# LOE

**Lösungen – Tag 01**

**zum Seminar 16961 (SAPUI und Fiori)**

##### Lösungen 3

##### Aufgabe 1 – Schneller Einstieg in SAPUI5 3

* 1. Vorlage 3
  2. Definition eines neuen Sliders & Event und Popup-Meldung 3
  3. Mehrere Slider 4

##### Aufgabe 2 – SAPUI5 Application in Eclipse 5

* 1. MVC für Motivationsregler 5
  2. View in XML mit Slider 7

##### Aufgabe 3 – Data Binding 8

##### Aufgabe 4 – SAPUI5 sap.ui.Table Control 11

##### Aufgabe 5 – Fiori-like App (Splitapp) 13

##### Aufgabe 6 – Fiori-like App (Tiles) 16

1. **Lösungen**
2. **Aufgabe 1 – Schneller Einstieg in SAPUI5**

#### Vorlage

<html>  
<head>  
<title>Aufgabe 1: "Motivationsregler"</title>  
<script   
src="https://ui5.sap.com/resources/sap-ui-core.js"   
id="sap-ui-bootstrap"  
data-sap-ui-libs="sap.m" data-sap-ui-theme="sap\_horizon">  
</script>

</head>

<body class="sapUiBody">

<div id="content"></div>

</body>

</html>

#### Definition eines neuen Sliders & Event und Popup-Meldung

<html>

<head>

<title>Aufgabe 1: "Motivationsregler"</title>

<script

src="https://ui5.sap.com/resources/sap-ui-core.js"

id="sap-ui-bootstrap"

data-sap-ui-libs="sap.m" data-sap-ui-theme="sap\_horizon">

</script>

<script>

var oSlider1 = new sap.m.Slider({

id: 'Motivationsslider', width: '300px',

change: onSliderChange

});

function onSliderChange(){

alert('you slided:' + oSlider1.getValue());

};

oSlider1.placeAt("content");

</script>

</head>

<body class="sapUiBody">

<div id="content"></div>

</body>

</html>

#### Mehrere Slider

<html>

<head>

<title>Aufgabe 1: "Motivationsregler"</title>

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

data-sap-ui-libs="sap.ui.commons" data-sap-ui-theme="sap\_bluecrystal">

</script>

<script>

var oSlider1 = new sap.ui.commons.Slider('Slider1'); oSlider1.setTooltip('Slider in Javascript'); oSlider1.setWidth('300px');

oSlider1.placeAt('content'); oSlider1.attachChange( onSlide );

var oSlider2 = new sap.ui.commons.Slider('Slider2'); oSlider2.setTooltip('Slider in Javascript'); oSlider2.setWidth('500px');

oSlider2.placeAt('content'); oSlider2.attachChange( onSlide );

function onSlide(oControlEvent){

alert('you '+ oControlEvent.getSource().getId()+

' slided:' + oControlEvent.getSource().getValue());

};

</script>

</head>

<body class="sapUiBody">

<div id="content"></div>

<h1 >überschrift</h1>

</body>

</html>

### Aufgabe 2 – SAPUI5 Application in Eclipse

#### MVC für Motivationsregler

##### Index.html

<!DOCTYPE HTML>

<html>

<head>

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

<meta http-equiv=*'Content-Type'* content=*'text/html;charset=UTF-8'*/>

<script src=*"resources/sap-ui-core.js"*

id=*"sap-ui-bootstrap"*

data-sap-ui-libs=*"sap.ui.commons"* data-sap-ui-theme=*"sap\_bluecrystal"*>

</script>

<!-- add sap.ui.table,sap.ui.ux3 and/or other libraries to 'data-sap-ui- libs' if required -->

<script>

sap.ui.localResources("zui5\_desktop");

**var** view = sap.ui.view({id:"idmotivation1", viewName:"zui5\_desktop.motivation", type:sap.ui.core.mvc.ViewType.JS});

view.placeAt("content");

</script>

</head>

<body class=*"sapUiBody"* role=*"application"*>

<div id=*"content"*></div>

</body>

</html>

##### motivation.view.js

sap.ui.jsview("zui5\_desktop.motivation", {

/\*\* Specifies the Controller belonging to this View.

