


Develop SAP Business One extensions on the SAP Cloud Platform





TABLE OF CONTENTS

PREREQUISITES.....	4
i. Download and Install Development Tools	4
ii. Create a SAP Cloud platform trial account	5
iii. Activate Web IDE Full Stack service.....	6
iv. SAP API Business Hub	9
STEP 1: CREATE A SAP FIORI APP CONNECTING TO SAP BUSINESS ONE SERVICE LAYER VIA SAP API BUSINESS HUB	10
i. Create a SAPUI5 Application	10
ii. Add a Data Source to the SAPUI5 Application	13
iii. Add controls to the View1 view.....	16
<i>Add a sap.m.Table control.....</i>	<i>16</i>
<i>Add a Search Field control to the sap.m.Table</i>	<i>21</i>
iv. Add a second view called Details	25
<i>Create a Details view.</i>	<i>25</i>
<i>Add an Object Header control.....</i>	<i>26</i>
v. Define navigation between View1 and Details.....	29
STEP 2: CREATE A NODEJS APP	31
STEP 3: DEPLOY THE NODEJS APP INTO SAP CLOUD FOUNDRY	32
STEP 4: CONSUME THE NODEJS APP FROM THE SAP FIORI APP	36
i. Add a Button “Add Freight” to the Details view.....	36
ii. Implement the Button business logic calling the NodeJS server side and Service Layer....	38



The objective of this hands on is to put in practice how to develop SAP Business One extensions on SAP Cloud Platform.

The exercise will be composed by

- Step 1: Create a Fiori application connecting to SAP Business One Service Layer via SAP API Business Hub
- Step 2: Implement a server side NodeJS application
- Step 3: Deploy the NodeJS application to SAP Cloud Foundry
- Step 4: Consume the server side NodeJS application from the Fiori application

This hands-on exercise will require several steps, please follow them in the proposed order as each step is counting on the precedent steps.

PREREQUISITES

i. Download and Install Development Tools

Download and install git version control on your system from the following link

<https://git-scm.com/downloads>



We will also make use of SAP Cloud Platform Cloud Foundry Environment.

To do so, we need the Cloud Foundry command line interface (CLI)

You can download it and install if the CF CLI for your operating system on.

<https://github.com/cloudfoundry/cli#downloads>

Downloads

Installing using a package manager

Mac OS X and Linux using [Homebrew](#) via the [cloudfoundry tap](#):

```
brew install cloudfoundry/tap/cf-cli
```

Debian and Ubuntu based Linux distributions:

```
# ...first add the Cloud Foundry Foundation public key and package repository to your system
wget -q -O - https://packages.cloudfoundry.org/debian/cli.cloudfoundry.org.key | sudo apt-key add -
echo "deb https://packages.cloudfoundry.org/debian stable main" | sudo tee /etc/apt/sources.list.d/cloudfo
# ...then, update your local package index, then finally install the cf CLI
sudo apt-get update
sudo apt-get install cf-cli
```

Enterprise Linux and Fedora systems (RHEL6/CentOS6 and up):

```
# ...first configure the Cloud Foundry Foundation package repository
sudo wget -O /etc/yum.repos.d/cloudfoundry-cli.repo https://packages.cloudfoundry.org/fedora/cloudfoundry-c
# ...then, install the cf CLI (which will also download and add the public key to your system)
sudo yum install cf-cli
```

Installers and compressed binaries

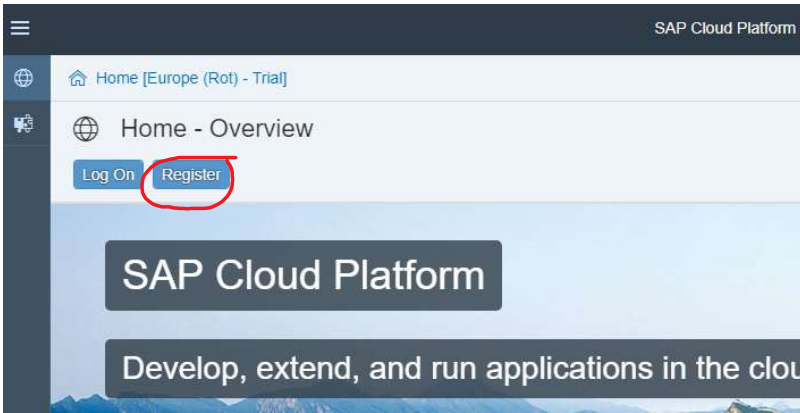
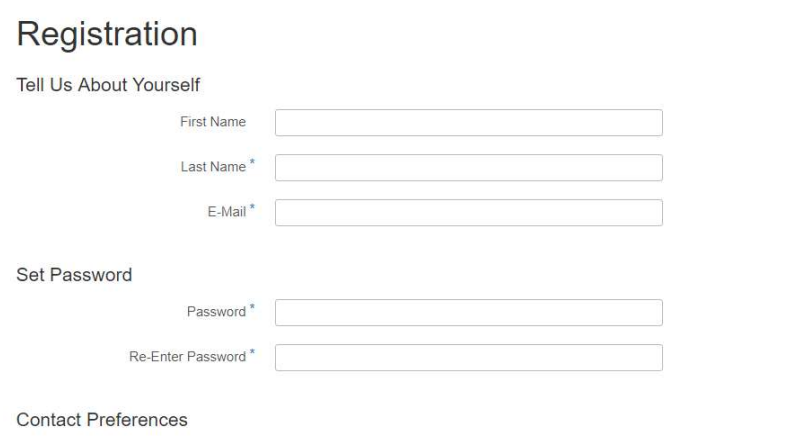
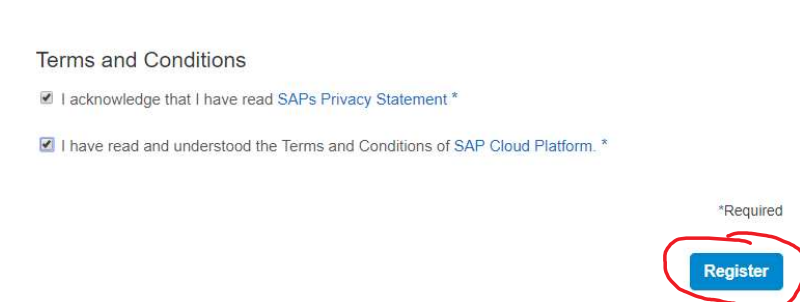
	Mac OS X 64 bit	Windows 64 bit	Linux 64 bit
Installers	pkg	zip	rpm / deb
Binaries	tgz	zip	tgz

ii. Create a SAP Cloud platform trial account

The exercises proposed in this hands on are implemented on top of the SAP Cloud Platform.

If you have already a trial SAP Cloud Platform account, you can skip this step.

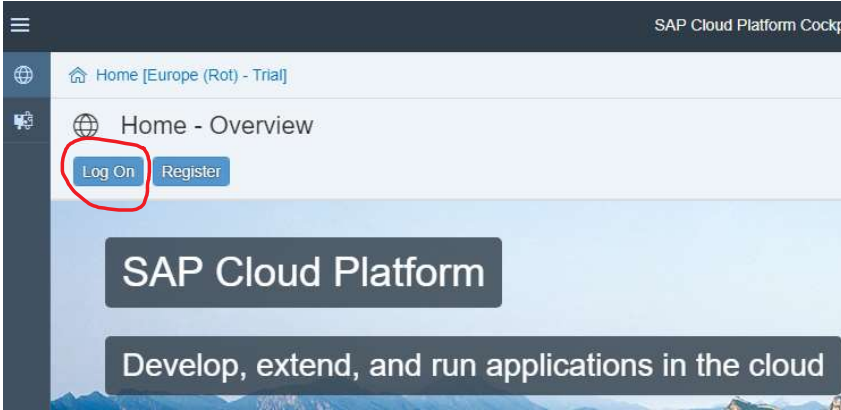
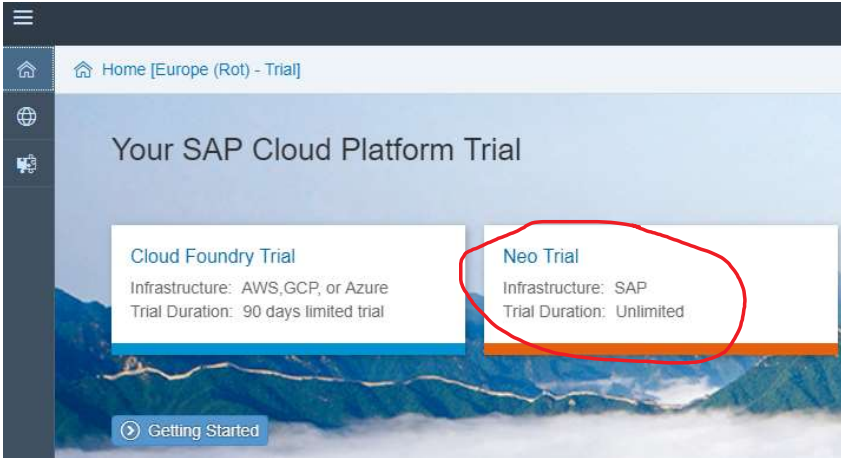
To create a trial SAP Cloud Platform account, go to the following link:

Explanation	Screenshot
<p>To create a trial SAP Cloud Platform account, go to the following link:</p> <p>https://account.hanatrial.ondemand.com</p> <p>Press the Register button</p>	 <p>The screenshot shows the SAP Cloud Platform Home - Overview page. At the top, there is a navigation bar with a home icon and the text 'Home [Europe (Rot) - Trial]'. Below this, there is a 'Home - Overview' section with a 'Log On' button and a 'Register' button. The 'Register' button is circled in red. Below the navigation bar, there is a large banner with the text 'SAP Cloud Platform' and 'Develop, extend, and run applications in the cloud'.</p>
<p>Enter all your details</p>	 <p>The screenshot shows the SAP Cloud Platform Registration form. The form is titled 'Registration' and has three main sections: 'Tell Us About Yourself', 'Set Password', and 'Contact Preferences'. The 'Tell Us About Yourself' section has three input fields: 'First Name', 'Last Name *', and 'E-Mail *'. The 'Set Password' section has two input fields: 'Password *' and 'Re-Enter Password *'. The 'Contact Preferences' section is currently empty. At the bottom right of the form, there is a blue 'Register' button, which is circled in red.</p>
<p>Accept the terms and conditions by checking both check boxes and press "Register".</p>	 <p>The screenshot shows the 'Terms and Conditions' section of the registration form. It contains two checked checkboxes: 'I acknowledge that I have read SAPs Privacy Statement *' and 'I have read and understood the Terms and Conditions of SAP Cloud Platform. *'. Below the checkboxes, there is a blue 'Register' button, which is circled in red. A small asterisk and the word 'Required' are visible above the button.</p>

iii. Activate Web IDE Full Stack service

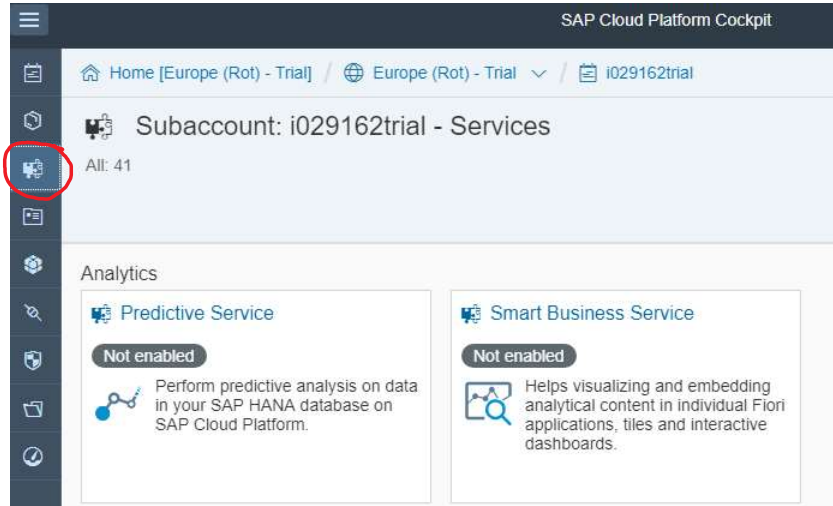
We will use Web IDE Full Stack for the creation and implementation of our application. Web IDE is offered as a service on the SAP Cloud Platform.

To activate Web IDE Full Stack service please follow the steps here below, if you already have Web IDE Full Stack service active in your account please skip this step.

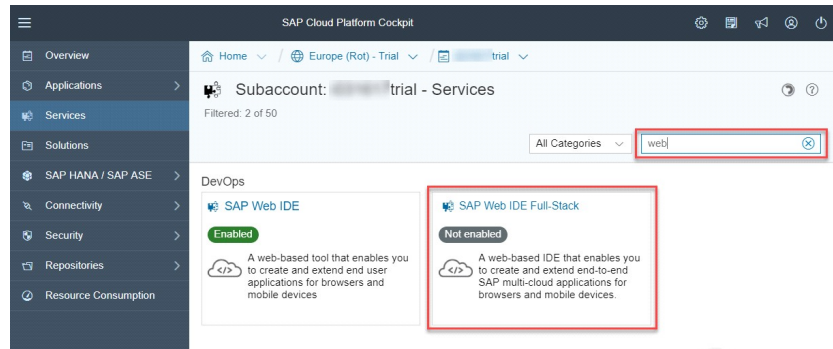
Explanation	Screenshot
<p>Open your trial SAP Cloud Platform account from the following link: https://account.hanatrial.ondemand.com</p>	
<p>Press the Log On button if you are not automatically logged in</p>	 A screenshot of the SAP Cloud Platform Home - Overview page. The page has a dark blue header with a hamburger menu icon on the left and the text 'SAP Cloud Platform Cockp' on the right. Below the header, there is a navigation bar with a home icon and the text 'Home [Europe (Rot) - Trial]'. Below that, there is a section titled 'Home - Overview' with a globe icon. Underneath, there are two buttons: 'Log On' and 'Register'. The 'Log On' button is circled in red. Below this, there is a large banner with the text 'SAP Cloud Platform' and 'Develop, extend, and run applications in the cloud'.
<p>After login if you are proposed between Cloud Foundry Trial and Neo Trial please choose Neo Trial.</p>	 A screenshot of the 'Your SAP Cloud Platform Trial' page. The page has a dark blue header with a hamburger menu icon on the left and the text 'Home [Europe (Rot) - Trial]' on the right. Below the header, there is a navigation bar with a home icon and the text 'Home [Europe (Rot) - Trial]'. Below that, there is a section titled 'Your SAP Cloud Platform Trial'. Underneath, there are two trial options: 'Cloud Foundry Trial' and 'Neo Trial'. The 'Neo Trial' option is circled in red. Below these options, there is a 'Getting Started' button.

Explanation	Screenshot
-------------	------------

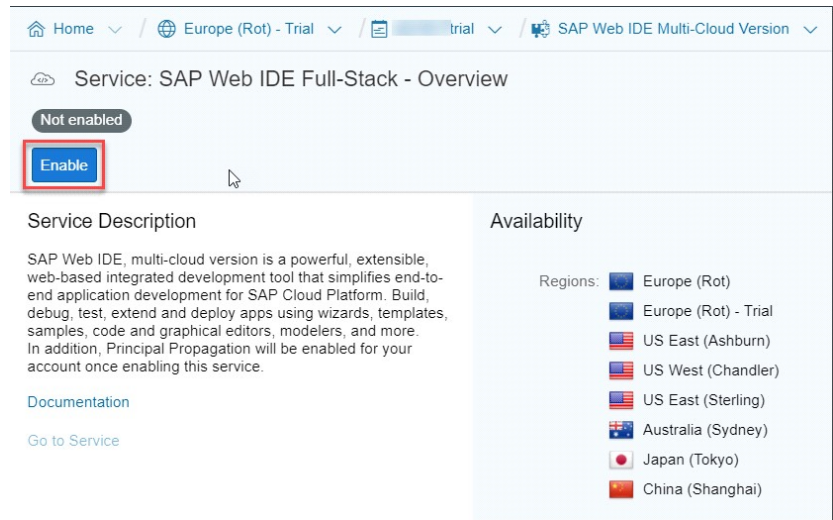
Select the Services icon on the left side bar.

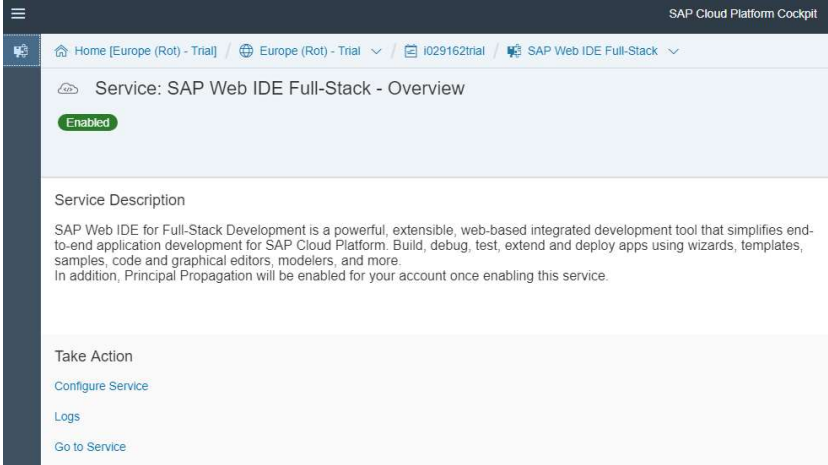
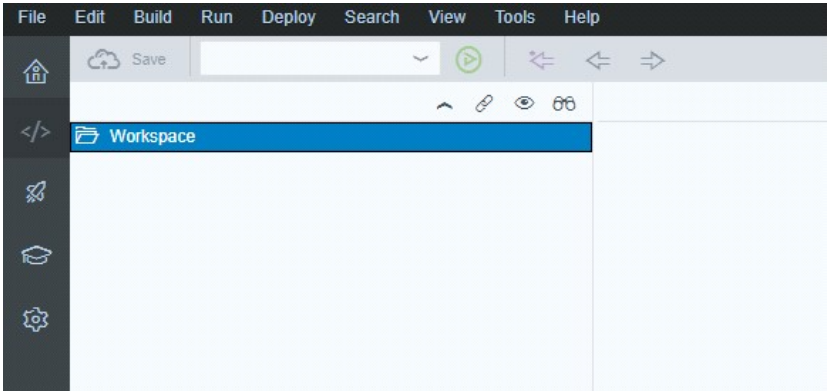


Enter **Web** in the search edit text.
Click on the SAP Web IDE Full Stack box.



