**Modification-free extensibility for existing apps**

Exercises

PUBLIC



Table of ContentS

[Prerequisites: Access the existing app 4](#_Toc11685954)

[Adding a button to a smart table 15](#_Toc11685955)

[Adding a Dimension Filter 22](#_Toc11685956)

[Adding a map to show the supplier location 23](#_Toc11685957)

[Related material 24](#_Toc11685958)

### Summary

You will learn

* How to use adaption projects to create app variants.
* What to keep in mind when you disable the safe mode.
* How to create fragments and controllers.

## Prerequisites: Access the existing app

### Summary

UI5 Air Cargo has bought the [Manage Products](https://sapes5.sapdevcenter.com/sap/bc/ui5_ui5/ui2/ushell/shells/abap/FioriLaunchpad.html#EPMProduct-manage) application that resides in the ES5 SAP Gateway demo system. This application allows users to search for products and look at the details of those products. This application was created using the SAP Fiori elements [list report](https://sapui5.hana.ondemand.com/#/topic/1cf5c7f5b81c4cb3ba98fd14314d4504) and [object page](https://sapui5.hana.ondemand.com/#/topic/645e27ae85d54c8cbc3f6722184a24a1).

### Preview

In this exercise you will configure a trial account on SAP Cloud Platform, SAP Web IDE and you will get a user to access the existing app and our configure the necessary steps in SAP Cloud Cockpit.

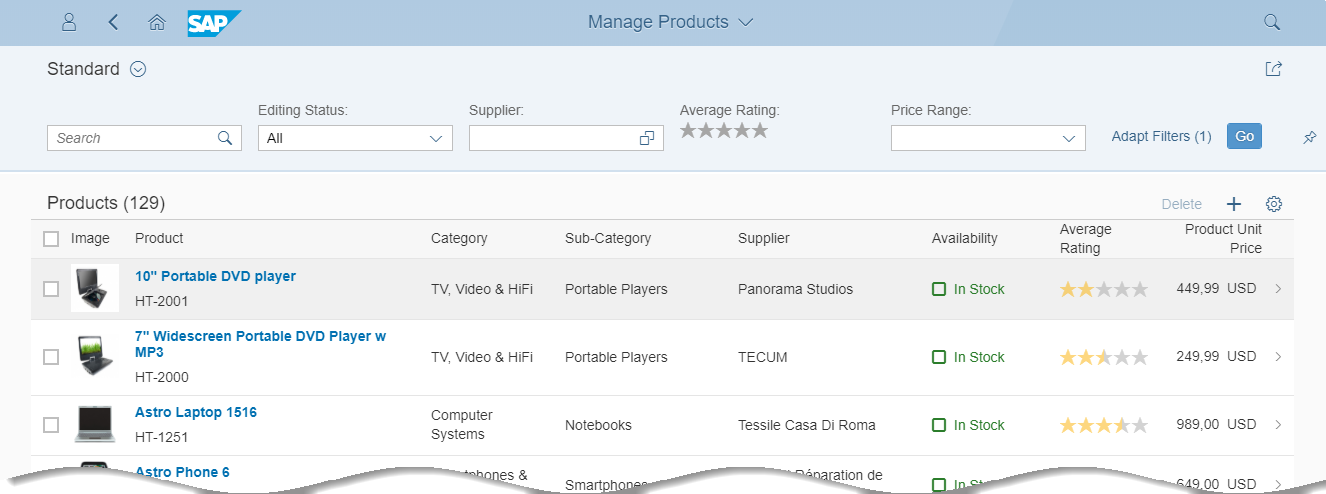


Figure 1 – Manage Products app

### Create an SAP Cloud Platform Developer Account

| Explanation | **Screenshot** |
| --- | --- |
| 1. Open your browser, and open the following URL:   <https://account.hanatrial.ondemand.com/>  **Note: Use Google Chrome** We recommend that you use Google Chrome as a browser for this course, as it has the best developer tools. Our instructions and browser-related tips throughout this course will be refer to Google Chrome.  If you don’t have it installed yet, go to the following URL, and follow the setup steps: [**https://www.google.de/chrome/browser/desktop/**](https://www.google.de/chrome/browser/desktop/) |  |
| 1. Choose *Register* to create a free SAP Cloud Platform developer account. |  |
| 1. Enter your name and e-mail address, and choose a password (at least 8 characters long, including upper- and lowercase letters, numbers and symbols). 2. Accept the terms & conditions, and click *Register*.   **Note: Trial Account** This will create a developer account on SAP Cloud Platform for you with a free and perpetual [developer license](https://accounts.sap.com/ui/public/viewTextResource?scenario=SAP_HANA_Cloud_Developer_Edition&resourceType=RESOURCE_TERMS_OF_USE&locale=en_US&spDisplayName=SAP%20HANA%20Cloud%20Developer%20Edition). You can use this account to test out the platform or run demo scenarios. |  |
| 1. A confirmation e-mail has been sent to your account. Open it, and choose *Click here to activate your account*. |  |
| 1. You should see a confirmation dialog. Choose *Continue* to go to the SAP Cloud Platform Cockpit. |  |
| 1. Choose *Neo Trial*. |  |
| 1. In the Neo Trial environment, you can see a navigation frame on the left. This is your entry point for managing cloud apps and configurations.   **Note: How to keep track of your user**  Two sets of users and passwords are created in this unit:   * For the SAP Cloud Platform Developer Account (what we just did) * For the SAP Gateway Demo System ES5 (later in this unit)   If you are in doubt what your user is do this:   * SAP Cloud Platform Developer Account: On the top right side of the Cloud Platform Cockpit you can display your user. * SAP Gateway Demo System ES5: You’ll receive an e-mail with the temporary password. This you have to change immediately, of course, but the user will remain the same. |  |

### Navigate to Your SAP Web IDE Workspace

In this step, you’ll navigate to your workspace in SAP Web IDE.