* In the case that it is not implemented, or that "null" is returned, this View does not have a Controller.
* **@memberOf** zui5\_desktop.motivation

\*/

getControllerName : **function**() {

**return** "zui5\_desktop.motivation";

},

/\*\* Is initially called once after the Controller has been instantiated. It is the place where the UI is constructed.

* Since the Controller is given to this method, its event handlers can be attached right away.
* **@memberOf** zui5\_desktop.motivation

\*/

createContent : **function**(oController) {

**var** oSlider1 = **new** sap.ui.commons.Slider(

{

id : "Motivationsregler", width : "300px",

//change : onSliderchange

}

);

}

});

oSlider1.attachChange(oController.onSliderchange);

**return** oSlider1;

##### motivation.controller.js

sap.ui.controller("zui5\_desktop.motivation", { onSliderchange : **function**(oControlEvent) {

alert("you slided " + oControlEvent.getSource().getValue() );

},

/\*\*

* + - Called when a controller is instantiated and its View controls (if available) are already created.
    - Can be used to modify the View before it is displayed, to bind event handlers and do other one-time initialization.
    - **@memberOf** zui5\_desktop.motivation

\*/

// onInit: function() {

//

// },

/\*\*

* + - Similar to onAfterRendering, but this hook is invoked before the controller's View is re-rendered
    - (NOT before the first rendering! onInit() is used for that one!).
    - **@memberOf** zui5\_desktop.motivation

\*/

// onBeforeRendering: function() {

//

// },

/\*\*

* + - Called when the View has been rendered (so its HTML is part of the document). Post- rendering manipulations of the HTML could be done here.
    - This hook is the same one that SAPUI5 controls get after being rendered.
    - **@memberOf** zui5\_desktop.motivation

\*/

// onAfterRendering: function() {

//

// },

/\*\*

* + - Called when the Controller is destroyed. Use this one to free resources and finalize activities.
    - **@memberOf** zui5\_desktop.motivation

\*/

// onExit: function() {

//

// }

});

#### View in XML mit Slider

##### xmlslider.html

<!DOCTYPE HTML>

<html>

<head>

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

<meta http-equiv=*'Content-Type'* content=*'text/html;charset=UTF-8'*/>

<script src=*"resources/sap-ui-core.js"*

id=*"sap-ui-bootstrap"*

data-sap-ui-libs=*"sap.ui.commons"* data-sap-ui-theme=*"sap\_bluecrystal"*>

</script>

<!-- add sap.ui.table,sap.ui.ux3 and/or other libraries to 'data-sap-ui- libs' if required -->

<script>

sap.ui.localResources("zui5\_desktop");

**var** view = sap.ui.view({id:"idmotivation1", viewName:"zui5\_desktop.xmlslider", type:sap.ui.core.mvc.ViewType.XML});

view.placeAt("content");

</script>

</head>

<body class=*"sapUiBody"* role=*"application"*>

<div id=*"content"*></div>

</body>

</html>

##### xmlslider.controller.js

sap.ui.controller("zui5\_desktop.xmlslider", { onSliderchange : **function**(oControlEvent) {

alert("you slided " + oControlEvent.getSource().getValue() );

});

},

##### xmlslider.view.xml

<core:View xmlns:core=*"sap.ui.core"* xmlns:mvc=*"sap.ui.core.mvc"* xmlns=*"sap.ui.commons"* controllerName=*"zui5\_desktop.xmlslider"*

xmlns:html=[*"http://www.w3.org/1999/xhtml"*](http://www.w3.org/1999/xhtml)>

<Slider id = *"Slider1"*

tooltip = *"Slider* *in* *XML"*

width = *"300px"*

change = *"onSliderchange"*

/>

</core:View>

### Aufgabe 3 – Data Binding

##### Index.html

<!DOCTYPE HTML>

<html>

<head>

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

<meta http-equiv=*'Content-Type'* content=*'text/html;charset=UTF-8'*/>

<script src=*"resources/sap-ui-core.js"*

id=*"sap-ui-bootstrap"*

data-sap-ui-libs=*"sap.ui.commons"* data-sap-ui-theme=*"sap\_bluecrystal"*>

</script>

<!-- add sap.ui.table,sap.ui.ux3 and/or other libraries to 'data-sap-ui- libs' if required -->