Click **Enable**.
This may take a few minutes.



Explanation	Screenshot
<p>Once Enabled select the link Go to Service to open Web IDE Full Stack.</p>	 <p>The screenshot shows the SAP Cloud Platform Cockpit interface. At the top, it says 'SAP Cloud Platform Cockpit'. Below that, there's a breadcrumb trail: 'Home [Europe (Rot) - Trial] / Europe (Rot) - Trial / i029162trial / SAP Web IDE Full-Stack'. The main heading is 'Service: SAP Web IDE Full-Stack - Overview' with a green 'Enabled' button. Under 'Service Description', it states: 'SAP Web IDE for Full-Stack Development is a powerful, extensible, web-based integrated development tool that simplifies end-to-end application development for SAP Cloud Platform. Build, debug, test, extend and deploy apps using wizards, templates, samples, code and graphical editors, modelers, and more. In addition, Principal Propagation will be enabled for your account once enabling this service.' The 'Take Action' section contains three links: 'Configure Service', 'Logs', and 'Go to Service'.</p>
<p>Web IDE opens with an empty Workspace unless you already developed applications with Web IDE in the past.</p>	 <p>The screenshot shows the SAP Web IDE Full-Stack workspace. It features a menu bar with 'File', 'Edit', 'Build', 'Run', 'Deploy', 'Search', 'View', 'Tools', and 'Help'. Below the menu is a toolbar with a 'Save' button, a search icon, and navigation arrows. The main workspace area is empty, with a 'Workspace' folder icon visible in the left sidebar. The sidebar also contains icons for home, code editor, and settings.</p>

iv. SAP API Business Hub

SAP API Business Hub is the central catalog of all SAP and partner APIs for developers to build sample apps, extensions and open integrations with SAP.

SAP Business One has exposed 3 packages as of today:

- SAP Business One – Sales
- SAP Business One – Business Partners
- SAP Business One - Inventory

Many other packages will follow very soon, stay tuned.

Please go to the following link for more details:

<https://api.sap.com>

STEP 1: CREATE A SAP FIORI APP CONNECTING TO SAP BUSINESS ONE SERVICE LAYER VIA SAP API BUSINESS HUB

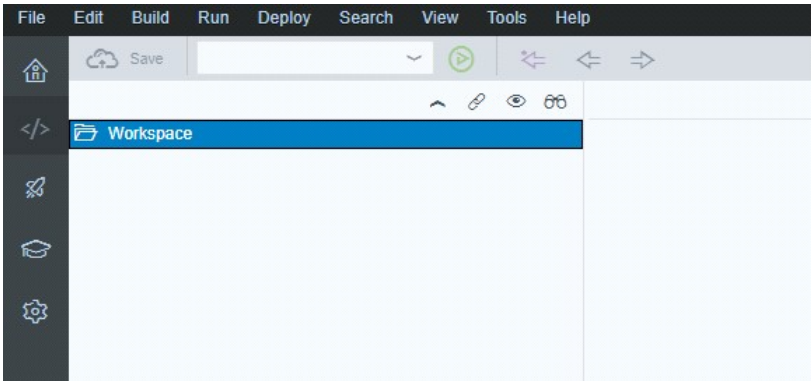
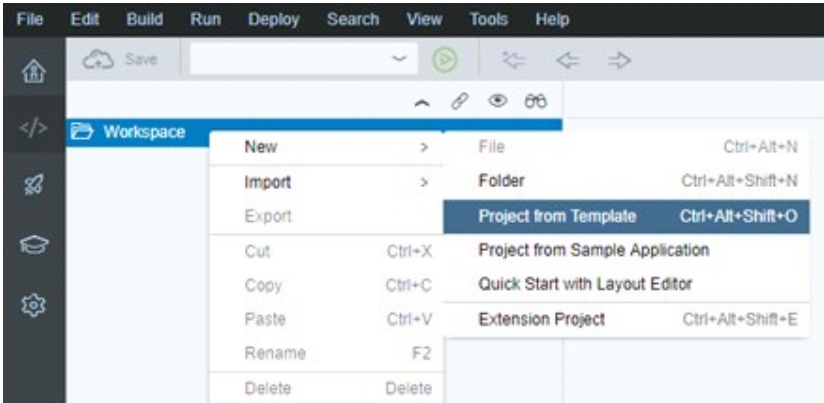
The objective of this first exercise is to develop a SAP Fiori app using the **SAP UI5** template.

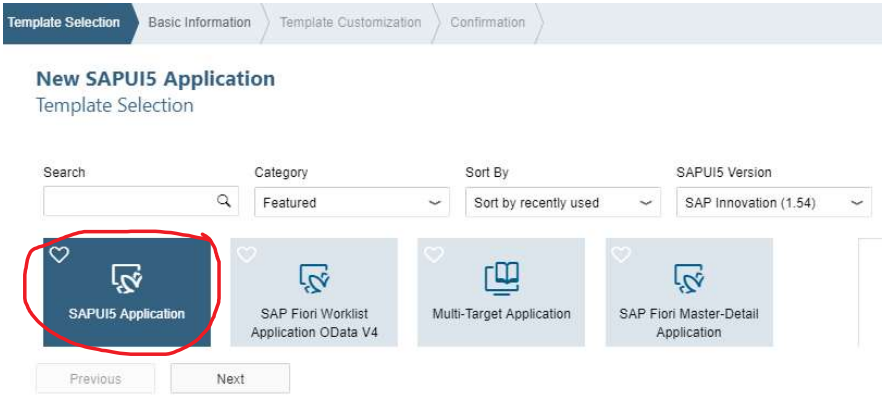
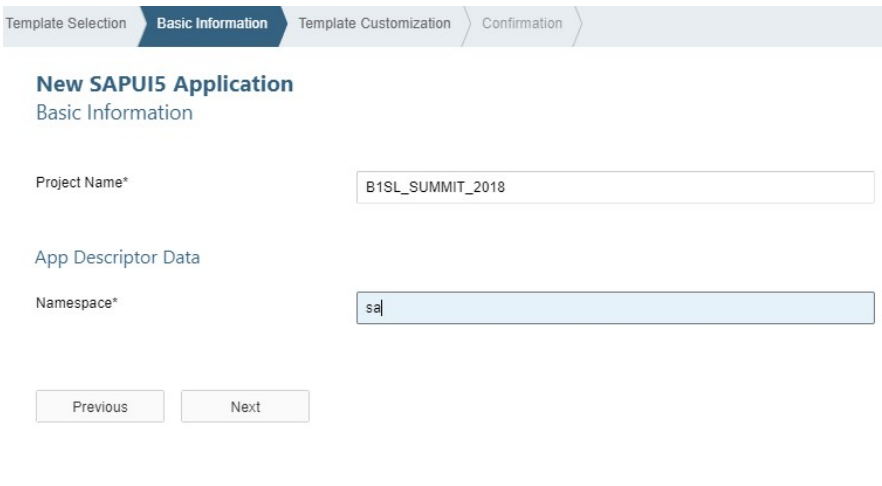
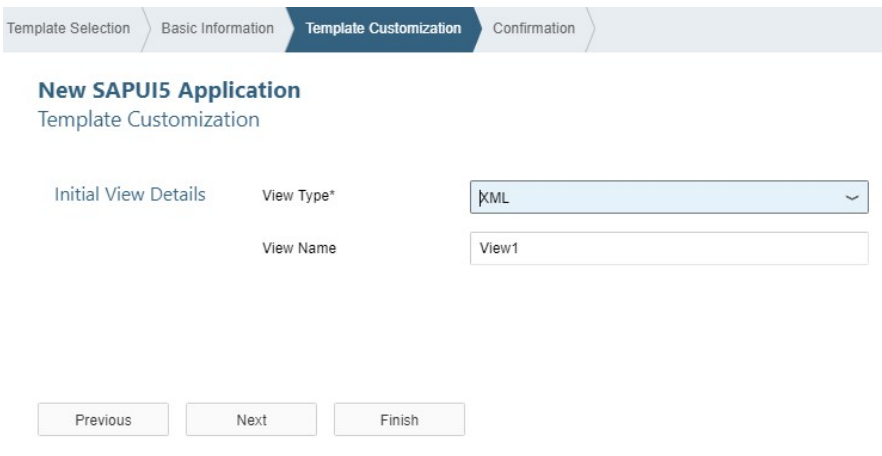
Service Layer provides OData v4 support since SAP Business One 9.3 PL04 version for SAP HANA.

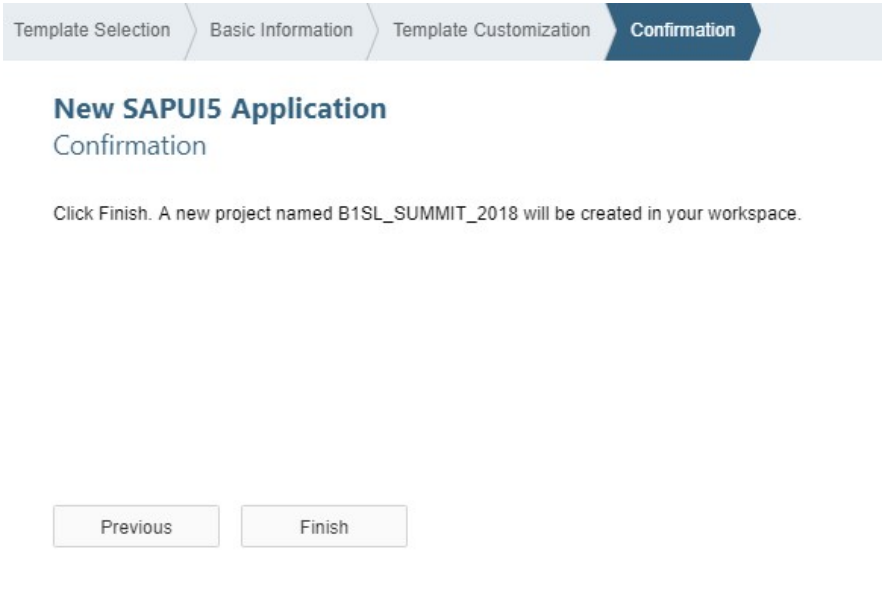
In this exercise, we will use the Service Layer APIs exposed via the SAP API Business Hub, please refer to the prerequisites section SAP API Business Hub for more details.

Web IDE supports OData v4 on some templates like SAP Fiori Worklist Application OData v4 and SAPUI5. More templates will support OData v4 gradually. In this exercise, we will use the SAPUI5 template.

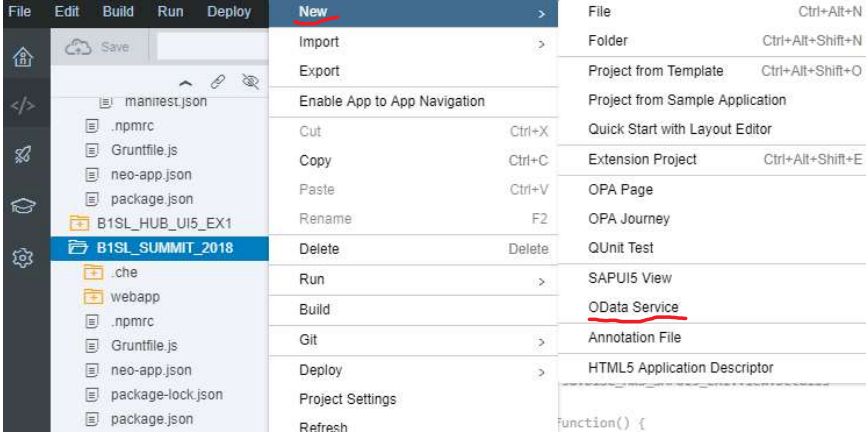
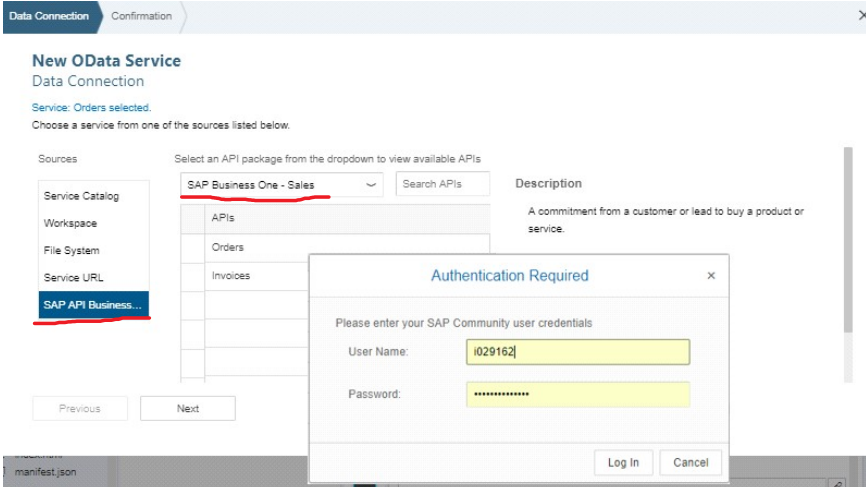
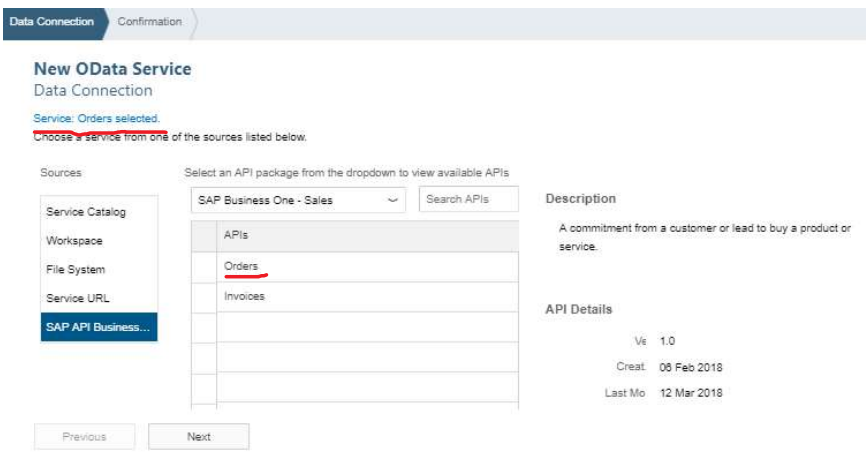
i. Create a SAPUI5 Application

Explanation	Screenshot
<p>Open SAP Web IDE Full Stack.</p> <p>Check the prerequisites sections “Create a SAP Cloud platform trial account” and “Activate Web IDE Full Stack service” if you don’t know how to open WebIDE Full Stack.</p>	 The screenshot shows the SAP Web IDE Full Stack interface. At the top, there is a menu bar with 'File', 'Edit', 'Build', 'Run', 'Deploy', 'Search', 'View', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for 'Save', a search icon, a play icon, and navigation arrows. The main workspace area is currently empty, with a 'Workspace' folder icon visible in the left sidebar. The sidebar also contains icons for home, code editor, and settings.
<p>Right click on your Workspace and select New -> Project from Template.</p>	 The screenshot shows the SAP Web IDE Full Stack interface with a context menu open over the 'Workspace' folder. The menu items are: 'New' (with a right arrow), 'Import' (with a right arrow), 'Export', 'Cut' (with keyboard shortcut 'Ctrl+X'), 'Copy' (with keyboard shortcut 'Ctrl+C'), 'Paste' (with keyboard shortcut 'Ctrl+V'), 'Rename' (with keyboard shortcut 'F2'), and 'Delete' (with keyboard shortcut 'Delete'). The 'New' menu is expanded, showing sub-items: 'File' (with keyboard shortcut 'Ctrl+Alt+N'), 'Folder' (with keyboard shortcut 'Ctrl+Alt+Shift+N'), 'Project from Template' (with keyboard shortcut 'Ctrl+Alt+Shift+O' and highlighted), 'Project from Sample Application', 'Quick Start with Layout Editor', and 'Extension Project' (with keyboard shortcut 'Ctrl+Alt+Shift+E').

