

openSAP Evolved Web Apps with SAPUI5

Week 2 Unit 1: Starting Your Journey to Evolved Web Apps

Exercises

PUBLIC



TABLE OF CONTENTS

WEEK INTRODUCTION..... 3

STARTING YOUR JOURNEY TO EVOLVED WEB APPS..... 4

 Create the App from the Template 5

 Evolved UI5 Best Practices 9

 Customize the template17

 Import an image18

 Use the image in the app21

★ CHALLENGE YOURSELF: LINK A MOVIE DATABASE22

RELATED MATERIAL.....23

WEEK INTRODUCTION

Summary

In this week's units you will go through the essentials of building a simple UI5 application. You will get acquainted with:

- The latest best practices for developing web apps
- Using UI controls and control libraries
- Data Binding and usage of models
- Navigation through the views in the app
- How product standards are injected into the framework

Preview

Everybody loves going to the movies! You want to develop an app that gives the users the possibility to show an overview of what's playing according to the movie genre.

Your movie application for movie enthusiasts will contain information about trending movies, and the user can search for a movie they are interested in and see more details about it.

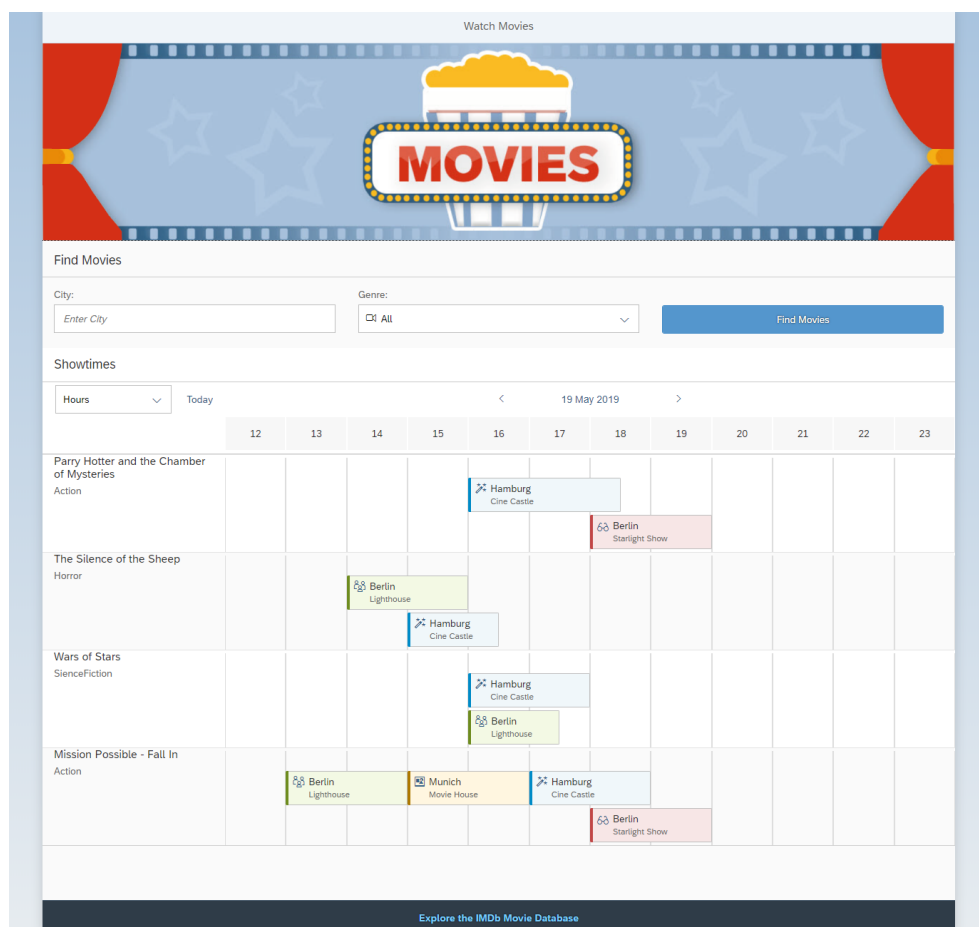


Figure 1 – The final app at the end of the week

STARTING YOUR JOURNEY TO EVOLVED WEB APPS

Summary

In this unit you will learn

- How to start application development with a template in SAP Web IDE
- Which best practices we recommend for UI5 app development
- How to add an image to the app

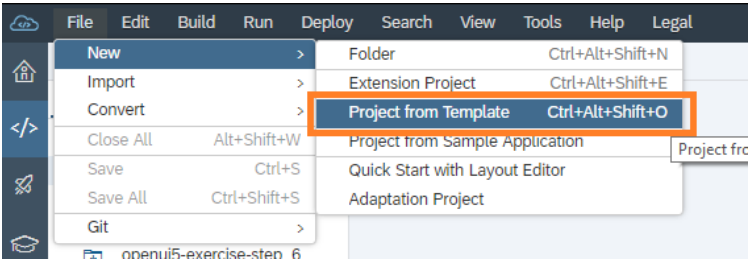
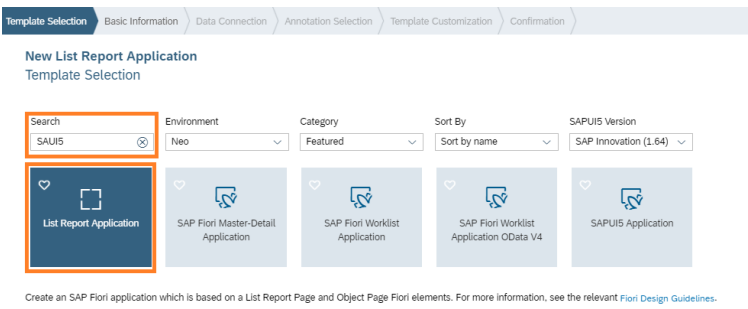
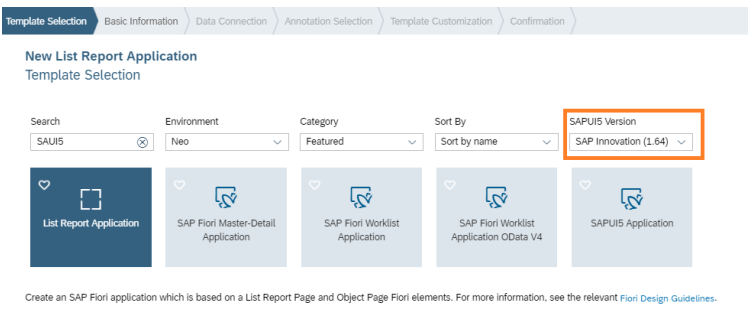