<script>

sap.ui.localResources("zui5\_desktop");

**var** view = sap.ui.view({id:"idmotivation1", viewName:"zui5\_desktop.motivation", type:sap.ui.core.mvc.ViewType.JS});

view.placeAt("content");

</script>

</head>

<body class=*"sapUiBody"* role=*"application"*>

<div id=*"content"*></div>

</body>

</html>

##### motivation.view.js

sap.ui.jsview("zui5\_desktop.motivation", {

/\*\* Specifies the Controller belonging to this View.

* In the case that it is not implemented, or that "null" is returned, this View does not have a Controller.
* **@memberOf** zui5\_desktop.motivation

\*/

getControllerName : **function**() {

**return** "zui5\_desktop.motivation";

},

/\*\* Is initially called once after the Controller has been instantiated. It is the place where the UI is constructed.

* Since the Controller is given to this method, its event handlers can be attached right away.
* **@memberOf** zui5\_desktop.motivation

\*/

createContent : **function**(oController) {

**var** oSlider1 = **new** sap.ui.commons.Slider(

{

id : "Motivationsregler", width : "300px",

//change : onSliderchange

}

);

oSlider1.attachChange(oController.onSliderchange);

//return oSlider1;

//bind Slider to Model

oSlider1.bindProperty("value", "/motivation"); //property Binding

**var** oLabel = **new** sap.ui.commons.Label("Label1"); oLabel.bindProperty("text", "/name");

**var** oTitle = **new** sap.ui.commons.Title("Title1"); oTitle.setText("Motivation von");

**var** oPanel = **new** sap.ui.commons.Panel("Panel1"); oPanel.setTitle(oTitle); oPanel.addContent(oLabel); oPanel.addContent(oSlider1);

**return** oPanel;

}

});

Lösung 4: Composite Binding

##### Main.view.xml

<mvc:View controllerName="schulung.MVCUebung.controller.Main" xmlns:mvc="sap.ui.core.mvc" displayBlock="true" xmlns="sap.m">

    <Shell id="shell">

        <App id="app">

            <pages>

                <Page id="page" title="{i18n>title}">

                    <content>

                        <List items="{/products}">

                            <items>

                                <StandardListItem title="{name}" description="{price}"/>

                            </items>

                        </List>

                    </content>

                    <footer>

                        <OverflowToolbar>

                            <content>

                                <Button text="Footertext"/>

                            </content>

                        </OverflowToolbar>

                    </footer>

                </Page>

            </pages>

        </App>

    </Shell>

</mvc:View>

##### Main.controller.js

sap.ui.define([

    "sap/ui/core/mvc/Controller",

    "sap/ui/model/json/JSONModel",

    "sap/m/MessageBox"

], function (Controller, JSONModel, MessageBox) {

    "use strict";

    return Controller.extend("schulung.bindingUebung.controller.Main", {

        onInit: function () {

            var oModel = new JSONModel({

                products: [

                    { name: "Laptop", price: "1200 EUR" },

                    { name: "Smartphone", price: "800 EUR" },

                    { name: "Tablet", price: "500 EUR" }

                ]

            });

            this.getView().setModel(oModel);

