



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	11
ii. Add a Data Source to the SAPUI5 Application.....	15
iii. Create a Model.....	17
iv. Add controls to the View1 view.	18
<i>Add a sap.m.Table control</i>	<i>18</i>
<i>Add a Search Field control to the sap.m.Table</i>	<i>23</i>
v. Add a second view called Details.....	27
<i>Create a Details view.</i>	<i>27</i>
<i>Add an Object Header control.....</i>	<i>28</i>
vi. Define navigation between View1 and Details.	31
STEP 2: CREATE A NODEJS APP	33
STEP 3: DEPLOY THE NODEJS APP INTO SAP CLOUD FOUNDRY	34
STEP 4: INTEGRATE THE NODEJS APP INTO THE SAP FIORI APP.....	38
i. Add a Button “Add Freight” to the Details view.	38
i. Implement the Button business logic calling the NodeJS server side and Service Layer....	40



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: Integrate the server side NodeJS application into the Fiori application created in Step 1

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

<p>Download and install git version control on your system from the following link</p> <p>https://git-scm.com/downloads</p>													
<p>We will also make use of SAP Cloud Platform Cloud Foundry Environment.</p> <p>To do so, we need the Cloud Foundry command line interface (CLI)</p> <p>You can download it and install if the CF CLI for your operating system on.</p> <p>https://github.com/cloudfoundry/cli#downloads</p>	<p>Downloads</p> <p>Installing using a package manager</p> <p>Mac OS X and Linux using Homebrew via the cloudfoundry tap:</p> <pre>brew install cloudfoundry/tap(cf-cli)</pre> <p>Debian and Ubuntu based Linux distributions:</p> <pre># ...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</pre> <p>Enterprise Linux and Fedora systems (RHEL6/CentOS6 and up):</p> <pre># ...first configure the Cloud Foundry Foundation package repository sudo wget -O /etc/yum.repos.d/cloudfoundry-clients.repo https://packages.cloudfoundry.org/fedora/cloudfoundry-cl # ...then, install the cf CLI (which will also download and add the public key to your system) sudo yum install cf-cli</pre> <p>Installers and compressed binaries</p> <table border="1"><thead><tr><th></th><th>Mac OS X 64 bit</th><th>Windows 64 bit</th><th>Linux 64 bit</th></tr></thead><tbody><tr><td>Installers</td><td>pkg</td><td>zip</td><td>rpm / deb</td></tr><tr><td>Binaries</td><td>tgz</td><td>zip</td><td>tgz</td></tr></tbody></table>		Mac OS X 64 bit	Windows 64 bit	Linux 64 bit	Installers	pkg	zip	rpm / deb	Binaries	tgz	zip	tgz
	Mac OS X 64 bit	Windows 64 bit	Linux 64 bit										
Installers	pkg	zip	rpm / deb										
Binaries	tgz	zip	tgz										

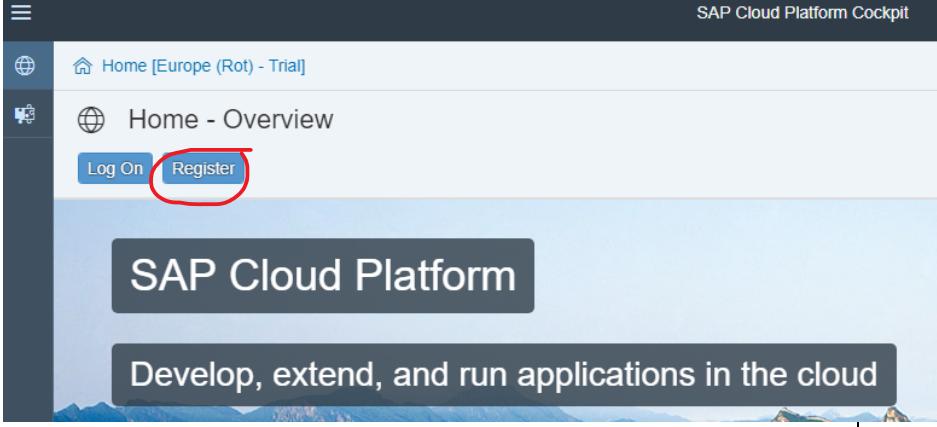
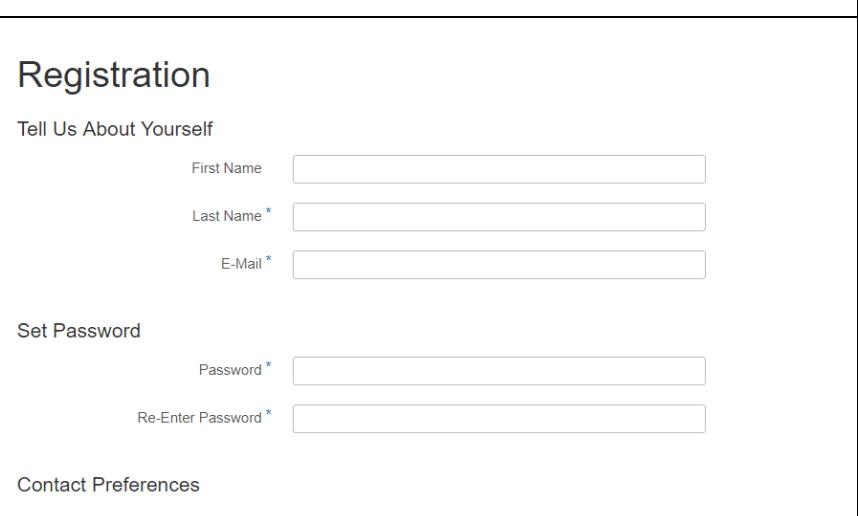
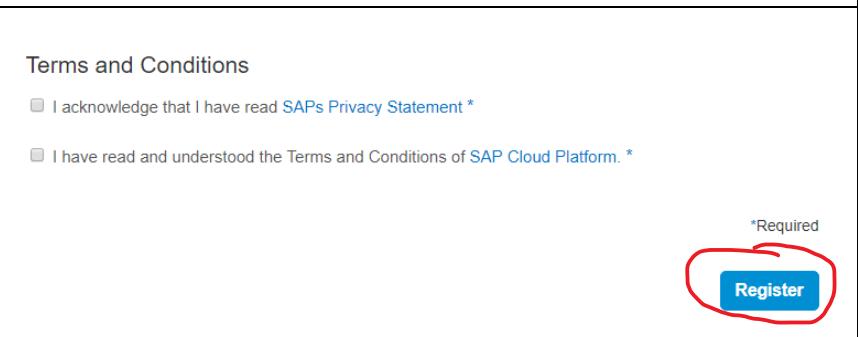


ii. Create a SAP Cloud platform trial account

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:

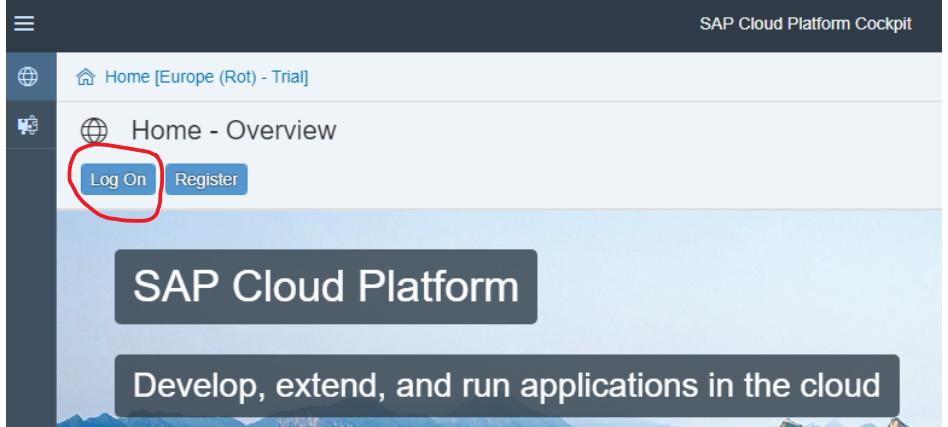
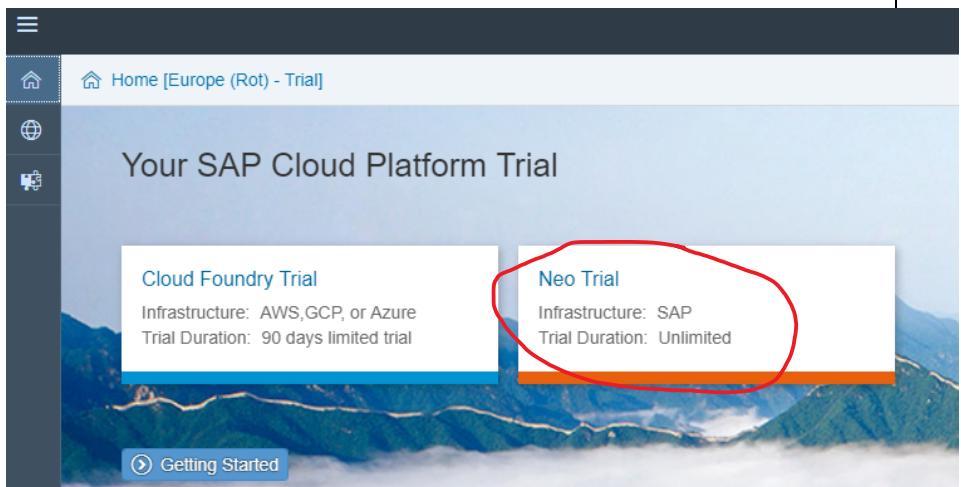
Explanation	Screenshot
To create a trial SAP Cloud Platform account, go to the following link: https://account.hanatrial.ondemand.com Press the Register button	
Enter all your details	
Accept the terms and conditions by checking both check boxes and press "Register".	



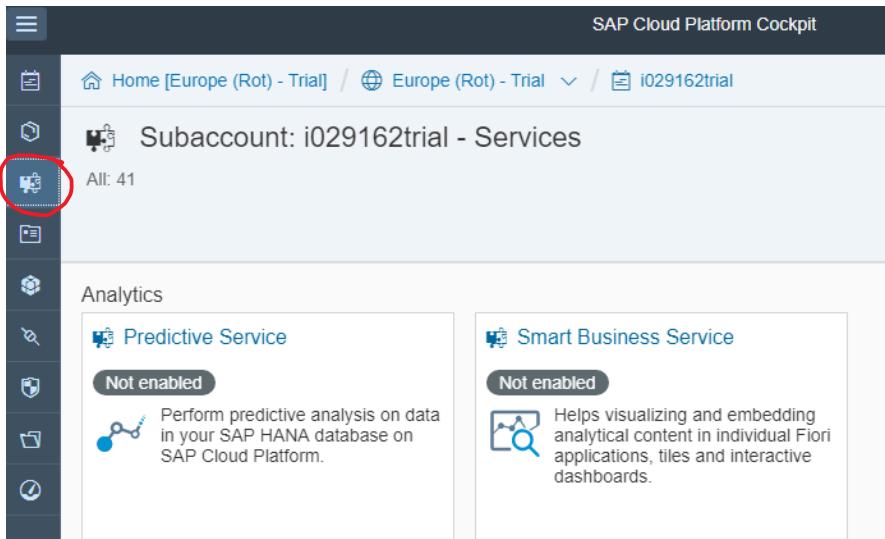
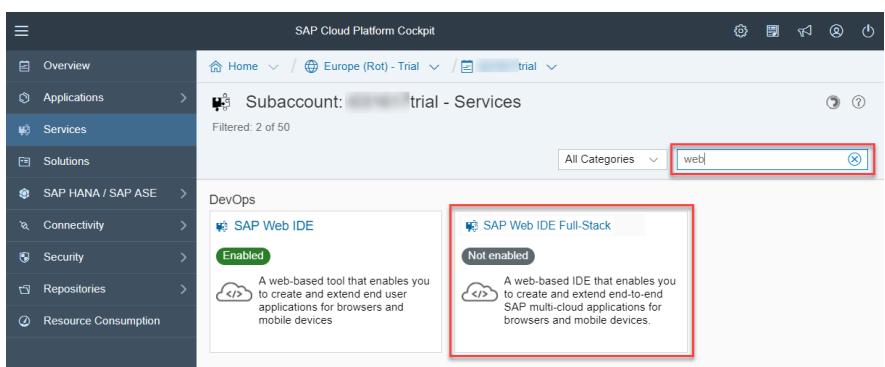
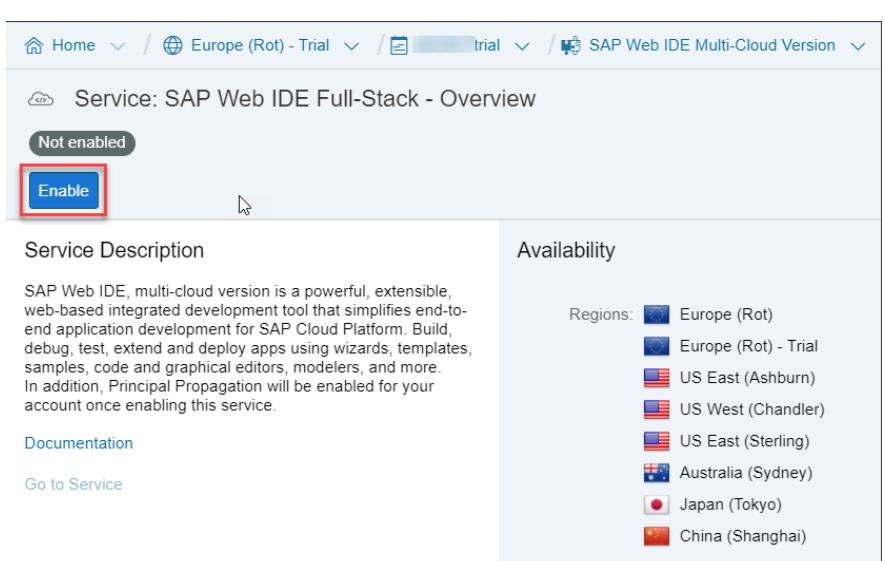
iii. Activate Web IDE Full Stack service

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.