Preview

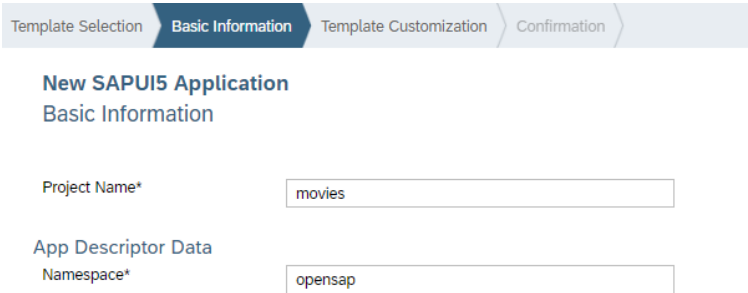

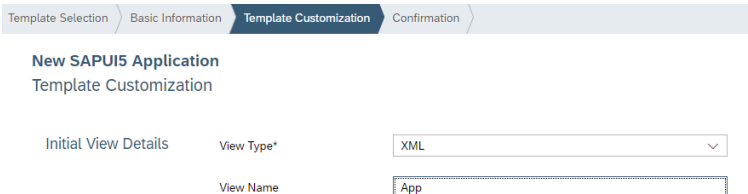
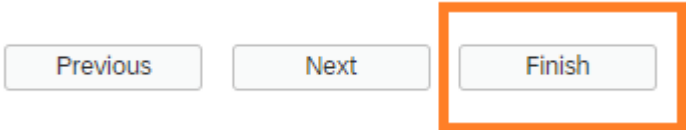
In this unit you will start your application from a template and will apply some of the best practices and tips in the development process with UI5. In the end you will add an image as a banner at the top of the movie app.

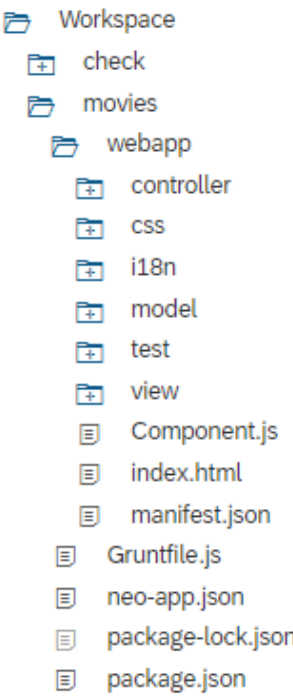
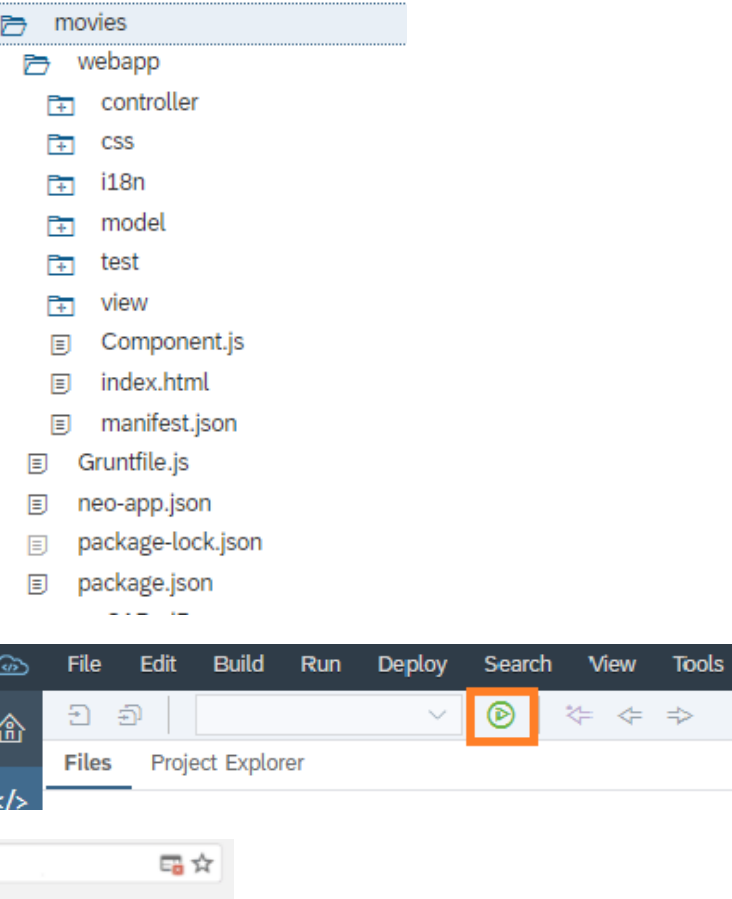


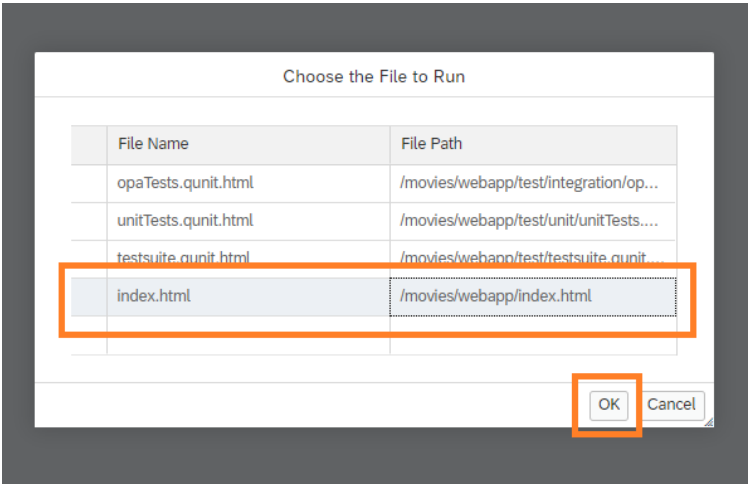
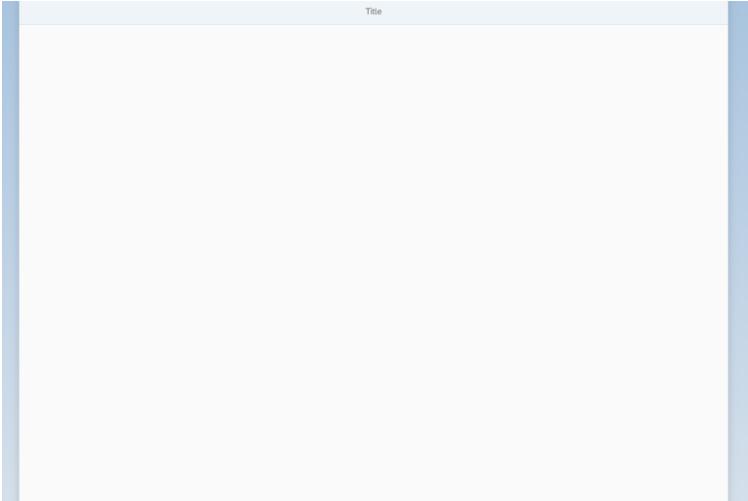
Figure 2 – The SAPUI5 Application template with an additional image

