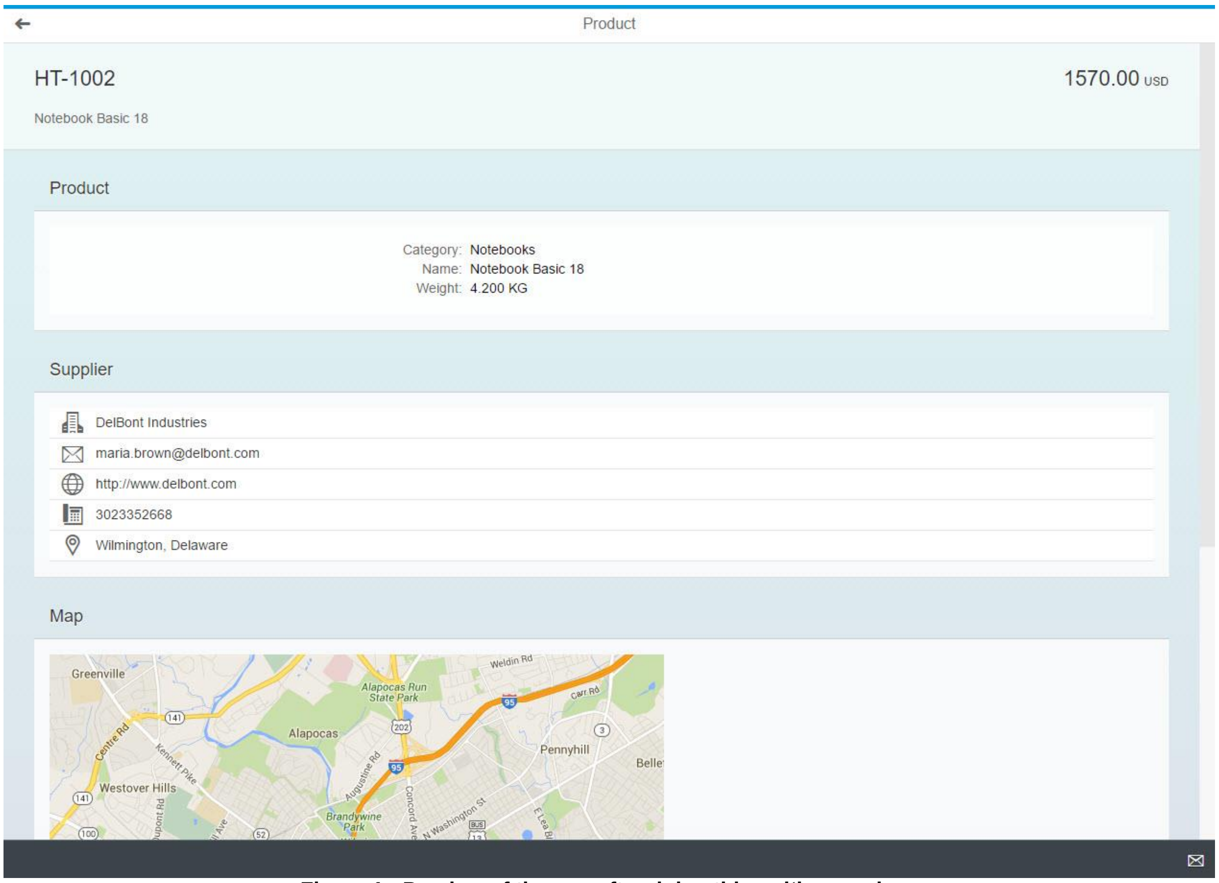
WEEK 3 UNIT 4
ADAPTING TO THE USER’S DEVICE

Please perform the exercises below in your app project as shown in the video.

Table of Contents

1 Add Content to Object View 3
2 Hide the Map on Phones 8
3 Make Panels Expandable/Collapsible on Phones, using the Device Model 9

Preview



Product

HT-1002 1570.00 USD

Notebook Basic 18

Product

Category: Notebooks
Name: Notebook Basic 18
Weight: 4.200 KG

Supplier

DelBont Industries
maria.brown@delbont.com
http://www.delbont.com
3023352668
Wilmington, Delaware

Map

Figure 1 - Preview of the app after doing this unit’s exercises

But in a mobile phone, the same page will looks like this:

