Week 3 Unit 2

Defining a Responsive Table

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

Table of Contents

[1 Responsive Table Behavior 2](#_Toc449112011)

[2 Update the Products Table and Header 3](#_Toc449112012)

[3 Add A Column for the Supplier 6](#_Toc449112013)

[4 Add A Column for the Supplier Web Address 8](#_Toc449112014)

[5 Testing the Responsiveness with Chrome 10](#_Toc449112015)

[6 Testing the Responsiveness with SAP Web IDE 13](#_Toc449112016)

## Preview

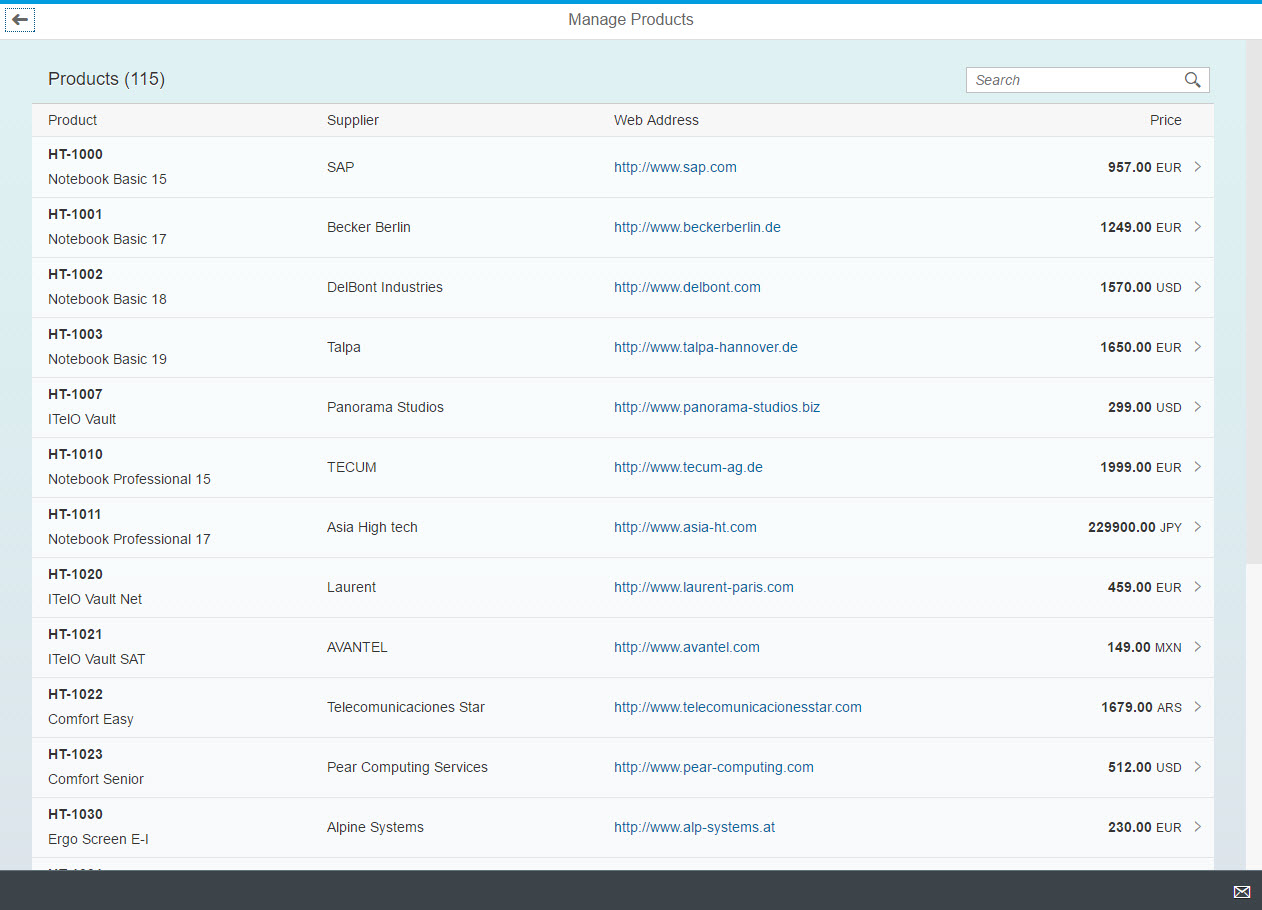


Figure 1 - Preview of the app after doing this unit’s exercises

# Responsive Table Behavior

At first you will enhance the display of products with the product name and include additional information about the product’s supplier:

