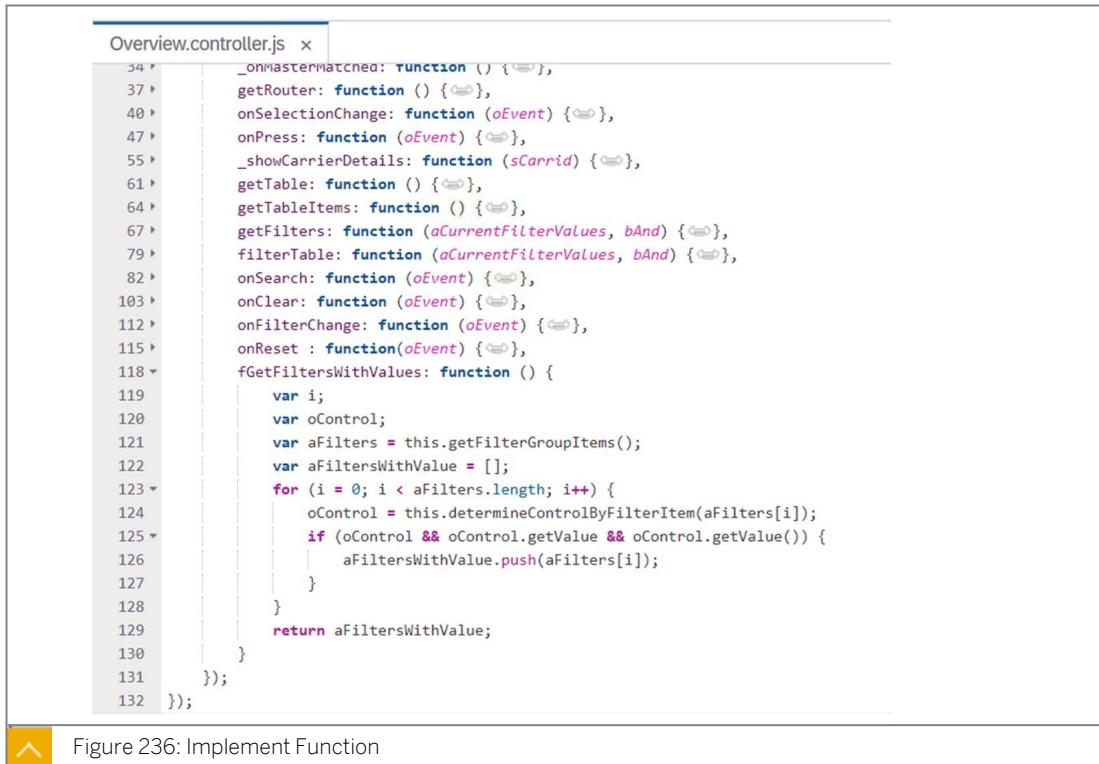
```

1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5 ], function (Controller, Filter, FilterOperator) {
6   "use strict";
7
8   return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
9     ...
112     onReset: function(oEvent) {
113       this.getTableItems().filter([]);
114     }
115   });
116 });
117 ...
118 });
119 });
120 });
121 ...

```

Figure 235: Implement Function `onReset`

**13. Implement a function with the name `fGetFiltersWithValues` as shown in the following figure, and assign the function as a callback for the `GetFiltersWithValues` event in the `onInit` function. Use the `registerGetFiltersWithValues` function on the `FilterBar` control.**



The screenshot shows the `Overview.controller.js` file in a code editor. The `fGetFiltersWithValues` function is being implemented. It iterates through an array of filters, checks if each filter has a value, and if so, adds it to a new array. Finally, it returns the array of filters with values.

```

54 *     _onMasterSwitched: function () { },
55 *     _showCarrierDetails: function (sCarrid) { },
56 *     getRouter: function () { },
57 *     onSelectionChange: function (oEvent) { },
58 *     onPress: function (oEvent) { },
59 *     _showCarrierDetails: function (sCarrid) { },
60 *     getTable: function () { },
61 *     getTableItems: function () { },
62 *     getFilters: function (aCurrentFilterValues, bAnd) { },
63 *     filterTable: function (aCurrentFilterValues, bAnd) { },
64 *     onSearch: function (oEvent) { },
65 *     onClear: function (oEvent) { },
66 *     onFilterChange: function (oEvent) { },
67 *     onReset: function (oEvent) { },
68 *     fGetFiltersWithValues: function () {
69 *         var i;
70 *         var oControl;
71 *         var aFilters = this.getFilterGroupItems();
72 *         var aFiltersWithValue = [];
73 *         for (i = 0; i < aFilters.length; i++) {
74 *             oControl = this.determineControlByFilterItem(aFilters[i]);
75 *             if (oControl && oControl.getValue() && oControl.getValue()) {
76 *                 aFiltersWithValue.push(aFilters[i]);
77 *             }
78 *         }
79 *         return aFiltersWithValue;
80 *     };
81 * },
82 * );
83 * );

```

Figure 236: Implement Function

#### 14. Extend the `onSearch` function as shown and save the changes.



The screenshot shows the `Overview.controller.js` file in a code editor. The `onSearch` function is being extended. It first initializes `aCurrentFilterValues` and `aKeys` arrays. Then, it checks if the search query is provided. If so, it adds the query to both `aCurrentFilterValues` and `aKeys`. If a selection set is provided, it adds the carrier ID and name to both arrays. Finally, it calls the `filterTable` function with the updated filter values and a boolean indicating if the search is case-sensitive.

```

55 *     _showCarrierDetails: function (sCarrid) { },
56 *     getTable: function () { },
57 *     getTableItems: function () { },
58 *     getFilters: function (aCurrentFilterValues, bAnd) { },
59 *     filterTable: function (aCurrentFilterValues, bAnd) { },
60 *     onSearch: function (oEvent) {
61 *         var aCurrentFilterValues = [];
62 *         this.aKeys = [];
63 *         if(oEvent.getParameter("query")) {
64 *             var sQuery = oEvent.getParameter("query");
65 *             this.aKeys = ["Carrid", "Carrname"];
66 *             aCurrentFilterValues.push(sQuery);
67 *             aCurrentFilterValues.push(sQuery);
68 *             this.filterTable(aCurrentFilterValues, false) ;
69 *         } else if(oEvent.getParameter("selectionSet")) {
70 *             if(this.oCarIdIF.getValue() !== "") {
71 *                 this.aKeys.push("Carrid");
72 *                 aCurrentFilterValues.push(this.oCarIdIF.getValue());
73 *             }
74 *             if(this.oCarNameIF.getValue() !== "") {
75 *                 this.aKeys.push("Carrname");
76 *                 aCurrentFilterValues.push(this.oCarNameIF.getValue());
77 *             }
78 *             this.filterTable(aCurrentFilterValues, true);
79 *         }
80 *     },
81 * );
82 * );

```

Figure 238: Extend the onSearch Function

#### 15. Test your application.

##### Task 2: Show the Number of Carriers Listed in the Carrier Table.

- Add an event handler to the carrier table located in the `Overview.view.xml` for the event `updateFinished`. Assign an event handler with the name `onUpdateFinished`.

2. Open the controller implementation of the Overview and add the following implementation of the onUpdateFinished function.



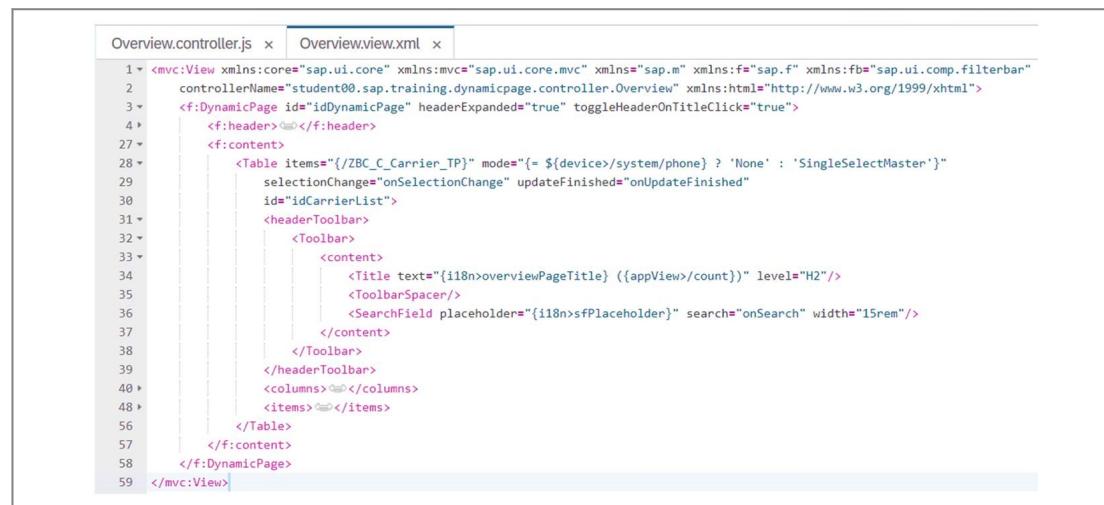
```

Overview.controller.js x
4      sap/ui/model/Filteroperator
5  ], function (Controller, Filter, Filteroperator) {
6      "use strict";
7
8      return Controller.extend("student00.sap.training.dynamicpage.controller.Overview",
9          {
10             onInit: function () { },
11             onBypassed: function () { },
12             _onMasterMatched: function () { },
13             getRouter: function () { },
14             onSelectionChange: function (oEvent) { },
15             onPress: function (oEvent) { },
16             _showCarrierDetails: function (sCarrid) { },
17             getTable: function () { },
18             getTableItems: function () { },
19             getFilters: function (aCurrentFilterValues, bAnd) { },
20             filterTable: function (aCurrentFilterValues, bAnd) { },
21             onSearch: function (oEvent) { },
22             onClear: function (oEvent) { },
23             onFilterChange: function (oEvent) { },
24             onReset: function (oEvent) { },
25             fGetFiltersWithValues: function () { },
26             onUpdateFinished: function(oEvent) {
27                 if(this.getTableItems().isLengthFinal()) {
28                     var iCount = oEvent.getParameter("total");
29                     var oJSONModel = this.getOwnerComponent().getModel("appView");
30                     oJSONModel.setProperty("/count", iCount);
31                     this.getOwnerComponent().setModel(oJSONModel,"appView");
32                 }
33             }
34         }
35     }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }

```

Figure 240: onUpdateFinished Implementation

3. Go back to the Overview.view.xml and update the binding of property text from the sap.m.Title control as shown.



```

Overview.controller.js x Overview.view.xml x
1  <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:fb="sap.ui.comp.filterbar"
2      controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3      <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4          <f:header></f:header>
5          <f:content>
6              <Table items="{/ZBC_C_Carrier_TP}" mode="{ ${device}/system/phone } ? 'None' : 'SingleSelectMaster'" 
7                  selectionChange="onSelectionChange" updateFinished="onUpdateFinished"
8                  id="idCarrierList">
9                  <headerToolbar>
10                     <Toolbar>
11                         <content>
12                             <Title text="{i18n>overviewPageTitle} ({appView>/count})" level="H2"/>
13                             <ToolbarSpacer/>
14                             <SearchField placeholder="{i18n>sfPlaceholder}" search="onSearch" width="15rem"/>
15                         </content>
16                     </Toolbar>
17                 </headerToolbar>
18                 <columns></columns>
19                 <items></items>
20             </Table>
21         </f:content>
22     </f:DynamicPage>
23 </mvc:View>

```

Figure 241: Update the Binding of Property Text

4. Test your application.

### Task 3: Implement Sorting and Grouping Functionality

An important aspect of implementing the list report floorplan is to provide a capability to sort and group the data in the list. In the next task you add basic and simple sorting and grouping capability to your list report.

1. Replace the sap.m.Toolbar control in the Overview.view.xml file and use a sap.m.OverflowToolbar control instead.

2. Add two sap.m.OverflowToolbarButton controls to the sap.m.OverflowToolbar control located in the *headerToolbar* aggregation of the sap.m.Table control in the Overview. Use the following attribute and values.

Table 70: Attributes of the first sap.m.OverflowToolbarButton Control

Attribute	Value
tooltip	Sort
press	onOpenViewSettings
type	Transparent
icon	sap-icon://sort

Table 71: Attributes of the second sap.m.OverflowToolbarButton Control

Attribute	Value
tooltip	Group
press	onOpenViewSettings
type	Transparent
icon	sap-icon://group-2



## Note:

Include the tooltip attribute in the i18n property file. Use the key **btnSort** for the sort button tooltip text and the key **btnGroup** for the group button. Use the i18n editor of the SAP Web IDE as shown in previous steps.

3. Implement a dialog where the user is able to select if the content of the table should be sorted and/or grouped by the ID or the name of a carrier. Implement the dialog as a fragment. Add the implementation in a file with the name **ViewSettingsDialog.fragment.xml**. The file should be located in the view folder of your project. Use the sap.m.ViewSettingsDialog implementation of SAPUI5. Add the following implementation to the **ViewSettingsDialog.fragment.xml** file.

```

ViewSettingsDialog.fragment.xml x
1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <ViewSettingsDialog id="viewSettingsDialog" confirm="onConfirmViewSettingsDialog">
3     <sortItems>
4       <ViewSettingsItem text="{i18n>filterCarId}" key="Carrid"/>
5       <ViewSettingsItem text="{i18n>filterCarname}" key="Carrname"/>
6     </sortItems>
7     <groupItems>
8       <ViewSettingsItem text="{i18n>filterCarId}" key="Carrid"/>
9       <ViewSettingsItem text="{i18n>filterCarname}" key="Carrname"/>
10    </groupItems>
11  </ViewSettingsDialog>
12 </core:FragmentDefinition>

```

Figure 245: Implementation Fragment

4. Implement the `onOpenViewSettings` function inside the controller of the Overview in the figure.

```
Overview.controller.js x
80  ...
81  ...
82  ...
83  ...
84  ...
85  ...
86  ...
87  ...
88  ...
89  ...
90  ...
91  ...
92  ...
93  ...
94  ...
95  ...
96  ...
97  ...
98  ...
99  ...
100 ...
101 ...
102 ...
103 ...
104 ...
105 ...
106 ...
107 ...
108 ...
109 ...
110 ...
111 ...
112 ...
113 ...
114 ...
115 ...
116 ...
117 ...
118 ...
119 ...
120 ...
121 ...
122 ...
123 ...
124 ...
125 ...
126 ...
127 ...
128 ...
129 ...
130 ...
131 ...
132 ...
133 ...
134 ...
135 ...
136 ...
137 ...
138 ...
139 ...
140 ...
141 ...
142 ...
143 ...
144 ...
145 ...
146 ...
147 ...
148 ...
149 ...
150 ...
151 ...
152 ...
153 ...);
```

Figure 249: Function `onOpenViewSettings`

5. Implement the function `onConfirmViewSettingsDialog` in the controller of the Overview as shown.

```
* Overview.controller.js x
111 ...
112 ...
113 ...
114 ...
115 ...
116 ...
117 ...
118 ...
119 ...
120 ...
121 ...
122 ...
123 ...
124 ...
125 ...
126 ...
127 ...
128 ...
129 ...
130 ...
131 ...
132 ...
133 ...
134 ...
135 ...
136 ...
137 ...
138 ...
139 ...
140 ...
141 ...
142 ...
143 ...
144 ...
145 ...
146 ...
147 ...
148 ...
149 ...
150 ...
151 ...
152 ...
153 ...
154 ...
155 ...
156 ...
157 ...
158 ...
159 ...
160 ...
161 ...
162 ...
163 ...
164 ...
165 ...
166 ...
167 ...
168 ...
169 ...
170 ...
171 ...);
```

Figure 250: Function `onConfirmViewSettingsDialog`

6. Test your application.

# Implement a List Report



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business Example

In this exercise you learn how to extend a standard SAP Fiori application to create a list report floorplan.

This exercise uses the `flexiblecolumnlayout` project created in the previous exercise..

### Task 1: Create a New Project

Export the project `flexiblecolumnlayout` that you created in the previous project and import the project with the name `listreport` into the workspace of your *SAP Web IDE Full-Stack*.

1. Log on to the SAP Web IDE Full-Stack, and export the project `flexiblecolumnlayout` that you created in one of the previous exercises.
  - a) Select the project `flexiblecolumnlayout` in the workspace and choose *Export* from the context menu.
  - b) Find the file downloaded in the footer area of your browser.



2. Import the content of the file `flexiblecolumnlayout.zip` as a project with the name `listreport`.
  - a) In the workspace folder, choose *Import* → *File or Project* from the context menu.
  - b) Use the *Browse* button to choose the exported file and change the file name in the *Import* field to `/listreport` and choose *OK*.
  - c) On successful import find the newly imported project with the given name.
3. Add the attribute `id` to the table control in the Overview-view and assign the value `idCarrierList` to the newly added attribute. Implement a search functionality to search the table of carriers, as described in the SAP Fiori guidelines. Add an `sap.m.SearchField`

control and place the control in the `sap.m.Toolbar`. Add an `sap.m.ToolbarSpacer` control between the existing `sap.m.Title` control and the newly added `sap.m.SearchField` control. Implement the `onSearch` function. Use the listed properties as attributes for the `SearchField` control. The search functionality returns a list of carriers where the `carrid` or the `carrname` matches the specified value.

Table 63: SearchField Control Properties

Attribute	Value
placeholder	Filter
search	onSearch
width	15rem



Note:

Implement the search functionality in a reusable way as we use the same search and filter aspects in a later step.



Note:

Use the i18n model for the placeholder text. Add the value **Filter** with the key **sfPlaceholder** and type **#XTXT** into the `i18n.property` file.

- a) Open the file `Overview.view.xml` and add an `sap.m.ToolbarSpacer` control immediately after the `sap.m.Title` control in the `sap.m.Toolbar`.

```

*Overview.view.xml x
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2   controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3   <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4     <f:content>
5       <Table items="{/ZBC_C_Carrier_TP}" mode="={ ${device}/system/phone } ? 'None' : 'SingleSelectMaster'" 
6         selectionChange="onSelectionChange" id="idCarrierList">
7           <headerToolbar>
8             <Toolbar>
9               <content>
10                 <Title text="{i18n>overviewPageTitle}" level="H2"/>
11                 <ToolbarSpacer/>
12               </content>
13             </Toolbar>
14           </headerToolbar>
15         </Table>
16       <!-- Other content blocks -->
17     </f:content>
18   </f:DynamicPage>
19 </mvc:View>

```

Figure 209: Add an `sap.m.ToolbarSpacer`

- b) Add an `sap.m.SearchField` to the existing `sap.m.Toolbar` control and insert the control immediately after the `sap.m.ToolbarSpacer` control. Add the listed attributes to the control and assign the given values for the attribute search and width. Assign an empty string to the placeholder attribute.



Figure 210: Placeholder Attribute

- c) Put the cursor in the empty string at the level of the placeholder attribute and choose *Create i18n String* from the context menu.

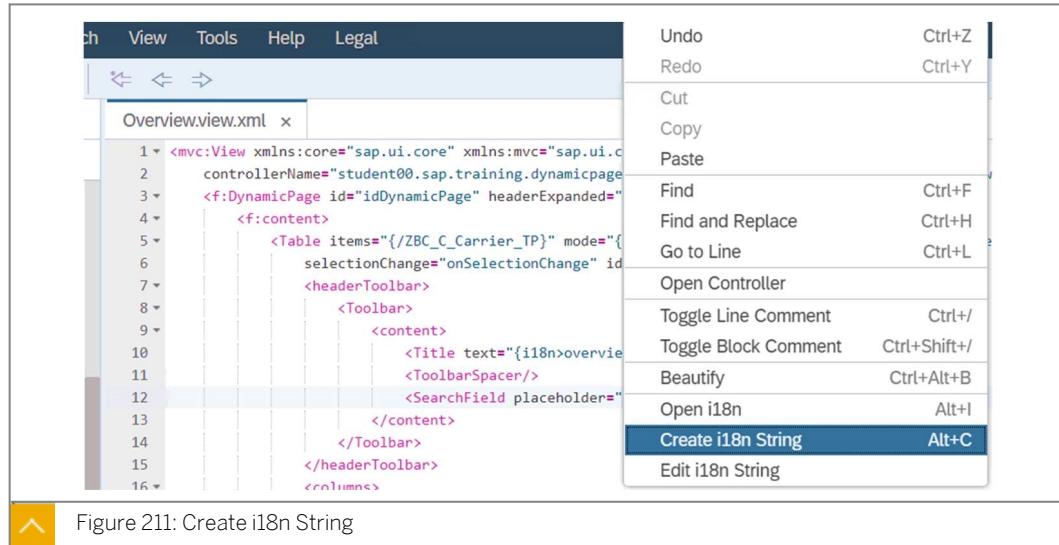


Figure 211: Create i18n String

- d) In the new string dialog, enter **sfPlaceholder** in the *String Key* field, **Filter** in the *Value* field and choose the entry **#XTXT** from the *Type* list. Choose **OK** to confirm the dialog.



Figure 212: Enter Details

- e) Check your `sap.m.SearchField` control appears as shown.



Figure 213: sap.m.SearchField control

- f) Save your work.
- g) Open the file `Overview.controller.js` located in the controller folder of your project.
- h) Add a reference to the `sap.ui.model.Filter` and to `sap.ui.model.FilterOperator`. Follow the general naming conventions for the Asynchronous Module Definition (AMD) syntax.



Figure 214: Add References

- i) Implement a function named `getTable` that returns the reference to the `sap.m.Table` control with the ID `idCarrierList`.

- j) Implement a function `getTableItems`. The function returns the items aggregation of the table returned by the `getTable` function.

■ Overview.controller.js x

```
8 *   return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
9 *     |   onInit: function () {↳},
27
28 *     |   onBypassed: function () {↳},
31
32 *     |   _onMasterMatched: function () {↳},
35
36 *     |   getRouter: function () {↳},
39
40 *     |   onSelectionChange: function (oEvent) {↳},
47
48 *     |   onPress: function (oEvent) {↳},
56
57 *     |   _showCarrierDetails: function (sCarrid) {↳},
63
64 *     |   getTable: function () {
65 *       |       return this.getView().ById("idCarrierList");
66 *     },
67 *     |   getTableItems: function () {
68 *       |       return this.getTable().getBinding("items");
69 *     },
70
```

- k) Implement a function named `getFilters`. The function gets two parameters. The first parameter is an array containing the filter values and the second holds the value to configure if the filters should be combined with an AND or OR instruction. Implement the function as shown below.



```

Overview.controller.js x
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/model/Filter",
4   "sap/ui/model/FilterOperator"
5  ], function (Controller, Filter, FilterOperator) {
6    "use strict";
7
8    return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
9      _onInit: function () {},
10     _onBypassed: function () {},
11     _onMasterMatched: function () {},
12     getRouter: function () {},
13     onSelectionChange: function (oEvent) {},
14     onPress: function (oEvent) {},
15     _showCarrierDetails: function (sCarrid) {},
16     getTable: function () {},
17     getTableItems: function () {},
18     getFilters: function (aCurrentFilterValues, bAnd) {
19       this.aFilters = [];
20       this.aFilters = this.aKeys.map(function (sCriteria, i) {
21         return new Filter(sCriteria, FilterOperator.Contains, aCurrentFilterValues[i]);
22       });
23       return new Filter({
24         filters: this.aFilters,
25         and: bAnd
26       });
27     },
28   },
29   {
30     _onMasterMatched: function () {}
31   }
32 );
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55     _showCarrierDetails: function (sCarrid) {},
56     getTable: function () {},
57     getTableItems: function () {},
58     getFilters: function (aCurrentFilterValues, bAnd) {
59       this.aFilters = [];
60       this.aFilters = this.aKeys.map(function (sCriteria, i) {
61         return new Filter(sCriteria, FilterOperator.Contains, aCurrentFilterValues[i]);
62       });
63       return new Filter({
64         filters: this.aFilters,
65         and: bAnd
66       });
67     },
68   },
69   {
70     _onMasterMatched: function () {}
71   }
72 );
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
626
627
627
628
629
629
630
631
632
633
634
635
635
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
15
```



```

*Overview.controller.js x
4  sap/ui/model/filteroperator
5  ], function (Controller, Filter, FilterOperator) {
6    "use strict";
7
8    return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
9      onInit: function () {},
10     onBypassed: function () {},
11     _onMasterMatched: function () {},
12     getRouter: function () {},
13     onSelectionChange: function (oEvent) {},
14     onPress: function (oEvent) {},
15     _showCarrierDetails: function (sCarrid) {},
16     getTable: function () {},
17     getTableItems: function () {},
18     getFilters: function (aCurrentFilterValues, bAnd) {},
19     filterTable: function (aCurrentFilterValues, bAnd) {},
20     onSearch: function (oEvent) {
21       var aCurrentFilterValues = [];
22
23       var sQuery = oEvent.getParameter("query");
24       this.aKeys = ["Carrid", "Carrname"];
25       aCurrentFilterValues.push(sQuery);
26       aCurrentFilterValues.push(sQuery);
27       this.filterTable(aCurrentFilterValues, false);
28     }
29   });
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89

```

Figure 219: Function onSearch

n) Save your work.

#### 4. Test the search implementation.

- a) Select the `index.html` file located in the `webapp` folder of your project and choose `Run → Run as Web Application` from the context menu.

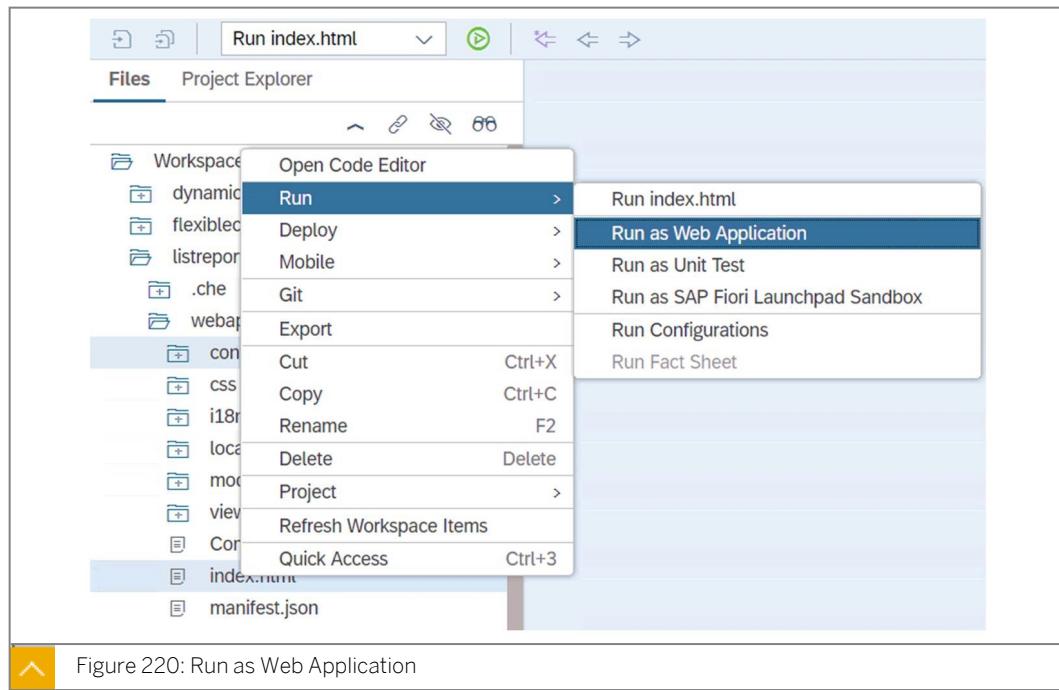
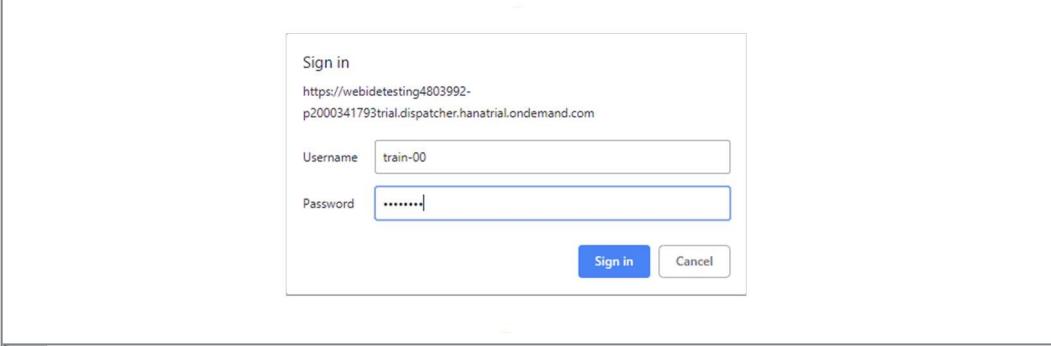


Figure 220: Run as Web Application

- b) If prompted to enter your front-end-server credentials, use `Train-##`, where `##` is your user group number with your password and choose `Sign In`.



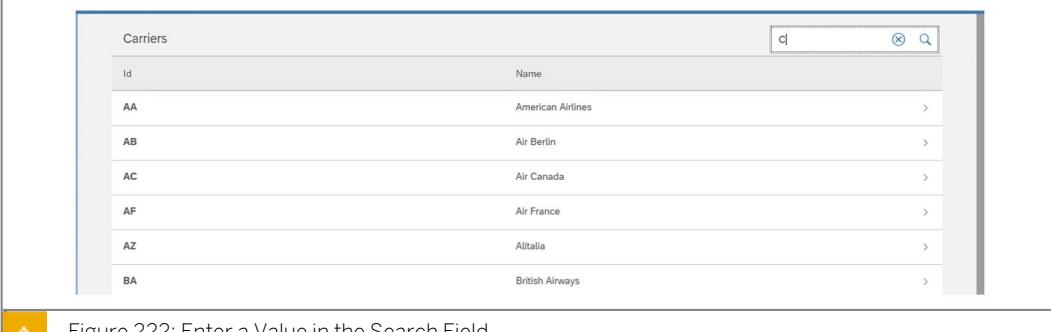
Sign in  
<https://webidetesting4803992-p2000341793trial.dispatcher.hanatrial.ondemand.com>

Username: train-00  
 Password: .....|

Sign in Cancel

Figure 221: Sign In

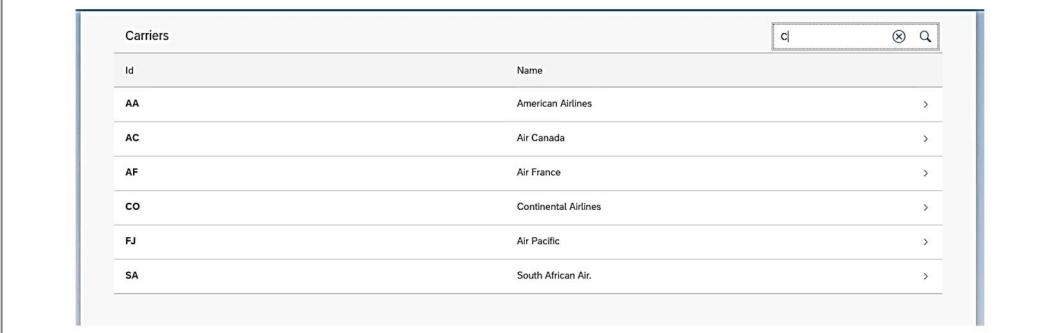
c) Enter a value in the search field and press *Enter*.



Carriers	
Id	Name
AA	American Airlines
AB	Air Berlin
AC	Air Canada
AF	Air France
AZ	Alitalia
BA	British Airways

Figure 222: Enter a Value in the Search Field

d) After the filter request is processed, find the filtered list of carriers.



Carriers	
Id	Name
AA	American Airlines
AC	Air Canada
AF	Air France
CO	Continental Airlines
FJ	Air Pacific
SA	South African Air.

Figure 223: List of Carriers

e) Close the browser tab.

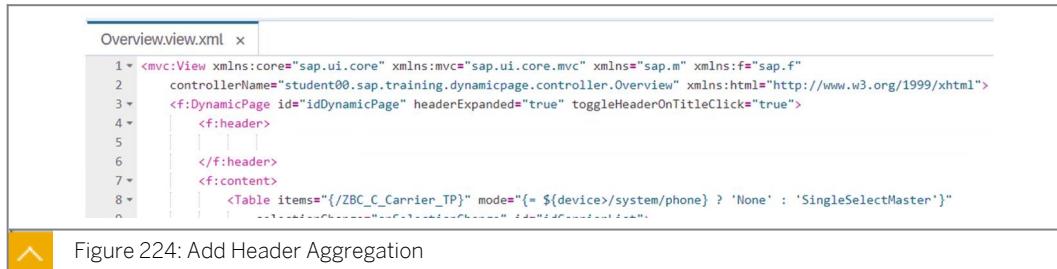
5. Add a UI control of type `sap.f.DynamicPageHeader` to the header aggregation of the existing `sap.f.DynamicPage` control located in the `Overview.view.xml` file. Assign the following attribute and value to the newly added `sap.f.DynamicPageHeader` control.

Table 64: `sap.f.DynamicPageHeader` attributes and values

Attribute	Value
<code>pinnable</code>	<code>false</code>

- a) Open the file `Overview.view.xml`. The file is located in the `view` folder of your project.

- b) Add the header aggregation to the existing `sap.f.DynamicPage` control.



```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2 controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3 <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4   <f:header>
5   </f:header>
6   <f:content>
7     <Table items="/ZBC_C_Carrier_TP" mode="${device>/system/phone} ? 'None' : 'SingleSelectMaster'">
8   </f:content>
9 </f:DynamicPage>
10 </mvc:View>

```

Figure 224: Add Header Aggregation

- c) Add a control of type `sap.f.DynamicPageHeader` to the header aggregation with the attribute and value listed in the table.



```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2 controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3 <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4   <f:header>
5     <DynamicPageHeader pinnable="false">
6     </DynamicPageHeader>
7   </f:header>
8   <f:content>
9     <Table items="/ZBC_C_Carrier_TP" mode="${device>/system/phone} ? 'None' : 'SingleSelectMaster'">
10 </f:content>
11 </f:DynamicPage>
12 </mvc:View>

```

Figure 225: Add to Header Aggregation

6. Add a UI control of type `sap.ui.comp.filterbar.FilterBar` to the content aggregation of the `sap.f.DynamicPageHeader` control. Use an XML namespace with alias `fb` to assign the SAPUI5 `sap.ui.comp.filterbar`. Add the listed attribute and values to the newly added `sap.ui.comp.filterbar.FilterBar` control.