| Explanation | **Screenshot** |
| --- | --- |
| 1. On SAP Cloud Platform Home Page, you can directly navigate to the *SAP Web IDE*. |  |
| 1. SAP Web IDE opens. Have a look at the start page. It contains useful links and shortcuts to the most important actions. Choose *Open my workspace*.   The direct link to SAP Web IDE is https://webidecp-p[XXXXXXXXXX]trial.dispatcher.hanatrial.ondemand.com/  **Note:** Replace [XXXXXXXXXX] with your trial account user ID. If you are using an existing trial account, you will see a workspace with your other projects in this step. |  |
| 1. (Optional) Add a bookmark to SAP Web IDE Full-Stack by pressing *CTRL+D* in Google Chrome. You will need this link to your development environment frequently throughout the course, so you should bookmark the page. |  |

### Sign Up and Get Started with the SAP Gateway Demo Consumption System

| Explanation | **Screenshot** |
| --- | --- |
| 1. Start the SAP Gateway demo system with this URL https://register.sapdevcenter.com/SUPSignForms/ 2. If you see the log on screen, log on to your SAP Cloud Platform account. If you are already logged in you’ll see the SAP Gateway Demo Server welcome page. 3. Read the terms and conditions, and choose *Register*.   **Note: How to keep track of your user**  Two sets of users and passwords are created in this unit:   * For the SAP Cloud Platform Developer Account (what we did as first step) * For the SAP Gateway Demo System ES5 (what we’re doing here)   If you are in doubt what your user is do this:   * SAP Cloud Platform Developer Account: On the top right side of the Cloud Platform Cockpit you can display your user (see above). * SAP Gateway Demo System ES5: You received this e-mail with the temporary password. This you have to change immediately, of course, but the user will remain the same. |  |
| 1. If an upgrade for your trial account is necessary, you will see an additional step for this during your registration process. Read and acknowledge SAP’s privacy statement, and choose *Register*. |  |
| 1. You will see a success messagefor your registration. 2. Go to your e-mail inbox to find the confirmation e-mail, or follow the instructions on the screen to log on to the SAP Gateway WebGUI (<https://sapes5.sapdevcenter.com/>). |  |
| 1. Log on with your user name and temporary password. |  |
| 1. Change the password from a temporary to a permanent password. 2. **Write down the new password for the next configuration step.** |  |
| 1. The initial screen appears, which means your user ID and password work. You’re all set to create a destination in the SAP Cloud Platform Cockpit. This destination will then be the connection between the service and your app. |  |
| 1. To get to the SAP Cloud Platform Cockpit, go to your Web IDE Full-Stack trial account (use your bookmark, if have set it). 2. Go to *Tools* 🡪 *SAP Cloud Platform Cockpit*. |  |
| 1. In the SAP Cloud Platform Cockpit, choose *Connectivity* 🡪 *Destinations*. 2. Choose *Import Destination*. Leave the import popup open as you perform the next step. |  |
| 1. In another tab, download the *ES5 destination.txt* file from the UI52 GitHub repository folder *import* by right clicking the *Raw* button and choosing *Save Link as*…   [ES5 Destination on GitHub](https://github.com/SAP/openSAP-ui5-course/blob/master/import/ES5)  Save it to your local computer. Then select it in the import popup that appeared after you clicked on *Import Destination*. |  |
| 1. Ensure the field entries match the screenshot on the right side, and enter your user name and password for the SAP Gateway Demo System. 2. Choose *Save*.   **Note: Additional Properties**  WebIDEEnabled: This makes this destination visible to SAP Web IDE.  WebIDESystem: This is the system ID that the service runs on.  WebIDEUsage: You can enter multiple usages for a destination, separated by commas without spaces. |  |
| 1. This is the resulting destination configuration. Next, click on *Check Connection* to test if the URL is plausible.   **Note: *Check Connection* does not check user credentials**  An incorrect user or password does not cause this test to fail, because it only tests the URL connectivity, not the authentication to the system. |  |
| 1. If you see the message on the right, the test was successful, and the URL is correct. |  |

### Preview the application

| Explanation | **Screenshot** |
| --- | --- |
| 1. Open the SAP Fiori launchpad in the ES5 system: <https://sapes5.sapdevcenter.com/sap/bc/ui5_ui5/ui2/ushell/shells/abap/FioriLaunchpad.html?sap-client=002&sap-language=EN#Shell-home> |  |
| 1. Open the [Manage Products](https://sapes5.sapdevcenter.com/sap/bc/ui5_ui5/ui2/ushell/shells/abap/FioriLaunchpad.html#EPMProduct-manage) application. | C:\Users\d045752\AppData\Local\Temp\SNAGHTMLd972ee.PNG |

## Adding a button to a smart table

### Summary

You like this application, but there is a lot of back and forth communication between the UI5 air cargo employees and their customers. They often copy and paste product IDs and prices to email. You would like to improve that by adding a button to the smart table that allows the end user to send search results to another user via email.

### Preview

In this exercise, you will learn how to use adaptation projects to extend existing SAPUI5 applications without changing the source application’s code. You will create a variant of an existing SAP Fiori elements application and add a button that sends an email containing the results of a search.