Explanation	Screenshot
<p>Select the SAPUI5 Application template.</p> <p>Press Next.</p> <p>If you don't see this template, change the Category to All categories.</p>	 <p>The screenshot shows the 'New SAPUI5 Application' wizard in the 'Template Selection' step. The breadcrumb navigation at the top includes 'Template Selection', 'Basic Information', 'Template Customization', and 'Confirmation'. Below the title, there are filters for 'Search', 'Category' (set to 'Featured'), 'Sort By' (set to 'Sort by recently used'), and 'SAPUI5 Version' (set to 'SAP Innovation (1.54)'). Four application templates are displayed as cards, each with a heart icon. The first card, 'SAPUI5 Application', is circled in red. The other cards are 'SAP Fiori Worklist Application OData V4', 'Multi-Target Application', and 'SAP Fiori Master-Detail Application'. At the bottom, there are 'Previous' and 'Next' buttons.</p>
<p>Enter a Project Name.</p> <p>Enter a Namespace.</p> <p>Press Next.</p>	 <p>The screenshot shows the 'New SAPUI5 Application' wizard in the 'Basic Information' step. The breadcrumb navigation at the top includes 'Template Selection', 'Basic Information', 'Template Customization', and 'Confirmation'. Below the title, there are two input fields: 'Project Name*' with the value 'B1SL_SUMMIT_2018' and 'Namespace*' with the value 'sa '. At the bottom, there are 'Previous' and 'Next' buttons.</p>
<p>Keep the Initial View Details with the default values.</p> <p>Press Next.</p>	 <p>The screenshot shows the 'New SAPUI5 Application' wizard in the 'Template Customization' step. The breadcrumb navigation at the top includes 'Template Selection', 'Basic Information', 'Template Customization', and 'Confirmation'. Below the title, there are two input fields: 'View Type*' with a dropdown menu set to 'XML' and 'View Name' with the value 'View1'. At the bottom, there are 'Previous', 'Next', and 'Finish' buttons.</p>

Explanation	Screenshot
<p>Press Finish to confirm the creation of the SAPUI5 template.</p>	 <p>Template Selection > Basic Information > Template Customization > Confirmation</p> <h3>New SAPUI5 Application</h3> <h4>Confirmation</h4> <p>Click Finish. A new project named B1SL_SUMMIT_2018 will be created in your workspace.</p> <p>Previous Finish</p>

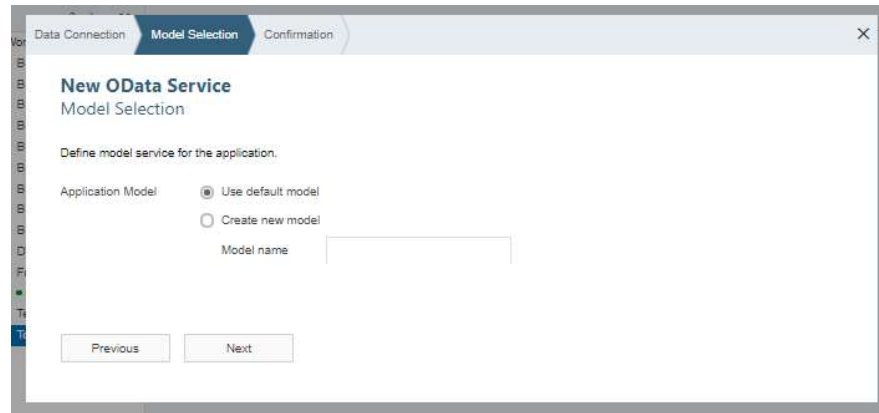
ii. Add a Data Source to the SAPUI5 Application

Explanation	Screenshot
<p>In Web IDE select your project and right click to get the menus.</p> <p>Select New -> OData Service.</p>	 <p>The screenshot shows the SAP Web IDE interface. The 'New' menu is open, and 'OData Service' is highlighted in red. The file explorer on the left shows the project structure, including files like 'manifest.json', 'package.json', and 'B1SL_SUMMIT_2018'.</p>
<p>Select SAP API Business Hub in the Sources (left side of the screen).</p> <p>Select the “SAP Business One – Sales” API package from the dropdown control.</p> <p>You might be prompted to enter your SAP Community User Name and Password. Enter your credentials and press Log In.</p>	 <p>The screenshot shows the 'New OData Service' dialog box. The 'Sources' list on the left has 'SAP API Business...' selected. The 'SAP Business One - Sales' package is selected in the dropdown. An 'Authentication Required' popup is overlaid, prompting for SAP Community user credentials (User Name: 029162, Password: masked).</p>
<p>From the available APIs presented select Orders and make sure the “Service: Orders selected” blue message is shown at the top of the Data Connection tab.</p> <p>Press Next.</p>	 <p>The screenshot shows the 'New OData Service' dialog box. The 'Orders' API is selected in the list. The 'Service: Orders selected.' message is visible at the top of the dialog. The 'Next' button is highlighted.</p>

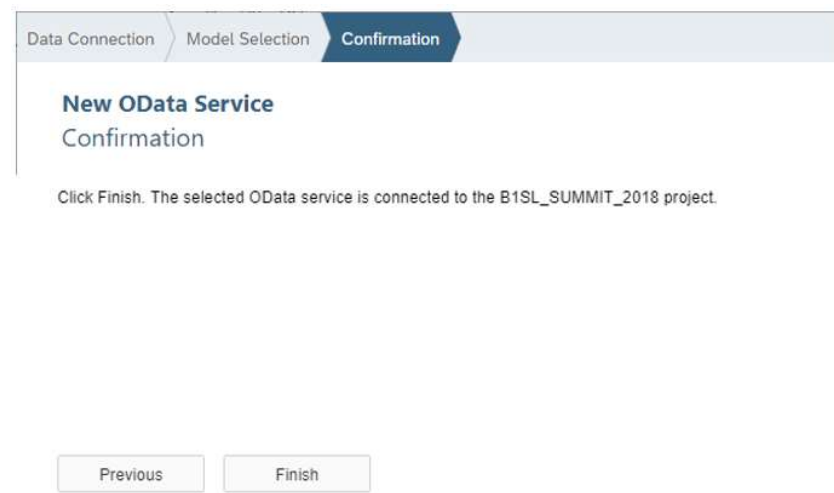
Explanation	Screenshot
-------------	------------

In the Model Selection step keep the radio button “**Use default model**” to create a default model associated to our OData Service.

Press **Next**.



Press **Finish**.

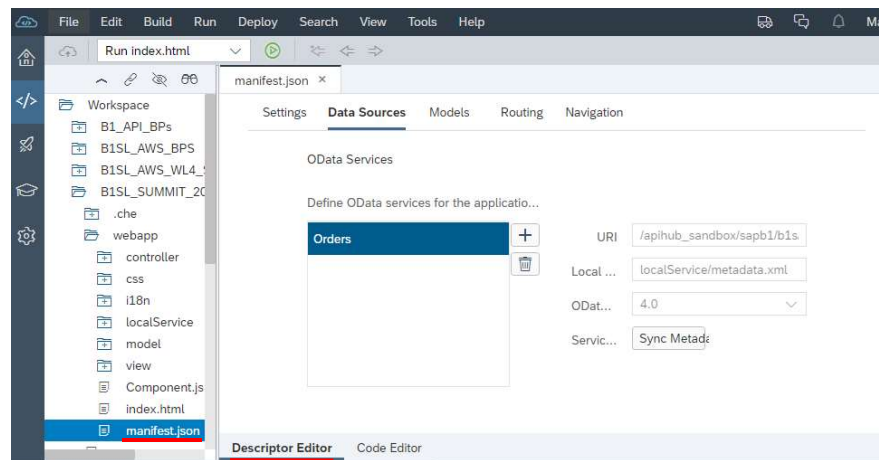


A new Data Source and its corresponding Default model is added automatically to your project.

Right click on the **webapp/manifest.json** file and select **Open With -> Descriptor Editor**.

Select the **Data Sources** tab.

Check that the Orders OData service has been added, points to the API Hub url and the version is 4.0.



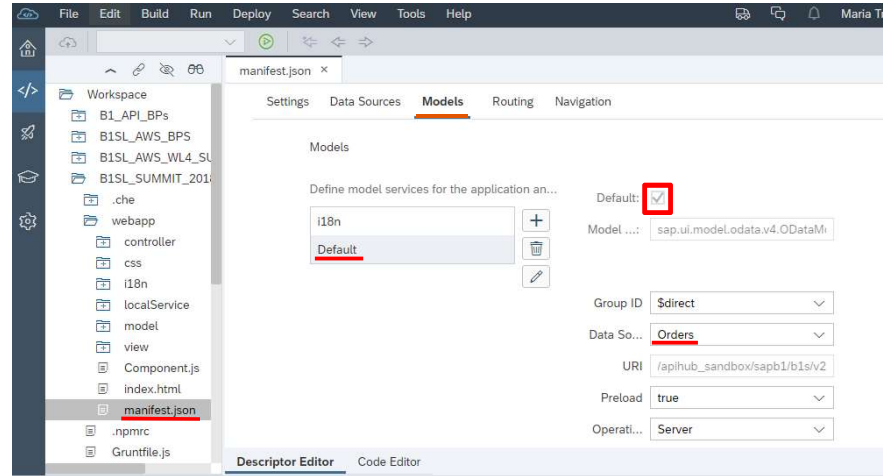
Explanation

In the same file webapp/manifest.json as in previous step select the **Models** tab.

Select the **Default** model and check:

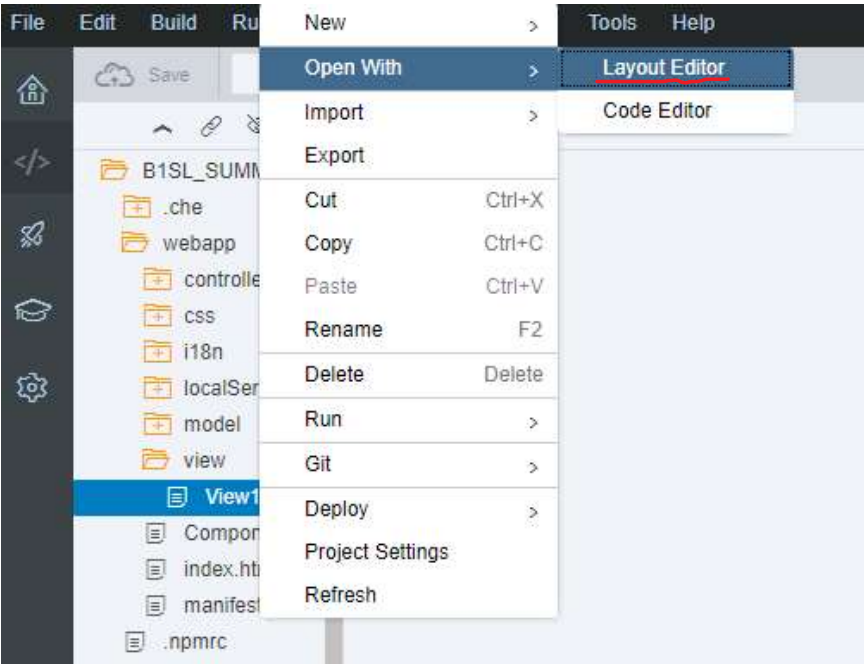
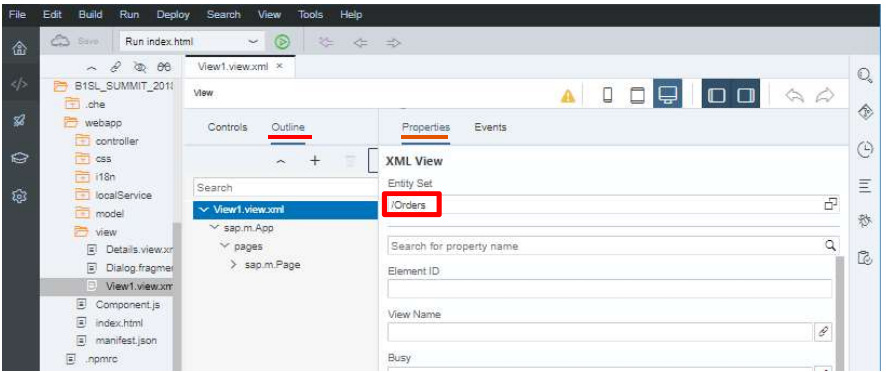

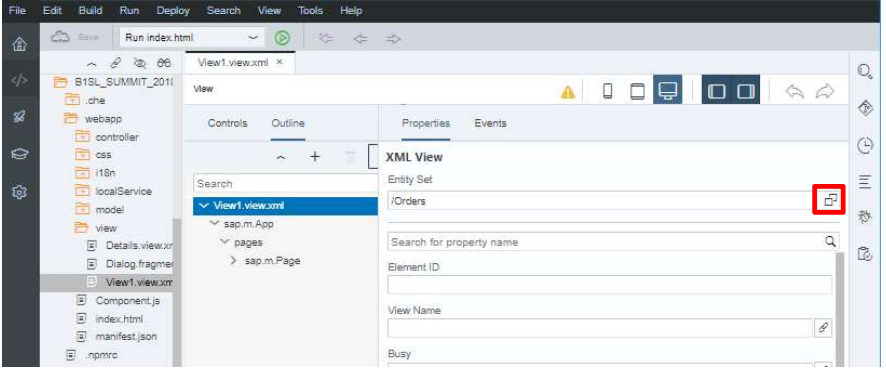
- the Default checkbox is selected
- the Data Source is Orders

Screenshot



iii. Add controls to the View1 view.

Select the Entity Type

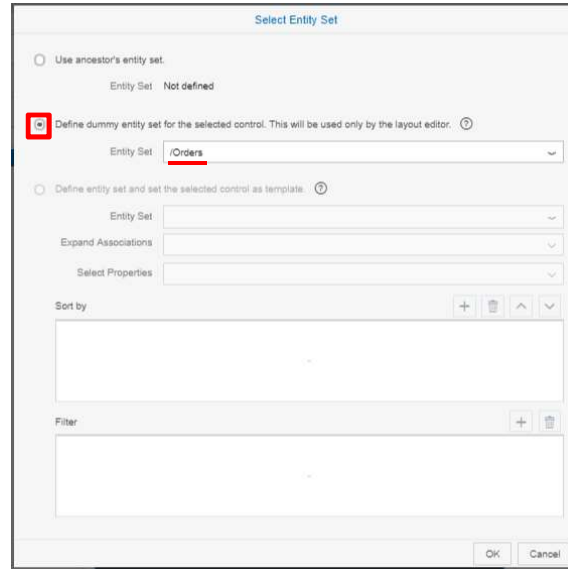
Explanation	Screenshot
<p>Open the view folder.</p> <p>Open the View1.view.xml file with the Layout Editor.</p>	
<p>Select the Outline tab.</p> <p>Select View.view.xml.</p> <p>On the Properties tab (right side of the screen Layout view screen) check if /Orders is already entered as the associated Entity Set.</p>	
<p>If /Orders is not yet entered follow the next steps to define the Entity Set.</p> <p>If /Orders is already available go to the next section "Add a sap.m.Table control".</p> <p>Click on the  button to select the Entity Set associated to our view.</p>	

Explanation

Check the second option **Define dummy entity set...** and choose the **/Orders** Entity Set.

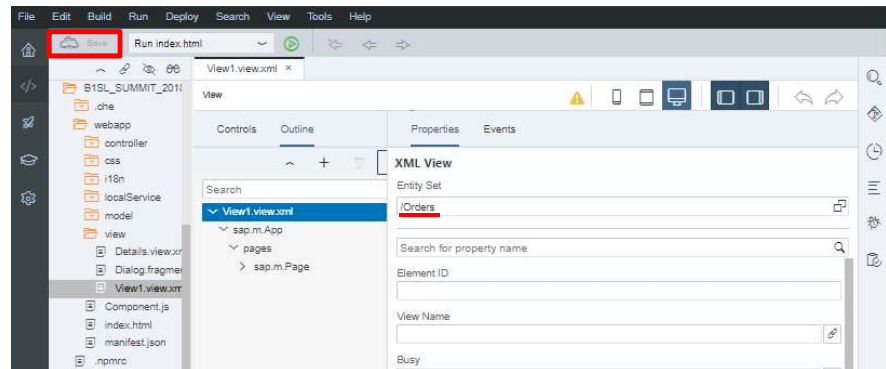
Press **OK**.

Screenshot



Once back to the Layout Editor the **/Orders** entity should be shown.

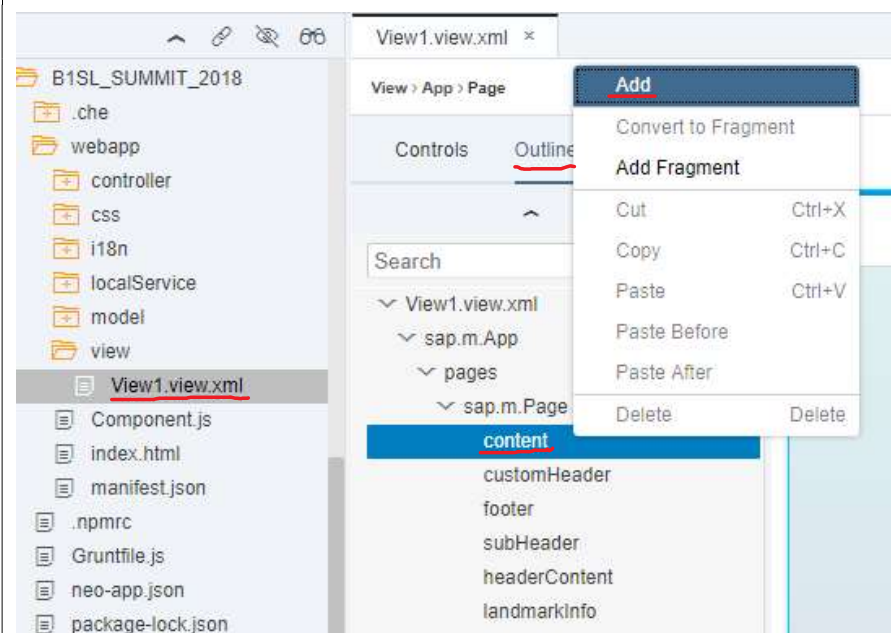
Press **Save** button.