Table 65: `sap.ui.comp.filterbar.FilterBar` control attribute and values

Attribute	Value
id	idFilterBar
reset	onReset
search	onSearch
clear	onClear
cancel	onCancel
filtersDialogClose	onFiltersDialogClose
useToolbar	true
showGoOnFB	true
showRestoreButton	true
showClearButton	true

- a) Add the XML namespace alias `fb` to the `View` control and assign the SAPUI5 namespace `sap.ui.comp.filterbar`.



Figure 226: fb Filterbar

```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2   xmlns:fb="sap.ui.comp.filterbar"
3   controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
4 <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">

```

- b) Add the content aggregation to the newly added sap.f.DynamicPageHeader control.



Figure 227: Add content aggregation to the Dynamic Page Header

```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2   xmlns:fb="sap.ui.comp.filterbar"
3   controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
4 <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
5   <f:header>
6     <f:DynamicPageHeader pinnable="false">
7       <f:content>
8       </f:content>
9     </f:DynamicPageHeader>
10   </f:header>
11 </f:DynamicPage>

```

- c) Add the sap.ui.comp.filterbar.FilterBar control to the content aggregation of the sap.f.DynamicPageHeader control and add the attribute and values to the UI control.

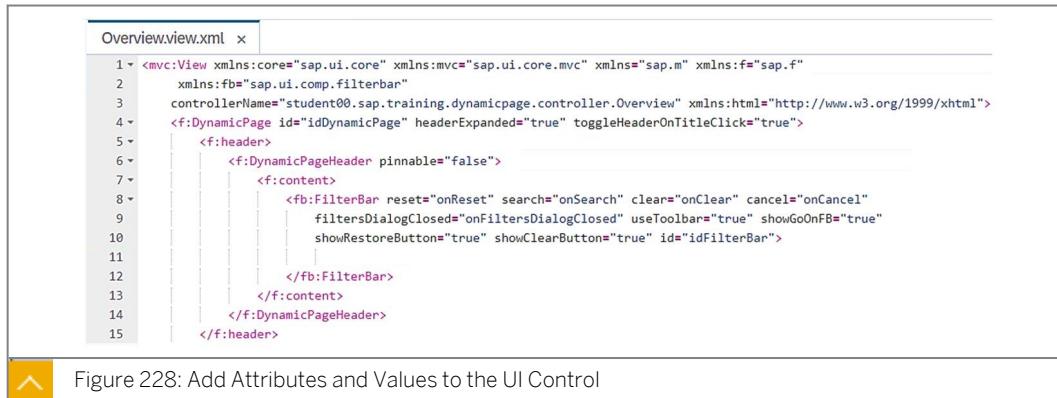


Figure 228: Add Attributes and Values to the UI Control

```

1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2   xmlns:fb="sap.ui.comp.filterbar"
3   controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
4 <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
5   <f:header>
6     <f:DynamicPageHeader pinnable="false">
7       <f:content>
8         <fb:FilterBar reset="onReset" search="onSearch" clear="onClear" cancel="onCancel"
9           filtersDialogClosed="onFiltersDialogClosed" useToolbar="true" showGoOnFB="true"
10          showRestoreButton="true" showClearButton="true" id="idFilterBar">
11        </fb:FilterBar>
12      </f:content>
13    </f:DynamicPageHeader>
14  </f:header>
15 </f:DynamicPage>

```

- d) Save your changes.

7. Add two sap.ui.comp.filterbar.FilterGroupItem controls to the *filterItems* aggregation of the sap.ui.comp.filterbar.FilterBar with the following attributes and values.

Table 66: sap.ui.comp.filterbar.FilterGroupItem Control Attributes and Values

Attribute	Value
groupName	G1
visibleInFilterBar	true
groupTitle	carridGroup
name	A
label	Carrier



## Note:

Use the i18n ResourceModel and add the key `filterCarId` to the i18n.properties file. Assign the value **Carrier** to the key and use this key instead of the hardwired text.

Table 67: sap.ui.comp.filterbar.FilterGroupItem Control Attributes and Values

Attribute	Value
groupName	G2
visibleInFilterBar	true
groupTitle	carrnameGroup
name	B
label	Carrier name

- a) Add `filterGroupItems` aggregation to the `sap.ui.comp.filterbar.FilterBar` control.



Figure 229: Add filterGroupItems

```

Overview.view.xml x
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:fb="sap.ui.comp.filterbar"
2 controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3 <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4   <f:header>
5     <f:DynamicPageHeader pinnable="false">
6       <f:content>
7         <fb:FilterBar reset="onReset" search="onSearch" clear="onClear" cancel="onCancel" filtersDialogClosed="onFiltersDialogClosed"
8           useToolbar="true" showGoOnFB="true" showRestoreButton="true" showClearButton="true" id="idFilterBar">
9           <fb:filterGroupItems>
10          </fb:filterGroupItems>
11        </fb:FilterBar>
12      </f:content>
13    </f:DynamicPageHeader>
14  </f:header>
15

```

- b) Add two `sap.ui.comp.filterbar.FilterGroupItem` controls to the `filterGroupItems` aggregation. Use the i18n tools to add the label attribute (as shown in the previous task step 3c and 3d).



Figure 230: Add Controls

```

Overview.view.xml x
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:fb="sap.ui.comp.filterbar"
2 controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3 <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4   <f:header>
5     <f:DynamicPageHeader pinnable="false">
6       <f:content>
7         <fb:FilterBar reset="onReset" search="onSearch" clear="onClear" cancel="onCancel" filtersDialogClosed="onFiltersDialogClosed"
8           useToolbar="true" showGoOnFB="true" showRestoreButton="true" showClearButton="true" id="idFilterBar">
9           <fb:filterGroupItems>
10          <fb:FilterGroupItem groupName="G1" visibleInFilterBar="true" groupTitle="carridGroup" name="A"
11            label="{i18n>filterCarId}">
12          </fb:FilterGroupItem>
13          <fb:FilterGroupItem groupName="G2" visibleInFilterBar="true" groupTitle="carrnameGroup" name="B"
14            label="{i18n>filterCarrname}">
15          </fb:FilterGroupItem>
16        </fb:filterGroupItems>
17      </fb:FilterBar>
18    </f:content>
19  </f:DynamicPageHeader>
20
21

```

- c) Save your work.

8. Add a control of type `sap.m.Input` to each control aggregation of the `sap.ui.comp.filter.FilterGroupItem` controls. Assign the listed attributes and values to the controls.

Table 68: `sap.m.Input` Control 1 Attributes and Values

Attribute	Value
placeholder	{i18n>filterCarId}
liveChange	onFilterChange
id	idFilterCarrid

Table 69: `sap.m.Input` Control 2 Attributes and Values

Attribute	Value
placeholder	{i18n>filterCarrname}
liveChange	onFilterChange
ID	idFilterCarrname

Add a `sap.m.Input` control to each of the `sap.ui.comp.filter.FilterGroupItem` controls and assign the listed attribute and values.

```

Overview.view.xml
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:fb="sap.ui.comp.filterbar"
2   controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3   <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4     <f:header>
5       <f:DynamicPageHeader pinnable="false">
6         <f:content>
7           <fb:FilterBar reset="onReset" search="onSearch" clear="onClear" cancel="onCancel" filtersDialogClosed="onFiltersDialogClosed"
8             useToolbars="true" showGoOnFB="true" showRestoreButton="true" showClearButton="true" id="idFilterBar">
9             <fb:filterGroupItems>
10               <fb:FilterGroupItem groupName="G1" visibleInFilterBar="true" groupTitle="carridGroup" name="A"
11                 labels="{i18n>filterCarId}">
12                 <fb:control>
13                   <Input id="idFilterCarrid" placeholder="{i18n>filterCarId}" liveChange="onFilterChange"/>
14                 </fb:control>
15               </fb:FilterGroupItem>
16               <fb:FilterGroupItem groupName="G2" visibleInFilterBar="true" groupTitle="carrnameGroup" name="B"
17                 labels="{i18n>filterCarrname}">
18                 <fb:control>
19                   <Input id="idFilterCarrname" placeholder="{i18n>filterCarrname}" liveChange="onFilterChange"/>
20                 </fb:control>
21               </fb:FilterGroupItem>
22             </fb:filterGroupItems>
23           </fb:FilterBar>
24         </f:content>
25       </f:DynamicPageHeader>
26     </f:header>

```

Figure 231: Add an Input Control

9. Declare a member variable with the name `oFilterBar` in the `onInit` function of the `Overview` controller and assign a reference to the `sap.ui.comp.filter.FilterBar` control. Declare another member with the name `oCarIdIF` and assign the reference to the UI control with ID `idFilterCarrid`. Declare a member variable with the name `oCarNameIF` and assign a reference to the UI control with the ID `idFilterCarrname`.
- Open the file `Overview.controller.js`.
  - Add the mentioned member variables and assign the references.

```

Overview.controller.js x
  8 *      return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
  9 *          onInit: function () {
10 *              this._oList = this.byId("idCarrierList");
11 *              this.getView().addEventDelegate({
12 *                  onBeforeFirstShow: function () {
13 *                      this.getOwnerComponent().oListSelector.setBoundMasterList(this._oList);
14 *                  }.bind(this)
15 *              });
16 *              this.getView().addEventDelegate({
17 *                  onBeforeFirstShow: function () {
18 *                      this.getOwnerComponent().oListSelector.setBoundMasterList(this._oList);
19 *                  }.bind(this)
20 *              });
21 *
22 *              this.getRouter().getRoute("Overview").attachPatternMatched(this._onMasterMatched, this);
23 *              this.getRouter().attachBypassed(this.onBypassed, this);
24 *
25 *              this.aKeys = ["Carrid", "Carname"];
26 *              this.oFilterBar = this.byId("idFilterBar");
27 *              this.oCarIdIF = this.byId("idFilterCarrid");
28 *              this.oCarNameIF = this.byId("idFilterCarname");
29 *          },

```

Figure 232: Add the Variables and Assign the References

10. Implement the function `onClear` in the controller of the `Overview.controller.js` as shown.

```

*Overview.controller.js x
  5 *  },
  6 *  "use strict";
  7 *
  8 *  return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
  9 *      onInit: function () { },
10 *      onBypassed: function () { },
11 *      _onMasterMatched: function () { },
12 *      getRouter: function () { },
13 *      onSelectionChange: function (oEvent) { },
14 *      onPress: function (oEvent) { },
15 *      _showCarrierDetails: function (sCarrid) { },
16 *      getTable: function () { },
17 *      getTableItems: function () { },
18 *      getFilters: function (aCurrentFilterValues, bAnd) { },
19 *      filterTable: function (aCurrentFilterValues, bAnd) { },
20 *      onSearch: function (oEvent) { },
21 *      onClear: function (oEvent) {
22 *          var oItems = this.oFilterBar.getAllFilterItems(true);
23 *          for (var i = 0; i < oItems.length; i++) {
24 *              var oControl = this.oFilterBar.determineControlByFilterItem(oItems[i]);
25 *              if (oControl) {
26 *                  oControl.setValue("");
27 *              }
28 *          }
29 *      },
30 *  });
31 *
32 *  });
33 *
34 *  });
35 *
36 *  });
37 *
38 *  });
39 *
40 *  });
41 *
42 *  });
43 *
44 *  });
45 *
46 *  });
47 *
48 *  });
49 *
50 *  });
51 *
52 *  });
53 *
54 *  });
55 *
56 *  });
57 *
58 *  });
59 *
60 *  });
61 *
62 *  });
63 *
64 *  });
65 *
66 *  });
67 *
68 *  });
69 *
70 *  });
71 *
72 *  });
73 *
74 *  });
75 *
76 *  });
77 *
78 *  });
79 *
80 *  });
81 *
82 *  });
83 *
84 *  });
85 *
86 *  });
87 *
88 *  });
89 *
90 *  });
91 *
92 *  });
93 *
94 *  });
95 *
96 *  });
97 *
98 *  });
99 *
100 *  });
101 *
102 *  });
103 *
104 *  });
105 *
106 *  });
107 *
108 *  });
109 *
110 *  });
111 *
112 *  });
113 *
114 *  });

```

Figure 233: Function `onClear`

11. Implement a function `onFilterChange` to fire the `onFilterChange` event.

```

Overview.controller.js x
1+ sap.ui.define([
2    "sap/ui/core/mvc/Controller",
3    "sap/ui/model/Filter",
4    "sap/ui/model/FilterOperator"
5+ ], function (Controller, Filter, FilterOperator) {
6    "use strict";
7
8+    return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
9+        onInit: function () { },
10+       onBypassed: function () { },
11+       _onMasterMatched: function () { },
12+       getRouter: function () { },
13+       onSelectionChange: function (oEvent) { },
14+       onPress: function (oEvent) { },
15+       _showCarrierDetails: function (sCarrid) { },
16+       getTable: function () { },
17+       getTableItems: function () { },
18+       getFilters: function (aCurrentFilterValues, bAnd) { },
19+       filterTable: function (aCurrentFilterValues, bAnd) { },
20+       onSearch: function (oEvent) { },
21+       onClear: function (oEvent) { },
22+       onFilterChange: function (oEvent) {
23+           this.oFilterBar.fireFilterChange(oEvent);
24+       }
25+    });
26+ });
27+
28+ });

```

Figure 234: Implement Function onFilterChange

12. Implement the function `onReset` and reset the applied filters on the table.

```

Overview.controller.js x
1+ sap.ui.define([
2    "sap/ui/core/mvc/Controller",
3    "sap/ui/model/Filter",
4    "sap/ui/model/FilterOperator"
5+ ], function (Controller, Filter, FilterOperator) {
6    "use strict";
7
8+    return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
9+        onInit: function () { },
10+       onBypassed: function () { },
11+       _onMasterMatched: function () { },
12+       getRouter: function () { },
13+       onSelectionChange: function (oEvent) { },
14+       onPress: function (oEvent) { },
15+       _showCarrierDetails: function (sCarrid) { },
16+       getTable: function () { },
17+       getTableItems: function () { },
18+       getFilters: function (aCurrentFilterValues, bAnd) { },
19+       filterTable: function (aCurrentFilterValues, bAnd) { },
20+       onSearch: function (oEvent) { },
21+       onClear: function (oEvent) { },
22+       onFilterChange: function (oEvent) { },
23+       onReset: function (oEvent) {
24+           this.getTableItems().filter([]);
25+       }
26+    });
27+
28+ });

```

Figure 235: Implement Function onReset

- a) Add the shown implementation of the function `onReset` to the controller implementation.

13. Implement a function with the name `fGetFiltersWithValues` as shown in the following figure, and assign the function as a callback for the `GetFiltersWithValues` event in the `onInit` function. Use the `registerGetFiltersWithValues` function on the `FilterBar` control.

Figure 236: Implement Function

- a) Add the shown implementation of the function `fGetFiltersWithValues` to the controller implementation.
  - b) Go to the `onInit` function and assign the function as a handler for the `GetFiltersWithValues` event using the `registerGetFiltersWithValues` function on the `filterbar` control.

```
*Overview.controller.js  x

1.sap.ui.define([
2.    "sap/ui/core/mvc/Controller",
3.    "sap/ui/model/Filter",
4.    "sap/ui/model/FilterOperator"
5.], function (Controller, Filter, FilterOperator) {
6.    "use strict";
7.
8.    return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {
9.        OnInit: function () {
10.            this._oList = this.byId("idCarrierList");
11.            this.getView().addEventDelegate( );
12.            this.getView().addEventDelegate( );
13.
14.            this.getRouter().getRoute("Overview").attachPatternMatched(this._onMasterMatched, this);
15.            this.getRouter().attachBypassed(this.onBypassed, this);
16.
17.
18.            this.aKeys = ["Carrid", "Carrname"];
19.            this.oFilterBar = this.byId("idfilterBar");
20.            this.oCarIdIF = this.byId("idfilterCarrid");
21.            this.oCarNameIF = this.byId("idfilterCarrname");
22.            this.oFilterBar.registerGetFiltersWithValues(this.fGetFiltersWithValues);
23.
24.
25.
26.
27.
28.
29.
30.    });
31.});
```

Figure 237: Assign as a Handler

14. Extend the `onSearch` function as shown and save the changes.

```

Overview.controller.js x
55  _showCarrierDetails: function (sCarrid) { },
56  getTable: function () { },
57  getTableItems: function () { },
58  getFilters: function (aCurrentFilterValues, bAnd) { },
59  filterTable: function (aCurrentFilterValues, bAnd) { },
60  onSearch: function (oEvent) {
61      var aCurrentFilterValues = [];
62      this.aKeys = [];
63      if(oEvent.getParameter("query")) {
64          var sQuery = oEvent.getParameter("query");
65          this.aKeys = ["Carrid", "Carrname"];
66          aCurrentFilterValues.push(sQuery);
67          aCurrentFilterValues.push(sQuery);
68          this.filterTable(aCurrentFilterValues, false) ;
69      } else if(oEvent.getParameter("selectionSet")) {
70          if(this.oCarIdIF.getValue() !== "") {
71              this.aKeys.push("Carrid");
72              aCurrentFilterValues.push(this.oCarIdIF.getValue());
73          }
74          if(this.oCarNameIF.getValue() !== "") {
75              this.aKeys.push("Carrname");
76              aCurrentFilterValues.push(this.oCarNameIF.getValue());
77          }
78          this.filterTable(aCurrentFilterValues, true);
79      }
80  },
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
},

```

Figure 238: Extend the onSearch Function

## 15. Test your application.

### Task 2: Show the Number of Carriers Listed in the Carrier Table.

1. Add an event handler to the carrier table located in the `Overview.view.xml` for the event `updateFinished`. Assign an event handler with the name `onUpdateFinished`.
  - a) Open the file `Overview.view.xml` located in the `view` folder.
  - b) Add the event handler `onUpdateFinished` to the event `updateFinished`.

```

Overview.view.xml x
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:fb="sap.ui.comp.filterbar"
2   controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3   <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4     <f:header>
5       <f:DynamicPageHeader pinnable="false">
6         <f:content></f:content>
7       </f:DynamicPageHeader>
8     </f:header>
9     <f:content>
10       <Table items="{/ZBC_C_Carrier_TP}" mode="={ ${device}/system/phone } ? 'None' : 'SingleSelectMaster'" 
11         selectionChange="onSelectionChange" updateFinished="onUpdateFinished"
12         id="idCarrierList">
13         <headerToolbar>
14           <Text text="Carriers" />
15         </headerToolbar>
16       </Table>
17     </f:content>
18   </f:DynamicPage>
19 </mvc:View>

```

Figure 239: Add Event Handler

2. Open the controller implementation of the `Overview` and add the following implementation of the `onUpdateFinished` function.

```
Overview.controller.js x
  4      sap/ui/model/filteroperator
  5  ], function (Controller, Filter, FilterOperator) {
  6      "use strict";
  7
  8  return Controller.extend("student00.sap.training.dynamicpage.controller.Overview",
  9  {
 10      _onInit: function () { },
 11      _onBypassed: function () { },
 12      _onMasterMatched: function () { },
 13      _getRouter: function () { },
 14      _onSelectionChange: function (oEvent) { },
 15      _onPress: function (oEvent) { },
 16      _showCarrierDetails: function (sCarrid) { },
 17      _getTable: function () { },
 18      _getTableItems: function () { },
 19      _getFilters: function (aCurrentFilterValues, bAnd) { },
 20      _filterTable: function (aCurrentFilterValues, bAnd) { },
 21      _onSearch: function (oEvent) { },
 22      _onClear: function (oEvent) { },
 23      _onFilterChange: function (oEvent) { },
 24      _onReset: function (oEvent) { },
 25      _fGetFiltersWithValues: function () { },
 26      _onUpdateFinished: function (oEvent) {
 27          if(this.getTableItems().isLengthFinal()) {
 28              var iCount = oEvent.getParameter("total");
 29              var oJSONModel = this.getOwnerComponent().getModel("appView");
 30              oJSONModel.setProperty("/count", iCount);
 31              this.getOwnerComponent().setModel(oJSONModel, "appView");
 32          }
 33      }
 34  });
 35
 36  }
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 579
 580
 581
 582
 583
 584
 585
 586
 587
 588
 589
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 779
 780
 781
 782
 783
 784
 785
 786
 787
 788
 789
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 879
 880
 881
 882
 883
 884
 885
 886
 887
 888
 889
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 899
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026
 1027
 1028
 1029
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1089
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1149
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1189
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1198
 1199
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1249
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1259
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1269
 1270
 1271
 1272
 1273
 1274
 1275
 1276
 1277
 1278
 1278
 1279
 1280
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1288
 1289
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1298
 1298
 1299
 1299
 1300
 1301
 1302
 1303
 1304
 1305
 1306
 1307
 1308
 1309
 1309
 1310
 1311
 1312
 1313
 1314
 1315
 1316
 1317
 1318
 1319
 1319
 1320
 1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328
 1329
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1339
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1349
 1349
 1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1359
 1360
 1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1369
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1388
 1389
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1398
 1399
 1399
 1400
 1401
 1402
 1403
 1404
 1405
 1406
 1407
 1408
 1409
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1419
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1429
 1429
 1430
 1431
 1432
 1433
 1434
 1435
 1436
 1437
 1438
 1439
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1449
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457
 1458
 1459
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1488
 1489
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1498
 1499
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1509
 1509
 1510
 1511
 1512
 1513
 1514
 1515
 1516
 1517
 1518
 1519
 1519
 1520
 1521
 1522
 1523
 1524
 1525
 1526
 1527
 1528
 1529
 1529
 1530
 1531
 1532
 1533
 1534
 1535
 1536
 1537
 1538
 1539
 1539
 1540
 1541
 1542
 1543
 1544
 1545
 1546
 1547
 1548
 1549
 1549
 1550
 1551
 1552
 1553
 1554
 1555
 1556
 1557
 1558
 1559
 1559
 1560
 1561
 1562
 1563
 1564
 1565
 1566
 1567
 1568
 1569
 1569
 1570
 1571
 1572
 1573
 1574
 1575
 1576
 1577
 1578
 1578
 1579
 1580
 1581
 1582
 1583
 1584
 1585
 1586
 1587
 1588
 1588
 1589
 1589
 1590
 1591
 1592
 1593
 1594
 1595
 1596
 1597
 1598
 1598
 1599
 1599
 1600
 1601
 1602
 1603
 1604
 1605
 1606
 1607
 1608
 1609
 1609
 1610
 1611
 1612
 1613
 1614
 1615
 1616
 1617
 1618
 1619
 1619
 1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1669
 1669
 1670
 1671
 1672
 1673
 1674
 1675
 1676
 1677
 1678
 1678
 1679
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1688
 1689
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1698
 1699
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1709
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1719
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1729
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779
 1779
 1780
 1781
 1782
 1783
 1784
 1785
 1786
 1787
 1788
 1788
 1789
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1798
 1799
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1819
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1829
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1839
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1888
 1889
 1889
 1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1898
 1899
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1919
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1929
 1929
 1930
 1931
 1932
 1933
 1934
 1935
 1936
 1937
 1938
 1939
 1939
 1940
 1941
 1942
 1943
 1944
 1945
 1946
 1947
 1948
 1949
 1949
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957
 1958
 1959
 1959
 1960
 1961
 1962
 1963
 1964
 1965
 1966
 1967
 1968
 1969
 1969
 1970
 1971
 1972
 1973
 1974
 1975
 1976
 1977
 1978
 1979
 1979
 1980
 1981
 1982
 1983
 1984
 1985
 1986
 1987
 1988
 1988
 1989
 1989
 1990
 1991
 1992
 1993
 1994
 1995
 1996
 1997
 1998
 1998
 1999
 1999
 2000
 2001
 2002
 2003
 2004
 2005
 2006
 2007
 2008
 2009
 2009
 2010
 2011
 2012
 2013
 2014
 2015
 2016
 2017
 2018
 2019
 2020
 2021
 2022
 2023
 2024
 2025
 2026
 2027
 2028
 2029
 2029
 2030
 2031
 2032
 2033
 2034
 2035
 2036
 2037
 2038
 2039
 
```

Figure 240: onUpdateFinished Implementation

- a) Open the file Overview.controller.js and implement the function onUpdateFinished as shown.

3. Go back to the Overview.view.xml and update the binding of property text from the sap.m.Title control as shown.

```
Overview.controller.js x Overview.view.xml x
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:fb="sap.ui.comp.filterbar"
2 controllerName="student00.sap.training.dynamicpage.controller.Overview" xmlns:html="http://www.w3.org/1999/xhtml">
3 <f:DynamicPage id="idDynamicPage" headerExpanded="true" toggleHeaderOnTitleClick="true">
4   <f:header></f:header>
5   <f:content>
6     <Table items="{/BC_C_Carrier_TP}" mode="{ ${device}/system/phone } ? 'None' : 'SingleSelectMaster'" 
7       selectionChange="onSelectionChange" updateFinished="onUpdateFinished"
8       id="idCarrierList">
9       <headerToolbar>
10         <Toolbar>
11           <content>
12             <Title text="{i18n>overviewPageTitle} ({appView>/count})" level="H2"/>
13             <ToolbarSpacer/>
14             <SearchField placeholder="{i18n>sfPlaceholder}" search="onSearch" width="15rem"/>
15           </content>
16         </Toolbar>
17       </headerToolbar>
18       <columns></columns>
19       <items></items>
20     <Table>
21       <f:content>
22         </f:content>
23       </f:DynamicPage>
24     </mvc:View>
```

Figure 241: Update the Binding of Property Text

4. Test your application.
    - a) To test your application run the `index.html` file as before.
    - b) Use the filter or the search functions for your table to show in the header-area entity-counter.

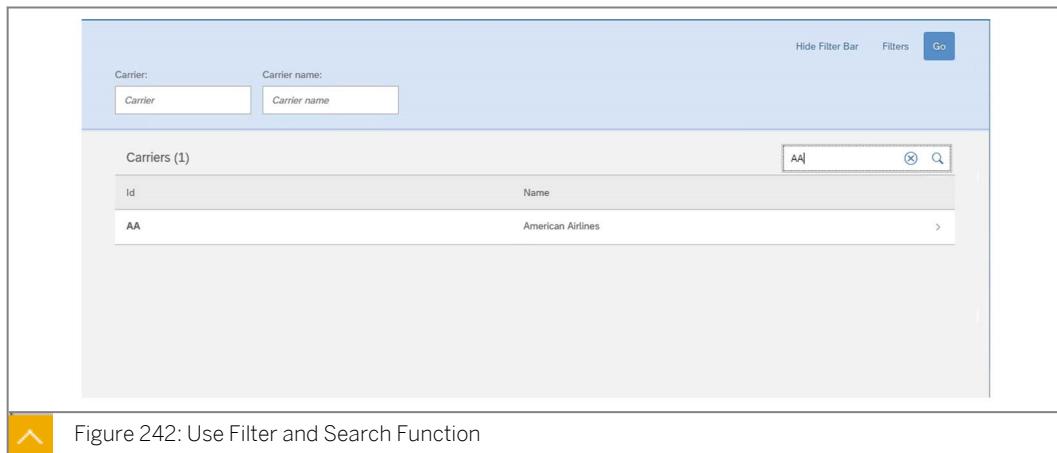


Figure 242: Use Filter and Search Function

c) Close your browser tab.

### Task 3: Implement Sorting and Grouping Functionality

An important aspect of implementing the list report floorplan is to provide a capability to sort and group the data in the list. In the next task you add basic and simple sorting and grouping capability to your list report.

1. Replace the `sap.m.Toolbar` control in the `Overview.view.xml` file and use a `sap.m.OverflowToolbar` control instead.
  - a) Open the file `Overview.view.xml` located in the `view` folder of your project.
  - b) Replace the `sap.m.Toolbar` with the `sap.m.OverflowToolbar` control.



Figure 243: Replace Toolbar

2. Add two `sap.m.OverflowToolbarButton` controls to the `sap.m.OverflowToolbar` control located in the `headerToolbar` aggregation of the `sap.m.Table` control in the `Overview`. Use the following attribute and values.

Table 70: Attributes of the first `sap.m.OverflowToolbarButton` Control

Attribute	Value
tooltip	Sort
press	onOpenViewSettings
type	Transparent
icon	sap-icon://sort

Table 71: Attributes of the second sap.m.OverflowToolbarButton Control

Attribute	Value
tooltip	Group
press	onOpenViewSettings
type	Transparent
icon	sap-icon://group-2



## Note:

Include the tooltip attribute in the *i18n* property file. Use the key **btnSort** for the sort button tooltip text and the key **btnGroup** for the group button. Use the *i18n* editor of the SAP Web IDE as shown in previous steps.

- a) Add two UI controls of type `sap.m.OverflowToolbarButton` to the content aggregation of `sap.m.OverflowToolbar` control that is located in the `headerToolbar` aggregation of `sap.m.Table`. Use the listed attributes.

```

<Table items="{/ZBC_C_Carrier_TP}" mode="{= ${device}/system/phone ? 'None' : 'SingleSelectMaster'}"
       selectionChange="onSelectionChange" updateFinished="onUpdateFinished" id="idCarrierList">
  <headerToolbar>
    <OverflowToolbar>
      <Title text="{i18n>overviewPageTitle} ({appView>/count})" level="H2"/>
      <ToolbarSpacer/>
      <SearchField placeholder="{i18n>sfPlaceholder}" search="onSearch" width="15rem"/>
      <OverflowToolbarButton press="onOpenViewSettings" icon="sap-icon://sort" type="Transparent" tooltip="{i18n>btnSort}" />
      <OverflowToolbarButton press="onOpenViewSettings" icon="sap-icon://group-2" type="Transparent" tooltip="{i18n>btnGrp}" />
    </OverflowToolbar>
  </headerToolbar>
</Table>

```

Figure 244: Add Toolbar Button Controls

- b) Use the *i18n* tools of the SAP Web IDE to add the *i18n* resource properties for both `tooltip` attributes.
3. Implement a dialog where the user is able to select if the content of the table should be sorted and/or grouped by the ID or the name of a carrier. Implement the dialog as a fragment. Add the implementation in a file with the name **ViewSettingsDialog.fragment.xml**. The file should be located in the view folder of your project. Use the `sap.m.ViewSettingsDialog` implementation of SAPUI5. Add the following implementation to the `ViewSettingsDialog.fragment.xml` file.



```

ViewSettingsDialog.fragment.xml x
1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <ViewSettingsDialog id="viewSettingsDialog" confirm="onConfirmViewSettingsDialog">
3     <sortItems>
4       <ViewSettingsItem text="{i18n>filterCarId}" key="Carrid"/>
5       <ViewSettingsItem text="{i18n>filterCarname}" key="Carrname"/>
6     </sortItems>
7     <groupItems>
8       <ViewSettingsItem text="{i18n>filterCarId}" key="Carrid"/>
9       <ViewSettingsItem text="{i18n>filterCarname}" key="Carrname"/>
10    </groupItems>
11  </ViewSettingsDialog>
12 </core:FragmentDefinition>

```

Figure 245: Implementation Fragment

- a) Select the view folder of your project and choose **New → File** from the context menu.

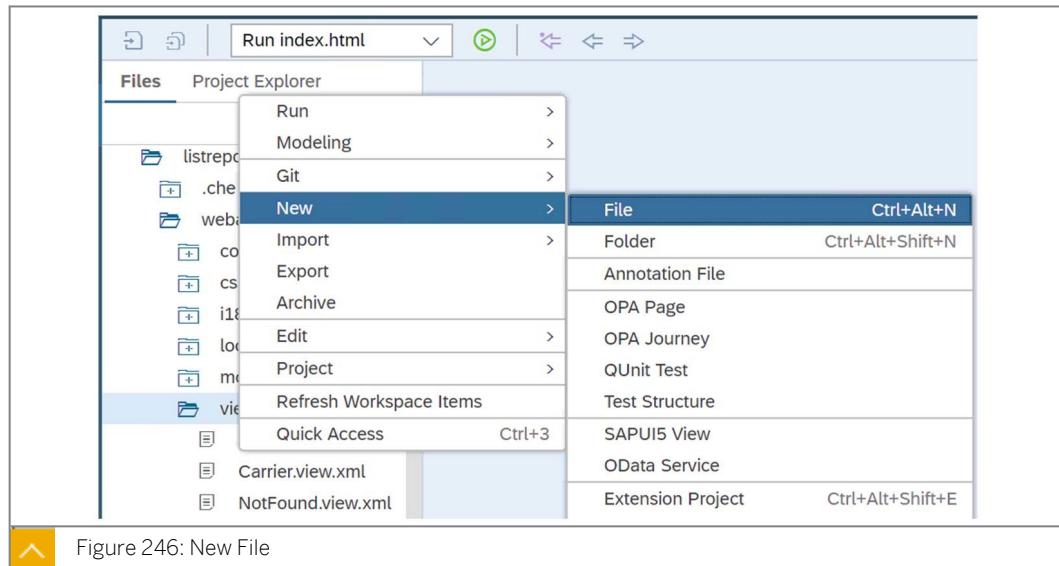


Figure 246: New File

- b) Enter the value **ViewSettingsDialog.fragment.xml** in the upcoming dialog and choose **OK**.

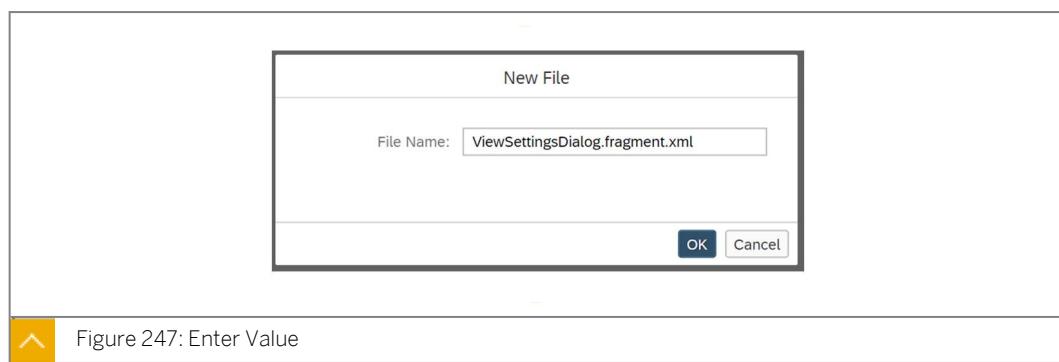


Figure 247: Enter Value

- c) Add the previously shown implementation to the newly created fragment.



