

 MK

Martin Koch

HOU|5 - Col. 23.11

Herzlich Willkommen!

Das Team der CloudDNA begrüßt Sie recht herzlich zur SAP Schulung **HOU|5** - Hands-on Foundation zur Entwicklung von SAPUI5 Applikationen!

Auf dieser interaktiven Seite werden Sie gemeinsam mit uns SAPUI5 in ihren vollen Zügen kennenlernen! In fünf Tagen werden wir eine Kundenapplikation entwickeln und Schritt-für-Schritt im Rahmen einer Hands-on Session das Framework verstehen und Komponenten kennenlernen.

Wir wünschen Ihnen eine lehrreiche Woche!

Ihr Team der CloudDNA

EINLEITUNG

 Über uns

 Über den Kurs

ÜBUNG 1 - GRUNDAPPLIKATION ERSTELLEN

 **1 Einleitung** **1.1 Applikation anlegen und testen** **1.2 UI-Elemente** **1.3 Formular mit SimpleForm** **1.4 Formular erweitern**

 **1.5 Zusammenfassung**

ÜBUNG 2 - GIT

 **2 Einleitung** **2.1 Git Grundlagen (optional)** **2.2 Git anbinden** **2.3 Zusammenfassung**

ÜBUNG 3 - ÜBERSICHTSSEITE ERSTELLEN

 **3 Einleitung** **3.1 SemanticPage erstellen** **3.2 Tabelle erstellen** **3.3 Zusammenfassung**

ÜBUNG 4 - LOKALE DATENHALTUNG



4 Einleitung



4.1 JSONModel



4.2 DataBinding



4.3 Formatter im Controller



4.4 Zusammenfassung

ÜBUNG 5 - DETAILANSICHT ERSTELLEN



5 Einleitung



5.1 Navigation erstellen



5.2 ElementBinding ohne Parameter



5.3 ObjectPageLayout



5.4 Navigation erweitern



5.5 ElementBinding mit Parameter



5.6 Zusammenfassung

ÜBUNG 6 - FRAGMENTE



6 Einleitung



6.1 Display&Edit-Fragment anlegen



6.2 Fragmente im Controller definieren

 **6.3 Zusammenfassung**

ÜBUNG 7 SAP-BACKEND ANBINDUNG **7 Einleitung** **7.1 OData Grundlagen (optional)** **7.2 ODataModel einbinden** **7.3 Main-View anpassen** **7.4 Detail-View anpassen** **7.5 Zusammenfassung**

ÜBUNG 8 - DATEN ANLEGEN **8 Einleitung** **8.1 Main-view anpassen** **8.2 Customer-view mit createEntry anpassen** **8.3 Zusammenfassung**

ÜBUNG 9 - DATEN LÖSCHEN **9 Einleitung** **9.1 Daten löschen** **9.2 Zusammenfassung**

ÜBUNG 9.5 UI ANPASSUNG

-  9.5 UI Anpassung

ÜBUNG 10 - MODULARISIERUNG

-  10 Einleitung

 10.1 Modul erstellen und Formatter auslagern (optional)

 10.2 BaseController erstellen

 10.3 Zusammenfassung

ÜBUNG 11 - SMART-CONTROLS (OPTIONAL)

-  11 Einleitung

 11.1 Grundlagen Low-Code-Entwicklung

 11.2 SmartTable mit Remote-Annotationen

 11.3 Zusammenfassung

ÜBUNG 12 - DEPLOYMENT SAPUI5 ABAP REPOSITORY

-  12 Einleitung

 12.1 Deployment durchführen

 12.2 Applikation Standalone starten

 12.3 Zusammenfassung

ÜBUNG 13 - SAP FIORI LAUNCHPAD KONFIGURATION



13 Einleitung



13.1 Katalog anlegen



13.2 Kachel und Target-Mapping anlegen



13.3 Rolle anlegen und Katalog zuweisen



13.4 Kachel auf die Startseite pinnen



13.5 Gruppe anlegen und der Rolle zuweisen (optional)



13.7 Zusammenfassung

ÜBUNG 14 - MEDIA-ENTITY & UPLOADSET (OPTIONAL)



14 Einleitung



14.1 UploadSet implementieren



14.2 Zusammenfassung

APPENDIX



Weitere Tutorials

Über uns

 MK Martin Koch



CloudDNA

Die SAP Fiori, Integration und Cloud Experten aus Österreich

Wir navigieren Unternehmen jeder Größe erfolgreich durch und in die SAP Cloud. Unsere Stärken sind SAPUI5 / Fiori, Cloud Integration, API Management, Workflow, Mobile und Cloud Security. Wir haben die passenden zertifizierten Experten für jedes Projekt. Mit CloudDNA bekommt Qualität einen Namen.

Unsere Standbeine

SAP BERATUNG	SAP ENTWICKLUNG	TRAININGS & COACHING	NON-SAP
Das Team der CloudDNA berät Sie zukunftssicher im Bereich der neuen SAP Technologien und umfassend über Cloud Engineering.			

SAP BERATUNG	SAP ENTWICKLUNG	TRAININGS & COACHING	NON-SAP
Wir begleiten Sie bei SAP On-Premise und Cloud Projekten (SAPUI5 / Fiori, ABAP, OData, CPI, CAP, ...) von der Idee bis hin zur Umsetzung.			

SAP BERATUNG	SAP ENTWICKLUNG	TRAININGS & COACHING	NON-SAP
Wir sind externe Trainer bei SAP SE und haben Best-in-class Kurse entwickelt, die im offiziellen SAP Schulungskatalog angeboten werden.			

SAP BERATUNG	SAP ENTWICKLUNG	TRAININGS & COACHING	NON-SAP
Wir stellen auch bei Non-SAP Webentwicklungen und Entwicklung von Webservices (Angular, VueJS, Spring, ...) unser Wissen stets unter Beweis.			

Das Team

Die CloudDNA GmbH besteht aus einem Team an SAP Experten in modernen SAP Technologien. Ein perfekter Mix aus alten Hasen und den jungen Wilden, die die SAP Welt erobern wollen. Unser Ziel ist es, unsere Kunden absolut zufriedenstellend und dem aktuellsten Stand der Technik entsprechend zu beraten. Es nützt Ihnen nichts, wenn Sie einen Tesla oder Ferrari in Ihrer Garage stehen haben und Sie von einem VW-Käfer-Experten beraten werden.

Unser Team deckt die Bereiche SAP Business Technology Platform (ehemals SAP Cloud Platform), SAP Fiori, SAPUI5, SAP HANA, SAP CPI, SAP PI, SAP PO, SAP Mobile Services, SAP IAS und SAP Security ab. Wir beherrschen unser Handwerk bis ins kleinste Detail und sind in allen Themenbereichen SAP® zertifiziert. Von unseren Kunden werden wir oft als Cloud Natives bezeichnet, mit denen man einfach und unkompliziert zusammenarbeiten kann.

Sie möchten mehr über uns erfahren?

Auf unserer Website finden Sie alle Informationen!

ZUR WEBSITE

Über den Kurs

MK

Martin Koch

Die Herausforderung

Im Bereich moderner SAP Technologien ist es ein gewagtes Unterfangen, Trainings mit topaktuellen Inhalten zu finden. Findet man sie, fehlt häufig der Praxisbezug. Die Entwicklung und Erweiterung ansprechender Benutzeroberflächen, sicherer Cloud Systeme und hybrider Integration stehen bei vielen SAP Kunden besonders hoch in der Prioritätenliste. Da alle Themen sehr schnelllebig sind, ist es besonders wichtig, aus den aktuellsten Releases zu lernen. Mit den Best-in-class SAP Trainings können Sie immer auf dem neuesten Stand der Technik bleiben.

Historie

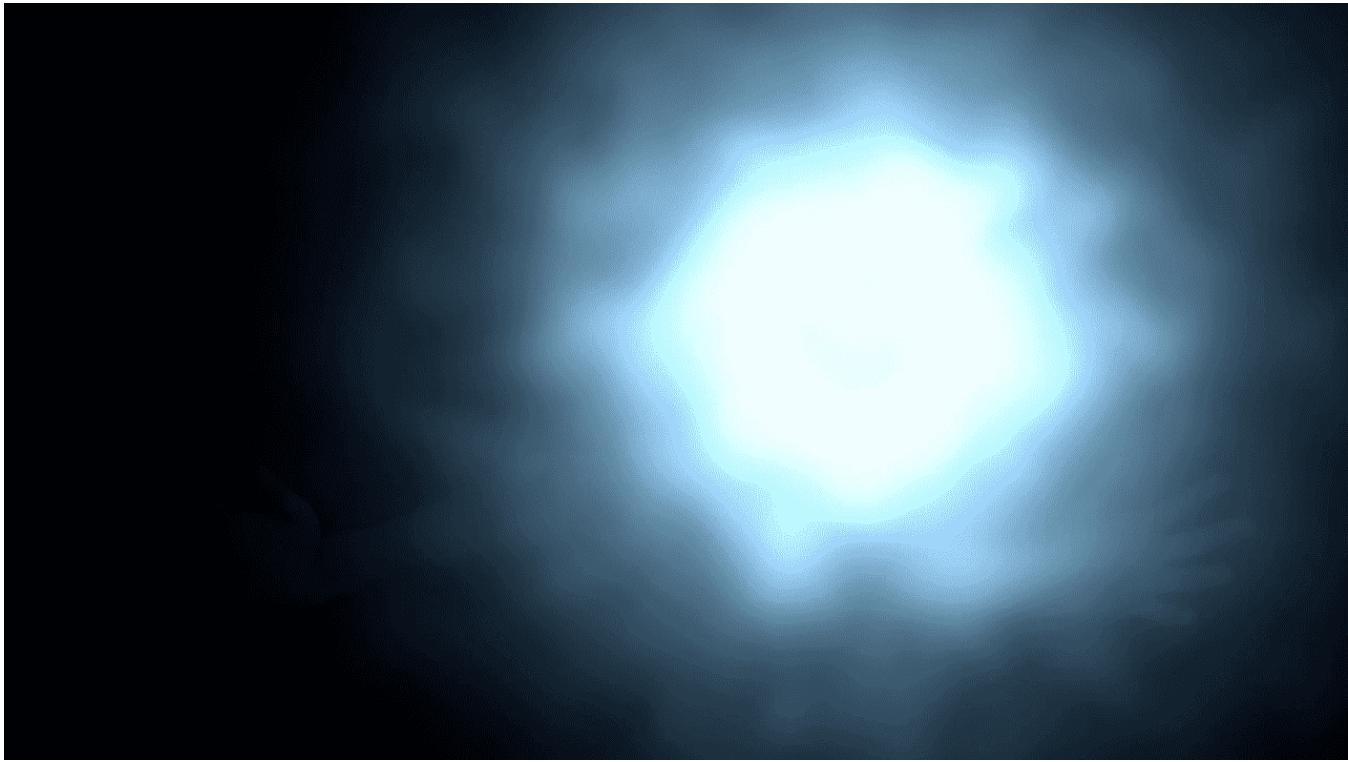
Das Team der ClouddNA hat die Anforderungen aus allen Projekten der letzten Jahren zusammengetragen, mit Kunden gesprochen und ihre Vorstellungen erhoben. Wir haben auf das Feedback in Trainings, Schulungen und Lehrgängen gehört. SAPUI5, SAP Fiori, SAP Cloud Integration und SAP Cloud Security waren dabei die wichtigsten Themen. Daraus haben wir praxisrelevante Trainings entwickelt. Unser Motto lautet „Von Experten für Experten“. Außerdem haben wir mit unseren Trainings auch die SAP überzeugt. Seitdem sind wir in den offiziellen SAP Schulungskatalog aufgenommen. Die Trainings werden organisatorisch und kommerziell von SAP abgewickelt. Unser Anspruch an uns selbst lautet mit „Top Experten und aktuellen Themen“ Kunden überzeugen. Wir stellen die Trainer und aktualisieren laufend unsere Trainings. Unsere KollegInnen gehören zu den am weltweit besten bewerteten Trainer. Vor, während und nach jedem Training werden die Unterlagen aktualisiert. Folgende Trainings sind daraus entstanden:

- **HOU15 – Hands-on SAPUI5 Entwicklung**
- **WDEI1 – SAP Cloud Integration, eine praxisorientierte Einführung**
- **WDECPS – SAP Cloud Security, eine ganzheitliche Betrachtung**

HOU15 - Ziele

- Verstehen der Prinzipien von SAPUI5, erklärt an einem durchgängigen Beispiel aus der Praxis
- Implementieren von SAPUI5 Applikationen
- Verstehen der Konzepte und Umsetzung von komplexen SAPUI5 Applikationen (90% Hands-on)
- Effizientes und zielgerichtetes Arbeiten mit der offiziellen SAPUI5 Dokumentation

Anbei finden Sie ein Video zu HOU15. Dieses Video finden Sie auch auf unserem [YouTube-Kanal](#).