        },

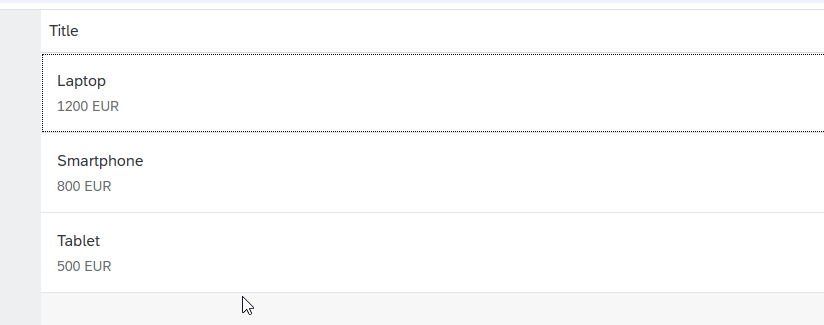
        formatFullName: function (sFirstName, sLastName) {

            return sFirstName + " " + sLastName;

        }

    });

});



Lösung 5: Composite Binding

##### Main.view.xml

<mvc:View controllerName="schulung.bindinguebung.controller.Main"

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

displayBlock="true"

xmlns="sap.m">

    <Shell id="shell">

        <App id="app">

            <pages>

                <Page id="page" title="{i18n>title}">

                    <content>

                        <Text text="{parts: [{path: '/user/firstName'}, {path: '/user/lastName'}], formatter: '.formatFullName'}" />

                    </content>

                    <footer>

                        <OverflowToolbar>

                            <content>

                                <Button text="Footertext"/>

                            </content>

                        </OverflowToolbar>

                    </footer>

                </Page>

            </pages>

        </App>

    </Shell>

</mvc:View>

##### Main.controller.js

sap.ui.define([

    "sap/ui/core/mvc/Controller",

    "sap/ui/model/json/JSONModel",

    "sap/m/MessageBox"

], function (Controller, JSONModel, MessageBox) {

    "use strict";

    return Controller.extend("schulung.bindingUebung.controller.Main", {

        onInit: function () {

            var oModel1 = new sap.ui.model.json.JSONModel({

                user: {

                    firstName: "John",

                    lastName: "Doe"

                }

            });

            this.getView().setModel(oModel1);

        },

        formatFullName: function (sFirstName, sLastName) {

            return sFirstName + " " + sLastName;

        }

    });

});

### Aufgabe 4 – SAPUI5 sap.ui.Table Control

##### table.html

<!DOCTYPE HTML>

<html>

<head>

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

<meta http-equiv=*'Content-Type'* content=*'text/html;charset=UTF-8'*/>

<script src=*"resources/sap-ui-core.js"*

id=*"sap-ui-bootstrap"*

data-sap-ui-libs=*"sap.ui.commons,* *sap.ui.table"*

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

</script>

<!-- add sap.ui.table,sap.ui.ux3 and/or other libraries to 'data-sap-ui- libs' if required -->

<script>

sap.ui.localResources("zui5\_desktop");

**var** view = sap.ui.view({id:"idmotivation1", viewName:"zui5\_desktop.table", type:sap.ui.core.mvc.ViewType.JS});

view.placeAt("content");

</script>

</head>

<body class=*"sapUiBody"* role=*"application"*>

<div id=*"content"*></div>

</body>

</html>

##### table.view.js

sap.ui.jsview("zui5\_desktop.table", {

/\*\* Specifies the Controller belonging to this View.

* In the case that it is not implemented, or that "null" is returned, this View does not have a Controller.
* **@memberOf** zui5\_desktop.table

\*/

getControllerName : **function**() {

**return** "zui5\_desktop.table";

},

/\*\* Is initially called once after the Controller has been instantiated. It is the place where the UI is constructed.

* Since the Controller is given to this method, its event handlers can be attached right away.
* **@memberOf** zui5\_desktop.table

\*/

createContent : **function**(oController) {

**var** oTable = **new** sap.ui.table.Table("Table1"); oTable.addColumn(**new** sap.ui.table.Column({

label : **new** sap.ui.commons.Label({text: "Name"}), width : "200px",

template : **new** sap.ui.commons.TextView().bindProperty("text", "Custname"), sortProperty : "Custname",

filterProperty : "Custname"

}));

oTable.addColumn(**new** sap.ui.table.Column({

label : **new** sap.ui.commons.Label({text: "City"}), width : "200px",

template : **new** sap.ui.commons.TextView().bindProperty("text", "City")

}

});

