Week 1 Unit 6

Containers and Layout

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

Table of Contents

[1 Containers 2](#_Toc449370495)

[2 Layout 4](#_Toc449370496)

[3 Icon Library 7](#_Toc449370497)

[4 Margins and Paddings 8](#_Toc449370498)

## Preview

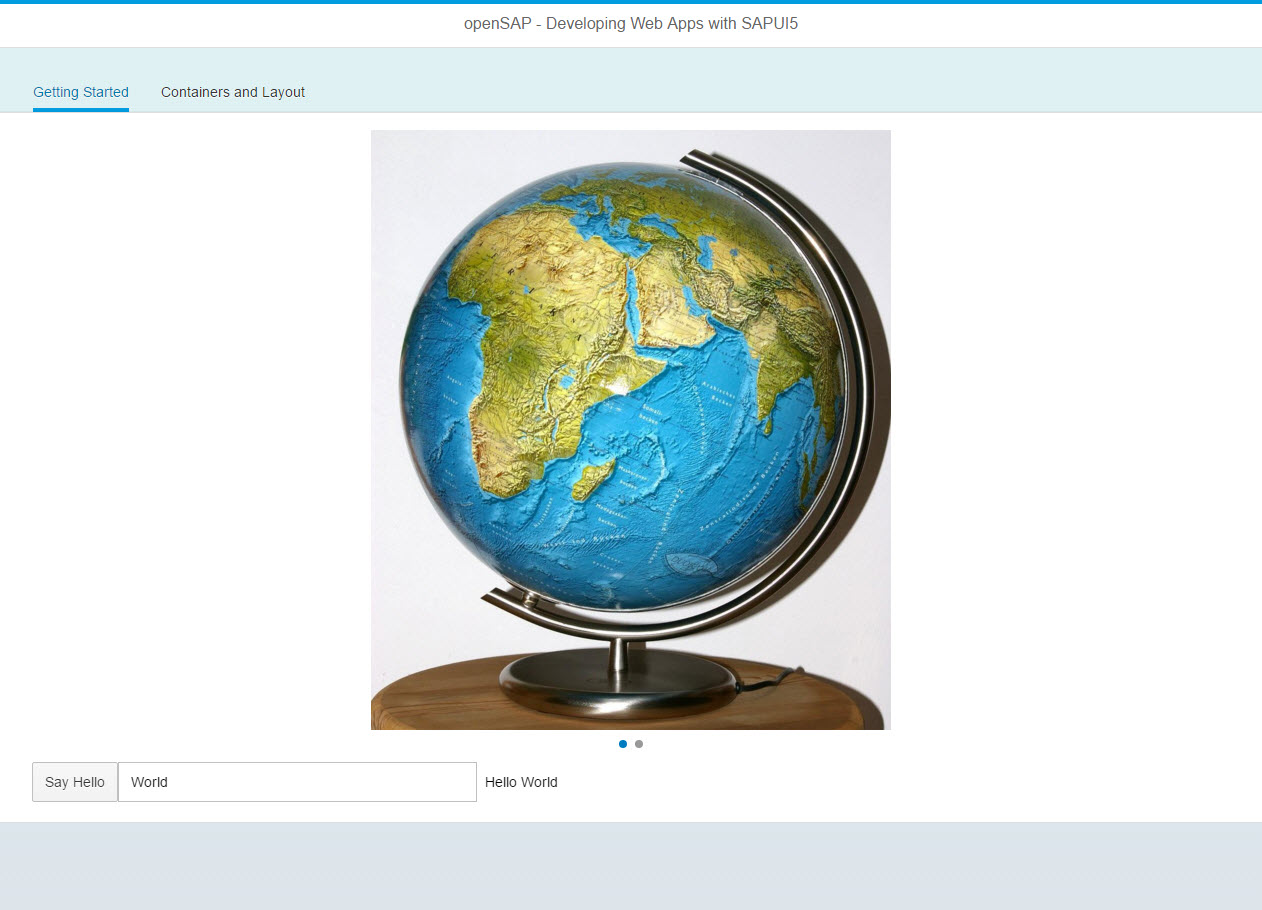


Figure 1 - Preview of the app after doing this unit’s exercises

# Containers

Let’s get started by adding a sap.m.IconTabBar container to our app.