1 Einleitung

 Martin Koch

Übung 1 – Grundapplikation erstellen

Erstellen einer einfachen SAPUI5 Applikation, um Daten in einer View darzustellen

Erstellen Sie eine View, in der alle Attribute eines Kunden statisch (ohne Datenmodell und Service-Anbindung) dargestellt werden. Die Attribute werden als sap.m.Label und sap.m.Text in einer sap.m.HBox dargestellt. Ersetzen Sie die HBox durch ein Formular, sprich durch das Control sap.ui.layout.form.SimpleForm. Die App soll ohne Routing auskommen und alle Daten in der sogenannten Root-View darstellen.

Verwendete Controls: [Label](#), [Text](#), [HBox](#), [SimpleForm](#), [Select](#)

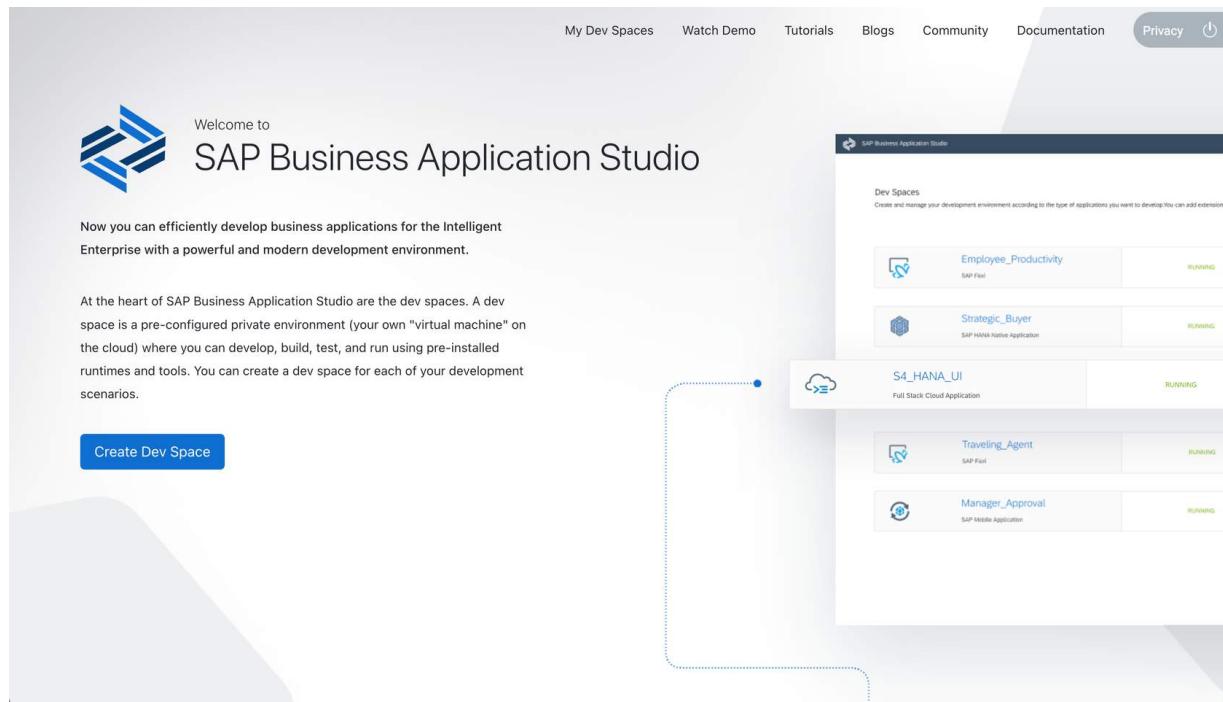
Verwendete Technologien: [i18n](#), [AggregationBinding](#), [ResponsiveGridLayout](#)

1.1 Applikation anlegen und testen

MK Martin Koch

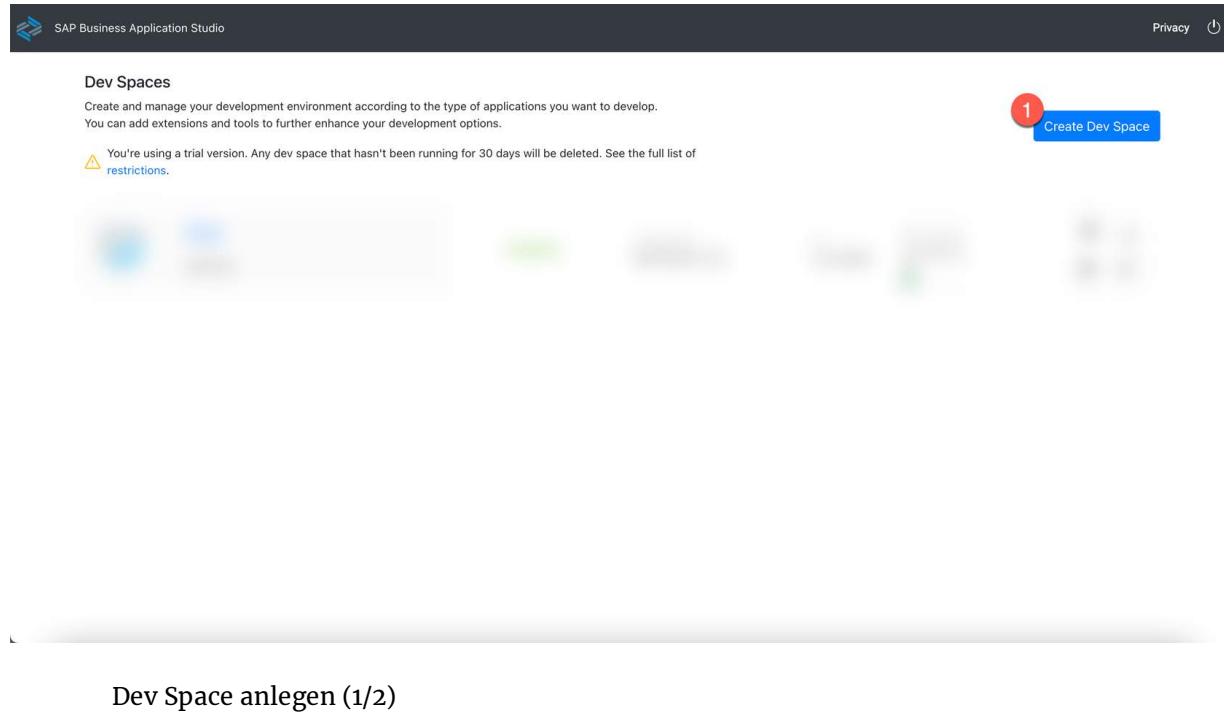
Entwicklungsumgebung öffnen

Öffnen Sie das SAP Business Application Studio und legen Sie in einem SAP Fiori Dev Space ein neues SAPUI5 -Freestyle Projekt an.



Wenn Sie die Entwicklungsumgebung das erste Mal starten, dann werden Sie mit einem Wizard begrüßt.

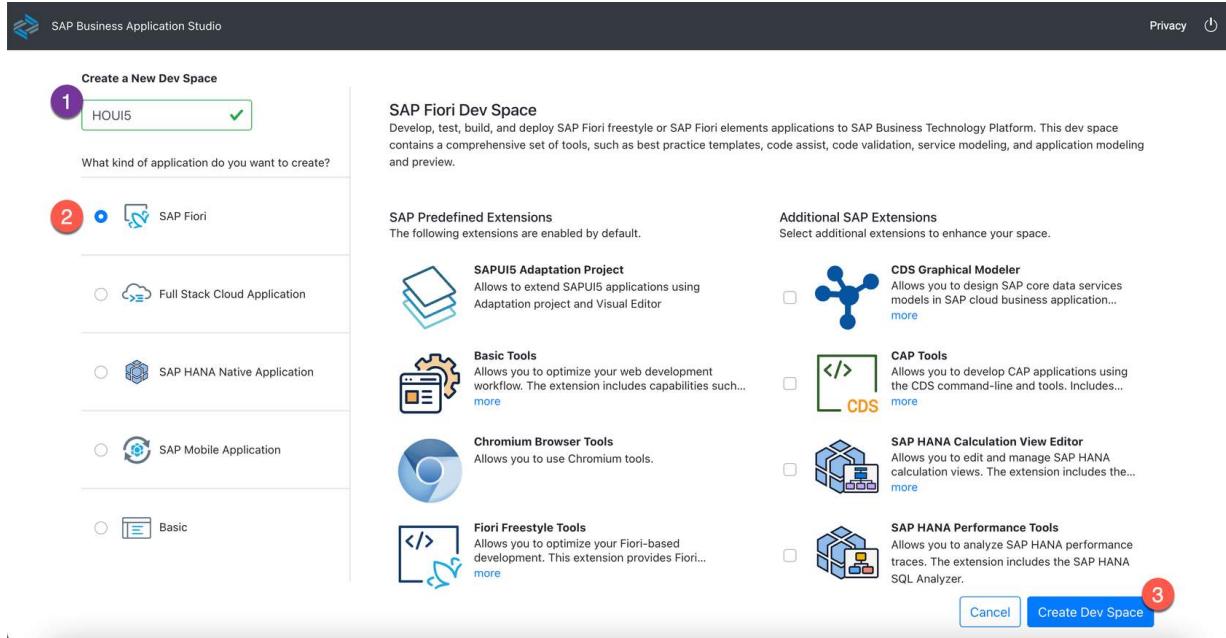
In jedem Fall müssen wir einen Dev Space anlegen und auf den Button **Create Dev Space** klicken.



Dev Space anlegen (1/2)

Bei jedem weiteren Start sehen Sie eine Auflistung aller existierenden Dev Spaces.