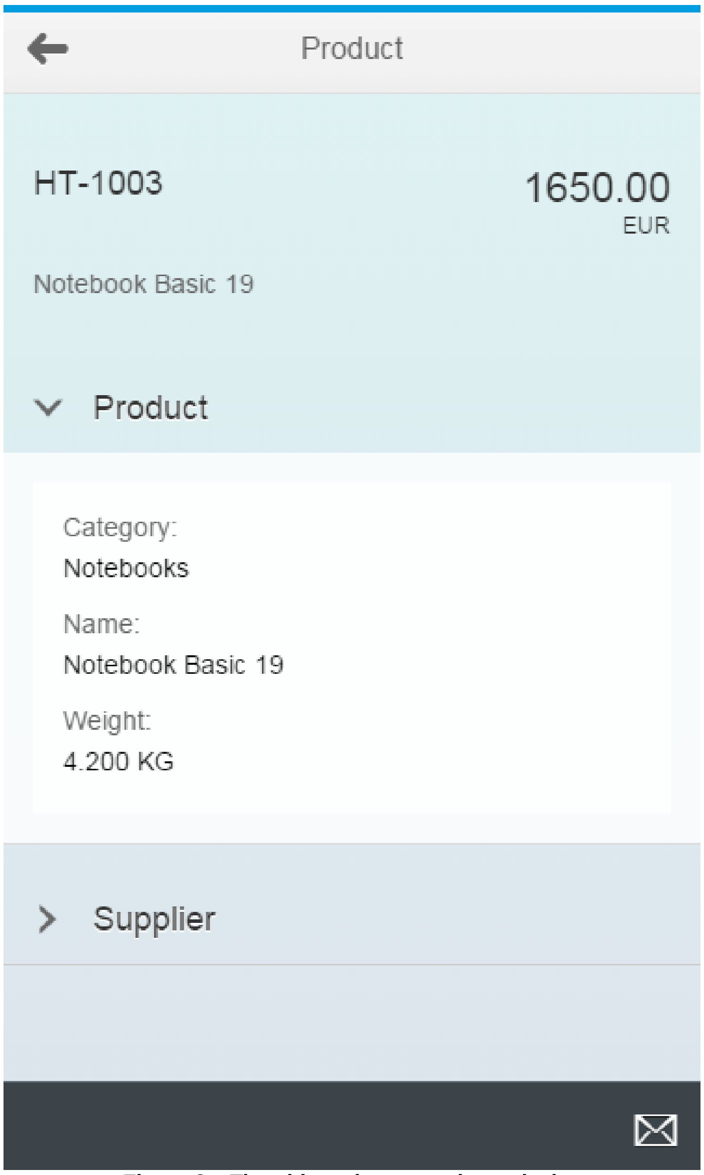


Figure 2 – The object view on a phone device

1 ADD CONTENT TO OBJECT VIEW

As the Product page of the application is still fairly empty, we first need some content to work with. So please add the highlighted code to the Object view.

webapp/view/Object.view.xml

```
<mvc:View
 controllerName="opensap.manageproducts.controller.Object"
 xmlns="sap.m"
 xmlns:mvc="sap.ui.core.mvc"
 xmlns:semantic="sap.m.semantic"
 xmlns:form="sap.ui.layout.form">

 <semantic:FullscreenPage
   id="page"
   navButtonPress="onNavBack"
   showNavButton="true"
   title="{i18n>objectTitle}"
   busy="{objectView>/busy}"
   busyIndicatorDelay="{objectView>/delay}">

   <semantic:content>
     <ObjectHeader
       id="objectHeader"
       title="{ProductID}"
       responsive="true"
       number="{
         path: 'Price',
         formatter: '.formatter.numberUnit'
       }"
       numberUnit="{CurrencyCode}">
       <attributes>
         <ObjectAttribute text="{Name}"/>
       </attributes>
     </ObjectHeader>

     <Panel
       class="sapUiResponsiveMargin"
       width="auto"
       headerText="{i18n>productTitle}"
       expandable="{device>/system/phone}"
       expanded="true">
       <content>
         <form:SimpleForm id="objectForm">
           <form:content>
             <Label text="{i18n>productCategoryLabel}"/>
             <Text text="{Category}"/>
             <Label text="{i18n>productNameLabel}"/>
             <Text text="{Name}"/>
             <Label text="{i18n>productWeightLabel}"/>
             <Text text="{= ${WeightMeasure} + ' ' + ${WeightUnit}}"/>
           </form:content>
         </form:SimpleForm>
       </content>
     </Panel>
```

```

      <Panel
        class="sapUiResponsiveMargin"
        width="auto"
        headerText="{i18n>supplierTitle}"
        expandable="{device>/system/phone}"
        expanded="false">
        <content>
          <List id="supplierList">
            <items>
              <StandardListItem icon="sap-icon://building"
title="{ToSupplier/CompanyName}"/>
              <StandardListItem icon="sap-icon://email"
title="{ToSupplier/EmailAddress}"/>
              <StandardListItem icon="sap-icon://world"
title="{ToSupplier/WebAddress}"/>
              <StandardListItem icon="sap-icon://phone"
title="{ToSupplier/PhoneNumber}"/>
              <StandardListItem icon="sap-icon://map"
title="{ToSupplier/Address/City}"/>
            </items>
          </List>
        </content>
      </Panel>

      <Panel
        class="sapUiResponsiveMargin"
        width="auto"
        headerText="{i18n>mapTitle}">
        <Image src="{
          parts: [
            'ToSupplier/Address/Street',
            'ToSupplier/Address/PostalCode',
            'ToSupplier/Address/City',
            'ToSupplier/Address/Country'
          ],
          formatter: '.formatter.formatMapUrl'
        }" />
      </Panel>

    </semantic:content>

    <semantic:sendEmailAction>
      <semantic:SendEmailAction id="shareEmail" press="onShareEmailPress"/>
    </semantic:sendEmailAction>

  </semantic:FullscreenPage>

</mvc:View>
```