}));

oTable.bindRows("/BapiscudatSet"); //für xml

//oTable.bindRows("/CustomerSet");

**return** oTable;

##### table.controller.js

sap.ui.controller("zui5\_desktop.table", {

/\*\*

* Called when a controller is instantiated and its View controls (if available) are already created.
* Can be used to modify the View before it is displayed, to bind event handlers and do other one-time initialization.
* **@memberOf** zui5\_desktop.table

\*/

onInit: **function**() {

**var** oModel = **new** sap.ui.model.xml.XMLModel("bapicusdat.xml"); **var** oTable = sap.ui.getCore().byId("Table1"); oTable.setModel(oModel);

},

});

### Aufgabe 5 – Fiori-like App (Splitapp)

##### app1.html

<!DOCTYPE HTML>

<html>

<head>

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

<meta http-equiv=*'Content-Type'* content=*'text/html;charset=UTF-8'*/>

<script src=*"resources/sap-ui-core.js"*

id=*"sap-ui-bootstrap"* data-sap-ui-libs=*"sap.m"*

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

</script>

<!-- only load the mobile lib "sap.m" and the "sap\_bluecrystal" theme -->

<script>

sap.ui.localResources("zui5\_mobile");

**var** app = **new** sap.m.App({initialPage:"idapp11"});

**var** page = sap.ui.view({id:"idapp11", viewName:"zui5\_mobile.app1", type:sap.ui.core.mvc.ViewType.JS});

app.addPage(page); app.placeAt("content");

</script>

</head>

<body class=*"sapUiBody"* role=*"application"*>

<div id=*"content"*></div>

</body>

</html>

##### app1.controller.js

sap.ui.controller("zui5\_mobile.app1", { toCustomerDetail: **function**(){

sap.ui.getCore().byId("Detaillist").setBindingContext(**this**.getBindingContext()); sap.ui.getCore().byId("app").toDetail("DetailPage");

},

/\*\*

* Called when a controller is instantiated and its View controls (if available) are already created.
* Can be used to modify the View before it is displayed, to bind event handlers and do other one-time initialization.
* **@memberOf** zui5\_mobile.app1

\*/

onInit: **function**() {

**var** oModel = **new** sap.ui.model.xml.XMLModel("bapicusdat.xml"); sap.ui.getCore().setModel(oModel);

},

/\*\*

* Similar to onAfterRendering, but this hook is invoked before the controller's View is re-rendered
* (NOT before the first rendering! onInit() is used for that one!).
* **@memberOf** zui5\_mobile.app1

\*/

// onBeforeRendering: function() {

//

// },

/\*\*

* Called when the View has been rendered (so its HTML is part of the document). Post- rendering manipulations of the HTML could be done here.
* This hook is the same one that SAPUI5 controls get after being rendered.
* **@memberOf** zui5\_mobile.app1

\*/

// onAfterRendering: function() {

//

// },

/\*\*

* Called when the Controller is destroyed. Use this one to free resources and finalize activities.
* **@memberOf** zui5\_mobile.app1

\*/

// onExit: function() {

//

// }

});

##### app1.view.js

sap.ui.jsview("zui5\_mobile.app1", {

/\*\* Specifies the Controller belonging to this View.

* + In the case that it is not implemented, or that "null" is returned, this View does not have a Controller.
  + **@memberOf** zui5\_mobile.app1

\*/

getControllerName : **function**() {

**return** "zui5\_mobile.app1";

},

/\*\* Is initially called once after the Controller has been instantiated. It is the place where the UI is constructed.