In diesem Fall müssen Sie rechts oben auf den Knopf **Create Dev Space** klicken

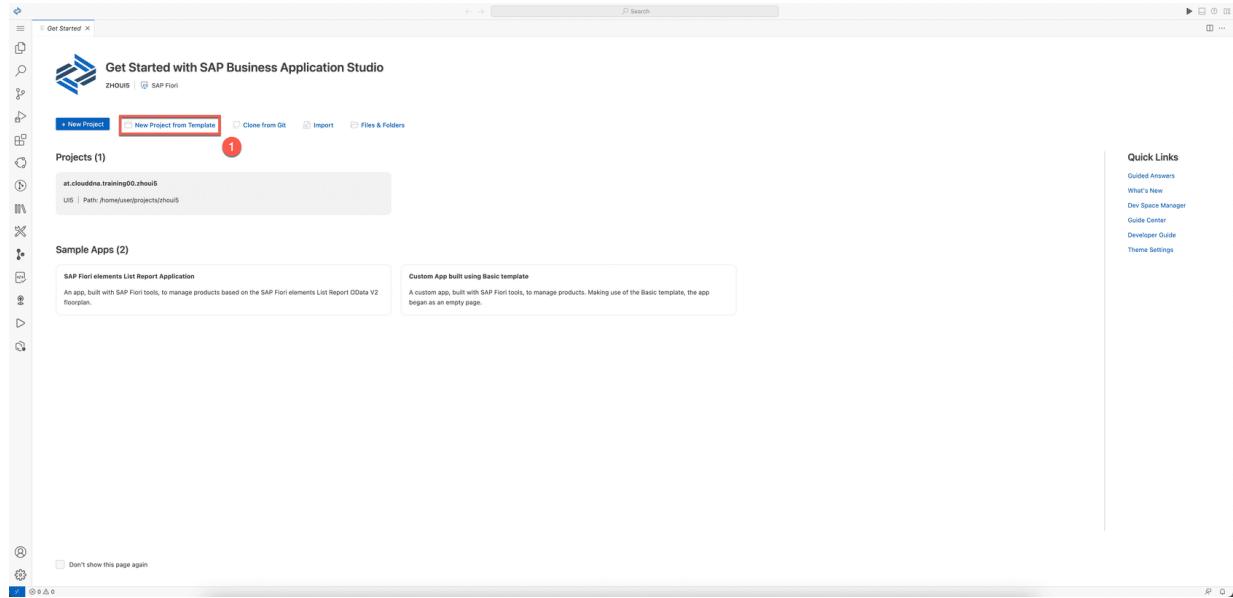


Dev Space anlegen (2/2)

1. Namen vergeben (Beispiel: **HOU15**)
2. **SAP Fiori** als Typ auswählen
3. Auf Knopf **Create Dev Space** klicken

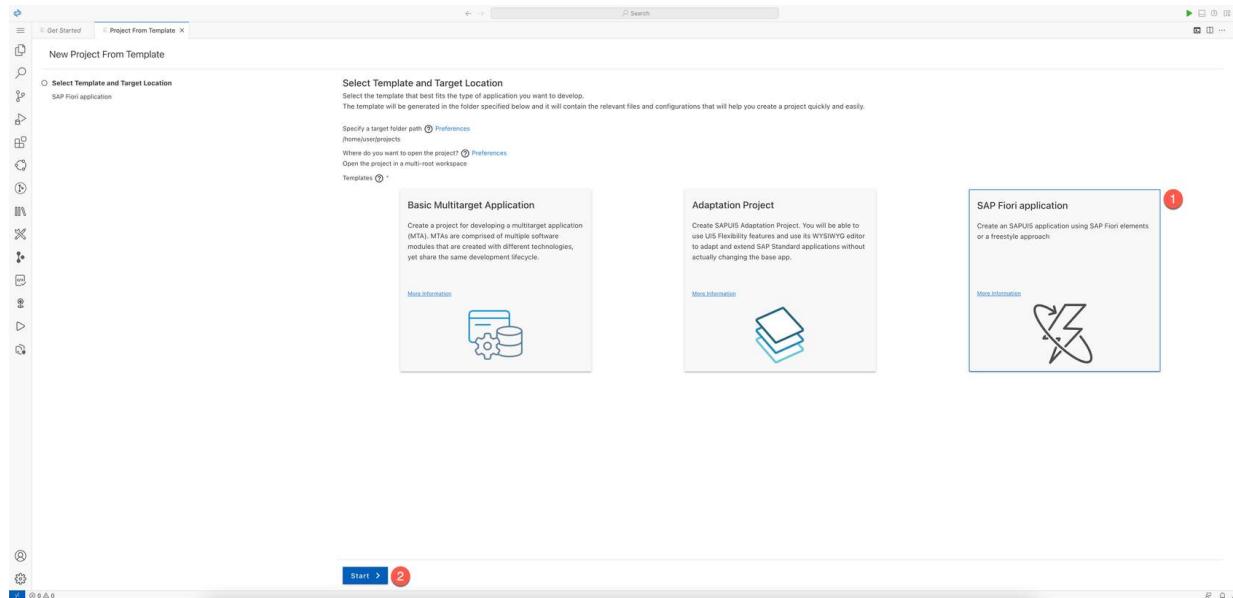
Applikation erstellen

Viele Wege führen nach Rom: Ihr Trainer wird Ihnen die verschiedenen Möglichkeiten für das Anlegen eines neuen Projekts zeigen. Für das Erste wählen wir die angebotene Option auf der Startseite.



Projekt anlegen (1/6)

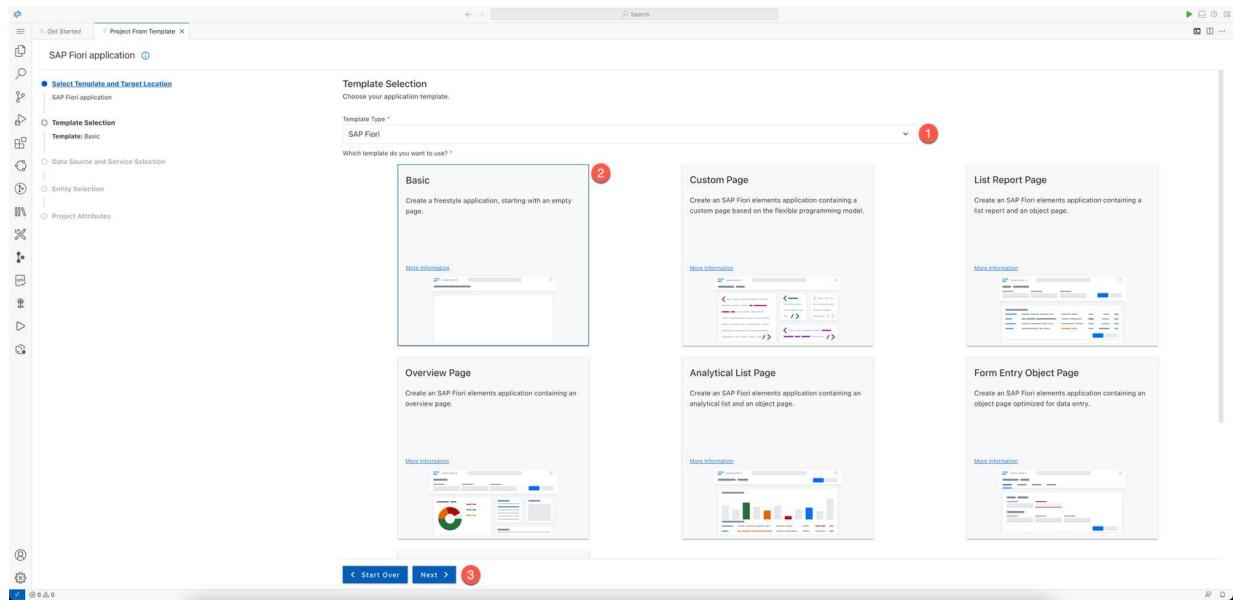
1. Auf New Project from template klicken



Projekt anlegen (2/6)

1. SAP Fiori application auswählen

2. Auf Start klicken

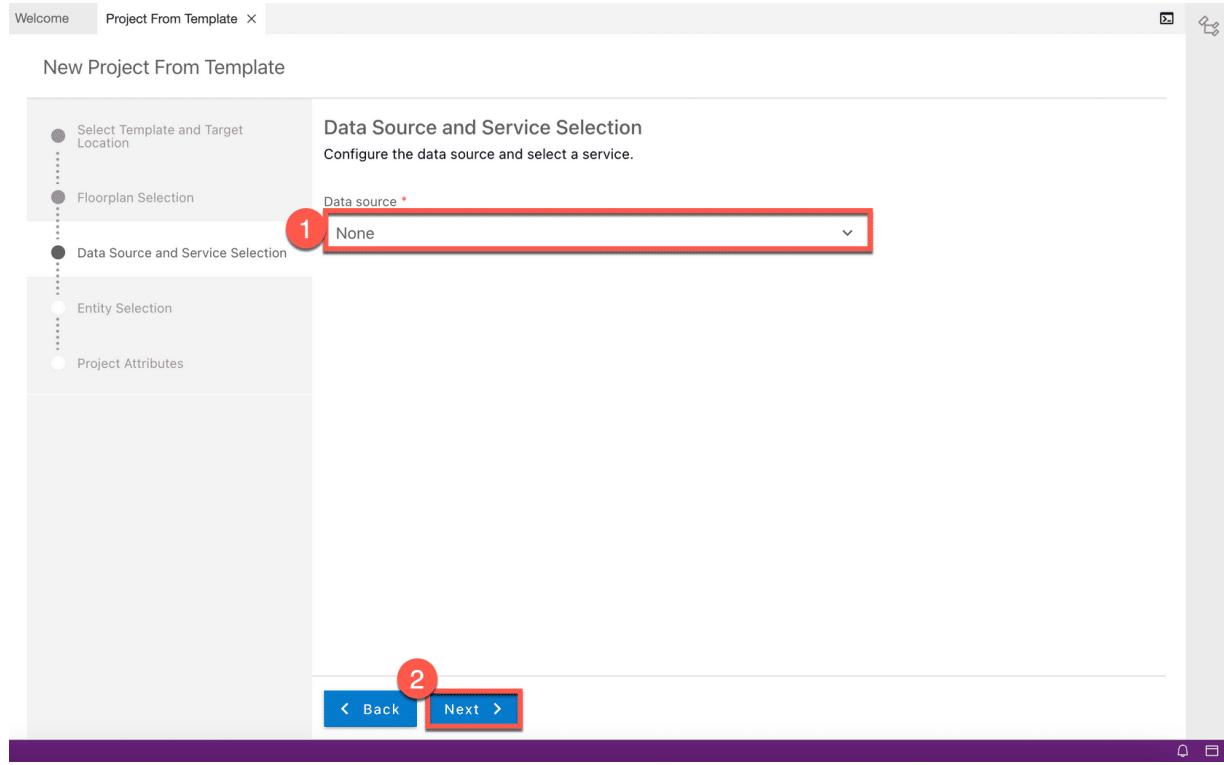


Projekt anlegen (3/6)

1. SAP Fiori auswählen

2. Leere SAPUI5 Application auswählen

3. Auf Next klicken



Projekt anlegen (4/6)

1. Data source **None** auswählen

2. Auf **Next** klicken

SAP Fiori application ⓘ

● [Select Template and Target Location](#)

SAP Fiori application

● [Template Selection](#)

Template: Basic

● [Data Source and Service Selection](#)

Data Source: None

○ [Entity Selection](#)

View Name: Customer

○ [Project Attributes](#)

Entity Selection

Configure the selected service.