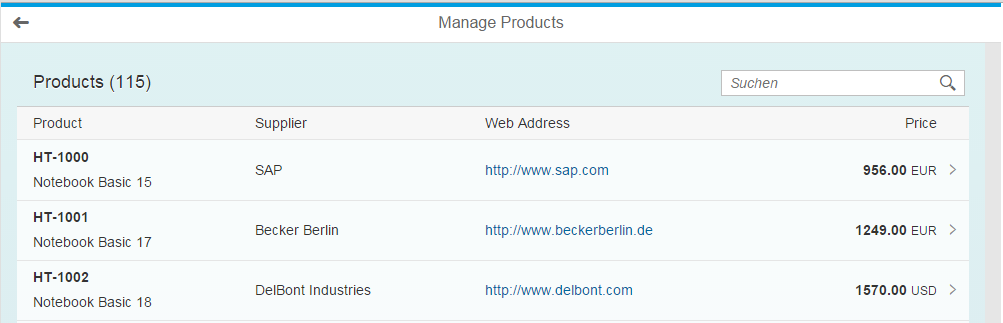


Figure 2 – The table on a desktop device

You will test the responsive behaviour on devices with small screens by simulating this environment in Chrome and Web IDE:

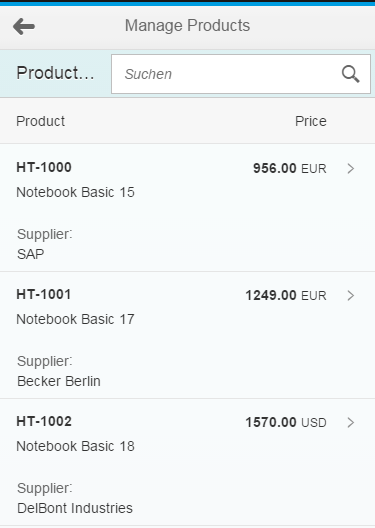


Figure 3 – The table on a phone device in portrait mode

# Update the Products Table and Header

At first you improve the generated code for properly displaying products and their name.

### Preview

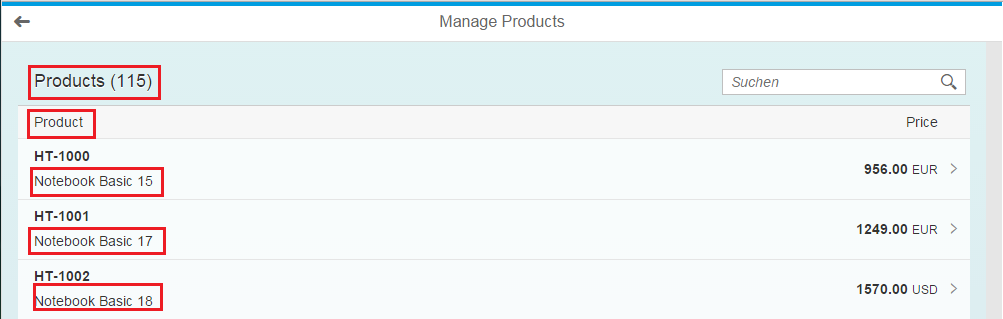


Figure 4 – Adjustments on the worklist view

**Worklist View**

* Update the table title
* Update the first column title
* Show in addition the product’s name