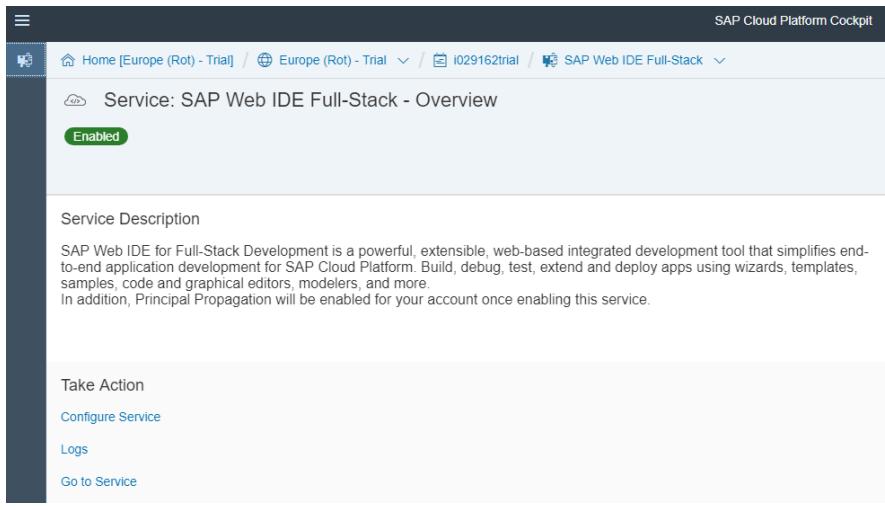
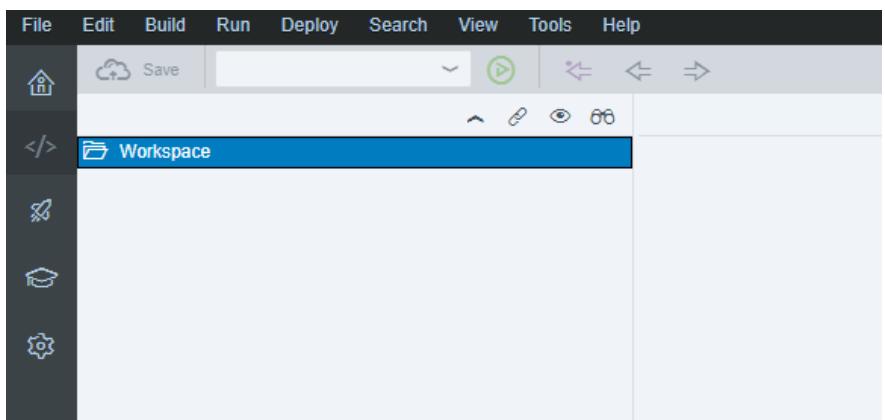
To activate Web IDE Full Stack service please follow the steps here below, if you already have Web IDE Full Stack service active in your account please skip this step.

Explanation	Screenshot
Open your trial SAP Cloud Platform account from the following link: https://account.hanatrial.ondemand.com	
Press the Log On button if you are not automatically logged in	
After login if you are proposed between Cloud Foundry Trial and Neo Trial please choose Neo Trial.	



Explanation	Screenshot
Select the Services icon on the left side bar.	 <p>SAP Cloud Platform Cockpit</p> <p>Home [Europe (Rot) - Trial] / Europe (Rot) - Trial / i029162trial</p> <p>Subaccount: i029162trial - Services</p> <p>All: 41</p> <p>Analytics</p> <p>Predictive Service: Not enabled. Perform predictive analysis on data in your SAP HANA database on SAP Cloud Platform.</p> <p>Smart Business Service: Not enabled. Helps visualizing and embedding analytical content in individual Fiori applications, tiles and interactive dashboards.</p>
Enter Web in the search edit text. Click on the SAP Web IDE Full Stack box.	 <p>SAP Cloud Platform Cockpit</p> <p>Overview</p> <p>Applications ></p> <p>Services</p> <p>Solutions</p> <p>SAP HANA / SAP ASE ></p> <p>Connectivity ></p> <p>Security ></p> <p>Repositories ></p> <p>Resource Consumption</p> <p>Subaccount: trial - Services</p> <p>Filtered: 2 of 50</p> <p>All Categories <input type="text" value="web"/> ×</p> <p>DevOps</p> <p>SAP Web IDE: Enabled. A web-based tool that enables you to create and extend end user applications for browsers and mobile devices.</p> <p>SAP Web IDE Full-Stack: Not enabled. A web-based IDE that enables you to create and extend end-to-end SAP multi-cloud applications for browsers and mobile devices.</p>
Click Enable . This may take a few minutes.	 <p>Home / Europe (Rot) - Trial / i029162trial / SAP Web IDE Multi-Cloud Version</p> <p>Service: SAP Web IDE Full-Stack - Overview</p> <p>Not enabled</p> <p>Enable</p> <p>Service Description</p> <p>SAP Web IDE, multi-cloud version 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.</p> <p>Documentation</p> <p>Go to Service</p> <p>Availability</p> <p>Regions:</p> <ul style="list-style-type: none">Europe (Rot)Europe (Rot) - TrialUS East (Ashburn)US West (Chandler)US East (Sterling)Australia (Sydney)Japan (Tokyo)China (Shanghai)



Explanation	Screenshot
<p>Once Enabled select the link Go to Service to open Web IDE Full Stack.</p>	 <p>SAP Cloud Platform Cockpit</p> <p>Home [Europe (Rot) - Trial] / Europe (Rot) - Trial / i029162trial / SAP Web IDE Full-Stack</p> <p>Service: SAP Web IDE Full-Stack - Overview</p> <p>Enabled</p> <p>Service Description</p> <p>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.</p> <p>Take Action</p> <p>Configure Service</p> <p>Logs</p> <p>Go to Service</p>
Web IDE opens with an empty Workspace unless you already developed applications with Web IDE in the past.	 <p>File Edit Build Run Deploy Search View Tools Help</p> <p>Save</p> <p>Workspace</p>



iv. SAP API Business Hub

With SAP API Business Hub, you can easily build sample apps, extensions, and integrations. Discover and consume digital content packages with APIs, pre-packaged integrations, and sample apps from SAP and select partners.

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.

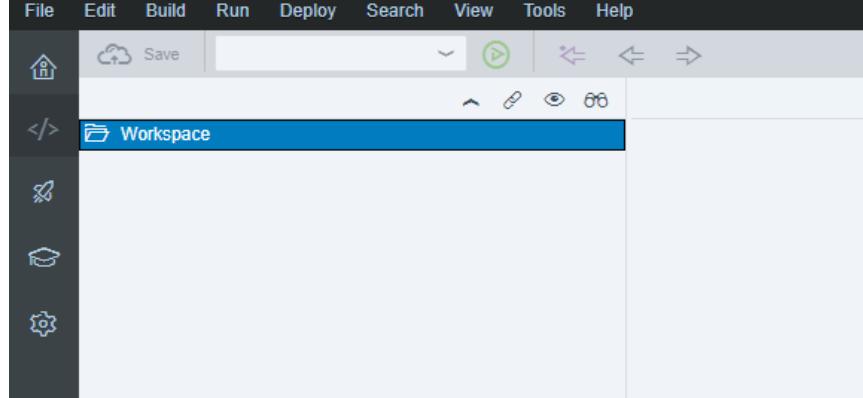
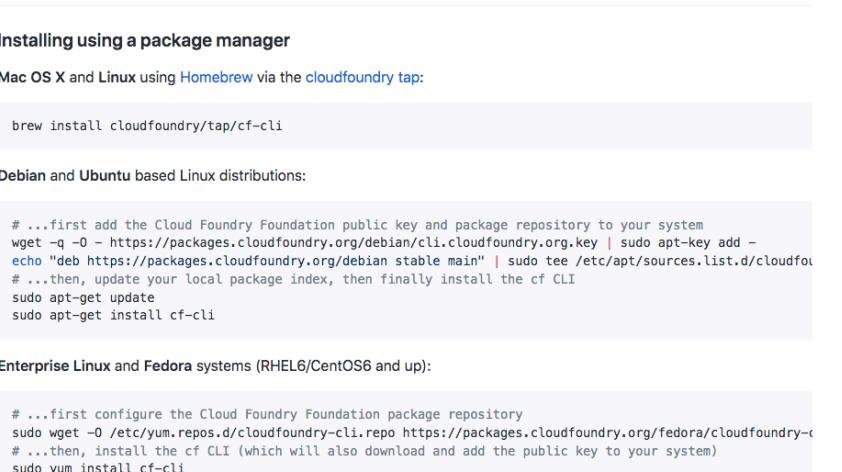
In this exercise, we will use the Service Layer APIs exposed via the SAP API Business Hub, please refer to the prerequisites section



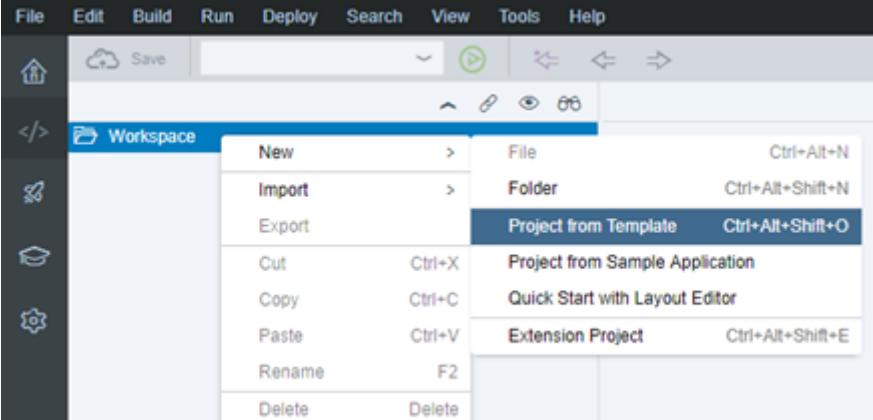
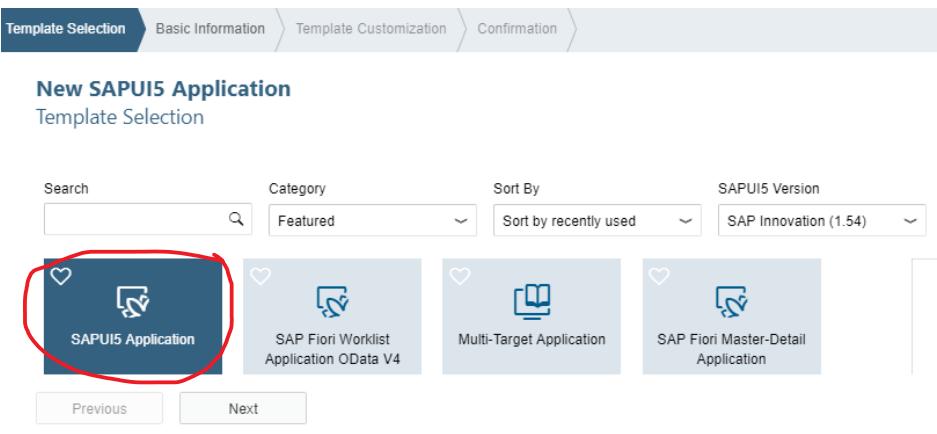
SAP API Business Hub for more details.

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
Open SAP Web IDE Full Stack.	
ii. Check the prerequisites sections Download and Install Development Tools	
Download and install git control on your system from following link https://git-scm.com/download	
We will also make use of Cloud Platform Cloud Foundry Environment. To do so, we need the Cloud Foundry command line interface (CLI) You can download it and install the CF CLI for your operating system on. https://github.com/cloudfoundry/cli#download	