ViewSettingsDialog.fragment.xml

```

1 <core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
2   <ViewSettingsDialog id="viewSettingsDialog" confirm="onConfirmViewSettingsDialog">
3     <sortItems>
4       <ViewSettingsItem text="{i18n>filterCarId}" key="Carrid"/>
5       <ViewSettingsItem text="{i18n>filterCarname}" key="Carname"/>
6     </sortItems>
7     <groupItems>
8       <ViewSettingsItem text="{i18n>filterCarId}" key="Carrid"/>
9       <ViewSettingsItem text="{i18n>filterCarname}" key="Carname"/>
10    </groupItems>
11  </ViewSettingsDialog>
12 </core:FragmentDefinition>

```

Figure 248: Add the Implementation to the New Fragment

d) Save your work.

4. Implement the `onOpenViewSettings` function inside the controller of the *Overview* in the figure.



Overview.controller.js

```

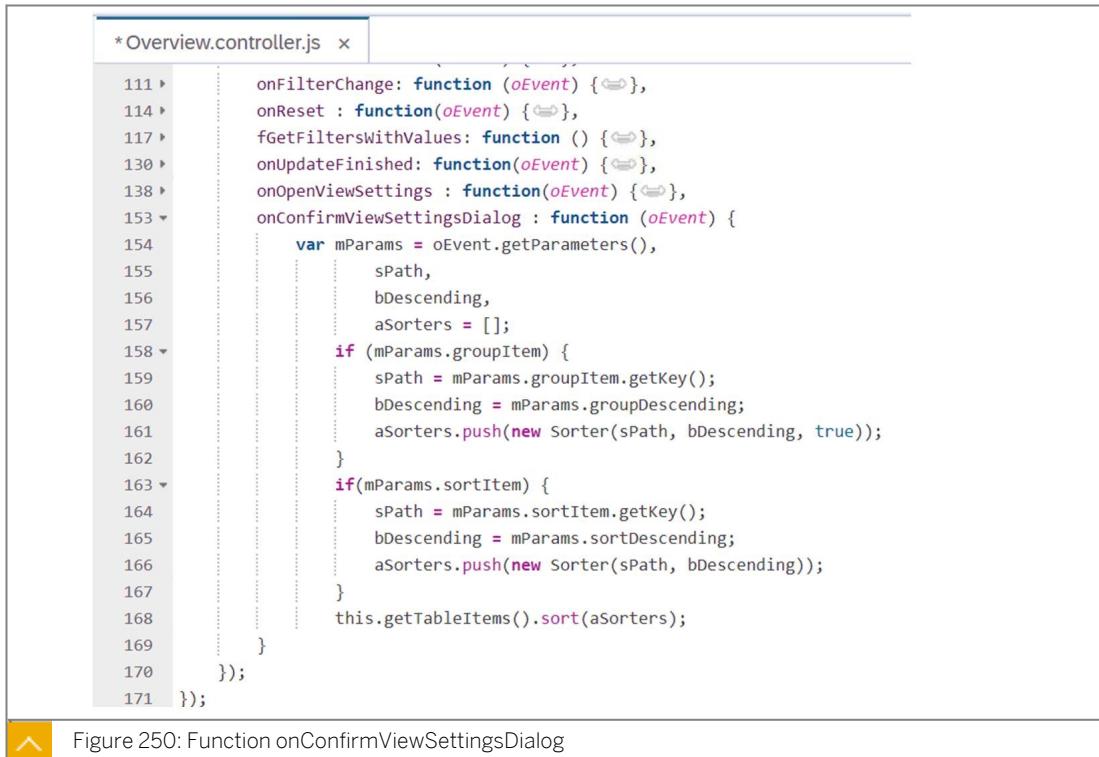
80 >   onSearch: function (oEvent) { },
81 >   onClear: function (oEvent) { },
82 >   onFilterChange: function (oEvent) { },
83 >   onReset: function (oEvent) { },
84 >   fGetFiltersWithValues: function () { },
85 >   onUpdateFinished: function (oEvent) { },
86 >   onOpenViewSettings: function (oEvent) {
87 >     if (!this._oViewSettingsDialog) {
88 >       this._oViewSettingsDialog = sap.ui.xmlfragment("student00.sap.training.dynamicpage.view.ViewSettingsDialog", this);
89 >       this.getView().addDependent(this._oViewSettingsDialog);
90 >       jQuery.sap.syncStyleClass("sapUiSizeCompact", this.getView(), this._oViewSettingsDialog);
91 >     }
92 >     var sDialogTab = "sort";
93 >     if (oEvent.getSource() instanceof sap.m.Button) {
94 >       var sbuttonId = oEvent.getSource().getId();
95 >       if (sbuttonId.match("group")) {
96 >         sDialogTab = "group";
97 >       }
98 >     }
99 >     this._oViewSettingsDialog.open(sDialogTab);
100 >   });
101 > });
102 > });
103 > });
104 > });
105 > });
106 > });
107 > });
108 > });
109 > });
110 > });
111 > });
112 > });
113 > });
114 > });
115 > });
116 > });
117 > });
118 > });
119 > });
120 > });
121 > });
122 > });
123 > });
124 > });
125 > });
126 > });
127 > });
128 > });
129 > });
130 > });
131 > });
132 > });
133 > });
134 > });
135 > });
136 > });
137 > });
138 > });
139 > });
140 > });
141 > });
142 > });
143 > });
144 > });
145 > });
146 > });
147 > });
148 > });
149 > });
150 > });
151 > });
152 > });
153 > });

```

Figure 249: Function `onOpenViewSettings`

- a) Open the file `Overview.controller.js` located in the controller folder of your project.
- b) Add the implementation of the `onOpenViewSettings` function as already shown.
- c) Save your work.

5. Implement the function `onConfirmViewSettingsDialog` in the controller of the *Overview* as shown.



```

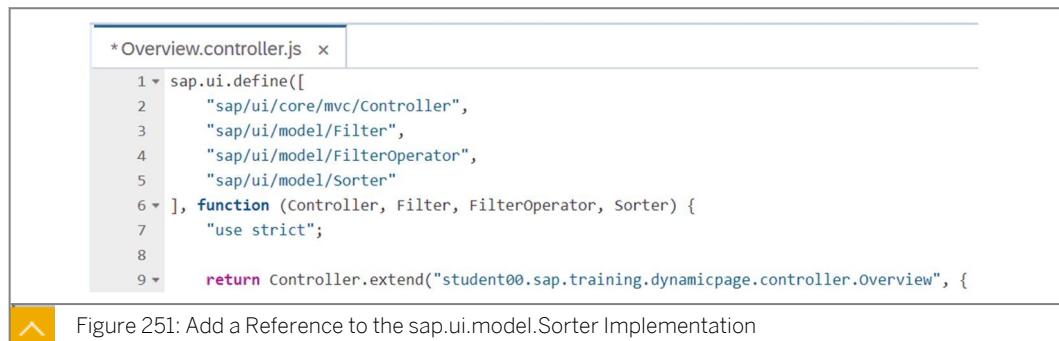
*Overview.controller.js  x
111 >     onFilterChange: function (oEvent) { },
114 >     onReset : function(oEvent) { },
117 >     fGetFiltersWithValues: function () { },
130 >     onUpdateFinished: function(oEvent) { },
138 >     onOpenViewSettings : function(oEvent) { },
153 >     onConfirmViewSettingsDialog : function (oEvent) {
154         var mParams = oEvent.getParameters(),
155             sPath,
156             bDescending,
157             aSorters = [];
158         if (mParams.groupItem) {
159             sPath = mParams.groupItem.getKey();
160             bDescending = mParams.groupDescending;
161             aSorters.push(new Sorter(sPath, bDescending, true));
162         }
163         if(mParams.sortItem) {
164             sPath = mParams.sortItem.getKey();
165             bDescending = mParams.sortDescending;
166             aSorters.push(new Sorter(sPath, bDescending));
167         }
168         this.getTableItems().sort(aSorters);
169     });
170 });
171 });

```



Figure 250: Function onConfirmViewSettingsDialog

- Open the file `Overview.controller.js` if not already open.
- Add a reference to the `sap.ui.model.Sorter` implementation to your `Overview` controller.



```

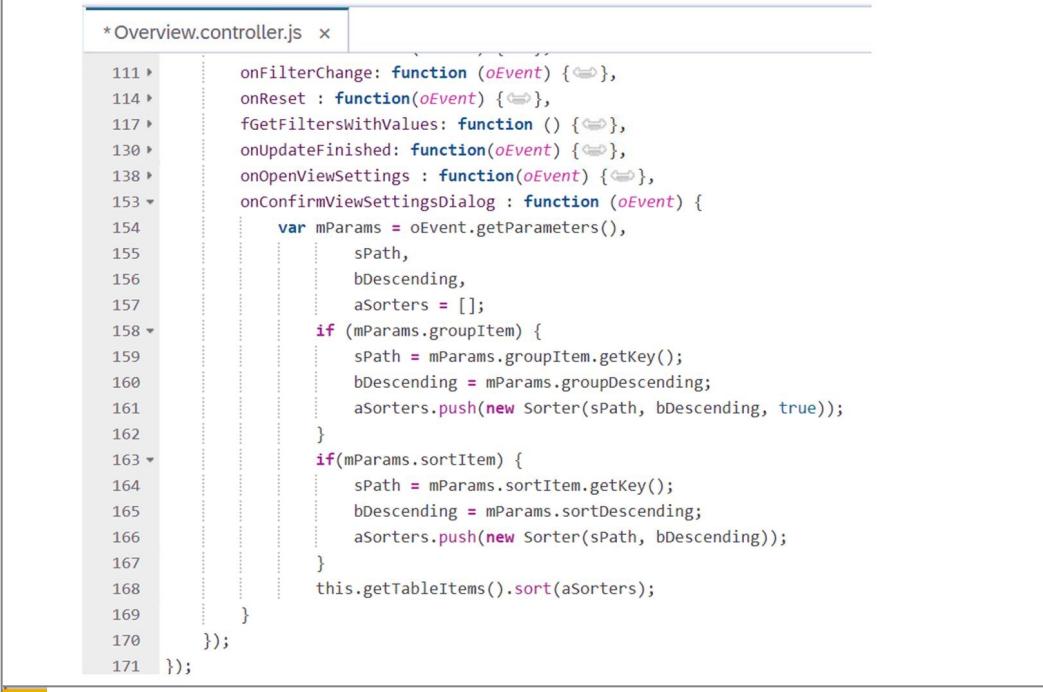
*Overview.controller.js  x
1 > sap.ui.define([
2     "sap/ui/core/mvc/Controller",
3     "sap/ui/model/Filter",
4     "sap/ui/model/FilterOperator",
5     "sap/ui/model/Sorter"
6 ], function (Controller, Filter, FilterOperator, Sorter) {
7     "use strict";
8
9     return Controller.extend("student00.sap.training.dynamicpage.controller.Overview", {

```



Figure 251: Add a Reference to the sap.ui.model.Sorter Implementation

- Add the implementation of `onConfirmViewSettingsDialog` to your controller. Implement the function as shown.



```

*Overview.controller.js x
111 >     onFilterChange: function (oEvent) {
114 >         onReset : function(oEvent) { },
117 >         fGetFiltersWithValues: function () { },
130 >         onUpdateFinished: function(oEvent) { },
138 >         onOpenViewSettings : function(oEvent) { },
153 >         onConfirmViewSettingsDialog : function (oEvent) {
154             var mParams = oEvent.getParameters(),
155                 sPath,
156                 bDescending,
157                 aSorters = [];
158             if (mParams.groupItem) {
159                 sPath = mParams.groupItem.getKey();
160                 bDescending = mParams.groupDescending;
161                 aSorters.push(new Sorter(sPath, bDescending, true));
162             }
163             if(mParams.sortItem) {
164                 sPath = mParams.sortItem.getKey();
165                 bDescending = mParams.sortDescending;
166                 aSorters.push(new Sorter(sPath, bDescending));
167             }
168             this.getTableItems().sort(aSorters);
169         }
170     });
171 });

```

Figure 252: Implement the Function

d) Save your work.

6. Test your application.

a) Restart your application.

b) Choose a configuration to test the sort and group dialog and apply by choosing OK.

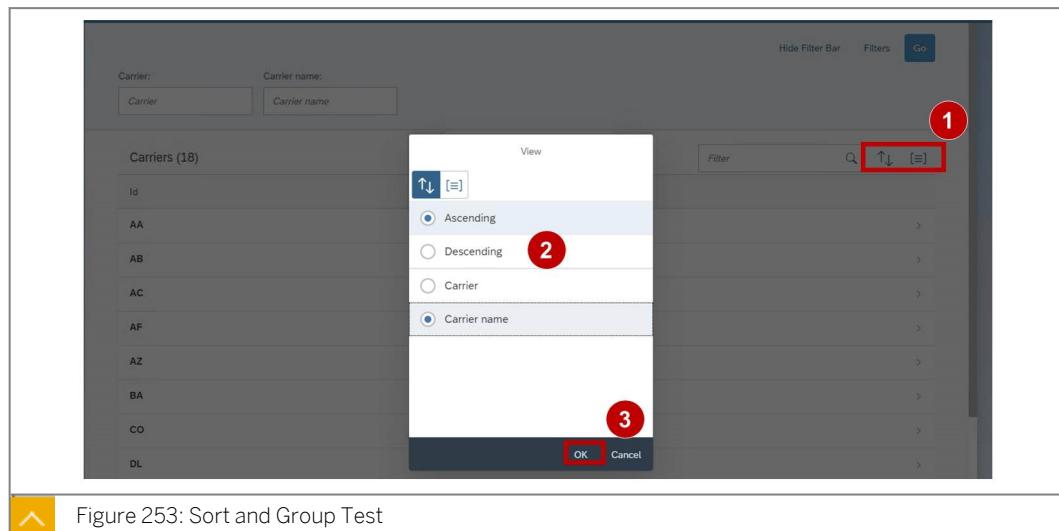


Figure 253: Sort and Group Test

The listed carriers should be sorted or grouped as expected.

c) Close the browser tab of the application after review.

## Unit 12

### Exercise 13

# Implement an Object Page

## Object Page



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

## Business Example

In this exercise, you will learn how to implement the Object Page floorplan.

### Task 1: Object Page

1. Export the project *listreport* from the previous exercise and import the project with the name **objectpage** into the workspace of your *SAP Web IDE Full Stack*.
2. Import the content of the file *listreport.zip* as a project with the name **objectpage**.

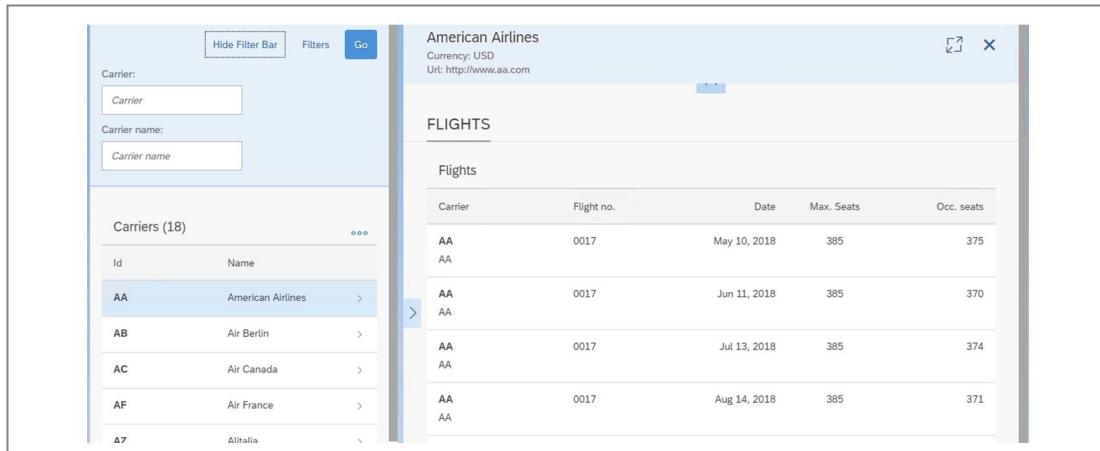
### Task 2: Implement the object page floorplan to show the flights and connections as sections.



### Note:

For your convenience you will find a template at `S:\Courses\UX410_20\Templates\Ex_ObjectPage_CarrierView.txt`. Be aware that you need to change the path to your controller by inserting your own group number into the path.

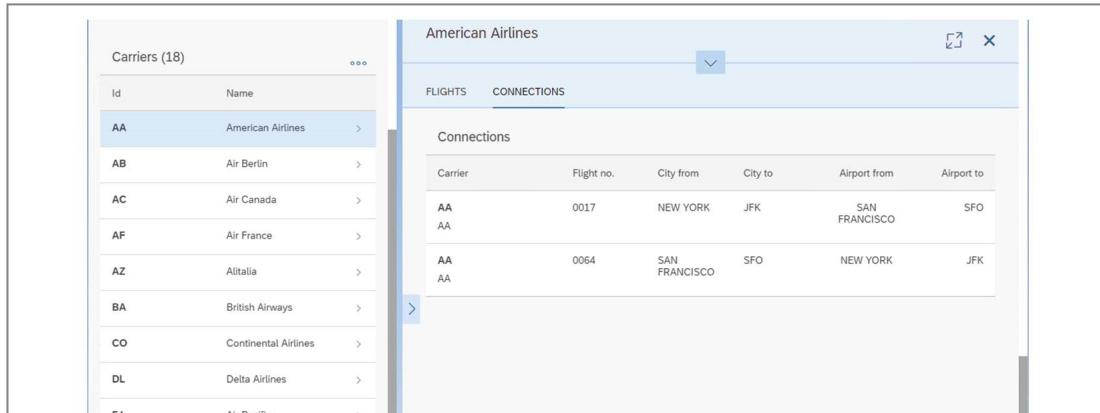
1. Change the implementation of the *Carrier* view. Remove the `sap.f.semantic.SemanticPage` control and use the `sap.uxap.ObjectPageLayout` instead. Leave the look and feel of the *Carrier* view unchanged.
2. Add a section to the `ObjectPageLayout` control and display the flights of the selected carrier. The section should look as shown. You can access the flights by using the navigation property `to_Flight`.



The screenshot shows an SAP Fiori object page for American Airlines. On the left, a sidebar titled 'Carriers (18)' lists various airline names. The main content area is titled 'American Airlines' and shows flight information. A table titled 'FLIGHTS' lists flights for carrier 'AA' (American Airlines) with columns for Carrier, Flight no., Date, Max. Seats, and Occ. seats. The data shows four flights on different dates: May 10, 2018 (385 seats, 375 occupied); Jun 11, 2018 (385 seats, 370 occupied); Jul 13, 2018 (385 seats, 374 occupied); and Aug 14, 2018 (385 seats, 371 occupied).

Figure 257: Flights of Carrier

3. Add another section to the object page and display the available connections. You can get the connections with the navigation property `to_Connection`.



The screenshot shows the same SAP Fiori object page for American Airlines, but with an additional 'CONNECTIONS' section. The 'FLIGHTS' section remains the same. The 'CONNECTIONS' section is titled 'Connections' and lists flight information for carrier 'AA' (American Airlines) with columns for Carrier, Flight no., City from, City to, Airport from, and Airport to. The data shows two flights: one from NEW YORK (JFK) to SAN FRANCISCO (SFO) and another from SAN FRANCISCO (SFO) to NEW YORK (JFK).

Figure 263: Display available Connections

4. Test your implementation.

## Implement an Object Page

### Object Page



#### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

### Business Example

In this exercise, you will learn how to implement the Object Page floorplan.

#### Task 1: Object Page

1. Export the project *listreport* from the previous exercise and import the project with the name **objectpage** into the workspace of your *SAP Web IDE Full Stack*.
  - a) Log on to the *SAP Web IDE Full-Stack*.
  - b) Choose the project *listreport* from the workspace and select *Export* from the context menu.
  - c) When downloaded successfully, the package displays in the bottom of your browser window.



Figure 254: listreport Zip File

2. Import the content of the file *listreport.zip* as a project with the name **objectpage**.
  - a) Select the workspace folder and from the context menu, choose *Import* → *File or Project*.
  - b) Select the exported file using the *Browse* button and change the file name in the *Import* field to **/objectpage** and choose *OK*.
  - c) When the import finishes successful you will see the newly imported project with the given name.

## Task 2: Implement the object page floorplan to show the flights and connections as sections.



### Note:

For your convenience you will find a template at `S:\Courses\UX410_20\Templates\Ex_ObjectPage_CarrierView.txt`. Be aware that you need to change the path to your controller by inserting your own group number into the path.

1. Change the implementation of the *Carrier* view. Remove the `sap.f.semantic.SemanticPage` control and use the `sap.uxap.ObjectPageLayout` instead. Leave the look and feel of the *Carrier* view unchanged.
  - a) Open the file `Carrier.view.xml` located in the view folder of your project.
  - b) Open the template `S:\Courses\UX410_20\Templates\Ex_ObjectPage_CarrierView.txt` using notepad or another editor.
  - c) Copy the content of the template file.
  - d) Paste the content of the template `S:\Courses\UX410_20\Templates\Ex_ObjectPage_CarrierView.txt` to the `Carrier.view.xml` in the SAP Web IDE.
  - e) Change the value of the `controllerName` attribute to use your own group number.
  - f) The code should look as shown.

```
*Carrier.view.xml *
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:layout="sap.ui.layout"
2   xmlns:uxap="sap.uxap" controllerName="student00.sap.training.dynamicpage.controller.Carrier" xmlns:html="http://www.w3.org/1999/xhtml">
3   <uxap:objectPageLayout id="dynamicPageId" showTitleInHeaderContent="true" alwaysShowContentHeader="false"
4     preserveHeaderStateOnScroll="false" headerContentPinnable="true" isChildPage="true" enableLazyLoading="false">
5     <uxap:headerTitle>
6       <uxap:objectPageDynamicHeaderTitle>
7         <uxap:expandedHeading>
8           <title text="{carname}">
9             </title>
10            <uxap:snappedHeading>
11              <title text="{carname}">
12                </title>
13              <uxap:snappedHeading>
14                <uxap:expandedContent>
15                  <flexbox alignItems="Start" justifyContent="spaceBetween">
16                    <items>
17                      <layout:HorizontalLayout allowWrapping="true">
18                        <layout:VerticalLayout classe="sapUiMediumMarginEnd">
19                          <ObjectAttribute title="{i18n>currLabelText}" text="{currCode}">
20                            <ObjectAttribute title="{i18n>urlLabelText}" text="{url}">
21                          </ObjectAttribute>
22                        </layout:VerticalLayout>
23                      </items>
24                    </FlexBox>
25                  <uxap:expandedContent>
26                    <uxap:navigationActions>
27                      <uxap:objectPageDynamicHeaderTitle>
28                      </uxap:objectPageDynamicHeaderTitle>
29                    </uxap:headerTitle>
30                  </uxap:objectPageLayout>
31                </mvc:View>
```



Figure 255: Code



```

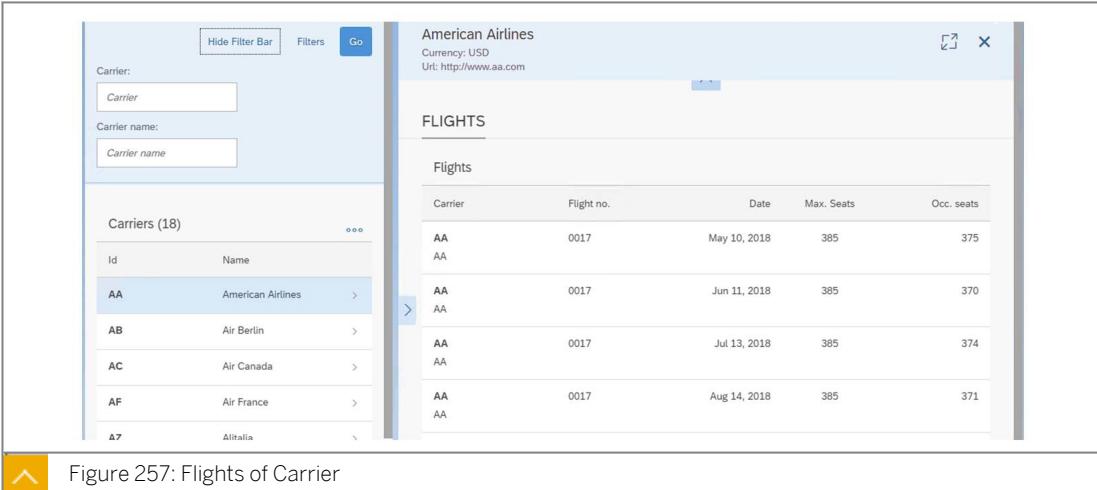
* Carrier.view.xml x
1  <!-- Carrier.view.xml -->
2  <!-- This file is part of the Fiori Design Guidelines. -->
3  <!-- For more information, see https://ui5.sap.com -->
4  <!-- © 2018 SAP SE or an SAP affiliate company. -->
5  <!-- All rights reserved. SAP, the SAP logo and SAP Fiori are -->
6  <!-- trademarks or registered trademarks of SAP SE in -->
7  <!-- Germany and/or other countries -->
8  <!-- All other trademarks, product names, and/or services -->
9  <!-- names are trademarks of their respective companies. -->
10 <!-- SAP does not endorse non-SAP products and services. -->
11 <!-- All SAP products and services are provided "AS IS" -->
12 <!-- without warranty of any kind. -->
13 <!-- SAP products and services may be used in conjunction with -->
14 <!-- other SAP products and services, but SAP is not responsible -->
15 <!-- for the performance of any integrated products. -->
16 <!-- SAP products and services are not a substitute for any -->
17 <!-- products and services in the SAP Fiori family. -->
18 <!-- SAP products and services are not a substitute for any -->
19 <!-- products and services in the SAP Fiori family. -->
20 <!-- SAP products and services are not a substitute for any -->
21 <!-- products and services in the SAP Fiori family. -->
22 <!-- SAP products and services are not a substitute for any -->
23 <!-- products and services in the SAP Fiori family. -->
24 <!-- SAP products and services are not a substitute for any -->
25 <!-- products and services in the SAP Fiori family. -->
26 <!-- SAP products and services are not a substitute for any -->
27 <!-- products and services in the SAP Fiori family. -->
28 <!-- SAP products and services are not a substitute for any -->
29 <!-- products and services in the SAP Fiori family. -->
30 <!-- SAP products and services are not a substitute for any -->
31 <!-- products and services in the SAP Fiori family. -->
32 <!-- SAP products and services are not a substitute for any -->
33 <!-- products and services in the SAP Fiori family. -->
34 <!-- SAP products and services are not a substitute for any -->
35 <!-- products and services in the SAP Fiori family. -->
36 <!-- SAP products and services are not a substitute for any -->
37 <!-- products and services in the SAP Fiori family. -->
38 <!-- SAP products and services are not a substitute for any -->
39 <!-- products and services in the SAP Fiori family. -->
40 <!-- SAP products and services are not a substitute for any -->

```

Figure 256: Code

 Note:  
The implementation shows the controller of group 00. Your implementation should show your group number.

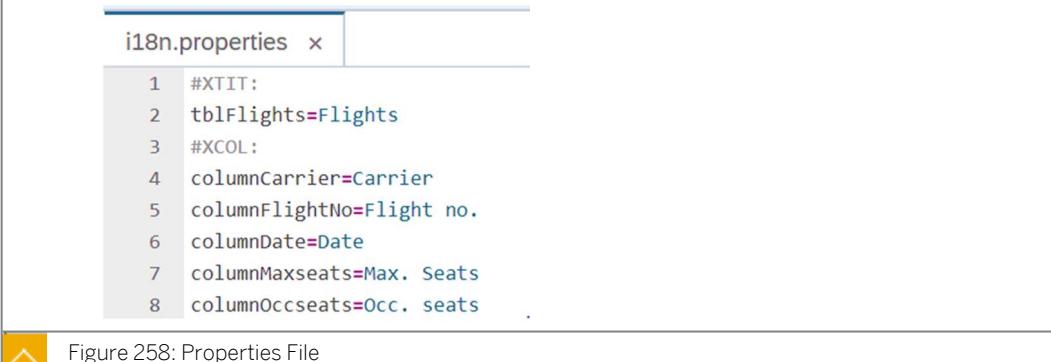
2. Add a section to the `ObjectPageLayout` control and display the flights of the selected carrier. The section should look as shown. You can access the flights by using the navigation property `to_Flight`.



Carrier	Flight no.	Date	Max. Seats	Occ. seats
AA	0017	May 10, 2018	385	375
AA	0017	Jun 11, 2018	385	370
AA	0017	Jul 13, 2018	385	374
AA	0017	Aug 14, 2018	385	371

Figure 257: Flights of Carrier

- a) Open the `i18n.properties` file located in the `i18n` folder of your project.
- b) Add the following key and value pairs to the `i18n.properties` file.



```

i18n.properties x
1 #XTIT:
2 tblFlights=Flights
3 #XCOL:
4 columnCarrier=Carrier
5 columnFlightNo=Flight no.
6 columnDate=Date
7 columnMaxseats=Max. Seats
8 columnOccseats=Occ. seats

```

Figure 258: Properties File

- c) Add the **sections** aggregation to the `sap.uxap.ObjectPageLayout` immediately after the `headerTitle` aggregation. Insert a control of type `sap.uxap.ObjectPageSection` to the **sections** aggregation and add the attribute `title` to it. Assign the property `i18n>tblFlights` to the `title` attribute. Add the **subSections** aggregation to the `sap.uxap.ObjectPageSection` control and insert a control of type `sap.uxap.ObjectPageSection` in the **subSections** aggregation. Add the **blocks** aggregation to the newly added `ObjectPageSubSection` control.



```

Carrier.view.xml x
1 <mvc:view xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:layout="sap.ui.layout"
2   xmlns:uxap="sap.uxap" controllerName="student00.sap.training.dynamicpage.controller.Carrier" xmlns:html="http://www.w3.org/1999/xhtml">
3   <uxap:ObjectPageLayout id="dynamicPageId" showTitleInHeaderContent="true" alwaysShowContentHeader="false">
4     <preserveHeaderStateOnScroll="false" headerContentPinnable="true" isChildPage="true" enableLazyLoading="false">
5       <uxap:headerTitle></uxap:headerTitle>
6       <uxap:sections>
7         <uxap:ObjectPageSection title="{i18n>tblFlights}">
8           <uxap:subSections>
9             <uxap:ObjectPageSubSection>
10            <uxap:blocks>
11              <!-- Content for the first sub-section -->
12            </uxap:blocks>
13          </uxap:ObjectPageSubSection>
14        </uxap:subSections>
15      </uxap:ObjectPageSection>
16    </uxap:sections>
17  </uxap:ObjectPageLayout>
18 </mvc:view>

```

Figure 259: Add Sections

- d) Add an `sap.m.Table` with the following implementation to the **blocks** aggregation.



```

Carrier.view.xml x
1 <mvc:view xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:layout="sap.ui.layout"
2   xmlns:uxap="sap.uxap" controllerName="student00.sap.training.dynamicpage.controller.Carrier" xmlns:html="http://www.w3.org/1999/xhtml">
3   <uxap:ObjectPageLayout id="dynamicPageId" showTitleInHeaderContent="true" alwaysShowContentHeader="false">
4     <preserveHeaderStateOnScroll="false" headerContentPinnable="true" isChildPage="true" enableLazyLoading="false">
5       <uxap:headerTitle></uxap:headerTitle>
6       <uxap:sections>
7         <uxap:ObjectPageSection title="{i18n>tblFlights}">
8           <uxap:subSections>
9             <uxap:ObjectPageSubSection>
10            <uxap:blocks>
11              <!-- Content for the first sub-section -->
12              <!-- Implementation of the sap.m.Table control -->
13              <Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'carrid' } }" growing="true"
14                growingThreshold="10" visible="true">
15                <headerToolbar>
16                  <overflowToolbar>
17                    <title text="{i18n>tblFlights}" level="H2"/>
18                  </overflowToolbar>
19                </headerToolbar>
20                <columns></columns>
21                <items></items>
22              </Table>
23            </uxap:blocks>
24          </uxap:ObjectPageSubSection>
25        </uxap:subSections>
26      </uxap:ObjectPageSection>
27    </uxap:sections>
28  </uxap:ObjectPageLayout>
29 </mvc:view>

```

Figure 260: Add Table

- e) Add an implementation of the **columns** aggregation to the table.

Figure 261: Add Columns

f) Add an implementation to the **items** aggregation of the table.

```
*Carrier.view.xml x
2  xmlns:uxap="sap.uxap" controllerName="student00.sap.training.dynamicpage.controller.Carrier" xmlns:html="http://www.w3.org/1999/xhtml">
3  <uxap:objectPageLayout id="dynamicPageId" showTitleInHeaderContent="true" alwaysShowContentHeader="false">
4    preserveHeaderStateOnScroll="false" headerContentPinnable="true" isChildPage="true" enableLazyLoading="false">
5      <uxap:headerTitle></uxap:headerTitle>
6      <uxap:sections>
7        <uxap:objectPageSection title="{i18n>tblFlights}">
8          <uxap:subSection>
9            <uxap:objectPageSubSection>
10           <uxap:blocks>
11             <Table id="idFlights" items="{ path: 'to_Flight', sorter: { path: 'Carrid' } }" growing="true"
12               growingThreshold="10" visible="true">
13               <headerToolbar></headerToolbar>
14               <columns></columns>
15               <items>
16                 <columnListItem>
17                   <cells>
18                     <objectIdentifier title="{Carrid}" text="{Carrid}" />
19                     <text text="{Connid}" />
20                     <text text="{Fidate}" type="sap.ui.model.type.Date" />
21                     <text text="{Seatsmax}" />
22                     <text text="{Seatsocc}" />
23                   </cells>
24                 </columnListItem>
25               </items>
26             </Table>
27           </uxap:blocks>
28         </uxap:objectPageSubSection>
29       </uxap:subSection>
30     </uxap:sections>
31   </uxap:objectPageLayout>
32 </Carrier>
```

Figure 262: Items Aggregation

3. Add another section to the object page and display the available connections. You can get the connections with the navigation property to `Connection`.