Add a sap.m.Table control

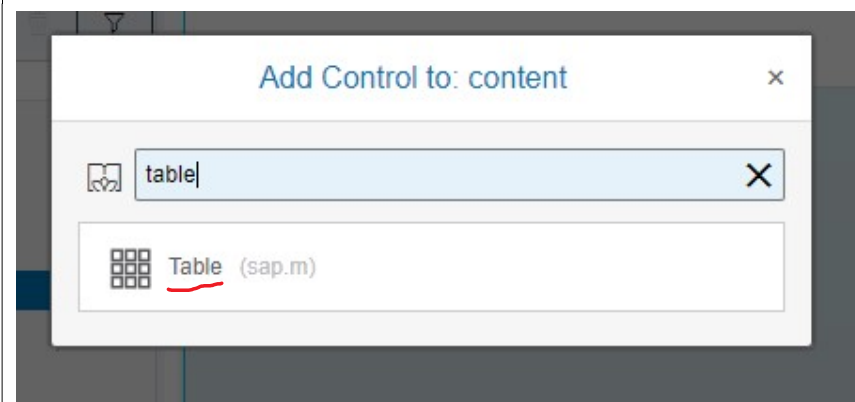
Expand **sap.m.Page**.

Right click on content and select **Add**.



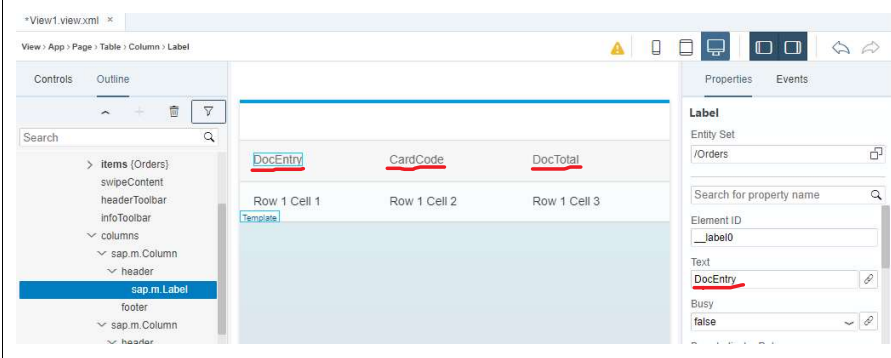
Enter **table** to filter the controls shown in the list.

Click on the **Table** control so it will be added to our view.



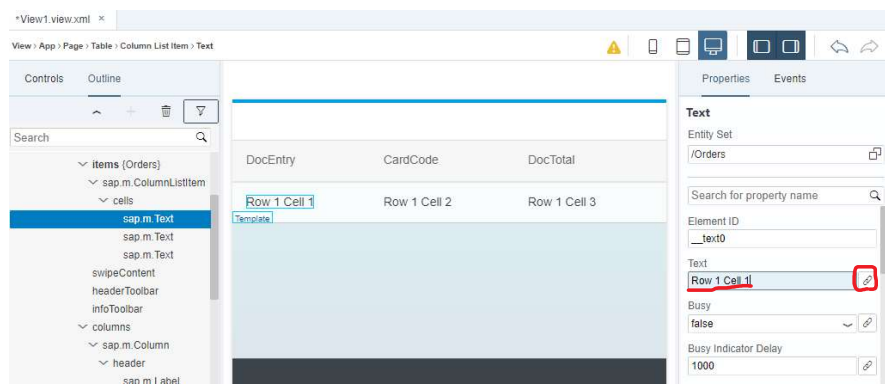
On the table created edit each **sap.m.Table** -> **columns** -> **sap.m.Column** -> **header** -> **sap.m.Label** Text Property:

- Change Header 1 by **DocEntry**.
- Change Header 2 by **CardCode**.
- Change Header 3 by **DocTotal**.



Select each one of the **items** -> **sap.m.ColumnListItem** -> **cells** -> **sap.m.Text** cell to bind them to their corresponding Orders properties.

On the **Text property** click the **Binding** button on the right side of the Text property.

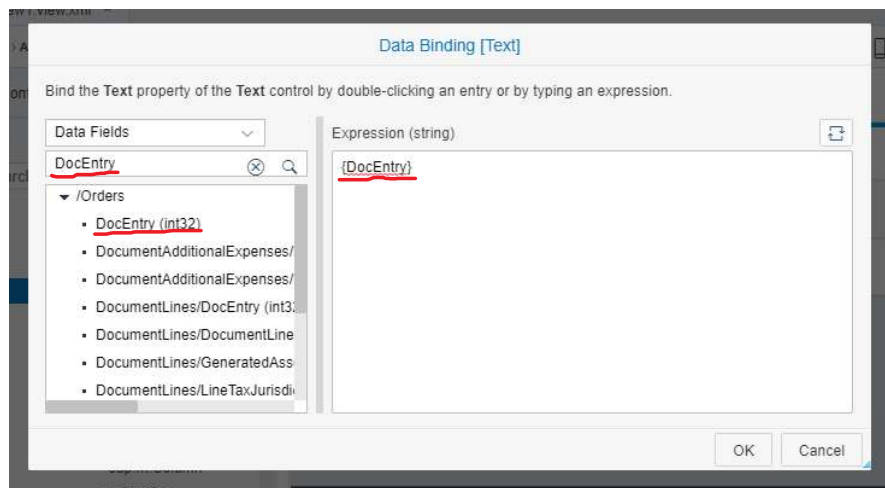


Delete the default Expression string.

In the Search for data field enter the name of the Orders property **DocEntry**.

Double click on the field name so it will be copied to the Expression string between curly brackets.


Repeat this step for CardCode column.

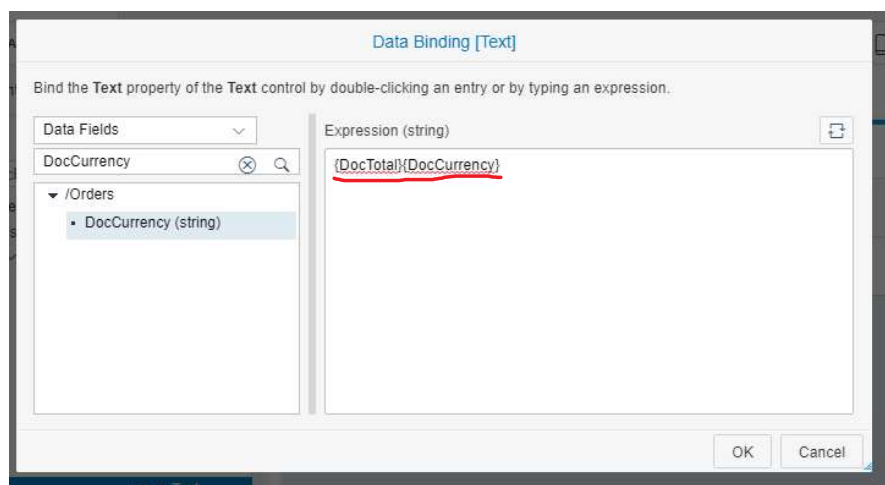


For the **DocTotal** column we will add 2 properties:

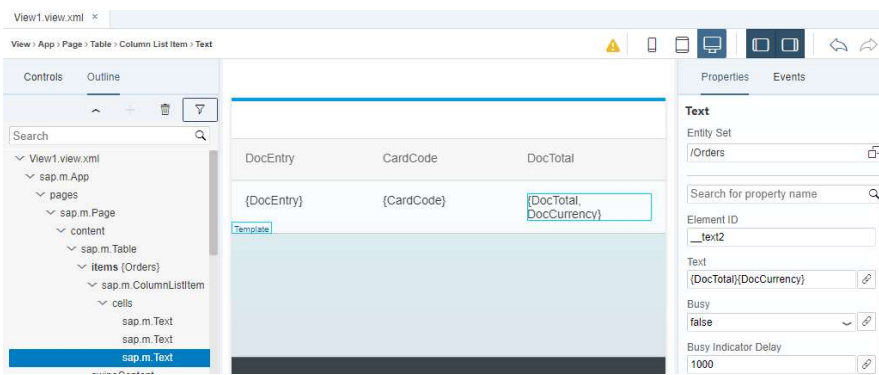
- **DocTotal**
- **DocCurrency**

We will then search and add first the **DocTotal** and afterwards search and add the **DocCurrency**.

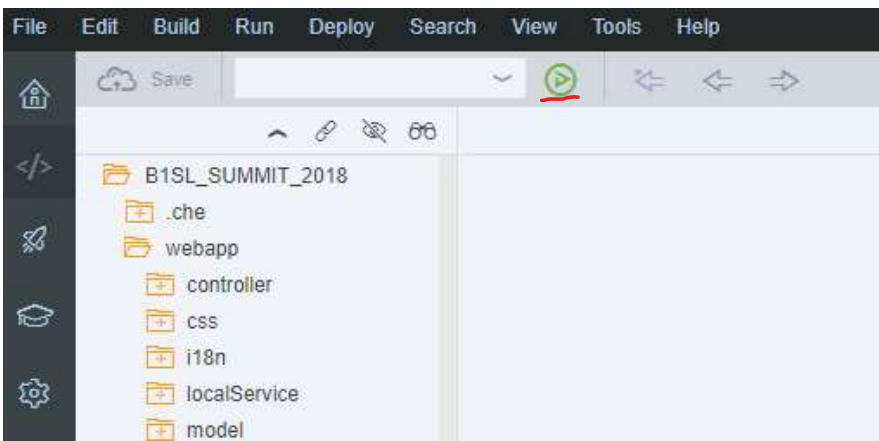
Press **Save** button  on the View1.view.xml file.



Your **View1.view.xml** layout should look like this at this point.



Click on the green arrow to run your application.

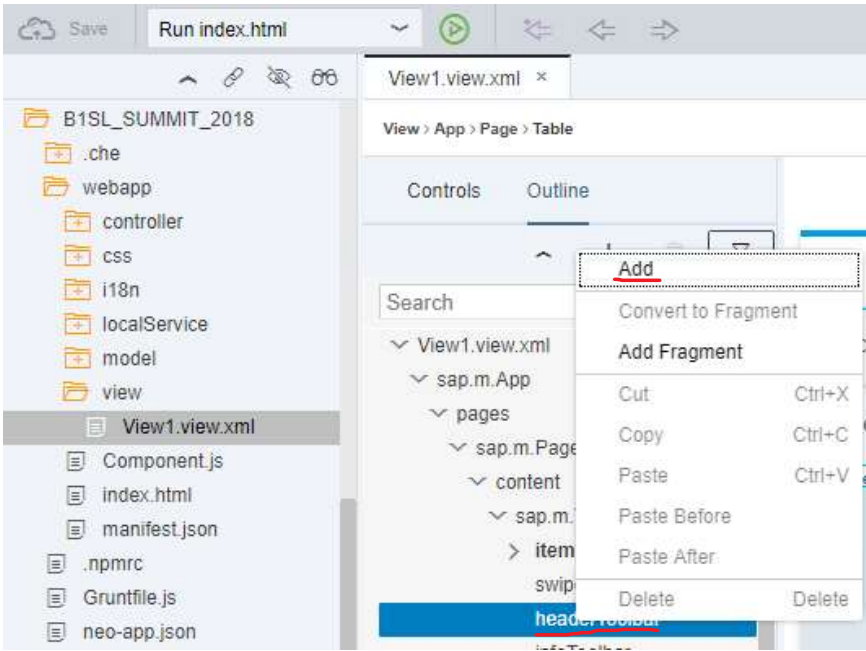
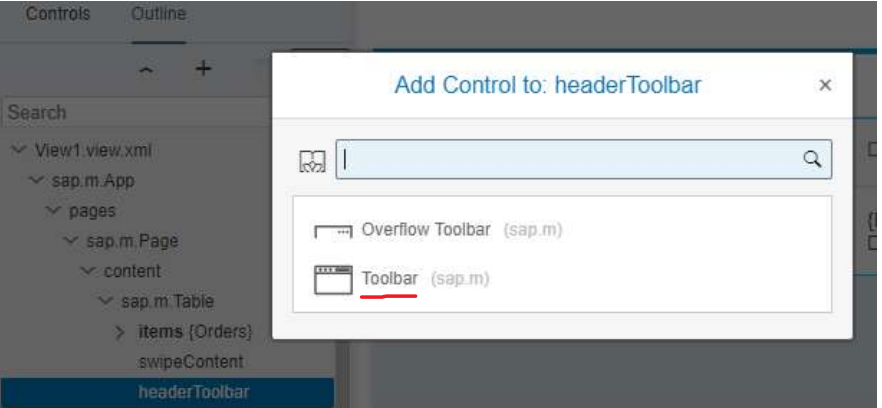


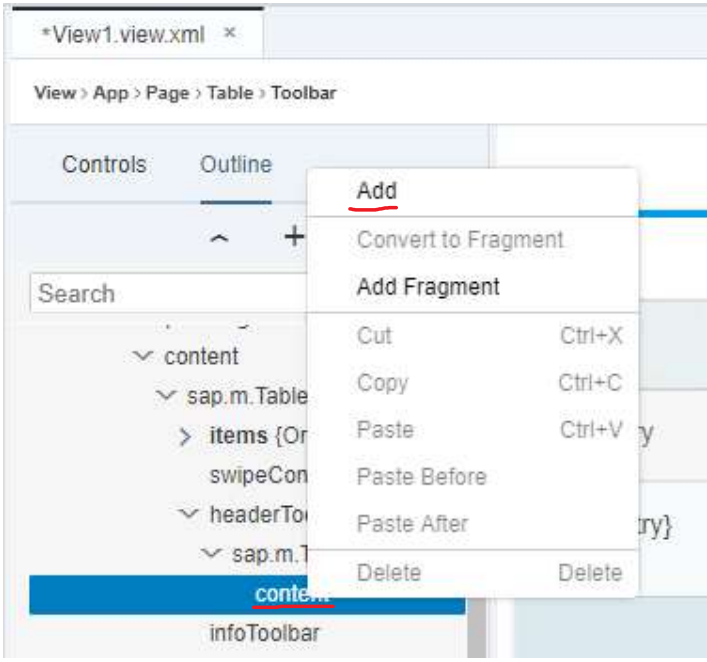
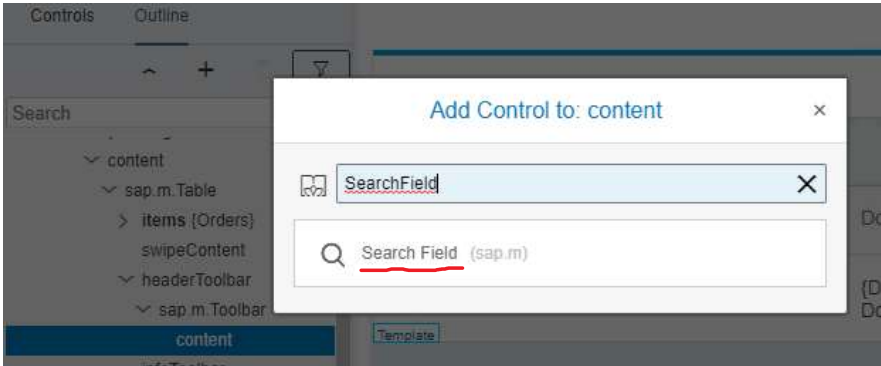

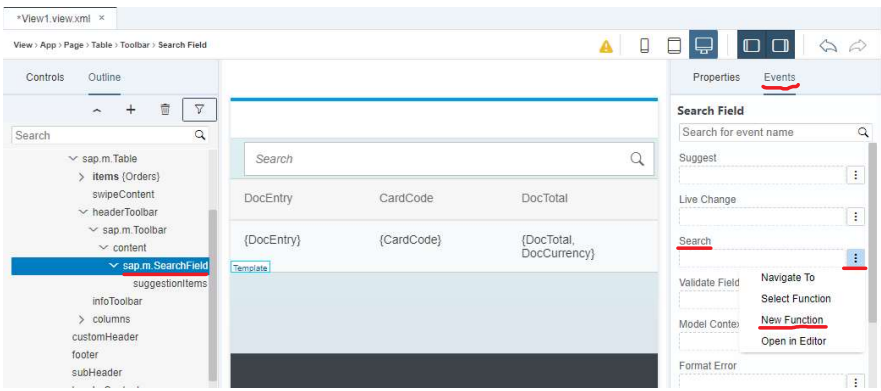
A new tab opens showing your application.

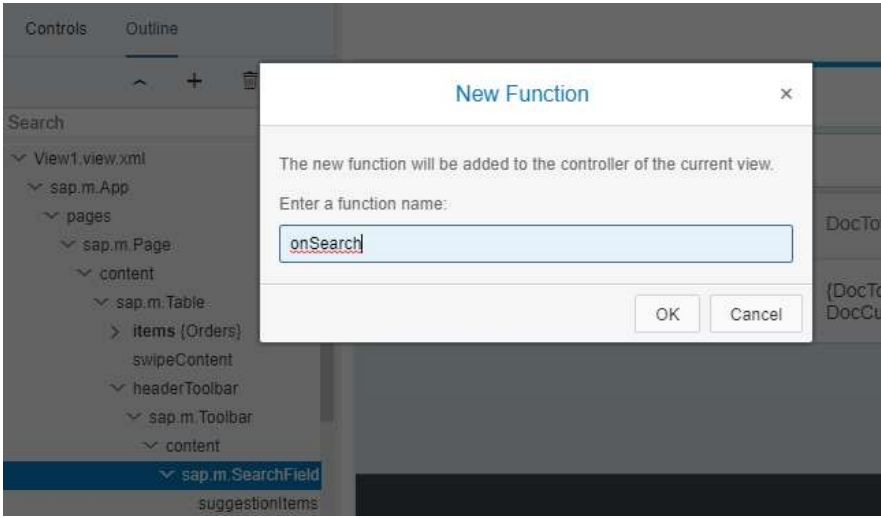
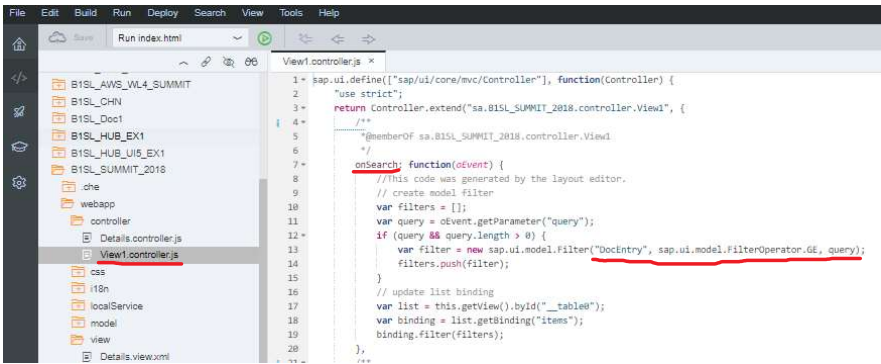
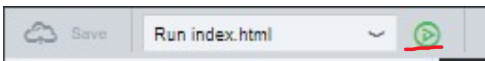
You can see the Table is automatically filled with the list of Orders available in the API Business Hub sandbox running SAP Business One.