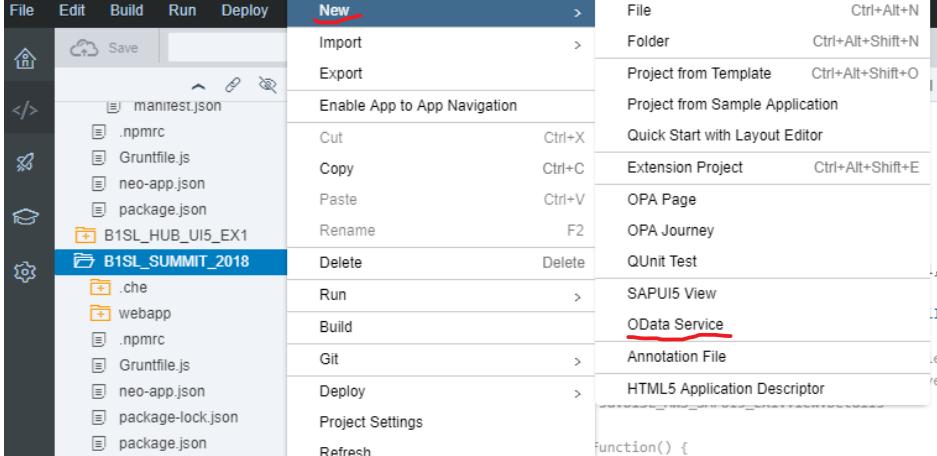
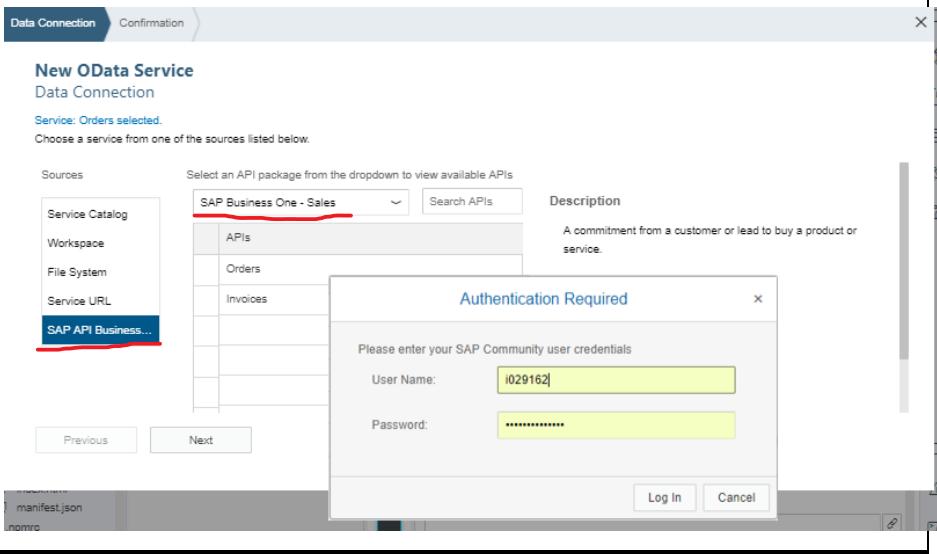
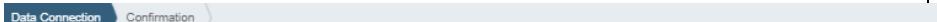
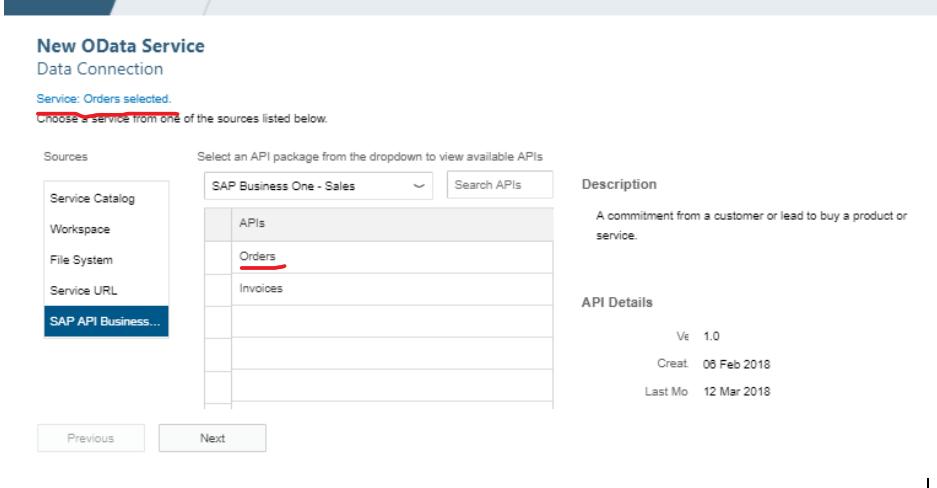
Explanation	Screenshot												
	<p style="text-align: center;">Installers and compressed binaries</p> <table border="1" style="margin-left: auto; margin-right: auto;"><thead><tr><th></th><th>Mac OS X 64 bit</th><th>Windows 64 bit</th><th>Linux 64 bit</th></tr></thead><tbody><tr><td>Installers</td><td>pkg</td><td>zip</td><td>rpm / deb</td></tr><tr><td>Binaries</td><td>tgz</td><td>zip</td><td>tgz</td></tr></tbody></table>		Mac OS X 64 bit	Windows 64 bit	Linux 64 bit	Installers	pkg	zip	rpm / deb	Binaries	tgz	zip	tgz
	Mac OS X 64 bit	Windows 64 bit	Linux 64 bit										
Installers	pkg	zip	rpm / deb										
Binaries	tgz	zip	tgz										
Create a SAP Cloud platform trial account and Activate Web IDE Full Stack service if you don't know how to open Web IDE Full Stack.													
Right click on your Workspace and select New -> Project from Template.													
Select the SAPUI5 Application template. Press Next . If you don't see this template, change the Category to All categories .	<p>Template Selection Basic Information Template Customization Confirmation</p> <p>New SAPUI5 Application Template Selection</p> <p>Search Category Sort By SAPUI5 Version</p> <p>Featured Sort by recently used SAP Innovation (1.54)</p> <p>SAPUI5 Application (circled in red) SAP Fiori Worklist Application OData V4 Multi-Target Application SAP Fiori Master-Detail Application</p> <p>Previous Next</p> 												



Explanation	Screenshot
<p>Enter a Project Name.</p> <p>Enter a Namespace.</p> <p>Press Next.</p>	<p>New SAPUI5 Application Basic Information</p> <p>Project Name* B1SL_SUMMIT_2018</p> <p>Namespace* sal</p> <p>Previous Next</p>
<p>Keep the Initial View Details with the default values.</p> <p>Press Next.</p>	<p>New SAPUI5 Application Template Customization</p> <p>Initial View Details View Type* XML</p> <p>View Name View1</p> <p>Previous Next Finish</p>
<p>Press Finish to confirm the creation of the SAPUI5 template.</p>	<p>New SAPUI5 Application Confirmation</p> <p>Click Finish. A new project named B1SL_SUMMIT_2018 will be created in your workspace.</p> <p>Previous Finish</p>



iii. Add a Data Source to the SAPUI5 Application

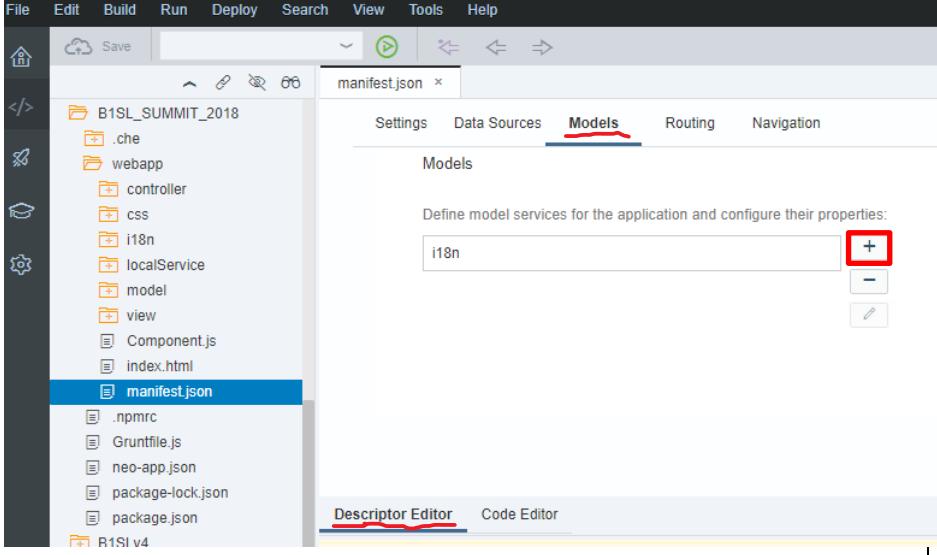
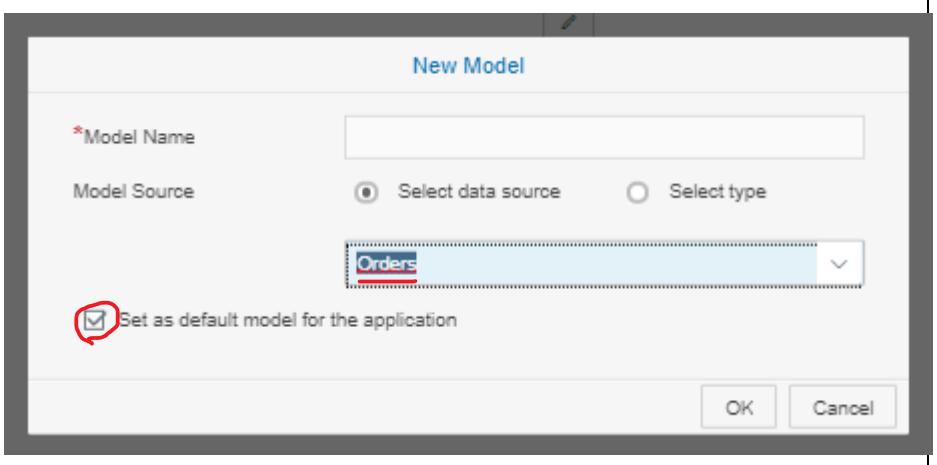
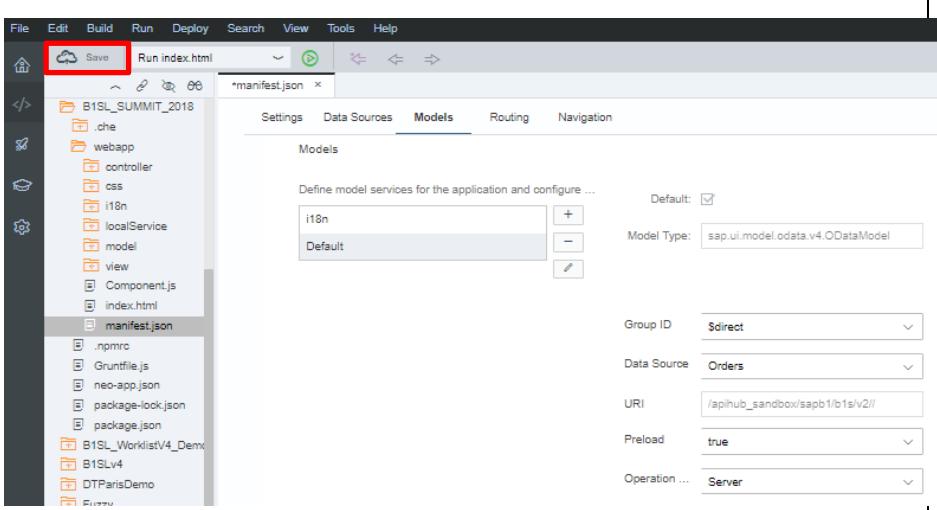
Explanation	Screenshot
In Web IDE select your project and right click to get the menus.	
Select New -> OData Service .	
Select SAP API Business Hub in the Sources (left side of the screen). Select the “ SAP Business One – Sales ” API package from the dropdown control. You might be prompted to enter your SAP Community User Name and Password. Enter your credentials and press Log In.	
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. Press Next .	



Explanation	Screenshot
Press Finish.	<p>New OData Service Confirmation</p> <p>Click Finish. The selected OData service is connected to the B1SL_SUMMIT_2018 project.</p> <p><input type="checkbox"/> Overwrite existing OData service connection</p> <p>Previous Finish</p>
<p>Open the file “webap/manifest.json” inside your application and select the Code Editor.</p> <p>Search for the element “dataSources”/“Orders” and change the “odataVersion” value inside “settings” to “4.0”.</p> <p>Save your changes.</p>	<pre>version": "1.0.0" }, "title": "{{appTitle}}", "description": "{{appDescription}}", "sourceTemplate": { "id": "servicecatalog.connectivityComponent", "version": "0.0.0" }, "dataSources": { "Orders": { "uri": "/apihub_sandbox/sapb1/b1s/v2/", "type": "OData", "settings": { "odataVersion": "4.0", "localUri": "localService/metadata.xml" } } }</pre> <p>File Edit Build Run Deploy Search View Tools Help</p> <p>Save</p> <p>manifest.json</p> <p>Descriptor Editor Code Editor</p>



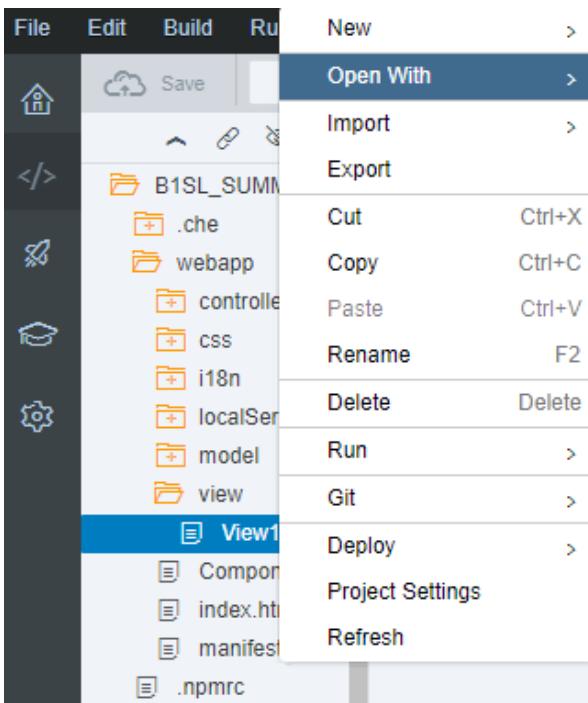
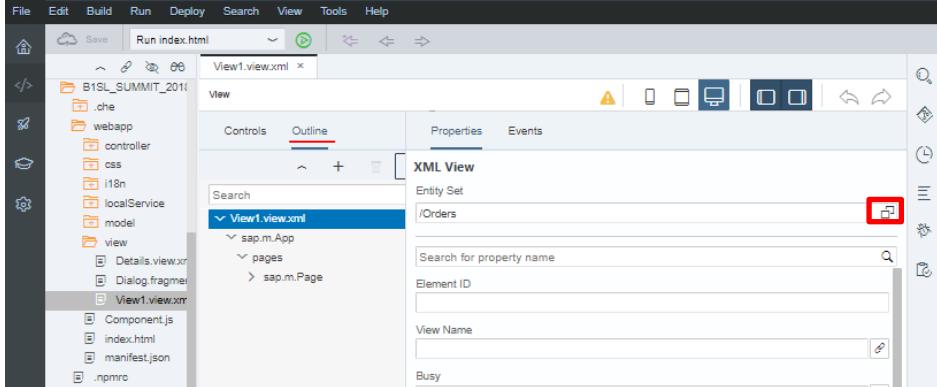
iv. Create a Model

Explanation	Screenshot
<p>Open the manifest.json file, select Descriptor Editor (bottom of the screen) and Models tab.</p> <p>Press the + button to add a new model.</p>	
<p>Check the box “Set as default model for the application”.</p> <p>There is no need to enter a Model Name as we checked the default model option.</p> <p>Select Orders as the Model Source.</p> <p>Press OK.</p>	
<p>A new Default model will be shown in the Models tab pointing to the API Hub URI for OData v4 Data Source.</p> <p>Press Save button to your changes.</p>	