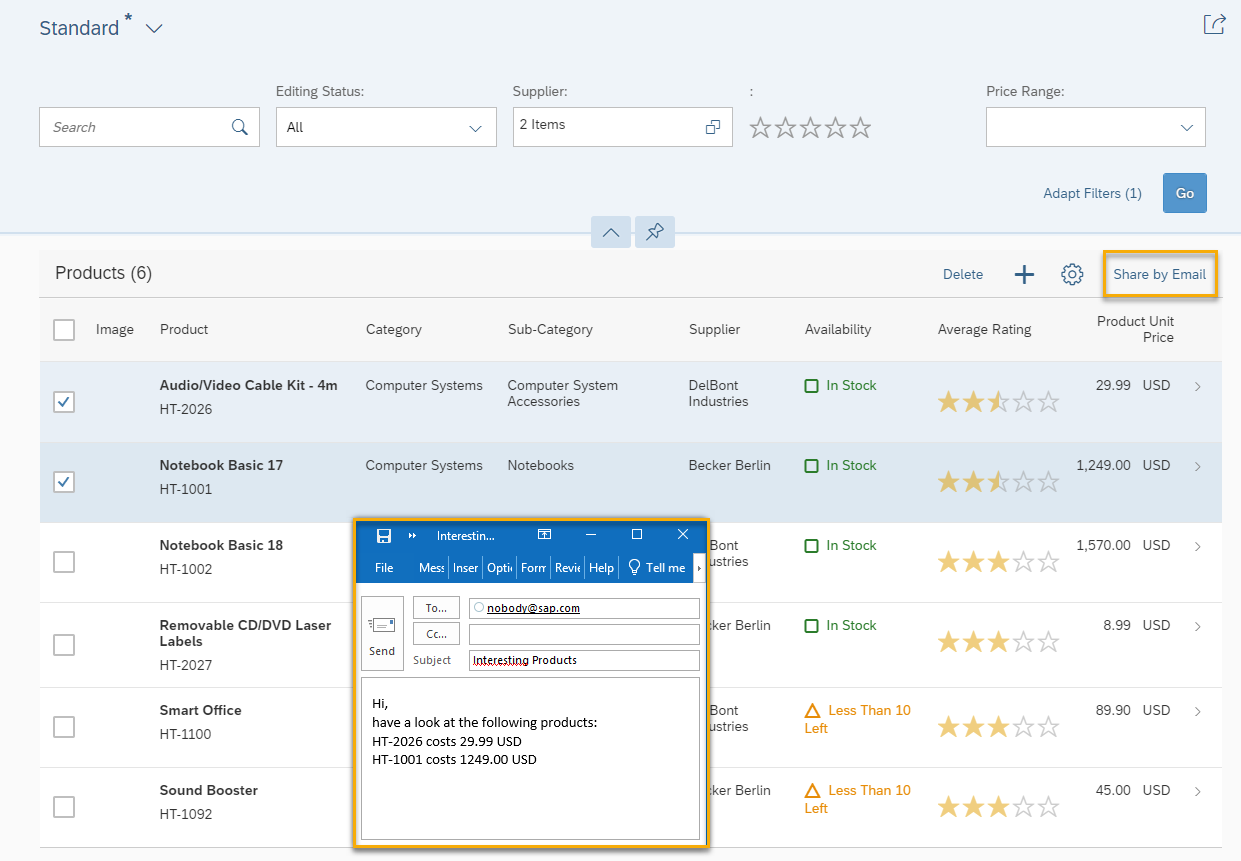


Figure 2 – A Share by Email button added to a variant of an existing application

### Create adaptation project

| Explanation | **Screenshot** |
| --- | --- |
| 1. Open SAP Web IDE. |  |
| 1. Create a new adaptation project: Open the context menu on the Workspacefolder, and choose *New* 🡪 *Adaptation Project*. |  |
| 1. Enter a project name, and an application title, then choose *Next*.  *Project Name*: **ui5con.manage.products** *Application Title*: **Manage Cargo** *Namespace*: <accept default value>   **Note: *Remember the namespace. It will be important for your accessing your added controls and controller extensions***. |  |
| 1. Select a system and source application upon which to base the variant you are about to create.   Select **SAP Gateway Demo System** (that’s ES5). You may need to enter your user name and password for that system. |  |
| 1. Select the application with the description *Manage Products* and choose *Finish*. |  |
| 1. A new project is created and shows you a working application based on the source application. Any changes you make in this variant are not going to change the code of the source application.  Open this project in the SAPUI5 Visual Editor, where you can preview what the application looks like.   Right-click on the project, and choose*SAPUI5 Visual Editor*.  **Note:** You might be prompted for a user name and password. If so, enter your user name and password for the ES5 system. | C:\Users\d045752\AppData\Local\Temp\SNAGHTML6f2f03.PNG |
| 1. Inside the running app, do a search where you filter on price range to see how it works.  Choose *Go.* |  |

### Safe Mode

When you’re in safe mode, you can only make changes that also key users can make in key user adaptation, such as moving sections, renaming fields, etc. However, we want to change more than that. But this also means that we will touch more parts of the source application that might change when the source app is upgraded.

| Explanation | **Screenshot** |
| --- | --- |
| 1. Choose *Safe Mode.* |  |
| 1. De-select the *Enable Safe Mode*checkbox, and choose *Apply* |  |
| 1. Choose *Edit*. |  |

### Adding a button as a fragment

With safe mode disabled, you have the generic option to add new SAPUI5 controls anywhere in the application by adding an XML fragment that contains them.

| Explanation | **Screenshot** |
| --- | --- |
| 1. Select the table’s toolbar. |  |
| 1. Right-click on the table toolbar area, and choose the *+**Add Fragment*menu item. |  |
| 1. The *Target Aggregation* for the new fragment is the preselected **content** aggregation, and *Index* should remain the last element in that aggregation.  Choose the *Create new*link. |  |
| 1. In the *Add Fragment* dialog box, enter **ShareButton** as the fragment name, and choose *Create*. |  |
| 1. Now you have a new ShareButton.fragment.xml file that's blank, and you need to add some code. Copy the following code, and paste it to the ShareButton.fragment.xml file. |  |

**webapp/changes/fragments/ShareButton.fragment.xml**

|  |
| --- |
| <!-- Use stable and unique id's!-->  <core:FragmentDefinition xmlns:core='sap.ui.core' xmlns='sap.m'>  <Button id="shareByEmail" text="{i18n>SHARE\_BY\_EMAIL}" press=".extension.customer.ui5con.manage.products.ListReport.onShare"></Button>  </core:FragmentDefinition> |

| Explanation | **Screenshot** |
| --- | --- |
| 1. *Save*. |  |
| 1. Go back to the *ui5con.manage.products* tab, accept the message that says changes were made outside the editor, and let it reload the editor. You may be asked if you want to save unsaved changes in the editor – if so, choose *Save*. |  |

