Week 1 Unit 2

Defining the ui using Controls and Views

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

Table of Contents

[1 View 2](#_Toc449019602)

[2 Controls 3](#_Toc449019603)

## Preview



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

# View

In this step, we will instantiate/create a plain XML view and place it to a node in the DOM.  
  
webapp/view/App.view.xml (NEW)

**<mvc:View**

**xmlns="sap.m"**

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

**</mvc:View>**

Create a new view folder in our app and a new file for our XML view inside the app folder. The root node of the XML structure is the view. Here, we reference the default namespace sap.m where the majority of our UI assets is located. We define an additional sap.ui.core.mvc namespace with alias mvc, where the SAPUI5 views and all other Model-View-Controller (MVC) assets are located.

### webapp/index.html

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta charset="utf-8">

<title>openSAP - Developing Web Apps with SAPUI5</title>

<script

id="sap-ui-bootstrap"

src="https://sapui5.hana.ondemand.com/resources/sap-ui-core.js"

data-sap-ui-theme="sap\_bluecrystal"

data-sap-ui-libs="sap.m"

data-sap-ui-compatVersion="edge"

data-sap-ui-preload="async"

**data-sap-ui-resourceroots='{**

**"opensap.myapp": "./"**

**}'>**

</script>

<script>

sap.ui.getCore().attachInit(function () {

**sap.ui.xmlview({**

**viewName: "opensap.myapp.view.App"**

**}).placeAt("content");**

});

</script>

</head>

<body class="sapUiBody" id="content">

</body>

</html>

In the newly created index.html we instantiate this view in JavaScript into the callback function for the init event of the SAPUI5 runtime. The view is created by a factory function of SAPUI5 which makes sure that the view is correctly configured and can be extended by customers.

The name is prefixed with the namespace opensap.myapp in order to uniquely identify this resource. We define this namespace in the SAPUI5 bootstrap of the runtime. Additionally we add the name of the folder and the name of the file itself to the resource prefix to tell the runtime which file holds the definition of the view to be instantiated.

# Controls

In this step we will add and configure controls to our view declaratively.

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

<mvc:View

**displayBlock="true"**

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

xmlns="sap.m">

**<Carousel>**

**<pages>**

**<Image src="**[**https://placehold.co/600x400/orange/white**](https://placehold.co/600x400/orange/white)**" height="600px"/>**

**<Image src="**[**https://placehold.co/600x400/white/red**](https://placehold.co/600x400/orange/white)**" height="600px"/>**

**</pages>**

**</Carousel>**

</mvc:View>

We will first add ansap.m.Carouselto our view. For this, we can make use of the default namespace sap.m declared in the xml and omit a namespace prefix. To add content into the pages aggregation of the control, we write a tag with this name for the aggregation, followed by the controls that should be added.

We place an image control into our Carousel and configure the control using the properties provided by the control as attributes. For this, we need to give the image a source using the src property, and to ensure a nice display among the two images, we will add an equal height using the height property.

You can easily add additional configuration properties to these controls and look up the available options in the “Explored” application that is part of the SAPUI5 Demo Kit.

### Conventions

• View names are capitalized, e.g. App

• All views are stored in the view folder.

• Names of XML views always end with \*.view.xml.

• The default XML namespace (xmlns) is sap.m.

• Other XML namespaces are referenced by an alias, e.g. xmlns:mvc

### Related Information

[Explored Application](https://sapui5.hana.ondemand.com/explored.html)