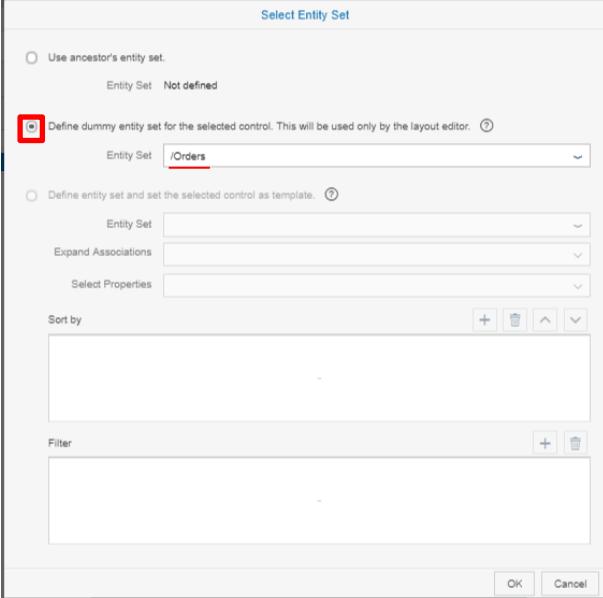
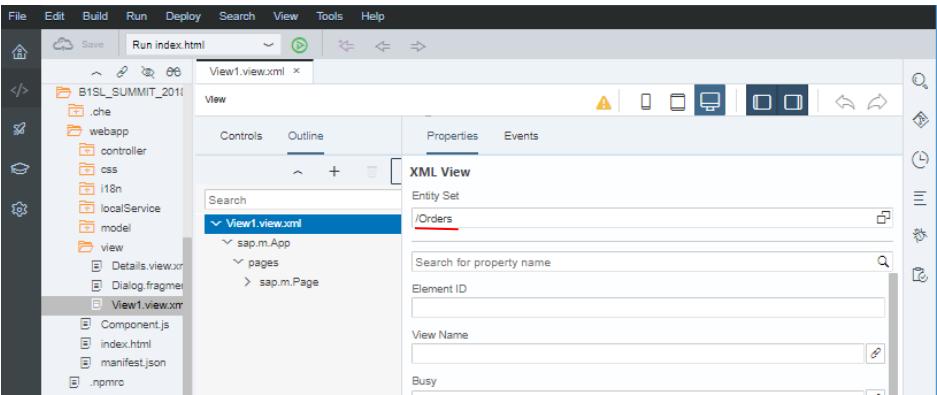


v. Add controls to the View1 view.

Add a sap.m.Table control

Explanation	Screenshot
<p>Open the view folder. Open the View1.view.xml file with the Layout Editor.</p>	
<p>Select the Outline tab. Select View.view.xml. Click on the  button to select the Entity Set associated to our view.</p>	

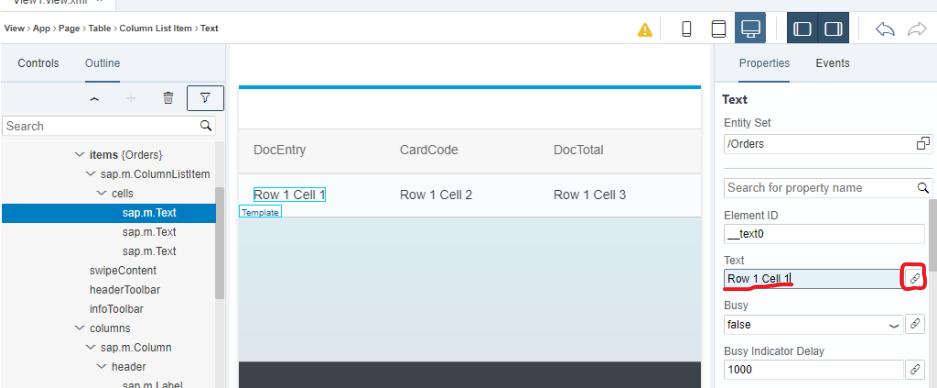
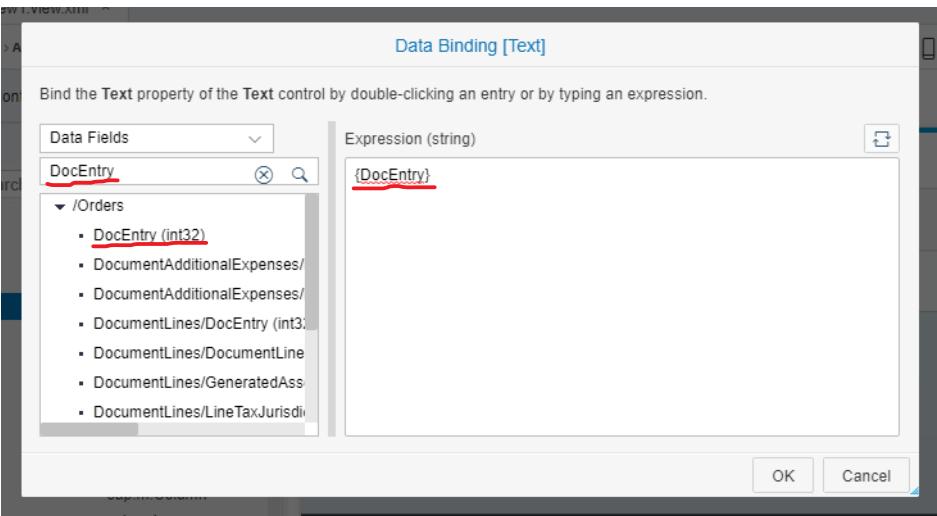
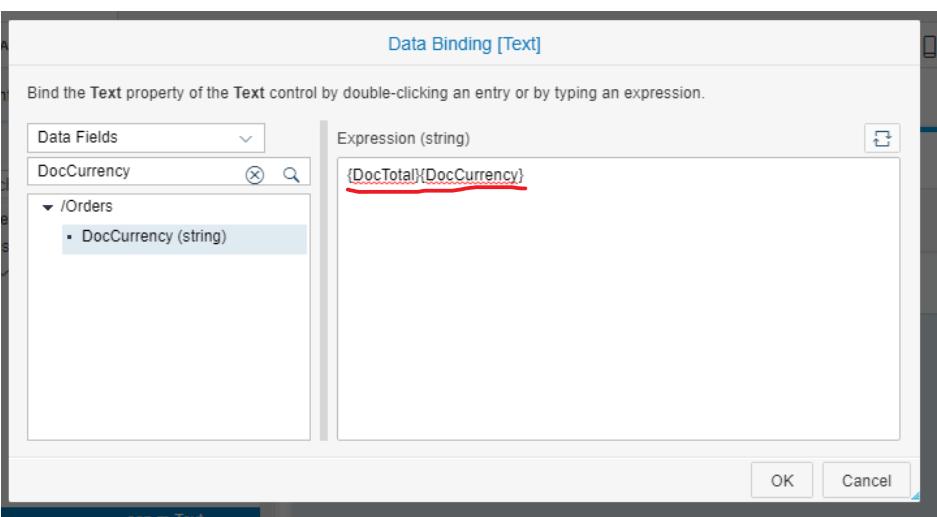


Explanation	Screenshot
<p>Check the second option Define dummy entity set... and choose the /Orders Entity Set.</p> <p>Press OK.</p>	
<p>Once back to the Layout Editor the /Orders entity should be shown.</p> <p>Press Save button.</p>	

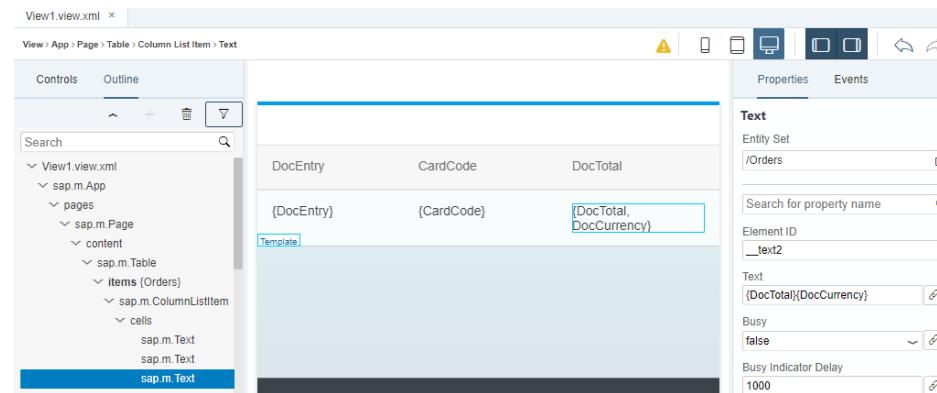
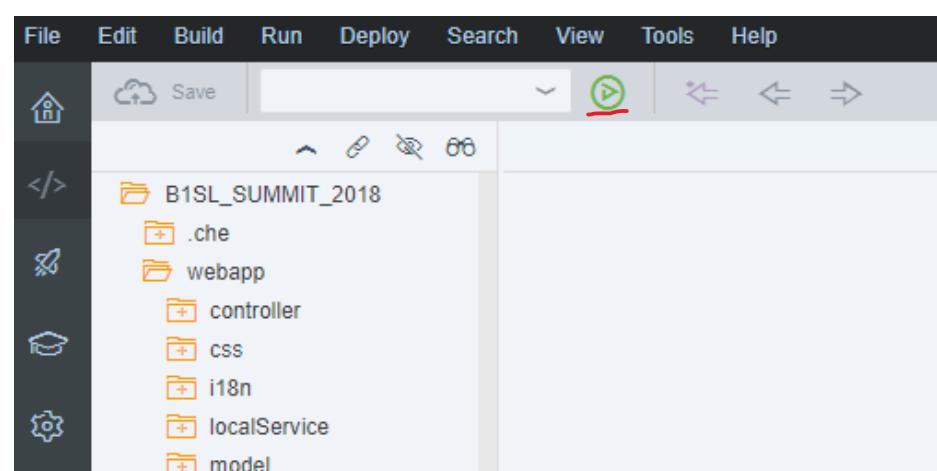
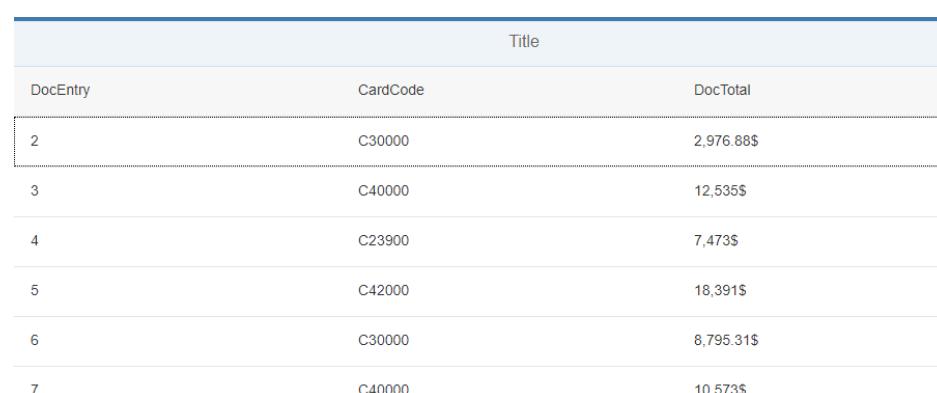


Explanation	Screenshot
<p>Expand sap.m.Page.</p> <p>Right click on content and select Add.</p>	<p>The screenshot shows the SAP Studio interface with the file tree on the left and the XML editor on the right. In the XML editor, under the 'sap.m.Page' node, there is a 'content' node. A context menu is open over the 'content' node, with the 'Add' option highlighted. Other options like 'Convert to Fragment' and 'Add Fragment' are also visible.</p>
<p>Enter table to filter the controls shown in the list.</p> <p>Click on the Table control so it will be added to our view.</p>	<p>The screenshot shows the SAP Studio interface with the 'Add Control to: content' dialog open. The search bar contains 'table', and the 'Table (sap.m)' control is listed and highlighted.</p>
<p>On the table created edit each sap.m.Column -> header -> sap.m.Label</p> <p>Text Property:</p> <ul style="list-style-type: none">- Change Header 1 by DocEntry.- Change Header 2 by CardCode.- Change Header 3 by DocTotal.	<p>The screenshot shows the SAP Studio interface with the XML editor displaying a table structure. In the properties panel on the right, for a 'sap.m.Label' control under the 'header' node of a 'sap.m.Column', the 'Text' property is set to 'DocEntry'. The XML code in the editor shows the 'DocEntry' placeholder in the header cells.</p>