### Adding translatable texts

In fragments you might need translatable texts. In the fragment here, there is a reference to a new i18n key SHARE\_BY\_EMAIL. Now you need to provide some texts for this key. The i18n.properties file can be translated, and language specific files can be provided next to it, like in any UI5 app.

| Explanation | **Screenshot** |
| --- | --- |
| 1. Right-click on the i18n.properties file for the ListReport folder of this project, and choose *Open Code Editor***.** |  |
| 1. Add the following text keys to the i18n.propertiesfile and *Save*. |  |

**webapp/i18n/ListReport/SEPMRA\_C\_PD\_Product/i18n.properties**

|  |
| --- |
| #Make sure you provide a unique prefix to the newly added keys in this file, to avoid overriding of SAP Fiori application keys.  # Share button texts  SHARE\_BY\_EMAIL=Share by Email  SHARE\_SUBJECT=Interesting Products  SHARE\_BODY=Hi, \nhave a look at the following products:\n{0}  SHARE\_PRODUCT={0} costs {1} {2} \n |

|  |  |
| --- | --- |
| 1. Go back to the *ui5con.manage.products* tab, accept the message that says changes were made outside the editor, and let it reload the editor. You’ll see the new *Share by Email* button, but there is still no code there to control how it works.   Choose *Edit*. |  |
| 1. Optional: If you cannot see the *Share by Email* button when you’re in edit mode, because the screen area is too small, choose the *Collapses left pane* button. This will allow you more space to see your application in the editor. |  |

### Adding event handler as controller extension

Controller extensions allow you to add functionality like the button’s press event handler.

| Explanation | **Screenshot** |
| --- | --- |
| 1. Right-click the *Share by Email* button, and choose *Extend With Controller*. |  |
| 1. Enter **ListReport** as the controller name, and choose *Extend.* |  |
| 1. Now you are looking at a new file, ListReport.js. Notice this is *not* an empty file.  Replace the contents with the following code and *Save* afterwards. |  |

**webapp/changes/coding/ListReport.js**

|  |
| --- |
| /\*\*\*  @controller Name:sap.suite.ui.generic.template.ListReport.view.ListReport,  \*@viewId:nw.epm.refapps.st.prod.manage::sap.suite.ui.generic.template.ListReport.view.ListReport::SEPMRA\_C\_PD\_Product  \*/  sap.ui.define([  "sap/ui/core/mvc/ControllerExtension"  // ,"sap/ui/core/mvc/OverrideExecution"  ],  function (  ControllerExtension  // ,OverrideExecution  ) {  "use strict";  return ControllerExtension.extend("customer.ui5con.manage.products.ListReport", {  onShare: function (oEvent) {  //Similar use case: https://ui5.sap.com/#/topic/a269671fc49e4c75920c108961bf31f2  // 1. Get SAP Fiori elements extensionAPI instance:  // \* SAP Fiori elements ListReport base controller is accessed by this.base  // \* templateBaseExtension provides public API for SAP Fiori elements extensions, like the extensionAPI that is also used by SAP Fiori elements developers  var oExtensionAPI = this.base.templateBaseExtension.getExtensionAPI();  // 2. Get selected products via extensionAPI methods  // see https:// ui5.sap.com/#/api/sap.suite.ui.generic.template.ListReport.extensionAPI.ExtensionAPI/methods/getSelectedContexts  var aSelection = oExtensionAPI.getSelectedContexts();  if (aSelection.length > 0) {  // access your i18n model where the texts are available  var oResourceBundle = this.getView().getModel("i18n").getResourceBundle();  var sTo = "nobody@sap.com";  var sSubject = oResourceBundle.getText("SHARE\_SUBJECT");  var sProducts = aSelection.reduce(function (sText, oSelectedContext) {  // for each selected product, get the OData entity  var mSelectedData = oSelectedContext.getObject();  // add a line with some entity informaton to your email  return sText + oResourceBundle.getText("SHARE\_PRODUCT", [mSelectedData.Product, mSelectedData.Price, mSelectedData.Currency]);  }, "");  // combine the body text  var sBody = oResourceBundle.getText("SHARE\_BODY", [sProducts]);  // use SAPUI5 helper functionality to prepare the email  sap.m.URLHelper.triggerEmail(sTo, sSubject, sBody);  }  }  });  }); |

### Preview results

Now let’s see your work in action and preview this as a web application.

| Explanation | **Screenshot** |
| --- | --- |
| 1. *Choose Run*🡪*Run as Web Application*. |  |
| 1. If asked which configuration to use, select the *Adaptation\_index.html*and choose *OK*. |  |
| 1. Now you’re running your application! Filter for a supplier’s name. Then select the items you’d like to share, and choose the *Share by Email* button. |  |
| 1. A new email is created using your computer’s default email program. It contains the data that you selected! |  |

## Adding a Dimension Filter

### Summary

For an air cargo company, it is essential to use all the space available on an air craft. UI5 Air Cargo uses standardized boxes for all products. Once in a while, there is space left and UI5 Air Cargo employees would like to know which products could fit in.

You should add a special filter to the filter bar for products that fit into one of the following box types that UI5 Air Cargo uses:

• Small boxes of 10 cm in each dimension

• Medium boxes of 20 cm in each dimension

• Large boxes above 20 cm in each dimension

In this exercise, you will continue to adapt the application variant you created in the exercises.