* + Since the Controller is given to this method, its event handlers can be attached right away.
  + **@memberOf** zui5\_mobile.app1

\*/

createContent : **function**(oController) {

**var** oApp = **new** sap.m.SplitApp("app");

**var** oList = **new** sap.m.List("listCustomernames"); oList.setHeaderText("customers");

**var** oItem = **new** sap.m.StandardListItem("item", { title : "{Custname}",

id : "{Customerid}",

type : sap.m.ListType.Active, icon : "sap-icon://documents"

});

oItem.attachPress(oController.toCustomerDetail);

//oList.bindItems("/CustomerSet", oItem); //uncomment when using SAP database oList.bindItems("/BapiscudatSet", oItem);

**var** oMasterPage = **new** sap.m.Page({ title: "Title",

content: [

oList

]

});

**var** oDetailPage = **new** sap.m.Page ('DetailPage',{ title: "Detail",

showNavButton: **true**, navButtonPress: **function**()

{sap.ui.getCore().byId("app").backMaster();}

});

**var** detailForm = **new** sap.m.List('Detaillist', { items : [

**new** sap.m.InputListItem({

}),

label : "Kundennummer",

content : **new** sap.m.Text({text: "{Customerid}"})

**new** sap.m.InputListItem({ label : "Anrede",

content : **new** sap.m.Text({text: "{Form}"})

}),

**new** sap.m.InputListItem({ label : "Kundennamen",

content : **new** sap.m.Text({text: "{Custname}"})

}),

**new** sap.m.InputListItem({ label : "Strasse",

content : **new** sap.m.Text({text: "{Street}"})

}),

**new** sap.m.InputListItem({ label : "Postbox",

content : **new** sap.m.Text({text: "{Pobox}"})

}),

**new** sap.m.InputListItem({ label : "PLZ",

content : **new** sap.m.Text({text: "{Postcode}"})

}),

**new** sap.m.InputListItem({ label : "Stadt",

content : **new** sap.m.Text({text: "{City}"})

}),

**new** sap.m.InputListItem({ label : "Country",

content : **new** sap.m.Text({text: "{Countr\_iso}"})

}),

**new** sap.m.InputListItem({ label : "Telefon",

content : **new** sap.m.Text({text: "{Phone}"})

}),

**new** sap.m.InputListItem({ label : "Email",

content : **new** sap.m.Text({text: "{Email}"})

})

]

});

oDetailPage.addContent(detailForm);

oApp.addMasterPage(oMasterPage); oApp.addDetailPage(oDetailPage);

**return** oApp;

}

});

### Aufgabe 6 – Fiori-like App (Tiles)

##### app2.html

<!DOCTYPE HTML>

<html>

<head>

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

<meta http-equiv=*'Content-Type'* content=*'text/html;charset=UTF-8'*/>

<script src=*"resources/sap-ui-core.js"*

id=*"sap-ui-bootstrap"* data-sap-ui-libs=*"sap.m"*

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

</script>

<!-- only load the mobile lib "sap.m" and the "sap\_bluecrystal" theme -->

<script>

sap.ui.localResources("zui5\_mobile");

**var** app = **new** sap.m.App({initialPage:"idapp21"});

**var** page = sap.ui.view({id:"idapp21", viewName:"zui5\_mobile.app2", type:sap.ui.core.mvc.ViewType.JS});

app.addPage(page); app.placeAt("content");

</script>

</head>

<body class=*"sapUiBody"* role=*"application"*>

<div id=*"content"*></div>

</body>

</html>

##### app2.controller.js

sap.ui.controller("zui5\_mobile.app2", { toCustomerDetail: **function**(){

sap.ui.getCore().byId("Detailpage").setBindingContext(**this**.getBindingContext()); sap.ui.getCore().byId("App2").to("Detailpage"); //put Detailpage on top

of page stack

},

onBackPress: **function**(){

sap.ui.getCore().byId("App2").back();// when tapped, the back button should navigate in page stack

},

/\*\*

* Called when a controller is instantiated and its View controls (if available) are already created.
* Can be used to modify the View before it is displayed, to bind event handlers and do other one-time initialization.
* **@memberOf** zui5\_mobile.app2

\*/

onInit: **function**() {

**var** oModel = **new** sap.ui.model.xml.XMLModel("bapicusdat.xml"); sap.ui.getCore().setModel(oModel); sap.ui.getCore().byId("CustomerTileContainer").setModel(oModel);

},

/\*\*

* Similar to onAfterRendering, but this hook is invoked before the controller's View is re-rendered
* (NOT before the first rendering! onInit() is used for that one!).
* **@memberOf** zui5\_mobile.app2