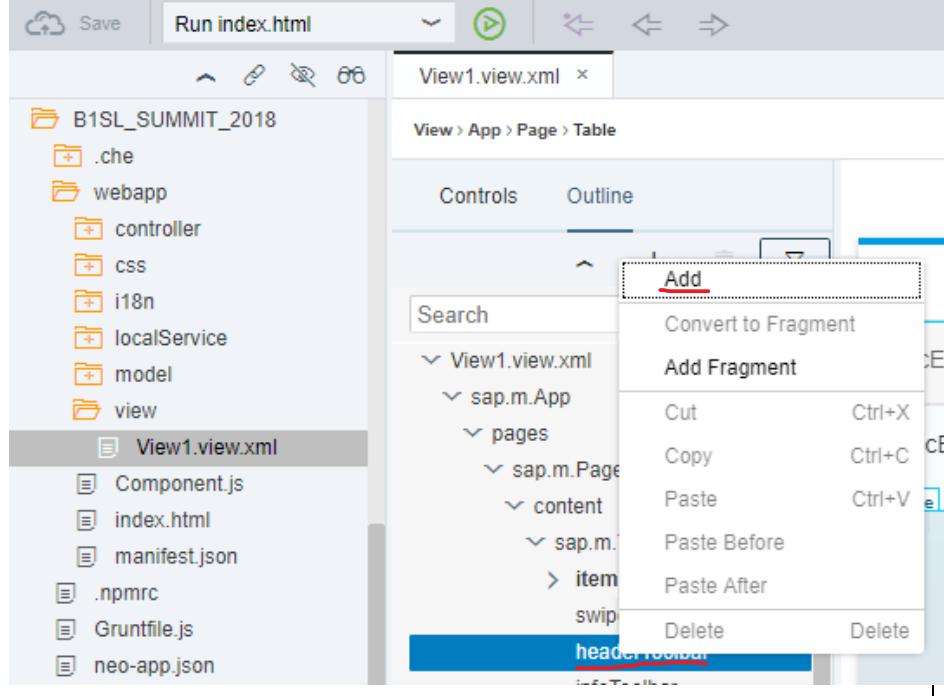
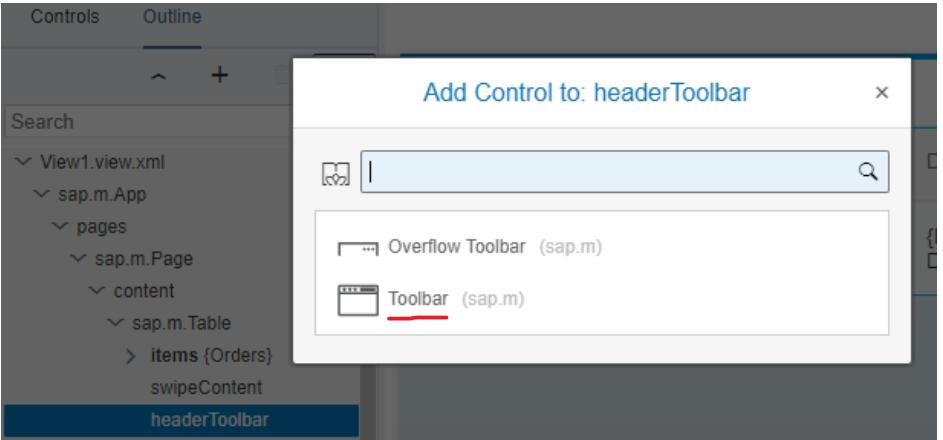


Explanation	Screenshot
<p>Select each one of the sap.m.ColumnListItem -> sap.m.Text cells to bind them to their corresponding Orders properties.</p> <p>Then on the Text property click the Binding button on the right side of the Text property.</p>	
<p>Delete the default Expression string.</p> <p>In the Search for data field enter the name of the Orders property DocEntry.</p> <p>Double click on the field name so it will be copied to the Expression string between curly brackets.</p> <p>Repeat this step for CardCode column.</p>	
<p>For the DocTotal column we will add 2 properties:</p> <ul style="list-style-type: none">- DocTotal- DocCurrency <p>We will then search and add first the DocTotal and afterwards search and add the DocCurrency.</p> <p>Press Save button.</p>	

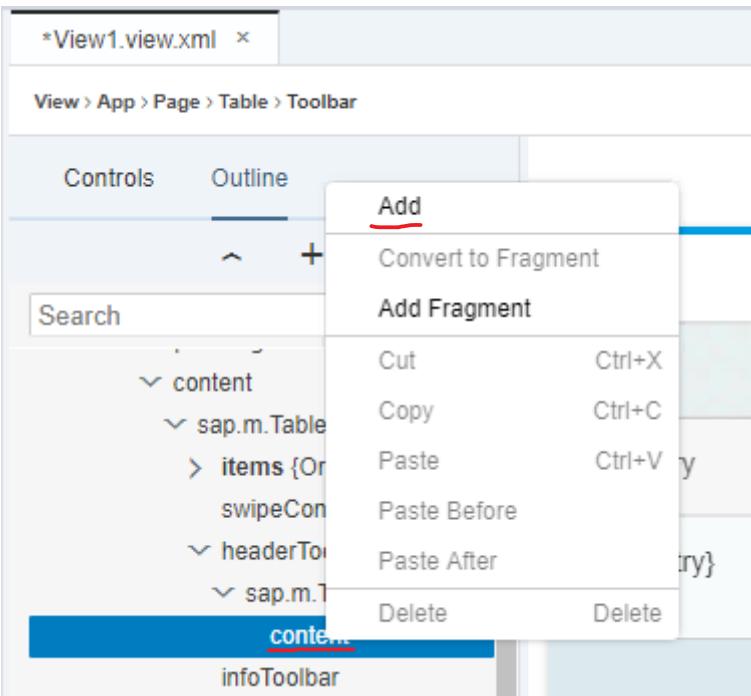
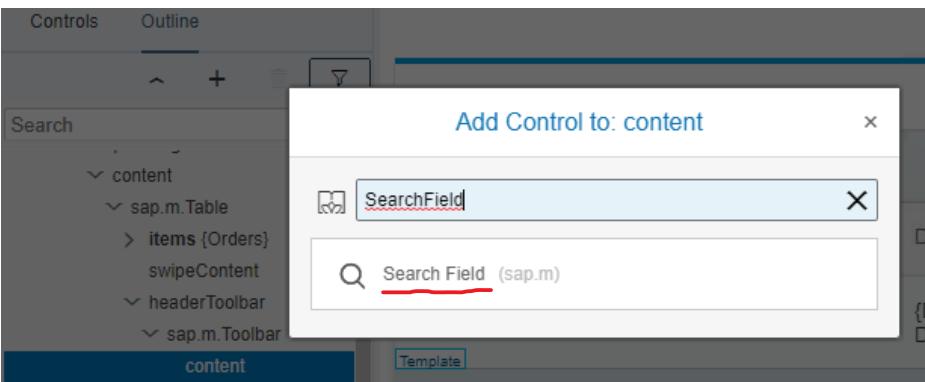
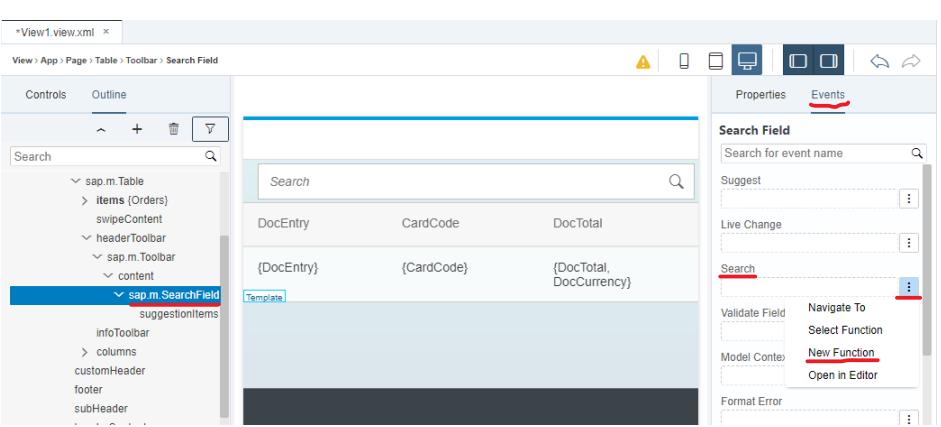


Explanation	Screenshot																						
Your View1.view.xml layout should look like this at this point.																							
Click on the green arrow to run your application.																							
A new tab opens showing your application. You can see the Table is automatically filled with the list of Orders available in the API Business Hub sandbox running SAP Business One.	 <table border="1"><thead><tr><th></th><th>DocEntry</th><th>CardCode</th><th>DocTotal</th></tr></thead><tbody><tr><td>2</td><td>C30000</td><td>2,976.88\$</td></tr><tr><td>3</td><td>C40000</td><td>12,535\$</td></tr><tr><td>4</td><td>C23900</td><td>7,473\$</td></tr><tr><td>5</td><td>C42000</td><td>18,391\$</td></tr><tr><td>6</td><td>C30000</td><td>8,795.31\$</td></tr><tr><td>7</td><td>C40000</td><td>10,573\$</td></tr></tbody></table>		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\$
	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\$																					

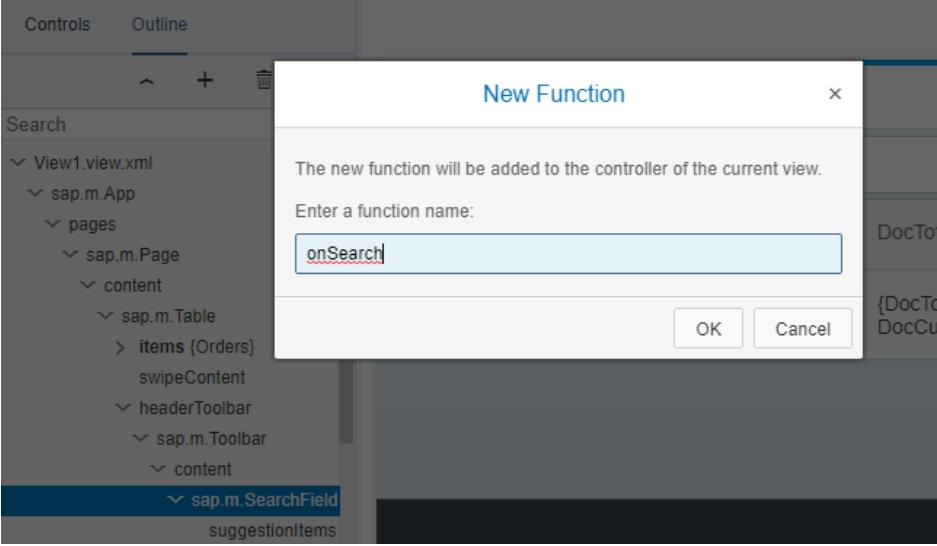
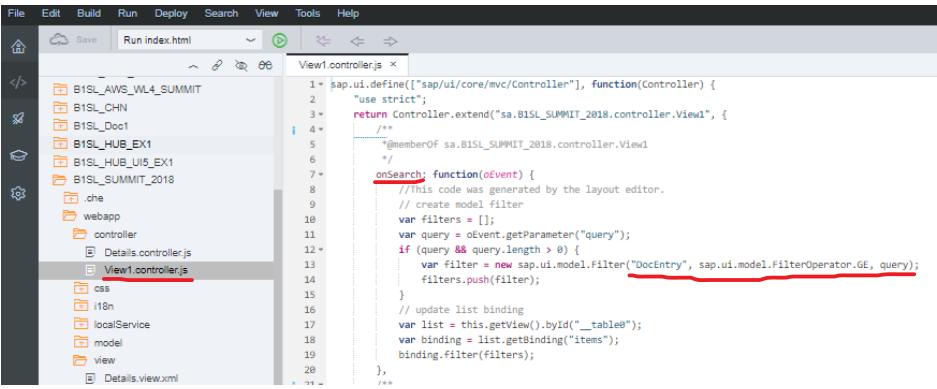
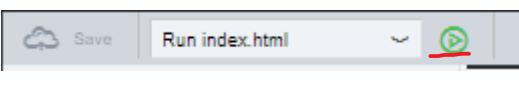
Add a Search Field control to the sap.m.Table

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>	 A screenshot of the SAP Fiori Launchpad interface. On the left, there's a file tree with a folder named 'B1SL_SUMMIT_2018' containing subfolders like '.che', 'webapp', 'controller', 'css', 'i18n', 'localService', 'model', and 'view'. Under 'view', the file 'View1.view.xml' is selected. In the center, the 'Outline' tab is active, showing a tree structure with 'View > App > Page > Table'. A context menu is open over one of the table items. The menu has a header 'Add' and several options: 'Convert to Fragment', 'Add Fragment', 'Cut (Ctrl+X)', 'Copy (Ctrl+C)', 'Paste (Ctrl+V)', 'Paste Before', 'Paste After', 'Swipe', 'Delete', and 'Delete' (highlighted with a red underline). The 'headerToolbar' option is also visible at the bottom of the menu.
Double click on the Toolbar proposed control.	 A screenshot of the 'Add Control to: headerToolbar' dialog box. The dialog has a search bar at the top. Below it, a list shows two items: 'Overflow Toolbar (sap.m)' and 'Toolbar (sap.m)'. The 'Toolbar (sap.m)' item is highlighted with a red underline.



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_SC_P_HandsOn/blob/master/snippets/View1.controller.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> <p>Check that the SearchField filters the table with Orders having a DocEntry higher than the entered value.</p>	