### Preview



Figure 2 - An sap.m.IconTabBar container

### 

### webapp/view/App.view.xml

<mvc:View

displayBlock="true"

xmlns:mvc="sap.ui.core.mvc"

xmlns="sap.m"

controllerName="opensap.myapp.controller.App">

**<App>**

**<pages>**

**<Page title="openSAP - Developing with SAPUI5">**

**<content>**

**<IconTabBar**

**id="idTopLevelIconTabBar">**

**<items>**

**<IconTabFilter id="start" text="{i18n>gettingStartedFilter}">**

<Carousel>

<pages>

<Image

src="https://upload.wikimedia.org/wikipedia/commons/9/9f/GEO\_Globe.jpg"

height="600px"/>

<Image

src="https://upload.wikimedia.org/wikipedia/commons/thumb/8/86/Wikipedia\_Hello\_World\_Graphic.svg/2000px-Wikipedia\_Hello\_World\_Graphic.svg.png"

height="600px"/>

</pages>

</Carousel>

<Button

text="{i18n>showHelloButtonText}"

press="onShowHello"/>

<Input

value="{helloPanel>/recipient/name}"

description="Hello {helloPanel>/recipient/name}"

valueLiveUpdate="true"

width="60%"/>

**</IconTabFilter>**

**</items>**

**</IconTabBar>**

**</content>**

**</Page>**

**</pages>**

**</App>**

</mvc:View>

We add a top level container called sap.m.Page to the view. The page provides an aggregation to 0..N other controls called content. It also displays the title attribute in a header section on top of the content. The page itself is placed into the pages aggregation of another control called sap.m.App which does the following important things for us:

* It writes a bunch of properties into the header of the index.html that are necessary for proper display on mobile devices.
* It offers functionality to navigate between pages with animations. We will use this in a later unit.

Then, we add the IconTabBar to the Page content. We define our tab by adding an IconTabFilter to the IconTabBar’s items. Now we can move Carousel, Button and Input into our IconTabFilter.

We add IDs to both the IconTabBar and the IconTabFilter to make sure that these controls can be found easily by test frameworks.

### webapp/i18n/i18n.properties

…

# Tabs

**gettingStartedFilter=Getting Started**

Add the new texts from the view to the resource bundle.

### Conventions

* Control ids are start lower-case

### Related Information

[SAPUI5 Explored: Search for ‘Container’](https://sapui5.hana.ondemand.com/explored.html)

# Layout

Next, we would like to show a SAPUI5 layout in action.

### Preview

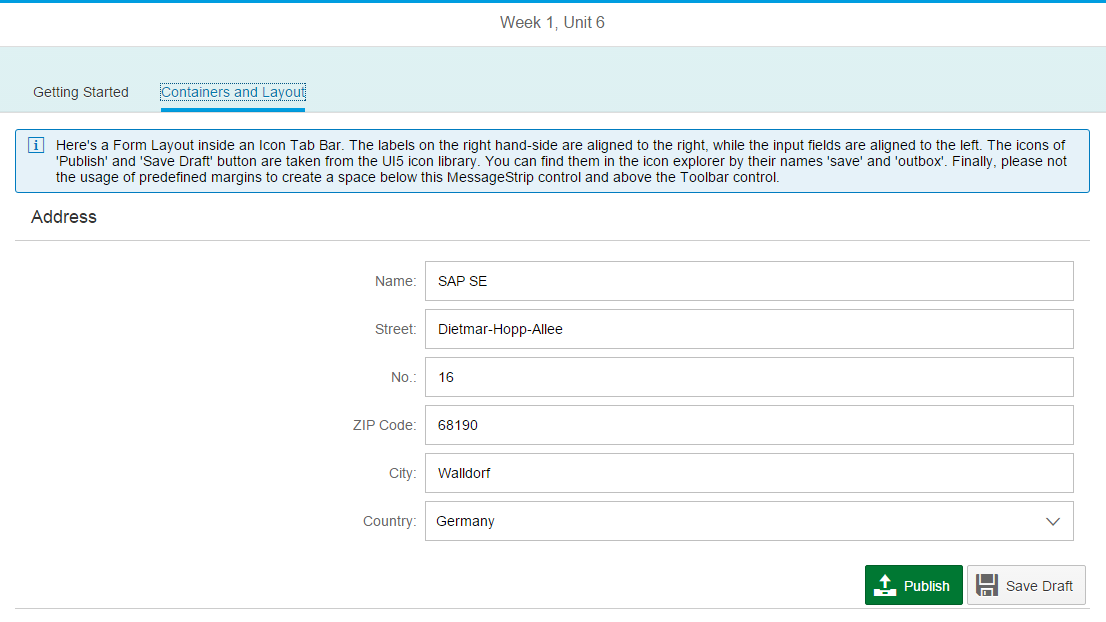


Figure 2: A sap.ui.layout.form.SimpleForm on the second tab

### webapp/view/App.view.xml

<mvc:View

displayBlock="true"

xmlns:mvc="sap.ui.core.mvc"

**xmlns:core="sap.ui.core"**

**xmlns:l="sap.ui.layout"**

**xmlns:f="sap.ui.layout.form"**

xmlns="sap.m"

controllerName="opensap.myapp.controller.App">

<App>

<pages>

<Page title="openSAP - Developing with SAPUI5">

<content>

<IconTabBar

id="idTopLevelIconTabBar"