Create the App from the Template

Explanation	Screenshot
<p>1. Choose <i>File</i> → <i>New</i> → <i>Project from Template</i> in your SAP Web IDE workspace to open the wizard with the <i>Template Selection</i> step.</p>	
<p>2. Search for SAPUI5 and select <i>SAPUI5 Application</i> to continue the wizard with this template.</p> <p>Note: If you cannot find this template, select <i>All</i> in the <i>Category</i> dropdown menu.</p>	
<p>3. The latest available SAPUI5 version should be selected by default, for example <i>SAP Innovation (1.64)</i>. If not, select it, it is generally recommended to use the latest version.</p>	
<p>4. Choose <i>Next</i> to go to the next wizard step.</p>	

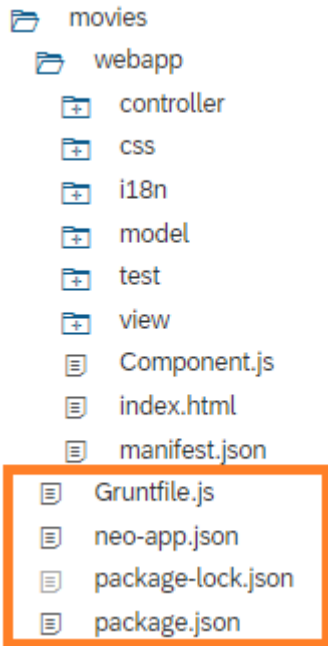
Explanation	Screenshot
<p>5. In the <i>Basic Information</i> step, enter the following values:</p> <ul style="list-style-type: none"> • <i>Project Name</i> = movies • <i>Namespace</i> = opensap 	
<p>6. Choose <i>Next</i> to go to the next wizard step.</p>	
<p>7. In the <i>Template Customization</i> step, enter the following values:</p> <ul style="list-style-type: none"> • <i>View Type</i> = <i>XML</i> • <i>View Name</i> = <i>App</i> <p>Note: Other view types (JSON, JavaScript, HTML) exist, but we generally recommend using XML views to be able to define the view declaratively and clearly separate view and controller logic.</p>	
<p>8. Choose <i>Finish</i> to end the wizard and generate the app.</p>	

Explanation	Screenshot
<p>9. Check out that a new folder – <code>movies</code> – is added to your local workspace. It should contain the files and folders of the initial app as displayed in the screenshot on the right.</p>	
<p>10. Run the app: Select the root folder of the project, and choose the <i>Run</i> icon to run the app.</p> <p>Info: If your browser prevents you from previewing your application, please disable the pop-up blocker by clicking on the blocker icon in the browser search bar.</p>	

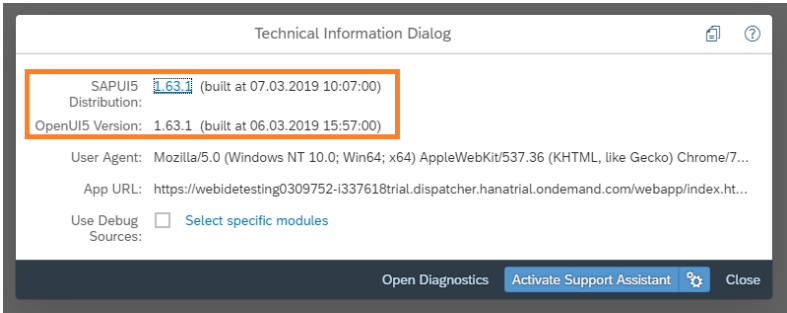
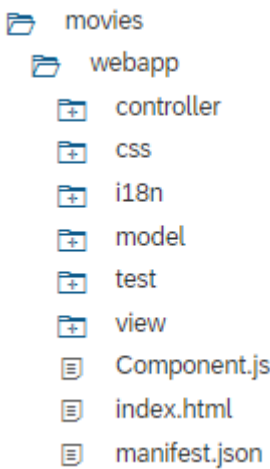


Explanation	Screenshot
11. The first time the app is run, a dialog is shown asking which file should be started. Select the <code>index.html</code> file and choose <i>OK</i> .	
12. A preview of the app opens in a separate browser tab for testing purposes. The app is still empty!	

Evolved UI5 Best Practices

Before you add new features to the app, see which best practices recommended by SAP are already included in the coding and structure of the template. Each template is updated frequently to match the latest recommendations and is set up according to the best practices for app developers.

Explanation	Screenshot
1. Look at the project structure. In the root directory, there are several configuration files.	 <pre>movies webapp controller css i18n model test view Component.js index.html manifest.json Gruntfile.js neo-app.json package-lock.json package.json</pre>

Explanation	Screenshot
<p>2. The <code>neo-app.json</code> file is needed for the SAP Web IDE. It defines some SAP Cloud Platform-specific settings, such as required services for the application.</p>	 <pre> 1 { 2 "welcomeFile": "/webapp/index.html", 3 "routes": [4 { 5 "path": "/resources", 6 "target": { 7 "type": "service", 8 "name": "sapui5", 9 "entryPath": "/resources" 10 }, 11 "description": "SAPUI5 Resources" 12 }, 13 { 14 "path": "/test-resources", 15 "target": { 16 "type": "service", 17 "name": "sapui5", 18 "entryPath": "/test-resources" 19 }, 20 "description": "SAPUI5 Test Resources" 21 } 22], 23 "sendWelcomeFileRedirect": true 24 }</pre>
<p>3. You can change the UI5 version either by manually adding the <code>version</code> field to the <code>neo-app.json</code> file, or by right-clicking on the project and choosing <i>Project</i> → <i>Project Settings</i>.</p> <p>4. Under <i>General</i> → <i>SAPUI5</i>, select a suitable version that you want to ship your app with, if the one selected does not fit your need.</p>	
<p>Note: It is recommended to set a specific UI5 version for your app before making it available to your users. This ensures that your app runs in the environment you tested it in during your development phase.</p>	

