SCP3C03 – Create an App Using the SAP Gateway System

|  |  |
| --- | --- |
| **Product**  SAP oud Platform  **Level**  Undergraduate/Graduate  Beginner  **Focus**  Application Development  **Author** Ross Hightower | MOTIVATION  This case illustrates many of the features and workflows required to develop applications on the SAP Cloud Platform  **PREREQUISITES**  SCP3C01 |
|  | |

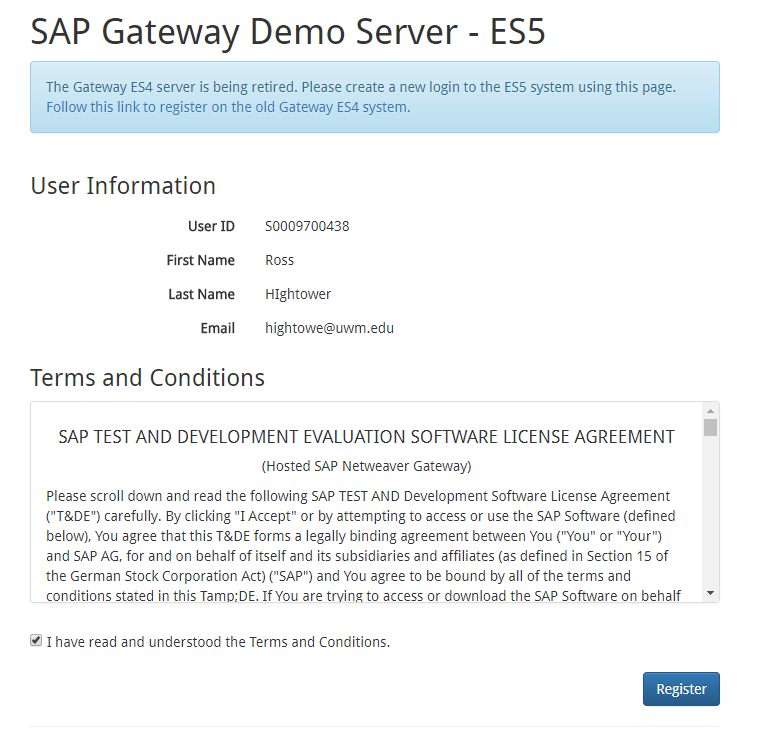
This case illustrates many aspects of developing applications on the SAP Cloud Platform including:

* Creating and using destinations to an SAP Gateway system
* Storing code in the git repository
* Using fragments to modularize code
* Using a Google map

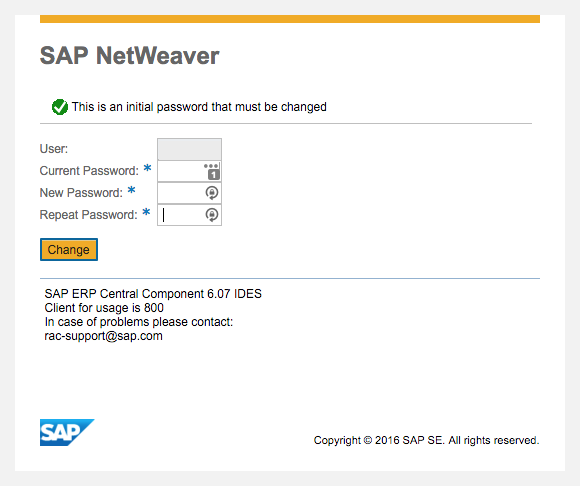
## Create an Account on the SAP Gateway Demo System

Navigate to <https://register.sapdevcenter.com/SUPSignForms> and log on using your SAP credentials.

Agree to the terms and conditions and click Register.



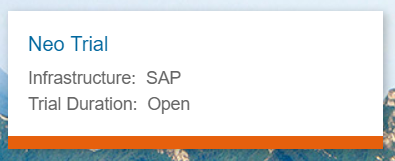
When you receive the email confirming your registration, click on the link to log on to the SAP Gateway WebGUI so you can create a password.



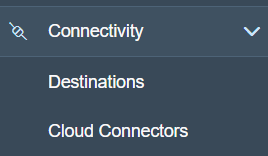
You now have an account on the SAP Gateway Demo System.

## Create a Destination in the SAP Cloud Platform

In order to access services on remote systems from the SAP Cloud Platform, you must create a Destination in the SAP Cloud Platform Cockpit. Log onto the SAP Cloud Platform ([https://account.hanatrial.ondemand.com](https://account.hanatrial.ondemand.com/)). Click New Trial.



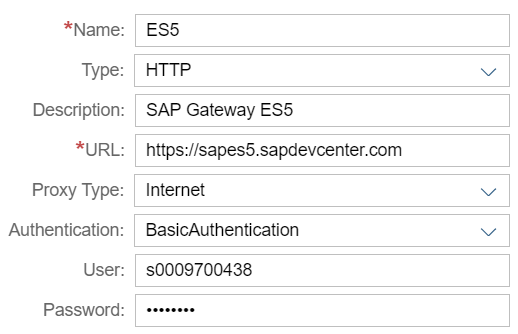
Navigate to Connectivity → Destinations using the menu on the left.