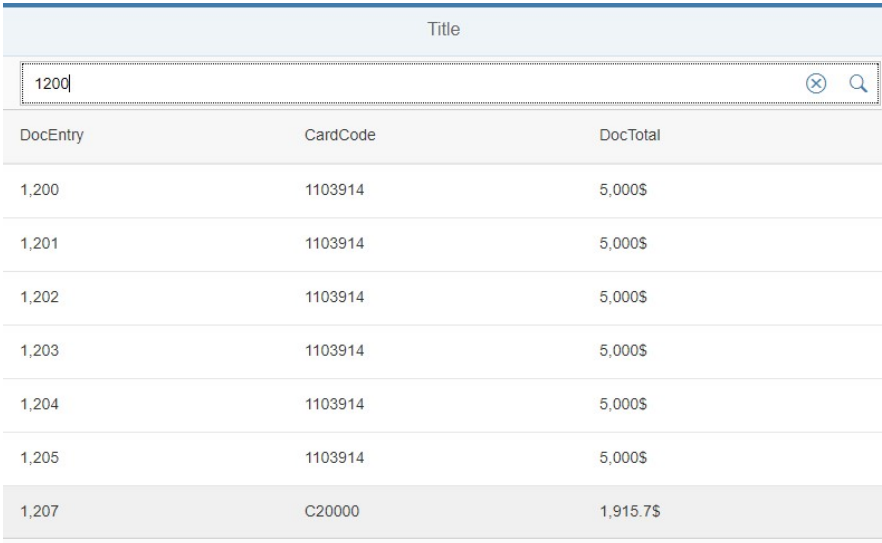
Title		
DocEntry	CardCode	DocTotal
2	C30000	2,976.88\$
3	C40000	12,535\$
4	C23900	7,473\$
5	C42000	18,391\$
6	C30000	8,795.31\$
7	C40000	10,573\$

Add a Search Field control to the sap.m.Table

Explanation	Screenshot
<p>Open the View.view.xml file with the Layout Editor.</p> <p>Go to the Outline tab and select sap.m.Table -> headerToolBar.</p> <p>Right click to get the context menus and select Add.</p>	 <p>The screenshot shows the SAP Web IDE interface. The Outline tab is active, displaying a tree view of the project structure. The file View1.view.xml is selected, and its content is expanded to show sap.m.App, pages, sap.m.Page, content, sap.m., and items. The headerToolBar control is highlighted in blue. A context menu is open over this control, with the Add option selected. Other options in the menu include Convert to Fragment, Add Fragment, Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), Paste Before, Paste After, and Delete (Delete).</p>
<p>Click on the Toolbar proposed control.</p>	 <p>The screenshot shows the 'Add Control to: headerToolBar' dialog box. The dialog has a search bar at the top and a list of proposed controls below. The Toolbar control is selected and highlighted with a red underline. Other controls in the list include Overflow Toolbar (sap.m).</p>

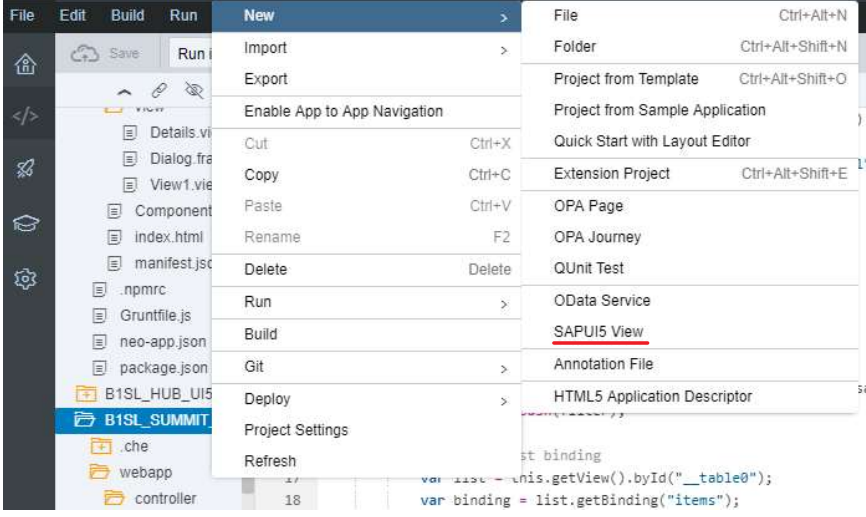
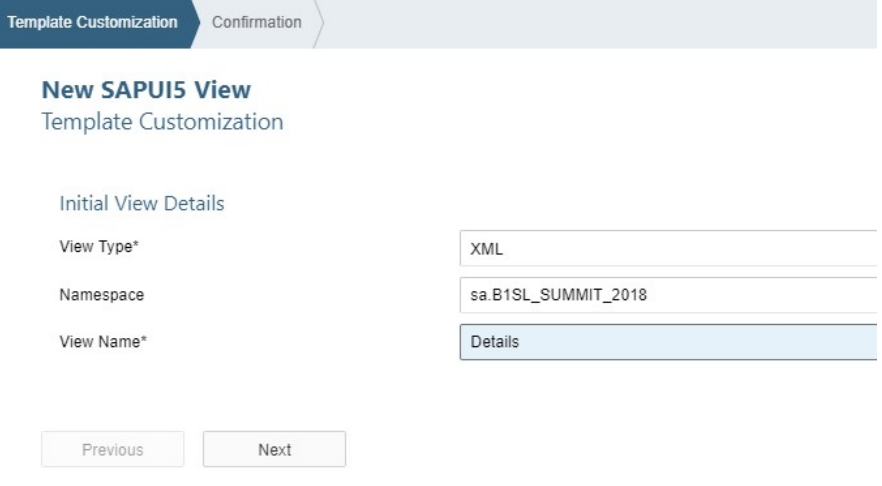
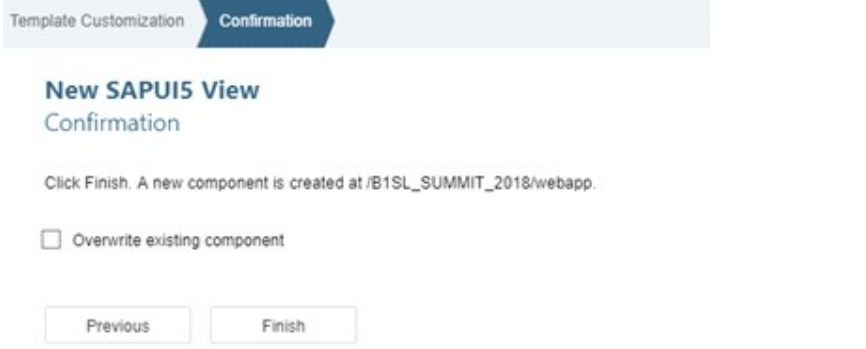
Explanation	Screenshot
<p>Go to the created sap.m.Toolbar -> content element and right click to select the Add menu.</p>	
<p>Enter SearchField to filter the controls.</p> <p>Click on the Search Field proposed control.</p>	
<p>Select the sap.m.SearchField control in the Outline tab.</p> <p>Select the Events tab in the right side of the screen.</p> <p>On the Search event click on the  button and select New Function option.</p>	

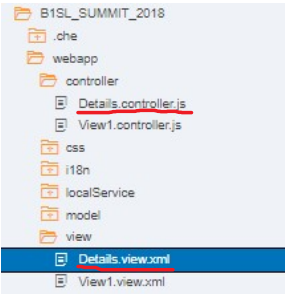
Explanation	Screenshot
<p>Enter onSearch as function name that will be called when the Search button will be pressed.</p> <p>Press OK.</p> <p>Press Save.</p> <p>A new function with that name will be created in the View.Controller.js.</p>	
<p>Open the View1.controller.js file.</p> <p>Copy the code from https://github.com/B1SA/B1_SCP_HandsOn/blob/master/snippets/View1.controller.js_onSearch.js into your onSearch method so it looks like here.</p> <p>This code filters the Orders with DocEntry greater or equal than the value entered in the SearchField.</p> <p>Press Save.</p>	
<p>Run your application.</p>	

Explanation	Screenshot																								
<p>Check that the SearchField filters the table with Orders having a DocEntry higher than the entered value.</p>	 <p>The screenshot shows a search interface with a search field containing the value '1200'. Below the search field is a table with the following data:</p> <table border="1"> <thead> <tr> <th>DocEntry</th> <th>CardCode</th> <th>DocTotal</th> </tr> </thead> <tbody> <tr> <td>1,200</td> <td>1103914</td> <td>5,000\$</td> </tr> <tr> <td>1,201</td> <td>1103914</td> <td>5,000\$</td> </tr> <tr> <td>1,202</td> <td>1103914</td> <td>5,000\$</td> </tr> <tr> <td>1,203</td> <td>1103914</td> <td>5,000\$</td> </tr> <tr> <td>1,204</td> <td>1103914</td> <td>5,000\$</td> </tr> <tr> <td>1,205</td> <td>1103914</td> <td>5,000\$</td> </tr> <tr> <td>1,207</td> <td>C20000</td> <td>1,915.7\$</td> </tr> </tbody> </table>	DocEntry	CardCode	DocTotal	1,200	1103914	5,000\$	1,201	1103914	5,000\$	1,202	1103914	5,000\$	1,203	1103914	5,000\$	1,204	1103914	5,000\$	1,205	1103914	5,000\$	1,207	C20000	1,915.7\$
DocEntry	CardCode	DocTotal																							
1,200	1103914	5,000\$																							
1,201	1103914	5,000\$																							
1,202	1103914	5,000\$																							
1,203	1103914	5,000\$																							
1,204	1103914	5,000\$																							
1,205	1103914	5,000\$																							
1,207	C20000	1,915.7\$																							

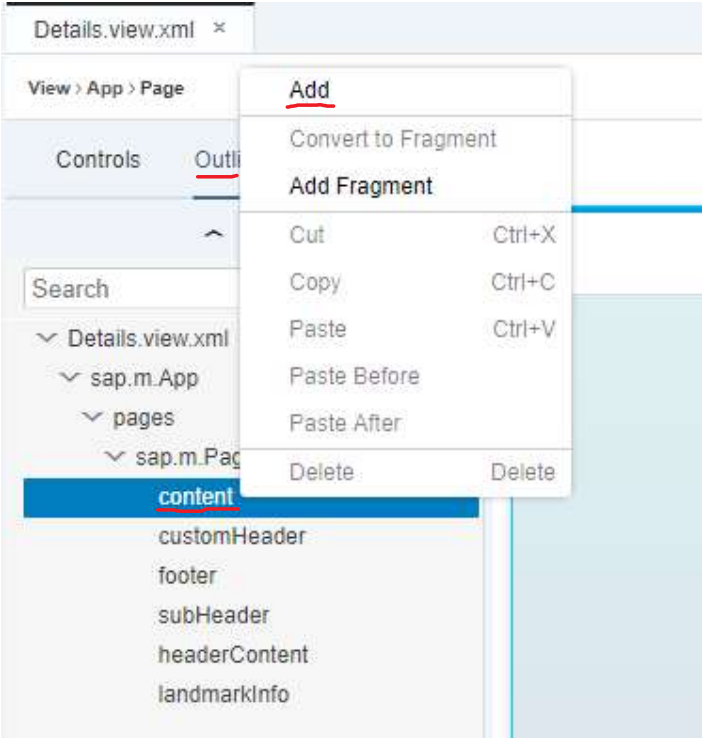
iv. Add a second view called Details

Create a Details view.

Explanation	Screenshot
<p>Right click on your application and select New -> SAPUI5 View.</p>	 <p>The screenshot shows the SAPUI5 IDE interface. A right-click context menu is open over the project folder 'B1SL_SUMMIT'. The 'New' option is selected, and a sub-menu is displayed with 'SAPUI5 View' highlighted. Other options in the sub-menu include 'Folder', 'Project from Template', 'Project from Sample Application', 'Quick Start with Layout Editor', 'Extension Project', 'OPA Page', 'OPA Journey', 'QUnit Test', 'OData Service', 'Annotation File', and 'HTML5 Application Descriptor'. The background shows a file explorer with various files like 'Details.view', 'Dialog.fragment', 'View1.view', 'index.html', 'manifest.json', '.npmrc', 'Gruntfile.js', 'neo-app.json', 'package.json', 'B1SL_HUB_UI5', 'B1SL_SUMMIT', '.che', 'webapp', and 'controller'.</p>
<p>Keep the default values for View Type and Namespace. Enter the name of the new View: Details.</p> <p>Press Next.</p>	 <p>The screenshot shows the 'New SAPUI5 View' dialog in the 'Template Customization' step. The title is 'New SAPUI5 View' and the subtitle is 'Template Customization'. Under 'Initial View Details', there are three input fields: 'View Type*' with 'XML' selected, 'Namespace' with 'sa.B1SL_SUMMIT_2018' entered, and 'View Name*' with 'Details' entered. At the bottom, there are 'Previous' and 'Next' buttons.</p>
<p>Press Finish to confirm the creation of the Details view.</p>	 <p>The screenshot shows the 'New SAPUI5 View' dialog in the 'Confirmation' step. The title is 'New SAPUI5 View' and the subtitle is 'Confirmation'. The text says 'Click Finish. A new component is created at /B1SL_SUMMIT_2018/webapp.' There is an unchecked checkbox labeled 'Overwrite existing component'. At the bottom, there are 'Previous' and 'Finish' buttons.</p>

Explanation	Screenshot
<p>Two files are created:</p> <ul style="list-style-type: none"> - Details.view.xml - Details.controller.js. 	

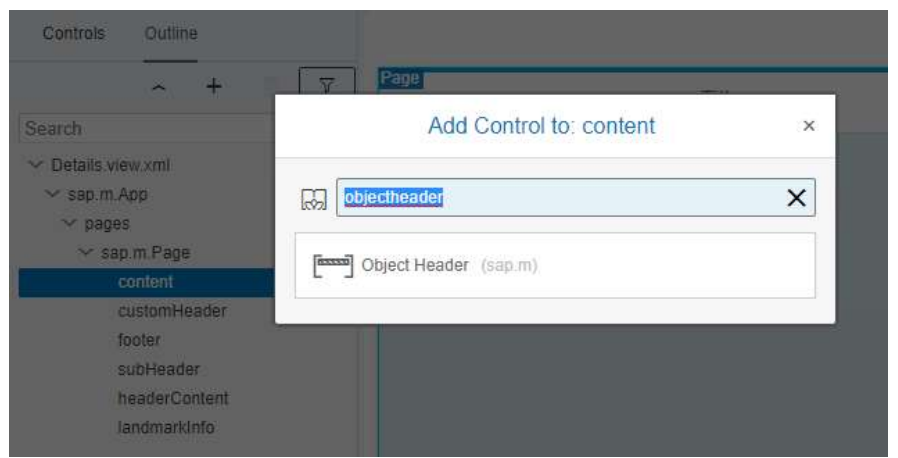
Add an Object Header control.

Explanation	Screenshot
<p>Open the Details.view.xml file with the Layout editor.</p> <p>Go to the Outline tab.</p> <p>Right click on Details.view.xml -> sap.m.App -> sap.m.Page -> content and select Add.</p>	

Explanation	Screenshot
-------------	------------

Enter **objectheader** to search for the Object Header control.

Click on the **Object Header** proposed control.

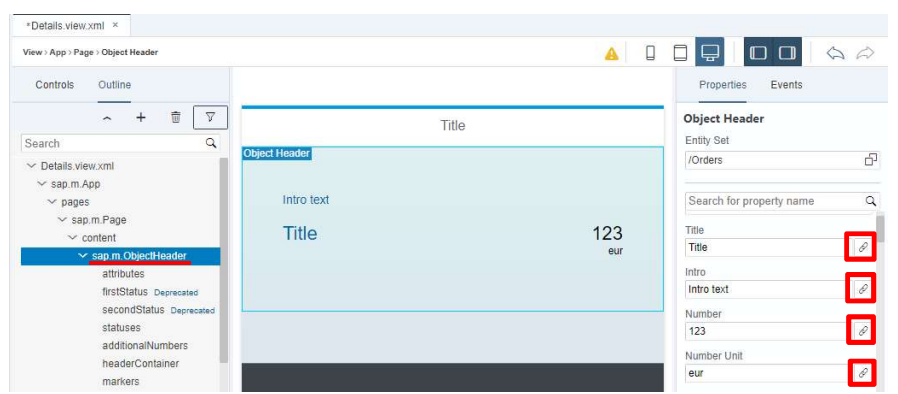


Select the created **sap.m.ObjectHeader** element in the Outline tree.

Go to the **Properties** tab in the right side of the screen.