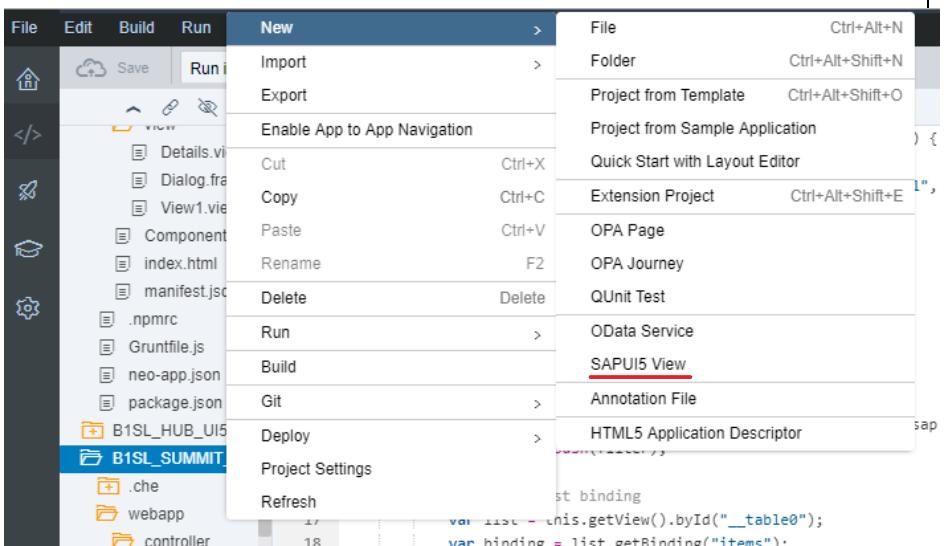
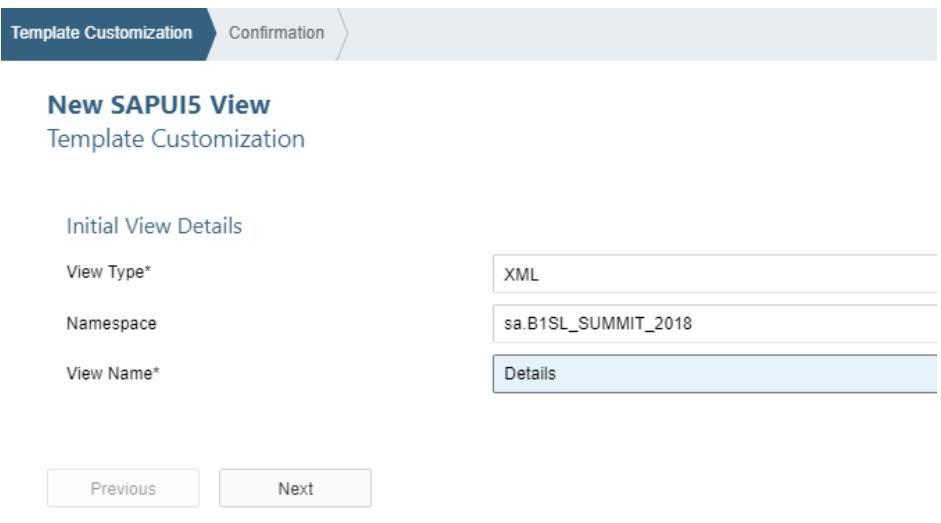
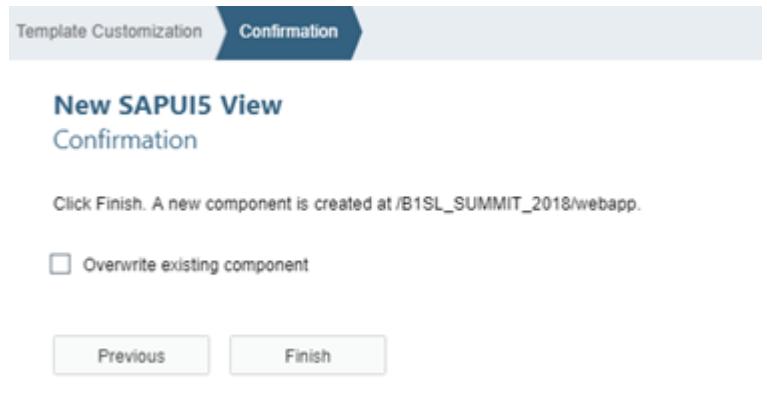


Explanation	Screenshot		
	DocEntry	CardCode	DocTotal
	1,200	1103914	5,000\$
	1,201	1103914	5,000\$
	1,202	1103914	5,000\$
	1,203	1103914	5,000\$
	1,204	1103914	5,000\$
	1,205	1103914	5,000\$
	1,207	C20000	1,915.7\$



vi. 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>Keep the default values for View Type and Namespace. Enter the name of the new View: Details. Press Next.</p>	
<p>Press Finish to confirm the creation of the Details view.</p>	

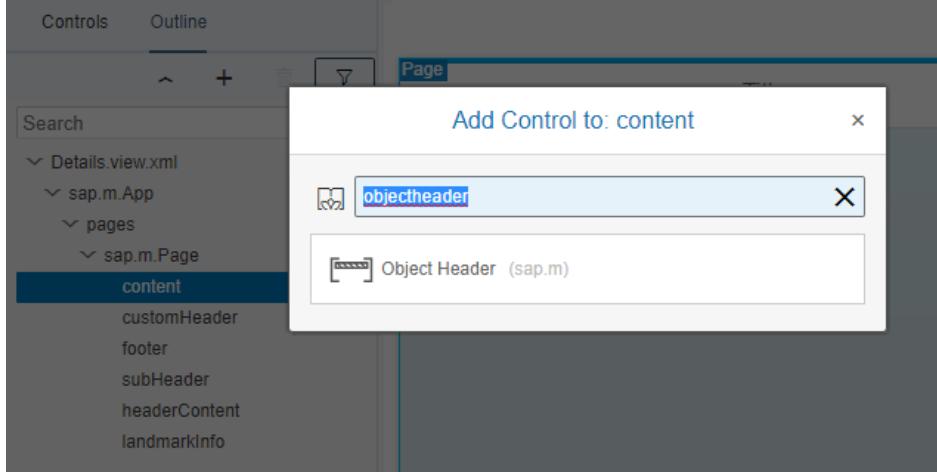
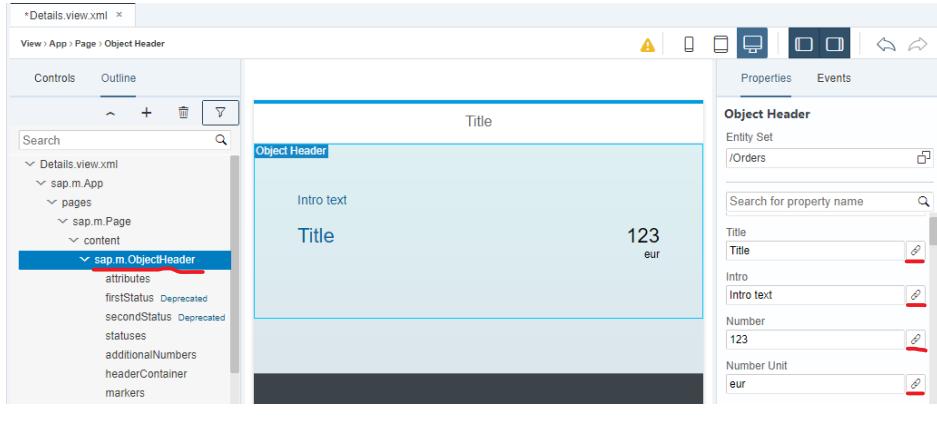
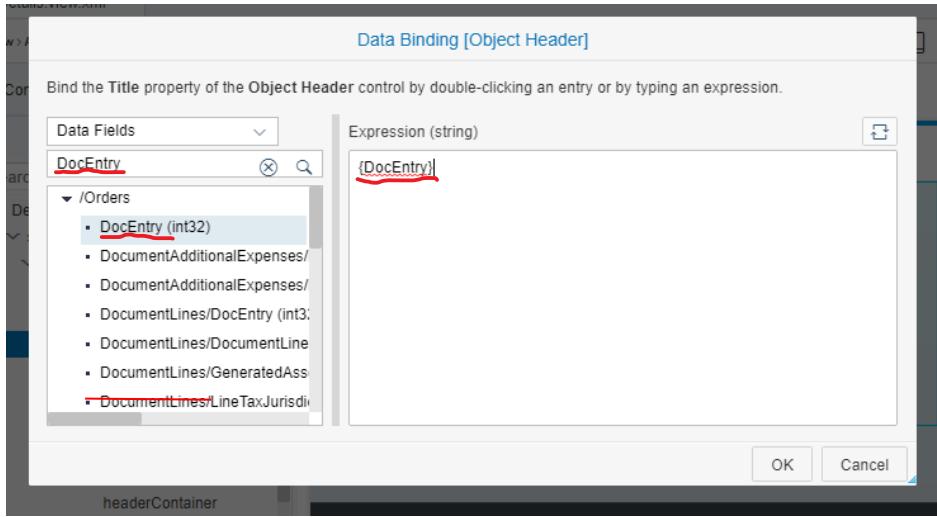


Explanation	Screenshot
<p>Two files are created:</p> <ul style="list-style-type: none">- Details.view.xml- Details.controller.js.	<p>The screenshot shows the SAP Studio file tree for a project named 'B1SL_SUMMIT_2018'. The 'controller' folder contains 'Details.controller.js' and 'View1.controller.js'. The 'view' folder contains 'Details.view.xml' and 'View1.view.xml'. The 'Details.view.xml' file is highlighted with a blue selection bar at the bottom of its icon.</p>

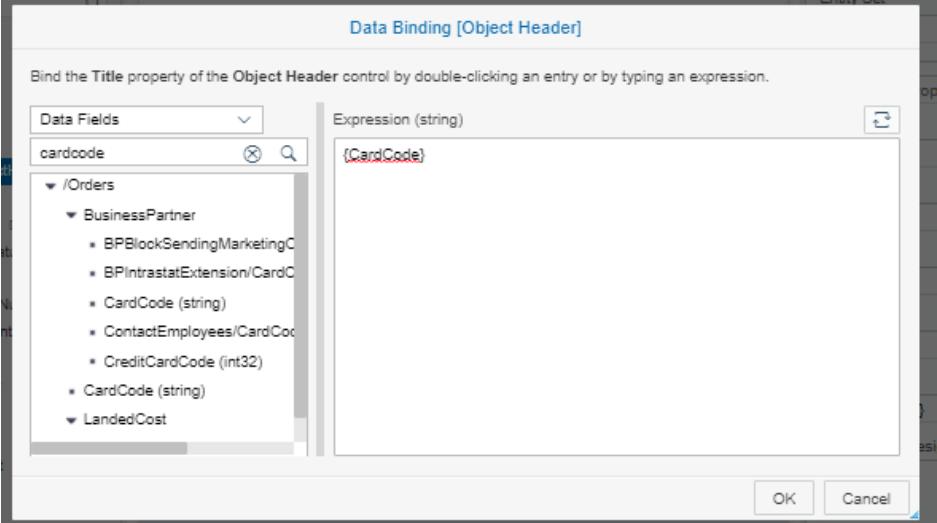
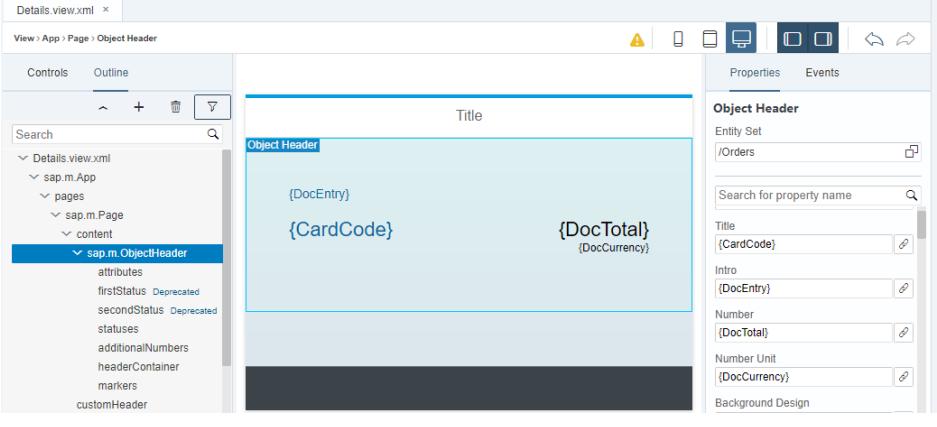
Add an Object Header control.

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>	<p>The screenshot shows the SAP Studio Layout editor with the 'Details.view.xml' file open. The 'Outline' tab is selected in the top navigation bar. A context menu is open over the 'content' node under 'sap.m.Page'. The 'Add' option is highlighted with a red underline. Other options in the menu include 'Convert to Fragment', 'Add Fragment', 'Cut', 'Copy', 'Paste', 'Paste Before', 'Paste After', and 'Delete'.</p>



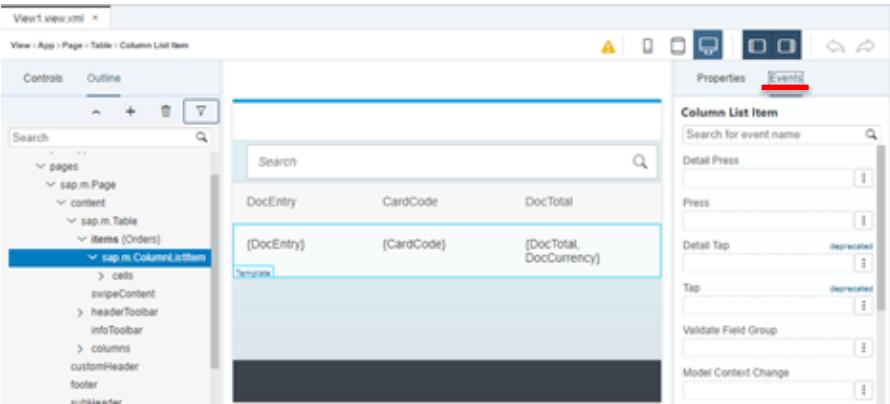
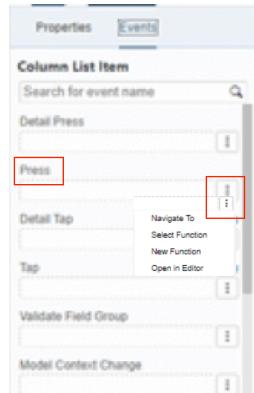
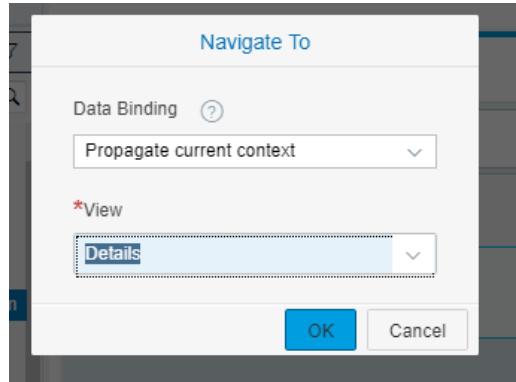
Explanation	Screenshot
<p>Enter objectheader to search for the Object Header control.</p> <p>Click on the Object Header proposed control.</p>	
<p>Select the created sap.m.ObjectHeader element in the Outline tree.</p> <p>Go to the Properties tab in the right side of the screen.</p> <p>Click one by one on the Data Binding icon for the properties:</p> <ul style="list-style-type: none">- Title- Intro- Number- Number Unit	
<p>A new dialog will be presented for each one of the properties.</p> <p>Search for the Orders Data Field to be assigned to each property:</p> <ul style="list-style-type: none">- Title -> CardCode- Intro -> DocEntry- Number -> DocTotal- Number Unit -> DocCurrency <p>and double click on the proposed list to get it into the Expression string.</p>	