Most of the new code adds three panels – the first one contains a form with product information, the second one a list with supplier information and the third one a map image.

As the form you are adding has its own package name `sap.ui.layout.form`, do not forget to define the respective XML namespace “form” in the root tag of the view XML document, as depicted in the code above!

webapp/controller/Object.controller.js

```
….
    _bindView : function (sObjectPath) {
          var oViewModel = this.getModel("objectView"),
             oDataModel = this.getModel();

          this.getView().bindElement({
             path: sObjectPath,
             parameters: {
                expand: "ToSupplier"
             },
             events: {
…
```

Furthermore, the list with supplier information displays data from the “ToSupplier” association in the data. To make the OData service include this associated information even when launching the product page directly, we need to give this association name as “expand parameter” when binding the Object view. Please do this in the controller belonging to the Object view.

The map in the third panel is actually a `sap.m.Image` control. The image source (the URL where the image is loaded from) is bound to the `formatMapUrl` formatter, to which supplier information like the street, city and country is provided. To make the map work, we still need to implement this formatter, so please add the following code to formatter.js.

webapp/model/formatter.js

```
…
        numberUnit : function (sValue) {
           if (!sValue) {
              return "";
           }
           return parseFloat(sValue).toFixed(2);
        },

        /**
         * Formats an address to a static google maps image
         * @public
         * @param {string} sStreet the street
         * @param {string} sZIP the postal code
         * @param {string} sCity the city
         * @param {string} sCountry the country
         * @returns {string} sValue a google maps URL that can be bound to an
image
         */
        formatMapUrl: function(sStreet, sZIP, sCity, sCountry) {
           return
"https://maps.googleapis.com/maps/api/staticmap?zoom=13&size=640x640&markers="
              + jQuery.sap.encodeURL(sStreet + ", " + sZIP +  " " + sCity + ", "
+ sCountry);
        }

     };

  }
);
```

This code appends the supplier location information to a Google Maps API URL and returns the URL. The image control will request the image from this URL and Google Maps will return a static bitmap image that shows the requested location.

Note:
This free Google Maps API is used for educational purposes only. When intending to display maps in a productive application, always check the terms of service of the map provider.

Note:
This Maps API has a usage limit per application. Many people doing this course will call the service from the same server at about the same time, so the limit may be exceeded and the API might then not return an actual map image. In this case you can still get an impression of what you would see, by opening the following URL in your browser (displaying the SAP headquarter's address): https://maps.googleapis.com/maps/api/staticmap?zoom=13&size=640x640&markers=dietmar+hopp+allee+16,walldorf

There is one more thing missing to make the vastly expanded Product page work: the translation texts for the new UI labels. Please add them to the i18n.properties file.