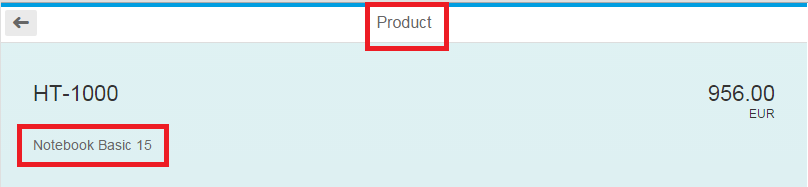


Figure 5 – Adjustments on the object view

**Object View**

* Update the page title
* Show the name as attribute

### webapp/view/Worklist.view.xml

<Table …>

…

<items>

<ColumnListItem

type="Navigation"

press="onPress">

<cells>

<ObjectIdentifier

title="{ProductID}"

**text="{Name}"**/>

<ObjectNumber

number="{

path: 'Price',

formatter: '.formatter.numberUnit'

}"

unit="{CurrencyCode}"/>

</cells>

</ColumnListItem>

</items>

</Table>

Find the ColumnListItem which represents a row of the table. The cells aggregation contains the controls that display the content for the different columns. Extend the first control – an ObjectIdentifer - with a text that is bound to the product name.

### webapp/view/Object.view.xml

<semantic:headerContent>

<ObjectAttribute text="{Name}"/>

<ObjectNumber

number="{

path: 'Price',

formatter: '.formatter.numberUnit'

}"

unit="{CurrencyCode}" />

</semantic:headerContent>

Add an ObjectAttribute control to the headerContent section and bind it to the Name property.

### webapp/i18n/ i18n.properties

…

#~~~ Worklist View ~~~~~~~~~~~~~~~~~~~~~~~~~~

#XTIT: Table view title

worklistViewTitle=Manage Products

#XTIT: Table view title

worklistTableTitle=**Products**

#XTOL: Tooltip for the search field

worklistSearchTooltip=Enter **a product** name or a part of it.

#XBLI: text for a table with no data with filter or search

worklistNoDataWithSearchText=No matching **products** found

#XTIT: Table view title with placeholder for the number of items

worklistTableTitleCount=**Products** ({0})

#XTIT: The title of the column containing the ProductID of ProductSet

tableNameColumnTitle=**Product**

#XTIT: The title of the column containing the Price and the unit of measure

tableUnitNumberColumnTitle=Price

#XBLI: text for a table with no data

tableNoDataText=No **products** are currently available

…

#~~~ Object View ~~~~~~~~~~~~~~~~~~~~~~~~~~

#XTIT: Object view title

objectTitle=**Product**

**…**

#~~~ Not Found View ~~~~~~~~~~~~~~~~~~~~~~~

#YMSG: The ProductSet not found text is displayed when there is no ProductSet with this id

noObjectFoundText=This **Product** is not available

#YMSG: The ProductSet not available text is displayed when there is no data when starting the app

noObjectsAvailableText=No **Products** are currently available

Update the existing i18n properties that have been generated by the template based on the service metadata with these more readable values.

# Add A Column for the Supplier

Add a new column showing the product’s supplier.