### Preview

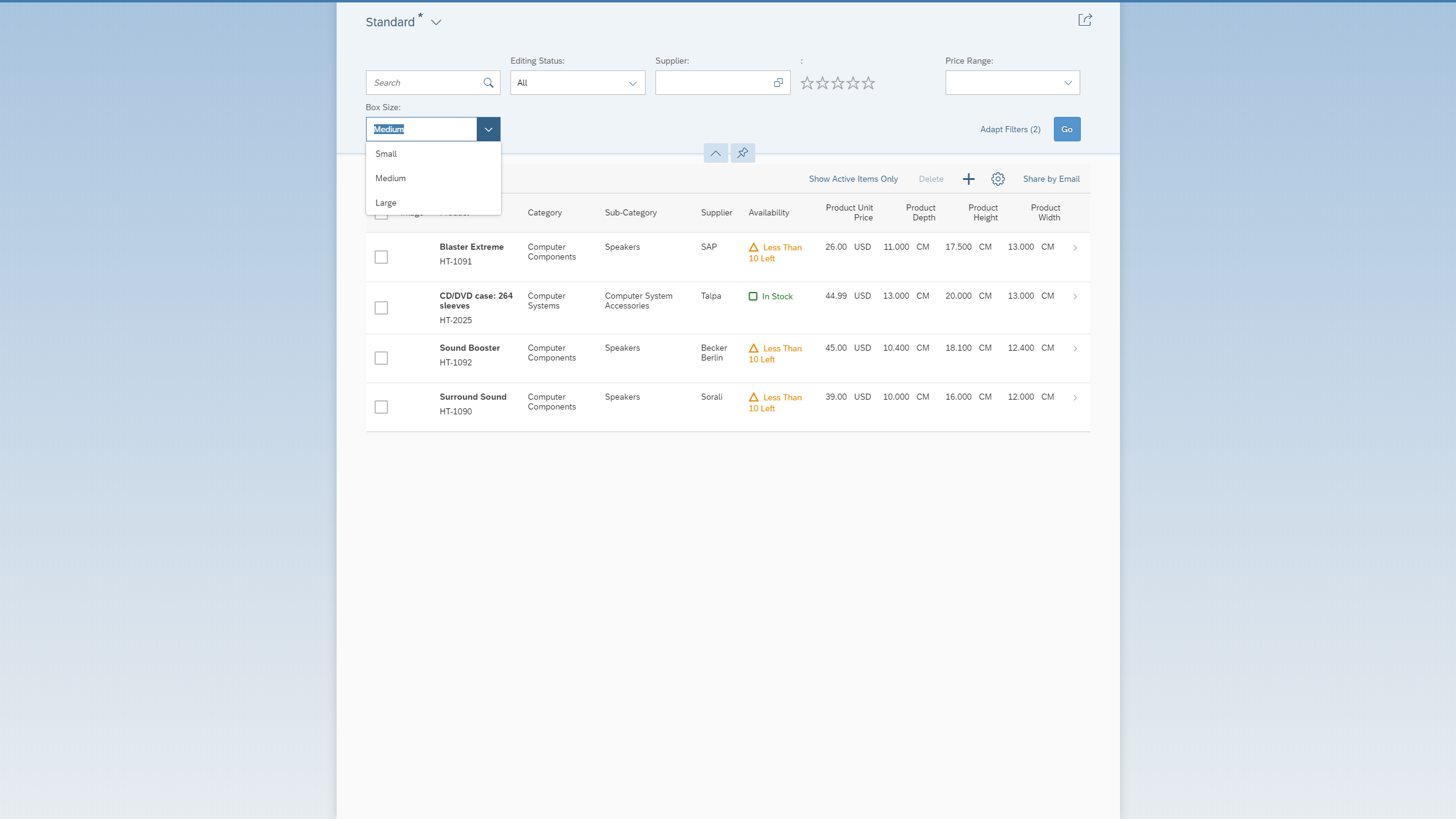


Figure 3 – An additional filter for box sizes added to the app variant

### Hints:

* [Adaptation Extension Example: Adding a Button to the Table Toolbar in the List Report](https://ui5.sap.com/#/topic/a269671fc49e4c75920c108961bf31f2)
* You can make additional columns visible via table personalization.

### Adding a filter as a fragment

With safe mode disabled, you have the generic option to add new SAPUI5 controls anywhere in the application by adding an XML fragment that contains them. We will also use this to add the filter.

| Explanation | **Screenshot** |
| --- | --- |
| 1. Open SAP Web IDE. |  |
| 1. First you need to open the application variant you created in the first exercise.  Right-click on *ui5con.manage.products,* and choose*SAPUI5 Visual Editor*.  **Note:** You might be prompted for a user name and password. If so, enter your user name and password for the ES5 system. | C:\Users\d045752\AppData\Local\Temp\SNAGHTML6f2f03.PNG |
| 1. Next, let’s do a search of what products are available. Choose*Go*. |  |
| 1. Personalize the table |  |
| 1. Select Product Depth, Height and Width columns |  |
| 1. Choose *Edit*. |  |
| 1. Select the filterbar area and open the context menu |  |
| 1. The *Target Aggregation* for the new fragment needs to be changes to **controlConfiguration**, and *Index* should remain the last element in that aggregation.  Choose the *Create new*link  **Note:** You can check the control’s API documentation about the functionality of the different aggregations, in this case: <https://ui5.sap.com/#/api/sap.ui.comp.smartfilterbar.SmartFilterBar> | C:\Users\d045752\AppData\Local\Temp\SNAGHTML561e228.PNG |
| 1. In the *Add Fragment* dialog box, enter **BoxFilter** as the fragment name, and choose *Create* |  |
| 1. Now you have a new BoxFilter.fragment.xml file that's blank, and you need to add some code. Copy the following code, and paste it to the BoxFilter.fragment.xml file. |  |