Click one by one on the **Data Binding** icon for the properties:

- **Title**
- **Intro**
- **Number**
- **Number Unit**

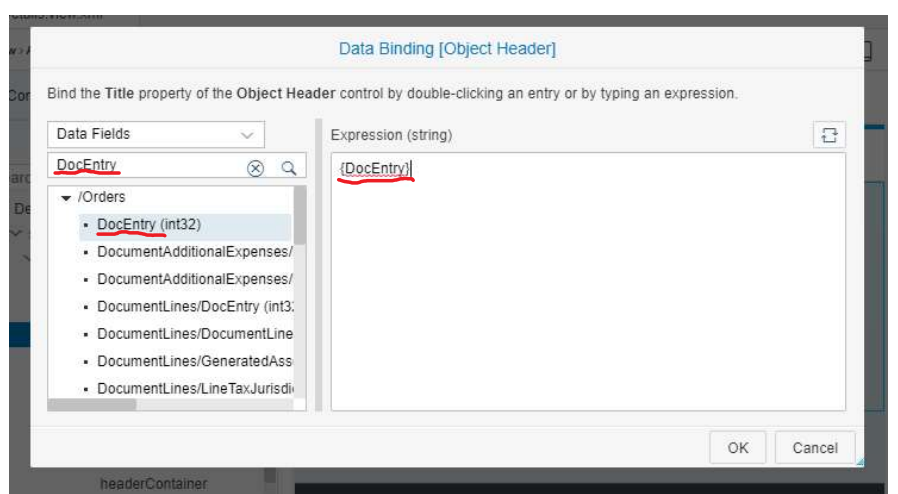


A new dialog will be presented for each one of the properties.

Search for the Orders Data Field to be assigned to each property:

- Title -> CardCode
- Intro -> DocEntry
- Number -> DocTotal
- Number Unit -> DocCurrency

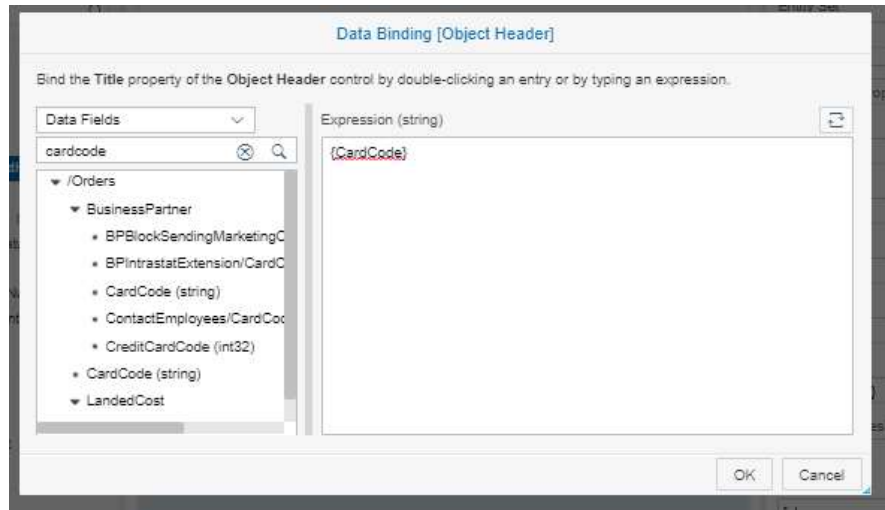
and double click on the proposed list to get it into the Expression string.



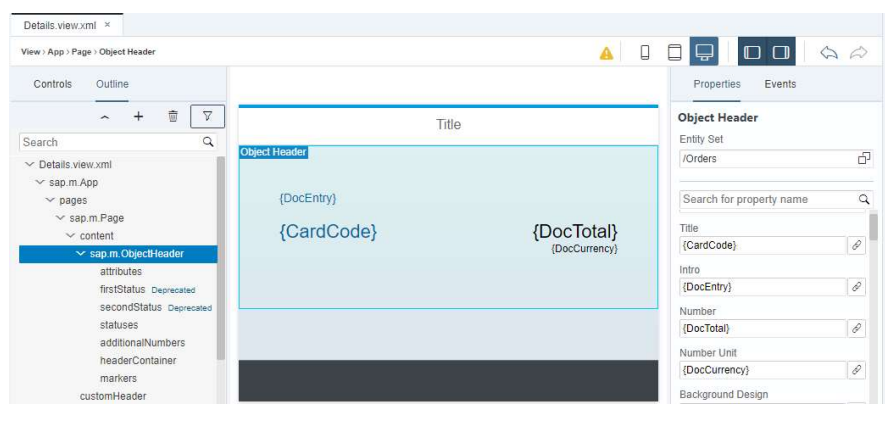
Explanation	Screenshot
-------------	------------

Note: **Do not select BusinessPartners/CardCode** for the CardCode property.

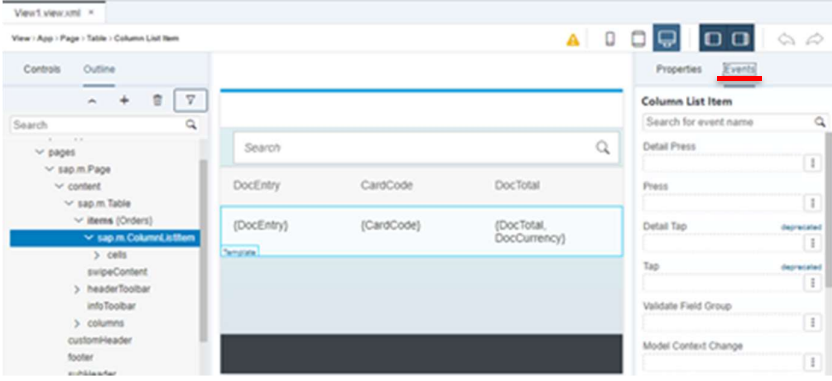

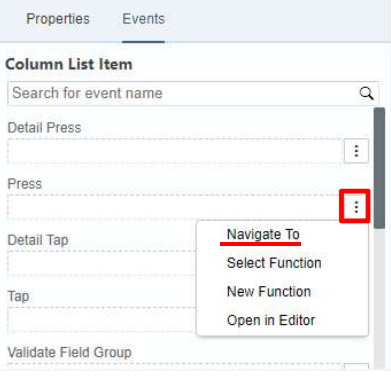
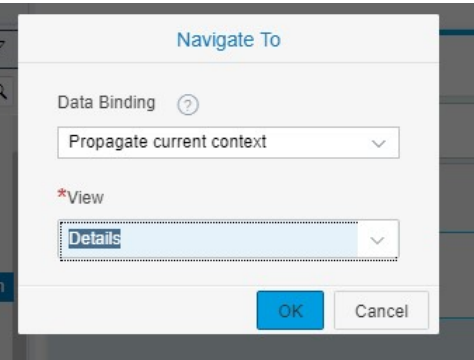
Press **Save** button.

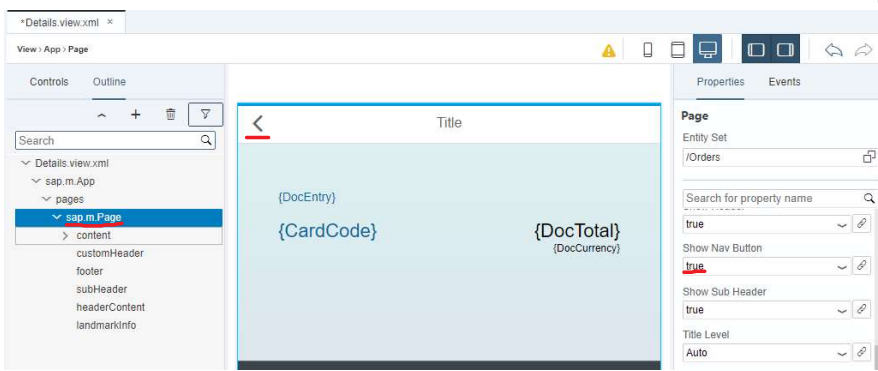

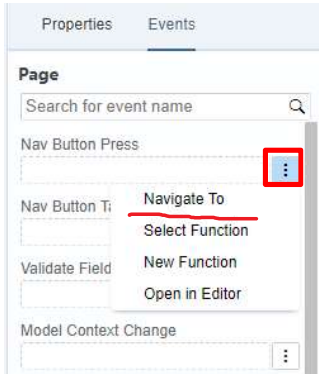
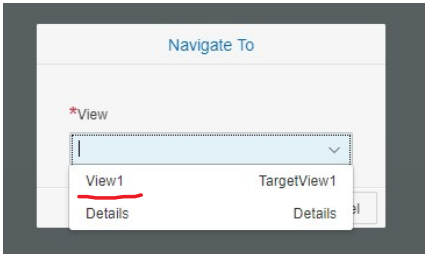
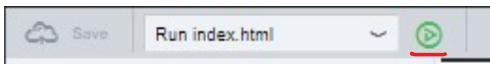


The sap.m.ObjectHeader should look like this at this point.



v. Define navigation between View1 and Details.

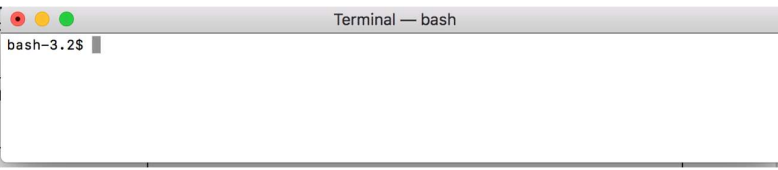

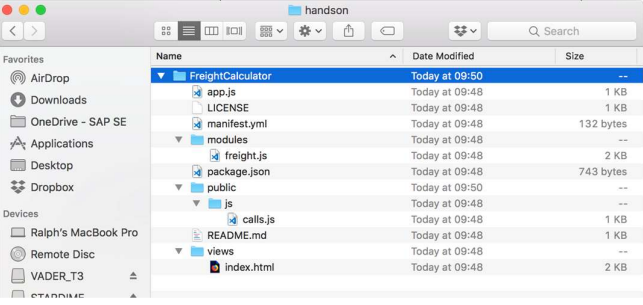
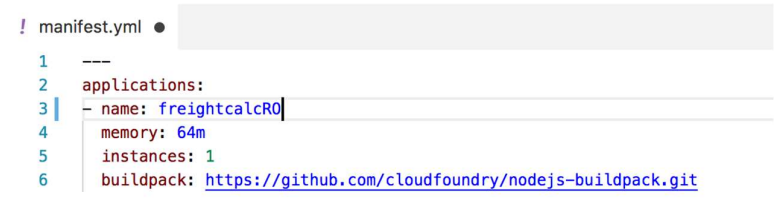
Explanation	Screenshot
<p>From our main View1 table we want to navigate to Details view when the user clicks on a row.</p> <p>Open View1.view.xml file with Layout Editor.</p> <p>Select Outline tab View1.view.xml -> sap.m.App -> sap.m.Page -> content -> sap.m.Table -> items -> sap.m.ColumnsListItem.</p> <p>Select the Events tab.</p>	
<p>Click on the  icon.</p> <p>Select Navigate To menu.</p>	
<p>Select Propagate current context on the Data Binding combo box.</p> <p>Select Details on the View combo box.</p> <p>Press OK.</p> <p>Press Save to save the View1.view.xml changes.</p> <p>The selected Order data from the View1 will be bound to the Details view.</p>	

Explanation	Screenshot
<p>From our Details view we want to navigate back to our main View1 when the user press on the Navigate back button.</p> <p>Open the Details.view.xml file with the Layout Editor.</p> <p>Go to the Outline tab and select sap.m.Page.</p> <p>On the Properties tab scroll down to see the Show Nav Button property and change its value to true.</p>	
<p>Go to the Events tab.</p> <p>On the Nav Button Press event click on the icon .</p> <p>Select the option Navigate To.</p>	
<p>Select View1 on the combo box.</p> <p>We will navigate back to View one when the Nav back button will be pressed.</p> <p>Click on the Save button.</p>	
<p>Test your app to see navigation is working fine between View1 and Details view.</p>	

Congratulations! You have created your first SAP UI5 application connecting to SAP API Business Hub

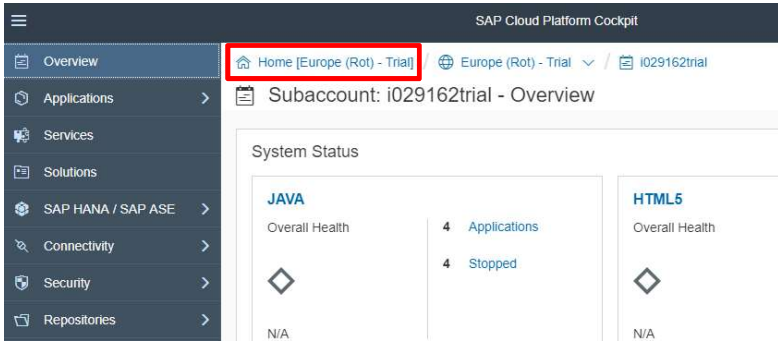
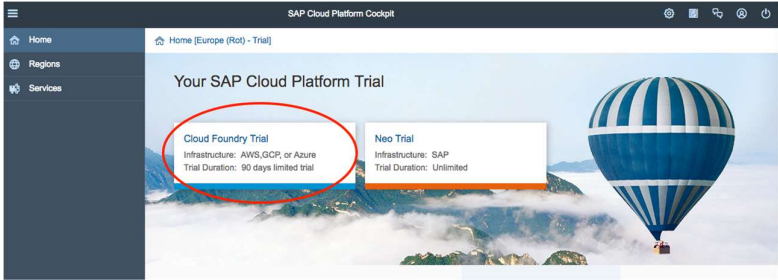
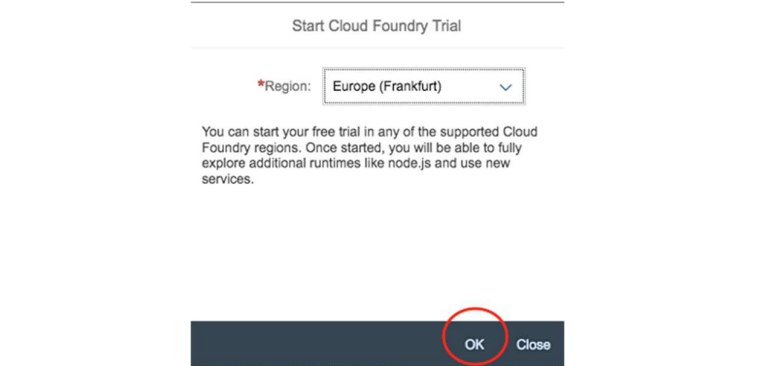
STEP 2: CREATE A NODEJS APP

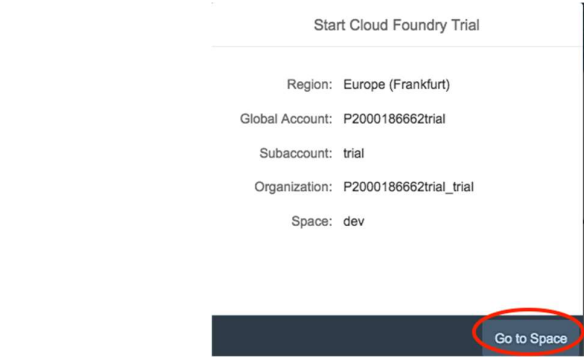
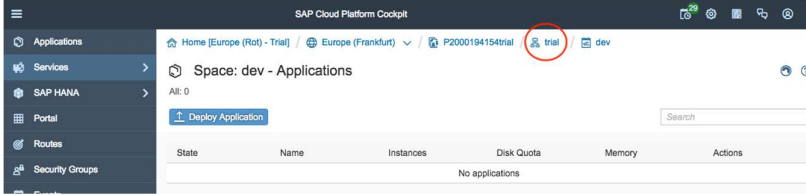
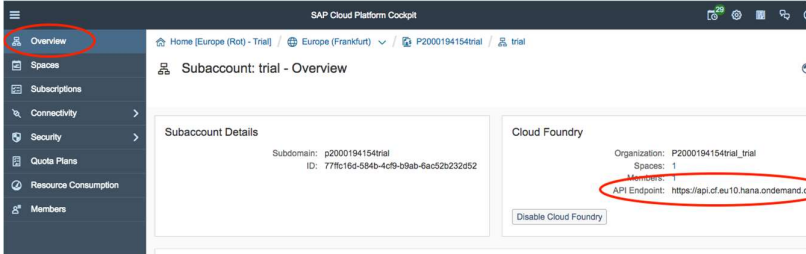
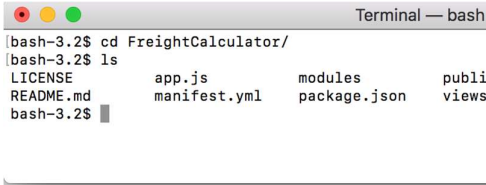

In this step we are going to implement the backend of our application. It will contain a simple business logic to calculate freight costs from different providers and consume 3rd Party services. The application is written in NodeJS and the source code is available on GitHub.