Explanation	Screenshot
<p>5. Run the app again, and press Ctrl+Alt+Shift+P to open the <i>Technical Information Dialog</i>. This contains some technical configurations of the app. For example, notice that the UI5 runtime version is the one you selected.</p>	 <p>The screenshot shows the 'Technical Information Dialog' window. It displays the following information:</p> <ul style="list-style-type: none"> SAPUI5: 1.63.1 (built at 07.03.2019 10:07:00) Distribution: (highlighted) OpenUI5 Version: 1.63.1 (built at 06.03.2019 15:57:00) User Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7... App URL: https://webideesting0309752-i337618trial.dispatcher.hanatrial.ondemand.com/webapp/index.ht... Use Debug: <input type="checkbox"/> Select specific modules Sources: <p>Buttons at the bottom: Open Diagnostics, Activate Support Assistant, Close.</p>
<p>6. All the application code is placed in the <code>webapp</code> folder.</p> <p>Info: The content of the <code>webapp</code> folder is typically deployed to a Web server to run the app in the cloud.</p>	 <p>The screenshot shows a file explorer view of the <code>movies</code> project. The <code>webapp</code> folder is expanded, showing the following files and folders:</p> <ul style="list-style-type: none"> controller css i18n model test view Component.js index.html manifest.json
<p>UI5 is bootstrapped relatively in the <code>index.html</code> file, as we can see in the screenshot on the right, and configured in the <code>neo-app.json</code> file as explained above. This allows you to easily update the UI5 version without touching the application code.</p>	 <p>The screenshot shows the <code>index.html</code> file in a code editor. The following code is highlighted:</p> <pre> <script id="sap-ui-bootstrap" src="../../resources/sap-ui-core.js" data-sap-ui-theme="sap_belize" data-sap-ui-resourceroots="{"opensap.movies": "./"}" data-sap-ui-compatVersion="edge" data-sap-ui-oninit="module:sap/ui/core/ComponentSupport" data-sap-ui-async="true" data-sap-ui-frameOptions="trusted"> </script> </pre>
<p>The initial components are defined in a declarative way. Directly executable code is not used in the HTML files, because this makes the files vulnerable. Instead, a good practice is to enable the <code>ComponentSupport</code> module in the bootstrapping script. Then, to declare the desired component in the body via a <code>div</code> tag. This will instantiate the component when the <code>onInit</code></p>	 <p>The screenshot shows the <code>index.html</code> file in a code editor. The following code is highlighted:</p> <pre> <div data-sap-ui-component data-name="opensap.movies" data-id="container" data-settings="{"id": "movies"}"></div> </pre>

Explanation	Screenshot
<p>method is executed. The <code>ComponentSupport</code> class provides functionality which allows you to declare your components in HTML.</p>	
<p>The bootstrapping tag <code>data-sap-ui-async="true"</code> in your <code>index.html</code> file loads the modules for all declared libraries asynchronously. This way the files are retrieved in parallel which speeds up the loading of the background processes and speeds up the whole app too.</p>	 <pre> 1 <!DOCTYPE html> 2 <html> 3 <head> 4 <meta charset="utf-8"> 5 <meta name="viewport" content="width=device-width, initial-scale=1.0"> 6 <title>movies</title> 7 <script id="sap-ui-bootstrap" 8 src="../../resources/sap-ui-core.js" 9 data-sap-ui-theme="sap_belize" 10 data-sap-ui-resourceroots="{ 'opensap.movies': './' }" 11 data-sap-ui-compatversion="edge" 12 data-sap-ui-oninit="module:sap/ui/core/ComponentSupport" 13 data-sap-ui-async="true" 14 data-sap-ui-frameoptions="trusted"> 15 </script> 16 </head> 17 <body class="sapUiBody"> 18 <div data-sap-ui-component data-name="opensap.movies" data-id="container" data-settings="{ 'id' : 'movies' }"></div> 19 </body> 20 </html> </pre>
<p>All UI assets are encapsulated in a component that is instantiated from our <code>index.html</code> page.</p> <p>Components are independent and reusable parts used in UI5 applications. The component configuration is stored in the <code>manifest.json</code> – the so-called application descriptor.</p> <p>Open the <code>Component.js</code> file and take a look at the implementation of the component. Note that the metadata is loaded from a so-called manifest.</p> <p>The <code>init</code> function typically configures additional models that are not defined in the manifest and initializes the router.</p>	 <pre> 1 sap.ui.define([2 "sap/ui/core/UIComponent", 3 "sap/ui/Device", 4 "opensap/movies/model/models" 5], function (UIComponent, Device, models) { 6 "use strict"; 7 8 return UIComponent.extend("opensap.movies.Component", { 9 10 metadata: { 11 manifest: "json" 12 }, 13 14 /** 15 * The component is initialized by UI5 automatically during the startup of the app and calls the init method once. 16 * @public 17 * @override 18 */ 19 init: function () { 20 // call the base component's init function 21 UIComponent.prototype.init.apply(this, arguments); 22 23 // enable routing 24 this.getRouter().initialize(); 25 26 // set the device model 27 this.setModel(models.createDeviceModel(), "device"); 28 } 29 }); 30 }); </pre>



Explanation


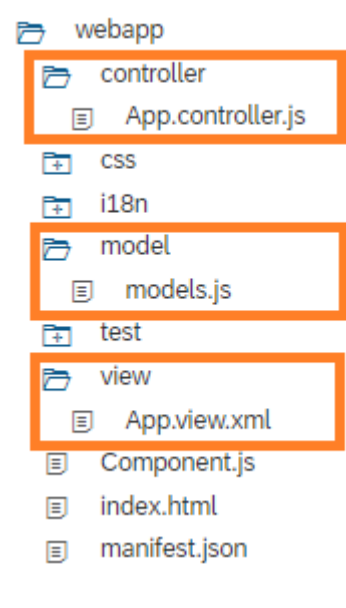
The `manifest.json` or app descriptor file clearly separates the application coding from the configuration settings and makes the app more flexible.


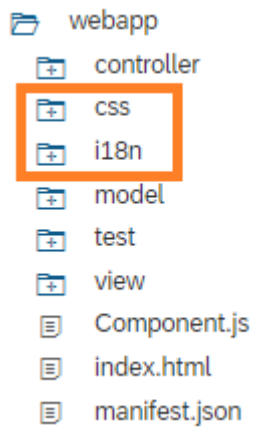
There are two modes – *Descriptor Editor* (default) and *Code Editor*. Both show the same information, but in a different way. Switch to *Code Editor* mode and explore it.

