

openSAP Evolved Web Apps with SAPUI5

Week 2 Unit 3: Binding Data and Using Models

Exercises

PUBLIC



TABLE OF CONTENTS

BINDING DATA AND USING MODELS	3
Add the planning calendar to the app.....	4
Add a new JSON model for movies and appointments.....	5
Convert date and time format for the PlanningCalendar	6
Bind the movie showtimes to the planning calendar	7
Use expression binding to hide movies image on phone devices	8
Move the hard-coded texts for headings and captions to the i18n properties	9
Add the search functionality	12
RELATED MATERIAL.....	13

BINDING DATA AND USING MODELS

Summary

In this unit, you will use data binding to display movie showtimes at various cinemas in a calendar. In addition, you will implement the search function to plan your evening by city and genre. You will also learn how to prepare your app for internationalization and move all hard-coded texts from the previous units to the internationalization properties.

Preview

Watch Movies

The app interface features a header with the text "Watch Movies". Below this is a decorative banner with a blue background, white stars, and a red curtain. In the center of the banner is a yellow and white movie theater marquee with the word "MOVIES" in red. Below the banner is a "Find Movies" section with a "City:" label, a text input field containing "Enter City", a "Genre:" label, a dropdown menu showing "All", and a blue "Find Movies" button. Below this is a "Showtimes" section. It includes a "Hours" dropdown menu, a "Today" label, and a calendar for "May 19, 2019". The calendar has columns for hours from 12PM to 11. Three movies are listed on the left: "Parry Hotter and the Chamber of Mysteries" (Action), "The Silence of the Sheep" (Horror), and "Wars of Stars". Showtimes are displayed as colored blocks: "Parry Hotter and the Chamber of Mysteries" has a blue block for "Hamburg Cine Castle" from 4 to 6 PM; "The Silence of the Sheep" has a green block for "Berlin Lighthouse" from 2 to 4 PM and a blue block for "Hamburg Cine Castle" from 4 to 6 PM; "Wars of Stars" has a red block for "Berlin Starlightshow" from 6 to 8 PM. At the bottom is a dark blue bar with the text "Explore Movie Database".

Find Movies

City:

Genre:

Find Movies

Showtimes

Hours

< May 19, 2019 >

	12PM	1	2	3	4	5	6	7	8	9	10	11
Parry Hotter and the Chamber of Mysteries Action					Hamburg Cine Castle							
The Silence of the Sheep Horror			Berlin Lighthouse		Hamburg Cine Castle		Berlin Starlightshow					
Wars of Stars												

Explore Movie Database

Figure 1 – Movie app with calendar allowing to search for a city and a genre

Add the planning calendar to the app

webapp/App.view.xml

```
<mvc:View
    controllerName="opensap.movies.controller.App"
    displayBlock="true"
    xmlns="sap.m"
    xmlns:mvc="sap.ui.core.mvc"
    xmlns:f="sap.ui.layout.form"
    xmlns:unified="sap.ui.unified">
    <Shell>

...
</f:SimpleForm>
<PlanningCalendar
    id="calendar"
    startDate=""
    rows=""
    appointmentsVisualization="Filled">
    <toolbarContent>
        <Title text="" titleStyle="H3"/>
    </toolbarContent>
    <rows>
        <PlanningCalendarRow
            title=""
            text=""
            appointments="">
            <appointments>
                <unified:CalendarAppointment
                    startDate=""
                    endDate=""
                    title=""
                    text=""
                    icon=""
                    type="">
                </unified:CalendarAppointment>
            </appointments>
        </PlanningCalendarRow>
    </rows>
</PlanningCalendar>
</content>
...
```

Underneath the `SimpleForm`, which contains the controls from the previous unit, add the XML code for the `PlanningCalendar`. Add the `rows` and the `appointments` to the aggregation. These will serve as the template for the data binding later.

The `PlanningCalendarRow` shows:

- o movie name (`title`) and
- o genre (`text`)

The `CalendarAppointment` shows:

- o cinema name (`title`)
- o city info (`text`)
- o icon (`icon`)
- o date and time of the show (`startDate` and `endDate`)

Don't forget to define the `unified` namespace at the very top.
For now, leave the properties empty.