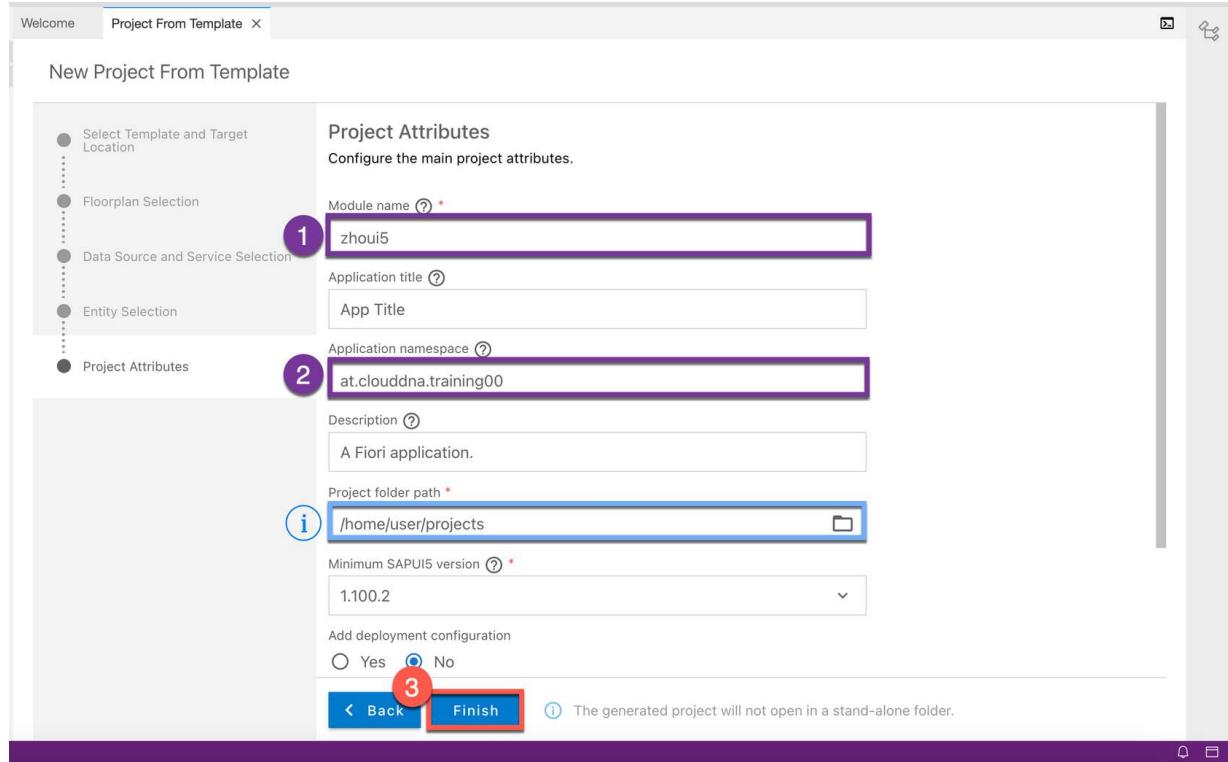
View name

Customer

Projekt anlegen (5/6)

1. Namen für die erste View vergeben: **Customer**

2. Auf **Next** klicken



Projekt anlegen (6/6)

1. Namen für das Projekt vergeben: **zhoui5**

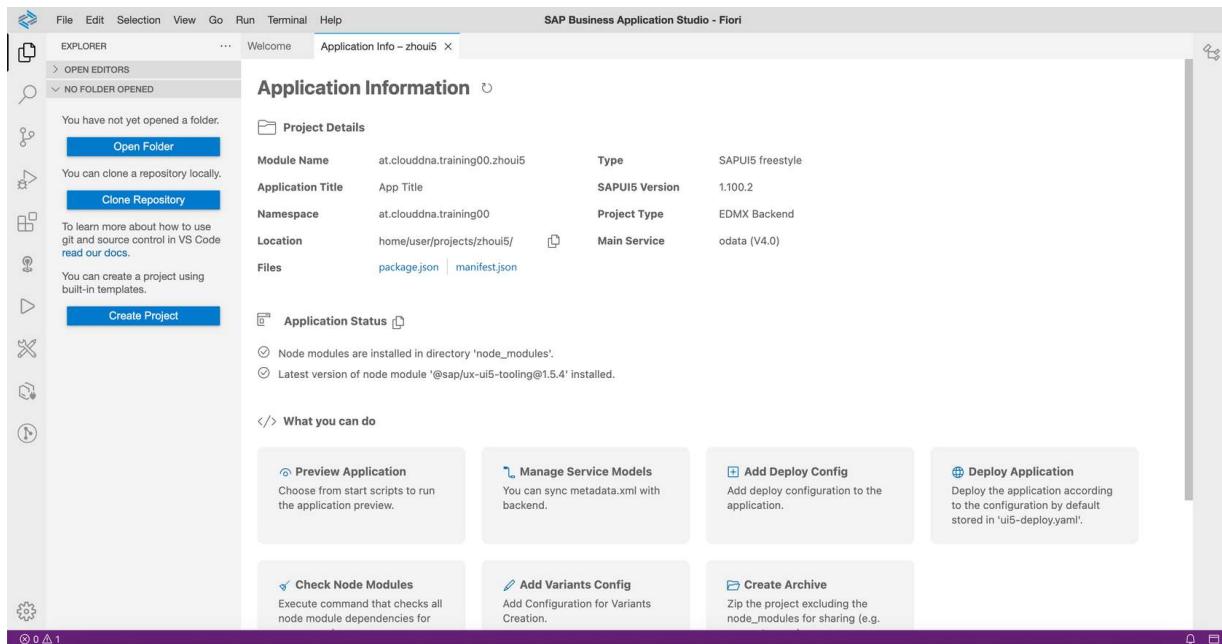
2. Namensraum vergeben: **at.clouddna.training## (Achtung! ## = Teilnehmernummer!)**

3. Auf **Finish** klicken

(i) Achten Sie darauf, dass Sie den richtigen Pfad angegeben haben! Standardmäßig ist der Ordner /home/user/projects für die Projekte vorgesehen.

Unter Umständen kann es passieren, dass dieses Feld mit einem Pfad vorbefüllt wird, welcher auf ein Projekt zeigt. In diesem Fall wird ein Projekt in das andere hineingeneriert!

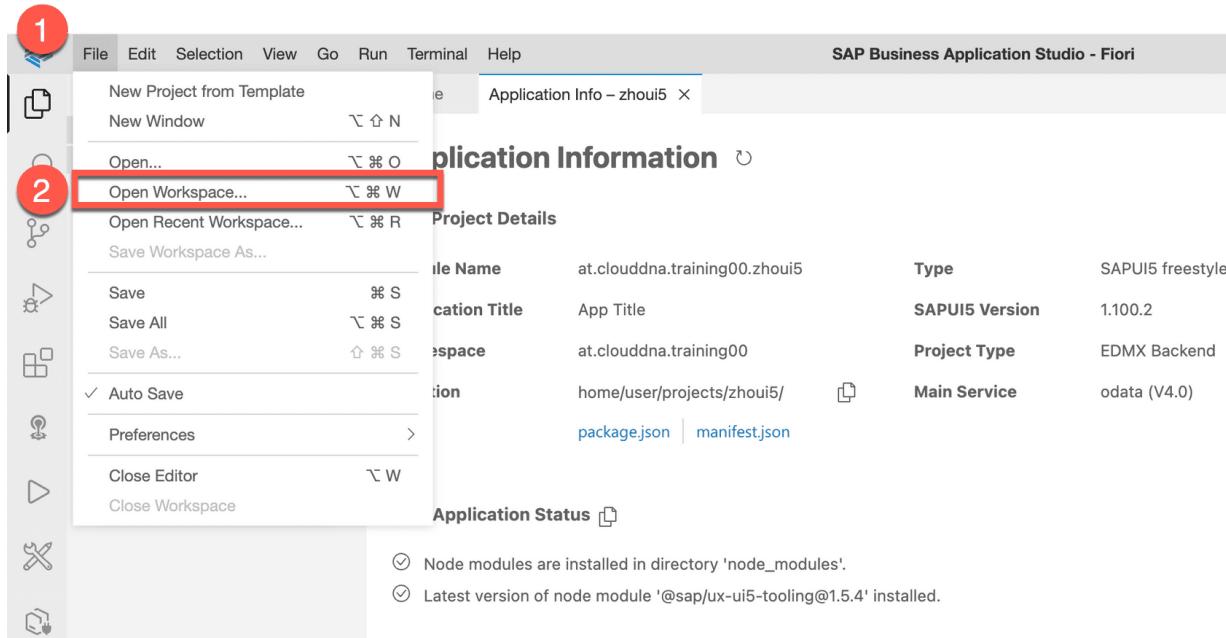
Nachdem ein Projekt erstellt wurde, werden die Informationen zur Applikation in einem eigenen Fenster geöffnet.



Applikationsinformationen

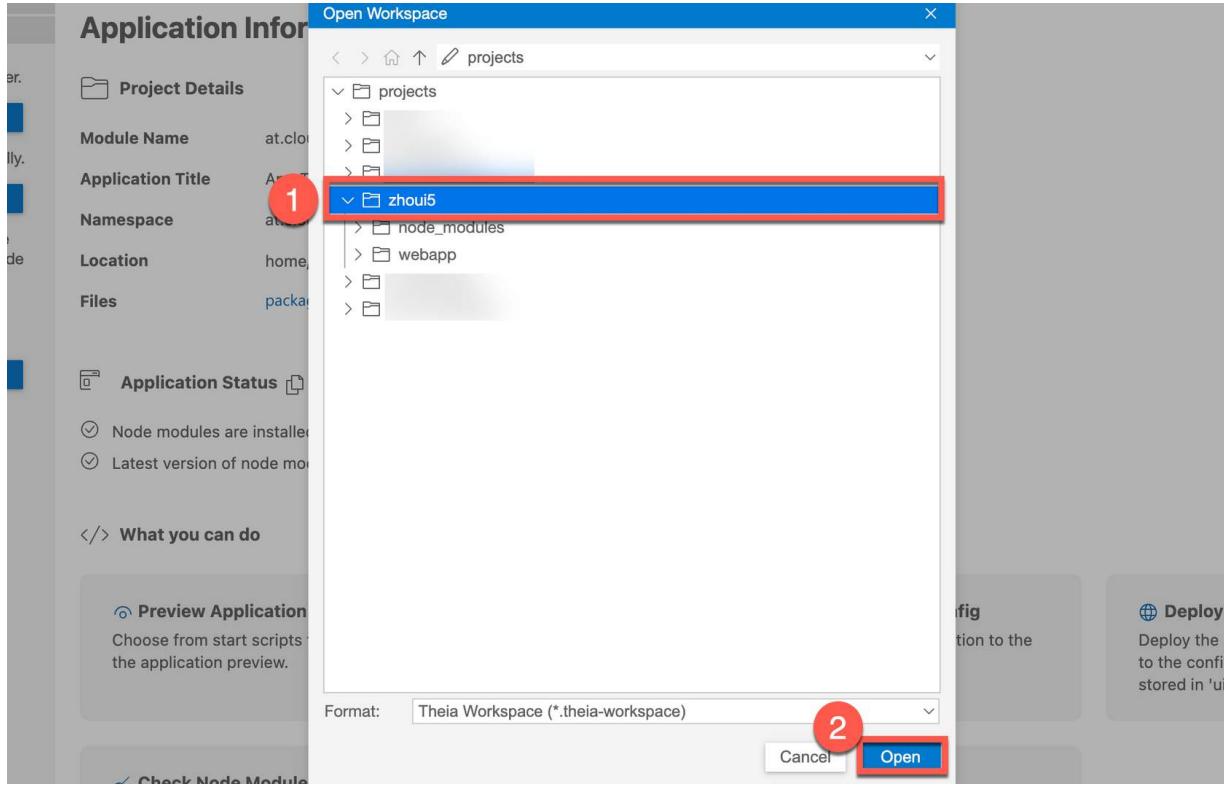
Applikation in einem Workspace öffnen

Ihre Applikation wird nicht automatisch in einem Workspace geöffnet. Öffnen Sie nun als nächstes Ihre Applikation selbst.