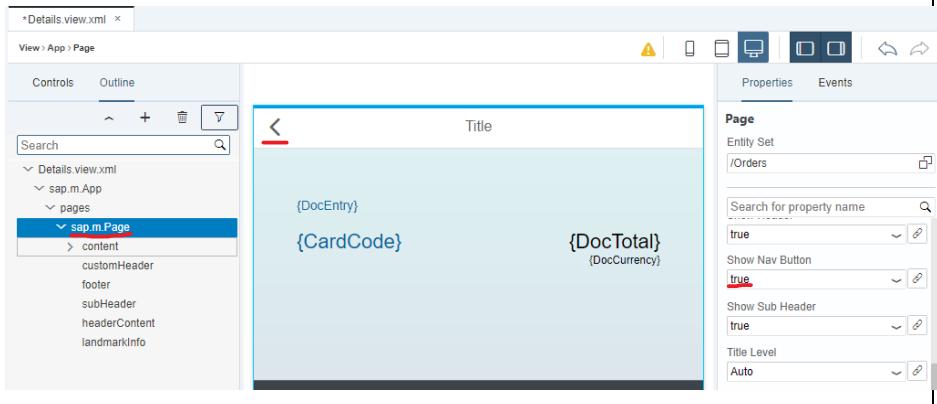
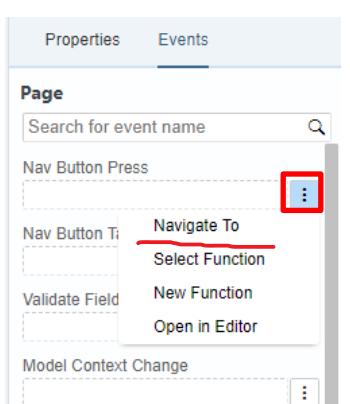
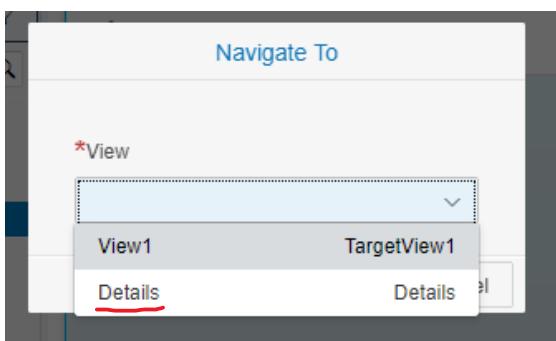
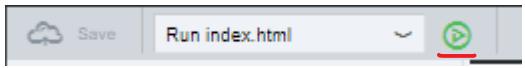
Explanation	Screenshot
<p>Note: Do not select BusinessPartners/CardCode for the CardCode property.</p>	
Press Save button.	
The sap.m.ObjectHeader should look like this at this point.	



vii. 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 then sap.m.ColumnsList Item. Select the Events tab.</p>	
<p>Click on the Press 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

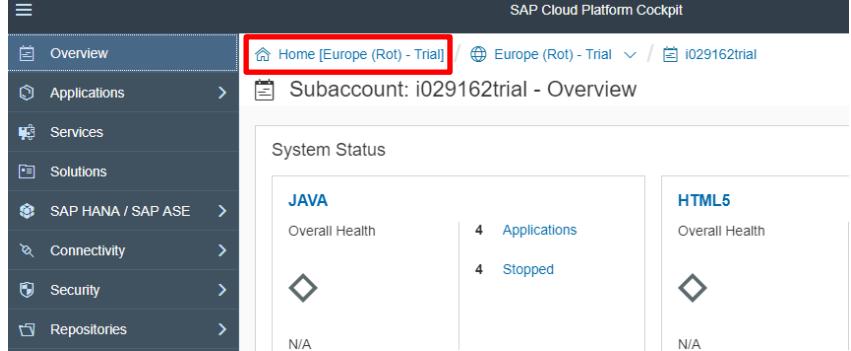
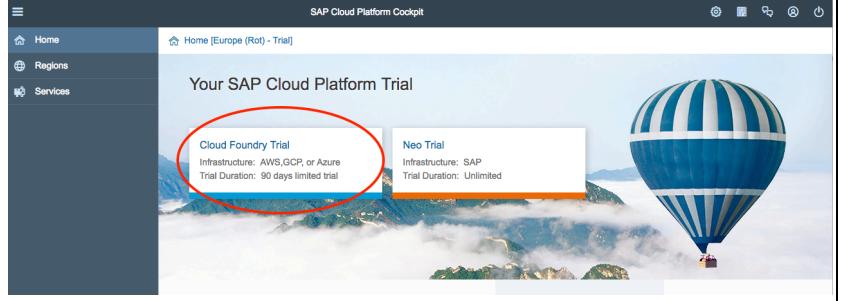
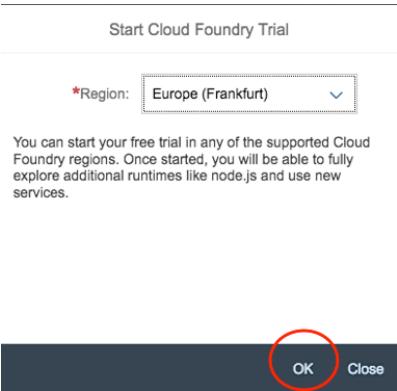
In this step we are going to implement the backend of our application. It will contain a simple business logic to calculate freight costs from different providers and consume 3rd Party services. The application is written in NodeJS and the source code is available on GitHub.

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

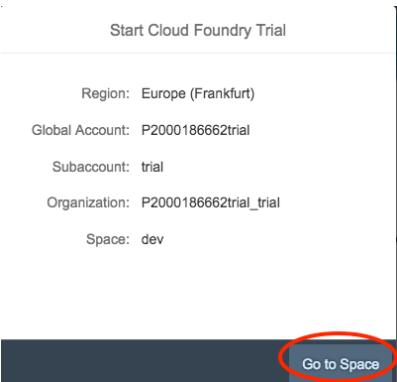
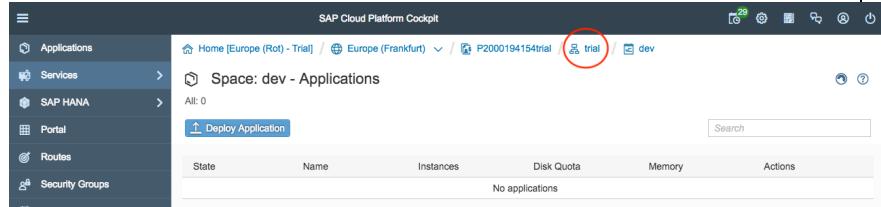
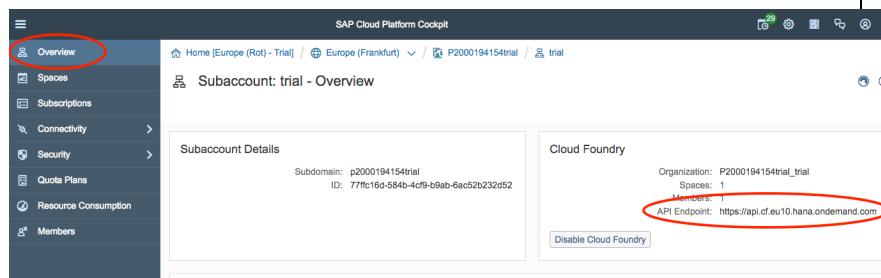
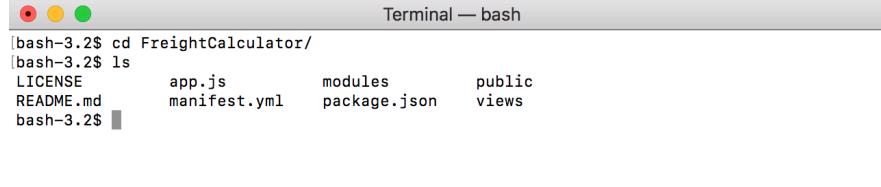


STEP 3: DEPLOY THE NODEJS APP INTO SAP CLOUD FOUNDRY

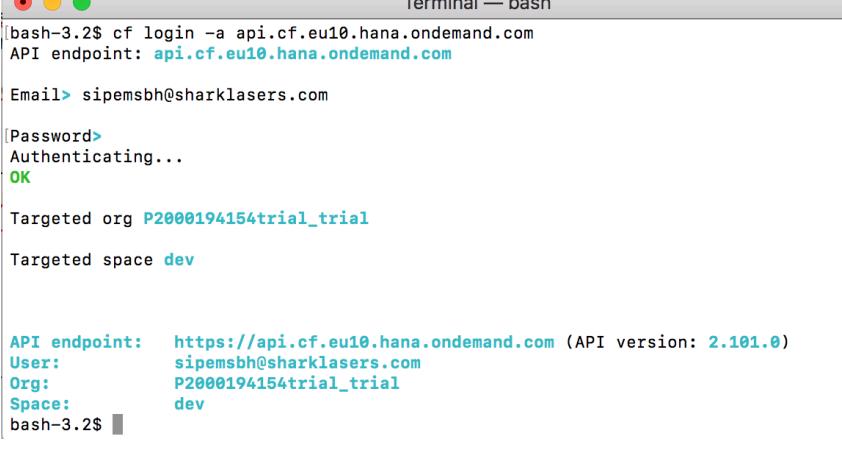
In this step, we are going to deploy our Freight Calculator app to SAP Cloud Platform Cloud Foundry.

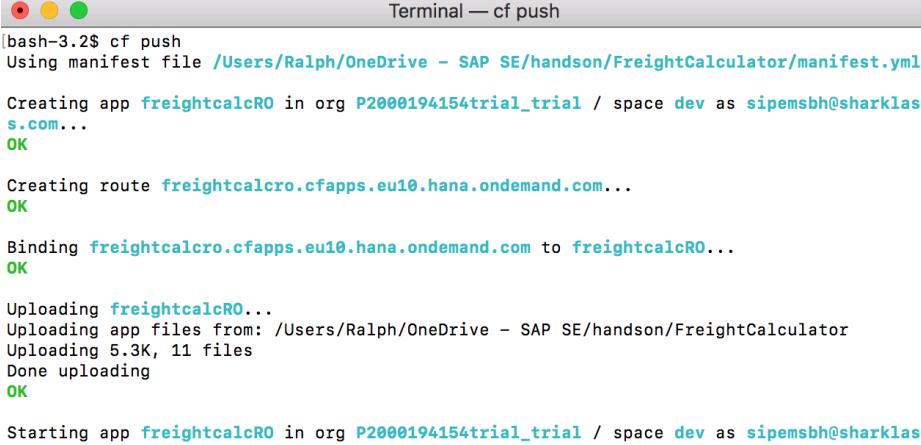
Explanation	Screenshot
<p>First, we need to activate the SAP Cloud Platform Cloud Foundry Environment.</p> <p>On the SAP Cloud Platform Dashboard click on the HOME option</p>	
<p>First, we need to activate the SAP Cloud Platform Cloud Foundry Environment.</p> <p>From the SAP CP Home Screen, click on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

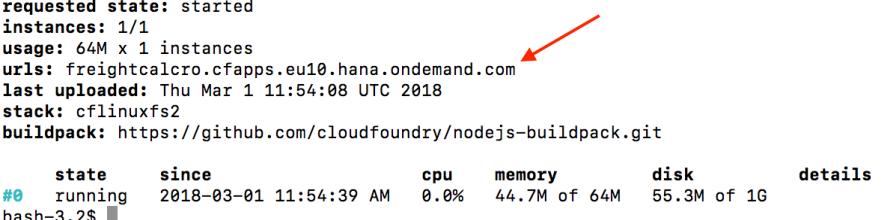


Explanation	Screenshot
<p>This will initialize your Cloud Foundry Trial and create a DEV space (where the solutions will be deployed). Go Ahead and access your space.</p>	
<p>Your space is created and ready to deploy your app. We now need your environment endpoint to be able to push our app. Click on the trial link</p>	
<p>And then on Overview option in the menu Select and copy your API Endpoint. E.g. https://api.cf.eu10.hana.ondemand.com</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 Congratulations! You have created your first SAP UI5 application connecting to SAP API Business Hub STEP 2: Create a NodeJS app of this guide</p>	



Explanation	Screenshot
<p>From that folder, login to Cloud foundry using the command <code>cf login -a <API ENDPOINT></code></p> <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>	 <pre>bash-3.2\$ cf login -a api.cf.eu10.hana.ondemand.com API endpoint: api.cf.eu10.hana.ondemand.com Email> sipemsbh@sharklasers.com Password> Authenticating... OK Targeted org P2000194154trial_trial Targeted space dev API endpoint: https://api.cf.eu10.hana.ondemand.com (API version: 2.101.0) User: sipemsbh@sharklasers.com Org: P2000194154trial_trial Space: dev bash-3.2\$</pre>

Explanation	Screenshot
<p>Now all you have to do is push your app to the SAP Cloud Platform Cloud Foundry by using the command: <code>cf push</code></p>	 <pre>bash-3.2\$ cf push Using manifest file /Users/Ralph/OneDrive - SAP SE/handson/FreightCalculator/manifest.yml Creating app freightcalcRO in org P2000194154trial_trial / space dev as sipemsbh@sharklasers.com... OK Creating route freightcalcro.cfapps.eu10.hana.ondemand.com... OK Binding freightcalcro.cfapps.eu10.hana.ondemand.com to freightcalcRO... OK Uploading freightcalcRO... Uploading app files from: /Users/Ralph/OneDrive - SAP SE/handson/FreightCalculator Uploading 5.3K, 11 files Done uploading OK Starting app freightcalcRO in org P2000194154trial_trial / space dev as sipemsbh@sharklasers.com...</pre>

Explanation	Screenshot
<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 state since cpu memory disk details #0 running 2018-03-01 11:54:39 AM 0.0% 44.7M of 64M 55.3M of 1G bash-3.2\$</pre>

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.