Add a new JSON model for movies and appointments

webapp/model/Movies.json (NEW)

```
{
  "initDate": "05/19/2019/12:0:0",
  "movies": [{
    "name": "Parry Hotter and the Chamber of Mysteries",
    "genre": "Action",
    "appointments": [
      {
        "startDate": "05/19/2019/16:0:0",
        "endDate": "05/19/2019/18:30:0",
        "cinemaName": "Cine Castle",
        "cinemaAddress": "Spree Strasse 110, Hamburg",
        "special": "Sneak Preview with 1 ice cream cone for free.",
        "icon": "sap-icon://activate",
        "seats": "50 seats available",
        "technicalDetails": "Laser projector for 2D and 3D shows in digital 4k quality. High-end audio system with 12 channels and special speaker arrangement for a breathtaking sound experience. Screen size: 22 x 27 m.",
        "info": "Hamburg",
        "type": "Type06",
        "pic": "images/CinemaHamburg.png"
      }
    ]
  }],
  ...
}
```

Download entire file from the import folder in the course's GitHub repository:
<https://raw.githubusercontent.com/SAP/openSAP/develop/import/Movies.json>

We need some movie data in our project, so download the `Movies.json` file from the GitHub repository at <https://raw.githubusercontent.com/SAP/openSAP-UI5-course/master/import/Movies.json>, then go to the `import` folder. After you saved it to your local computer, go back to your SAP Web IDE workspace, right-click on the `model` folder, and choose *Import* → *File or Project*. Then select the `Movies.json` file on your local computer. Afterwards, open the file and make sure that it contains a list of movies.

webapp/manifest.json

```
...
"sap.ui5": {
  "models": {
    "i18n": {
      "type": "sap.ui.model.resource.ResourceModel",
      "settings": {
        "bundleName": "movieapp.MovieApp.i18n.i18n"
      }
    },
    "movies": {
      "type": "sap.ui.model.json.JSONModel",
      "uri": "model/Movies.json"
    }
  },
  "resources": {
    "css": [{
      "uri": "css/style.css"
    }]
  }
},
...
```

In this step we are adding a `JSONModel` to the `models` section in our application descriptor. It will be created automatically at application start and is available under the configured name `movies`.

Convert date and time format for the PlanningCalendar

The `PlanningCalendar` control needs the date and the time as a JavaScript `Date` object. We'll use a method in the formatter file to convert the raw data.

webapp/model/formatter.js (NEW)

```
sap.ui.define([], function () {
    "use strict";

    return {
        formatDate: function (sValue) {
            if (!sValue) {
                return null;
            }
            return new Date(sValue);
        }
    };
});
```

Go to the `model/` folder and create a new file `formatter.js`. The function receives a string value as parameter into `sValue` (the leading "s" indicates a string data type) and converts it via the JavaScript function `Date` to the JavaScript `Date` object.

webapp/App.controller.js

```
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/base/Log",
    "../model/formatter"
], function (Controller, Log, formatter) {
    "use strict";

    return Controller.extend("opensap.movies.controller.App", {
        formatter: formatter,

        onInit: function () {
            ...
        }
    });
});
```

Add the formatter as dependency to the `sap.ui.define` statement of the app controller and instantiate the formatter in the app controller so that the controller can handle the call for our formatter method.

Bind the movie showtimes to the planning calendar

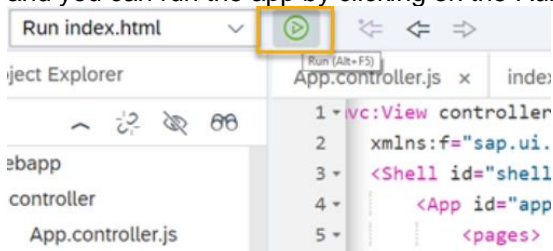
webapp/App.view.xml

```
...
<PlanningCalendar
  id="calendar"
  startDate="{path: 'movies>/initDate', formatter: '.formatter.formatDate'}"
  rows="{movies>/movies}"
  appointmentsVisualization="Filled">
  <toolbarContent>
    <Title text="Showtimes" titleStyle="H3"/>
  </toolbarContent>
  <rows>
    <PlanningCalendarRow
      title="{movies>name}"
      text="{movies>genre}"
      appointments="{path: 'movies>appointments', templateShareable: 'true'}">
      <appointments>
        <unified:CalendarAppointment
          startDate="{path: 'movies>startDate', formatter:
            '.formatter.formatDate'}"
          endDate="{path: 'movies>endDate', formatter:
            '.formatter.formatDate'}"
          title="{movies>info}"
          text="{movies>cinemaName}"
          icon="{movies>icon}"
          type="{movies>type}">
        </unified:CalendarAppointment>
      </appointments>
    </PlanningCalendarRow>
  </rows>
</PlanningCalendar>
...
```