Workspace öffnen

1. In der Menüleiste auf **File** klicken
2. **Open Workspace...** auswählen



Ordner/Projekt auswählen

1. Projekt **zoui5** auswählen

2. Auf **Open** klicken

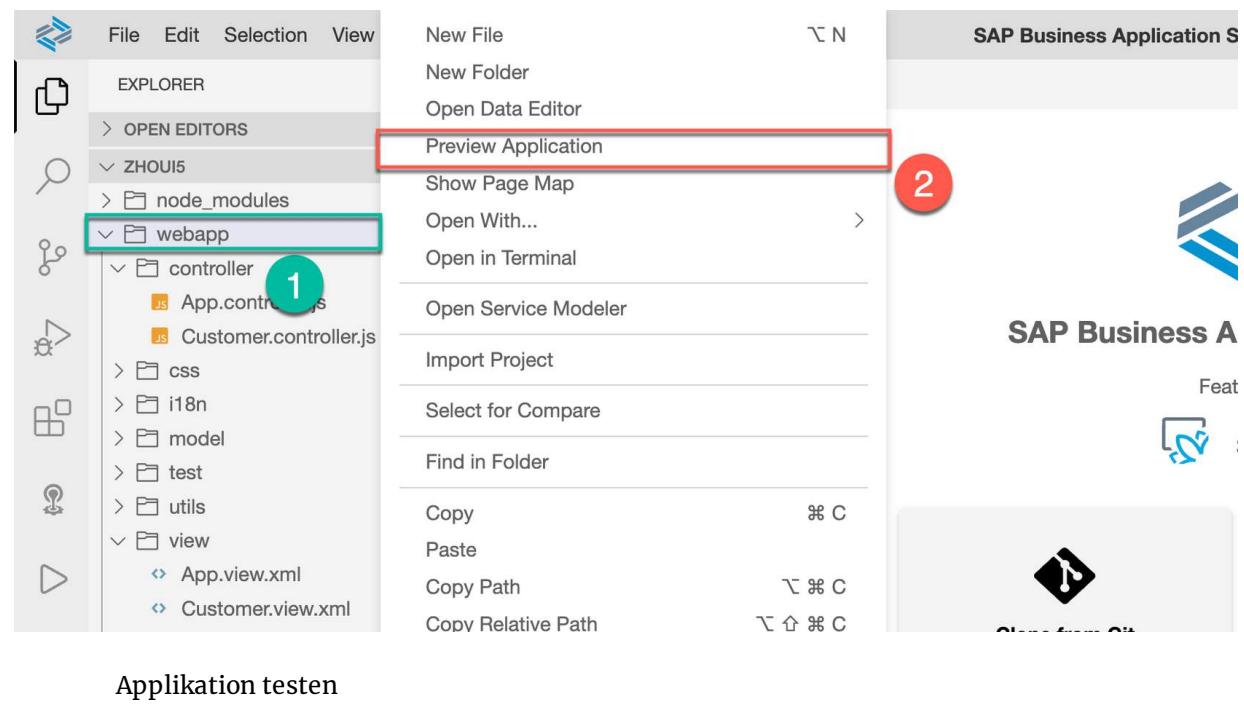
Ordnerstruktur und Dateien

Bevor Sie die Applikation testen, sehen Sie sich auch die erstellte Ordnerstruktur und Dateien an.

Die Ordner **controller** und **view** ausklappen. Hier wurden die ersten Views mit den dazugehörigen JS-Controllern angelegt.

Applikation testen

Der initiale Start der leeren Applikation wird durchgeführt.



1. Kontextmenü (Rechtsklick) auf irgendeinen Ordner
2. Preview Application auswählen



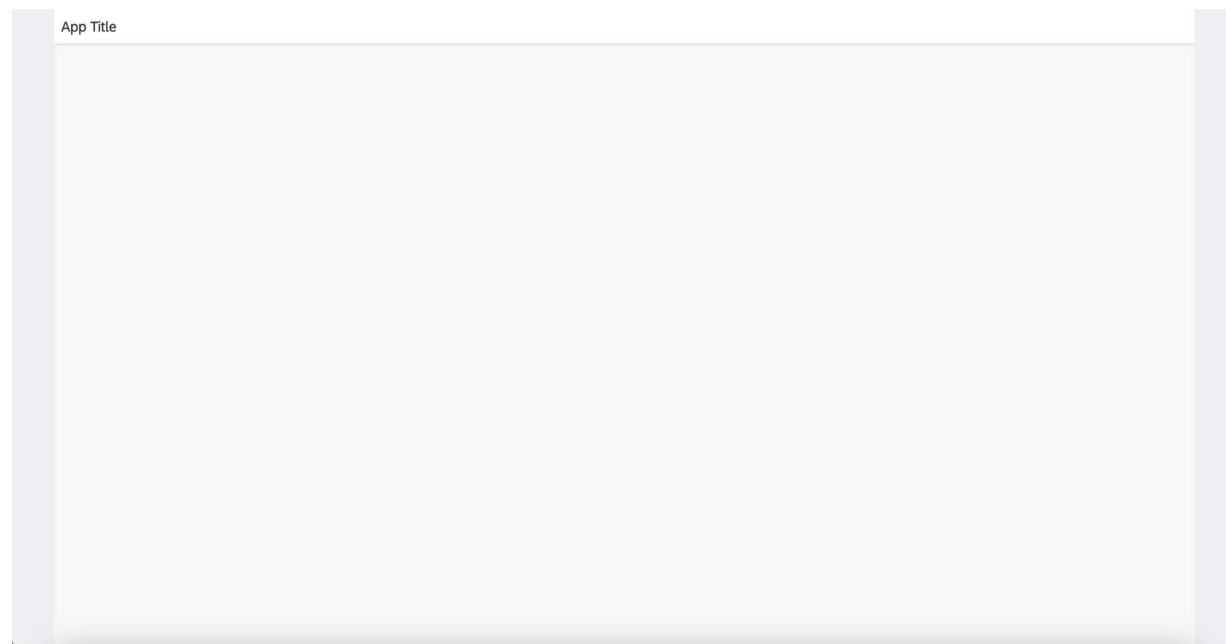
```
1
Select a npm script
Welcome ×
start fiori run --open "test/fipSandbox.html?sap-ui-xx-viewCache=false#atclouddnattraining00zhoui5-di
start-local fiori run --config ./ui5-local.yaml --open "test/fipSandbox.html?sap-ui-xx-viewCache=false#
start-noflp fiori run --open "index.html?sap-ui-xx-viewCache=false"
start-variants-management fiori run --open "preview.html?sap-ui-xx-viewCache=false&fiori-tools-rtar
unit-tests fiori run --open test/unit/unitTests.qunit.html
int-tests fiori run --open test/integration/opaTests.qunit.html
```

SAP Business Application Studio

Featuring:



App über die index.html starten

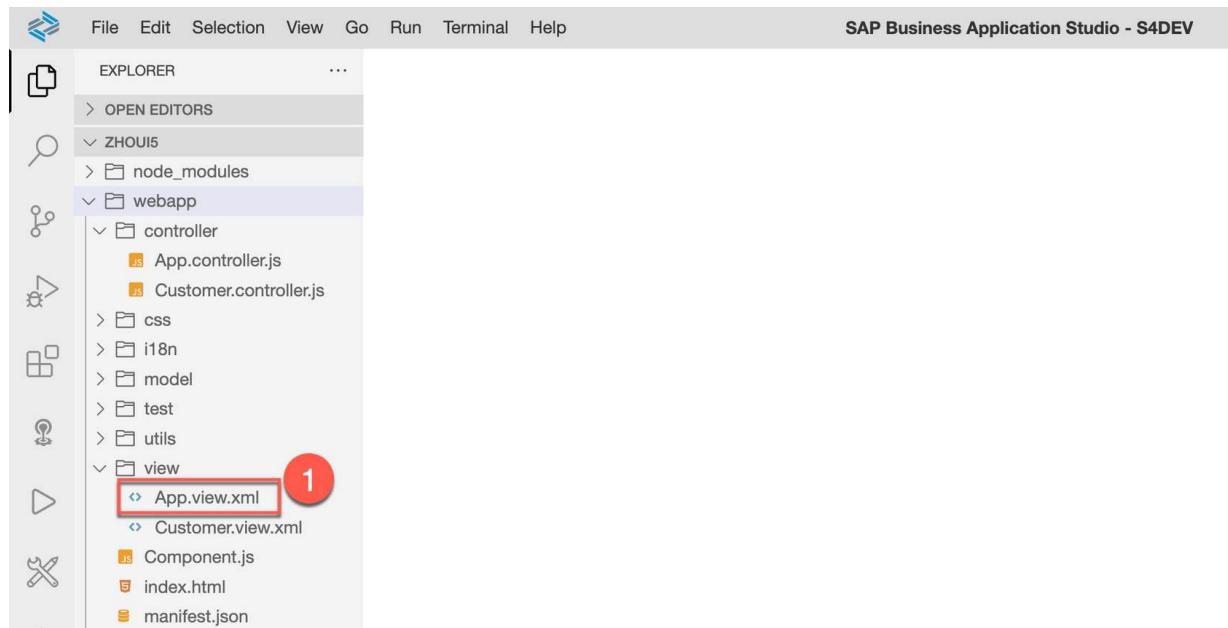


Applikation das erste Mal gestartet

1.2 UI-Elemente

MK Martin Koch

Die initiale View analysieren

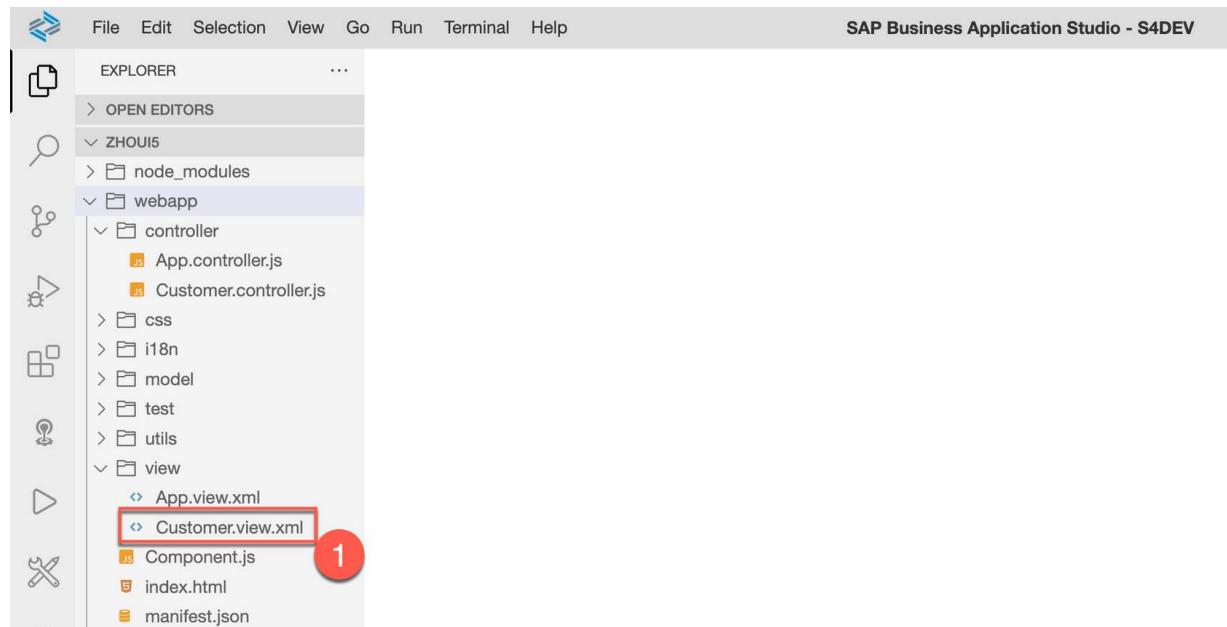


Initiale View im Ordner view öffnen (**App.view.xml**)

Diese View dient nur als Container und beinhaltet im Endeffekt das UI-Element **App**

Customer View analysieren

Diese View beinhaltet nur ein UI-Element Page. Diese Page wird dann vom Framework automatisch in den Container (welche sich in der App.view.xml befindet) eingefügt. Warum das so passiert und wie wir hier eingreifen können, werden wir erst später kennenlernen.