**webapp/changes/fragments/BoxFilter.fragment.xml**

|  |
| --- |
| <!-- Use stable and unique id's!-->  <core:FragmentDefinition xmlns:core='sap.ui.core' xmlns='sap.ui.comp.smartfilterbar' xmlns:m='sap.m'>  <ControlConfiguration  id="boxFilter"  key="boxFilter"  index="20"  label="{i18n>BOX\_FILTER}"  groupId="\_BASIC"  visibleInAdvancedArea="true"  filterType="multiple"  >  <customControl>  <m:ComboBox id="boxTypes">  <core:Item id="smallBox" key="small" text="{i18n>SMALL\_BOX}" />  <core:Item id="mediumBox" key="medium" text="{i18n>MEDIUM\_BOX}" />  <core:Item id="largeBox" key="large" text="{i18n>LARGE\_BOX}" />  </m:ComboBox>  </customControl>  </ControlConfiguration>  </core:FragmentDefinition> |

### Adding translatable texts

| Explanation | **Screenshot** |
| --- | --- |
| 1. Right-click on the i18n.properties file for the ListReport folder of this project, and choose *Open Code Editor***.** |  |
| 1. Add the following text keys to the i18n.propertiesfile and *Save*. |  |

**webapp/i18n/ListReport/SEPMRA\_C\_PD\_Product/i18n.properties**

|  |
| --- |
| ...  SHARE\_BODY=Hi, \nhave a look at the following products:\n{0}  SHARE\_PRODUCT={0} costs {1} {2} \n  # Box Filter texts  BOX\_FILTER=Box Size  SMALL\_BOX=Small  MEDIUM\_BOX=Medium  LARGE\_BOX=Large |

|  |  |
| --- | --- |
| 1. Go back to the *ui5con.manage.products* tab, accept the message that says changes were made outside the editor, and let it reload the editor. You’ll see the new texts for the filter, but there is still no code there to filter how it works. |  |

### Adding event handler as controller extension

Controller extensions allow you to hook into functionality of the base controller.

**webapp/changes/coding/ListReport.js**

|  |
| --- |
| /\*\*\*  @controller Name:sap.suite.ui.generic.template.ListReport.view.ListReport,  \*@viewId:nw.epm.refapps.st.prod.manage::sap.suite.ui.generic.template.ListReport.view.ListReport::SEPMRA\_C\_PD\_Product  \*/  sap.ui.define([  "sap/ui/core/mvc/ControllerExtension",  "sap/ui/model/Filter",  "sap/ui/model/FilterOperator"  // ,"sap/ui/core/mvc/OverrideExecution"  ],  function (  ControllerExtension,  Filter,  FilterOperator  // ,OverrideExecution  ) {  "use strict";  //box type definition:  var BOX\_TYPES = {  small: new Filter({  filters: [  new Filter({  path: "Height",  operator: FilterOperator.BT,  value1: 0,  value2: 10  }),  new Filter({  path: "Width",  operator: FilterOperator.BT,  value1: 0,  value2: 10  }),  new Filter({  path: "Depth",  operator: FilterOperator.BT,  value1: 0,  value2: 10  })  ],  and: true  }),  medium: new Filter({  filters: [  new Filter({  path: "Height",  operator: FilterOperator.BT,  value1: 10,  value2: 20  }),  new Filter({  path: "Width",  operator: FilterOperator.BT,  value1: 10,  value2: 20  }),  new Filter({  path: "Depth",  operator: FilterOperator.BT,  value1: 10,  value2: 20  })  ],  and: true  }),  large: new Filter({  filters: [  new Filter({  path: "Height",  operator: FilterOperator.GT,  value1: 20  }),  new Filter({  path: "Width",  operator: FilterOperator.GT,  value1: 20  }),  new Filter({  path: "Depth",  operator: FilterOperator.GT,  value1: 20  })  ],  and: true  })  };  return ControllerExtension.extend("customer.ui5con.manage.products.ListReport", {  // this section allows to extend lifecycle hooks or override public methods of the base controller  override: {  // override public methods of the ListReport controller and its members like templateBaseExtension  templateBaseExtension: {  /\*\*  \* Can be used to add filters. They will be combined via AND with all other filters  \* sControlId is the ID of the control on which extension logic to be applied.  \* For each filter the extension must call fnAddFilter(oControllerExtension, oFilter)  \* oControllerExtension must be the ControllerExtension instance which adds the filter  \* oFilter must be an instance of sap.ui.model.Filter  \*/  addFilters: function (fnAddFilter, sControlId) {  var oComboBox = this.byId("boxTypes");  var sSelectedBoxType = oComboBox.getSelectedKey();  //lookup the filters from the definition  var oFilter = BOX\_TYPES[sSelectedBoxType];  if (oFilter) {  fnAddFilter(this, oFilter);  }  }  }  },  onShare: function (oEvent) {  ... |

### Preview results

Now let’s see your work in action and preview this as a web application.

| Explanation | **Screenshot** |
| --- | --- |
| 1. *Choose Run*🡪*Run as Web Application*. |  |
| 1. If asked which configuration to use, select the *Adaptation\_index.html*and choose *OK*. |  |
| 1. Now you’re running your application!  Add the product dimension columns via personalization and filter the product list with new the *Box Size* filter. |  |

## Adding a map to show the supplier location

### Summary

As a very special carrier, UI5 Air cargo, directly flies to its suppliers to pick up the products. To find out where to land close to the supplier’s location, they need the supplier’s address and a map showing the surrounding area. You will continue adapting the application variant you created in the first exercise by adding a map to the application’s object page.