Click New Destination and then enter the details shown below:

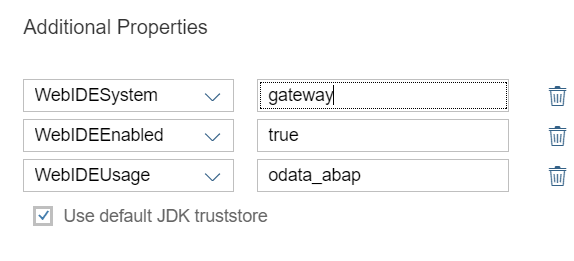


|  |  |
| --- | --- |
| Name | SAP Gateway |
| Type | HTTP |
| Description | SAP Gateway ES5 |
| URL | https://sapes5.sapdevcenter.com |
| Proxy Type | Internet |
| Authentication | BasicAuthentication |
| User | Your SAP Gateway User ID |
| Password | Your SAP Gateway Password |

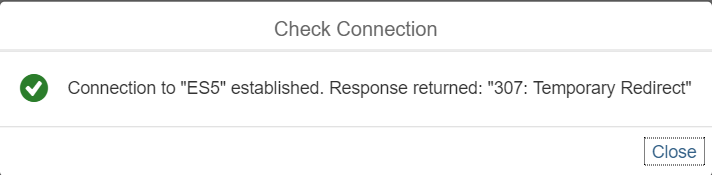


**Before saving** use the New Property button to add the following three properties:

|  |  |
| --- | --- |
| WebIDEEnabled | True |
| WebIDESystem | Gateway |
| WebIDEUsage | odata\_abap |



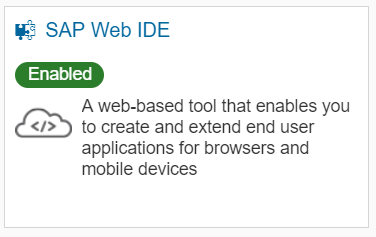
Once you save the destination, you can use the Check Connection button to make sure the URL can be reached.



The results are sometimes ambiguous. The only result that is definately a problem is a 404 – Not Found error.

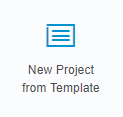
## Create an SAPUI5 App Using the Destination

Use the menu on the left to navigate to Services. Either search for or scroll down until you find the SAP Web IDE tile (if you’ve bookmarked the link to the Web IDE you can go directly there).

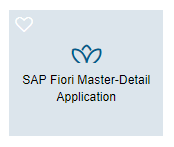


Click the tile and then click Go to Service to open the Web IDE.

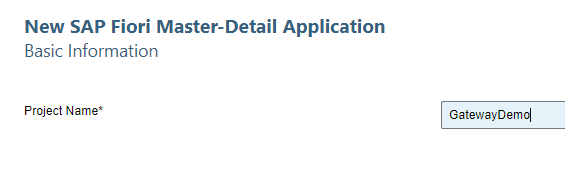
Either click the New Project from Template icon or use File → Project From Template to create a new project.



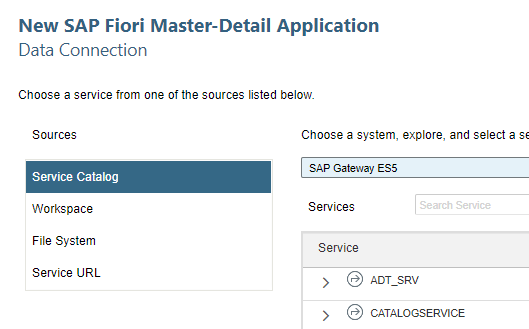
Select the Fiori Master-Detail Application template. You may have to use the fields above the template tiles to search for the correct template. Click Next.



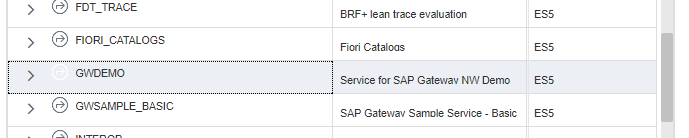
Name the project and click Next.



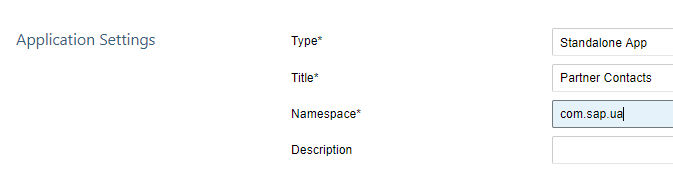
Select the Destination you created above from the dropdown list.



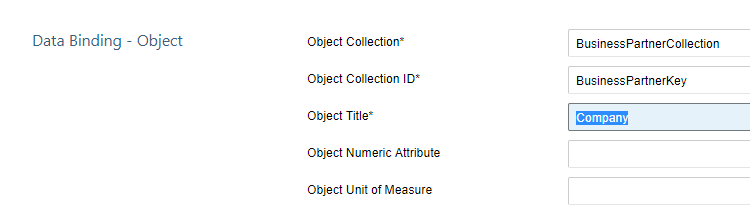
Scroll down and select the GWDEMO oData Service. Click Next.



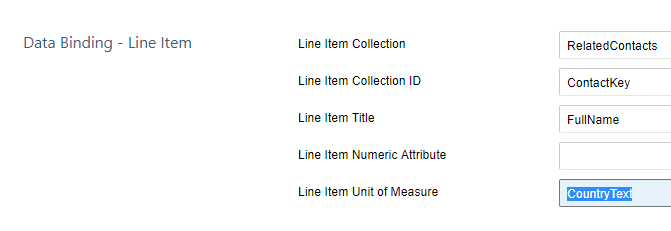
Enter the following Application Settings:



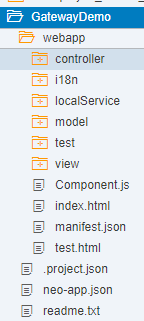
Enter the following for the Data Binding – Object:



Enter the following for the Data Binding – Line Item:



Click Finish to create the Project.