Carriers (18)

Id	Name
AA	American Airlines >
AB	Air Berlin >
AC	Air Canada >
AF	Air France >
AZ	Alitalia >
BA	British Airways >
CO	Continental Airlines >
DL	Delta Airlines >
FJ	Air Pacific >

American Airlines

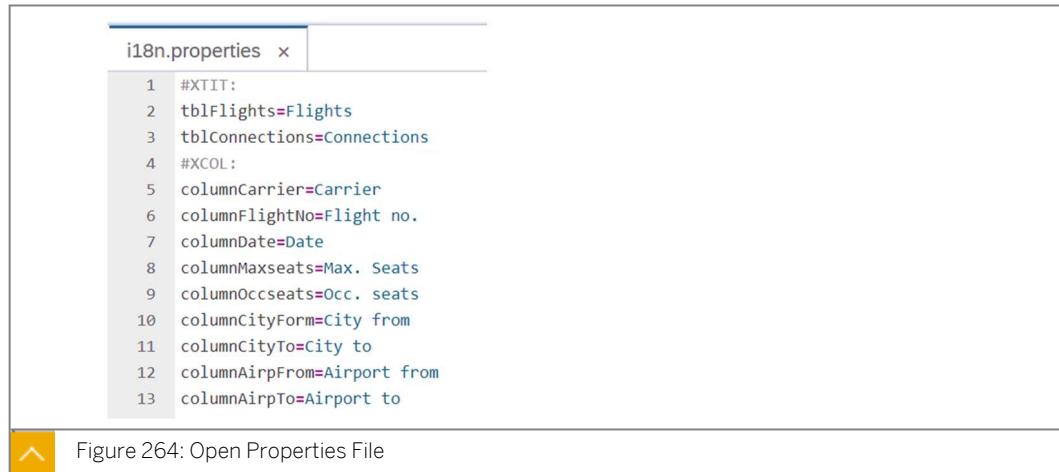
FLIGHTS CONNECTIONS

Connections

Carrier	Flight no.	City from	City to	Airport from	Airport to
AA	0017	NEW YORK	JFK	SAN FRANCISCO	SFO
AA	0064	SAN FRANCISCO	SFO	NEW YORK	JFK

Figure 263: Display available Connections

a) Open the `i18n.properties` file located in the `i18n` folder of your project and add the following implementation.



```
i18n.properties x
1 #XTIT:
2 tblFlights=Flights
3 tblConnections=Connections
4 #XCOL:
5 columnCarrier=Carrier
6 columnFlightNo=Flight no.
7 columnDate=Date
8 columnMaxseats=Max. Seats
9 columnOccseats=Occ. seats
10 columnCityFrom=City from
11 columnCityTo=City to
12 columnAirpFrom=Airport from
13 columnAirpTo=Airport to
```

Figure 264: Open Properties File

- b) Insert a control of type `sap.uxap.ObjectPageSection` to the **sections** aggregation immediately after the `sap.uxap.ObjectPageSection` created in the previous step. Add the attribute `title` to it. Assign the property `{i18n>tblConnections}` to the `title` attribute. Add the **subSections** aggregation to the `sap.uxap.ObjectPageSection` control and insert a control of type `sap.uxap.ObjectPageSection` into the **subSections** aggregation. Add the **blocks** aggregation to the newly added `ObjectPageSubSection` control.



```
*Carrier.view.xml x
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f" xmlns:layout="sap.ui.layout"
2   xmlns:uxap="sap.uxap" controllerName="student00.sap.training.dynamicpage.controller.Carrier" xmlns:html="http://www.w3.org/1999/xhtml">
3   <uxap:objectPageLayout id="dynamicPaged" showTitleInHeaderContent="true" alwaysShowContentHeader="false"
4     preserveHeaderStateOnScroll="false" headerContentPinnable="true" isChildPage="true" enableLazyLoading="false">
5     <uxap:headerTitle></uxap:headerTitle>
6     <uxap:sections>
7       <uxap:objectPageSection title="{i18n>tblFlights}"></uxap:objectPageSection>
8       <uxap:objectPageSection title="{i18n>tblConnections}">
9         <uxap:subSections>
10           <uxap:objectPageSubSection>
11             <uxap:blocks>
12               </uxap:blocks>
13             </uxap:objectPageSubSection>
14           </uxap:subSections>
15         </uxap:objectPageSection>
16       </uxap:sections>
17     </uxap:objectPageLayout>
18   </mvc:View>
```

Figure 265: Add control

- c) Add an `sap.m.Table` control to the **blocks** aggregation of the newly added `sap.uxap.ObjectPageSubSection`. Add the shown attributes and aggregations to the `sap.m.Table` control.



The screenshot shows the SAP Fiori View XML code editor with the file name 'Carrier.view.xml'. The code is a fragment of an Object Page layout. It includes sections for header, object pages, and sub-sections. A specific section for 'Connections' is highlighted, showing the addition of a Table control ('

Figure 266: Add a Table Control to the Blocks Aggregation

d) Add an implementation to the columns aggregation of the `sap.m.Table` control.



The screenshot shows the SAP Fiori View XML code editor with the file name 'Carrier.view.xml'. The code is a fragment of an Object Page layout. It includes sections for header, object pages, and sub-sections. A specific section for 'Connections' is highlighted, showing the implementation of the 'columns' aggregation for the Table control. The columns are defined with widths and text labels for 'columnCarrier', 'columnFlightNo', 'columnCityForm', 'columnCityTo', and 'columnAirpFrom' (horizontal alignment="Center") and 'columnAirpTo' (horizontal alignment="Right"). The code is numbered from line 81 to 109.

Figure 267: Implementation

e) Add an implementation of the items aggregation to the table.



The screenshot shows the SAP Fiori View XML code editor with the file name 'Carrier.view.xml'. The code is a fragment of an Object Page layout. It includes sections for header, object pages, and sub-sections. A specific section for 'Connections' is highlighted, showing the addition of items aggregation to the Table control. The table has a header toolbar and a columns aggregation. The items aggregation contains a column list item (''), which in turn contains cells (''), each with an object identifier ('') and text labels for 'Connid', 'Cityfrom', 'Airfrom', 'Cityto', and 'Airpto'. The code is numbered from line 4 to 122.

Figure 268: Add items Aggregation to Table

4. Test your implementation.

- Select the `index.html` file located in the `webapp` folder of your project and choose from the context menu `Run → Run as Web Application`.
- If you are asked to enter your front-end server credentials input the user `Train-##` with your password and choose `Sign in`.

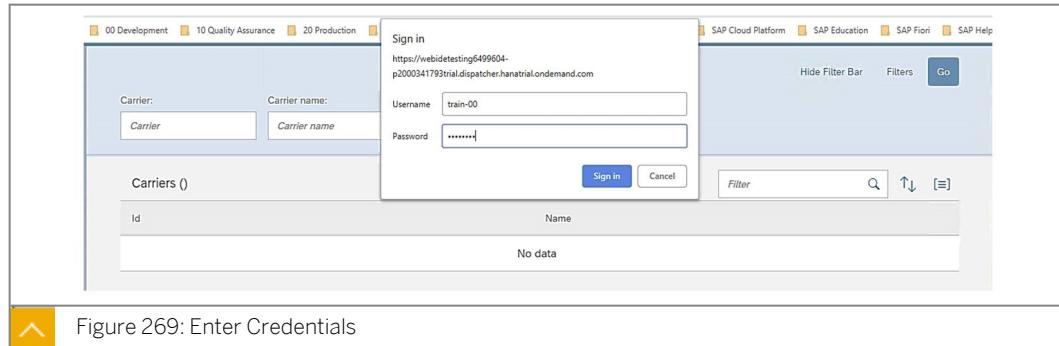


Figure 269: Enter Credentials

- Select a carrier from the list of carriers.
- Explore your implementation by selecting the CONNECTIONS tab to display the available list of connections.

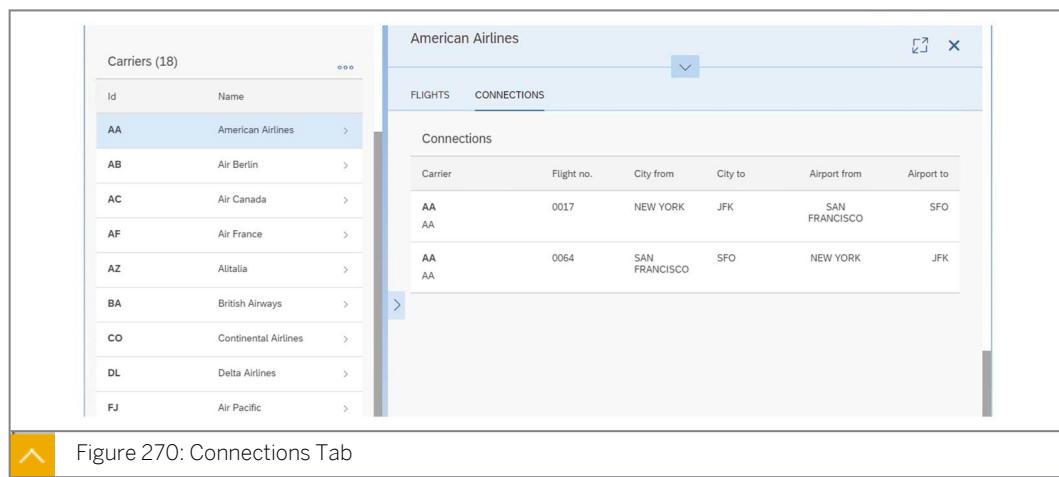


Figure 270: Connections Tab

**Note:**

There may be carriers in the list with neither flights nor connections. In these cases, there are no result in the tables. Try another carrier.

## Unit 13

### Exercise 14

# Implement and Extend an Extension Point



#### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

In this exercise you will work with SAP Fiori Extension techniques. The basis for this exercise is the implementation of the exercise **Object Page**.

Extend your deployed application `ZUX410_APP##` (system is FSD).

1. Open the result of the exercise **Object Page**. Add the ID `idCarrierDetails` to the `sap.m.FlexBox` control located in the `sap.uxap.ObjectPageLayout` control.



#### Note:

Create a new separate project by using the export and import function of the SAP Web IDE. Export your `objectpage` project and import it with the new name of `objectPageWithExtension`.

2. Add an `sap.ui.core.ExtensionPoint` with the name `ux410_extension` immediately after the `sap.m.FlexBox` control with the new ID `idCarrierDetails`.
3. Add a controller `hook` function with the name `extUX410HookFunction` to the controller of view `Carrier`. Call, if it exists, the `hook` function at the end of the `onInit` function of the controller. Add the following documentation before the invocation.

Table 72: `extUX410HookFunction` Properties and Documentation

JSDoc property	Documentation
<code>@ControllerHook</code>	This hook method can be used to add some extra functionality
<code>@callback</code>	<code>student##.sap.training.dynamicpage.controller.Carrier~extUX410HookFunction</code>
<code>@return</code>	<code>{void}</code>

4. Deploy your application.

Table 73: Application Properties and Values

Property	Value
Application name	ZUX410_App##
Description	Flight app
System	FSD_100 - FSD
Package	ZTRAIN_##
Transport request	Choose the transport request provide by your teacher.

### Extend the Deployed Application.

1. Extend your deployed application ZUX410\_APP## (system is FSD). Create an extension project with the name **ZUX410\_APP##Extension**. Hide the UI control with ID **idCarrierDetails** by using the extension pane. Create a view extension for the extension point **ux410\_extension** using the extension pane. Add the code provided in the example for the extension. Test your extended application.



```

Carrier_ux410_extensionCustom x
1 <core:FragmentDefinition xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:layout="sap.ui.layout"
3   xmlns:uxap="sap.uxap" xmlns:html="http://www.w3.org/1999/xhtml">
4   <FlexBox alignItems="Start" justifyContent="SpaceBetween">
5     <items>
6       <layout:HorizontalLayout allowWrapping="true">
7         <layout:VerticalLayout classe="sapUiMediumMarginEnd">
8           <ObjectAttribute title="{i18n>currLabelText}" text="{CurrCode}" />
9           <ObjectAttribute title="{i18n>urlLabelText}" text="{Url}" />
10          <ObjectAttribute title="Carrier id" text="{Carrid}" />
11          <ObjectAttribute title="Carrier name" text="{Carrname}" />
12        </layout:VerticalLayout>
13      </layout:HorizontalLayout>
14    </items>
15  </FlexBox>
16 </core:FragmentDefinition>

```

Figure 275: Extension Code

2. Add an i18n-resource customizing extension to your project and enter the new text properties listed. Replace the hard coded text literals located in the view extension. Test your application.

Table 74: Text Properties

Key	Value	Replace the following hard coded string in view extension
idLabelText	This is the carrier id	Carrier id
nameLabelText	This is the carrier name	Carrier name

3. Implement a controller extension for the controller of *Carrier* view and implement the function `extUX410HookFunction`. The function shows an `sap.m.MessageToast` with the message **Called from Init**. Test if your changes are applied.

# Unit 13 Solution 14

## Implement and Extend an Extension Point



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.

In this exercise you will work with SAP Fiori Extension techniques. The basis for this exercise is the implementation of the exercise **Object Page**.

Extend your deployed application **ZUX410\_APP##** (system is FSD).

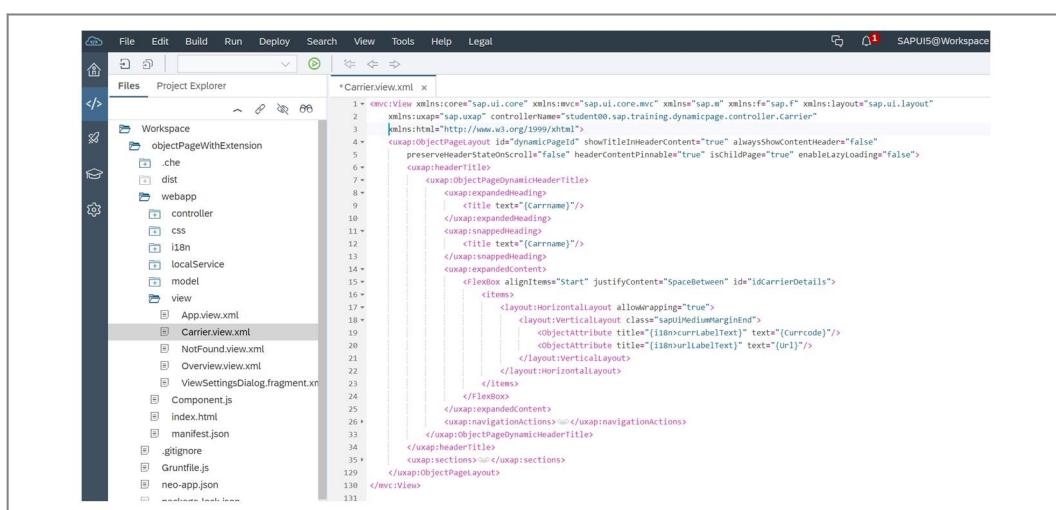
1. Open the result of the exercise **Object Page**. Add the ID `idCarrierDetails` to the `sap.m.FlexBox` control located in the `sap.uxap.ObjectPageLayout` control.



### Note:

Create a new separate project by using the export and import function of the SAP Web IDE. Export your objectpage project and import it with the new name of **objectPageWithExtension**.

- a) Open the file `Carrier.view.xml` located in the view folder of the project.
- b) Add the `idCarrierDetails` to the `sap.m.FlexBox` control, located in the `expandedContent` aggregation of the `sap.uxap.ObjectPageLayout` control.



```
<smvc:view xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:fx="sap.ui.layout" xmlns:layout="sap.ui.layout">
  <mln:html href="http://www.w3.org/1999/xhtml">
    <uxap:objectPageLayout id="dynamicPageId" showTitleInHeaderContent="true" alwaysShowContentHeader="false" preserveHeaderStateOnScroll="false" headerContentInnable="true" isChildPages="true" enableLazyLoading="false">
      <uxap:headerTitle>
        <uxap:objectPageDynamicHeaderTitle>
          <uxap:expandedHeader>
            <title text="{carname}">
              <uxap:snappedHeader>
                <title text="{carname}">
                  <uxap:snappedHeader>
                    <uxap:expandedContent>
                      <flexBox alignItems="Start" justifyContent="SpaceBetween" id="idCarrierDetails">
                        <items>
                          <layout:HorizontalLayout allowWrapping="true">
                            <layout:VerticalLayout class="sapUiMediumMarginEnd">
                              <objAttribute title="{l10n:currLabelText}" text="{currCode}">
                            </layout:VerticalLayout>
                            <objAttribute title="{l10n:urlLabelText}" text="{url}">
                          </layout:HorizontalLayout>
                        </items>
                      </flexBox>
                    </uxap:expandedContent>
                    <uxap:navigationActions></uxap:navigationActions>
                  </uxap:snappedHeader>
                </uxap:snappedHeader>
              </uxap:snappedHeader>
            </uxap:expandedHeader>
          </uxap:objectPageDynamicHeaderTitle>
        </uxap:headerTitle>
        <uxap:sections></uxap:sections>
      </uxap:objectPageLayout>
    </mln:html>
  </smvc:view>
```

Figure 271: Add `idCarrierDetails`

2. Add an `sap.ui.core.ExtensionPoint` with the name `ux410_extension` immediately after the `sap.m.FlexBox` control with the new ID `idCarrierDetails`.

- a) Add the `sap.ui.core.ExtensionPoint` with the name `ux410_extension` immediately after the `sap.m.FlexBox` control with the new ID `idCarrierDetails`.



```
*Carrier.view.xml x
1 <mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" xmlns:f="sap.f"
2   xmlns:layout="sap.ui.layout" xmlns:uxap="sap.uxap" controllerName="student00.sap.training.dynamicpage.controller.Carrier"
3   xmlns:html="http://www.w3.org/1999/xhtml">
4   <uxap:objectPageLayout id="dynamicPageId" showTitleInHeaderContent="true" alwaysShowContentHeader="false"
5     preserveHeaderStateOnScroll="false" headerContentPinnable="true" isChildPage="true" enableLazyLoading="false">
6     <uxap:headerTitle>
7       <uxap:objectPageDynamicHeaderTitle>
8         <uxap:expandedHeading>
9           <title text="({Carrname})"/>
10          <uxap:expandedHeading>
11            <title text="({Carrname})"/>
12            <uxap:snappedHeading>
13              <title text="({Carrname})"/>
14            <uxap:snappedContent>
15              <flexBox alignItems="start" justifyContent="SpaceBetween" id="idCarrierDetails">
16                <items>
17                  <layout:HorizontalLayout allowWrapping="true">
18                    <layout:verticalLayout class="sapUiMediumMarginEnd">
19                      <objectAttribute title="({i18n:currLabelText}" text="({Currcode})"/>
20                      <objectAttribute title="({i18n:urlLabelText}" text="({Url!})"/>
21                    </layout:verticalLayout>
22                  </layout:HorizontalLayout>
23                </items>
24              </FlexBox>
25              <core:ExtensionPoint name="ux410_extension"/>
26            </uxap:expandedContent>
27          <uxap:navigationActions></uxap:navigationActions>
28        <uxap:objectPageDynamicHeaderTitle>
29          <uxap:headerTitle>
30            <uxap:sections></uxap:sections>
31          <uxap:objectPageLayout>
32        </mvc:View>
33      
```

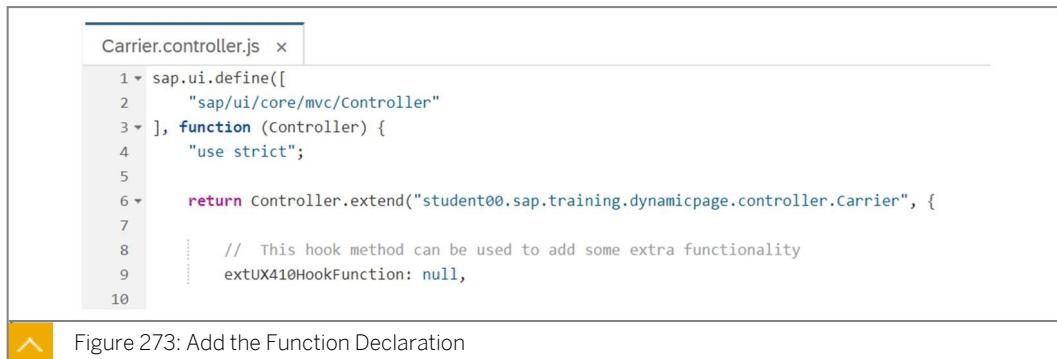
Figure 272: Add Extension

3. Add a controller `hook` function with the name `extUX410HookFunction` to the controller of view `Carrier`. Call, if it exists, the `hook` function at the end of the `onInit` function of the controller. Add the following documentation before the invocation.

Table 72: `extUX410HookFunction` Properties and Documentation

JSDoc property	Documentation
<code>@ControllerHook</code>	This hook method can be used to add some extra functionality
<code>@callback</code>	<code>student##.sap.training.dynamicpage.controller.Carrier~extUX410HookFunction</code>
<code>@return</code>	<code>{void}</code>

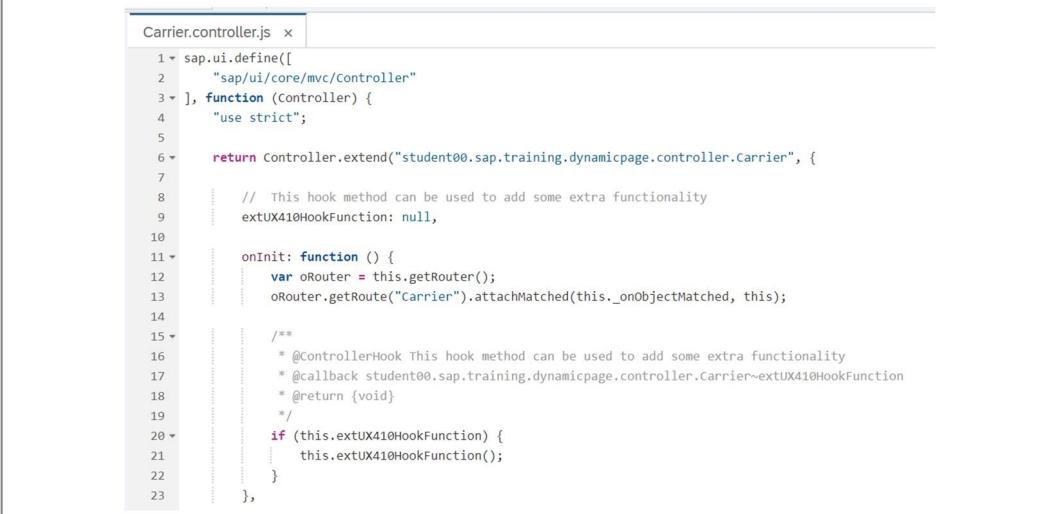
- a) Open the controller `Carrier.controller.js` located in the controller folder of your project.
- b) Add the `extUX410HookFunction` declaration to the controller and assign `null`. Use your own user group number.



```
Carrier.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Carrier", {
7
8     // This hook method can be used to add some extra functionality
9     extUX410HookFunction: null,
10
11   });
12
13 });
14 
```

Figure 273: Add the Function Declaration

- c) Go to the `onInit` function of the controller and check that the function `extUX410HookFunction` exists. Call the function. Add the listed comments before the function.



```

Carrier.controller.js x
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ], function (Controller) {
4   "use strict";
5
6   return Controller.extend("student00.sap.training.dynamicpage.controller.Carrier", {
7
8     // This hook method can be used to add some extra functionality
9     extUX410HookFunction: null,
10
11   onInit: function () {
12     var oRouter = this.getRouter();
13     oRouter.getRoute("Carrier").attachMatched(this._onObjectMatched, this);
14
15   /**
16    * @ControllerHook This hook method can be used to add some extra functionality
17    * @callback student00.sap.training.dynamicpage.controller.Carrier~extUX410HookFunction
18    * @return {void}
19    */
20   if (this.extUX410HookFunction) {
21     this.extUX410HookFunction();
22   }
23 },

```

Figure 274: Add Comments to Function

- d) Save the changes.

#### 4. Deploy your application.

Table 73: Application Properties and Values

Property	Value
Application name	ZUX410_App##
Description	Flight app
System	FSD_100 - FSD
Package	ZTRAIN_##
Transport request	Choose the transport request provide by your teacher.

- a) From the context menu of your project, select *Deploy* → *Deploy to SAPUI5 ABAP Repository*.
- b) Choose the system *FSD\_100 – FSD* from the list of available systems in the upcoming dialog. Ensure that the radio button *Deploy a new application* is selected and choose *Next*.  
Enter your username **Train-##** and password if requested.
- c) Enter the value **ZUX410\_App##** in the *Name* field and the value **Flight app** in the *Description* field. Browse to choose the package.
- d) Search for your package *ZTRAIN\_##* from the list and choose *OK*.  
Ensure you use your group packages. The packages listed may vary.
- e) Choose *Next*.

- f) Select the *transport request* provided by your teacher. The dialog transport request selection may vary.
- g) Choose *Finish*
- h) Wait until your project is deployed.

### Extend the Deployed Application.

1. Extend your deployed application **ZUX410\_APP##** (system is FSD). Create an extension project with the name **ZUX410\_APP##Extension**. Hide the UI control with ID **idCarrierDetails** by using the extension pane. Create a view extension for the extension point **ux410\_extension** using the extension pane. Add the code provided in the example for the extension. Test your extended application.



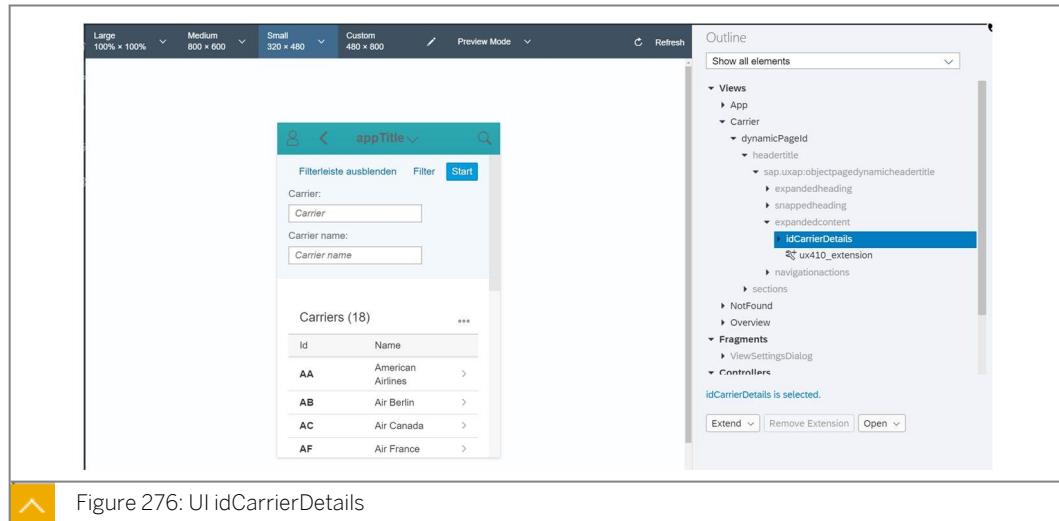
```

Carrier_ux410_extensionCusto... x
1 <core:FragmentDefinition xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:layout="sap.ui.layout"
3   xmlns:uxap="sap.uxap" xmlns:html="http://www.w3.org/1999/xhtml">
4   <FlexBox alignItems="Start" justifyContent="SpaceBetween">
5     <items>
6       <layout:HorizontalLayout allowWrapping="true">
7         <layout:VerticalLayout class="sapUiMediumMarginEnd">
8           <ObjectAttribute title="{i18n>currLabelText}" text="{CurrCode}" />
9           <ObjectAttribute title="{i18n>urlLabelText}" text="{Url}" />
10          <ObjectAttribute title="Carrier id" text="{Carrid}" />
11          <ObjectAttribute title="Carrier name" text="{Carrname}" />
12        </layout:VerticalLayout>
13      </layout:HorizontalLayout>
14    </items>
15  </FlexBox>
16 </core:FragmentDefinition>

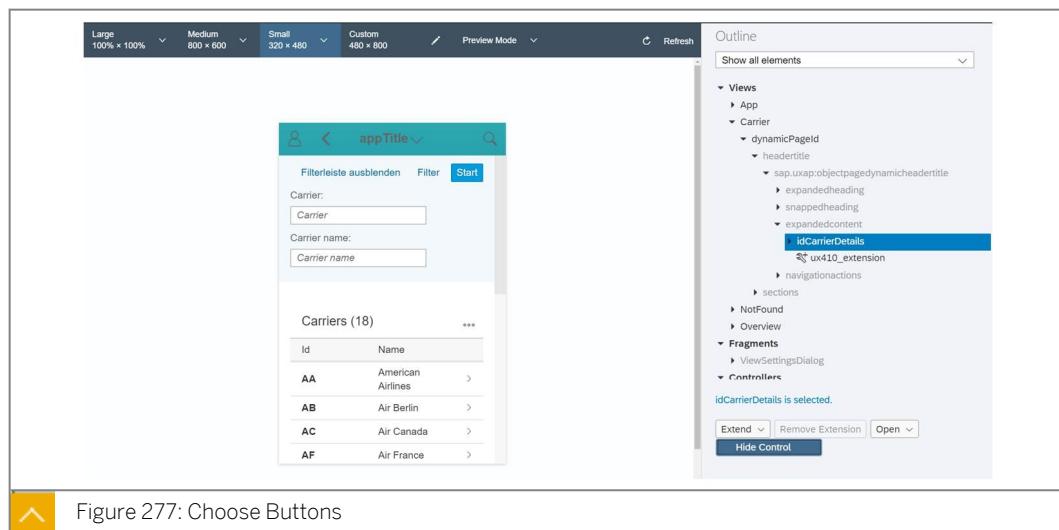
```

Figure 275: Extension Code

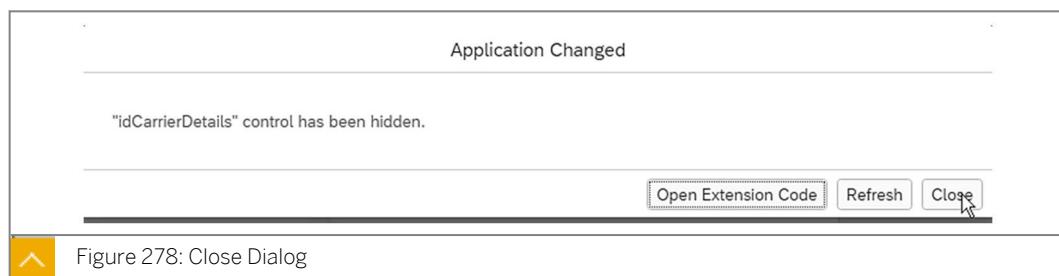
- a) Select the *Workspace* folder from your project explorer and from the context menu choose *New → Extension Project*.
- b) In the *Select Application* selection box of the new dialog that appears, choose *SAPUI5 ABAP Repository*.
- c) Select the system FSD from the list of available systems and enter **zux410\_** in the *Search* field. Select your application from the list of available applications and choose *OK*. Enter your user **Train-##** and password if requested.
- d) Leave the upcoming dialog unchanged and choose *Next*.
- e) Choose *Finish* to confirm the project creation. Select the *Open extension project in extensibility pane* checkbox.
- f) The extension pane opens when it is created. Enter your credentials for the front-end server and choose *Sign In*.
- g) Expand the *Carrier* view in the outline and drill down to the UI control with the ID **idCarrierDetails**.



h) Choose the *Extend-Selection* button and *Hide Control*.



i) Close the dialog box that appears.



j) Choose the extension point *ux410\_extension* and the *Extend View/Fragment* button from the selection.

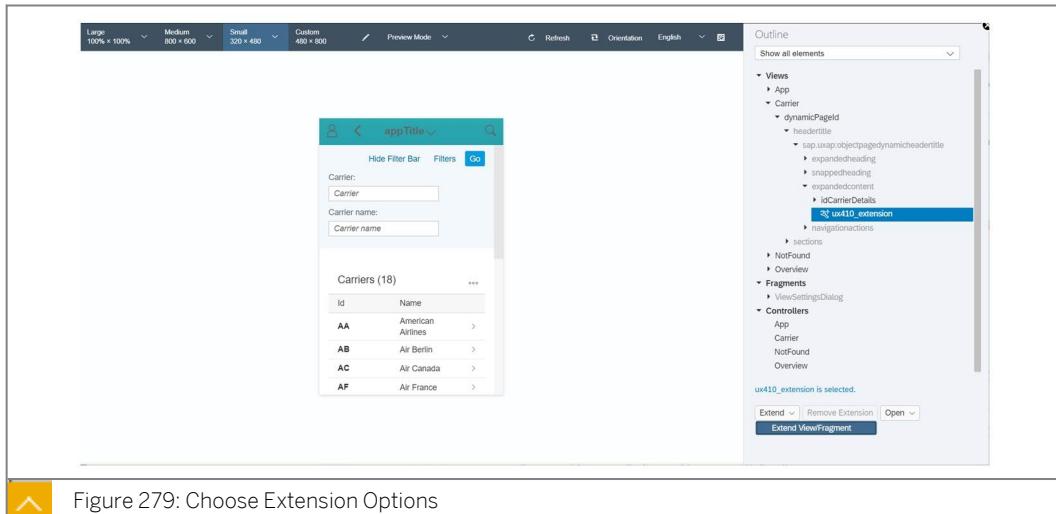


Figure 279: Choose Extension Options

k) Choose *Open Extension Code*.

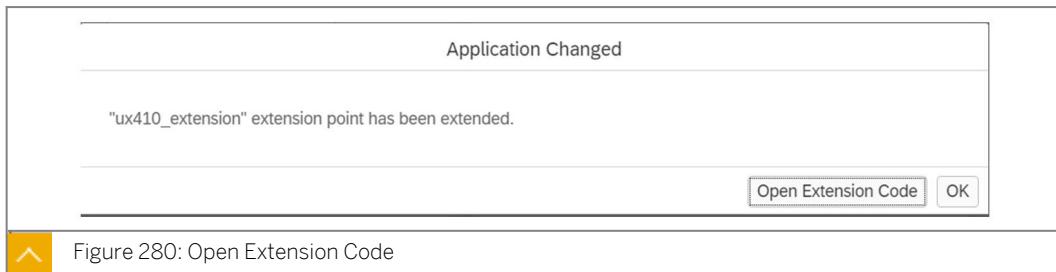


Figure 280: Open Extension Code

I) After creation, the view extension is opened by the SAP Web IDE.