### Preview

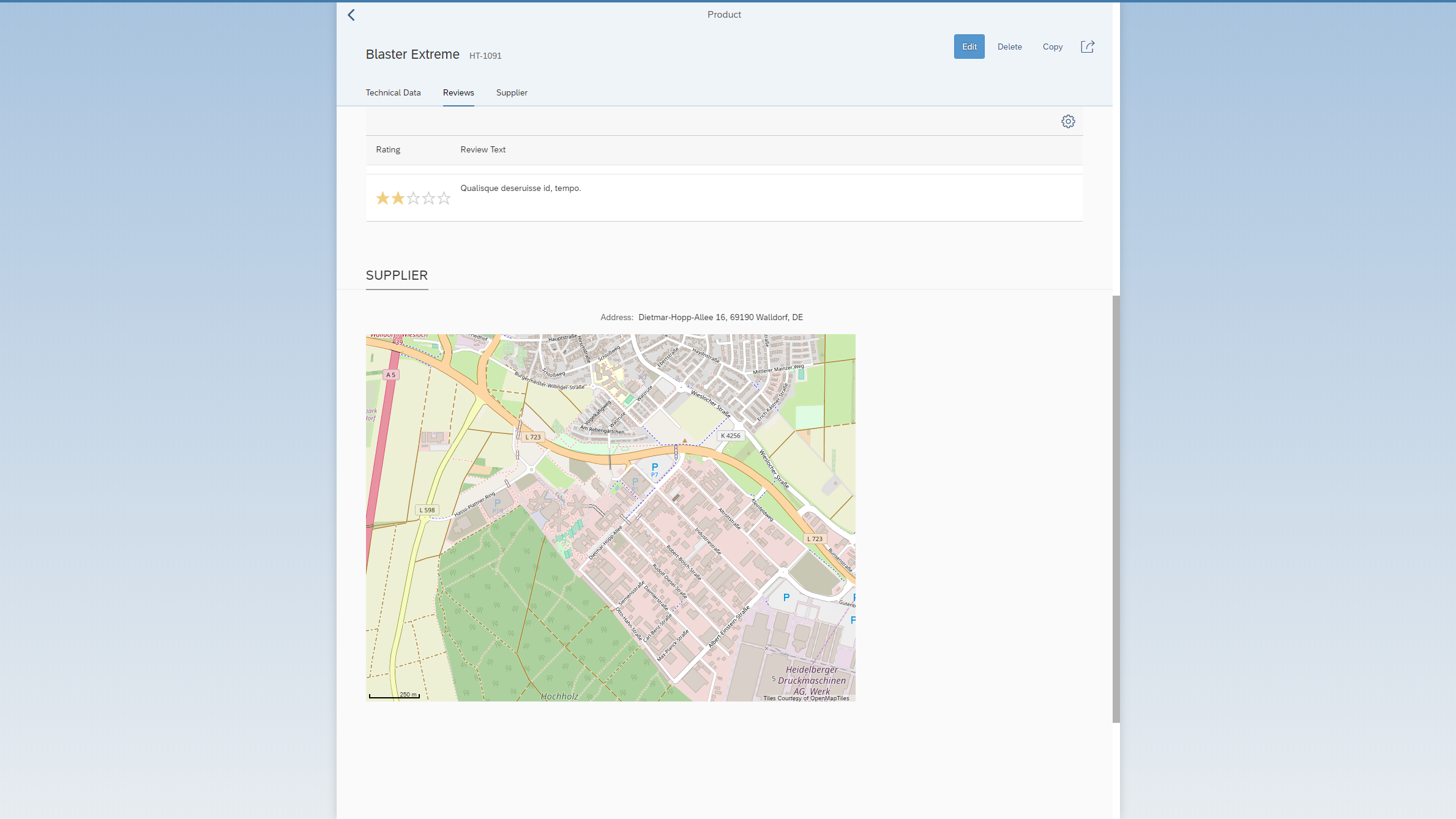


Figure 2 – A map of the product supplier’s location added to the app variant

### Hints:

* <https://ui5.sap.com/test-resources/sap/ui/vbm/bestpractices.html>
* <https://open.sap.com/courses/3dmv1>

### Adding the map

| Explanation | **Screenshot** |
| --- | --- |
| 1. Open SAP Web IDE. |  |
| 1. First you need to open the application variant you created in the first exercise.  Right-click on *ui5con.manage.products,* and choose*SAPUI5 Visual Editor*.  **Note:** You might be prompted for a user name and password. If so, enter your user name and password for the ES5 system. | C:\Users\d045752\AppData\Local\Temp\SNAGHTML6f2f03.PNG |
| 1. Next, let’s do a search of what products are available. Choose*Go*. |  |
| 1. A list of available products will appear. Select the first item in the list. |  |
| 1. This will open the object page view of the application, displaying details of the product you selected.  What you are going to do in this exercise is adding a *Supplier* section and a map to this page, with the map showing the location of the supplier for the product. |  |
| 1. Choose *Edit.* |  |
| 1. To add a new section, you need to add a new fragment.   Right click on the object page layout, and choose *+ Add Fragment*. The properties panel shows whether you selected the correct control. |  |
| 1. Here you are going to create a new section for the supplier map.  The target aggregation is the preselected **sections** aggregation, and the value for *Index* should remain the last in that aggregation  Choose the *Create new*link. |  |
| 1. In the *Add Fragment* dialog box, enter **SupplierMap** as the fragment name, and choose *Create*. |  |
| 1. Now you have a new SupplierMap.fragment.xml file that's blank, and you need to add some content. Copy the following code, paste it to the SupplierMap.fragment.xml, and *Save* it. |  |

**webapp/changes/fragments/SupplierMap.fragment.xml**

|  |
| --- |
| <!-- Use stable and unique id's!-->  <core:FragmentDefinition xmlns:core='sap.ui.core' xmlns:uxap='sap.uxap' xmlns:vbm='sap.ui.vbm' xmlns:form='sap.ui.layout.form' xmlns='sap.m' >  <uxap:ObjectPageSection  id="supplierSection"  title="{i18n>SUPPLIER}">  <uxap:ObjectPageSubSection id="supplierSubSection">  <VBox id="supplierAddress">  <form:SimpleForm id="supplierAddressForm">  <Label id="supplierAddressLabel" text="{i18n>ADDRESS}"></Label>  <Text id="supplierFormattedAddress" text="{to\_Address/FormattedAddress}"></Text>  </form:SimpleForm>  <!-- https://sapui5.hana.ondemand.com/test-resources/sap/ui/vbm/bestpractices.html -->  <vbm:GeoMap id="supplierMap" initialZoom="15">  </vbm:GeoMap>  </VBox>  </uxap:ObjectPageSubSection>  </uxap:ObjectPageSection>  </core:FragmentDefinition> |