Screenshot

The screenshot displays the SAP Fiori app descriptor editor interface. At the top, there's a tab labeled 'manifest.json'. Below it, a navigation bar contains tabs for 'Settings', 'Data Sources', 'Models', 'Routing', and 'Navigation'. The 'Settings' tab is selected, showing a form with various configuration fields. The form is organized into sections: 'General' (including *Version, *ID, *Type, *Title, *Description, *i18n File Path, *Application Version, and Tags (Keywords)), 'User Interface' (including *Technology, *Devices, and *Themes), and 'Application Icons' (including *Main, *Phone 57 px, and *Phone 112 px). At the bottom of the interface, there are two tabs: 'Descriptor Editor' and 'Code Editor'. The 'Code Editor' tab is highlighted with an orange box, indicating it is the selected mode.

Explanation	Screenshot
<p>You can also use the descriptor file to define application settings, load additional resources, and instantiate models like the <code>i18n</code> resource bundle automatically.</p>	 <pre> 1 { 2 "_version": "1.8.0", 3 "sap.app": { 4 "id": "opensap.movies", 5 "type": "application", 6 "i18n": "i18n/i18n.properties", 7 "applicationVersion": { 8 "version": "1.0.0" 9 }, 10 "title": "{{appTitle}}", 11 "description": "{{appDescription}}", 12 "sourceTemplate": { 13 "id": "ui5Template.basicSAPUI5ApplicationProject", 14 "version": "1.40.12" 15 } 16 }, 17 18 "sap.ui": { 19 "technology": "UI5", 20 "icons": { 21 "icon": "", 22 "favicon": "", 23 "phone": "", 24 "phone@2": "", 25 "tablet": "", 26 "tablet@2": "" 27 }, 28 "deviceTypes": { 29 "desktop": true, 30 "tablet": true, 31 "phone": true 32 }, 33 "supportedThemes": [</pre>
<p>The routing configuration is used to load and show the XML views of the application. It shows a single route to the app view in this project. The connection of the views is accomplished by triggering navigation events and letting the router do the work.</p> <p>Targets are typically referenced in a route and define which view should be displayed when a route was hit. In the routing configuration, you can even add multiple targets for the same route. All the views configured in the respective targets will be instantiated automatically.</p>	 <pre> 60 "settings": { 61 "bundleName": "opensap.movies.i18n.i18n" 62 } 63 }, 64 65 "resources": { 66 "css": [{ 67 "uri": "css/style.css" 68 }] 69 }, 70 71 "routing": { 72 "config": { 73 "routerClass": "sap.m.routing.Router", 74 "viewType": "XML", 75 "async": true, 76 "viewPath": "opensap.movies.view", 77 "controlAggregation": "pages", 78 "controlId": "idAppControl", 79 "clearControlAggregation": false 80 }, 81 "routes": [{ 82 "name": "RouteApp", 83 "pattern": "RouteApp", 84 "target": ["TargetApp"] 85 }], 86 "targets": { 87 "TargetApp": { 88 "viewType": "XML", 89 "transition": "slide", 90 "clearControlAggregation": false, 91 "viewName": "App" 92 } 93 } 94 } 95 } </pre>

Explanation	Screenshot
<p>To improve the performance of your app, you should always load resources asynchronously.</p> <p>The <code>manifest.json</code> contains additional settings for asynchronous loading of the root view and the views instantiated by the routing configuration.</p>	 <pre> 60 "settings": { 61 "bundleName": "opensap.movies.i18n.i18n" 62 }, 63 }, 64 }, 65 "resources": { 66 "css": [{ 67 "uri": "css/style.css" 68 }] 69 }, 70 "routing": { 71 "config": { 72 "routerClass": "sap.m.routing.Router", 73 "viewType": "XML", 74 "async": true, 75 "viewPath": "opensap.movies.view", 76 "controlAggregation": "pages", 77 "controlId": "idAppControl", 78 "clearControlAggregation": false 79 }, 80 "routes": [{ 81 "name": "RouteApp", 82 "pattern": "RouteApp", 83 "target": ["TargetApp"] 84 }], 85 "targets": { 86 "TargetApp": { 87 "viewType": "XML", 88 "transition": "slide", 89 "clearControlAggregation": false, 90 "viewName": "App" 91 } 92 } 93 } 94 } 95 } </pre>
<p>The model-view-controller pattern (MVC) applied in UI5, is reflected in the <code>webapp</code> folder structure:</p> <ul style="list-style-type: none"> The <code>model</code> folder contains additional logic related to data models. Models are used for data management and control filtering, sorting and formatting of data. The <code>view</code> folder contains views and fragments which define the UI of your app. The <code>controller</code> folder contains controllers and helper classes with logic to define the behavior of your views. 	 <pre> webapp ├── controller │ └── App.controller.js ├── css ├── i18n ├── model │ └── models.js ├── test ├── view │ └── App.view.xml ├── Component.js ├── index.html └── manifest.json </pre>

Explanation	Screenshot
<p>Asynchronous loading of dependencies can also be seen in the <code>App.controller.js</code> file included in the template.</p> <p>In the controller file, <code>sap.ui.define</code> is used for asynchronous loading of the controller base class before extending it.</p>	 <p>The screenshot shows a code editor with the file <code>App.controller.js</code> open. The code is as follows:</p> <pre> 1 sap.ui.define([2 "sap/ui/core/mvc/Controller" 3], function (Controller) { 4 "use strict"; 5 6 return Controller.extend("opensap.movies.controller.App", { 7 onInit: function () { 8 // ... 9 } 10 }); 11 }); </pre> <p>An orange rectangle highlights the <code>sap.ui.define</code> function call and its arguments, illustrating asynchronous loading of the <code>sap/ui/core/mvc/Controller</code> dependency.</p>
<p>The <code>css</code> and <code>i18n</code> folders contain the style-related and internationalization-related files.</p>	 <p>The screenshot shows a file explorer view of the <code>webapp</code> directory. The contents are:</p> <ul style="list-style-type: none"> controller css i18n model test view Component.js index.html manifest.json <p>An orange rectangle highlights the <code>css</code> and <code>i18n</code> folders, indicating they contain style-related and internationalization-related files.</p>

Customize the template

Let's change the template a little bit to make it more applicable to the user scenario and change the generated strings from the template to more meaningful ones.

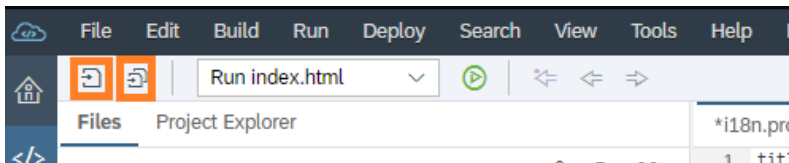
webapp/i18n/i18n.properties

```
title=Title
appTitle=MovieApp
appDescription=App Description

title=Watch Movies
appTitle=Watch Movies
appDescription=Find your favorite movie showtimes
```

Delete the existing definitions of the properties and define the new texts.