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://goshippo.com/>

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



Explanation	Screenshot																				
<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>	<p>Terminal — bash</p> <pre>[bash-3.2\$ cf set-env freightcalcro SHIPPO_KEY shippo_test_19385705180736e352abedb642306bf;2A] Setting env variable 'SHIPPO_KEY' to 'shippo_test_19385705180736e352abedb642306bf' for app freightcalcRO in org P2000194154trial_trial / space dev as sipemsbh@sharklasers.com... OK TIP: Use 'cf restage freightcalcRO' to ensure your env variable changes take effect bash: 2A: command not found bash-3.2\$]</pre>																				
<p>Up next, restart your app</p> <pre>cf restart <appname></pre>	<p>Terminal — bash</p> <pre>[bash-3.2\$ cf restart freightcalcRO Restarting app freightcalcRO in org P2000194154trial_trial... 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.ondemand.com... last uploaded: Thu 01 Mar 11:54:08 GMT 2018 stack: cflinuxfs2 buildpack: https://github.com/cloudfoundry/nodejs start command: npm start</pre>																				
<p>You can test the app with a test page on the app URL</p>	<p>Ship Item To Zip Code: 01318001 Country: BR 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>DHL Express</td><td>Worldwide</td><td>4</td><td>USD 94.24</td></tr><tr><td>2</td><td>UNITED STATES POSTAL SERVICE</td><td>Priority Mail International</td><td>8</td><td>USD 54.82</td></tr><tr><td>3</td><td>UNITED STATES POSTAL SERVICE</td><td>Priority Mail Express International</td><td>4</td><td>USD 77.43</td></tr></tbody></table>	#	Provider	Service	Est. Days	Amount	1	DHL Express	Worldwide	4	USD 94.24	2	UNITED STATES POSTAL SERVICE	Priority Mail International	8	USD 54.82	3	UNITED STATES POSTAL SERVICE	Priority Mail Express International	4	USD 77.43
#	Provider	Service	Est. Days	Amount																	
1	DHL Express	Worldwide	4	USD 94.24																	
2	UNITED STATES POSTAL SERVICE	Priority Mail International	8	USD 54.82																	
3	UNITED STATES POSTAL SERVICE	Priority Mail Express International	4	USD 77.43																	

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

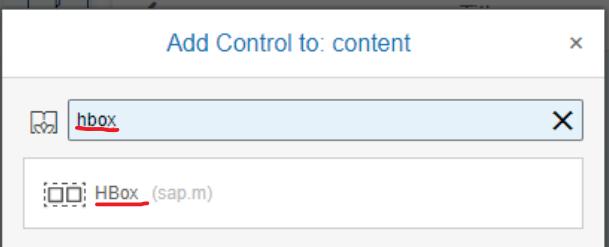
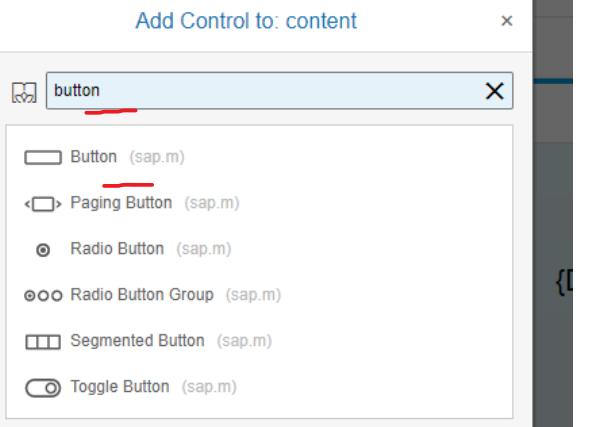
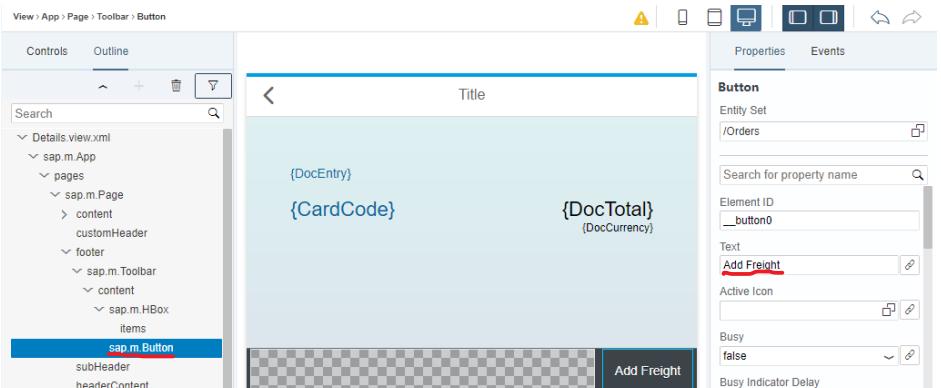


STEP 4: INTEGRATE THE NODEJS APP INTO THE SAP FIORI APP

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 title bar says 'Details.view.xml'. The top navigation bar has 'View > App > Page'. Below it, there are two tabs: 'Controls' and 'Outline', with 'Outline' being the active tab. On the left, there's a tree view of the XML structure: 'Details.view.xml' > 'sap.m.App' > 'pages' > 'sap.m.Page' > 'content' > 'customHeader' > 'footer'. A context menu is open at the 'footer' node, with 'Add' highlighted in red. Other options in the menu include 'Convert to Fragment', 'Add Fragment', 'Cut (Ctrl+X)', 'Copy (Ctrl+C)', 'Paste (Ctrl+V)', 'Paste Before', 'Paste After', 'Delete', and 'Delete' again.</p>
Select the Toolbar control proposed.	<p>The screenshot shows the SAP Fiori Layout Editor interface with the 'Controls' tab selected. A dialog box titled 'Add Control to: footer' is open. Inside the dialog, there's a search bar and a list of controls. The 'Toolbar (sap.m)' control is listed and highlighted with a red underline.</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>	

i. 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 data-bbox="520 1221 1108 1410"> /** * @memberOf sa.B1SL_SUMMIT_2018.controller.Details */ onAddFreightPress: function(oEvent) { //This code was generated by the layout editor. } </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/B1SA/B1 SCP HandsOn/blob/master/snippets/Details.controller.js as</p>	<pre data-bbox="498 1513 1308 1603"> 1 = sap.ui.define(["sap/ui/core/mvc/Controller"], function(Controller) { 2 "use strict"; } </pre> <pre data-bbox="498 1671 1387 1963"> 1 = sap.ui.define([2 "sap/ui/core/mvc/Controller", 3 "sap/m/MessageToast", 4 "sap/ui/core/Fragment", 5 "sap/ui/model/Filter", 6 "sap/ui/model/json/JSONModel", 7 "jquery.sap.global" 8], 9 function(Controller, MessageToast, Fragment, Filter, JSONModel, jQuery) { 10 "use strict"; } </pre>



Explanation	Screenshot
shown in the second screen.	
<p>Let now implement the function onAddFreightPress.</p> <p>When the user clicks on the AddFreight button we call the server side NodeJS to get freight options.</p> <p>Replace the URL to the freight calculator by your backend application URL (code marked in red in the screen capture).</p> <p>In case of success we call the function <code>openFreightDialog</code>.</p> <p>In case of error we simply show a <code>MessageToast</code>.</p> <p>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.</p>	<pre>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); } }); },</pre>
<p>On the <code>Details.controller.js</code> scroll and search for the function openFreightDialog. This function opens a new Dialog showing the freight options returned by the server side.</p> <p>This function will use an xml fragment to design the freight options dialog.</p> <p>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 <code>Details.controller.js</code> file. In my case it is</p>	<pre>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 }</pre>



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>	

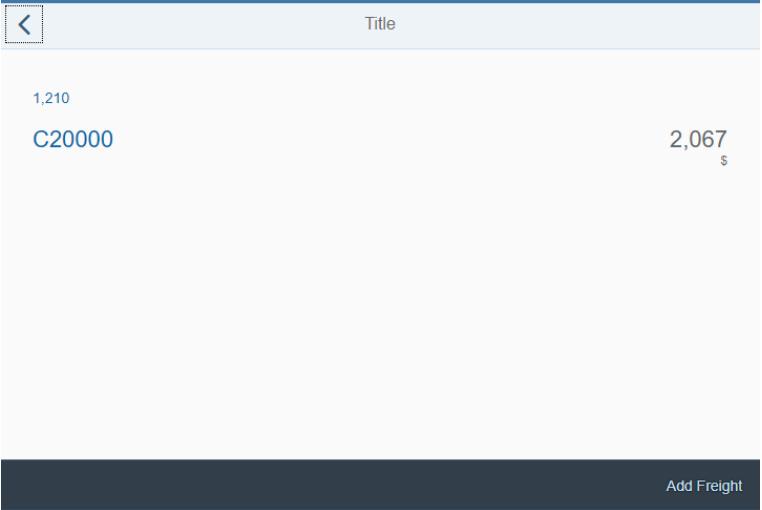
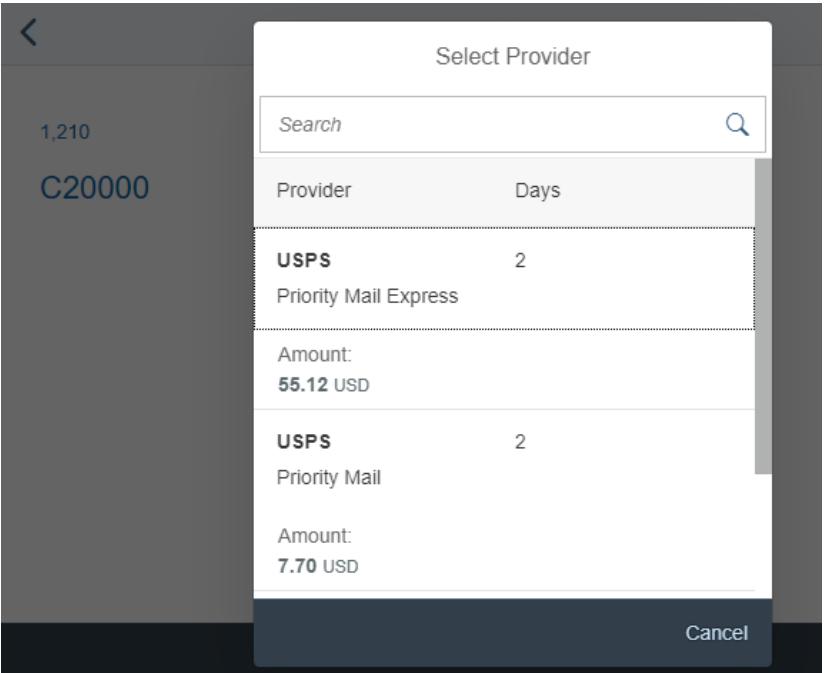


Explanation	Screenshot
<p>Open the Details.controller.js file.</p>	
<p>The handleFreightDialogSearch (code copied already earlier) is called when the user enters data to filter the list of freight options. The filter will be based on the provider property.</p>	<pre>handleFreightDialogSearch: function(oEvent) { 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]); },</pre>
<p>The handleFreightDialogConfirm function is called when the user selects one of the freight options proposed by the Dialog. This function calls the function <code>UpdateSO</code> to update the Order via Service Layer API.</p>	<pre>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 + "."); },</pre>
<p>The handleFreightDialogClose function is called when the user closes the Dialog without selecting a freight option. A <code>MessageToast</code> will show to the user the message "No Provider selected."</p>	



Explanation	Screenshot															
<p>The <code>updateSO</code> function calls SAP Business One Service Layer via the SAP API Business Hub sandbox.</p> <p>We do an ajax call to the destination <code>/apihub_sandbox/</code> 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.</p> <p>In case of success we show a MessageToast message "SL success".</p> <p>In case of error we show a MessageToast message "SL error".</p>	<pre>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 { 229 "ExpenseCode": 1, 230 "LineTotal": providerDetails.amount, 231 "Remarks": providerDetails.name 232 } 233 }; 234 235 \$.ajax({ 236 url: "/apihub_sandbox/sapb1/b1s/v2/Orders(" + orderDetails.DocEntry + ")", 237 type: "PATCH", 238 data: JSON.stringify(body), 239 contentType: "application/json", 240 success: function(data) { 241 oThis.soUpdateSuccess(data); 242 }, 243 error: function(e) { 244 oThis.soUpdateError(e); 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 });</pre>															
Test your application.	<p>Run index.html</p>															
<p>Filter the Orders table with DocEntry higher than 1211.</p> <p>Please follow directions from instructors so you only update a specific Order DocEntry to avoid conflicts between hands-on participants.</p>	<table border="1"><thead><tr><th colspan="3">Title</th></tr><tr><th>DocEntry</th><th>CardCode</th><th>DocTotal</th></tr></thead><tbody><tr><td>1,211</td><td>C20000</td><td>2,067\$</td></tr><tr><td>1,213</td><td>C20000</td><td>2,067\$</td></tr><tr><td>1,215</td><td>C20000</td><td>2,067\$</td></tr></tbody></table>	Title			DocEntry	CardCode	DocTotal	1,211	C20000	2,067\$	1,213	C20000	2,067\$	1,215	C20000	2,067\$
Title																
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.	 <p>The screenshot shows a mobile application interface. At the top, there is a navigation bar with a back arrow icon and the word "Title". Below this, the order details are displayed: "1,210" (likely quantity), "C20000" (order number), and "2,067 \$" (total amount). A dark blue button at the bottom right is labeled "Add Freight".</p>
Select one of the providers.	 <p>The screenshot shows a modal dialog titled "Select Provider". It contains a search bar with a magnifying glass icon. Below the search bar, there are two sections for "USPS": the first section shows "Priority Mail Express" with "2" days and an amount of "55.12 USD"; the second section shows "Priority Mail" with "2" days and an amount of "7.70 USD". At the bottom of the modal is a "Cancel" button.</p>



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>	<p>The screenshot shows the SAP Business One Order screen. At the top, there is a navigation bar with a back arrow and a title field. Below the title, the document number 'C20000' is displayed. To the right of the document number, the 'DocTotal' value '1,210' is shown above the 'Add Freight' button. Further to the right, the total amount '2,074.7' is displayed with a '\$' sign below it. A success message 'Order updated successfully!' is visible at the bottom of the screen.</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.