The data binding syntax allows you to make the following assignments:

- The curly brackets indicate the beginning and end of binding info to the browser.
- `movies` is the data model name we set in your manifest file.
- The close-angular-bracket character ">" separates the model name from the binding path to a particular entry.
- The `path` property of the data binding info is necessary when two properties are listed inside the data binding info. This is called "complex binding syntax". You apply the `formatter` and set the `templateShareable` property.
- The `formatter` syntax points to your `formatDate` formatter method in your model folder.

Now, all properties of the `PlanningCalendar` are connected to the respective data from your data model, and you can run the app by clicking on the *Run* button:



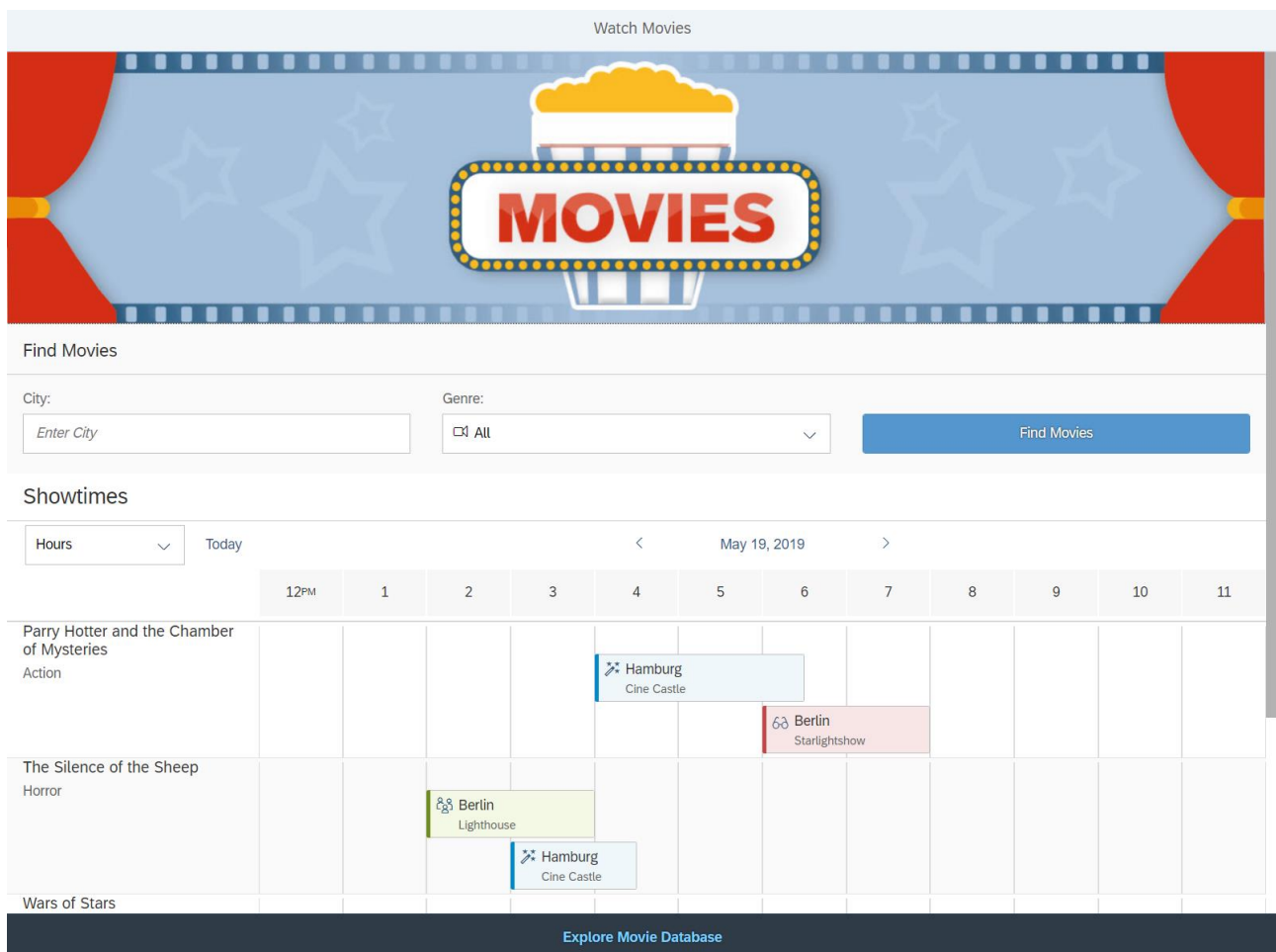


Figure 2 - PlanningCalendar displays movies and presentations in our app by means of data binding

Use expression binding to hide movies image on phone devices

[webapp/App.view.xml](#)

```
...
<App id="app">
  <Page title="{i18n>title}">
    <Image
      visible="{= !${device}>/system/phone} }"
      src="images/MoviesHeader.png"
      width="100%"
      tooltip="{i18n>imageTooltip}"
      press="sap.m.MessageToast.show('Do you feel like going to the movies?')"/>
    <f:SimpleForm
      id="form"
    </f:SimpleForm>
  </Page>
</App>
...
```

You will now use expression binding to hide the movies image for phone devices and show it for desktop and tablet computers.

The `visible` property is bound against the `phone` property of the device model. The logical “not” (!) will negate the returned value from the binding syntax. The result will be `visible=false` for phone devices and `visible=true` for others.