\*/

// onBeforeRendering: function() {

//

// },

/\*\*

* Called when the View has been rendered (so its HTML is part of the document). Post- rendering manipulations of the HTML could be done here.
* This hook is the same one that SAPUI5 controls get after being rendered.
* **@memberOf** zui5\_mobile.app2

\*/

// onAfterRendering: function() {

//

// },

/\*\*

* Called when the Controller is destroyed. Use this one to free resources and finalize activities.
* **@memberOf** zui5\_mobile.app2

\*/

// onExit: function() {

//

// }

});

##### app2.view.js

sap.ui.jsview("zui5\_mobile.app2", {

/\*\* Specifies the Controller belonging to this View.

* + In the case that it is not implemented, or that "null" is returned, this View does not have a Controller.
  + **@memberOf** zui5\_mobile.app2

\*/

getControllerName : **function**() {

**return** "zui5\_mobile.app2";

},

/\*\* Is initially called once after the Controller has been instantiated. It is the place where the UI is constructed.

* + Since the Controller is given to this method, its event handlers can be attached right away.
  + **@memberOf** zui5\_mobile.app2

\*/

createContent : **function**(oController) {

//Example 2 is to create a 'normal' App, not SplitApp

**var** oApp = **new** sap.m.App('App2', {

});

//page with tiles

**var** oTilePage = **new** sap.m.Page('TilePage', {

title : "Flugkunden", enableScrolling : **false**

});

**var** oTile = **new** sap.m.StandardTile({

icon : "sap-icon://employee-pane", title: "{Custname}",

number : "{Customerid}", info : "{City}",

infoState : sap.ui.core.ValueState.Success, press : oController.toCustomerDetail

});

**var** oTileContainer = **new** sap.m.TileContainer("CustomerTileContainer", { editable: **false**,

allowAdd: **false**,

});

//oTileContainer.bindAggregation("tiles", "/CustomerSet", oTile);

//uncomment when using sap database

oTileContainer.bindAggregation("tiles", "/BapiscudatSet", oTile); oTilePage.addContent(oTileContainer);

oApp.addPage(oTilePage);

//page with details (almost same as in app1)

**var** oDetailPage = **new** sap.m.Page ({ id : "Detailpage",

title: "{Custname}" //here i changed the title to the customer name, see controller to adapt the binding context

});

**var** detailForm = **new** sap.m.List('Detaillist', { items : [

**new** sap.m.InputListItem({ label : "Kundennummer",

content : **new** sap.m.Text({text: "{Customerid}"})

}),

**new** sap.m.InputListItem({ label : "Anrede",

content : **new** sap.m.Text({text: "{Form}"})

Customername

}),

**new** sap.m.InputListItem({ label : "Kundennamen",

content : **new** sap.m.Text({text: "{Custname}"}) //vorher

}),

**new** sap.m.InputListItem({ label : "Strasse",

content : **new** sap.m.Text({text: "{Street}"})

}),

**new** sap.m.InputListItem({ label : "Postbox",

content : **new** sap.m.Text({text: "{Pobox}"})

}),

**new** sap.m.InputListItem({ label : "PLZ",

content : **new** sap.m.Text({text: "{Postcode}"})

}),

**new** sap.m.InputListItem({ label : "Stadt",

content : **new** sap.m.Text({text: "{City}"})

}),

**new** sap.m.InputListItem({ label : "Country",

content : **new** sap.m.Text({text: "{Countr\_iso}"})

}),

**new** sap.m.InputListItem({ label : "Telefon",

content : **new** sap.m.Text({text: "{Phone}"})

}),

**new** sap.m.InputListItem({ label : "Email",

content : **new** sap.m.Text({text: "{Email}"})

})

]

});

oDetailPage.addContent(detailForm);

oDetailPage.setShowNavButton(**true**); // detailpage needs this time a back button

oDetailPage.attachNavButtonPress(oController.onBackPress); oApp.addPage(oDetailPage);

**return** oApp;

}

});