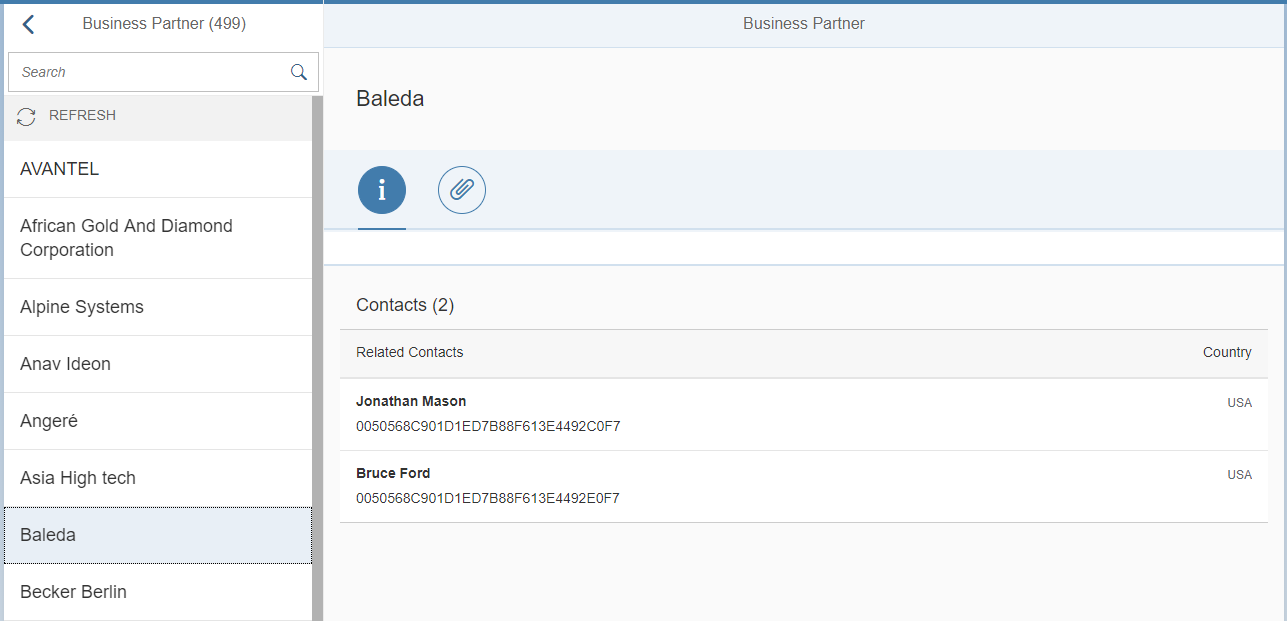


## Update the Application Texts

To facilitate internationalization of application texts, text values are stored separately in properties files. Open the i18n.properties file located in the project’s webapp/i18n folder. Make the following substitutions (you can use ctrl+H to open the find and replace dialog):

|  |  |
| --- | --- |
| <RelatedContactsPlural> | Contacts |
| <BusinessPartnerCollectionPlural> | Business Partners |
| <BusinessPartnerCollection> | Business Partner |
| <FirstColumnName> | Related Contacts |
| <LastColumnName> | Country |

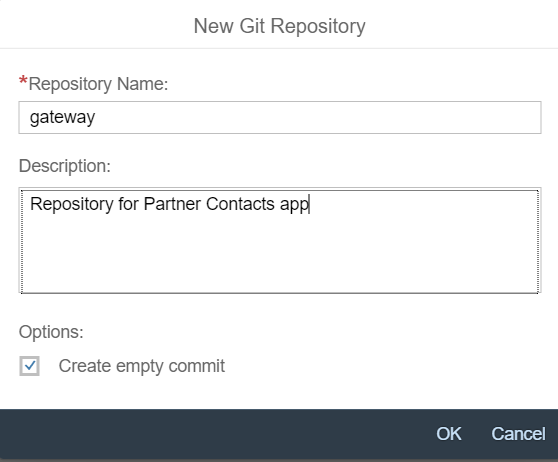
Run your app by selecting the project folder and clicking .