<items>

<IconTabFilter id="start" text="{i18n>gettingStartedFilter}">

<Carousel>

…

</Carousel>

<Button

text="{i18n>showHelloButtonText}"

press="onShowHello"/>

<Input

value="{helloPanel>/recipient/name}"

description="Hello {helloPanel>/recipient/name}"

valueLiveUpdate="true"

width="60%"/>

</IconTabFilter>

**<IconTabFilter id="layouts" text="{i18n>layoutFilter}">**

**<MessageStrip**

**type="Information"**

**showIcon="true"**

**text="Here's a Form Layout inside an Icon Tab Bar …"/>**

**<f:SimpleForm id="simpleFormChange"**

**title="Address"**

**editable="true"**

**class="editableForm">**

**<f:content>**

**<Label text="{i18n>layoutFormName}"/>**

**<Input value="{address>/Name}"/>**

**<Label text="{i18n>layoutFormStreet}"/>**

**<Input value="{address>/Street}"/>**

**<Label text="{i18n>layoutFormHouseNumber}"/>**

**<Input value="{address>/HouseNumber}"/>**

**<Label text="{i18n>layoutFormZIPCode}"/>**

**<Input value="{address>/ZIPCode}"/>**

**<Label text="{i18n>layoutFormCity}"/>**

**<Input value="{address>/City}" />**

**<Label text="{i18n>layoutFormCountry}"/>**

**<Select>**

**<items>**

**<core:Item text="{i18n>layoutCountryGermany}"/>**

**<core:Item text="{i18n>layoutCountryUSA}"/>**

**<core:Item text="{i18n>layoutCountryEngland}"/>**

**</items>**

**</Select>**

**</f:content>**

**</f:SimpleForm>**

**<Toolbar>**

**<ToolbarSpacer/>**

**<Button**

**text="{i18n>layoutPublishButton}"**

**type="Accept"/>**

**<Button**

**text="{i18n>layoutDraftButton}"/>**

**</Toolbar>**

**</IconTabFilter>**

</items>

</IconTabBar>

</content>

</Page>

</pages>

</App>

</mvc:View>

A prominent example is the sap.ui.layout.form.Form which can be used for displaying Label/Input pairs. In our case, we will use a so-called SimpleForm to display 6 rows of label/input combinations for editing address information, most of which being sap.m.Input controls and a sap.m.Select to help the user with country selection.

We create a new tab ‘Layout’ in our IconTabBar and start it off with a MessageStrip for a descriptive text. Next, we add the SimpleForm and its rows. We finish off our form with a two button toolbar.

### webapp/model/Address.json (NEW)

**{**

**"Name": "SAP SE",**

**"Street": "Dietmar-Hopp-Allee",**

**"HouseNumber": "16",**

**"ZIPCode": "69190",**

**"City": "Walldorf",**

**"Country": "Germany",**

**"Url": "http://www.sap.com",**

**"Twitter": "@sap",**

**"Tel" : "+49 6227 747474",**

**"Email" : "info@sap.com"**

**}**

Add a new JSON file to the model folder for the form data.

### webapp/manifest.json

**…**

**"models"**: {

…

**"address"**: {  
 **"type"**: **"sap.ui.model.json.JSONModel"**,  
 **"uri"**: **"model/Address.json"** }

Declare the “address” model so it becomes accessible within the app.

### webapp/i18n/i18n.properties

…

# Tabs

gettingStartedFilter=Getting Started

**layoutFilter=Containers and Layout**

**# Layout Content**

**layoutFormName=Name**

**layoutFormStreet=Street**

**layoutFormHouseNumber=Nr.**

**layoutFormZIPCode=ZIP Code**

**layoutFormCity=City**

**layoutFormCountry=Country**

**layoutPublishButton=Publish**

**layoutDraftButton=Save Draft**

**layoutCountryGermany=Germany**

**layoutCountryUSA=USA**

**layoutCountryEngland=England**

Add the new texts from the view to the resource bundle.

### Related Information

[SAPUI5 Explored: Search for ‘Layout’](https://sapui5.hana.ondemand.com/explored.html)

# Icon Library

Next we add icons from the UI5 Icon Library to the ‘Publish’ and ‘Save Draft’ button.