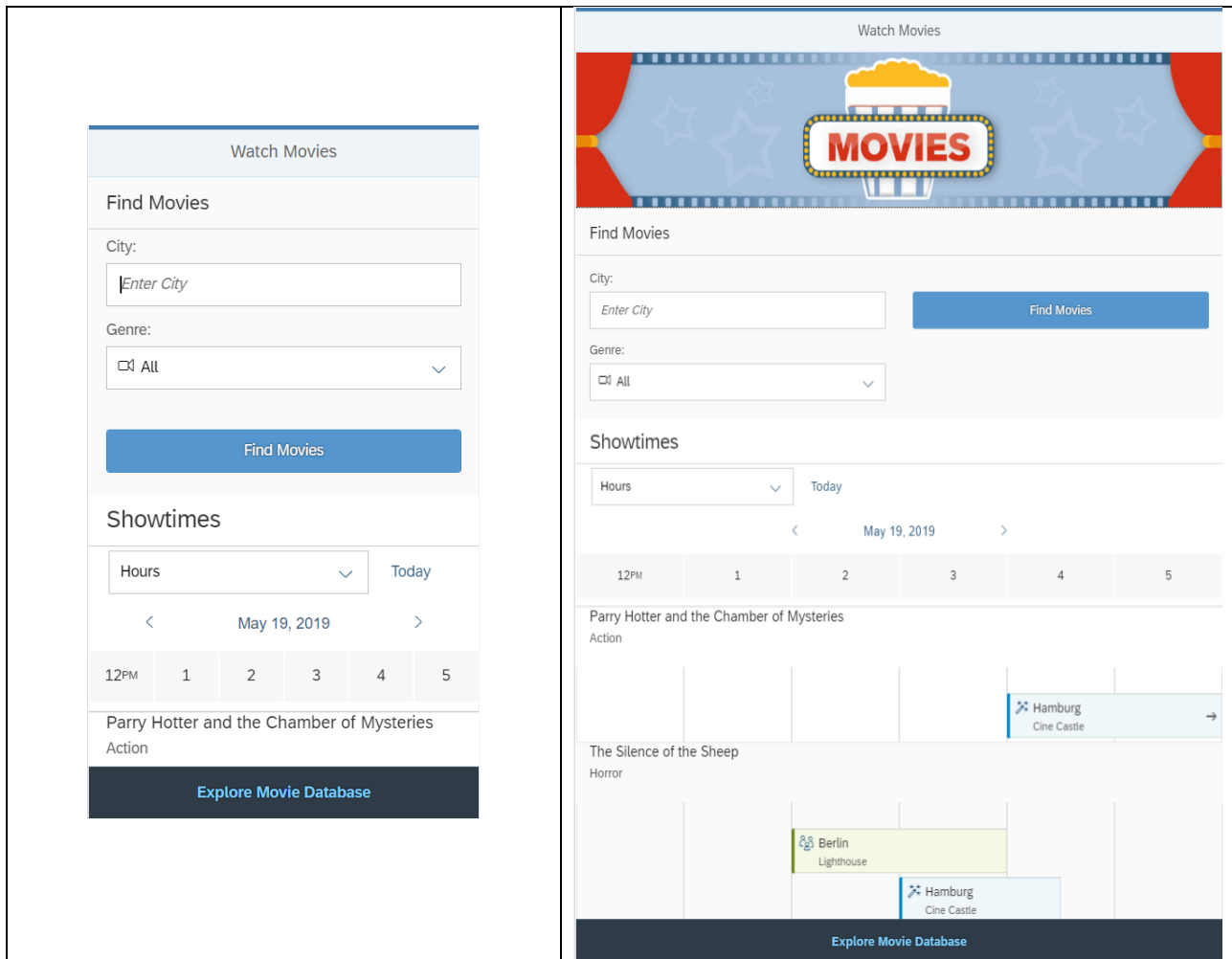


Figure 3 - Movie app in phone and tablet modes

Tip: Device emulation

To test this behavior, you can use the device emulation feature of Google Chrome by clicking on the button in the developer tools. The device model is initialized once at application startup, so you need to reload the page to see the changes in effect once you change the emulated device.

Move the hard-coded texts for headings and captions to the `i18n` properties

Your UI5 application should be ready to be easily translated into other languages. The localization folder - `i18n`, a short name for internationalization – contains the `*.properties`, one for each language. Our template project contains only one file by default. In real life, you would add more `*.properties`, depending on your target languages.

For example, the locale is declared in the file name suffix. The default file name is `i18n.properties`, then another `i18n_de.properties` should contain German localization. You can test it by setting your browser language to German or adding the URL parameter `sap-ui-language=DE_de` to the URL of your app.

Since you based your app on a SAPUI5 Application template, the `i18n` model is defined in the manifest with the name `i18n`. You will move your initially hard-coded texts to this file now based on what you have learned about data binding.

webapp/i18n/i18n.properties

```
#~~~ Home View ~~~~~~

title=Watch Movies
appTitle=MovieApp
appDescription=App Description
calTitle=Showtimes
imageTooltip=Movie illustration
labelCity=City
cityPlaceholder=Enter City
labelGenre=Genre
buttonMovieSearch=Find Movie
calendarTitle=Showtimes
search=Searching...
titleForm=Find Movies
footerLink=Explore the IMDb Movie Database
messageToast=Do you feel like going to the movies?
genreAll=All
genreAction=Action
genreHorror=Horror
genreScienceFiction=Science Fiction
```