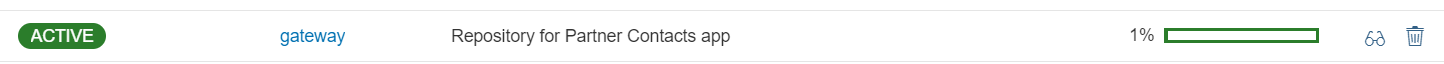


## Commit the Project to a Git Repository

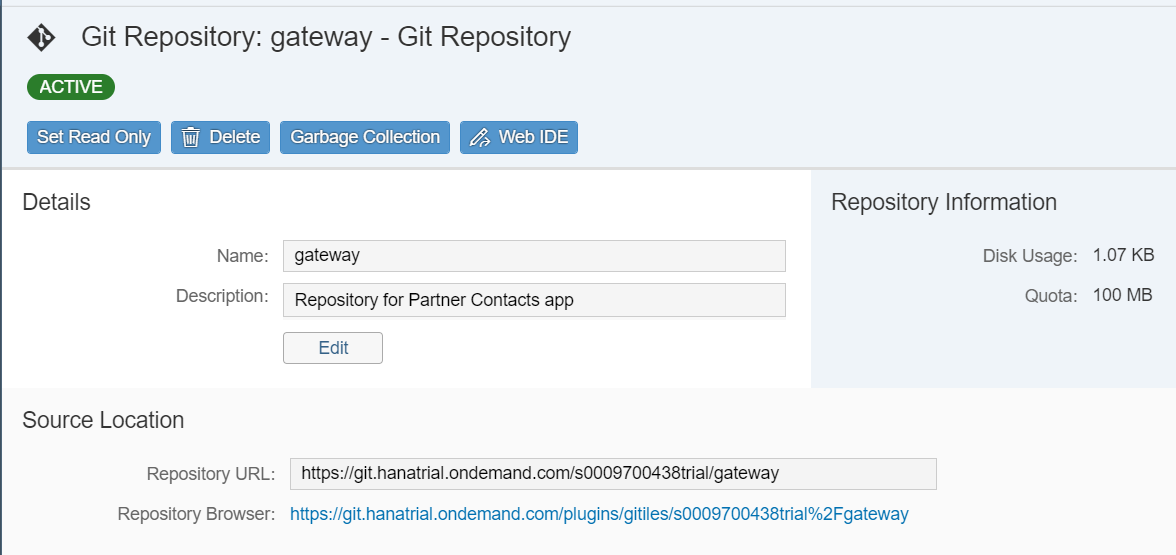
Git has become one of the more popular code management systems and is integrated in the SAP Cloud Platform. In this section, you will commit your project code to a git repository.

Log onto the SAP Cloud Cockpit and select Repositories → Git Repositories on the menu on the left. Click New Repository and enter a name and description as shown below. Make sure Create empty commit is checked and click OK.

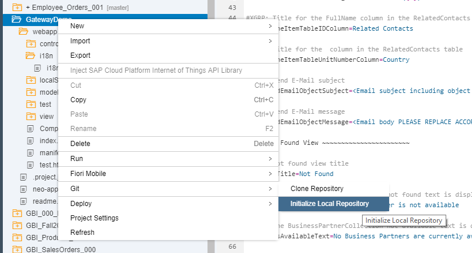




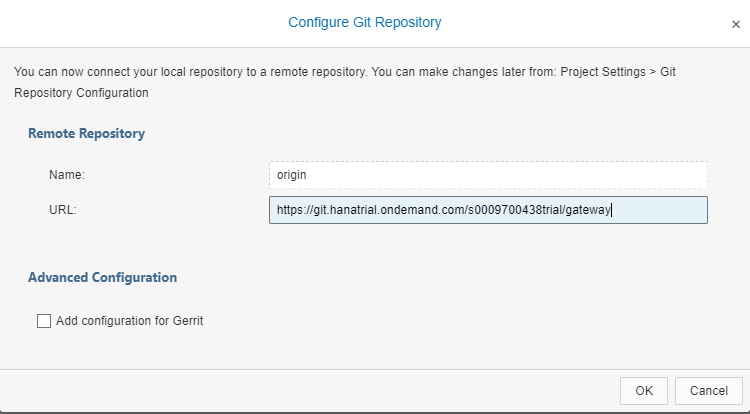
Click the repository link to open the repository Overview page.



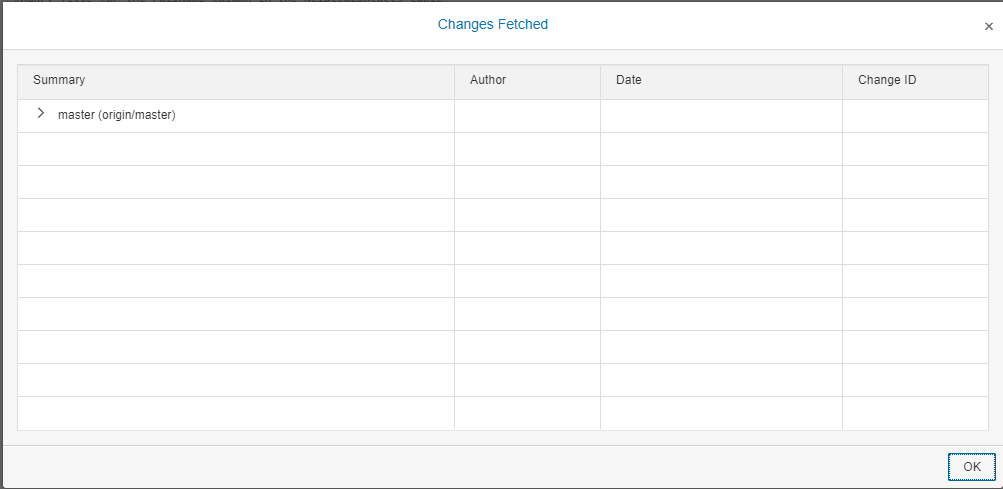
Copy the Git Repository URL. In the Web IDE, right-click your Gateway project folder and select Git → Initialize Local Repository.



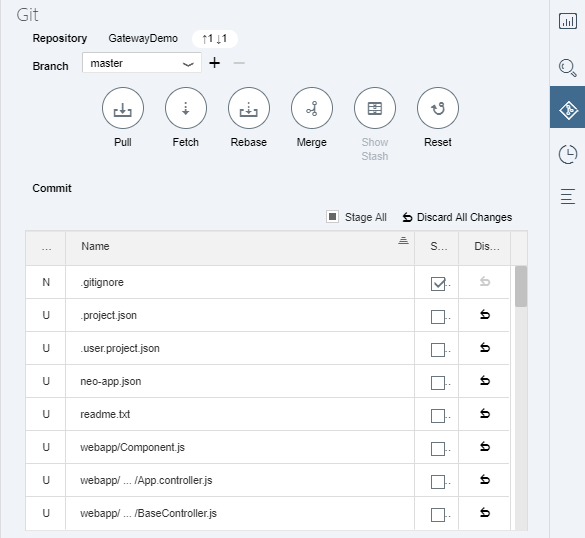
An alert box will appear in the upper right side of the Web IDE. Click the link Set Remote. Paste your URL into the URL field.