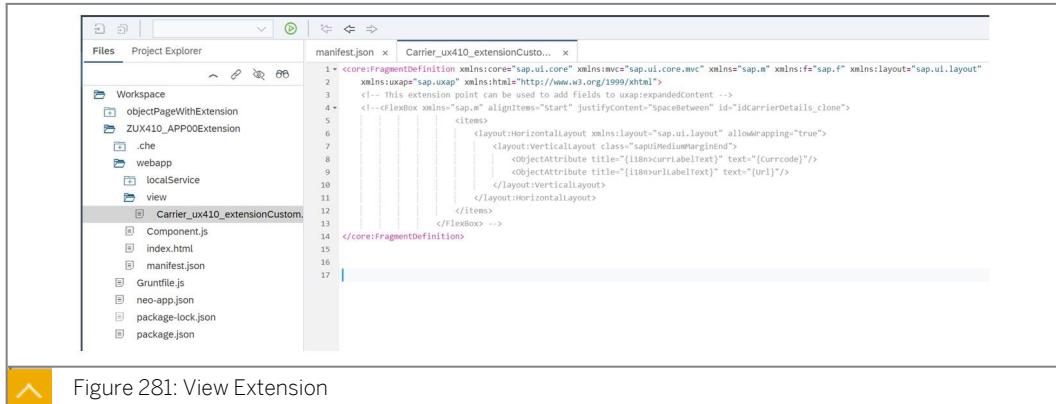


Figure 281: View Extension

m) Add view fragments details.



```

1 <core:FragmentDefinition xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
2   xmlns:f="sap.f" xmlns:layout="sap.ui.layout"
3   xmlns:uxap="sap.uxap" xmlns:html="http://www.w3.org/1999/xhtml">
4   <FlexBox alignItems="Start" justifyContent="SpaceBetween">
5     <items>
6       <layout:HorizontalLayout allowWrapping="true">
7         <layout:VerticalLayout class="sapUiMediumMarginEnd">
8           <ObjectAttribute title="{i18n>currLabelText}" text="{Currcode}" />
9           <ObjectAttribute title="{i18n>urlLabelText}" text="{Url}" />
10          <ObjectAttribute title="Carrier id" text="{Carrid}" />
11          <ObjectAttribute title="Carrier name" text="{Carrname}" />
12        </layout:VerticalLayout>
13      </layout:HorizontalLayout>
14    </items>
15  </FlexBox>
16 </core:FragmentDefinition>

```

Figure 282: View Fragments

- n) Save your changes.
- o) To test your application select the `index.html` file and from the context menu, choose `Run → Run as Web Application`.
- p) Allow SAP Web IDE to store files on your device when requested.
- q) Choose the tile `ZUX410_APP##Extension`.
- r) Enter your credentials for the front-end server.
- s) Select a carrier from the list and check the result.

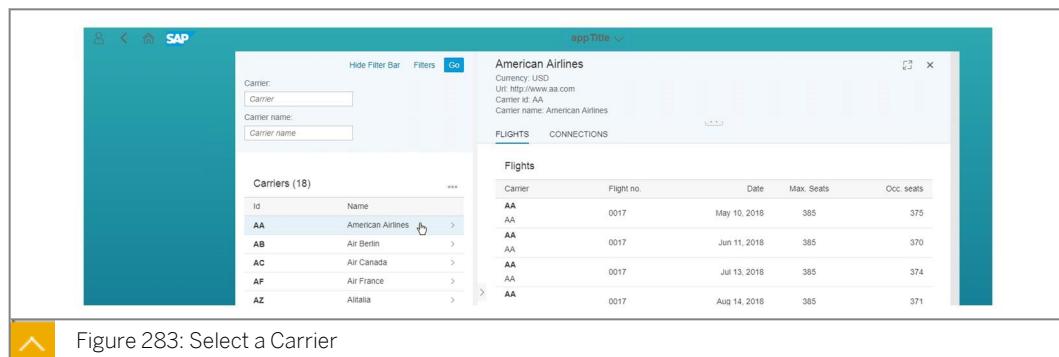


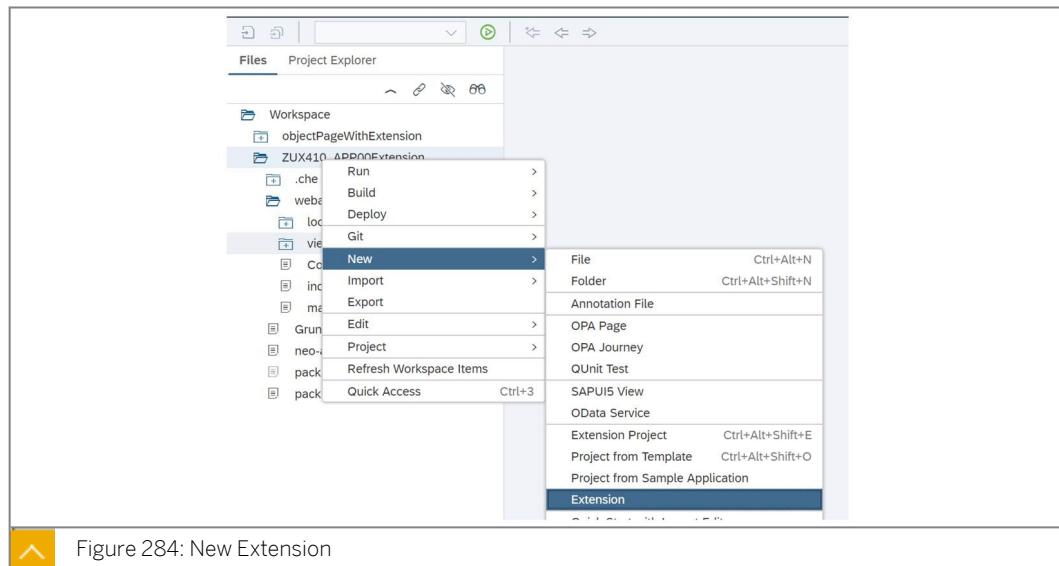
Figure 283: Select a Carrier

2. Add an i18n-resource customizing extension to your project and enter the new text properties listed. Replace the hard coded text literals located in the view extension. Test your application.

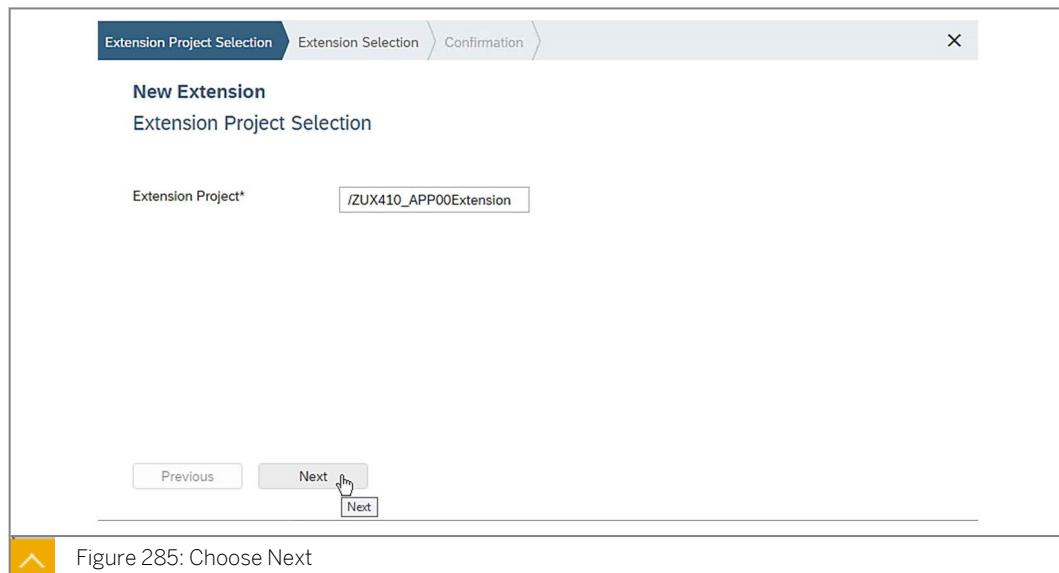
Table 74: Text Properties

Key	Value	Replace the following hard coded string in view extension
idLabelText	This is the carrier id	Carrier id
nameLabelText	This is the carrier name	Carrier name

- a) Choose your project root folder and from the context menu, choose `New → Extension`.



b) Confirm the selection by choosing *Next* in the dialog box.



c) Choose *i18n Resource Text Customization* and *Finish*.

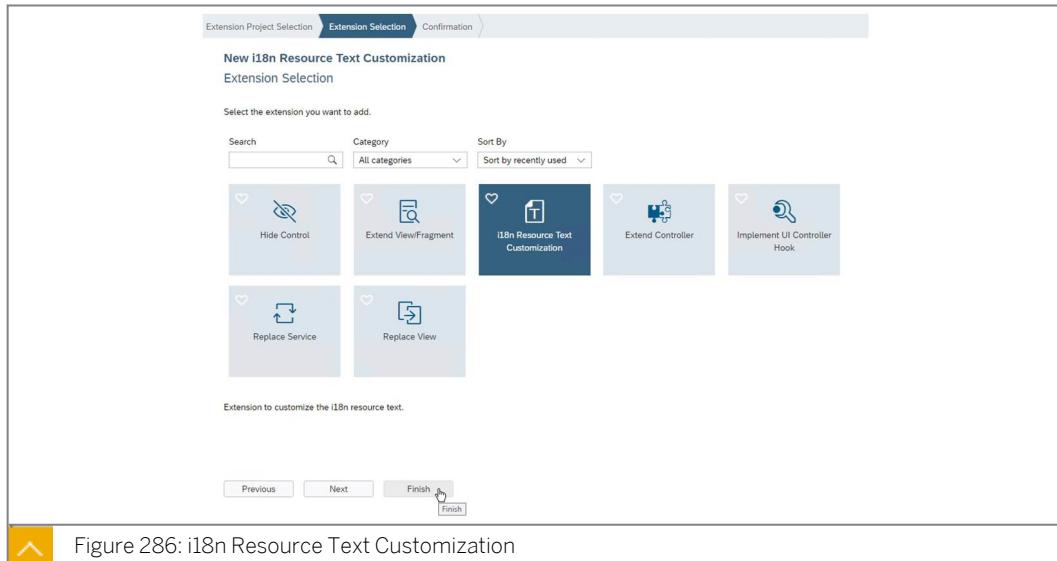


Figure 286: i18n Resource Text Customization

d) Change the value key of the i18n resource file.

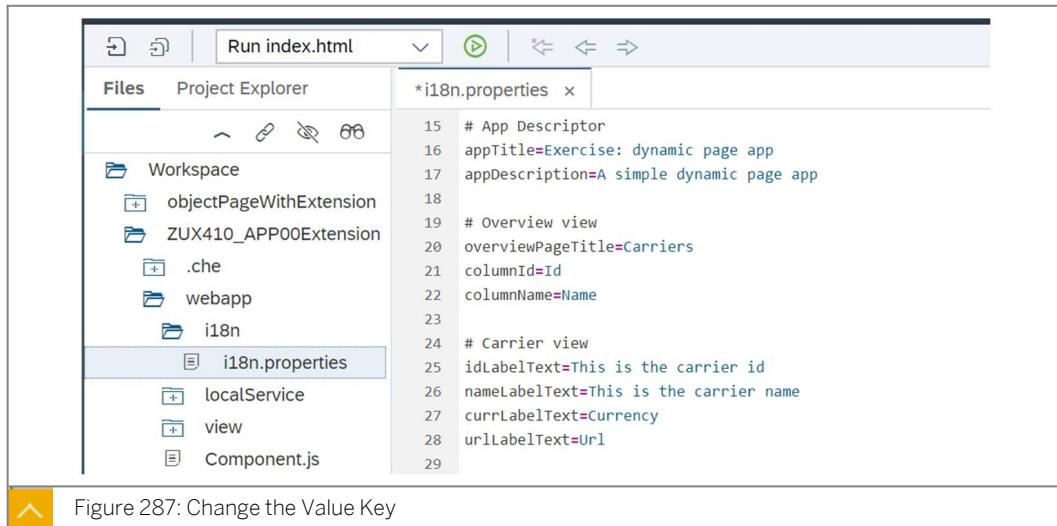


Figure 287: Change the Value Key

e) Add the i18n resource binding to the view extension.

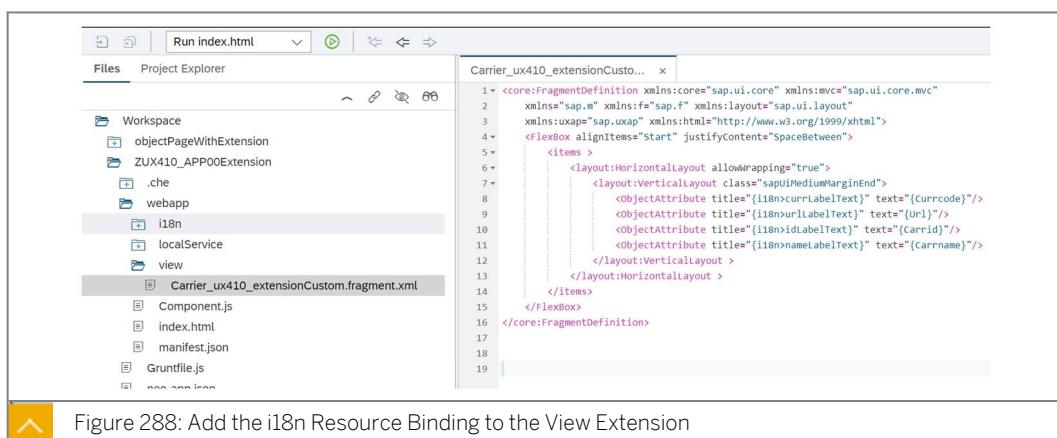
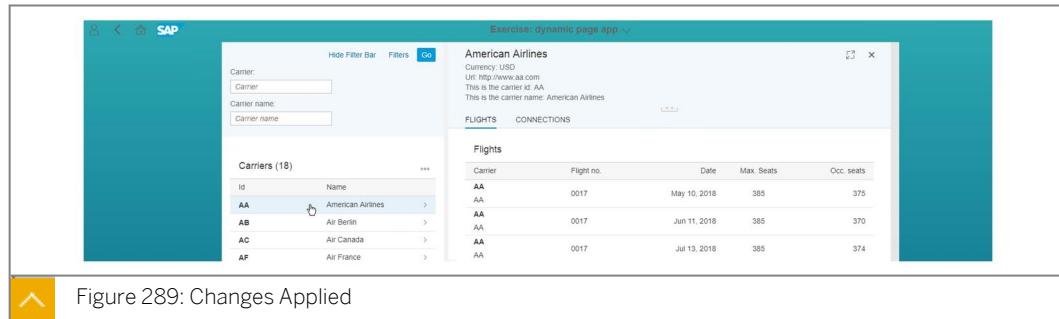


Figure 288: Add the i18n Resource Binding to the View Extension

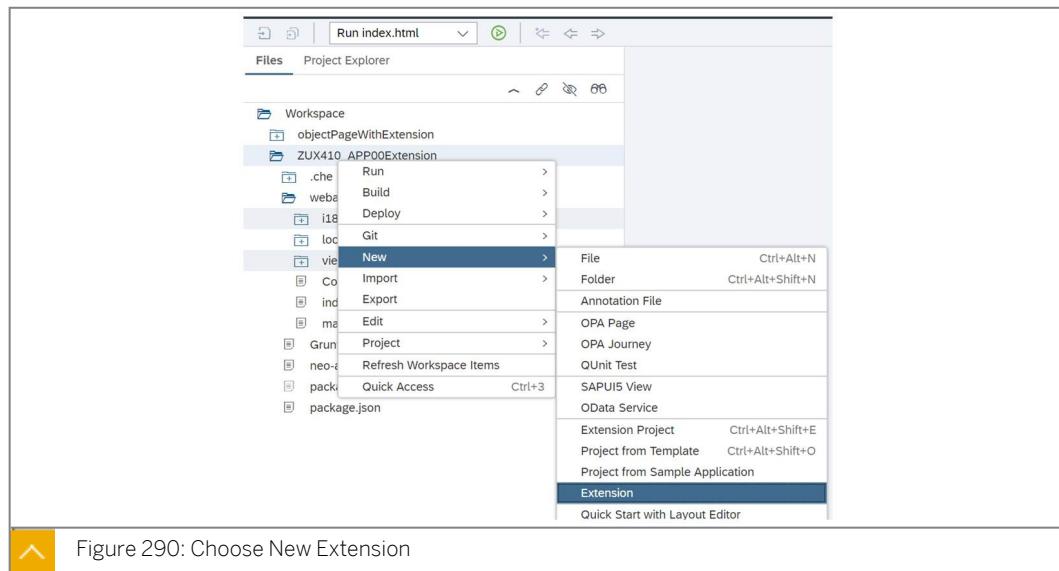
f) Save your work.

g) Start your application again to test if the changes are applied.

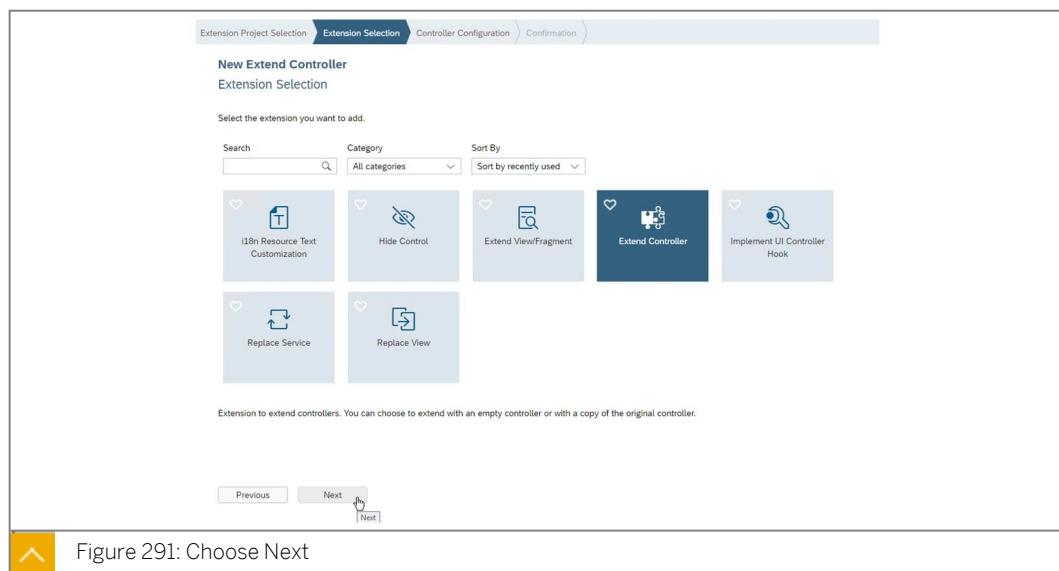


3. Implement a controller extension for the controller of *Carrier* view and implement the function `extUX410HookFunction`. The function shows an `sap.m.MessageToast` with the message **Called from Init**. Test if your changes are applied.

a) In the root folder of your project, from the context menu, choose *New → Extension* .



b) Confirm by choosing *Next*.



c) Choose the controller of the *Carrier* view from the available list of controllers. Also choose *Empty Controller* from the *Replace with selection* list and choose *Finish*.

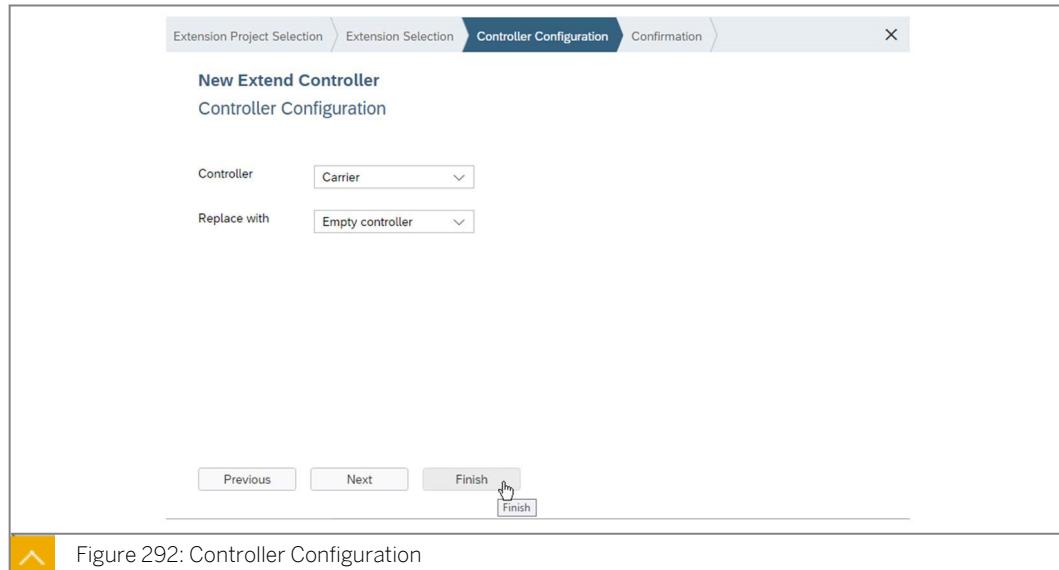


Figure 292: Controller Configuration

d) Implement the hook function `extUX410HookFunction`.



Figure 293: Hook Implementation

e) Test your application.

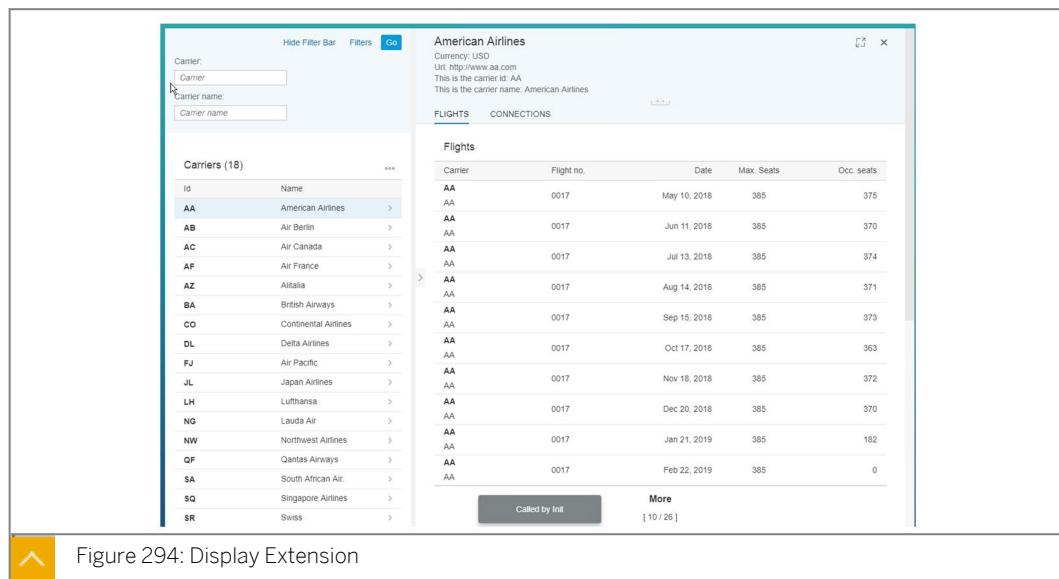


Figure 294: Display Extension

f) Change the `sap_bluecrystal` theme to `sap_belize`.



## Note:

Should you wish to change the sap\_bluecrystal theme in your extension project, open the `index.html` file and navigate to the bootstrapping.

```
<script id="sap-ui-bootstrap"
       src="resources/sap-ui-core.js"
       data-sap-ui-libs="sap.ushell, sap.collaboration, sap.m, sap.ui.layout"
       data-sap-ui-theme="sap_belize"
       data-sap-ui-xx-bindingSyntax="complex"
       data-sap-ui-compatVersion="edge"
       data-sap-ui-resourceroots='{"student00.sap.training.dynamicpage.ZUX410_APP00Extension": "./"}'
       data-sap-ui-frameOptions='allow' // NON-SECURE setting for testing environment
     </script>
```

Figure 295: Change Theme Option

- g) Restart the application to see the SAP Fiori Launchpad of the SAP Web IDE in the `sap_belize` theme.
- h) Select the tile.

The screenshot shows the SAP Fiori Launchpad with the 'American Airlines' tile selected. The tile displays flight information for American Airlines. The table data is as follows:

Carrier	Flight no.	Date	Max. Seats	Occ. seats
AA	0017	May 10, 2018	385	375
AA	0017	Jun 11, 2018	385	370
AA	0017	Jul 13, 2018	385	374
AA	0017	Aug 14, 2018	385	371
AA	0017	Sep 15, 2018	385	373
AA	0017	Oct 17, 2018	385	363

Figure 296: Extension View

## Unit 14 Exercise 15

# Implement List Report using SAP Fiori Elements



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.



### Note:

Sometimes after the annotations are changed in the **ABAP CDS**, they are not reflected in the **SAP Fiori Elements Application**, or in the **SAP Web IDE**. If this happens, clean the browser caches and restart the application.

Create an ABAP project in *Eclipse JEE*.

1. Create an eclipse project in *Eclipse JEE* with the following values.

Table 75: Project Values

Property	Value
system	S4D
client	100
user	Train-## (replace ## with your group number)
password	The password for your train-## user from the back-end system.
project name	Use the default project name S4D_100_train##_en (replace ## with your group number)
package	ZTRAIN_## (replace ## with your group number)

### Task 1: Create a CDS View

1. Add a new CDS view with the following values.

Table 76: CDS View Values

Property	Value
name	ZUX410_C_FlightUI## (replace ## with your group number)
description	UX410 Fiori Elements ##

Property	Value
package	ZTRAIN_## (replace ## with your group number)
transport Request	Use the transport request provided by the trainer.
template for creating New Data Definition	No template

2. Copy the content of the file `S:\Courses\UX410_20\Templates\ZUX410_C_FLIGHTUI##.txt` to the newly created ABAP CDS view. Replace ## with your group number.
3. Activate your changes.

#### Task 2: Register the OData Service on the front-end server

1. For the next step, you need to register the OData Service on the front-end server of the training environment. Use the value `ZUX410_C_FLIGHTUI##_CDS` as a technical service name.

#### Task 3: Create a project in the SAP Web IDE.

1. Use the SAPUI5 project template *List Report Application* to create a new SAPUI5 project. Use the following settings.

Table 77: SAPUI5 Project Settings

Property	Value
Project name	fiorielements
Title	FlightExercise
Namespace	student##.sap.training
Service Catalog	FSD_100 - FSD
OData-Service	ZUX410_C_FLIGHTUI##_CDS (replace ## with your group number)
Description	<leave empty>
Annotation Selection	ZUX410_C_FLIGHTUI##_CDS_VAN (replace ## with your group number)
OData Collection	ZUX410_C_FLIGHTUI## (replace ## with your group number)
Smart Variant Management for List Report	true
Flexible Column Layout	true

2. Test your application. The application shows no data in the list initially. Select columns from the list of available columns.

#### Task 4: Show flights in the List Report Using UI Annotations

In the next step, you enhance your ABAP-CDS to display columns in the list report.

1. Add annotation for the `zux410_C_FLIGHTUI##` entity type using the term `UI.headerInfo`. Use the information in the table to describe basic information of the data you want to display.

Table 78: Data to Display

Property	Datatype	Value
<code>typeNamePlural</code>	string	Flights
<code>typeName</code>	string	Flight

2. Use the term `UI.lineItem` to add the following information to your CDS and activate your changes.

Table 79: UI.lineItem Information

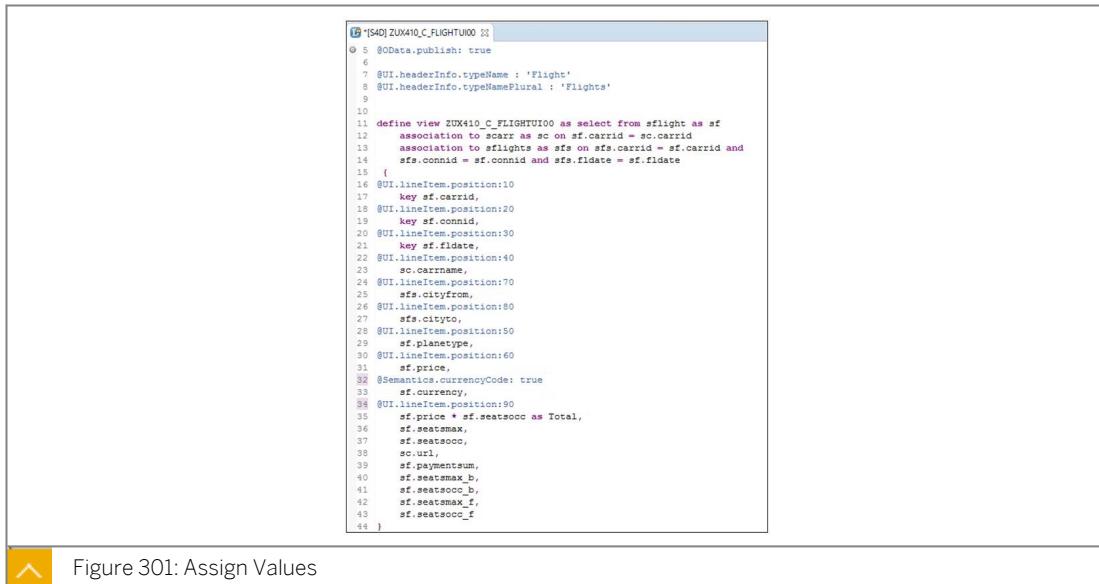
Property	Position
<code>carrid</code>	10
<code>connid</code>	20
<code>fldate</code>	30
<code>carrname</code>	40
<code>planetype</code>	50
<code>price</code>	60
<code>cityfrom</code>	70
<code>cityto</code>	80

3. To test your implementation, go to the *SAP Web IDE* and start the application created in the previous task.

### Task 5: Add Semantic Information to the ABAP-CDS

In the next task you add the *Total* field to the list report and assign semantic information to the *Total* field of your ABAP-CDS.

1. Add the semantic term `currencyCode` to the `currency` field of your projection list and assign the value `true`.
2. Add the annotation `@Semantics.currencyCode` to the `Currency` field of your projection list and assign the boolean value `true`.



```

5 @Data.publish: true
6
7 @UI.headerInfo.typeName : 'Flight'
8 @UI.headerInfo.typeNamePlural : 'Flights'
9
10
11 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
12   association to scarr as sc on sf.carrid = sc.carrid
13   association to sflight as sfs on sf.sfs.carrid = sc.carrid and
14   sf.sfs.comnid = sc.comnid and sfs.fldate = sf.fldate
15 {
16   @UI.lineItem.position:10
17   key sf.carrid,
18   @UI.lineItem.position:20
19   key sf.comnid,
20   @UI.lineItem.position:30
21   key sc.comnid,
22   @UI.lineItem.position:40
23   sc.carname,
24   @UI.lineItem.position:70
25   sfs.cityfrm,
26   @UI.lineItem.position:80
27   sfs.cityto,
28   @UI.lineItem.position:50
29   sf.planetype,
30   @UI.lineItem.position:60
31   sf.price,
32   @Semantics.currencyCode: true
33   sf.currency,
34   @UI.lineItem.position:90
35   sf.price * sf.seatscco as Total,
36   sf.seatsmax,
37   sf.seatscco,
38   sc.url,
39   sf.paymentsum,
40   sf.seatsmax_b,
41   sf.seatscco_b,
42   sf.seatsmax_f,
43   sf.seatscco_f
44 }

```

Figure 301: Assign Values

3. Add the annotation `@Semantics.amount.currencyCode` to the *Total* field of your projection list and assign the string value **currency** to the term. Activate your changes.
4. Test your application in the SAP Web IDE.

### Task 6: Add Responsive Behaviour to the List Report.

In the next task you will add responsive behaviour to your application.

1. Add the following annotations and information to the fields of the projection list.

Table 80: Projection List Field Information

Field	Term	Property	Value
planetype	lineItem	importance	#HIGH
Total	lineItem	importance	#MEDIUM
price	lineItem	importance	#LOW

2. Test your application in the SAP Web IDE. Use the features of the Google Chrome Development Tools to change the form factor.

## Implement List Report using SAP Fiori Elements



Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.



Note:

Sometimes after the annotations are changed in the **ABAP CDS**, they are not reflected in the *SAP Fiori Elements Application*, or in the *SAP Web IDE*. If this happens, clean the browser caches and restart the application.

Create an ABAP project in *Eclipse JEE*.

1. Create an eclipse project in *Eclipse JEE* with the following values.

Table 75: Project Values

Property	Value
system	S4D
client	100
user	Train-## (replace ## with your group number)
password	The password for your train-## user from the back-end system.
project name	Use the default project name S4D_100_train##_en (replace ## with your group number)
package	ZTRAIN_## (replace ## with your group number)

- a) Choose *Eclipse Jee 2019-06* from the windows *Start* menu.
- b) When Eclipse opens, choose the *Workbench* button in the upper right corner.
- c) The Eclipse workbench appears.
- d) Choose *File* → *New* → *Other* from the window.
- e) Enter the value **ABAP** in the *Search* field, choose *ABAP Project* from the list, and choose *Next*.

- f) Choose *00 S4D GUI non SNC [PAS]* from the list of available systems and choose *Next*.
- g) Choose *Next* to confirm the settings in the following dialog.
- h) Enter **Client 100** and your user **Train-##** (replace ## with your group number). Enter your password for the S4D system. Choose *Next*.
- i) Add the development package **ZTRAIN-00** to the list of favourite packages by choosing the *Add* button.
- j) Enter the package name **ZTRAIN** in the *Search* field, choose the package assigned to your group from the list of found packages and choose *OK*.
- k) Create the package by choosing *Finish*.
- l) Confirm the dialog to switch the perspective to the ABAP perspective.
- m) Expand the newly created project in the new perspective. Then expand the entry *Favorite Packages*. Here you will see your package.