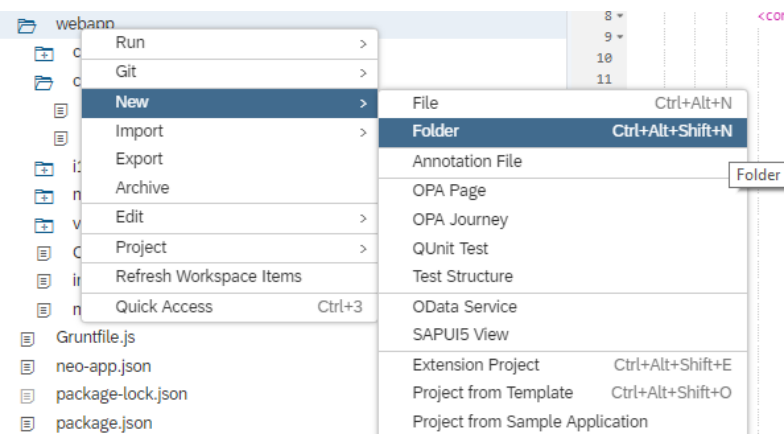
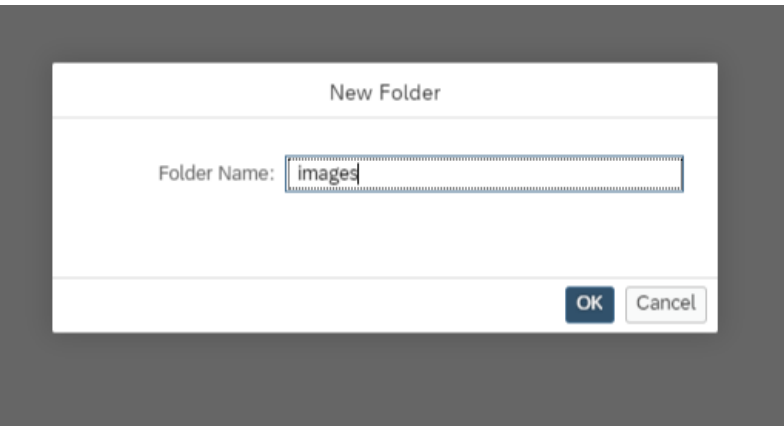
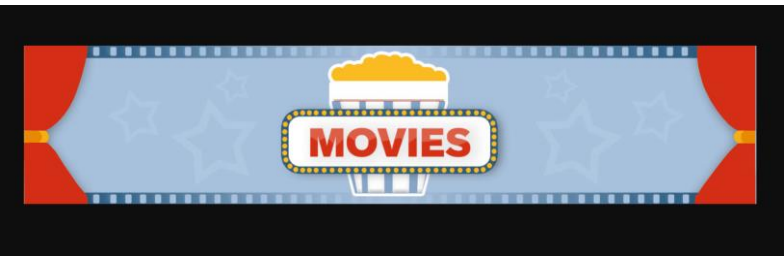
Change the `title` from the title area of the Page to **Watch Movies**. The `appTitle` and `appDescription` strings are used in the `manifest.json` for describing the whole app. Use the same `appTitle` as the `title` and a brief description of the main app purpose as `appDescription`.

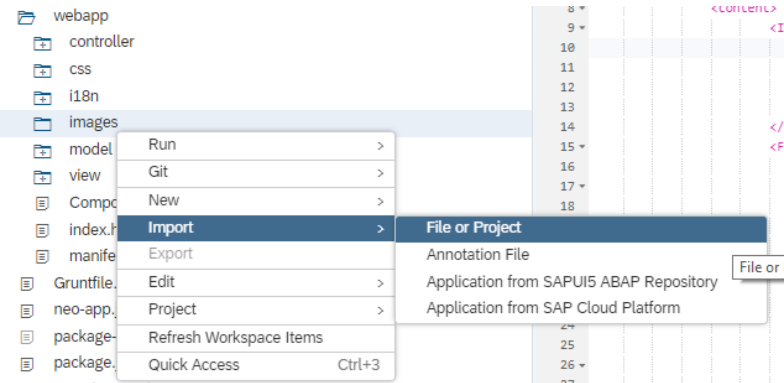
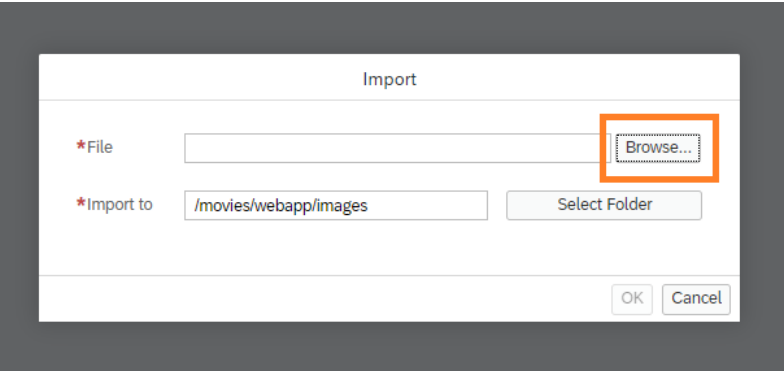
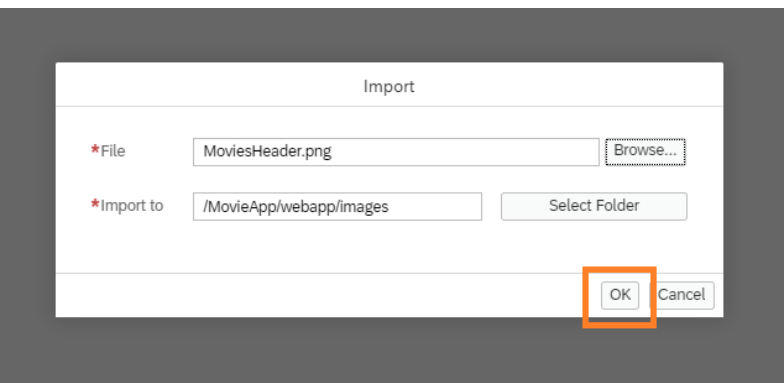
Explanation	Screenshot
<ol style="list-style-type: none">After writing the code, save the changes with one of the buttons at the top of SAP Web IDE. The button on the left is used for saving the current file. The next one is used for saving the whole project. The keyboard shortcuts could be used accordingly too – Ctrl+S and Ctrl+Shift+S.	 The screenshot shows the top toolbar of the SAP Web IDE. It includes a menu bar with File, Edit, Build, Run, Deploy, Search, View, Tools, and Help. Below the menu bar, there are icons for saving the current file (a floppy disk) and saving the whole project (a folder), followed by a dropdown menu showing 'Run index.html' and a green play button icon. The Project Explorer on the right shows a file named '*i18n.pr'.