Insert a `Home View` section to structure your entries. This improves readability when you have more than one view. Move the hard-coded texts from the `App.xml` file to the `i18n` properties. Hard-coded texts are maintained for the App title, the image tooltip, the labels for the input fields, the placeholder text of the two input fields, the button caption, and the title of the planning calendar. The three entries at the top of the `i18n` properties are pre-set in the template, and we change the title of the app to a more meaningful text. If you now our app again, you will see the updated title.

webapp/App.view.xml

```
...
<Page title="{i18n>title}">
  <Image
    src="images/MoviesHeader.png"
    width="100%"
    tooltip="{i18n>imageTooltip}"
    press="sap.m.MessageToast.show({i18n>messageToast})" />

  <f:SimpleForm
    id="form"
    editable="true"
    layout="ColumnLayout"
    title="{i18n>titleForm}"
    columnsM="2"
    columnsL="3"
    columnsXL="3">
    <f:content>
```

```

...
<Label
  text="{i18n>labelCity}"
  labelFor="city"
/>
  <SearchField
    id="city"
    width="100%"
    showSearchButton="false"
    placeholder="{i18n>cityPlaceholder}" />
  <Label
    text="{i18n>labelGenre}"
    labelFor="genre"
    />
    <Select
      id="genre"
      width="100%">
      <core:ListItem icon="sap-icon://video" key="" text="{i18n>genreAll}" />
      <core:ListItem icon="sap-icon://physical-activity" key="Action"
        text="{i18n>genreAction}" />
      <core:ListItem icon="sap-icon://electrocardiogram" key="Horror"
        text="{i18n>genreHorror}" />
      <core:ListItem icon="sap-icon://paper-plane" key="ScienceFiction"
        text="{i18n>genreScienceFiction}" />
      </Select>

...
  <Button
    text="{i18n>buttonMovieSearch}"
    press="onPress" />
...

<PlanningCalendar
  id="calendar"
  startDate="{path: 'movies>/initDate', formatter: '.formatter.formatDate'}"
  rows="{movies>/Movies}"
  appointmentsVisualization="Filled">
  <toolbarContent>
    <Title text="{i18n>calendarTitle}" titleStyle="H3" />
  </toolbarContent>
  <rows>
...
    <Toolbar>
      <ToolbarSpacer />
      <Link emphasized="true" target="_blank" href="https://www.imdb.com/"
        text="{i18n>footerLink}" />