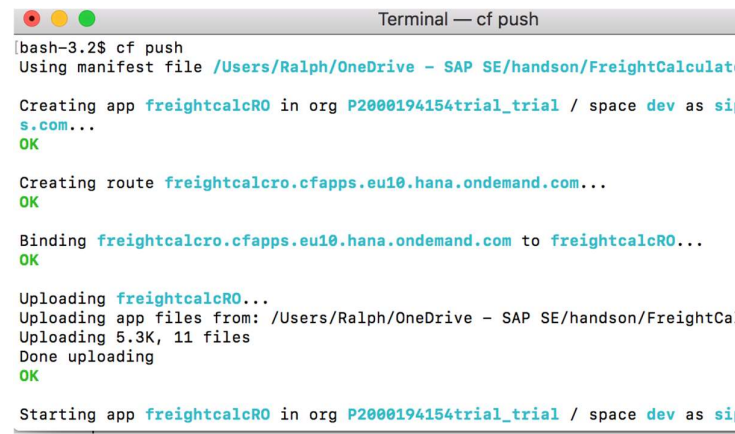
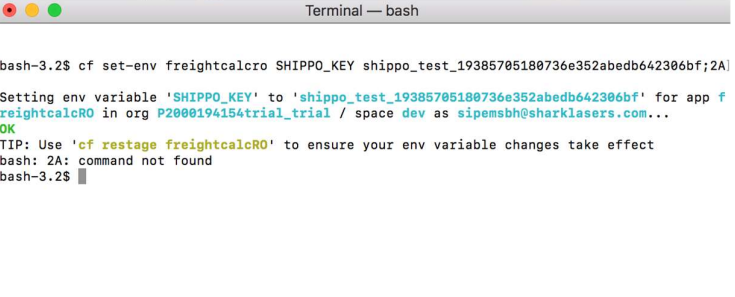
Explanation	Screenshot
<p>Once git is installed (according to the pre requisites), open your system terminal (cmd, bash..)</p> <p>Navigate to a specific folder where you will download a sample application.</p> <p>Pay attention what folder is it, we will access it later</p>	
<p>Execute the following command to clone our solution backend repository:</p> <pre>\$ git clone https://github.com/B1SA/FreightCalculator.git</pre>	
<p>You can see the app code on your file explorer:</p>	
<p>Edit the File manifest.yml and set a unique name for your application. E.eg FreightCalc<Your Initials></p>	

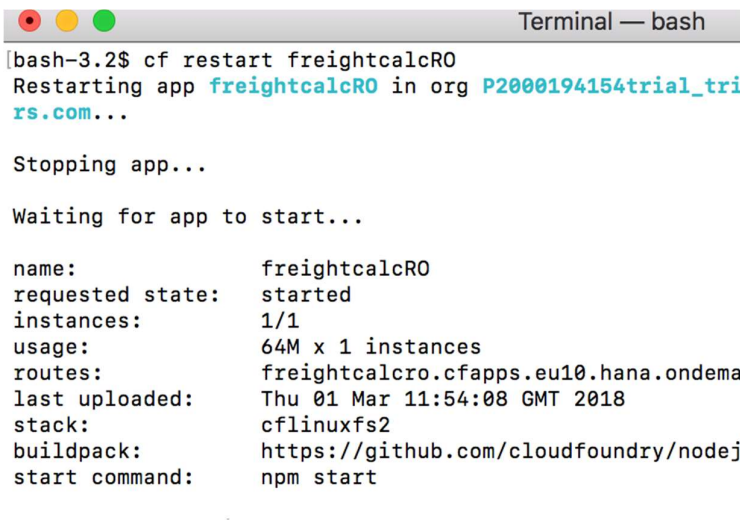
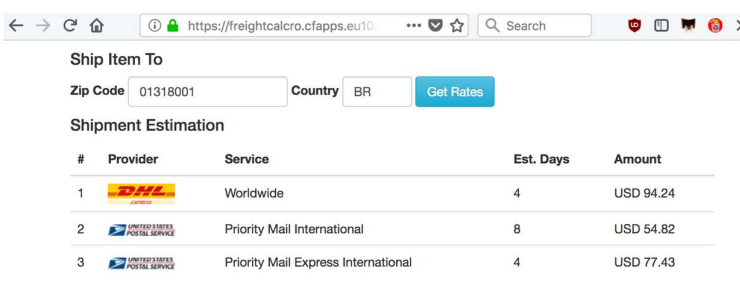
STEP 3: DEPLOY THE NODEJS APP INTO SAP CLOUD FOUNDRY

In this step, we are going to deploy our Freight Calculator app to SAP Cloud Platform Cloud Foundry.

Explanation	Screenshot
<p>First, we need to activate the SAP Cloud Platform Cloud Foundry Environment.</p> <p>On the SAP Cloud Platform Dashboard click on the HOME option</p>	
<p>First, we need to activate the SAP Cloud Platform Cloud Foundry Environment.</p> <p>From the SAP CP Home Screen, click on Cloud Foundry Trial</p>	
<p>Select the Trial Region that most suits you. And Click on OK</p>	

Explanation	Screenshot
<p>This will initialize your Cloud Foundry Trial and create a DEV space (where the solutions will be deployed). Go Ahead and access your space.</p>	
<p>Your space is created and ready to deploy your app. We now need your environment endpoint to be able to push our app.</p> <p>Click on the trial link</p>	
<p>And then on Overview option in the menu</p> <p>Select and copy your API Endpoint. E.g.</p> <p><code>https://api.cf.eu10.hana.ondemand.com</code></p>	
<p>With the CLI installed (according to the pre requisites), open your system terminal and navigate to the folder of the backend app cloned on STEP 2 of this guide</p>	
<p>From that folder, login to Cloud foundry using the command</p> <pre>cf login -a <API ENDPOINT></pre> <p>e.g.</p> <pre>\$ cf login -a api.cf.eu10.hana.ondemand.com</pre> <p>When prompted provide your SAP Cloud Platform email and password</p>	

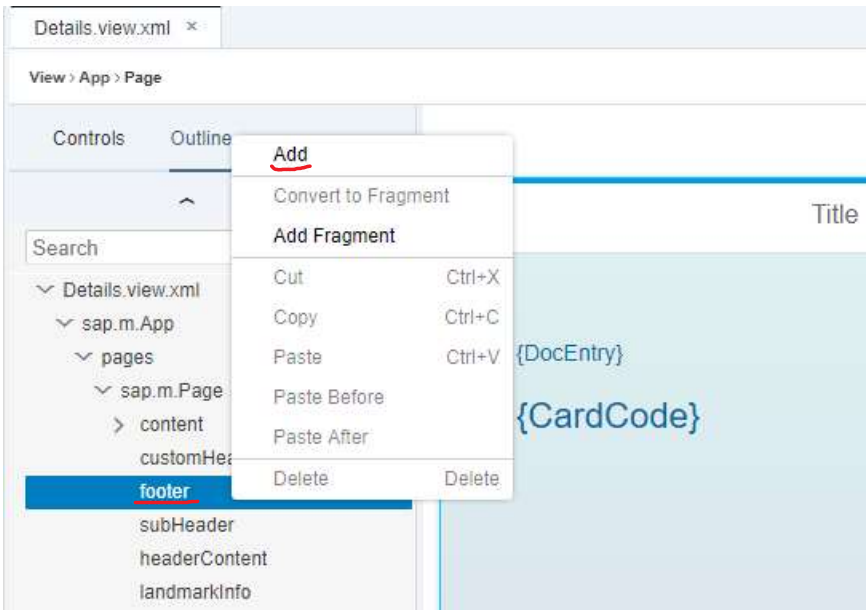
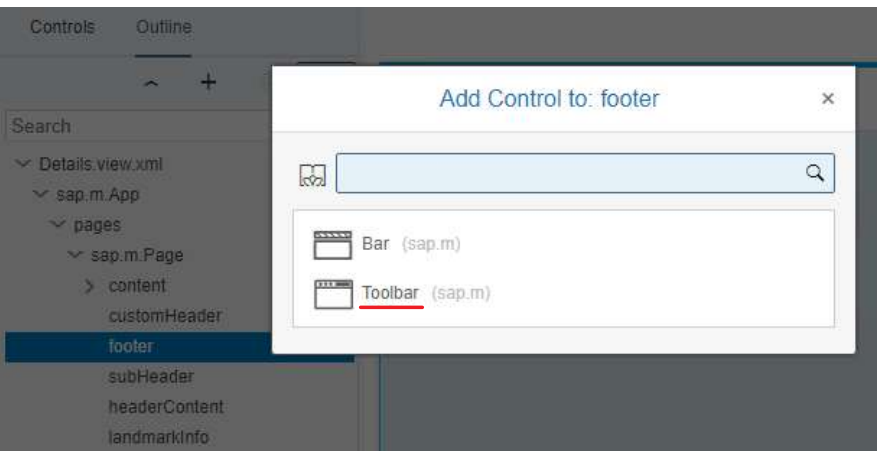
Explanation	Screenshot												
<p>Now all you have to do is push your app to the SAP Cloud Platform Cloud Foundry by using the command:</p> <pre>cf push</pre>													
<p>This process will read the manifest.yml to name your application and also upload and deploy all the artefacts in a container in the Cloud Foundry Environment. Once the Process finishes, you can see your app URL:</p>	<pre>requested state: started instances: 1/1 usage: 64M x 1 instances urls: freightcalcro.cfapps.eu10.hana.ondemand.com last uploaded: Thu Mar 1 11:54:08 UTC 2018 stack: cflinuxfs2 buildpack: https://github.com/cloudfoundry/nodejs-buildpack.git</pre> <table border="1"> <thead> <tr> <th>#0</th> <th>state</th> <th>since</th> <th>cpu</th> <th>memory</th> <th>disk</th> </tr> </thead> <tbody> <tr> <td></td> <td>running</td> <td>2018-03-01 11:54:39 AM</td> <td>0.0%</td> <td>44.7M of 64M</td> <td>55.3M of 1G</td> </tr> </tbody> </table> <p>bash-3.2\$</p>	#0	state	since	cpu	memory	disk		running	2018-03-01 11:54:39 AM	0.0%	44.7M of 64M	55.3M of 1G
#0	state	since	cpu	memory	disk								
	running	2018-03-01 11:54:39 AM	0.0%	44.7M of 64M	55.3M of 1G								
<p>This application makes use of a 3rd Party service called Shippo to calculate shipping costs. In order to consume their services, we need an API KEY.</p> <p>The Instructors of this hands-on session will provide a set of keys you can use in the next step. However, you can also get your own FREE test key on their website https://qoshippo.com/</p> <p>Best practices of cloud development (see https://12factor.net/config) suggests that any kind of configuration (such as keys) should not be part of the codebase but set as environment variables. And that is easily done with Cloud Foundry</p>													
<p>Back to the terminal, set the Environment Variable SHIPPO_KEY to a valid API Key with the following command</p> <pre>cf set-env <appname> <variable name> <variable value e.g. cf set-env freightcalcRO SHIPPO_KEY shippo_test_1234</pre>													

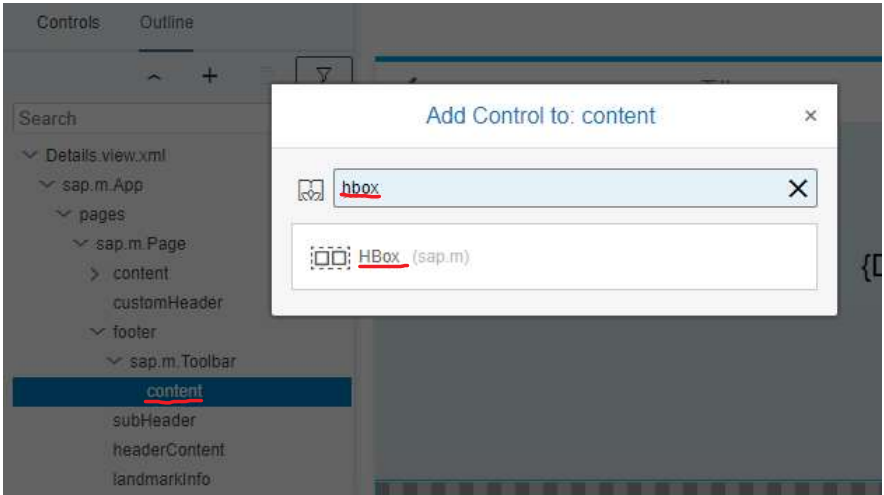
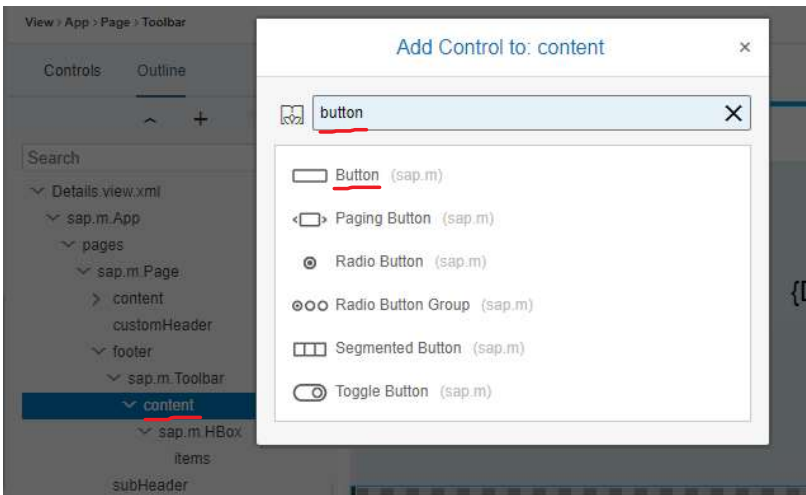
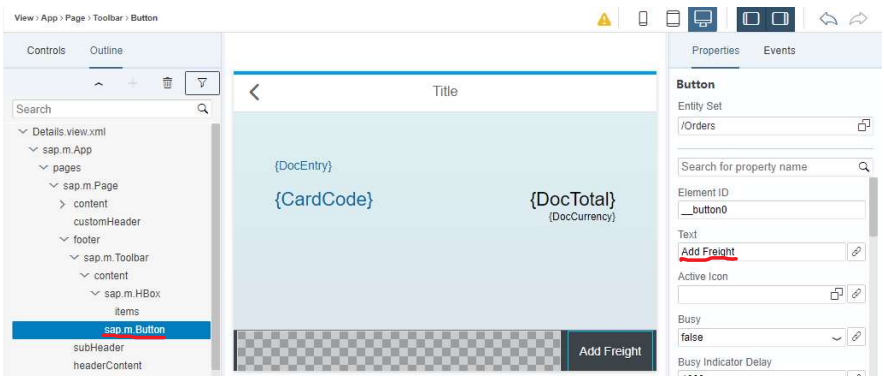
Explanation	Screenshot																				
<p>Up next, restart your app</p> <pre>cf restart <appname></pre>	 <pre>Terminal — bash bash-3.2\$ cf restart freightcalcRO Restarting app freightcalcRO in org P2000194154trial_trirs.com... Stopping app... Waiting for app to start... name: freightcalcRO requested state: started instances: 1/1 usage: 64M x 1 instances routes: freightcalcro.cfapps.eu10.hana.ondema last uploaded: Thu 01 Mar 11:54:08 GMT 2018 stack: cflinuxfs2 buildpack: https://github.com/cloudfoundry/nodej start command: npm start</pre>																				
<p>You can test the app with a test page on the app URL</p>	 <p>Ship Item To</p> <p>Zip Code <input type="text" value="01318001"/> Country <input type="text" value="BR"/> <input type="button" value="Get Rates"/></p> <p>Shipment Estimation</p> <table border="1"> <thead> <tr> <th>#</th> <th>Provider</th> <th>Service</th> <th>Est. Days</th> <th>Amount</th> </tr> </thead> <tbody> <tr> <td>1</td> <td></td> <td>Worldwide</td> <td>4</td> <td>USD 94.24</td> </tr> <tr> <td>2</td> <td></td> <td>Priority Mail International</td> <td>8</td> <td>USD 54.82</td> </tr> <tr> <td>3</td> <td></td> <td>Priority Mail Express International</td> <td>4</td> <td>USD 77.43</td> </tr> </tbody> </table>	#	Provider	Service	Est. Days	Amount	1		Worldwide	4	USD 94.24	2		Priority Mail International	8	USD 54.82	3		Priority Mail Express International	4	USD 77.43
#	Provider	Service	Est. Days	Amount																	
1		Worldwide	4	USD 94.24																	
2		Priority Mail International	8	USD 54.82																	
3		Priority Mail Express International	4	USD 77.43																	

Congratulations! You have implemented and deployed your first Cloud Foundry application on SAP Cloud Platform!


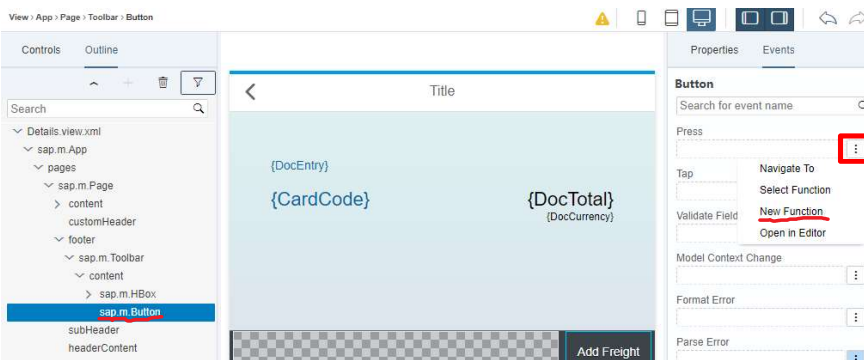
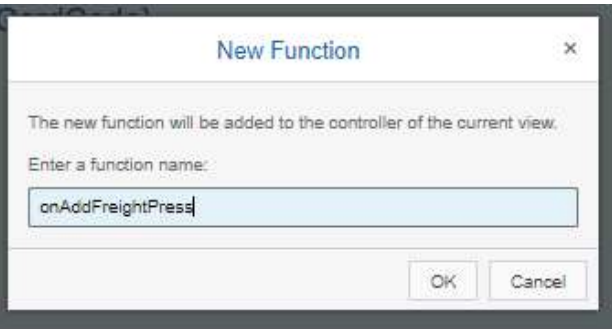


STEP 4: CONSUME THE NODEJS APP FROM THE SAP FIORI APP

- i. Add a Button “Add Freight” to the Details view.