### Preview

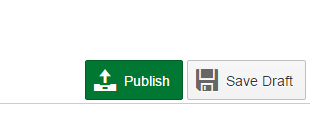


Figure 3: sap.m.Button controls with icons

### webapp/view/App.view.xml

<mvc:View …>

<App>

<pages>

<Page title="openSAP - Developing with SAPUI5">

<content>

<IconTabBar

id="idTopLevelIconTabBar"

<items>

<IconTabFilter id="start" text="{i18n>gettingStartedFilter}">

…

</IconTabFilter>

<IconTabFilter id="layouts" text="{i18n>layoutFilter}">

…

<Toolbar>

<ToolbarSpacer/>

<Button

text="{i18n>layoutPublishButton}"

**icon="sap-icon://outbox"**

type="Accept"/>

<Button

text="{i18n>layoutDraftButton}"

**icon="sap-icon://save"** />

</Toolbar>

</IconTabFilter>

</items>

</IconTabBar>

</content>

</Page>

</pages>

</App>

</mvc:View>

The sap-icon:// protocol is indicating that an icon from the icon font should be loaded. After the sap-icon prefix, simply add the icon name as you found it in the [Icon Explorer](https://sapui5.hana.ondemand.com/iconExplorer.html).

### Related Information

[Developer Guide: Icon and Icon Pool](https://sapui5.hana.ondemand.com/" \l "docs/guide/21ea0ea94614480d9a910b2e93431291.html)

# Margins and Paddings

For the final polish, we add some margins and paddings.

### Preview

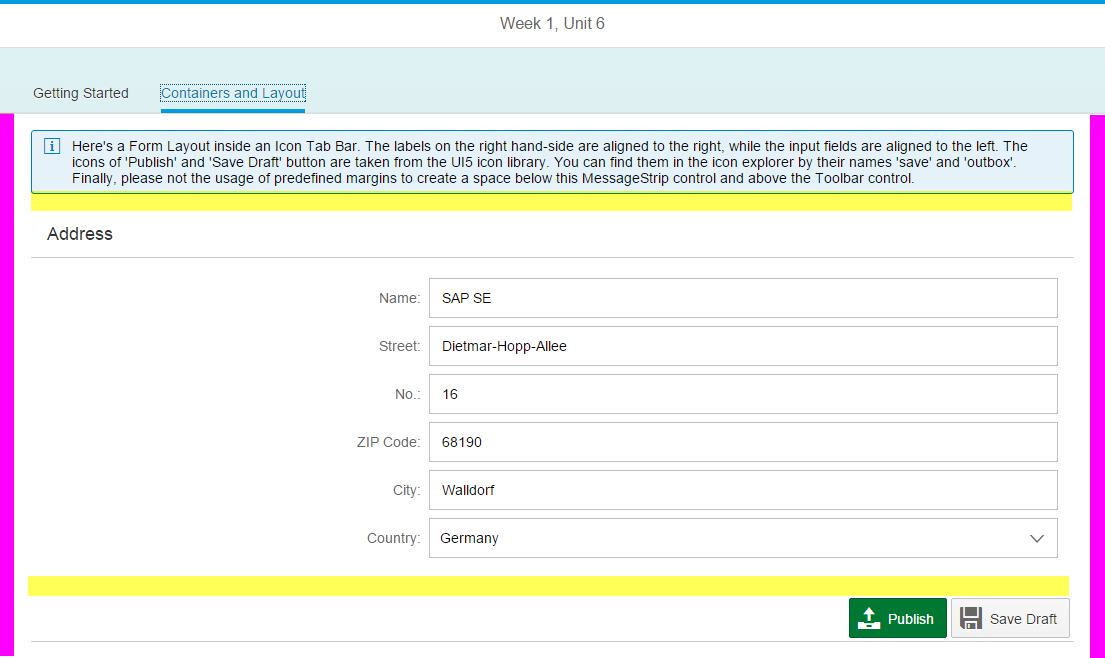


Figure 4: padding (pink) and margins (yellow) in our app

### webapp/view/App.view.xml

<mvc:View …>

<App>

<pages>

<Page title="openSAP - Developing with SAPUI5">

<content>

<IconTabBar

id="idTopLevelIconTabBar"

**class="sapUiResponsiveContentPadding">**

<items>

<IconTabFilter id="start" text="{i18n>gettingStartedFilter}">

…

</IconTabFilter>

<IconTabFilter id="layouts" text="{i18n>layoutFilter}">

…

<f:SimpleForm id="simpleFormChange"

title="Address"

editable="true"

class="editableForm **sapUiSmallMarginTopBottom**">

…

<f:SimpleForm>

…

</IconTabFilter>

</items>

</IconTabBar>

</content>

</Page>

</pages>

</App>

</mvc:View>

We will apply a sapUiResponsiveContentPadding to our IconTabBar to get some more space between the IconTabBar’s content and its border (marked pink in our preview above). In addition, we will add a bottom margin to the MessageStrip and a top margin to the toolbar, to have the form separated more clearly from its neighboring controls (marked bright yellow).

### Related Information

[Developer Guide: Using predefined CSS Margin Classes](https://sapui5.hana.ondemand.com/" \l "docs/guide/777168ffe8324873973151dae2356d1c.html)

[Developer Guide: Using Container Content Padding Classes](https://sapui5.hana.ondemand.com/" \l "docs/guide/c71f6df62dae47ca8284310a6f5fc80a.html)