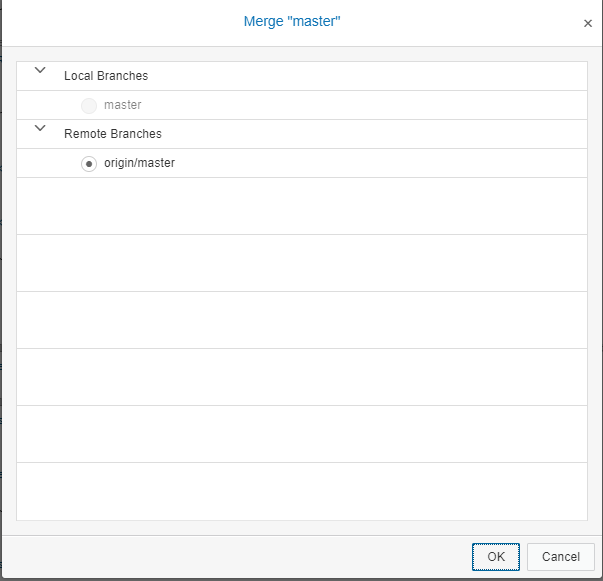
Click OK.



Click OK and then click the Git pane icon on the right side of the editor . If your project doesn’t appear, click on the project folder on the left side of the editor.

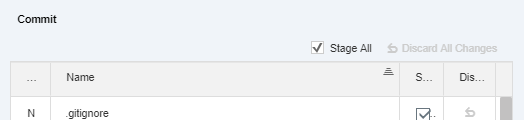


The table shows all the files that have been changed since the last time you pushed them to the remote repository. To connect the local (in the Web IDE) with the remote (Cockpit) repositories, click the Merge icon. When the **Merge “master”** dialog box opens, make sure the **origin/master** remote branch is selected and click **OK**.

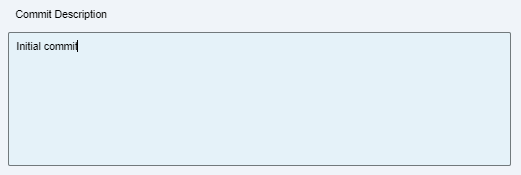


Now you can commit your project files to the remote repository. Check the Stage All

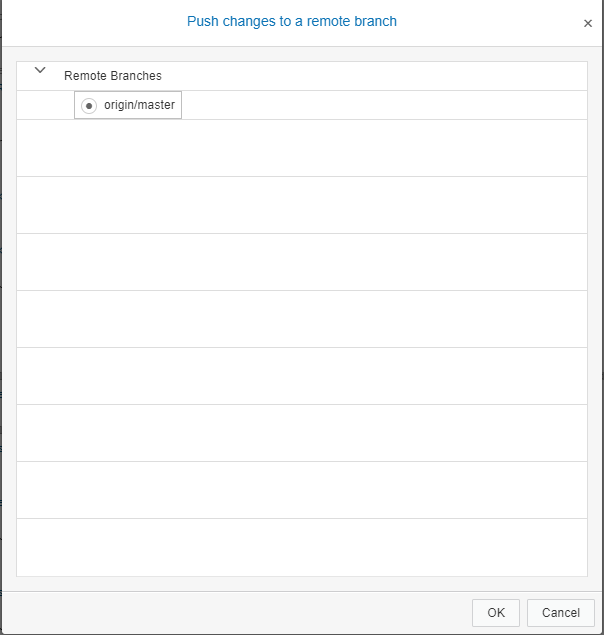
checkbox in the Git pane.



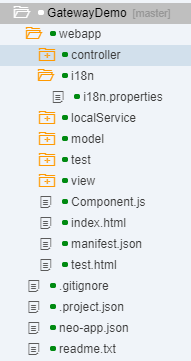
Enter a description in the Commit Description field.



Click Commit and Push and select the Remote Branch from the drop down. If you get this dialog, click origin/master and click OK.

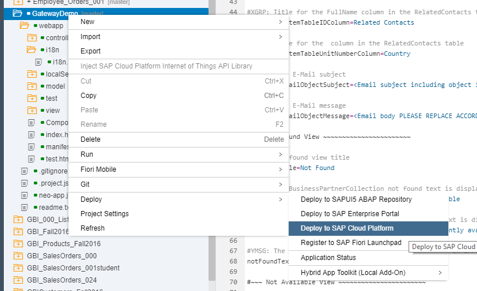


This step “pushed” the files from the local repository to the remote repository. Once all the files are safely stored in the remote repository you will see green dots next to the files in the project explorer.