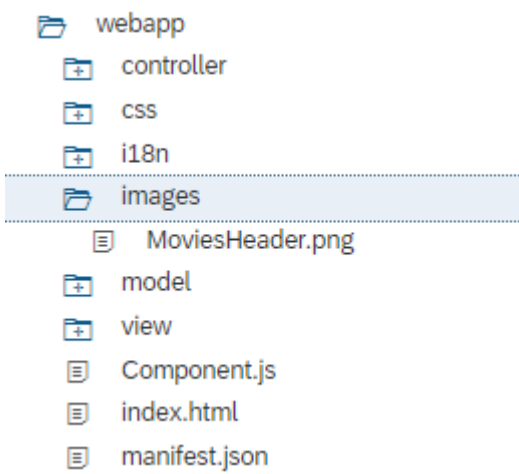
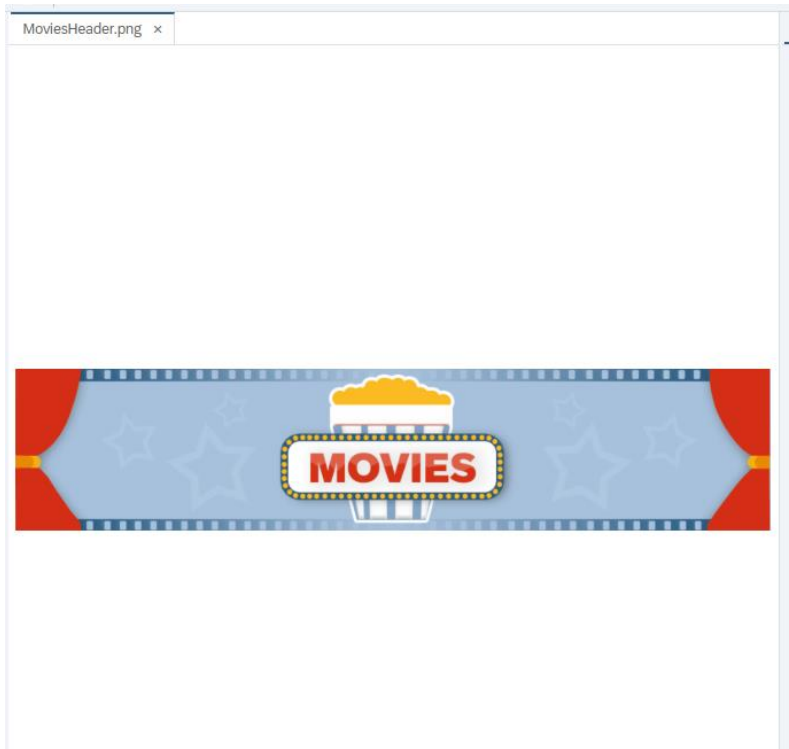
Import an image

As a next step, include an image to the app.

First, you add the image resource file to the project. Then you display it on the page via the UI5 controls `sap.m.Image`.

Explanation	Screenshot
1. First, create a new folder images , where the image resources should be stored. Right-click on the webapp folder. Then choose New → Folder .	
2. Name the folder images and choose OK .	
3. Download the image to be imported from the course repository: https://raw.githubusercontent.com/SAP/openSAP-ui5-course/master/import/MovieHeader.png	

Explanation	Screenshot
<p>4. To import the image to the newly created folder, right-click on the <code>images</code> folder. Then choose <i>Import</i> → <i>File or Project</i>.</p>	
<p>5. A dialog will appear asking for the image destination on your machine and for the path to the folder in the SAP Web IDE project where it will be imported.</p> <p>Choose <i>Browse</i>, and search for the image on your machine. The path to the import destination will automatically be populated with the <code>images</code> folder you just created.</p>	
<p>6. Choose <i>OK</i> button to confirm the action.</p>	
<p>7. The image resource should appear in the <code>images</code> folder.</p>	

Explanation	Screenshot
	
8. Preview the image. It should appear on the right.	

Use the image in the app

Now you add the `sap.m.Image` control. For now, you can think of it as a wrapper around the actual image resource.

webapp/view/App.view.xml

```
...
    <App id="app">
      <pages>
        <Page title="{i18n>title}">
          <content>
            <Image
              src="images/MoviesHeader.png"
              width="100%"
              tooltip="Movie illustration">
            </Image>
          </content>
        </Page>
      </pages>
    </App>
  ...
```

With adding the `sap.m.Image` to the main view, you specify where the source image is contained in the project via the `src` property. You also set a tooltip text via the `tooltip` property of the control.

More information about the settings of each UI5 element will be explained in the next unit. As a reference, you can look at the [Demo Kit](#).