Customer View im Ordner view öffnen (**Customer.view.xml**)

In diesem Fall wird die Page zur **Kundenansicht** benötigt. Das Datenmodell von unseren Kunden sieht wie folgt aus:

- Kundennummer (Key)
- Vorname
- Nachname
- Titel
- Geschlecht
- Email

- Telefon
- Website

Erste UI-Elemente / -Controls verwenden

In Page-Aggregation <content> ein Control-Element vom Typ HBox hinzufügen. In diese HBox ein Label und einen Text für die Kundennummer einfügen.

sap.m.HBox

Bei der HBox werden die Controls horizontal (von links nach rechts) eingefügt. Eine VBox hat die gleiche Grundfunktionalität wie eine Hbox, nur werden die Controls vertical (untereinander) eingefügt.

sap.m.Label

Ein Label fungiert als Titel für ein Control bzw. für eine Gruppe von Controls. Sie können anzeigen, dass ein dazugehöriges Control required/auszufüllen ist, des Weiteren kann die Textgestaltung des Labels mit verschiedenen Properties reguliert werden.

Most used Properties:

- id – ID des Controls
- text – Angezeigter Labeltext

- `labelFor` – ID des zu zugehörigen Controls

sap.m.Text

Das Text-Control dient als Container für einen nicht editierbaren Text. Dieses Textfeld kann durch mehrere Properties formatiert und geregelt werden.

Most used Properties:

- `id` – ID des Controls
- `text` – Angezeigter Text

Welcome Customer.view.xml ×

webapp > view > Customer.view.xml

```
1  <mvc:View controllerName="at.clouddna.training00.zhoui5.controller.Customer"
2      xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3      xmlns="sap.m">
4      <Page id="page" title="{i18n>title}">
5          <content>
6              <HBox id="app_hbox_customerid">
7                  <items>
8                      <Label id="app_label_customerid" text="CustomerId"
9                          labelFor="app_text_customerid"/>
10                     <Text id="app_text_customerid" text="1000001"/>
11                 </items>
12             </HBox>
13         </content>
14     </Page>
15 </mvc:View>
```

1

1. HBox mit Label und Text hinzufügen

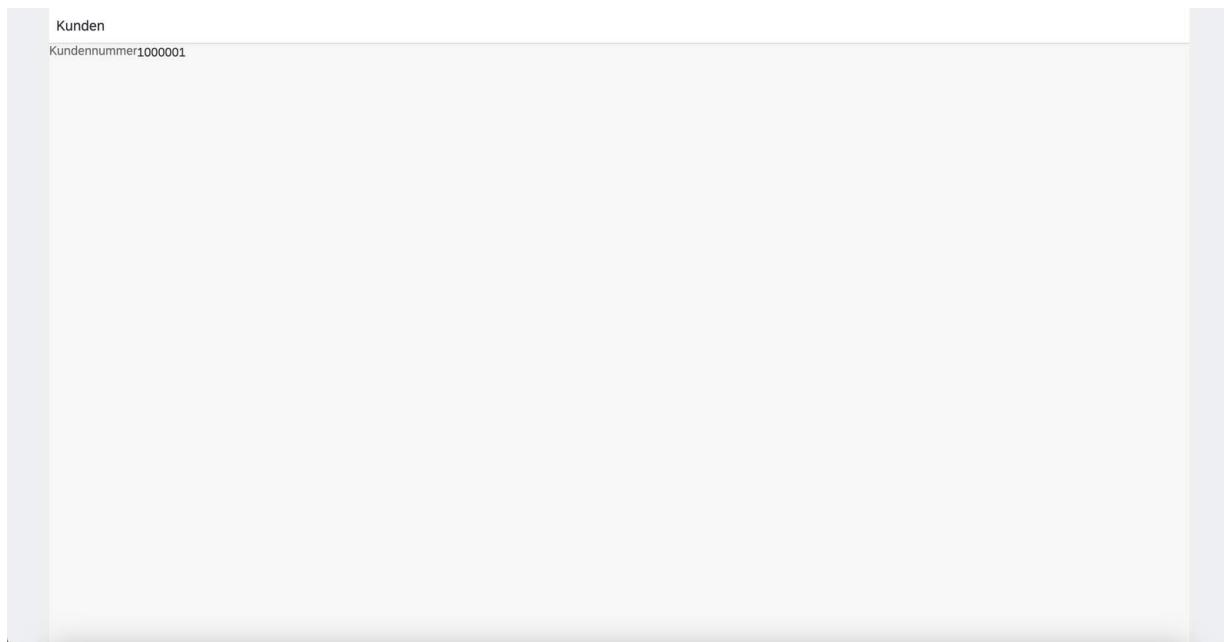
```
<HBox id="app_hbox_customerid">
    <items>
        <Label id="app_label_customerid" text="Kundennummer"
            labelFor="app_text_customerid"/>
        <Text id="app_text_customerid" text="1000001"/>
    </items>
</HBox>
```

i18n-Texte anpassen

The screenshot shows the SAP Business Application Studio interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar indicates "SAP Business Application Studio - F". The left sidebar is the Explorer, showing a tree structure of a project named "ZHOU15". A file named "i18n.properties" is selected and highlighted with a red box, circled with a red number 1. The right pane displays the contents of "i18n.properties":

```
1 # This is the resource bundle for at.clouddna.training00.zhou15
2
3 #Texts for manifest.json
4
5 #XTIT: Application name
6 appTitle=H0UI5 2
7
8 #YDES: Application description
9 appDescription=A Fiori application.
10 #XTIT: Main view title
11 title=Kunden 3
```

1. Datei **i18n.properties** im Verzeichnis **i18n** öffnen
2. Eintrag **appTitle** bearbeiten
3. Eintrag **title** bearbeiten



Vorschau mit HBox

1.3 Formular mit SimpleForm

MK

Martin Koch

Formular anstatt der horizontalen Box verwenden

Customer.view.xml öffnen. In der content-Aggregation statt der HBox die SimpleForm einfügen. ID und Titel der SimpleForm ändern. Einfügen von Label und Text aus der HBox in die content-Aggregation der SimpleForm.

Die [UI5-Dokumentation](#), auch bekannt als SAPUI5 API-Referenz, ist eine umfassende Ressource, die von SAP bereitgestellt wird, um Entwicklern detaillierte Informationen über die SAPUI5-Bibliothek zu bieten. In der Dokumentation finden Entwickler umfassende Informationen zu UI-Elementen, Steuerelementen, Modellen, Datenbindung, Ereignissen und anderen Aspekten der SAPUI5-Entwicklung. Die Dokumentation dient als zentrale Anlaufstelle für technische Details, Beispiele, Best Practices und Anleitungen, die für die Entwicklung von SAPUI5-Anwendungen relevant sind.

sap.ui.layout.form.SimpleForm

Die SimpleForm ist eine einfach Layoutkomponente, die es ermöglicht, Daten in einem Formular dazustellen. sap.ui.core.Title sorgen für einen Spalten- und sap.m.Label für einen Zeilenumbruch.

Most used Properties:

- id - ID des Controls
- columns – Wieviele Spalten sollen bei welcher Displaygröße angezeigt werden
- labelSpan – Wieviel Platz soll ein sap.m.Label einnehmen
- layout – Welches Layout bekommt die SimpleForm

GridLayout

Ein Layout, welches ein 12-Zeiliges Raster zur Verfügung stellt, in dem die Child-Elemente basierend auf ihrer Spaltenbreite dargestellt werden.

1. In der Dokumentation **Samples** öffnen

2. Nach **SimpleForm** suchen

3. **SimpleForm** in der Liste auswählen

4. Das erste Beispiel auswählen

1. Code-Ansicht öffnen

```

4   xmlns:f="sap.ui.layout.form"
5   xmlns:core="sap.ui.core">
6   <f:SimpleForm id="SimpleFormChange354"
7     editable="true"
8     layout="ResponsiveGridLayout"
9     title="Address"
10    labelSpanXL="3"
11    labelSpanL="3"
12    labelSpanM="3"
13    labelSpanS="12"
14    adjustLabelSpan="false"
15    emptySpanXL="4"
16    emptySpanL="4"
17    emptySpanM="4"
18    emptySpanS="0"
19    columnsXL="1"
20    columnsL="1"
21    columnsM="1"
22    columnsS="1"
23    singleContainerFullSize="false" >
24    <f:content>
25      <Label text="Name" />
26      <Input id="name" value="{SupplierName}" />
27      <Label text="Street/No." />
28      <Input value="{Street}" />
29      <Input value="{HouseNumber}" />
30      <layoutData>
31        <l:GridData span="XL1 L2 M2 S4" />
32      </layoutData>
33    </Input>
34    <Label text="ZIP Code/City" />
35    <Input value="{ZIPCode}" />
36    <layoutData>
37      <l:GridData span="XL1 L2 M2 S4" />
38    </layoutData>
39  </Input>
40  <Label value="{City}" />
41  <Label text="Country" />
42  <Select id="country" selectedKey="{Country}" />
43    <items>
44      <item key="DE" text="Germany" />
45      <item key="US" text="USA" />

```

1. Code-Ausschnitt zwischen und inkl. <SimpleForm> ... </SimpleForm> aus Change.fragment.xml kopieren

```

9   <Page id="page" title="{i18n>title}">
10  <content>
11    <f:SimpleForm id="app_simpleform"
12      editable="true"
13      layout="ResponsiveGridLayout"
14      title="Address" 1
15      labelSpanXL="3"
16      labelSpanL="3"
17      labelSpanM="3"
18      labelSpanS="12"
19      adjustLabelSpan="false"
20      emptySpanXL="4"
21      emptySpanL="4"
22      emptySpanM="4"
23      emptySpanS="0"
24      columnsXL="1"
25      columnsL="1"
26      columnsM="1"
27      columnsS="1"
28      singleContainerFullSize="false" >
29      <f:content> 2
30        <Label id="app_label_customerid" text="{i18n>app.customerid}" labelFor="app_text_customerid"/>
31        <Text id="app_text_customerid" text="1000001"/>
32      </f:content>

```

1. Titel anpassen

2. Inhalt aus HBox in den Inhaltsbereich zwischen <f:content> und </f:content> einfügen.

```
<f:SimpleForm id="app_simpleform"
    editable="true"
    layout="ResponsiveGridLayout"
    title="{i18n>title}"
    labelSpanXL="3"
    labelSpanL="3"
    labelSpanM="3"
    labelSpanS="12"
    adjustLabelSpan="false"
    emptySpanXL="4"
    emptySpanL="4"
    emptySpanM="4"
    emptySpanS="0"
    columnsXL="1"
    columnsL="1"
    columnsM="1"
    singleContainerFullSize="false" >
<f:content>
    <Label id="app_label_customerid" text="{i18n>app.customerid}"
        labelFor="app_text_customerid"/>
    <Text id="app_text_customerid" text="1000001"/>
</f:content>
</f:SimpleForm>
```

```
Welcome      App.view.xml      i18n.properties X
webapp > i18n > i18n.properties
1 # This is the resource bundle for at.clouddna.training00.zhoui5
2
3 #Texts for manifest.json
4
5 #XTIT: Application name
6 appTitle=HOUIS
7
8 #YDES: Application description
9 appDescription=A Fiori application.
10 #XTIT: Main view title
11 title=Kunden
12
13 #App View:
14 app.customerid=Kundennummer
```

1. Text für Kundennummer im i18n aufnehmen

Fehlender Namensraum / Import

Beim Starten der Applikation werden Sie in der Browser-Konsole eine entsprechende Fehlermeldung sehen. Diese Fehlermeldung besagt, dass ein gewisser Namensraum nicht gefunden worden ist.