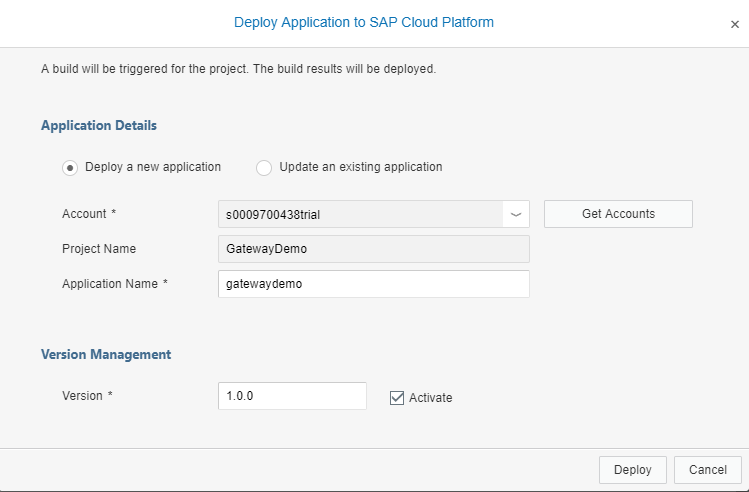


## Deploy the App to the SAP Cloud Platform

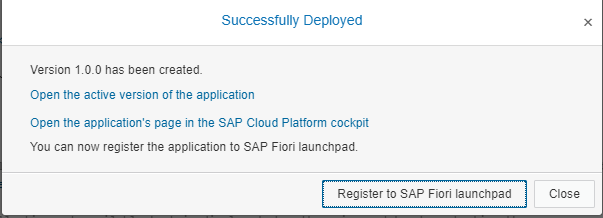
In the Web IDE, right-click the project folder and select Deploy → Deploy to SAP Cloud Platform.



If prompted, login with your SAP Cloud Platform id and password. You can change the name in the dialog that appears. Make sure Deploy a new application and Activate are both checked and click Deploy.



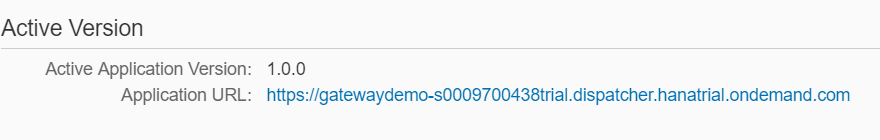
Click Open the active version of the application in the dialog that appears.



Now that it is active, you can use the URL to open the app in a mobile device or desktop browser. If you return the SAP Cloud Cockpit and navigate to Applications → HTML5 Applications, you will see the deployed application.



Click the name to find details of the application including the URL to reach it.



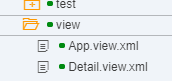
It’s a long URL so you could email it to yourself to open it on your phone.

## Add Some Additional Details

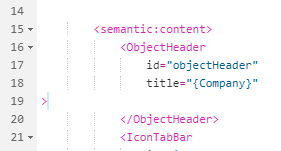
In this section, we’ll add some detail to the Detail view.

### Add fields to the Detail Header

Open the Detail.view.xml file located in the view folder of your project.



Insert the code shown below into the <ObjectHeader> control. In the image below, you would insert the code between lines 19 and 20. Note the project generating wizard placed the > on line 19 which should be at the end of line 18.



|  |
| --- |
| <ObjectAttribute title="{i18n>headerCity}" text="{Address/City}"/>  <ObjectAttribute title="{i18n>headerCountry}" text="{Address/CountryText}"/>  <ObjectAttribute title="{i18n>headerURL}" text="{WebAddress}"/>  <ObjectAttribute title="{i18n>headerBusinessPartnerRole}" text="{BusinessPartnerRoleText}"/> |

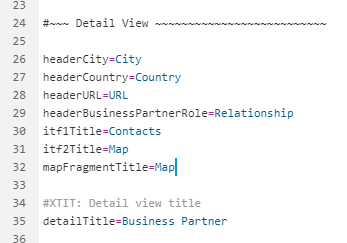
Listing 1

Once you paste the code in, you can format the code by right-clicking in the editor and selecting Beautify.

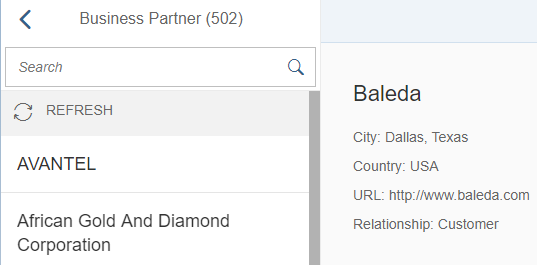
Open the i18n.properties file and insert the following code. You can add it anywhere but it pays to be organized.

|  |
| --- |
| headerCity=City  headerCountry=Country  headerURL=URL  headerBusinessPartnerRole=Relationship  itf1Title=Contacts  itf2Title=Map  mapFragmentTitle=Map |

Listing 2

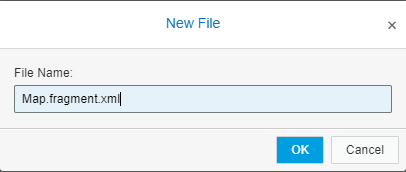


If you run the app now, you will see the details:



### Add an XML Fragment

Right-click the view folder in your project and select New → File. Name the file Map.fragment.xml.