### Task 1: Create a CDS View

1. Add a new CDS view with the following values.

Table 76: CDS View Values

Property	Value
name	ZUX410_C_FlightUI## (replace ## with your group number)
description	UX410 Fiori Elements ##
package	ZTRAIN_## (replace ## with your group number)
transport Request	Use the transport request provided by the trainer.
template for creating New Data Definition	No template

- a) Select the entry *Favorite Packages* in your project and from the context menu, choose *New → Other → ABAP → ABAP Repository Object*.
  - b) Expand the *Core Data Services* folder and choose *Data Definition*.
  - c) Enter the value **ZUX410\_C\_FlightUI##** in the *Name* field and in the *description* field, enter **UX410 Fiori Elements ##** and choose *Next*.
  - d) Select the transport request provided by your trainer and choose *Next*.
  - e) Deselect the *Use the selected template* checkbox.
  - f) Choose *Finish*.
2. Copy the content of the file `S:\Courses\UX410_20\Templates\ZUX410_C_FlightUI##.txt` to the newly created *ABAP CDS view*. Replace ## with your group number.

- a) Open *File Explorer* and navigate to `S:\Courses\UX410_20\Templates\ZUX410_C_FLIGHTUI##.txt`.
- b) Right click on the file name and choose *Sappad* from the context menu to open the file.
- c) Copy the content from the file into the newly created *CDS* view.
- d) Replace the `##` with your group number.

3. Activate your changes.

- a) Activate your changes by choosing the  icon on the toolbar
- b) Eclipse shows a warning. This is correct. Ignore the warning and proceed.

**Task 2: Register the OData Service on the front-end server**

1. For the next step, you need to register the *OData Service* on the front-end server of the training environment. Use the value `ZUX410_C_FLIGHTUI##_CDS` as a technical service name.
  - a) Open *SAP GUI*.
  - b) Choose the entry `00 FSD` from the folder `00 Development`.
  - c) Enter the credentials for the FSD System of your training environment. Enter the user `train-##` and your password.
  - d) Expand your *User* menu and expand the entry *SAP Gateway*. Double-click *Activate and Maintain Services* from the list.
  - e) After the transaction has started, choose the *Add Service* button.
  - f) Enter the value `S4HANA` in the *system alias* field (you can use F4-value help). Enter `z*` in the *Technical Service Name* field, and choose *Enter*.
  - g) Select `ZUX410_C_FLIGHTUI##_CDS` from the list of available services, and choose the *Add Service* button.
  - h) Assign your changes to the *Package ZTRAIN\_##* (replace `##` with your group number).
  - i) During the generation of the necessary artefacts you will be asked to assign the artefacts to a transport request. Use the request assigned to your user. (The dialog for assignment might pop up more than once).
  - j) Choose *OK* in the success dialog.
  - k) Switch to the initial screen of transaction `/IWFND/MAINT_SERVICE` and check your services were added successfully. Use *F3* or the *Back* button of *SAP GUI*.

**Task 3: Create a project in the SAP Web IDE.**

1. Use the SAPUI5 project template *List Report Application* to create a new SAPUI5 project. Use the following settings.

Table 77: SAPUI5 Project Settings

Property	Value
Project name	fiorielements
Title	FlightExercise
Namespace	student##.sap.training
Service Catalog	FSD_100 - FSD
OData-Service	ZUX410_C_FLIGHTUI##_CDS (replace ## with your group number)
Description	<leave empty>
Annotation Selection	ZUX410_C_FLIGHTUI##_CDS_VAN (replace ## with your group number)
OData Collection	ZUX410_C_FLIGHTUI## (replace ## with your group number)
Smart Variant Management for List Report	true
Flexible Column Layout	true

- a) Log on to the SAP Web IDE Full-Stack in Google Chrome and select the workspace folder and choose from the context menu *New* → *Project from Template*.
  - b) Select the *List Report Application* tile and choose *Next*.
  - c) Enter the project settings defined in the table and choose *Next*.
  - d) Choose the system *FSD\_100 – FSD* from the list of available service catalog systems.
  - e) Enter your user credentials for the front-end server and choose *Sign In*.
  - f) Enter **zux** in the *Search* field and choose your OData Service *ZUX410\_C\_FLIGHTUI##\_CDS* from the search result. Wait until the *Next* button is enabled and choose *Next*.
  - g) In the next dialog, select the *ZUX410\_C\_FLIGHTUI##\_CDS\_VAN* checkbox and choose *Next*.
  - h) Choose *ZUX410\_C\_FLIGHTUI##* as OData collection, check the *select* box for *Flexible Column Layout* and ensure the *select* box for *Smart Variant Management for List Report* is selected. Choose *Finish*.
  - i) Your new project appears in the workspace of your SAP Web IDE.
2. Test your application. The application shows no data in the list initially. Select columns from the list of available columns.
    - a) To test the application, select the project root folder, and choose from the context menu *Run* → *Run as SAP Fiori Launchpad Sandbox*.
    - b) Choose *Store files on the device* in the dialog and enter your user credentials for the front-end server. Enter **train-##** using your group number and password and choose *Sign In*.

c) Choose the cogwheel  symbol in the application.

- d) Select some of the columns in the new dialog, and choose *OK*.
- e) Choose the *Go* button to execute the server request.
- f) The table now shows some flight data.

#### Task 4: Show flights in the List Report Using UI Annotations

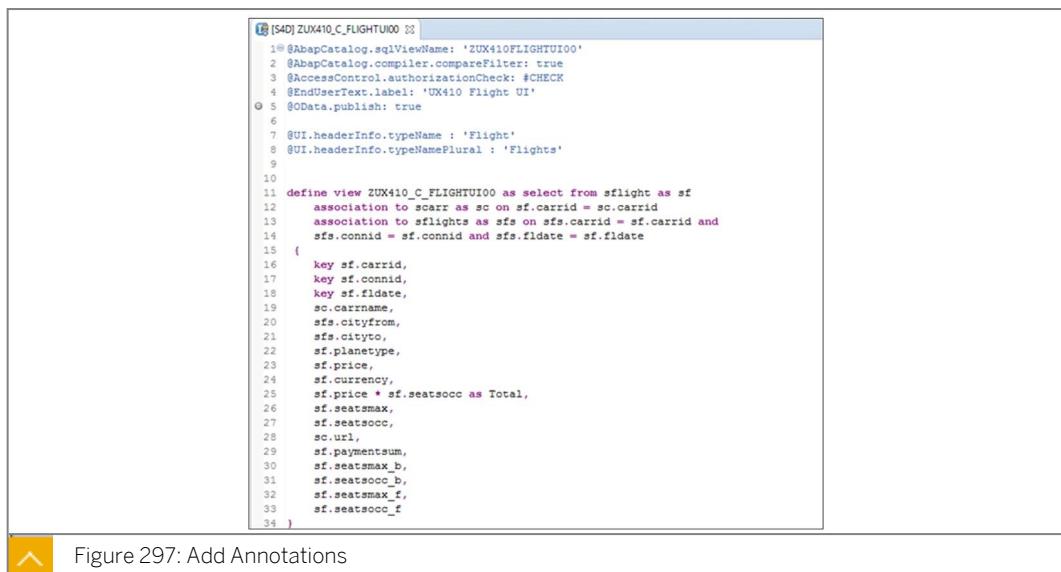
In the next step, you enhance your ABAP-CDS to display columns in the list report.

1. Add annotation for the `ZUX410_C_FLIGHTUI##` entity type using the term `UI.headerInfo`. Use the information in the table to describe basic information of the data you want to display.

Table 78: Data to Display

Property	Datatype	Value
<code>typeNamePlural</code>	string	Flights
<code>typeName</code>	string	Flight

- a) Start *Eclipse Oxygen* as shown in the first task of the exercise, if not already open.
- b) Open the created CDS located in the package `ZTRAIN_##`, if not already displayed.
- c) Add the listed annotations to your CDS.



The screenshot shows the Eclipse Oxygen interface with the code editor open. The code is a SQL view definition for 'ZUX410\_C\_FLIGHTUI00'. It includes annotations for the entity type 'UI.headerInfo' with properties 'typeName' set to 'Flight' and 'typeNamePlural' set to 'Flights'. The code also defines associations between tables 'sflight', 'sc', and 'sfs' based on their primary keys.

```

1@<*> [540] ZUX410_C_FLIGHTUI00 <*>
2 @sapCatalog.sqlViewName: 'ZUX410FLIGHTUI00'
3 @sapCatalog.compiler.compareFilter: true
4 @accessControl.authorizationCheck: #CHECK
5 @endUserText.label: 'UX410 Flight UI'
6
7 @UI.headerInfo.typeName : 'Flight'
8 @UI.headerInfo.typeNamePlural : 'Flights'
9
10
11 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
12   association to sc as sc on sf.carrid = sc.carrid
13   association to sfs as sfs on sfs.carrid = sf.carrid and
14     sfs.connid = sf.connid and sfs.fldate = sf.fldate
15  {
16    key sf.carrid,
17    key sf.connid,
18    key sf.fldate,
19    sc.carrname,
20    sfs.cityfrom,
21    sfs.cityto,
22    sf.planetyp,
23    sf.price,
24    sf.currency,
25    sf.price * sf.seatsocc as Total,
26    sf.seatsmax,
27    sf.seatsocc,
28    sc.url,
29    sf.paymentsum,
30    sf.seatsmax_b,
31    sf.seatsocc_b,
32    sf.seatsmax_f,
33    sf.seatsocc_f
34  }

```

Figure 297: Add Annotations

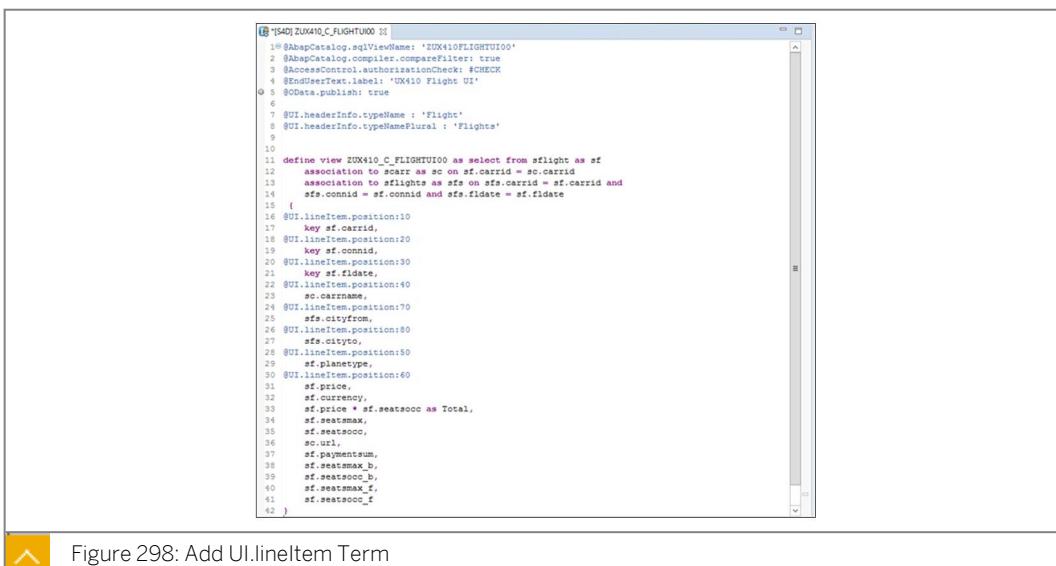
- d) Save your changes.
- 2. Use the term `UI.lineItem` to add the following information to your CDS and activate your changes.

Table 79: UI.lineItem Information

Property	Position
<code>carrid</code>	10

Property	Position
connid	20
fldate	30
carrname	40
planetype	50
price	60
cityfrom	70
cityto	80

- a) Add the UI.lineItem term to the fields.



The screenshot shows the SAP Web IDE interface with a code editor window. The code is a Fiori List Report configuration in JSON. The UI.lineItem term is added to the fields as follows:

```

10 15401 Z0X410_C_FLIGHTRUI00 {
11   "@sap-catalog:entityName": "Z0X410_FLIGHTRUI00",
12   "@sapCatalog:compiler:composeFilter: true",
13   "@AccessControl:authorizationCheck: #CHECK",
14   "@EndUserText:label: 'Z0X410 Flight UI'",
15   "@Data.publish: true",
16   "@UI:headerInfo:typeName: 'Flight'",
17   "@UI:headerInfo:typeNamePlural: 'Flights'",
18   "@UI.lineItem:position:10",
19   "  key sf.carrid",
20   "  key sf.connid",
21   "  key sf.fldate",
22   "  key sf.planetype",
23   "  key sf.carrname",
24   "  key sf.cityfrom",
25   "  key sf.cityto",
26   "@UI.lineItem:position:20",
27   "  key sf.url",
28   "  key sf.persentsum",
29   "  key sf.seatmax_a",
30   "  key sf.seatmax_b",
31   "  key sf.seatmax_c",
32   "  key sf.seatmax_d",
33   "  key sf.price + sf.seatsoco as Total",
34   "  key sf.seatmax",
35   "  key sf.seatsoco",
36   "  key sf.url",
37   "  key sf.persentsum",
38   "  key sf.seatmax_a",
39   "  key sf.seatmax_b",
40   "  key sf.seatmax_c",
41   "  key sf.seatsoco"
42 }

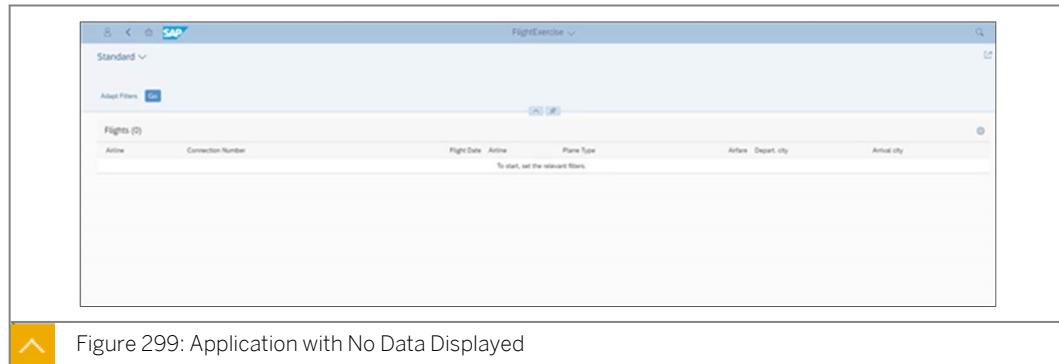
```

Figure 298: Add UI.lineItem Term

- b) Save your changes.

- c) Activate your changes with the  icon.

- To test your implementation, go to the SAP Web IDE and start the application created in the previous task.
  - Open the SAP Web IDE Fullstack if not already open.
  - Start your application as described in the previous task.
  - Confirm the application is allowed to store files on your device.
  - The application displays with no data, which is correct.



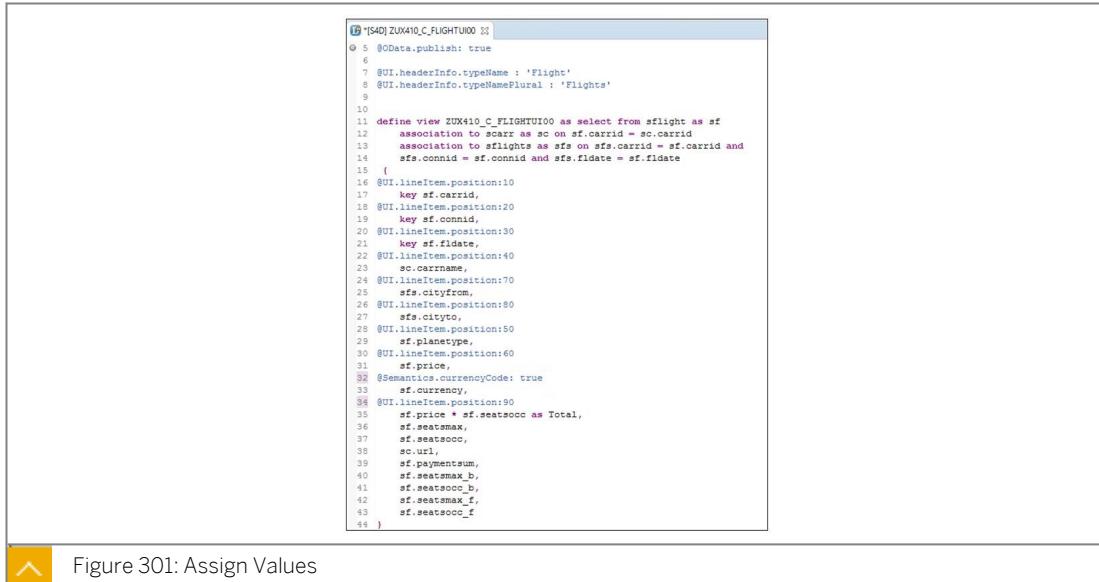
- e) Choose Go to send the get request.
- f) The results appear in the table when the request is processed successfully.

Figure 300: Results Table

### Task 5: Add Semantic Information to the ABAP-CDS

In the next task you add the *Total* field to the list report and assign semantic information to the *Total* field of your ABAP-CDS.

1. Add the semantic term `currencyCode` to the `currency` field of your projection list and assign the value *true*.
  - a) Start Eclipse as shown in the first task of the exercise, if not already open.
  - b) Open the created CDS located in the package `ZTRAIN_##`, if not already displayed.
  - c) Add the term `lineItem` to the calculated *Total* field and assign the *position* *90* to the `lineItem`.
2. Add the annotation `@Semantics.currencyCode` to the `Currency` field of your projection list and assign the boolean value *true*.



```

5 #odata.publish: true
6
7 #UI.headerInfo.typeName : 'Flight'
8 #UI.headerInfo.typeNamePlural : 'Flights'
9
10
11 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
12   association to scarr as sc on sf.carrid = sc.carrid
13   association to flights as sfs on sfs.carrid = sf.carrid and
14   sfs.connid = sf.connid and sfs.fldate = sf.fldate
15 {
16   #UI.lineitem.position:10
17   key sf.carrid,
18   #UI.lineitem.position:20
19   key sf.connid,
20   #UI.lineitem.position:30
21   sf.fldate,
22   #UI.lineitem.position:40
23   sc.carrname,
24   #UI.lineitem.position:70
25   sfs.cityfrom,
26   #UI.lineitem.position:80
27   sfs.cityto,
28   #UI.lineitem.position:50
29   sf.planetype,
30   #UI.lineitem.position:60
31   sf.price,
32   #Semantics.currencyCode: true
33   sf.currency,
34   #UI.lineitem.position:90
35   sf.price * sf.seatsocc as Total,
36   sf.seatsmax,
37   sf.seatsocc,
38   sc.url,
39   sf.paymentsum,
40   sf.seatsmax_b,
41   sf.seatsocc_b,
42   sf.seatsmax_f,
43   sf.seatsocc_f
44 }

```

Figure 301: Assign Values

3. Add the annotation `@Semantics.amount.currencyCode` to the *Total* field of your projection list and assign the string value **currency** to the term. Activate your changes.
  - a) Add the annotation to the total field.



```

5 #odata.publish: true
6
7 #UI.headerInfo.typeName : 'Flight'
8 #UI.headerInfo.typeNamePlural : 'Flights'
9
10
11 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
12   association to scarr as sc on sf.carrid = sc.carrid
13   association to flights as sfs on sfs.carrid = sf.carrid and
14   sfs.connid = sf.connid and sfs.fldate = sf.fldate
15 {
16   #UI.lineitem.position:10
17   key sf.carrid,
18   #UI.lineitem.position:20
19   key sf.connid,
20   #UI.lineitem.position:30
21   key sf.price,
22   #UI.lineitem.position:40
23   sc.carrname,
24   #UI.lineitem.position:70
25   sfs.cityfrom,
26   #UI.lineitem.position:80
27   sfs.cityto,
28   #UI.lineitem.position:50
29   sf.planetype,
30   #UI.lineitem.position:60
31   sf.price,
32   #Semantics.currencyCode: true
33   sf.currency,
34   #Semantics.amount.currencyCode: 'currency'
35   #UI.lineitem.position:90
36   sf.price * sf.seatsocc as Total,
37   sf.seatsmax,
38   sf.seatsocc,
39   sc.url,
40   sf.paymentsum,
41   sf.seatsmax_b,
42   sf.seatsocc_b,
43   sf.seatsmax_f,
44   sf.seatsocc_f
45 }

```

Figure 302: Add Annotation

4. Test your application in the SAP Web IDE.
  - a) Open the SAP Web IDE Fullstack if not already open.
  - b) Start your application as described in the previous task.
  - c) Confirm the application is allowed to store files on your device.
  - d) The application displays with no data, which is correct.
  - e) Choose Go to send the get request.



Figure 303: Go Button

- f) The results are displayed in the table when the request is processed successfully.

Arrive	Connection Number	Flight Date	Arrive	Plane Type	Arrive - Depart. city	Arrive city	Total	
AA	17	May 10, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	156,802.50 USD
AA	17	Jun 13, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	156,487.80 USD
AA	17	Jul 13, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	156,379.56 USD
AA	17	Aug 14, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	156,933.74 USD
AA	17	Sep 15, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	157,796.42 USD
AA	17	Oct 17, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	153,527.22 USD
AA	17	Nov 18, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	157,333.68 USD
AA	17	Dec 20, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	156,487.80 USD
AA	17	Jan 21, 2019	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	76,870.08 USD
AA	17	Feb 22, 2019	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	0.00 USD

Figure 304: Results Table

## Task 6: Add Responsive Behaviour to the List Report.

In the next task you will add responsive behaviour to your application.

1. Add the following annotations and information to the fields of the projection list.

Table 80: Projection List Field Information

Field	Term	Property	Value
planetype	lineItem	importance	#HIGH
Total	lineItem	importance	#MEDIUM
price	lineItem	importance	#LOW

- a) Start *Eclipse* as shown in the first task of the exercise, if not already open.
- b) Open the created *CDS* located in the package *ZTRAIN\_##*, if not already displayed.
- c) Add the noted terms to the fields of the projection list.



```

11  define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
12    association to scarr as sc on sf.carrid = sc.carrid
13    association to sflights as sfs on sfs.carrid = sf.carrid and
14      sfs.connid = sf.connid and sfs.flidate = sf.flidate
15  {
16    @UI.lineItem.position:10
17    sf.flidate,
18    @UI.lineItem.position:20
19    key sf.connid,
20    @UI.lineItem.position:30
21    key sf.flidate,
22    @UI.lineItem.position:40
23    sc.carriename,
24    @UI.lineItem.position:70
25    sfs.cityfrom,
26    @UI.lineItem.position:80
27    sfs.cityto,
28    @UI.lineItem.position:50
29    @UI.lineItem.importance: #HIGH
30    sf.planetype,
31    @UI.lineItem.importance: #LOW
32    @UI.lineItem.position:60
33    sf.price,
34    @Semantics.currencyCode: true
35    sf.currency,
36    @Semantics.amount.currencyCode: 'currency'
37    @UI.lineItem.importance: #MEDIUM
38    @UI.lineItem.position:90
39    sf.price * sf.seatsocc as Total,
40    sf.seatsmax,
41    sf.seatsocc,
42    sc.url,
43    sf.seatsmaxsum,
44    sf.seatsmax_b,
45    sf.seatsocc_b,
46    sf.seatsmax_f,
47    sf.seatsocc_f
48 }

```

Figure 305: Add Terms to Define the View

d) Activate your changes by choosing .

2. Test your application in the SAP Web IDE. Use the features of the Google Chrome Development Tools to change the form factor.
- a) Open the SAP Web IDE as described in the previous task, if not already open.
- b) Start your application as described in the previous task.
- c) When your application starts, choose F12 to access the development tools of Google Chrome. Your browser window looks as shown.

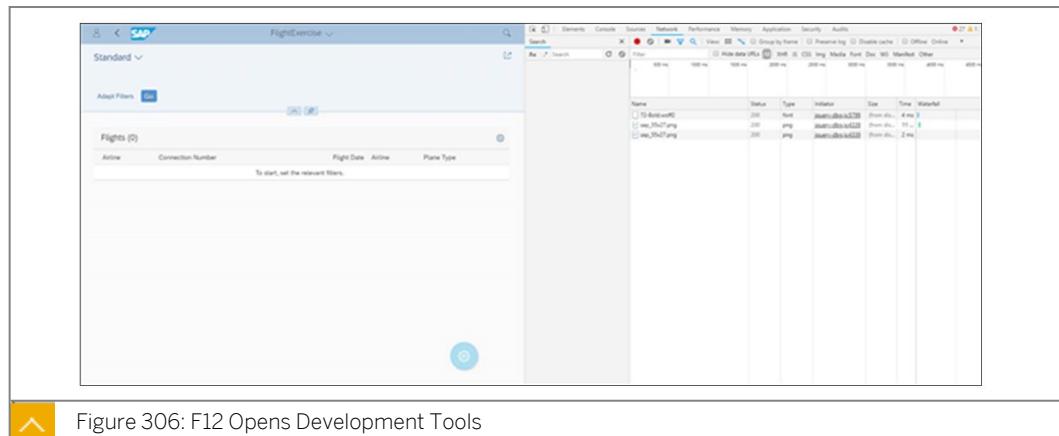


Figure 306: F12 Opens Development Tools

d) Choose the icon  to access the switch form factor feature.

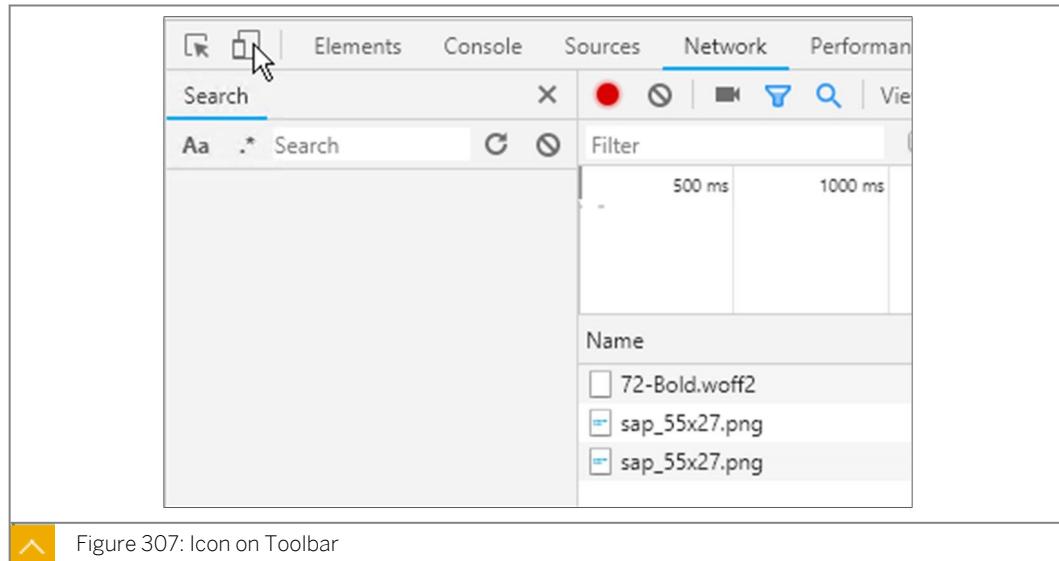


Figure 307: Icon on Toolbar

e) You now see the application switch-form-factor feature.

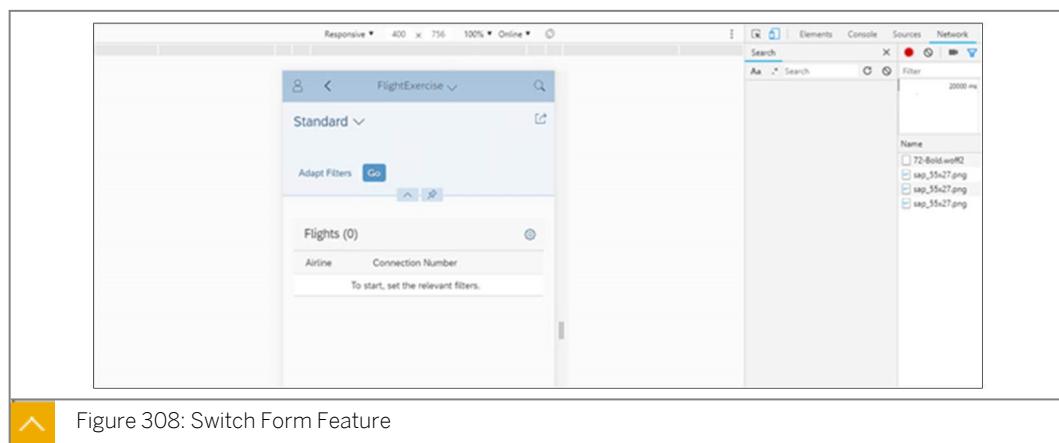


Figure 308: Switch Form Feature

f) Choose Go to trigger a get request.

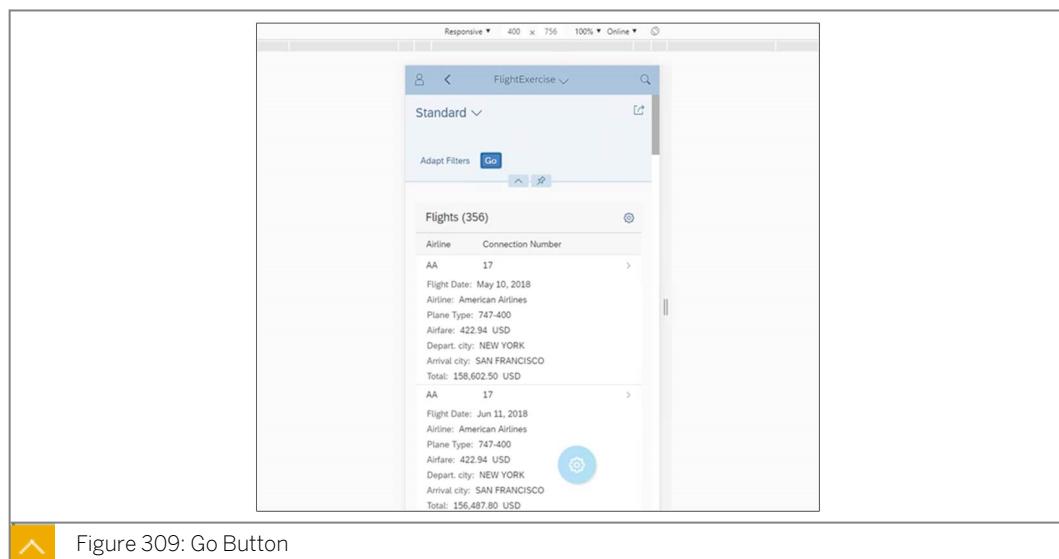


Figure 309: Go Button

g) Now switch the form factor from responsive to, for example, iPhone 5/SE.

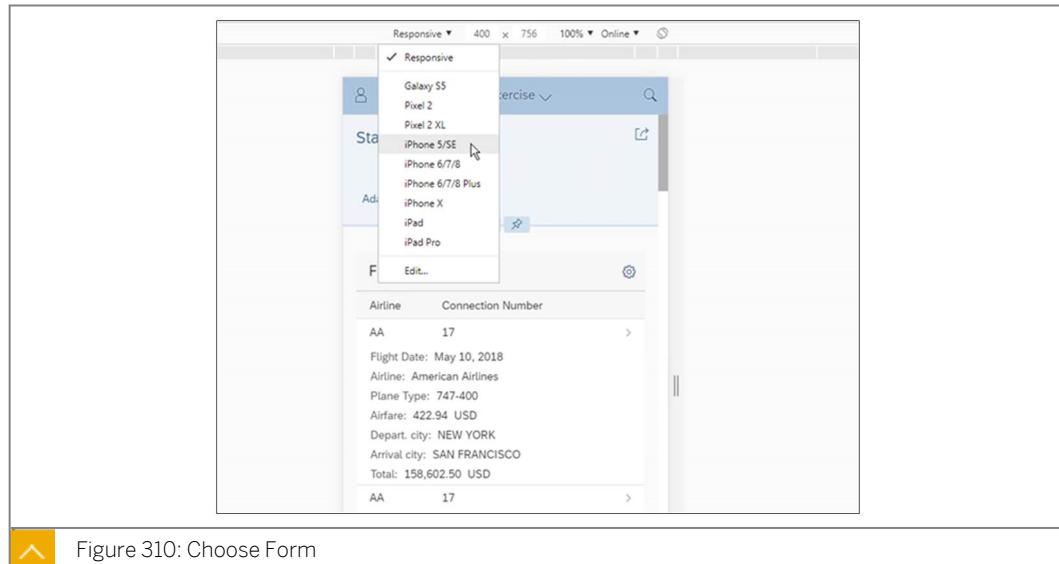


Figure 310: Choose Form

- h) After you have switched form factor, reload your application by choosing *F5*.
- i) Choose the Go button to trigger a new get request.
- j) After the request is successfully completed, you see *planetype* is still displayed as importance *#HIGH*.

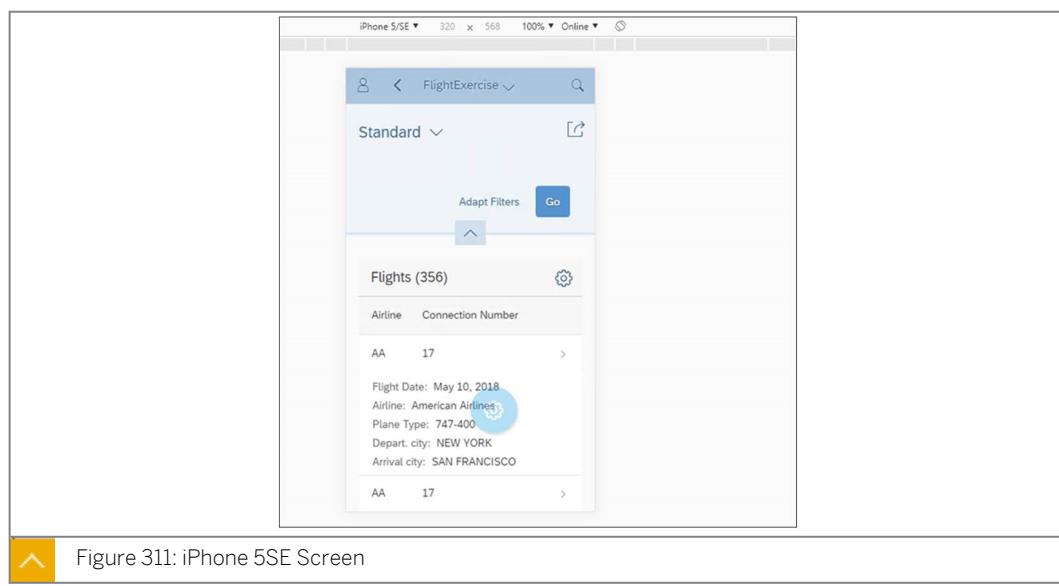


Figure 311: iPhone 5SE Screen

- k) Close the current browser tab.