(i) Nicht vergessen: Namensräume/Bibliotheken müssen importiert werden, damit UI-Elemente aus diesen Bibliotheken verwendet werden können.

```
<mvc:View controllerName="at.clouddna.training00.zhoui5.controller.
    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
    xmlns:f="sap.ui.layout.form"
    xmlns="sap.m"
```

1. Namensraum **sap.ui.layout.form** importieren

2. Namensraum **sap.ui.core** importieren

```
xmlns:f="sap.ui.layout.form"  
xmlns:core="sap.ui.core"
```

1.4 Formular erweitern

MK

Martin Koch

Formular anpassen und erweitern

Weitere Informationen sollen zum Kunden aufgenommen und änderbar (=Eingabeformular) angeboten werden:

- customerid – Kundennummer: 1000001
- firstname – Vorname: Max
- lastname – Nachname: Mustermann
- title – Titel: Dr.
- gender – Geschlecht: männlich
- email – EMail: max.mustermann@clouddna.at
- phone – Telefon: +43676123123
- website – Website: www.clouddna.at

sap.m.Input

Ermöglicht dem Benutzer, Text oder numerische Werte in einer Zeile einzugeben und zu bearbeiten.

Most used Properties:

- id – ID des Controls
- value – Wert
- placeholder – Text, der bei einem leeren Inputfeld angezeigt wird
- editable – Feld eingabebereit

sap.m.Select

Dieses Control stellt eine Liste von Elementen bereit, aus der Benutzer ein Element auswählen können.

Most used properties:

- id – ID des Controls
- selectedKey – Schlüssel des ausgewählten Elements

Most used aggregations:

- items – Aggregation für die Elemente im Select

sap.ui.core.Item

Dient als Konstruktor für ein neues Item.

Most used Properties:

- key – Gespeicherter Wert
- text – Angezeigter Wert

```
10      <Page id="page" title="{i18n>title}">
11          <content>
12              <f:SimpleForm id="app_simpleform"
13                  editable="true"
14                  layout="ResponsiveGridLayout"
15                  title="{i18n>title}"
16                  labelSpanXL="3"
17                  labelSpanL="3"
18                  labelSpanM="3"
19                  labelSpans="12"
20                  adjustLabelSpan="false"
21                  emptySpanXL="4"
22                  emptySpanL="4"
23                  emptySpanM="4"
24                  emptySpans="0"
25                  columnsXL="1"
26                  columnsL="1"
27                  columnsM="1"
28                  singleContainerFullSize="false" >
29                  <f:content>
30                      <Label id="app_label_customerid" text="{i18n>app.customerid}" labelFor="app_input_customerid"/>
31                      <Input id="app_input_customerid" value="1000001"/>
32                  </f:content>
33          </f:SimpleForm>
```

1. Text auf Input ändern

"text"-property auf "value" ändern

Zwischenergebnis

Kunden

Kunden

Kundennummer:

Formular eingabebereit

```

27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
    columnsL="1"
    columnM="1"
    singleContainerFullSize="false" >
<f:content>
    <Label id="app_label_customerid" text="{i18n>app.customerid}" labelFor="app_input_customerid"/>
    <Input id="app_input_customerid" value="1000001"/>

    <Label id="app_label_firstname" text="{i18n>app.firstname}" labelFor="app_input_firstname"/>
    <Input id="app_input_firstname" value="Max"/>

    <Label id="app_label_lastname" text="{i18n>app.customerid}" labelFor="app_input_lastname"/>
    <Input id="app_input_lastname" value="Mustermann"/>

    <Label id="app_label_title" text="{i18n>app.title}" labelFor="app_input_title"/>
    <Input id="app_input_title" value="Dr."/>

    <Label id="app_label_gender" text="{i18n>app.gender}" labelFor="app_select_gender"/>
    <Select id="app_select_gender">
        <items>
            <core:Item id="app_item_female" key="female" text="{i18n>female}"/>
            <core:Item id="app_item_male" key="male" text="{i18n>male}"/>
        </items>
    </Select>

    <Label id="app_label_email" text="{i18n>app.email}" labelFor="app_input_email"/>
    <Input id="app_input_email" value="max.mustermann@clouddna.at"/>

    <Label id="app_label_phone" text="{i18n>app.phone}" labelFor="app_input_phone"/>
    <Input id="app_input_phone" value="+43676123123"/>

    <Label id="app_label_website" text="{i18n>app.website}" labelFor="app_input_website"/>
    <Input id="app_input_website" value="https://clouddna.at"/>
</f:content>
</f:SimpleForm>
```

1

1. SimpleForm um alle Felder erweitern

```

<f:content>
    <Label id="app_label_customerid" text="{i18n>app.customerid}"
           labelFor="app_text_customerid"/>
    <Text id="app_text_customerid" text="1000001"/>

    <Label id="app_label_firstname" text="{i18n>app.firstname}"
           labelFor="app_input_firstname"/>
    <Input id="app_input_firstname" value="Max"/>

    <Label id="app_label_lastname" text="{i18n>app.lastname}"
           labelFor="app_input_lastname"/>
    <Input id="app_input_lastname" value="Mustermann"/>

    <Label id="app_label_title" text="{i18n>app.title}"
           labelFor="app_input_title"/>
    <Input id="app_input_title" value="Dr."/>

    <Label id="app_label_gender" text="{i18n>app.gender}"
           labelFor="app_select_gender"/>
    <Select id="app_select_gender">
        <items>
            <core:Item id="app_item_female" key="female"
                       text="{i18n>female}"/>
            <core:Item id="app_item_male" key="male"
                       text="{i18n>male}"/>
        </items>
    </Select>

    <Label id="app_label_email" text="{i18n>app.email}"
           labelFor="app_input_email"/>
    <Input id="app_input_email" value="max.mustermann@clouddna.at"/>

    <Label id="app_label_phone" text="{i18n>app.phone}"
           labelFor="app_input_phone"/>
    <Input id="app_input_phone" value="+43676123123"/>

    <Label id="app_label_website" text="{i18n>app.website}"
           labelFor="app_input_website"/>
    <Input id="app_input_website" value="https://clouddna.at"/>
</f:content>

```

```
Welcome App.view.xml i18n.properties X  
webapp > i18n > i18n.properties  
-  
6 appTitle=HOUIS  
7  
8 #YDES: Application description  
9 appDescription=A Fiori application.  
10 #XTIT: Main view title  
11 title=Kunden  
12  
13 #General  
14 female=weiblich  
15 male=männlich  
16  
17 #App View:  
18 app.customerid=Kundennummer  
19 app.firstname=Vorname  
20 app.lastname=Nachname  
21 app.title=Titel  
22 app.gender=Geschlecht  
23 app.email=E-Mail  
24 app.phone=Tel.nr.  
25 app.website=Homepage
```

1. Die weiteren i18n-Texte hinzufügen

```
#General  
female=weiblich  
male=männlich  
  
#App View:  
app.customerid=Kundennummer  
app.firstname=Vorname  
app.lastname=Nachname  
app.title=Titel  
app.gender=Geschlecht  
app.email=E-Mail
```

```
app.phone=Telefon  
app.website=Website
```

Das Ergebnis

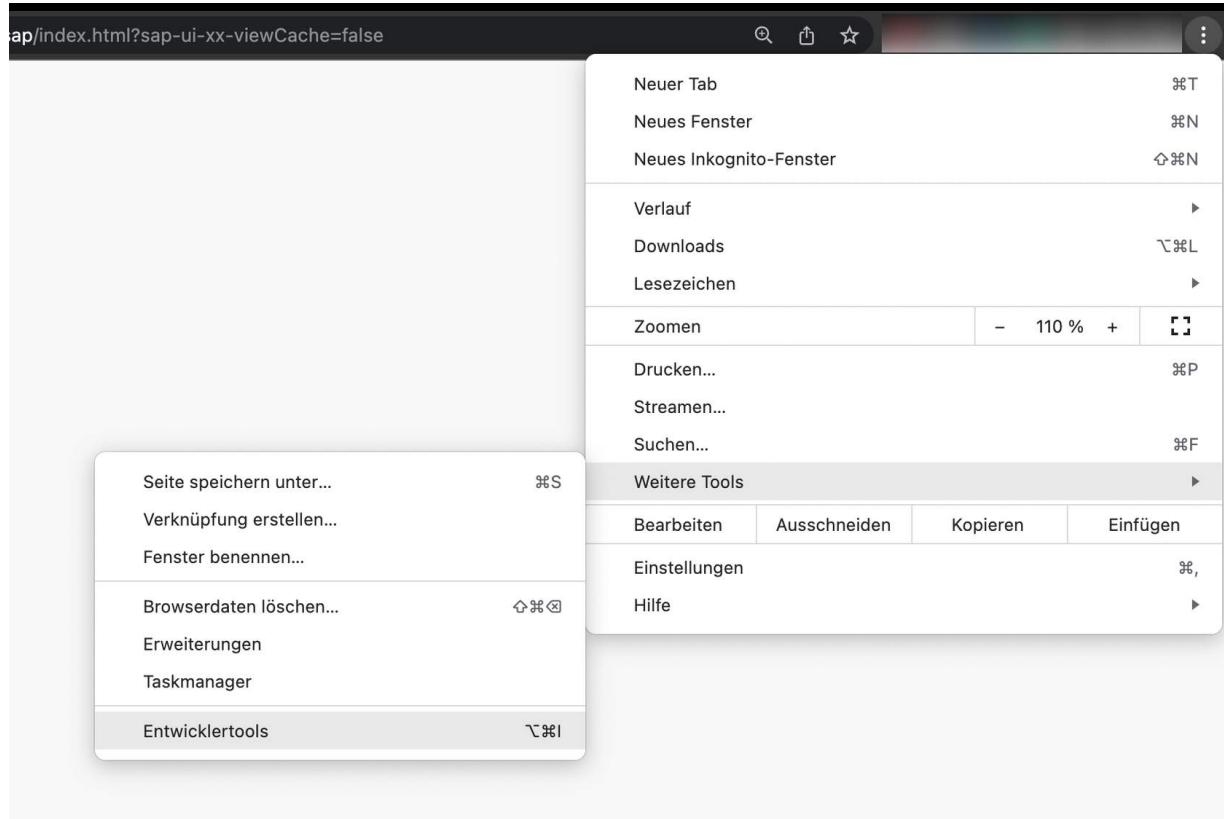
The screenshot shows a SAPUI5 application interface for a customer record. The top navigation bar has the title "Kunden". Below it, there is a header with the word "Kunden". The main content area contains a form with the following fields and their values:

Kundennummer:	1000001
Vorname:	Max
Nachname:	Mustermann
Titel:	Dr.
Geschlecht:	weiblich
E-Mail:	max.mustermann@clouddna.at
Tel.nr.:	+43676123123
Homepage:	https://clouddna.at