Insert the following code into the new file:

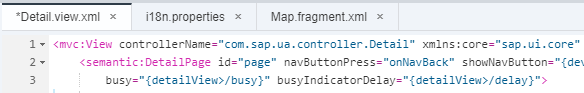
|  |
| --- |
| <core:FragmentDefinition xmlns:core="sap.ui.core" xmlns:f="sap.ui.layout.form" xmlns:l="sap.ui.layout" xmlns="sap.m">  <l:Grid defaultSpan="L12 M12 S12" id="contactFragment" width="auto">  <l:content>  <f:SimpleForm columnsL="6" columnsM="1" editable="false"  layout="ResponsiveGridLayout" maxContainerCols="6" minWidth="1024" title="{i18n>mapFragmentTitle}">  <f:content>  <Label text="Address "/>  <Text text="{Address/Building}"/>  <Text text="{Address/Street}"/>  <Text text="{Address/City}"/>  <Text text="{Address/PostalCode}"/>  <Text text="{Address/CountryText}"/>  </f:content>  </f:SimpleForm>  </l:content>  </l:Grid>  </core:FragmentDefinition> |

Listing 3

Open the Detail.view.xml file and add the following namespace (code library) to the <mvc:View> control at the top of the file. The <FragmentDefintion> control is located in this library.

|  |
| --- |
| xmlns:core="sap.ui.core" |

Listing 4



Scroll down in the Detail.view.xml file to find the <IconTabFilter> elements and update them with the following code:

First Element

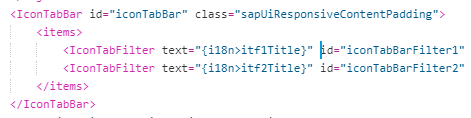
|  |
| --- |
| text="{i18n>itf1Title}" |

Listing 5

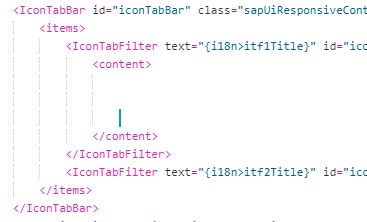
Second Element

|  |
| --- |
| text="{i18n>itf2Title}" |

Listing 6



Now insert the code <content></content> between the first <IconTabFilter> and </IconTabFilter> tags.



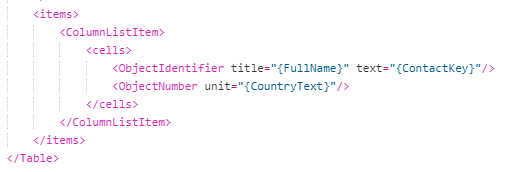
Cut the Table code as shown below and paste it into the <content> element you just created. This moves the table into the tab.



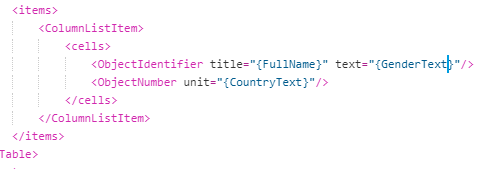
It should like the image below. If you make a mistake, ztrl-z will undo changes.



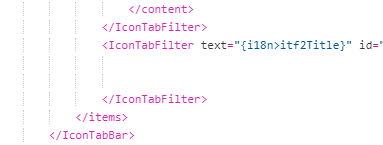
The ContactKey is not very informative so locate that field…



…and replace it with GenderText.



Now insert the code shown below between the second <IconTabFilter> and </IconTabFilter> tags. This code loads the fragment you created above.

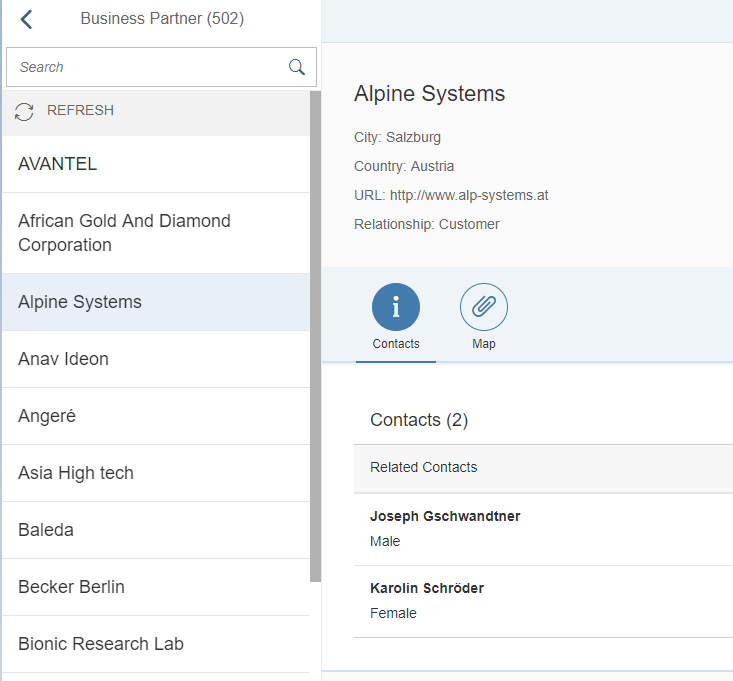


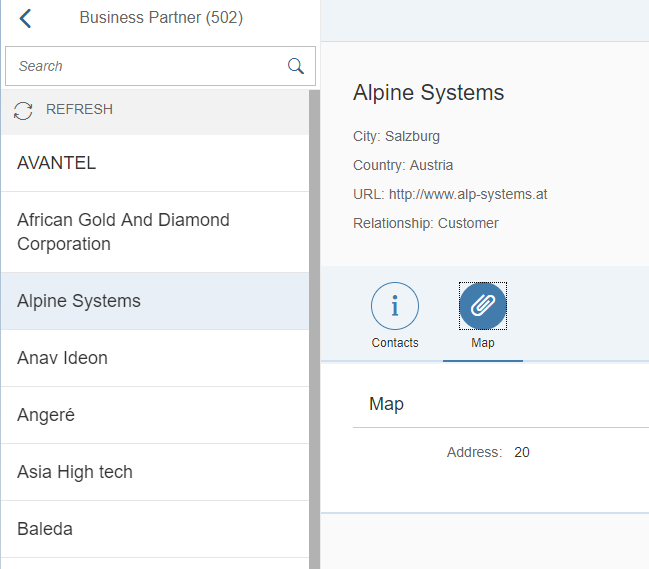
|  |
| --- |
| <content>  <core:Fragment fragmentName="com.sap.ua.view.Map " type="XML" />  </content> |

Listing 8



Now if you run the app, you can switch between views using the tabs.