## Unit 14

### Exercise 16

# Implement Search and Filter



#### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.



#### Note:

Sometimes after the annotations are changed in the **ABAP CDS**, they are not reflected in the **SAP Fiori Elements Application**, or in the **SAP Web IDE**. If this happens, clean the browser cache and restart the application.

The ABAP-CDS view, created in the previous exercise is required.

1. Activate the search capabilities in the ABAP-CDS view, created in the previous exercise. Add the search facility and assign the following information.

Table 81: Search and Assign the Following Information

Attribute	Datatype	Value
searchable	boolean	true

2. Add an annotation that the field *carrname* is searchable. Therefore add **Search** to the *carrname* field of your projection list and assign the following information.

Table 82: Assign the Information to the Carrname Field

Attribute	Datatype	Value
defaultSearchElement	boolean	true

3. Start your application to test whether your application now has search capabilities.

#### Task 1: Fuzzy Search

1. Add the fuzzy search behaviour to the *carrname* field of the *projectlist*. Use the property *fuzzinessThreshold* and assign the float value **0 . 7** to the property.
2. Test your application in the **SAP Web IDE**.

#### Task 2: Add filter options to the Filtering Application

1. Activate the fields *carrname*, *cityto*, and *cityfrom* as filter criteria in the application. Use the annotation *selectionField* for this.

2. To describe basic information of data you want to display, add filter options using the term **UI.selectionField** and the information provided in the table.

Table 83: Basic Data Information

Field	Position
carrname	10
cityto	20
cityfrom	30

3. Test your application in the *SAP Web IDE*.

## Implement Search and Filter



Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.



Note:

Sometimes after the annotations are changed in the **ABAP CDS**, they are not reflected in the *SAP Fiori Elements Application*, or in the *SAP Web IDE*. If this happens, clean the browser cache and restart the application.

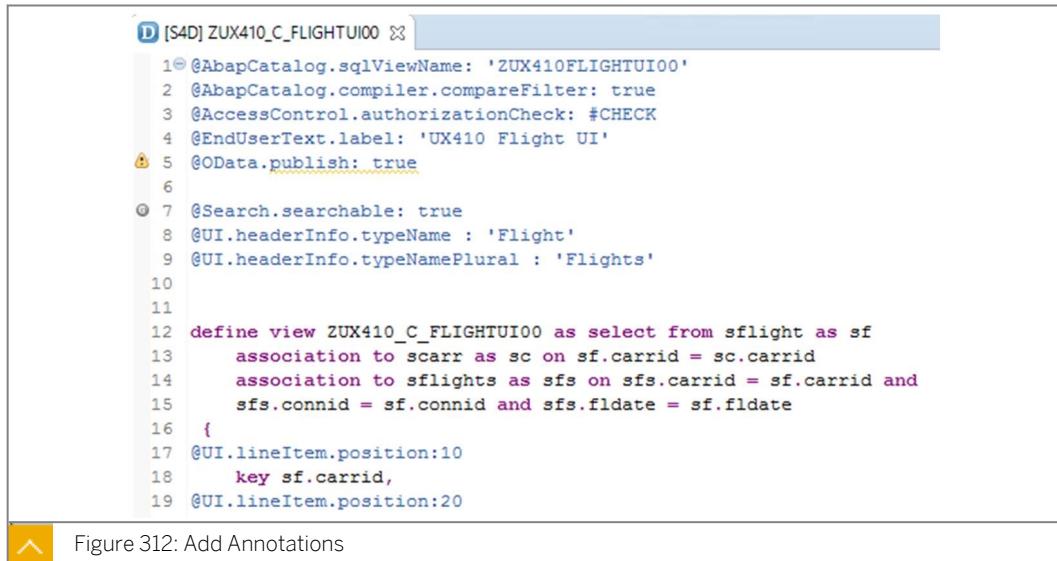
The ABAP-CDS view, created in the previous exercise is required.

1. Activate the search capabilities in the ABAP-CDS view, created in the previous exercise.  
Add the search facility and assign the following information.

Table 81: Search and Assign the Following Information

Attribute	Datatype	Value
searchable	boolean	true

- a) Start eclipse if not already open.
- b) Open the created CDS (Core Data Services) located in the package ZTRAIN\_##.
- c) Add the annotations to the ABAP-CDS.



```

D [S4D] ZUX410_C_FLIGHTUI00
1 @AbapCatalog.sqlViewName: 'ZUX410FLIGHTUI00'
2 @AbapCatalog.compiler.compareFilter: true
3 @AccessControl.authorizationCheck: #CHECK
4 @EndUserText.label: 'UX410 Flight UI'
5 @OData.publish: true
6
7 @Search.searchable: true
8 @UI.headerInfo.typeName : 'Flight'
9 @UI.headerInfo.typeNamePlural : 'Flights'
10
11
12 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
13   association to scarr as sc on sf.carrid = sc.carrid
14   association to sflights as sfs on sfs.carrid = sf.carrid and
15   sfs.connid = sf.connid and sfs.flidate = sf.flidate
16 {
17   @UI.lineItem.position:10
18   key sf.carrid,
19   @UI.lineItem.position:20

```

Figure 312: Add Annotations

2. Add an annotation that the field `carrname` is searchable. Therefore add `Search` to the `carrname` field of your projection list and assign the following information.

Table 82: Assign the Information to the Carrname Field

Attribute	Datatype	Value
defaultSearchElement	boolean	true

- a) Add the annotation to the `carrname` field of your projection list.



```

D [S4D] ZUX410_C_FLIGHTUI00
1 @AbapCatalog.sqlViewName: 'ZUX410FLIGHTUI00'
2 @AbapCatalog.compiler.compareFilter: true
3 @AccessControl.authorizationCheck: #CHECK
4 @EndUserText.label: 'UX410 Flight UI'
5 @OData.publish: true
6
7 @Search.searchable: true
8 @UI.headerInfo.typeName : 'Flight'
9 @UI.headerInfo.typeNamePlural : 'Flights'
10
11
12 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
13   association to scarr as sc on sf.carrid = sc.carrid
14   association to sflights as sfs on sfs.carrid = sf.carrid and
15   sfs.connid = sf.connid and sfs.flidate = sf.flidate
16 {
17   @UI.lineItem.position:10
18   key sf.carrid,
19   @UI.lineItem.position:20
20   key sf.connid,
21   @UI.lineItem.position:30
22   key sf.flidate,
23   @Search.defaultSearchElement: true
24   @UI.lineItem.position:40
25   sc.carrname
26   @UI.lineItem.position:70
27   sfs.cityfrom,
28   @UI.lineItem.position:80
29   sfs.cityto,
30   @UI.lineItem.position:50
31   @UI.lineItem.importance: #HIGH
32   sf.planetype,

```

Figure 313: Add Annotation

- b) Activate your changes.

3. Start your application to test whether your application now has search capabilities.
- Start the SAP Web IDE as described in the previous exercise, if not already started.
  - To test the application, select the project root folder and choose from the context menu `Run → Run an SAP Fiori Launchpad Sandbox`.

- c) When the application starts, you see the search field and the Go button to trigger a get request.

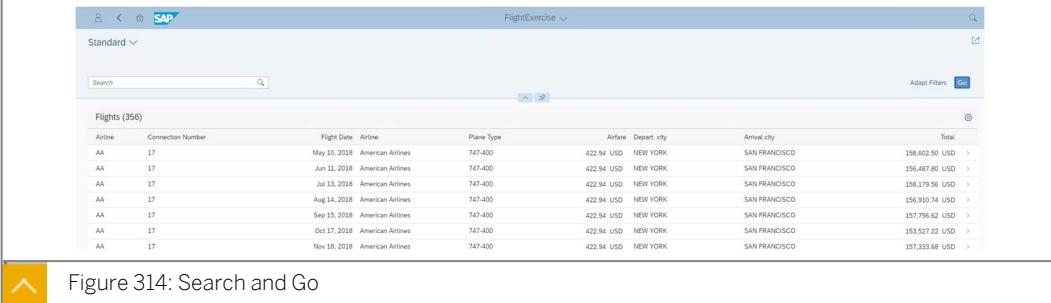


Figure 314: Search and Go

Flights (356)								
Airline	Connection Number	Flight Date	Airline	Plane Type	Airfare	Depart. city	Arrival city	Total
AA	17	May 10, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	158,602.50 USD
AA	17	Jun 11, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	156,487.80 USD
AA	17	Jul 12, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	159,919.74 USD
AA	17	Aug 14, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	157,756.62 USD
AA	17	Sep 15, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	153,527.22 USD
AA	17	Oct 17, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	157,333.68 USD
AA	17	Nov 18, 2018	American Airlines	747-400	422.94 USD	NEW YORK	SAN FRANCISCO	



Note:  
If the search field is not shown, clear the browser cache (hard reload).

- d) Enter the name of a carrier in the search field and choose *Enter*.

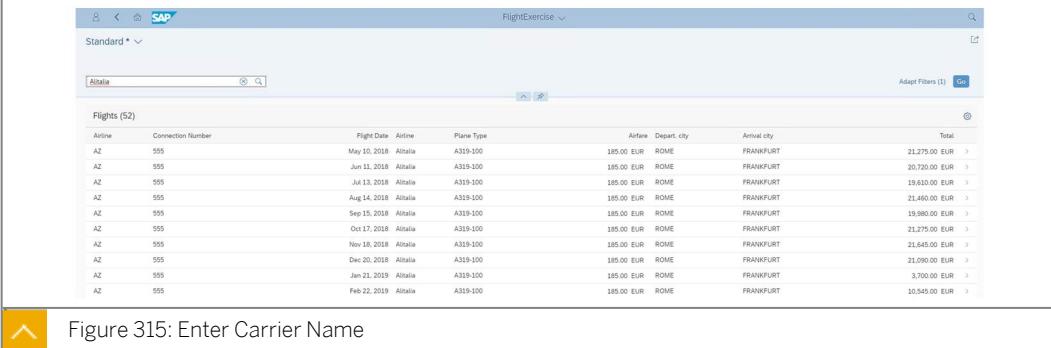


Figure 315: Enter Carrier Name

Flights (52)								
Airline	Connection Number	Flight Date	Airline	Plane Type	Airfare	Depart. city	Arrival city	Total
AZ	555	May 10, 2018	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	21,275.00 EUR
AZ	555	Jun 11, 2018	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	20,720.00 EUR
AZ	555	Jul 12, 2018	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	19,650.00 EUR
AZ	555	Aug 14, 2018	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	21,460.00 EUR
AZ	555	Sep 15, 2018	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	19,980.00 EUR
AZ	555	Oct 17, 2018	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	21,275.00 EUR
AZ	555	Nov 18, 2018	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	21,645.00 EUR
AZ	555	Dec 20, 2018	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	21,090.00 EUR
AZ	555	Jan 21, 2019	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	3,700.00 EUR
AZ	555	Feb 22, 2019	Alitalia	A319-100	185.00 EUR	ROME	FRANKFURT	10,545.00 EUR



- e) Add a little typo into the name of a carrier. You will get no search result.



Figure 316: Test with a Typing Error

Flights (0)								
Airline	Connection Number	Flight Date	Airline	Plane Type	Airfare	Depart. city	Arrival city	Total
No data found. Try adjusting the filter settings.								

## Task 1: Fuzzy Search

- Add the fuzzy search behaviour to the *carrname* field of the *projectlist*. Use the property *fuzzinessThreshold* and assign the float value **0.7** to the property.
  - Add the listed annotation to the *carrname* field of your project list.



```

1 @AbapCatalog.sqlViewName: 'ZUX410FLIGHTUI00'
2 @AbapCatalog.compiler.compareFilter: true
3 @AccessControl.authorizationCheck: #CHECK
4 @EndUserText.label: 'UX410 Flight UI'
5 @OData.publish: true
6
7 @Search.searchable: true
8 @UI.headerInfo.typeName : 'Flight'
9 @UI.headerInfo.typeNamePlural : 'Flights'
10
11
12 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
13   association to scarr as sc on sf.carrid = sc.carrid
14   association to sflights as sfs on sfs.carrid = sf.carrid and
15   sfs.connid = sf.connid and sfs.fldate = sf.fldate
16 {
17   @UI.lineItem.position:10
18   key sf.carrid,
19   @UI.lineItem.position:20
20   key sf.connid,
21   @UI.lineItem.position:30
22   key sf.fldate,
23 @Search.defaultSearchElement: true
24 @Search.fuzzinessThreshold: 0.7
25 @UI.lineItem.position:40

```

Figure 317: Add Annotation to the Carrname Field

- b) Activate your changes.
2. Test your application in the SAP Web IDE.
    - a) Open the SAP Web IDE, if not already open.
    - b) Start your application as previously described.
    - c) Choose the Go button to trigger a get request.
    - d) Enter the value **Akitalia** in the search field and choose *Enter*.
    - e) Close the current browser tab.

### Task 2: Add filter options to the Filtering Application

1. Activate the fields *carrname*, *cityto*, and *cityfrom* as filter criteria in the application. Use the annotation *selectionField* for this.
2. To describe basic information of data you want to display, add filter options using the term **UI.selectionField** and the information provided in the table.

Table 83: Basic Data Information

Field	Position
carrname	10
cityto	20
cityfrom	30

- a) Add the *UI.selectionField* annotation to the field of the projectlist and assign the integer value to the property positions.

```
 1  "@sapCatalog.sqlViewName: 'ZUX410FLIGHTUI00'  
 2  @sapCatalog.compiler.compareFilter: true  
 3  @accessControl.authorizationCheck: #CHECK  
 4  @endUserText.label: 'UX410 Flight UI'  
 5  @Data.publish: true  
 6  
 7  @search.searchable: true  
 8  @UI.headerInfo.typeName : 'Flight'  
 9  @UI.headerInfo.typeNamePlural : 'Flights'  
10  
11  
12  define view ZUX410_C_FLIGHTUI00 as select from sflight as sf  
13      association to scarr as sc on sf.carrid = sc.carrid  
14      association to sflight as sfs on sfs.carrid = sf.carrid and  
15      sfs.connid = sf.connid and sfs.fldate = sf.fldate  
16  {  
17      @UI.lineItem.position:10  
18      key sf.carrid,  
19      @UI.lineItem.position:20  
20      key sf.connid,  
21      @UI.lineItem.position:30  
22      key sf.fldate,  
23      @search.defaultSearchable: true  
24      @search.defaultSearchThreshold: 0.7  
25      @UI.selectionField.position:10  
26      @UI.lineItem.position:40  
27      sc.carname,  
28      @UI.selectionField.position:20  
29      @UI.lineItem.position:70  
30      sfs.cityfrom  
31      @UI.selectionField.position:30  
32      @UI.lineItem.position:80  
33      sfs.cityto,  
34      @UI.lineItem.position:50  
35      @UI.lineItem.importance: #HIGH  
  sc  
  as  
  sfs
```

Figure 318: Add UI.selectionField Annotation

b) Activate your changes.

### 3. Test your application in the SAP Web IDE.

a) Open the SAP Web IDE, if not already open.

**b)** Start your application as previously described.

c) Choose the Go button to trigger a get request.

d) Click the value help symbol at the *Depart.city* selection field.

- e) The *conditions* dialog for departure city will be opened.

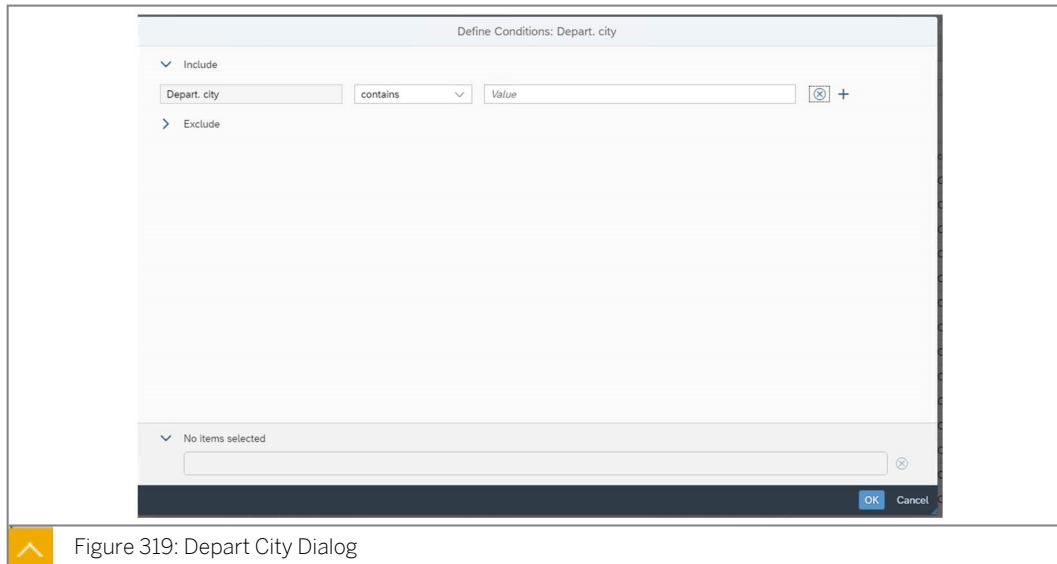


Figure 319: Depart City Dialog

f) Enter the value **SAN** into the *input* field and choose *OK*.

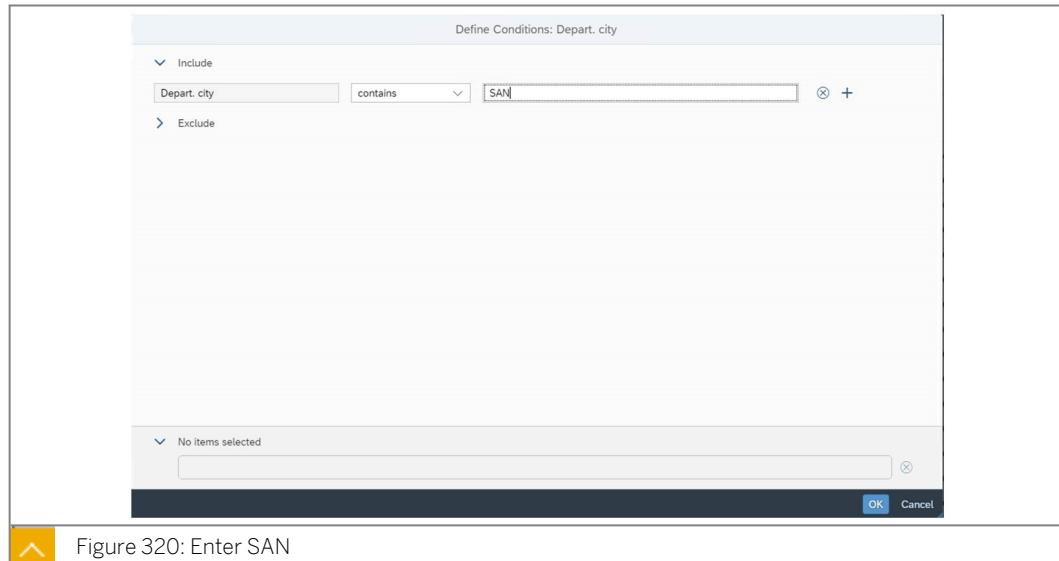


Figure 320: Enter SAN

g) Choose the Go button to apply the filter.

FlightExercise									
Standard		Airline:		Depart. city:		Arrival city:		Adapt Filters (1) 	
Flights (54)									
Airline	Connection Number	Flight Date	Airline	Plane Type	Airfare	Depart. city	Arrival city	Total	
AA	64	May 12, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,917.86 USD	
AA	64	Jun 13, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,494.42 USD	
AA	64	Jul 15, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,917.86 USD	
AA	64	Aug 16, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	133,013.10 USD	
AA	64	Sep 17, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	132,002.20 USD	
AA	64	Oct 19, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	131,534.30 USD	
AA	64	Nov 20, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	131,534.34 USD	
AA	64	Dec 22, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	133,002.00 USD	
AA	64	Jan 23, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	78,668.40 USD	
AA	64	Feb 24, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	54,326.30 USD	
AA	64	Mar 26, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	39,796.20 USD	
AA	64	Apr 29, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	17,763.40 USD	
AA	64	May 31, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,819.10 USD	
DL	1984	May 8, 2018	Delta Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,819.10 USD	
DL	1985	Jun 9, 2018	Delta Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,819.10 USD	

Figure 321: Apply the Filter

h) After the filter is applied, the list only shows flights with a departure city of SAN FRANCISCO.

FlightExercise									
Standard		Airline:		Depart. city:		Arrival city:		Adapt Filters (1) 	
Flights (54)									
Airline	Connection Number	Flight Date	Airline	Plane Type	Airfare	Depart. city	Arrival city	Total	
AA	64	May 12, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,917.86 USD	
AA	64	Jun 13, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,494.42 USD	
AA	64	Jul 15, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,917.86 USD	
AA	64	Aug 16, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	133,013.10 USD	
AA	64	Sep 17, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	132,002.20 USD	
AA	64	Oct 19, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	131,534.30 USD	
AA	64	Nov 20, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	131,534.34 USD	
AA	64	Dec 22, 2018	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	133,002.00 USD	
AA	64	Jan 23, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	78,668.40 USD	
AA	64	Feb 24, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	54,326.30 USD	
AA	64	Mar 26, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	39,796.20 USD	
AA	64	Apr 29, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	17,763.40 USD	
AA	64	May 31, 2019	American Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,819.10 USD	
DL	1984	May 8, 2018	Delta Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,819.10 USD	
DL	1985	Jun 9, 2018	Delta Airlines	A340-600	422.94 USD	SAN FRANCISCO	NEW YORK	134,819.10 USD	

Figure 322: Displays Flights for San Francisco

i) Try testing other selections in the field.

## Unit 14 Exercise 17

# Implement Object Page SAP Fiori Elements



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and names such as *buttons* and *fields* may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.



### Note:

In some cases the changes in the *ABAP CDS* are not directly reflected in the *SAP Fiori Elements Application*. If this happens, clean the browser cache and restart the application.

### Implement Header Data

1. Set the title of every flight to the value of field *carrid*, and the description to the value of field *carrname*.
2. Test your application in the *SAP Web IDE Full-Stack*.

#### Task 1: Adding a HeaderFacet to the object page

1. Assign the following fields of the projection list to *fieldGroup* with qualifier *flightinfo*, and assign the position as listed in the table.

Table 84: Assign Positions as Listed for *fieldGroup* with Qualifier *flightinfo*

Field	Position
cityfrom	10
cityto	20
url	30

2. Assign the following fields of the projection list to *fieldGroup* with qualifier *seatinfo*, and assign the positions as listed in the table.

Table 85: Assign *Flightinfo*

Field	Position
seatsmax	10
seatsocc	20

3. Add two facets of purpose #HEADER to your ABAP CDS. The first facet should reference the qualifier *flightinfo* as a target and the second should reference the qualifier *seatinfo* as a target

### Task 2: Adding fields to the standard facet

1. Assign the annotation @UI.identification to the listed fields in your ABAP-CDS and assign the position as an integer. Test the result in the SAP Web IDE.

Table 86: Assign to the Listed Fields in Your ABAP-CDS

Field	Position
carrname	10
flightdate	20
cityfrom	30
cityto	40
planetype	50

### Task 3: Add Your Own Facets

1. Assign the following fields of the projection list to a *fieldGroup* with a qualifier *seatdata*, and assign the position as listed in the table.

Table 87: Assign Fields

Field	Position
seatsmax_b	10
seatsmax_f	20

2. Assign the fields of the projection list to a *fieldGroup* with qualifier *seatdata2*, and assign the positions as listed in the table. Then activate your changes.

Table 88: Assign Fields

Field	Position
seatsocc_b	10
seatsocc_f	20

3. Add two facets of purpose #STANDARD and type #FIELDGROUP\_REFERENCE to your @UI.facet-implementation. The first should reference the targetQualifier *seatdata* and the second the targetQualifier *seatdata2*.
4. In the local annotation file of your SAP Web IDE project, add the property label with the following values to the two new UI.ReferenceFacet-entry.

Property	Value
ReferenceFacet for qualifier seatdata	Seatinfo1

Property	Value
ReferenceFacet for qualifier seatdata	Seatinfo2



**Note:**

You can also add the text to the resource files but, for the sake of simplicity, we are not working with i18n resources.

## Implement Object Page SAP Fiori Elements



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and names such as *buttons* and *fields* may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.



### Note:

In some cases the changes in the ABAP CDS are not directly reflected in the SAP Fiori Elements Application. If this happens, clean the browser cache and restart the application.

### Implement Header Data

1. Set the title of every flight to the value of field *carrid*, and the description to the value of field *carrname*.
  - a) Open Eclipse as described in the previous exercise, if not already open.
  - b) Open the ABAP CDS *ZUX410\_C\_FLIGHTUI##* from your favourite package *ZTRAIN\_N##*, if not already open.
  - c) Add title and description annotation to the ABAP CDS and assign the fields *carrid* to the title and *carrname* to the description.

```
① [S4D] ZUX410_C_FLIGHTUI00
  1 @AbapCatalog.sqlViewName: 'ZUX410FLIGHTUI00'
  2 @AbapCatalog.compiler.compareFilter: true
  3 @AccessControl.authorizationCheck: #CHECK
  4 @EndUserText.label: 'UX410 Flight UI'
⑤ 5 @OData.publish: true
  6
⑥ 7 @Search.searchable: true
  8 @UI.headerInfo.typeName : 'Flight'
  9 @UI.headerInfo.typeNamePlural : 'Flights'
10 @UI.headerInfo.title.value: 'carrid'
11 @UI.headerInfo.description.value: 'carrname'
  12
  13
14 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
15   association to scarr as sc on sf.carrid = sc.carrid
16   association to sflights as sfs on sfs.carrid = sf.carrid and
17   sfs.connid = sf.connid and sfs.flidate = sf.flidate
  18 {
```



Figure 323: Title and Description Annotation

- d) Activate your changes

2. Test your application in the SAP Web IDE Full-Stack.

- Open the SAP Web IDE Full-Stack as described in previous exercises, if not already open.
- Start the application created in the previous exercise.
- Choose Go on the filter bar to trigger a get-request.
- Select a flight from the list.

Figure 324: Select a Flight

- You now see the `carrid` and `carrname` in the object page.

Figure 325: Information Displayed

### Task 1: Adding a HeaderFacet to the object page

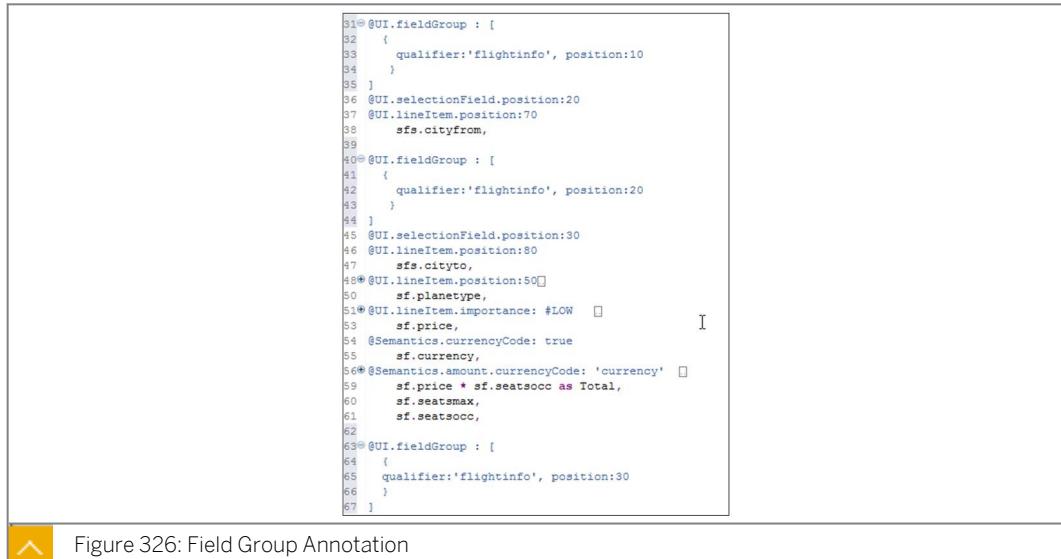
- Assign the following fields of the projection list to `fieldGroup` with qualifier `flightinfo`, and assign the position as listed in the table.

Table 84: Assign Positions as Listed for `fieldGroup` with Qualifier `flightinfo`

Field	Position
<code>cityfrom</code>	10
<code>cityto</code>	20

Field	Position
url	30

- Open Eclipse as described in previous exercises, if not already open.
- Open the ABAP CDS `ZUX410_C_FLIGHTUI##`, if not already open.
- Add **fieldGroup** annotation with the qualifier **flightinfo** to the listed fields and add the position.



The screenshot shows a code editor with ABAP CDS code. A specific section of the code is highlighted with a yellow background, representing the field group annotation. The code within the annotation is as follows:

```

51@UI.fieldGroup : [
52  {
53    qualifier:'flightinfo', position:10
54  }
55 ]
56 @UI.selectionField.position:20
57 @UI.lineItem.position:70
58   sfs.cityfrom,
59
60@UI.fieldGroup : [
61  {
62    qualifier:'flightinfo', position:20
63  }
64 ]
65 @UI.selectionField.position:30
66 @UI.lineItem.position:80
67   sfs.cityto,
68@UI.lineItem.position:50
69   sf.planetype,
70@UI.lineItem.importance: #LOW
71   sf.price,
72   @Semantics.currencyCode: true
73   sf.currency,
74@Semantics.amount.currencyCode: 'currency'
75   sf.price * sf.seatsocc as Total,
76   sf.seatsmax,
77   sf.seatsocc,
78
79@UI.fieldGroup : [
80  {
81    qualifier:'flightinfo', position:30
82  }
83 ]

```

Figure 326: Field Group Annotation

- Assign the following fields of the projection list to *fieldGroup* with qualifier *seatinfo*, and assign the positions as listed in the table.

Table 85: Assign Flightinfo

Field	Position
seatsmax	10
seatsocc	20

- Add the *fieldGroup* annotation with the qualifier *seatinfo* to the fields from the table, and assign the listed positions.

```

48@UI.lineItem.position:50
50    sf.planetype,
51@UI.lineItem.importance: #LOW
53    sf.price,
54 @Semantics.currencyCode: true
55    sf.currency,
56@Semantics.amount.currencyCode: 'currency'
59    sf.price * sf.seatsocc as Total,
60
61@UI.fieldGroup :
62  {
63    qualifier:'seatinfo', position:10
64  }
65 ]
66    sf.seatsmax,
67@UI.fieldGroup :
68  {
69    qualifier:'seatinfo', position:20
70  }
71 ]
72    sf.seatsocc,

```

Figure 327: Add Annotation

b) Activate your changes .

3. Add two facets of purpose #HEADER to your ABAP CDS. The first facet should reference the qualifier *flightinfo* as a target and the second should reference the qualifier *seatinfo* as a target

a) Open the ABAP CDS ZUX410\_C\_FLIGHTUI#, if not already open.

b) Add the following facet-implementation at the beginning of the projection list implementation of your ABAP CDS.

```

@UI.facet: [
  { purpose:#HEADER, type: #FIELDGROUP_REFERENCE, targetQualifier: 'flightinfo', position:10 },
  { purpose:#HEADER, type: #FIELDGROUP_REFERENCE, targetQualifier: 'seatinfo', position:20 },
  { purpose:#STANDARD, type: #FIELDGROUP_REFERENCE, targetQualifier: 'seatacc', position:10 },
  { purpose:#STANDARD, type: #FIELDGROUP_REFERENCE, targetQualifier: 'seatacc2', position:20 },
  { purpose:#STANDARD, id: 'seat1', label:'Seatinfo 1', type:#COLLECTION, position:30 },
  { purpose:#STANDARD, id: 'seat2', label:'Seatinfo 2', type:#COLLECTION, position:40 }
]

```

Figure 328: First Facet

c) Add the second facet to your implementation.

```

23 {
24
25@UI.facet: [
26  { purpose: #HEADER, type: #FIELDGROUP_REFERENCE, targetQualifier: 'flightinfo', position:10 },
27  { purpose: #HEADER, type: #FIELDGROUP_REFERENCE, targetQualifier: 'seatinfo', position:20 }
28
29    @UI.lineItem.position:10
30    key sf.carrid,
31    @UI.lineItem.position:20
32    key sf.connid,
33

```

Figure 329: Second Facet

d) Save and activate your changes.

e) Start the application as described in previous exercises.

f) Choose Go to trigger a get request and select a flight from the list.

g) You will now see the new HeaderFacet.

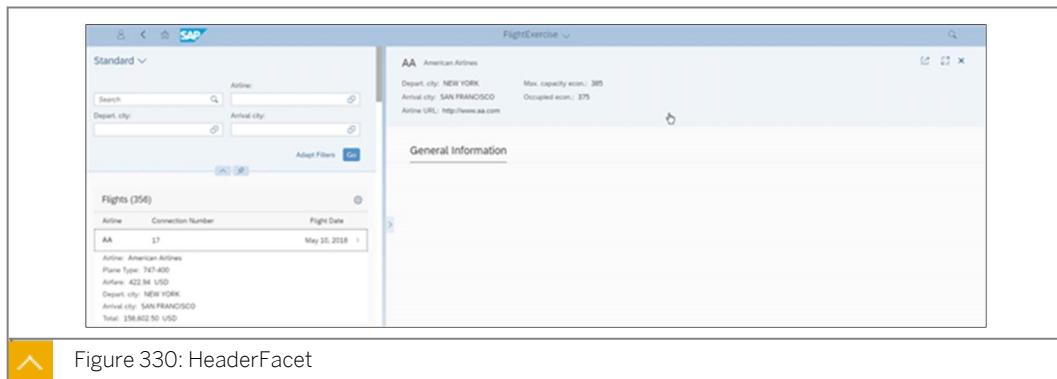


Figure 330: HeaderFacet

## Task 2: Adding fields to the standard facet

1. Assign the annotation `@UI.identification` to the listed fields in your ABAP-CDS and assign the position as an integer. Test the result in the SAP Web IDE.