webapp/i18n/i18n.properties

```
…
#~~~ Object View ~~~~~~~~~~~~~~~~~~~~~~~~~~

#XTIT: Object view title
objectTitle=Product

#XGRP: Product panel title
productTitle=Product

#XFLD: Product category
productCategoryLabel=Category

#XFLD: Product name
productNameLabel=Name

#XFLD: Product weight
productWeightLabel=Weight

#XGRP: Supplier panel title
supplierTitle=Supplier

#XGRP: Map panel title
mapTitle=Map

#~~~ Footer Options ~~~~~~~~~~~~~~~~~~~~~~~
…
```

You should now be able to run the application and see the content displayed in the first screenshot. Note that the margins at the left and right of the panels change when you are changing the width of the browser window. When the window is very narrow, these margins will completely disappear. This is due to the “sapUiResponsiveMargin” CSS class which is set for all three panels in the code you added. The margin and padding classes have already been explained in week 1 of this course, but now we will add more responsiveness features.

2 HIDE THE MAP ON PHONES

We now use one of the classes for responsive hiding of content. Add the “sapUiHideOnPhone” class to the third panel to hide the map on small screens of smartphone size.
You can find more details in the SDK.

webapp/view/Object.view.xml

```
<Panel
   class="sapUiResponsiveMargin sapUiHideOnPhone"
   width="auto"
   headerText="{i18n>mapTitle}">
   <Image src="{
      parts: [
         'ToSupplier/Address/Street',
         'ToSupplier/Address/PostalCode',
         'ToSupplier/Address/City',
         'ToSupplier/Address/Country'
      ],
      formatter: '.formatter.formatMapUrl'
   }" />
</Panel>
```

Making the window narrow will now hide the map panel.

Note that this hiding happens on CSS level, so the HTML of the Panel will still be created. To entirely avoid the creation of the HTML for performance reasons in cases where it is complex, it is recommended to set the “visible” property of controls to “false” on SAPUI5 level instead, e.g. using the Device Model explained below.

3 MAKE PANELS EXPANDABLE/COLLAPSIBLE ON PHONES, USING THE DEVICE MODEL

The application template used to create this application already contains code, which creates a so-called "Device Model" – a JSON model that contains the sap.ui.Device object as data. The `createDeviceModel` function in the `webapp/models/model.js` file instantiates this JSONModel. This function is called from the `init` function in `webapp/Component.js` which then assigns the model to the component itself, using the name "device", so it is available throughout the application.
(Find more information about Device Models in the SDK.)

We'll now use this model to declaratively bind a control property to a characteristics of the device: we'll make the panels expandable when the app is running on a smartphone.

To do this, add the following code, which binds the "expandable" property of the panels to a data property "system/phone" in the device model. Furthermore make the first panel expanded by default and the second one collapsed by default. We can ignore the third panel because we already made it disappear on phones.

webapp/view/Object.view.xml

```
…
      <Panel
         class="sapUiResponsiveMargin"
         width="auto"
         headerText="{i18n>productTitle}"
         expandable="{device>/system/phone}"
         expanded="true">
         <content>
            <form:SimpleForm id="objectForm">
               <form:content>
                  <Label text="{i18n>productCategoryLabel}"/>
                  <Text text="{Category}"/>
                  <Label text="{i18n>productNameLabel}"/>
                  <Text text="{Name}"/>
                  <Label text="{i18n>productWeightLabel}"/>
                  <Text text="{= ${WeightMeasure} + ' ' + ${WeightUnit}}"/>
               </form:content>
            </form:SimpleForm>
         </content>
      </Panel>

      <Panel
         class="sapUiResponsiveMargin"
         width="auto"
         headerText="{i18n>supplierTitle}"
         expandable="{device>/system/phone}"
         expanded="false">
         <content>
…
```

The `sap.ui.Device.system.phone` property does not only depend on the screen size, but also on other characteristics like touch support, so you need to test this either in the Chrome browser's device emulation mode or on an actual smartphone. See the instructions in e.g. unit 3.2 in case you are unsure how to use the Chrome device emulation.

The Product page of the application should then look as depicted in the second screenshot. Note the collapse/expand arrows, which are not present when the same app is running in a normal desktop browser.

Note:
Bindings in this device model are not updated when a property of sap.ui.Device (like e.g. the device orientation) changes. To bind to such a dynamic property, you would need to listen to the orientation change event and update the device model when it occurs.

Related Information

Overview: Adapting to Operating Systems and Devices
Options for Further Adaptations – CSS Classes
API Reference: sap.ui.Device
Using Device Models
Content Densities