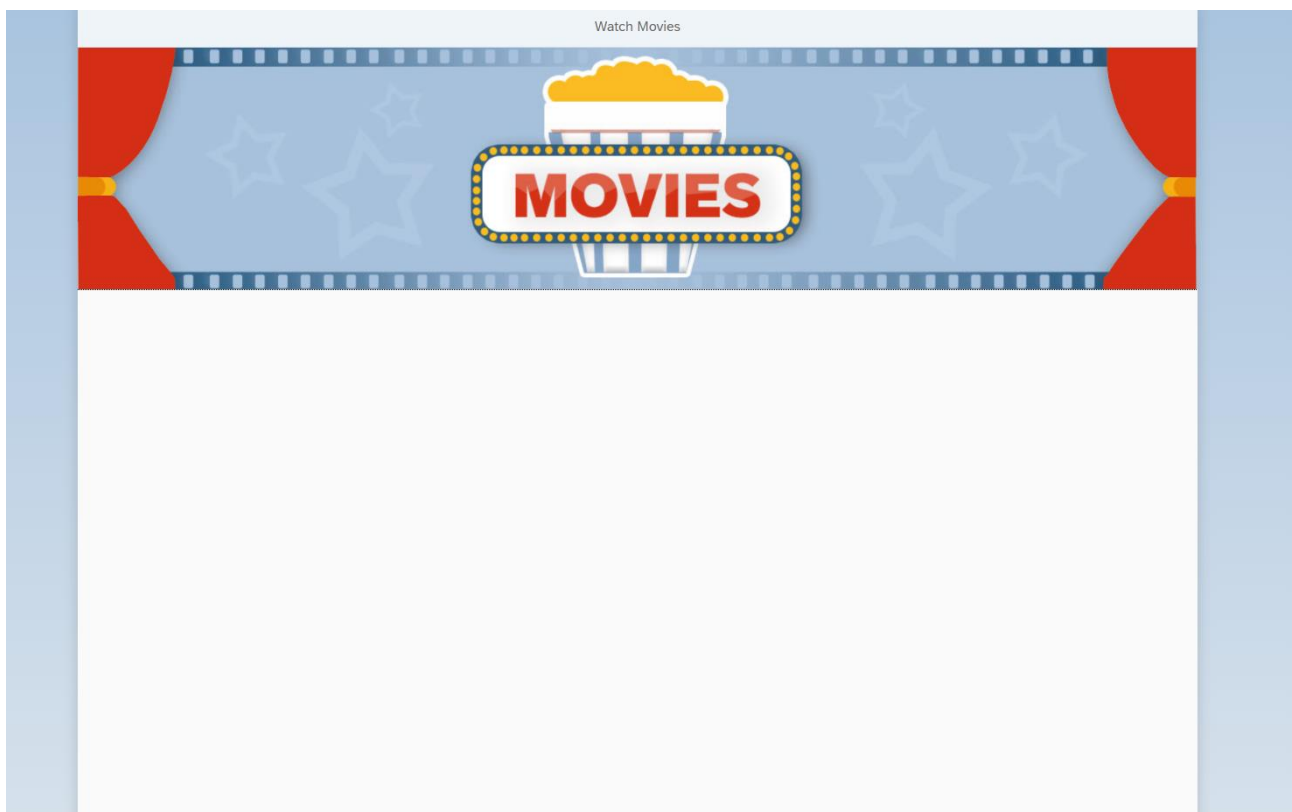


Figure 3 – The result app at the end of the unit

★ CHALLENGE YOURSELF: LINK A MOVIE DATABASE

This task does not come with a predefined solution and can be solved creatively – dive a bit deeper into the topics and exchange with other learners to make the most out of your learning experience. Good luck!

Summary

Early user feedback has shown that a desired feature for the movie app is to have a quick link to a movie database in the footer. That way the users can explore a movie before choosing to reserve seats for it in the cinema. One good answer to this requirement is the **Internet Movie Database** (IMDb).

Can you help increasing the user experience by adding this cool feature?

Details

- **Position:** Footer
- **Text:** Explore Movie Database
- **Link:** [IMDb](https://www.imdb.com/) (should open in a new tab)

Preview

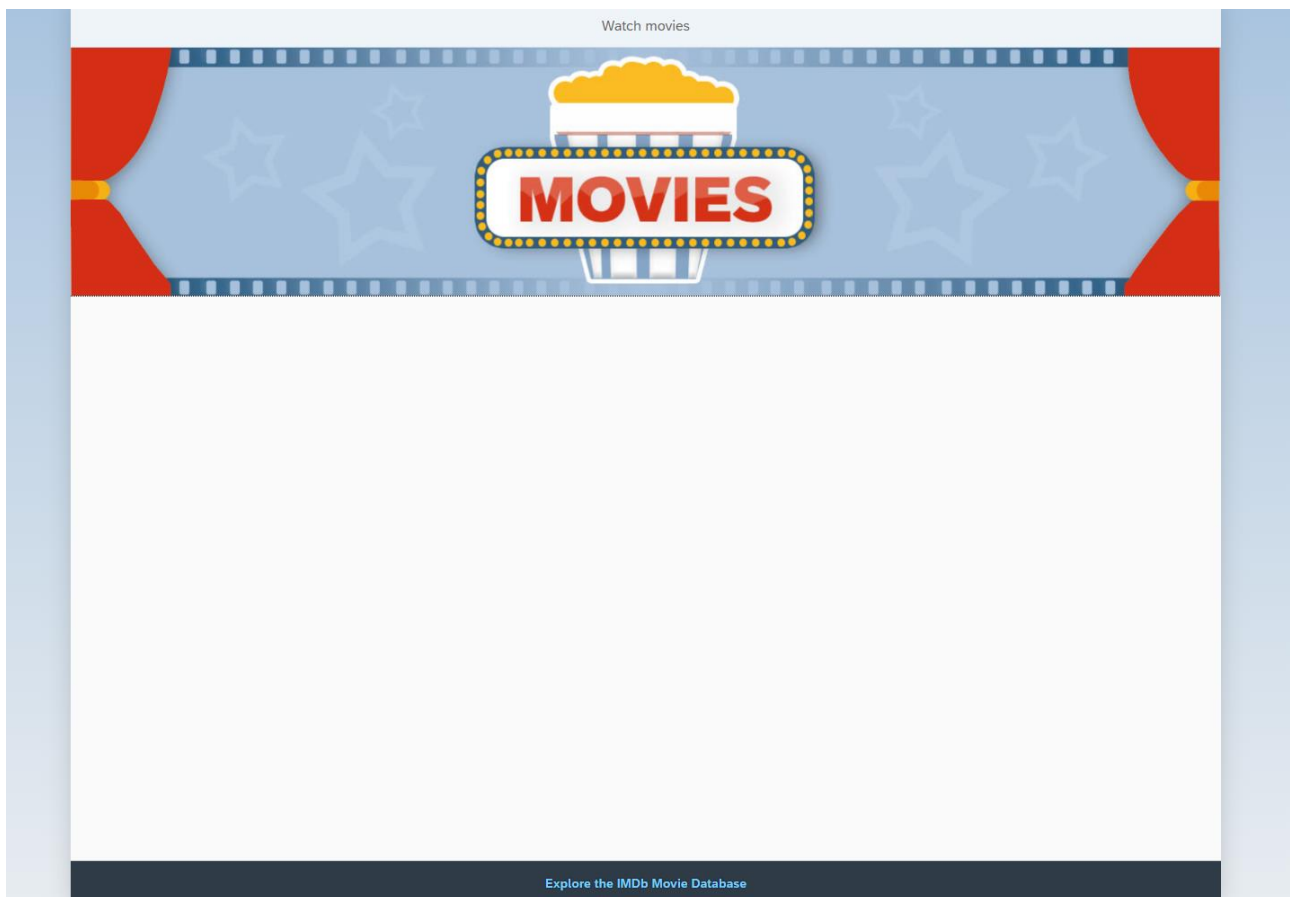


Figure 4 – The footer shows a link to a movie database

Hints:

- `sap.m.Page` has nice examples of using a footer
- `sap.m.Link` might be helpful as well
- Toolbars control the position of its content

RELATED MATERIAL

- [Demo Kit: Controls section](#)
- [Demo Kit: Best Practices for App Developers](#)

Coding Samples

Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a production system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

www.sap.com/contactsap

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.