Table 86: Assign to the Listed Fields in Your ABAP-CDS

Field	Position
carrname	10
flightdate	20
cityfrom	30
cityto	40
planetype	50

- a) Open the ABAP CDS `ZUX410_C_FLIGHTUI#`, if not already open.
- b) Add `@UI.identification` annotation to the field shown in the table and add the `position` property with the listed integer value.

```

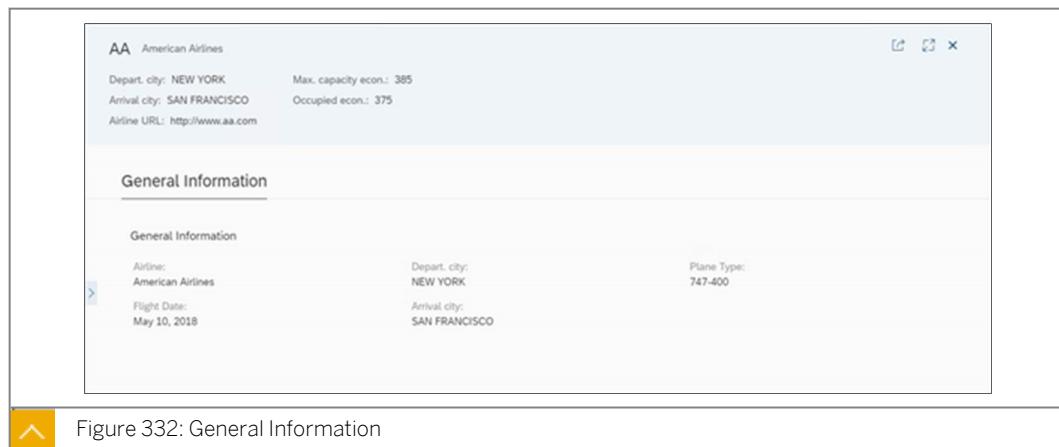
19 @UI.lineItem.position:10
20   key sf.carrid,
21   @UI.lineItem.position:20
22   key sf.connid,
23
24 @UI.identification.position:20
25 @UI.lineItem.position:30
26   key sf.fldate,
27
28 @UI.identification.position:30
29 @Search.defaultSearchElement: true
30 @Search.fuzzinessThreshold: 0.7
31 @UI.selectionField.position:10
32 @UI.lineItem.position:40
33   sc.carrname,
34
35 @UI.identification.position:30
36 @UI.fieldGroup : [
37   {
38     qualifier:'flightinfo', position:10
39   }
40 ]
41 @UI.selectionField.position:20
42 @UI.lineItem.position:70
43   sfsc.cityfrom,
44
45 @UI.identification.position:40
46 @UI.fieldGroup : [
47   {
48     qualifier:'flightinfo', position:20
49   }
50 ]
51 @UI.selectionField.position:30
52 @UI.lineItem.position:80
53   sfsc.cityto,
54
55 @UI.identification.position:50
56 @UI.lineItem.position:50
57 @UI.lineItem.importance: #HIGH
58   sf.planetype,
59 @UI.lineItem.importance: #LOW

```

Figure 331: Add Annotation

- c) Activate your changes 

- d) Start the SAP Web IDE as described in previous exercises, if not already started.
- e) Close the annotations.xml file if open and reopen it again using the *Annotation Modeler*.
- f) Expand the UI.Facets node in the *Local Annotations* and ensure the @UI.identification annotation is shown.
- g) Start the application as described in previous exercises.
- h) Choose the Go button and select a flight from the list.
- i) The General Information facet of the object page now shows the annotated fields.



### Task 3: Add Your Own Facets

1. Assign the following fields of the projection list to a *fieldGroup* with a qualifier *seatdata*, and assign the position as listed in the table.

Table 87: Assign Fields

Field	Position
seatsmax_b	10
seatsmax_f	20

- a) Start Eclipse if not already started.
- b) Open ABAP CDS ZUX410\_C\_FLIGHTUI##.
- c) Add the *fieldGroup* annotation to the fields of your projection list and assign the positions.

```

89@UI.fieldGroup : [
90  {qualifier:'seatdata', position:10}
91 ]
92  sf.seatsmax_b,
93  sf.seatsocc_b,
94@UI.fieldGroup : [
95  {qualifier:'seatdata', position:20}
96 ]
97  sf.seatsmax_f,
98  sf.seatsocc_f
99 }

```

Figure 333: Add Annotation

2. Assign the fields of the projection list to a *fieldGroup* with qualifier *seatdata2*, and assign the positions as listed in the table. Then activate your changes.

Table 88: Assign Fields

Field	Position
seatsocc_b	10
seatsocc_f	20

- a) Add the *fieldGroup* annotation to the fields of your projection list and assign the positions.

```

89@UI.fieldGroup : [
90  {qualifier:'seatdata', position:10}
91 ]
92  sf.seatsmax_b,
93@UI.fieldGroup : [
94  {qualifier:'seatdata2', position:10}
95 ]
96  sf.seatsocc_b,
97@UI.fieldGroup : [
98  {qualifier:'seatdata', position:20}
99 ]
100 sf.seatsmax_f,
101@UI.fieldGroup : [
102  {qualifier:'seatdata2', position:10}
103 ]
104 sf.seatsocc_f
105 }

```

Figure 334: Assign Position

- b) Activate your changes .

3. Add two facets of purpose #STANDARD and type #FIELDGROUP\_REFERENCE to your @UI.facet-implementation. The first should reference the targetQualifier *seatdata* and the second the targetQualifier *seatdata2*.

- a) Extend the @UI.facet-implementation as shown in the next figure.

```

@UI.facet: [
  { purpose: '#HEADER', type: '#FIELDGROUP_REFERENCE', targetQualifier: 'flightinfo', position:10 },
  { purpose: '#HEADER', type: '#FIELDGROUP_REFERENCE', targetQualifier: 'seatinfo', position:20 },
  { purpose: '#STANDARD', type: '#FIELDGROUP_REFERENCE', targetQualifier: 'seatdata', position:10 },
  { purpose: '#STANDARD', type: '#FIELDGROUP_REFERENCE', targetQualifier: 'seatdata2', position:20 },
  {purpose:#STANDARD, id:'seat1', label:'Info1', type:#COLLECTION,position:30, label: 'Seatsinfo 1'} ,
  {purpose:#STANDARD, id:'seat2', label:'Info2', type:#COLLECTION,position:40, label:'Seatsinfo 2'}
]

```

Figure 335: Extend the @UI.facet-implementation

**b)** Save and activate your changes.

4. In the local annotation file of your SAP Web IDE project, add the property label with the following values to the two new UI.ReferenceFacet-entry.

Property	Value
ReferenceFacet for qualifier seatdata	Seatinfo1
ReferenceFacet for qualifier seatdata	Seatinfo2



Note:

You can also add the text to the resource files but, for the sake of simplicity, we are not working with i18n resources.

- Open the SAP Web IDE if not already opened.
- Open the *annotation.xml* file located in the annotations folder of your project.
- Expand the UI.Facets entry of your local annotations.
- Click the +-button on the level of the UI.ReferenceFacet that points to the target *@UI.FieldGroup#seatdata*.
- Choose *Label* in the upcoming dialog and choose *OK*.
- Insert the value from the table into the value field of the label.
- Repeat the steps d to f for the ReferenceFacet with target *@UI.FieldGroup#seatdata2*.
- Save your changes.
- Start the application as described in previous exercises.
- Choose the Go button of the *FacetFilter* and choose a flight.
- You will now see the newly added facets.

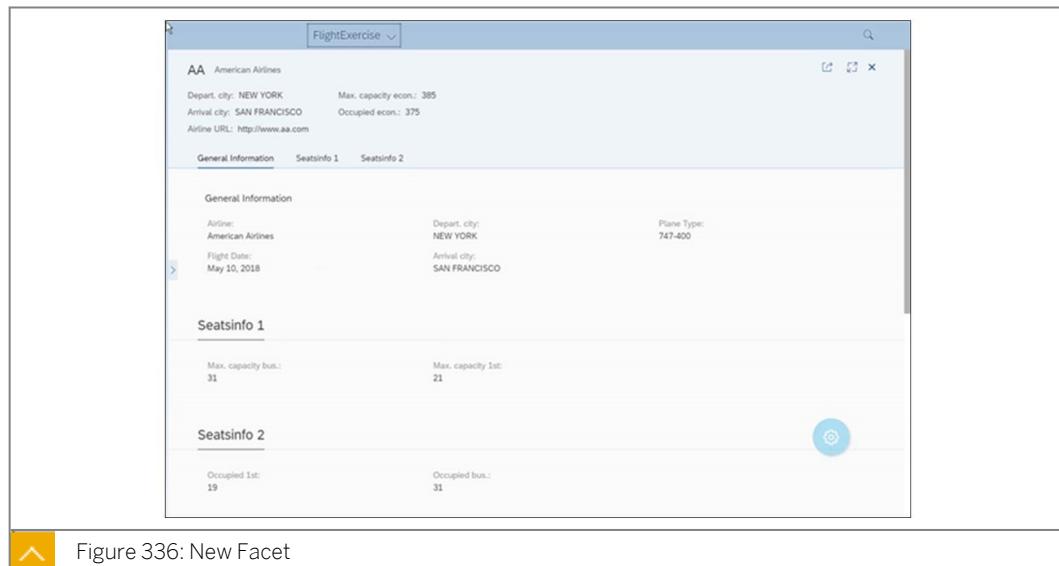


Figure 336: New Facet

## Unit 14 Exercise 18

# Display Dependent Entities as SAP Fiori Elements



### Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.



### Note:

Sometimes after the annotations are changed in the **ABAP CDS**, they are not reflected in the *SAP Fiori Elements Application*, or in the *SAP Web IDE*. When this happens, clean the browser cache and restart the application.

Implement a CDS view for the selection of bookings.

1. Implement a CDS view for the selection of bookings.

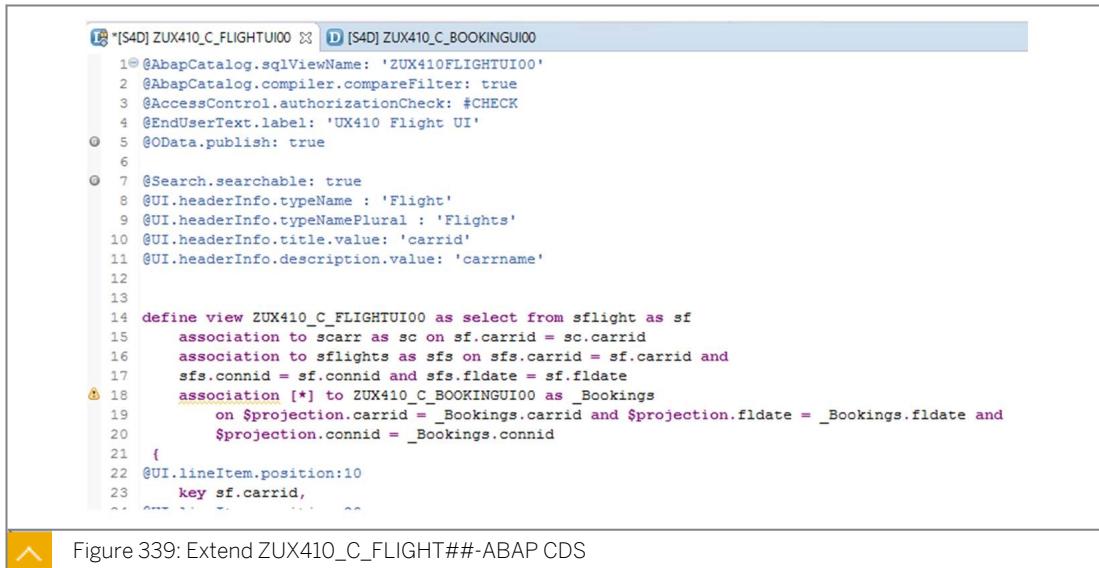
Table 89: ABAP CDS Configuration

Property	Value
Name	ZUX410_C_BOOKINGUI00## (replace ## with your group number)
Description	Flight bookings
Package	ZTRAIN_## (replace ## with your group number)
Transport request	Use the request assigned to your group by your instructor
Template for creating New Data Definition	No template

2. Copy the content of the file `S:\Courses\UX410_20\Templates\ZUX410_C_BookingUI##.txt` to the newly created ABAP CDS view and replace the ## in the copied content with your group number.

**Task 1: Add an association to `ZUX410_C_FLIGHTUI##`.**

1. Extend the `ZUX410_C_FLIGHT##-ABAP CDS` as shown in the figure and expose the `_Bookings` alias. Remember to use your own group number.



```

 1 * [S4D] ZUX410_C_FLIGHTUI00 [S4D] ZUX410_C_BOOKINGUI00
 2 @AbapCatalog.sqlViewName: 'ZUX410FLIGHTUI00'
 3 @AbapCatalog.compiler.compareFilter: true
 4 @AccessControl.authorizationCheck: #CHECK
 5 @EndUserText.label: 'UX410 Flight UI'
 6
 7 @Search.searchable: true
 8 @UI.headerInfo.typeName : 'Flight'
 9 @UI.headerInfo.typeNamePlural : 'Flights'
10 @UI.headerInfo.title.value: 'carrid'
11 @UI.headerInfo.description.value: 'carrname'
12
13
14 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
15   association to scarr as sc on sf.carrid = sc.carrid
16   association to sflights as sfs on sfs.carrid = sf.carrid and
17   sfs.connid = sf.connid and sfs.fldate = sf.fldate
18   association [*] to ZUX410_C_BOOKINGUI00 as _Bookings
19     on $projection.carrid = _Bookings.carrid and $projection.fldate = _Bookings.fldate and
20     $projection.connid = _Bookings.connid
21   {
22     @UI.lineItem.position:10
23     key sf.carrid,
24     ...
25   }

```

Figure 339: Extend ZUX410\_C\_FLIGHT##-ABAP CDS

### Task 2: Add the bookings to the object page as a facet.

1. Add the `to_bookings` navigation property generated by SSDL to your local annotations. Add `to_bookings` as a `ReferenceFacet` to the existing `UI.Facets` annotation. Add a property label to the newly created `ReferenceFacet` and assign the text **Bookings** to the `Label` property.
2. Test your application.

## Display Dependent Entities as SAP Fiori Elements



Note:

SAP Cloud Platform and its tools are cloud services. Due to the nature of cloud software, step procedures and naming of fields and buttons may differ from the exercise solution. The following figures show the application of student ## and group 00. Use your own student and group number assigned by your instructor for the exercises.



Note:

Sometimes after the annotations are changed in the **ABAP CDS**, they are not reflected in the *SAP Fiori Elements Application*, or in the *SAP Web IDE*. When this happens, clean the browser cache and restart the application.

Implement a CDS view for the selection of bookings.

1. Implement a CDS view for the selection of bookings.

Table 89: ABAP CDS Configuration

Property	Value
Name	ZUX410_C_BOOKINGUI00## (replace ## with your group number)
Description	Flight bookings
Package	ZTRAIN_## (replace ## with your group number)
Transport request	Use the request assigned to your group by your instructor
Template for creating New Data Definition	No template

- a) Start *Eclipse*, as described in the exercise , if not already started.
- b) Select your favorite package ZTRAIN\_## in the *Project Explorer* of *Eclipse* and from the context menu, choose *New → Other ABAP Repository Object*.
- c) Expand the Core Data Service folder in the new dialog and choose the entry Data Definition and then *Next*.
- d) Enter the listed attributes from the table in the following dialog and choose *Next*.

- e) Select a transport request and choose *Next*.
  - f) In the next dialog, ensure the checkbox *Use the selected template* is deselected and choose *Finish*.

2. Copy the content of the file `S:\Courses\UX410_20\Templates\zUX410_C_BookingUI##.txt` to the newly created *ABAP CDS* view and replace the `##` in the copied content with your group number.

  - a) Open *Windows file explorer* and navigate to the file `S:\Courses\UX410_20\Templates\zUX410_C_BookingUI##.txt`.
  - b) Open the file `zUX410_C_BookingUI##.txt` and copy the content.
  - c) Go back to *Eclipse* and paste the copied content to the newly created *ABAP CDS*.
  - d) Your *ABAP CDS* will look as shown.

```
[540] ZUX410_C_FLIGHTU00 [540] ZUX410_C_BOOKINGU00 ::  
1 @#abapCatalog,sqlViewName: 'ZUX410Booking##'  
2 @#abapCatalog.compiler.compareFilter: true  
3 @#abapCatalog.compiler.authenticationCheck: 'CHECK'  
4 @#endabapText, label: 'ZUX410Booking##'  
5 define view ZUX410_C_BOOKINGU00 as select from abook {  
6   //abook  
7   key mandt,  
8   key carrid,  
9   key connid,  
10  @UI.lineItem.position: 20  
11  key fidate,  
12  key bookid,  
13  customid,  
14  custtype,  
15  @UI.lineItem.position: 60  
16  smoker,  
17  lugweight,  
18  weight,  
19  @UI.lineItem.position: 50  
20  class,  
21  forcuram,  
22  forcurkey,  
23  @UI.lineItem.position: 40  
24  locoram,  
25  loconyy,  
26  @UI.lineItem.position: 30  
27  order_date,  
28  counter,  
29  agencynum,  
30  cancelled,  
31  reserved,  
32  @UI.lineItem.position: 10  
33  passname,  
34  passform,  
35  passbirth  
36    
37  passbirth  
38 }
```

Figure 337: ABAP CDS

- e) Replace the ## with your group number. For group 00 the result will look as shown.

```
[540] ZUX410_C_FLIGHTUI00 [540] ZUX410_C_BOOKINGUI00 23
1#<BookLog> select * from ZUX410Booking00
2StepCastJavaCompiler.compileFilter: true
3AccessControl.authorizationCheck: #CHECK
4#EndUserText.label: 'bookings'
5define view ZUX410_C_BOOKINGUI00 as select from sbook {
6    //sbook
7    key mandt,
8    key carrier,
9    key bookid,
10    @UI.lineItem.position: 20
11    key fldate
12    key bookid,
13    customid,
14    custtype,
15    @UI.lineItem.position: 60
16    smoker,
17    luggageight,
18    wnum,
19    invoice,
20    @UI.lineItem.position: 50
21    class,
22    forcusram,
23    forcuckey,
24    @UI.lineItem.position: 40
25    location,
26    loccuckey,
27    @UI.lineItem.position: 30
28    order_date,
29    counter,
30    agencycode,
31    cancellation,
32    reserved,
33    @UI.lineItem.position: 10
34    passname,
35    passform,
36
37    passbirth
38 }
```

Figure 338: Result for Group 00

- f) Activate your changes.

## Task 1: Add an association to ZUX410 C FLIGHTUI##.

1. Extend the ZUX410\_C\_FLIGHT##-ABAP CDS as shown in the figure and expose the \_Bookings alias. Remember to use your own group number.

```
[*S4D] ZUX410_C_FLIGHTUI00 & [S4D] ZUX410_C_BOOKINGUI00
1 @AbapCatalog.sqlViewName: 'ZUX410FLIGHTUI00'
2 @AbapCatalog.compiler.compareFilter: true
3 @AccessControl.authorizationCheck: #CHECK
4 @EndUserText.label: 'UX410 Flight UI'
5 @OData.publish: true
6
7 @Search.searchable: true
8 @UI.headerInfo.typeName : 'Flight'
9 @UI.headerInfo.typeNamePlural : 'Flights'
10 @UI.headerInfo.title.value: 'carrid'
11 @UI.headerInfo.description.value: 'carrrname'
12
13
14 define view ZUX410_C_FLIGHTUI00 as select from sflight as sf
15   association to scarr as sc on sf.carrid = sc.carrid
16   association to sflights as sfs on sfs.carrid = sf.carrid and
17   sfs.connid = sf.connid and sfs.fldate = sf.fldate
18   association [*] to ZUX410_C_BOOKINGUI00 as _Bookings
19     on $projection.carrid = _Bookings.carrid and $projection.fldate = _Bookings.fldate and
20     $projection.connid = _Bookings.connid
21 {
22   @UI.lineItem.position:10
23   key sf.carrid,
```

Figure 339: Extend ZUX410 C FLIGHT##-ABAP CDS

- a) Open the implementation of `ZUX410_C_FLIGHTUI##` and add the implementation as shown in the figure.
  - b) Expose the `Bookings` alias.

```
    sf.statsocc_n,
100 @UI.fieldGroup : [
101     {qualifier:'seatdata', position:20}
102 ]
103     sf.seatsmax_f,
104 @UI.fieldGroup : [
105     {qualifier:'seatdata2', position:10}
106 ]
107     sf.seatsocc_f,
108     _Bookings
109 }
```

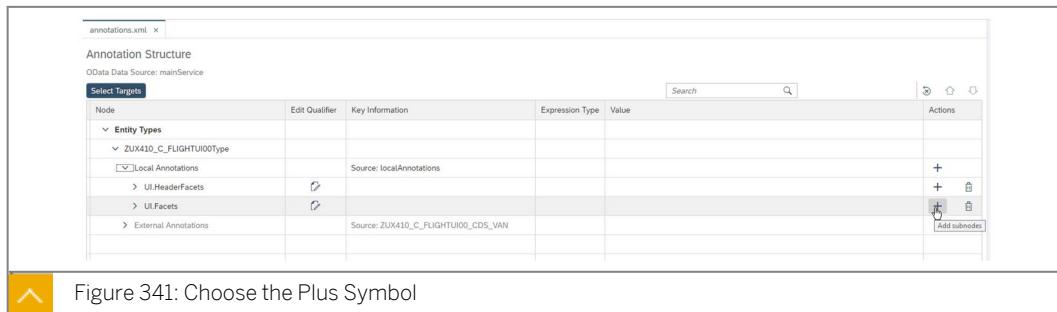
Figure 340: Expose the Alias

- c) Activate your changes.

## Task 2: Add the bookings to the object page as a facet.

1. Add the `to_bookings` navigation property generated by SADL to your local annotations. Add `to_bookings` as a `ReferenceFacet` to the existing `UI.Facets` annotation. Add a property label to the newly created `ReferenceFacet` and assign the text **Bookings** to the `Label` property.
    - a) Start the SAP Web IDE Full-Stack if not already started, as described in previous exercises.
    - b) Close the `annotations.xml` file If already open, and open it again to ensure the new annotations from the ABAP CDS are fetched

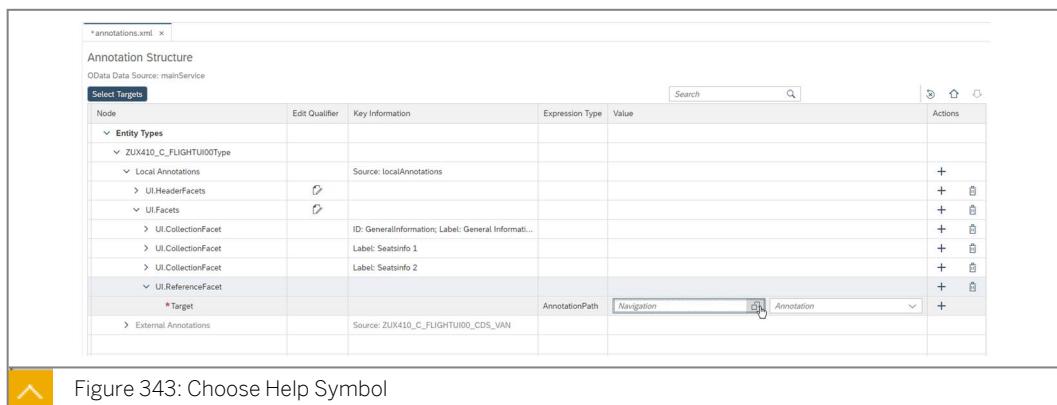
- c) Navigate to *Local Annotations* and click on the + symbol on the level of the already existing *UI.Facets* entry.



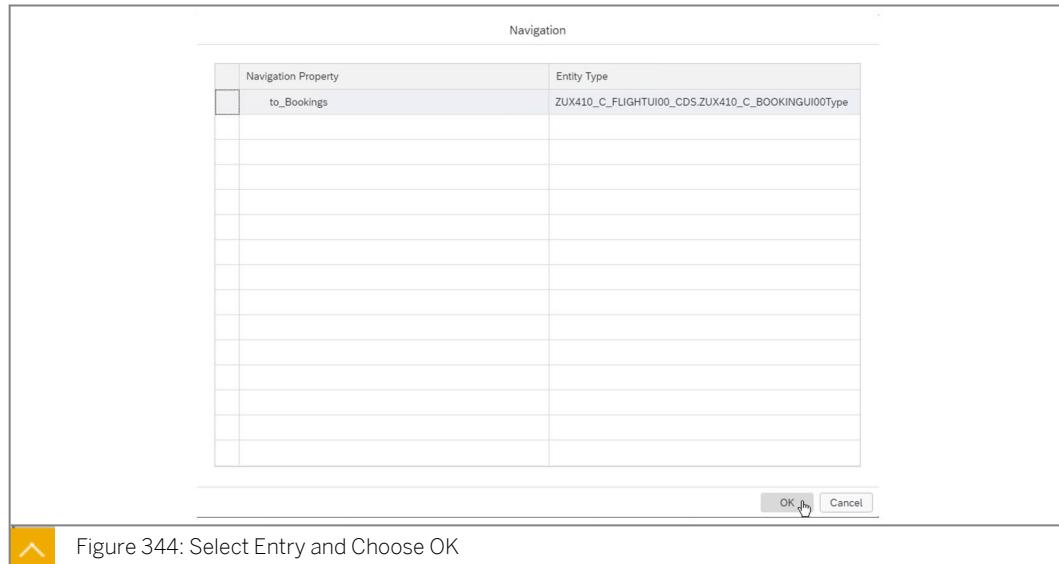
- d) Check the checkbox *ReferenceFacet* in the new dialog and choose *OK*.



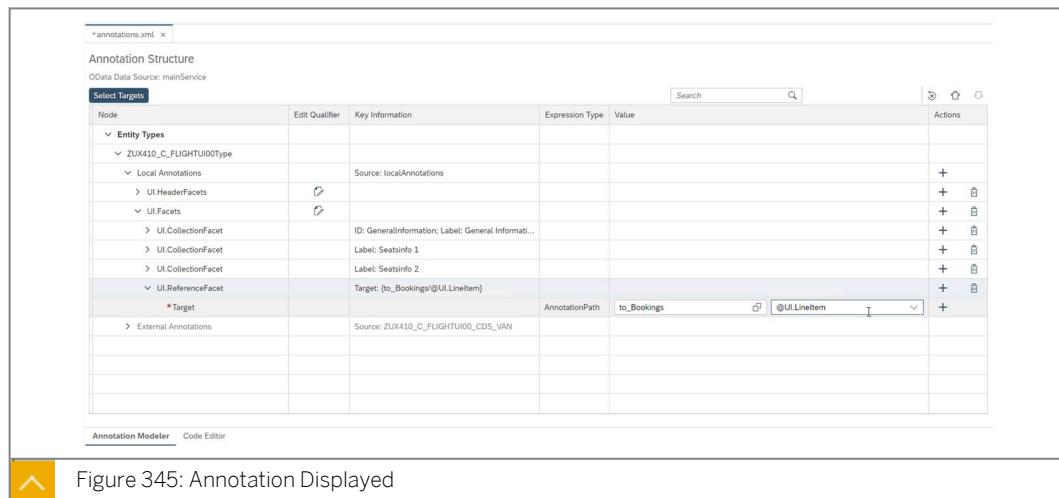
- e) Choose the *F4* help symbol on the *Target* entry.



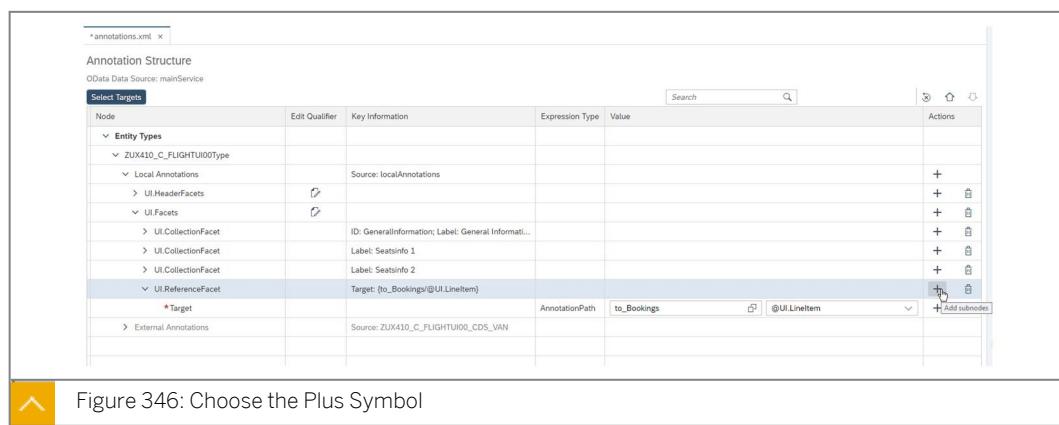
- f) Select the *to\_Bookings* entry in the new dialog and choose *OK*.



g) Your annotation should look as shown when the dialog is closed.



h) Click the + symbol on the level of the currently changed *UI.ReferenceFacet*.



- i) Select the checkbox beside the *Label* property in the following dialog and choose OK.

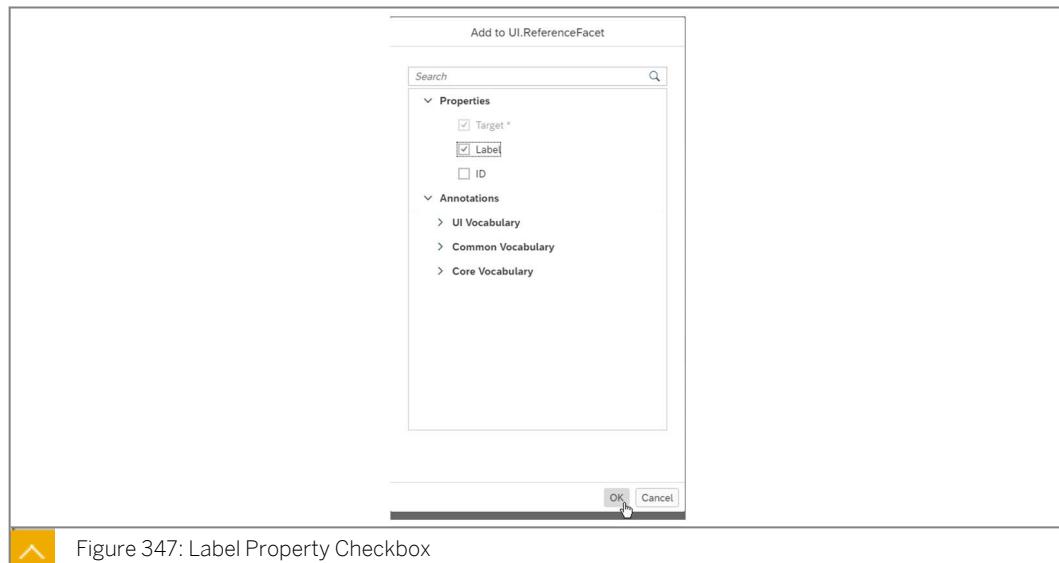


Figure 347: Label Property Checkbox

j) Enter the value **Bookings** in the *Value* field of the newly added property.

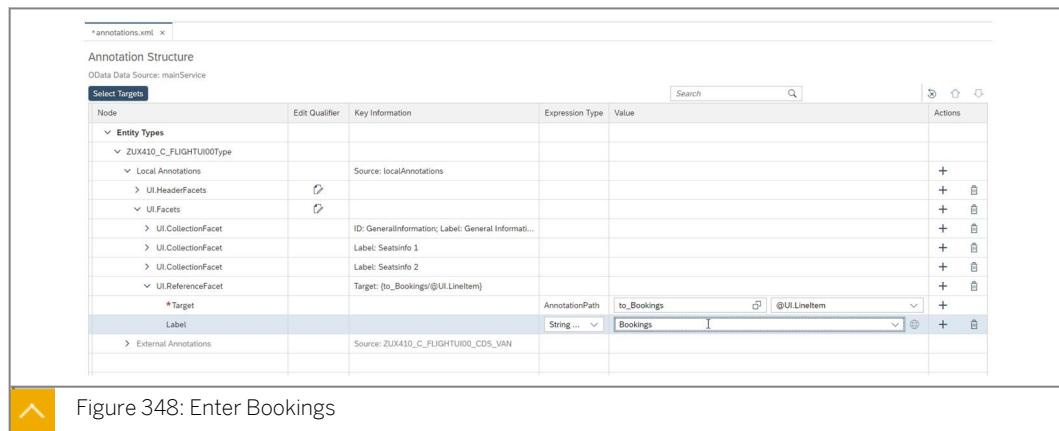


Figure 348: Enter Bookings

k) Save your changes.

2. Test your application.

- Start your application as described in previous exercises.
- Choose the Go button in the *FilterBar* to trigger a get-request.
- Select a flight from the list.
- Select the *Bookings* section to view your result.

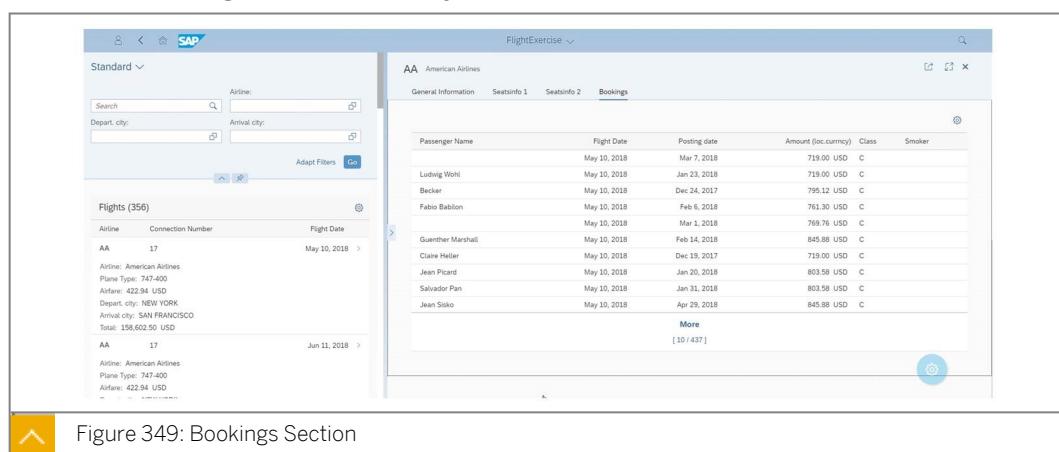


Figure 349: Bookings Section