### Adding the texts

| Explanation | **Screenshot** |
| --- | --- |
| 1. To get translatable texts for the new section, add keys to the i18n.properties file. In the tree view of SAP Web IDE, right-click on the i18n.properties for the *ObjectPage* of this project, and choose *Open Code Editor***.** |  |
| 1. Copy the following code, and replace the contents of i18n.properties**.** Then choose *Save* |  |

**webapp/i18n/ObjectPage/SEPMRA\_C\_PD\_Product/i18n.properties**

|  |
| --- |
| #Make sure you provide a unique prefix to the newly added keys in this file, to avoid overriding of SAP Fiori application keys.  SUPPLIER=Supplier  ADDRESS=Address |

### Adding controller extension

| Explanation | **Screenshot** |
| --- | --- |
| 1. Go back to the *ui5con.manage.products* tab, accept the message that says changes were made outside the editor, and let it reload the editor. You may be asked if you want to save unsaved changes in the editor – if so, choose *Save*. |  |
| 1. Next, do a search of what products are available. Choose*Go*. |  |
| 1. A list of available products will appear. Choose the first item in the list. |  |
| 1. Now you see a *Supplier* section on this page.   Choose *Edit*. |  |

|  |  |
| --- | --- |
| 1. Right-click on the *Supplier* section, and choose *Extend With Controller*. |  |
| 1. Enter **ObjectPage** as the name,and choose *Extend.* |  |
| 1. Now you are looking at a new file, ObjectPage.js.   Replace the contents with the following code and *Save* afterwards. |  |

**webapp/changes/coding/ObjectPage.js**

|  |
| --- |
| /\*\*\*  @controller Name:sap.suite.ui.generic.template.ObjectPage.view.Details,  \*@viewId:nw.epm.refapps.st.prod.manage::sap.suite.ui.generic.template.ObjectPage.view.Details::SEPMRA\_C\_PD\_Product  \*/  sap.ui.define([  'sap/ui/core/mvc/ControllerExtension'  // ,'sap/ui/core/mvc/OverrideExecution'  ],  function (  ControllerExtension  // ,OverrideExecution  ) {  "use strict";  return ControllerExtension.extend("customer.ui5con.manage.products.ObjectPage", {  // this section allows to extend lifecycle hooks or override public methods of the base controller  override: {  // /\*\*  // \* Called when a controller is instantiated and its View controls (if available) are already created.  // \* Can be used to modify the View before it is displayed, to bind event handlers and do other one-time initialization.  // \* @memberOf customer.acme.manage.products.ObjectPage.js  // \*/  onInit: function () {  //configure map to use openstreetmap data  var oMapConfig = {  "MapProvider": [{  "name": "OSM",  "type": "",  "description": "",  "tileX": "256",  "tileY": "256",  "maxLOD": "20",  "copyright": "Tiles Courtesy of OpenMapTiles",  "Source": [{  "id": "s1",  "url": "https://a.tile.openstreetmap.org/{LOD}/{X}/{Y}.png"  }]  }],  "MapLayerStacks": [{  "name": "Default",  "MapLayer": [{  "name": "OSM",  "refMapProvider": "OSM"  }]  }]  };  var oSupplierMap = this.byId("supplierMap");  oSupplierMap.setMapConfiguration(oMapConfig);  //bind the whole section to the supplier including the supplier address  var oSupplierSection = this.byId("supplierSection");  oSupplierSection.bindElement("to\_Supplier", {  //inline the address data in the supplier  expand: "to\_Address"  });  oSupplierMap.bindProperty("initialPosition", {  parts: [  "to\_Address/Longitude",  "to\_Address/Latitude"  ],  formatter : function(sLong,sLat){  return sLong +";" + sLat + ";0" ;  }  });  }  }  });  }); |

### Preview results

Now let’s see your work in action and preview this as a web application.

| Explanation | **Screenshot** |
| --- | --- |
| 1. *Choose Run*🡪*Run as Web Application*. |  |
| 1. If asked which configuration to use, select **Adaptation\_index.html**, and choose *OK*. |  |
| 1. Now you’re running your application!  Choose *Go* to start a search, and then select the first item in the results list. |  |
| 1. On the product details page, notice that there is a *Supplier* tab now.  Click on the *Supplier*tab (or scroll down to the Supplier section). |  |
| 1. Voilà, you have a map to show the address and location of the supplier! |  |

# Related material

For more information on adaptation projects check out the following links:

* [Documentation: SAPUI5 Visual Editor](https://help.sap.com/viewer/825270ffffe74d9f988a0f0066ad59f0/CF/en-US/17874ff21cb04d799657afb03d9e8b39.html)
* [Documentation: Adaptation Projects for Fiori Elements Applications](https://help.sap.com/viewer/825270ffffe74d9f988a0f0066ad59f0/CF/en-US/94f024b04a6245d8ba9e02ee2facc527.html)
* [Demo Kit: Extending Delivered Apps Using Adaptation Extensions](https://ui5.sap.com/#/topic/52fc48b479314d0688be24f699778c47)
* [Demo Kit: Reusing UI Parts with Fragments](https://ui5.sap.com/#/topic/36a5b130076e4b4aac2c27eebf324909)
* [Demo Kit: Using Controller Extension](https://ui5.sap.com/#/topic/21515f09c0324218bb705b27407f5d61)
* [Demo Kit: Adaptation Extension Example: Adding a Button to the Table Toolbar in List Report](https://ui5.sap.com/#/topic/a269671fc49e4c75920c108961bf31f2)
* [Documentation: Deploy the App Variant](https://help.sap.com/viewer/825270ffffe74d9f988a0f0066ad59f0/CF/en-US/c7b29c2550724f54b913cd6882eb9f70.html)
* [Blog: Adaptation Projects - A Tutorial](https://blogs.sap.com/2019/01/14/adaptation-projects-a-tutorial/)