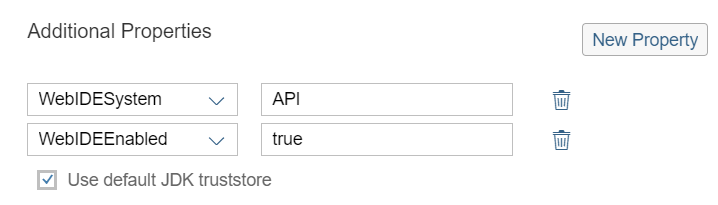
### Add a Google Map

The Google Map API makes it easy to add maps to applications. You can read about the api here <https://developers.google.com/maps/documentation/static-maps/intro>

The first thing you have to do is add a destination for the Google Map service. In the SAP Cloud Cockpit, navigate to Connectivity → Destinations and click New Destination. Enter the following:

|  |  |
| --- | --- |
| Name | GoogleMaps |
| Type | HTTP |
| Description | Google Maps API |
| URL | https://maps.googleapis.com/maps/api |
| Proxy Type | Internet |
| Authentication | NoAuthentication |

Before clicking save, use the New Property button to enter the following properties



Now we have to add a reference to the destination in the application. Open the neo-app.json file and paste this at the bottom of the routes section.

|  |
| --- |
| ,  {  "path": "/GoogleMaps",  "target": {  "type": "destination",  "name": "GoogleMaps",  "entryPath": "/"  },  "description": "Google Maps API"  } |

Listing 9

Use the image to make sure you are putting it in the right place.



Now, to insert the map, delete the existing <SimpleForm> element from the Map.fragment.xml.



Replace it with this code.

|  |
| --- |
| <f:SimpleForm columnsL="2" columnsM="2" editable="false" layout="ResponsiveGridLayout" maxContainerCols="2" minWidth="1024"  title="{i18n>mapFragmentTitle}">  <f:content>  <core:Title text="{i18n>mapAddressTitle} "/>  <Label text="{i18n>mapStreet}"/>  <Text text="{Address/Building} {Address/Street}"/>  <Label text="{i18n>mapCity}"/>  <Text text="{Address/City}"/>  <Label text="{i18n>mapCountry}"/>  <Text text="{Address/CountryText}"/>  <Label text="{i18n>mapZip}"/>  <Text text="{Address/PostalCode}"/>  <core:Title text=" "/>  <Image src="{ parts: [ 'Address/Building', 'Address/Street', 'Address/PostalCode', 'Address/City' ], formatter: '.formatter.formatMapUrl' }"  alt="{i18n>mapAltText}"/>  </f:content>  </f:SimpleForm> |

Listing 10



Add the following code to the i18n.properties file.

|  |
| --- |
| #~~~ Map Fragment ~~~~~~~~~~~~~~~~~~~~~~~~~~  mapAddressTitle=Address  mapStreet=Street  mapCity=City  mapCountry=Country  mapZip=Zipcode  mapAltText=Map of location |

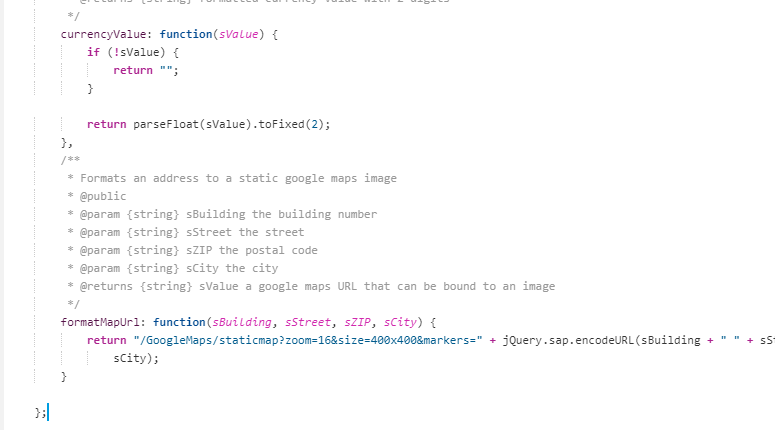
Listing 11



We have to compute the URL for the map each time the user clicks a new contact. We can do that with a formatter. These are used to format values before they are inserted into the HTML. Open the formatter.js file in the model folder and insert this code.

|  |
| --- |
| /\*\*  \* Formats an address to a static google maps image  \* @public  \* @param {string} sBuilding the building number  \* @param {string} sStreet the street  \* @param {string} sZIP the postal code  \* @param {string} sCity the city  \* @returns {string} sValue a google maps URL that can be bound to an image  \*/  formatMapUrl: function(sBuilding, sStreet, sZIP, sCity) {  return "/GoogleMaps/staticmap?zoom=16&size=400x400&markers="  + jQuery.sap.encodeURL(sBuilding +" "+ sStreet + ", " + sZIP + " " + sCity );  } |

Listing 12



Run the app and check the Map tab.