### Preview

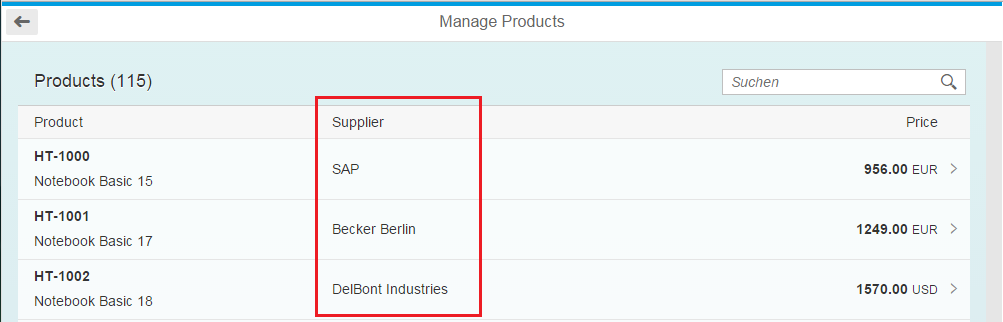


Figure 6 – Adding a “Supplier” column

### webapp/view/Worklist.view.xml

<Table … >

…

<columns>

<Column id="nameColumn">

<Text text="{i18n>tableNameColumnTitle}" id="…"/>

</Column>

**<Column**

**id="supplierNameColumn"**

**demandPopin="true"**

**minScreenWidth="Tablet">**

**<Text text="{i18n>tableSupplierColumnTitle}"/>**

**</Column>**

<Column id="unitNumberColumn" hAlign="End">

<Text text="{i18n>tableUnitNumberColumnTitle}" id="…"/>

</Column>

</columns>

<items>

<ColumnListItem

type="Navigation"

press="onPress">

<cells>

<ObjectIdentifier

title="{ProductID}"

text="{Name}"/>

**<Text text="{SupplierName}"/>**

<ObjectNumber

number="{

path: 'Price',

formatter: '.formatter.numberUnit'

}"

unit="{CurrencyCode}"/>

</cells>

</ColumnListItem>

</items>

</Table>

* Add a new Column between the existing two columns.
  + The column control has an aggregation to the Text control that is used to render the column header.
  + Setting the minScreenWidth to Tablet indicates that the column is only shown on tablets or larger screen.
  + demandPopin will show the cell in-place if the column is not shown.
  + Add a new Text control between the existing two cells and bind it to the SupplierName.

### webapp/i18n/ i18n.properties

…

#~~~ Worklist View ~~~~~~~~~~~~~~~~~~~~~~~~~~

…

#XTIT: The title of the column containing the ProductID of ProductSet

tableNameColumnTitle=Product

#XTIT: The title of the column containing the product's supplier name

tableSupplierColumnTitle=Supplier

…

Add a new i18n property for the supplier column title.

# Add A Column for the Supplier Web Address

You now add a second column that displays the supplier’s web address as a link.

### Preview

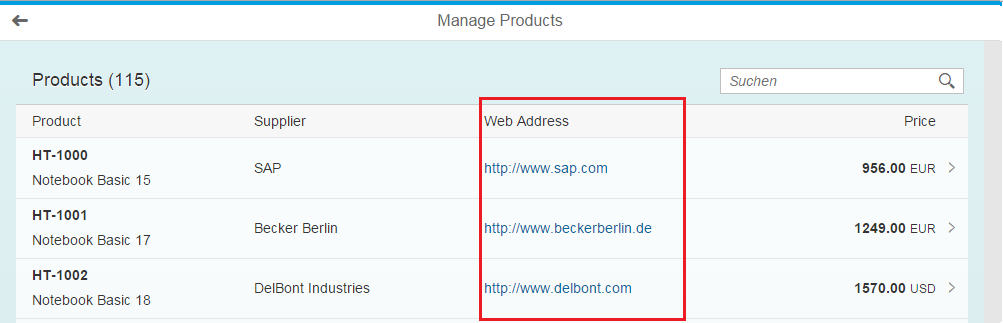


Figure 7 –Adding a “Web Address” Column

### webapp/view/Worklist.view.xml

<Table … >

…

<columns>

<Column id="nameColumn">

<Text text="{i18n>tableNameColumnTitle}" id="…"/>

</Column>

<Column

id="supplierNameColumn"

demandPopin="true"

minScreenWidth="Small">

<Text text="{i18n>tableSupplierColumnTitle}"/>

</Column>

**<Column id="webAddressColumn"**

**demandPopin="false"**

**minScreenWidth="Tablet">**

**<Text text="{i18n>tableSupplierWebAddressColumnTitle}"/>**

**</Column>**

<Column id="unitNumberColumn" hAlign="End">

<Text text="{i18n>tableUnitNumberColumnTitle}" id="…"/>

</Column>

</columns>

<items>

<ColumnListItem

type="Navigation"

press="onPress">

<cells>

<ObjectIdentifier

title="{ProductID}"

text="{Name}"/>

<Text text="{SupplierName}"/>

**<Link**

**text="{ToSupplier/WebAddress}"**

**href="{ToSupplier/WebAddress}"**

**target="\_blank"/>**

<ObjectNumber

number="{

path: 'Price',

formatter: '.formatter.numberUnit'