```

Add the binding info to the controls after moving the text to the i18n properties.

Add the search functionality

Users should be able to search for a city and a genre in our movie app. The result should be displayed in the calendar. Technically, user input is captured in two filters that are then applied to the data model. First, create the filters in the controller.

webapp/App.controller.js

```
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/base/Log",
    "../model/formatter",
    "sap/ui/model/Filter",
    "sap/ui/model/FilterOperator"
], function (Controller, Log, formatter, Filter, FilterOperator) {
    "use strict";

    return Controller.extend("movieapp.MovieApp.controller.App", {

        formatter: formatter,

        onInit: function () {
            Log.info("Controller has been initialized.");
        },

        onPress: function (sValue) {
            sap.ui.require(["sap/m/MessageToast"], function (oMessage) {
                var oResourceBundle =
                    this.getOwnerComponent().getModel("i18n").getResourceBundle();
                oMessage.show(oResourceBundle.getText("search") + sValue
            );
            }.bind(this));

            var sCity = this.byId('city').getValue(),
                sGenre = this.byId('genre').getSelectedItem().getKey(),
                oCalendar = this.byId("calendar"),
                oRowBinding = oCalendar.getBinding("rows"),
                oFilterGenre,
                oFilterCity;

            // Create filters for genre and city according to user inputs
            oFilterGenre = sGenre ? new Filter("genre", FilterOperator.EQ, sGenre) : null;
            oFilterCity = sCity ? new Filter("info", FilterOperator.Contains, sCity) :
            null;

            // Apply genre filter to calendar rows
            oRowBinding.filter(oFilterGenre);

            // Apply city filter to row appointments
            var aRows = oCalendar.getAggregation("rows");
            aRows.forEach(function (oItem) {
                var oAppointmentsBinding = oItem.getBinding("appointments");
                oAppointmentsBinding.filter(oFilterCity);
            });
        }
    });
    ...
});
```

You add code for the `onPress` event that gets fired when the *Find Movie* button is chosen. You collect the two string values the user entered into variables `sCity` and `sGenre`. You collect your `PlanningCalendar` in a variable by its ID `calendar` to derive the data binding information for the rows. This allows you to filter the rows by `genre` directly. For the appointments, you need to apply the filter in a loop with `forEach`.

When you create the filters, you check for empty variables. If the user left an input field empty, you need to apply the filters with a null value, so that the data gets refreshed. If you don't do this, filter settings from previous searches will not get overwritten, and the user will continue to see filtered data in your app, even if you have deleted the filter.

Job done!

The screenshot shows a mobile app interface for watching movies. At the top, there's a header 'Watch Movies' and a large 'MOVIES' sign with a film reel. Below this is a 'Find Movies' section with a 'City' input field (placeholder 'Enter City'), a 'Genre' dropdown menu (selected 'Action'), and a 'Find Movies' button. The 'Showtimes' section below features a date selector set to 'May 19, 2019' and a time slot grid from 12PM to 11PM. Two movies are listed: 'Parry Hotter and the Chamber of Mysteries' (Action) and 'Mission Possible - Fall In' (Action). The showtimes for these movies are displayed as colored bars across the time slots, with specific theater names like 'Hamburg Cine Castle', 'Berlin Starlightshow', and 'Munich Movie House' indicated. A dark blue bar at the bottom contains the text 'Explore Movie Database'.

Figure 4 - app showing filter result for action movies

In the next unit, you will learn how to add another view and how to configure navigation.

RELATED MATERIAL

- [Demo Kit: Data Binding Tutorial](#)
- [Demo Kit: Planning Calendar Samples](#)
- [Demo Kit: Expression Binding](#)

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.