Explanation	Screenshot
<p>Open the Details.view.xml file with the Layout Editor.</p> <p>On the Outline tab right click on the sap.m.App -> pages -> sap.m.Page -> footer element and select Add.</p>	 <p>The screenshot shows the SAP Fiori Layout Editor interface. The 'Outline' tab is active, displaying a tree view of the XML structure. The 'footer' element is selected and highlighted in blue. A context menu is open over the 'footer' element, with the 'Add' option highlighted in red. Other options in the menu include 'Convert to Fragment', 'Add Fragment', 'Cut', 'Copy', 'Paste', 'Paste Before', 'Paste After', and 'Delete'. The main editor area shows a snippet of XML code with placeholders like {DocEntry} and {CardCode}.</p>
<p>Select the Toolbar control proposed.</p>	 <p>The screenshot shows the SAP Fiori Layout Editor interface. The 'Outline' tab is active, and the 'footer' element is selected. A dialog box titled 'Add Control to: footer' is open, displaying a search bar and a list of proposed controls. The 'Toolbar (sap.m)' control is highlighted in red in the list.</p>

Explanation	Screenshot
<p>On the Outline tab right click on sap.m.Toolbar -> content and select Add.</p> <p>Enter hbox in the Add Control to: content new dialog.</p> <p>Select the proposed HBox control.</p>	
<p>On the Outline tab right click on sap.m.Toolbar -> content and select Add.</p> <p>Enter button in the Add Control to: content dialog.</p> <p>Select the Button control.</p>	
<p>On the Outline tab select the sap.m.Button.</p> <p>On the Properties tab change the Text property by Add Freight.</p>	

ii. Implement the Button business logic calling the NodeJS server side and Service Layer.

Explanation	Screenshot
<p>Select Outline tab, sap.m.Button.</p> <p>Open Events tab.</p> <p>Click on the Press event  icon and select New Function.</p>	
<p>Enter onAddFreightPress as the function name.</p> <p>This function will be called when the event Press is fired on the AddFreight button.</p>	
<p>Open the Details.controller.js file and scroll down to the bottom of the file.</p> <p>A new empty function called onAddFreightPress has been automatically created.</p>	 <pre> 1- /* 2- * @memberOf sa.B15L_SUMMIT_2018.controller.Details 3- */ 4- onAddFreightPress: function(oEvent) { 5- //This code was generated by the layout editor. 6- } 7- */ </pre>
<p>Before implementing this function lets add some definitions to the Details.controller.js file.</p> <p>Open the Details.controller.js file.</p> <p>Remove the code at the beginning of the file until "use strict"; as shown here in the first screen.</p> <p>Replace that code from by the one shared at https://github.com/B15A/B1_SCP_HandsOn/blob/master/snippets/Details.controller.js_definitions.js as</p>	 <pre> 1- sap.ui.define(["sap/ui/core/mvc/Controller"], function(Controller) { 2- "use strict"; 3- 4- sap.ui.define(["sap/ui/core/mvc/Controller", 5- "sap/m/MessageToast", 6- "sap/ui/core/Fragment", 7- "sap/ui/model/Filter", 8- "sap/ui/model/json/JSONModel", 9- "jquery.sap.global" 10-], 11- function(Controller, MessageToast, Fragment, Filter, JSONModel, jQuery) { 12- "use strict"; </pre>

Explanation	Screenshot
-------------	------------

shown in the second screen.

Let now implement the function **onAddFreightPress**.

When the user clicks on the AddFreight button we call the server side NodeJS to get freight options.

Replace the URL to the freight calculator by **your backend application URL** (code marked in red in the screen capture).

In case of success we call the function `openFreightDialog`.

In case of error we simply show a `MessageToast`.

You can get this code from https://github.com/B1SA/B1_SCP_HandsOn/blob/master/snippets/Details.controller.js_onAddFreight.js.

PS: The code contains more functions than this one, please copy the full set of functions. We will explain each one of the functions in the coming steps.

```

onAddFreightPress: function(oEvent) {
    //This code was generated by the layout editor.

    // Get Data from ODataModel V4 /Orders
    var body = {
        "to": {
            "zip": this.getView().getBindingContext().getProperty("AddressExtension/ShipToZipCode"),
            "country": this.getView().getBindingContext().getProperty("AddressExtension/ShipToCountry")
        }
    };
    // open Freight view
    // from Freight view selection we will get back to Object view
    var oThis = this;
    $.ajax({
        url: "https://freightcalc.cfapps.eu10.hana.ondemand.com/Rates",
        type: "POST",
        data: JSON.stringify(body),
        contentType: "application/json",
        success: function(data) {
            oThis.openFreightDialog(data);
        },
        error: function(e) {
            MessageToast.show("POST Freight error: " + e);
        }
    });
},

```

On the `Details.controller.js` scroll and search for the function **openFreightDialog**. This function opens a new Dialog showing the freight options returned by the server side.

This function will use an xml fragment to design the freight options dialog.

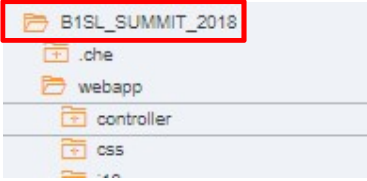
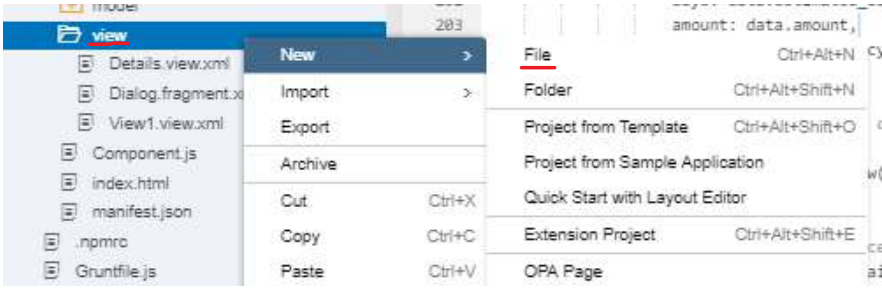
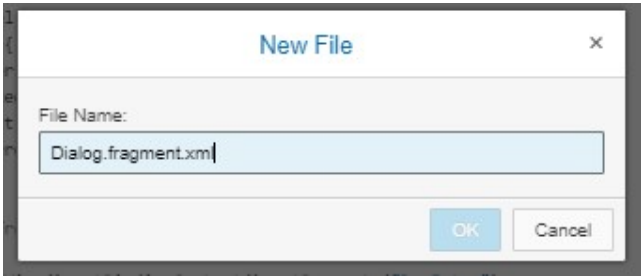
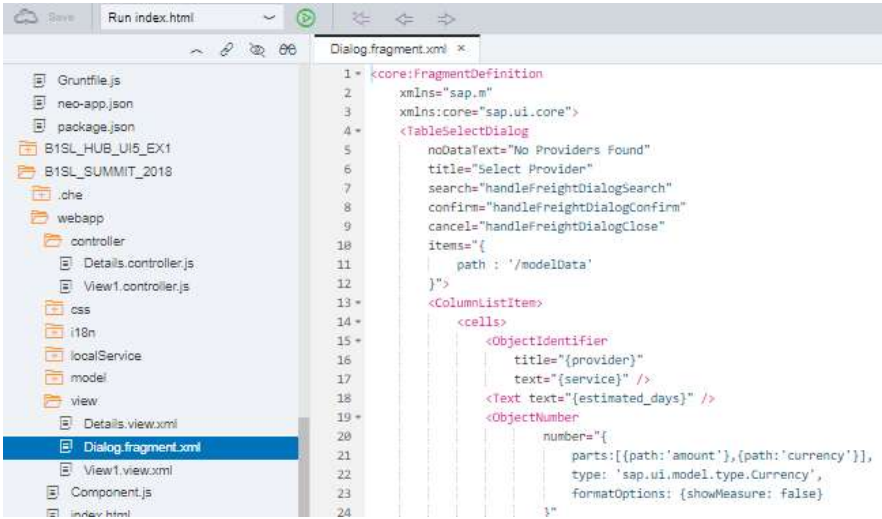
This function will use an xml fragment to design the freight options dialog.

Edit this function and **change** the pointer to the **xmlfragment** to match the name of your namespace and SAPUI5 application (you can get this information from the beginning of your `Details.controller.js` file. In my case it is

```

Details.controller.js x Details.view.xml x
159 // open Freight Dialog
160 openFreightDialog: function(data) {
161     var detailsView = this.getView();
162
163     //Create a model and bind the table rows to this model
164     var oModel = new sap.ui.model.json.JSONModel();
165     // created a JSON model
166     oModel.setData({
167         modelData: data
168     });
169
170     // create dialog lazily
171     if (!this._oDialog) {
172         // create dialog via fragment factory
173         this._oDialog = sap.ui.xmlfragment("sa.B1SL_SUMMIT_2018.view.Dialog", this);
174         this._oDialog.setModel(oModel);
175     }
176     // connect dialog to view (models, lifecycle)
177     detailsView.addDependent(this._oDialog);
178
179     // toggle compact style
180     jQuery.sap.syncStyleClass("sapUiSizeCompact", detailsView, this._oDialog);
181     this._oDialog.open();
182
183 },

```

Explanation	Screenshot
<p>“sa.B1SL_SUMMIT_2018.view.Dialog”.</p>	
<p>Let's add the Fragment required to show the Dialog.</p> <p>Right click on view folder and select New->File menu.</p>	
<p>Give the name Dialog.fragment.xml to the new file.</p> <p>Press OK.</p>	
<p>Download the Dialog.fragment.xml file from https://github.com/B1SA/B1_SCP_HandsOn/blob/master/snippets/Dialog.fragment.xml.</p> <p>Copy the code inside your Dialog.fragment.xml file.</p> <p>You can notice in that file contains the controls of the Dialog. Also, that file contains 3 callback functions will be called for the events:</p> <ul style="list-style-type: none"> - search - confirm - cancel <p>We will implement those functions in our Details.controller.js file.</p>	

Explanation	Screenshot
<p>Open the Details.controller.js file.</p> <p>The handleFreightDialogSearch (code copied already earlier) is called when the user enters data to filter the list of freight options. The filter will be based on the provider property.</p> <p>The handleFreightDialogConfirm function is called when the user selects one of the freight options proposed by the Dialog. This function calls the function UpdateSO to update the Order via Service Layer API.</p> <p>The handleFreightDialogClose function is called when the user closes the Dialog without selecting a freight option. A MessageToast will show to the user the message "No Provider selected."</p>	<pre>handleFreightDialogSearch: function(oEvent) { var sValue = oEvent.getParameter("value"); var oFilter = new Filter("provider", sap.ui.model.FilterOperator.Contains, sValue); var oBinding = oEvent.getSource().getBinding("items"); oBinding.filter([oFilter]); }, handleFreightDialogConfirm: function(oEvent) { // Get SelectedItem details var oSelectedItem = oEvent.getParameter("selectedItem"); var ctx = oSelectedItem.getBindingContext(); var data = ctx.getModel().getProperty(ctx.getPath()); var providerDetails = { name: data.provider, days: data.estimated_days, amount: data.amount, currency: data.currency }; // Get Order details from current view binding context var orderDetails = { DocEntry: this.getView().getBindingContext().getProperty("DocEntry") }; // Update Order via Service Layer this.updateSO(providerDetails, orderDetails); console.log("You have chosen " + providerDetails.name + ", " + providerDetails.days + " Days, " + providerDetails.amount + providerDetails.currency + "."); }, handleFreightDialogClose: function(oEvent) { MessageToast.show("No Provider selected."); },</pre>

Explanation

The **updateSO** function calls SAP Business One Service Layer via the SAP API Business Hub sandbox.

We do an ajax call to the destination `/apihub_sandbox/` automatically created when we added the Data Source pointing to the SAP API Business Hub. Then we specify the DocEntry of the Order to be updated with a PATCH request.

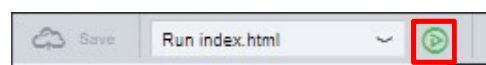
In case of success we show a MessageToast message "SL success".

In case of error we show a MessageToast message "SL error".

Screenshot

```
Details.controller.js x Details.view.xml x
218     },
219
220     // Update Order via Service Layer
221     updateSO: function(providerDetails, orderDetails) {
222
223         var oThis = this;
224
225         var body = {
226             "DocEntry": orderDetails.DocEntry,
227             "DocumentAdditionalExpenses": [{
228                 "ExpenseCode": 1,
229                 "LineTotal": providerDetails.amount,
230                 "Remarks": providerDetails.name
231             }]
232         };
233
234         $.ajax({
235             url: "/apihub_sandbox/sapb1/b1s/v2/Orders(" + orderDetails.DocEntry + ")",
236             type: "PATCH",
237             data: JSON.stringify(body),
238             contentType: "application/json",
239             success: function(data) {
240                 oThis.soUpdateSuccess(data);
241             },
242             error: function(e) {
243                 oThis.soUpdateError(e);
244             }
245         });
246     },
247
248     soUpdateSuccess: function(data) {
249         // Refresh the Model to update the Order DocTotal we have changed
250         this.getView().getBindingContext().getModel().refresh();
251         MessageToast.show("Success. Check the DocTotal amount has changed.");
252     },
253     soUpdateError: function(e) {
254         MessageToast.show("SL error: " + e);
255     }
256 });
257 });
```

Test your application.



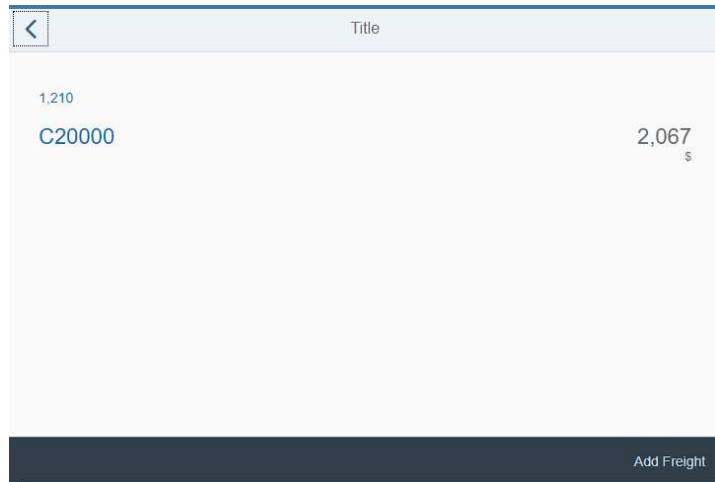
Filter the Orders table with **DocEntry higher than 1211**.

Please follow directions from instructors so you only update a specific Order DocEntry to avoid conflicts between hands-on participants.

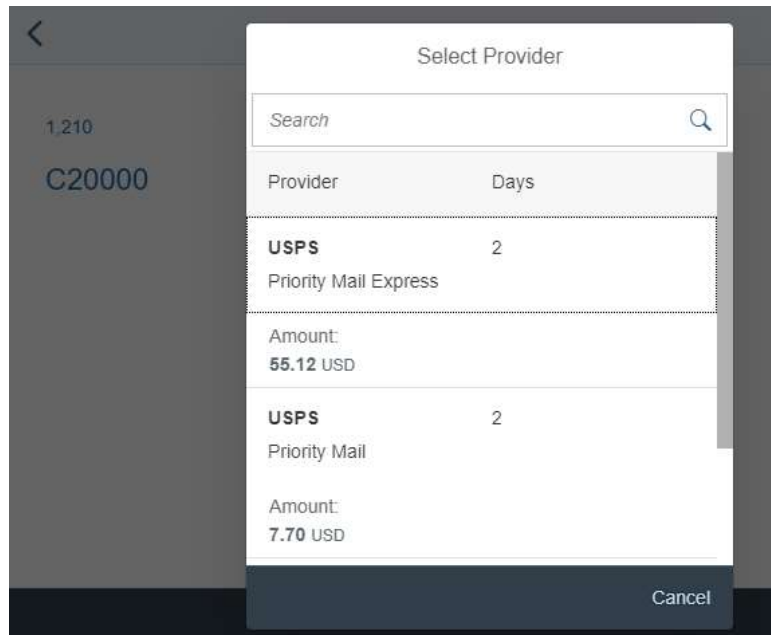
Title		
1211		
DocEntry	CardCode	DocTotal
1,211	C20000	2,067\$
1,213	C20000	2,067\$
1,215	C20000	2,067\$

Explanation	Screenshot
-------------	------------

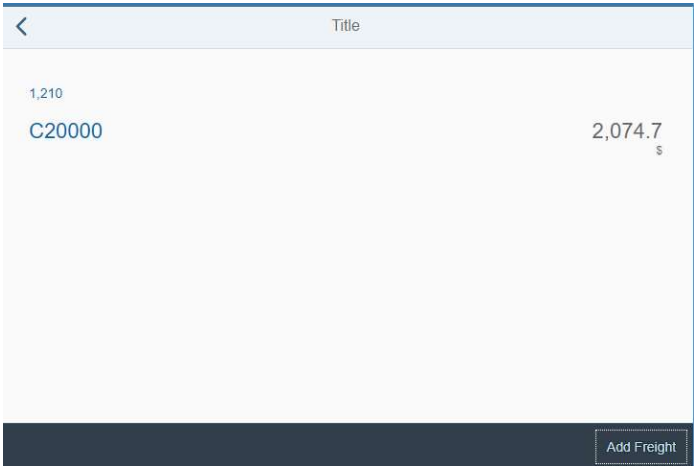
Check the Order DocTotal before adding the freight costs.



Select one of the providers.





Explanation	Screenshot
<p>Check the Order DocTotal amount is updated after you select a freight provider based on the provider costs.</p> <p>You will also see the success message if the update of the Order is successful.</p>	

Congratulations! You have just implemented your first full stack loosely coupled SAP Business One extension!

www.sap.com/contactsap

© 2018 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.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

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 or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP 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, and they should not be relied upon in making purchasing decisions.

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. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.