}"

unit="{CurrencyCode}"/>

</cells>

</ColumnListItem>

</items>

</Table>

* Add the additional Column to the columns aggregation of the Table
* Add a Link control to the cells aggregation of the ObjectListItem. Bind the text and hrefto the path ToSupplier/WebAdress. The ToSupplier/ prefix represents a navigation to a different object in the OData metamodel.

<Table

id="table"

width="auto"

items="{

path: '/ProductSet',

sorter: {

path: 'ProductID',

descending: false

}**,**

**parameters: {**

**expand: 'ToSupplier'**

**}**

}"

noDataText="{worklistView>/tableNoDataText}"

… >

We extend the items aggregation of the table with an expand parameter for the ToSupplier navigation property. With this the supplier data will be already included in the first service request for the products and no additional requests are required.

### webapp/i18n/ i18n.properties

…

#~~~ Worklist View ~~~~~~~~~~~~~~~~~~~~~~~~~~

…

#XTIT: The title of the column containing the ProductID of ProductSet

tableNameColumnTitle=Product

#XTIT: The title of the column containing the product's supplier name

tableSupplierColumnTitle=Supplier

**#XTIT: The title of the column containing the product's supplier web address**

**tableSupplierWebAddressColumnTitle=Web Address**

…

Add a new property for the supplier web address column title.

# Testing the Responsiveness with Chrome

The responsiveness of sap.m.Table can most easily tested by changing the width of the browser window:

| Explanation | **Screenshot** |
| --- | --- |
| 1. Run the application in the Chrome browser and maximize the window. 2. Observe that all columns are shown and that the UI does not grow beyond a certain width. The rest of the space is filled with a darker background color. |  |
| 1. Gradually reduce the width of the browser window. 2. Observe that at a certain threshold 2 columns disappear. The **Supplier** is now shown as a “popin”. The **Web Address** is hidden. |  |

Other controls than sap.m.Table and potentially your application logic might react on other factors than the screen size. Thus for proper testing you should really simulate the device with the Chrome developer tools:

| Explanation | **Screenshot** |
| --- | --- |
| 1. Run the application in the Chrome browser and press the **F12** key to open the developer tools 2. Press the small icon in the left upper corner of the toolbar to **Toggle the device mode**. |  |
| 1. Set the device to **Apple iPhone 5** |  |
| 1. Observe that the **Supplier** column is shown in each row as a “popin”. 2. Observe that the **Web Address** column is hidden. |  |
| 1. Refresh the application’s browser tab. |  |
| 1. End the device mode by pressing on the **Toggle device model** icon again. |  |

# Testing the Responsiveness with SAP Web IDE

We will test the responsiveness of our application with SAP Web IDE’s testing tool that can simulate different screen resolutions and even the device orientation.

| Explanation | **Screenshot** |
| --- | --- |
| 1. Select the folder **ManageProducts** |  |
| 1. Open the **Run** menu 2. Choose **Run Configurations …** |  |
| 1. Select the configuration **App** |  |
| 1. In the **General** tab set the **Preview Mode** to **With Frame** |  |
| 1. Press the **Save and Run** button |  |
| 1. Check that “Small” screen size is selected at the top of the page |  |
| 1. Observe that the **Supplier** column is shown in each row as a “popin”. 2. Observe that the **Web Address** column is hidden. |  |
| **Optional:**   1. Click on the QR-Code icon in the right upper corner. 2. Scan the shown QR code with your smart phone and run the application with a web browser |  |