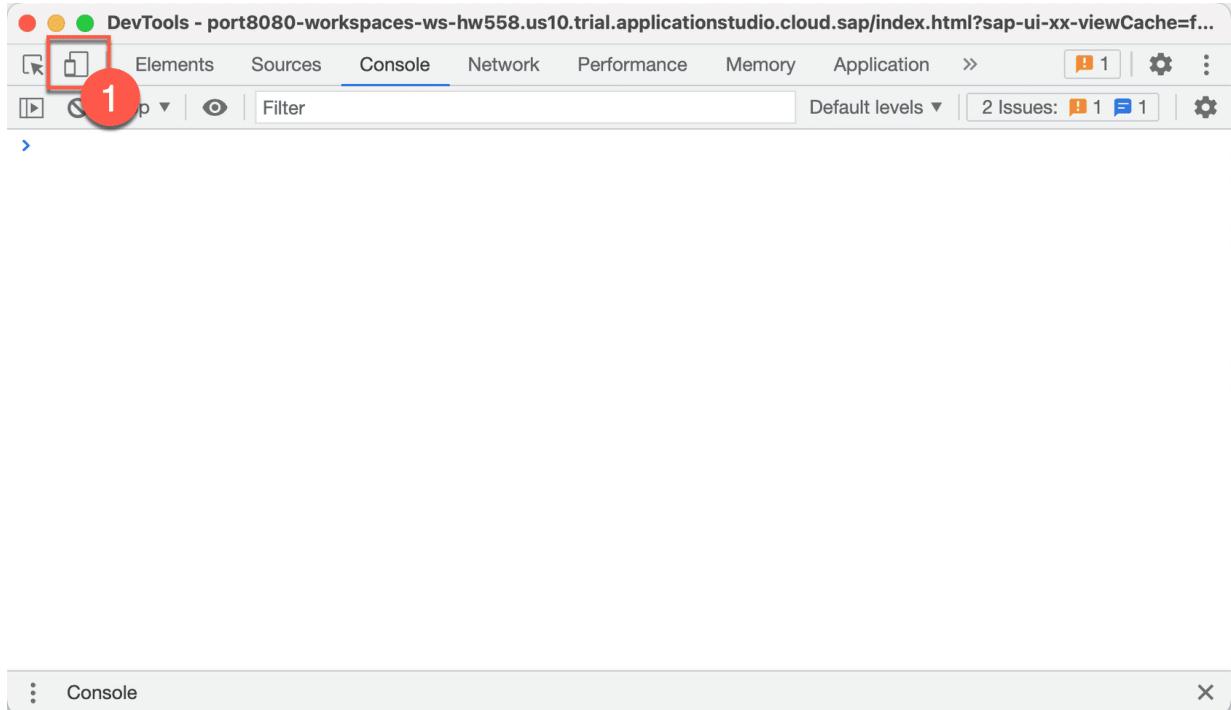
Übung 1 - Ergebnis

Responsive Design dank ResponsiveGridLayout

Dank des ResponsiveGridLayout in SAPUI5 können wir ein ansprechendes Design realisieren, das sich optimal an unterschiedliche Bildschirmgrößen und Gerätetypen anpasst. Dieses Layout ermöglicht eine flexible Gestaltung von Benutzeroberflächen, die eine konsistente und benutzerfreundliche Darstellung auf verschiedenen Endgeräten gewährleistet.



1. Entwicklertools öffnen



1. Embedded Device Emulator starten

A screenshot of a SAP Fiori application titled "Kunden". The application displays a form for creating a customer record. The fields and their values are:

- Kundennummer: 1000001
- Vorname: Max
- Nachname: Mustermann
- Titel: Dr.
- Geschlecht: männlich
- E-Mail: max.mustermann@clouddna.at
- Tel.nr.: +43676123123
- Homepage: www.clouddna.at

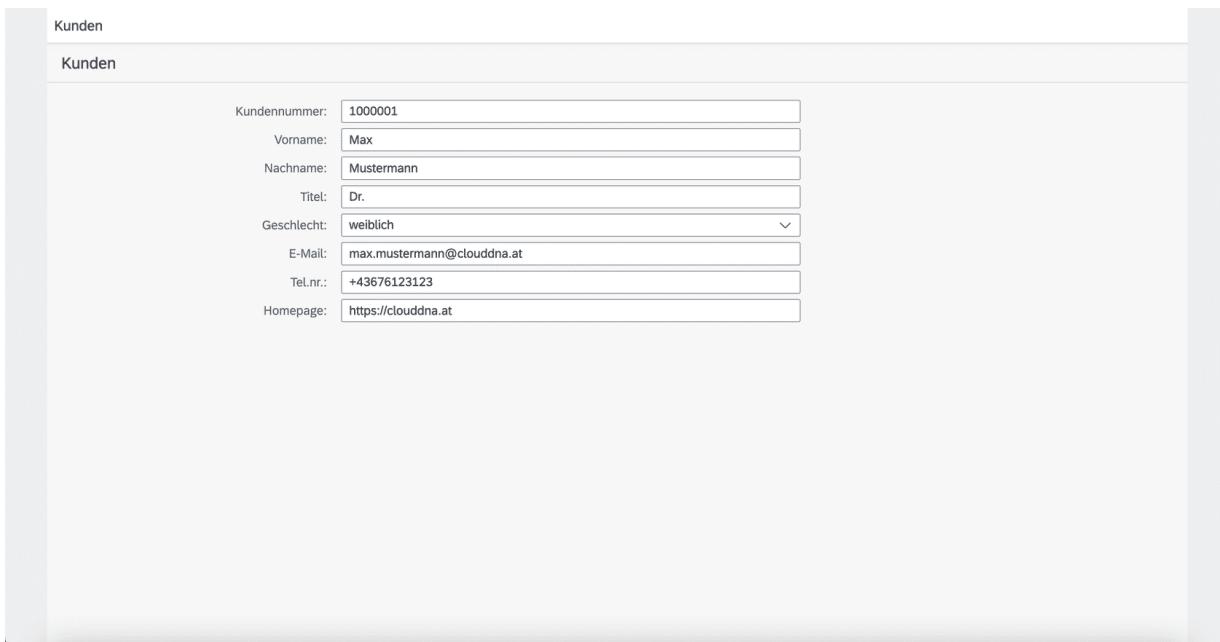
The application interface includes a header with "Dimensions: Responsive" and "No throttling" settings, and a toolbar with various icons.

1. Responsive Design testen

1.5 Zusammenfassung

 Martin Koch

Es wurde eine Single-Page Applikation erstellt, die einfache Kundendaten anzeigt. Neue verwendete Controls sind: HBox zur Layoutanordnung, Text und Label zur Datenanzeige. Danach wurde die View auf eine SimpleForm umgebaut, um eine schönere, vereinheitlichte Übersicht der Kundendaten zu ermöglichen. Weiteres wurde dieses Formular mittels Input Controls eingabebereit gemacht.



The screenshot shows a JavaFX application window titled "Kunden". Inside, there is a sub-titled section also labeled "Kunden". Below this, a form is displayed with the following fields and values:

Kundennummer:	1000001
Vorname:	Max
Nachname:	Mustermann
Titel:	Dr.
Geschlecht:	weiblich
E-Mail:	max.mustermann@clouddna.at
Tel.nr.:	+43676123123
Homepage:	https://clouddna.at

Übung 1 - Ergebnis

2 Einleitung

 Martin Koch

Übung 2 – Git

Damit Sie die Inhalte aus dem HOU15 längerfristig behalten können und in Hinblick auf eine eventuelle zukünftige Source-Code-Verwaltung mittels Git, werden wir in dieser Übung Git anbinden. Nach jeder Übung werden wir einen Commit absetzen, damit die Historie auch dem Kursverlauf entspricht.

Verwendete Controls: -

Verwendete Technologien: [Git](#)

2.1 Git Grundlagen (optional)

MK

Martin Koch

In dieser Übung werden die wichtigsten Begrifflichkeiten und Funktionen rund um Git besprochen.

Repositories

Repositories können Sie sich wie Verzeichnisse vorstellen, die einen bestimmten Namen haben und Dateien speichern. Die Dateien können verschiedene Typen haben und müssen keinen ausführbaren Programmcode oder ein funktionsfähiges Projekt beinhalten.

Ein Repository merkt sich alle Änderungen, die im Laufe des Projekts gemacht worden sind, sodass Sie auch im Nachhinein Schritt für Schritt die Entwicklungen der Dateien in Form einer Historie (engl. History) abrufen können.

Grundsätzlich werden bei Repositorys zwei Arten unterschieden. Es gibt lokale Repositorys und Remote-Repositorys:

- **Lokale Repositorys** sind die, die wirklich bei der Entwicklerin beziehungsweise beim Entwickler liegen, egal ob auf einem privaten Computer oder in einer IDE in der Cloud. Dieses Repository ist nur für diese eine Person sichtbar und ist weder öffentlich noch von anderen Personen oder Personengruppen einsehbar.
- **Remote-Repositorys** hingegen liegen entweder selbst gehostet innerhalb des Unternehmensnetzwerks oder bei einem externen Anbieter in der Cloud. Auf diese Repositorys könnten auch mehrere Personen zugreifen. Im besten Fall sind diese Repositorys entweder selbst durch das Anbinden von zum Beispiel eines Active Directories

abgesichert oder in der Cloud sind bestimmte Benutzer in bestimmten Rollen als Mitwirkende eingetragen.

Branches

Git bringt eine besondere Möglichkeit der Source-Code-Verwaltung und Versionierung mit. Sie können Ihre Entwicklung in mehrere Stränge bzw. Abzweigungen aufteilen und diese getrennt voneinander weiterentwickeln, sie wieder zusammenführen oder einzelne Abzweige gar verwerfen. Diese Abzweigungen werden auch Branches (englisch für Ast) genannt.

Ein Repository kann aus mehreren solchen Branches bestehen, jedoch gibt es auch immer einen Hauptstrang, der sich in den meisten Fällen Main-Branch nennt. Dieser Name kann übersteuert bzw. mit einer Konfigurationsmöglichkeit festgelegt werden.

Clone

Mit diesem Befehl bestimmen Sie, dass Sie lokal ein genaues Abbild eines Remote-Repositories haben möchten, gegen das Sie arbeiten.

Commit

Die kleinsten messbaren Einheiten im Git-Umfeld nennen sich Commits und sind quasi eine Momentaufnahme (engl. Snapshot) des aktuellen Entwicklungsstands.

Ein Commit wird mittels eines Befehls durch eine*n Entwickler*in ausgelöst und beinhaltet zumeist alle Änderungen, die seit dem letzten Commit gemacht wurden.

Es stellt sich natürlich die Frage, welche Änderungen, die seit dem letzten Commit getätigter wurden, mit dem nächsten Commit abgespeichert werden. Es sind all jene Änderungen, die sich in der sogenannten Staging Area befinden. Die Staging Area ist ein Bereich, der eine

ausgewählte Anzahl an Änderungen hält, welche im nächsten Commit aufgenommen werden. Welche Files es in die Staging Area schaffen, haben Sie in der Hand.

Fetch

In den meisten Fällen arbeiten Sie natürlich nicht allein. Teamkolleg*innen arbeiten ebenso am Projekt und teilen ihre Änderungen im Remote-Repository mit Ihnen und allen anderen.

Ihr lokales Repository weiß von diesen Änderungen nichts, solange Sie nicht dafür sorgen, dass es nachsehen und die Änderungen herunterladen soll. Jedoch wissen Sie bis zu diesem Zeitpunkt noch nicht, ob es irgendwelche Änderungen gegeben hat und wenn ja, welche und wie viele neue Commits sich am Remote-Repository befinden.

Mit einem Fetch haben Sie in der Hand, wann Ihr lokales Repository nachsehen soll, ob sich etwas am Remote-Repository geändert hat. Mit einem Fetch prüft Ihr lokales Repository, ob sich etwas an allen verbundenen Remote-Repositories geändert hat und lädt die Änderungen auch herunter.

Merge

Sie wissen jetzt dank Ihrer Fetch-Abfrage, dass sich am Remote-Repository etwas getan hat. Jetzt geht es darum, dass Sie den Stand, den Sie gerade heruntergeladen haben, mit Ihrem Ist-Stand zusammenführen. Dieser Vorgang nennt sich Merge.

Das Zusammenführen von Dateien kann sich als eine heikle Angelegenheit herausstellen, vor allem, wenn es zu Konflikten kommt.

Push

Eine zusammengeführte Version hat, solange Sie sie nicht wieder mit anderen teilen, keinen wirklichen Mehrwert, außer, dass Sie lokal auf dem neuesten Stand der Entwicklungen sind.

Damit Sie Ihren Entwicklungsstand mit anderen teilen können, müssen Sie einen sogenannten Push ausführen. Dieser Befehl versucht alle Commits, die seit dem letzten Push entstanden sind, Richtung Remote-Repository zu schicken und so anderen zur Verfügung zu stellen. Sie pushen (englisch für schieben) also Ihre Änderungen in das Remote-Repository.

Pull

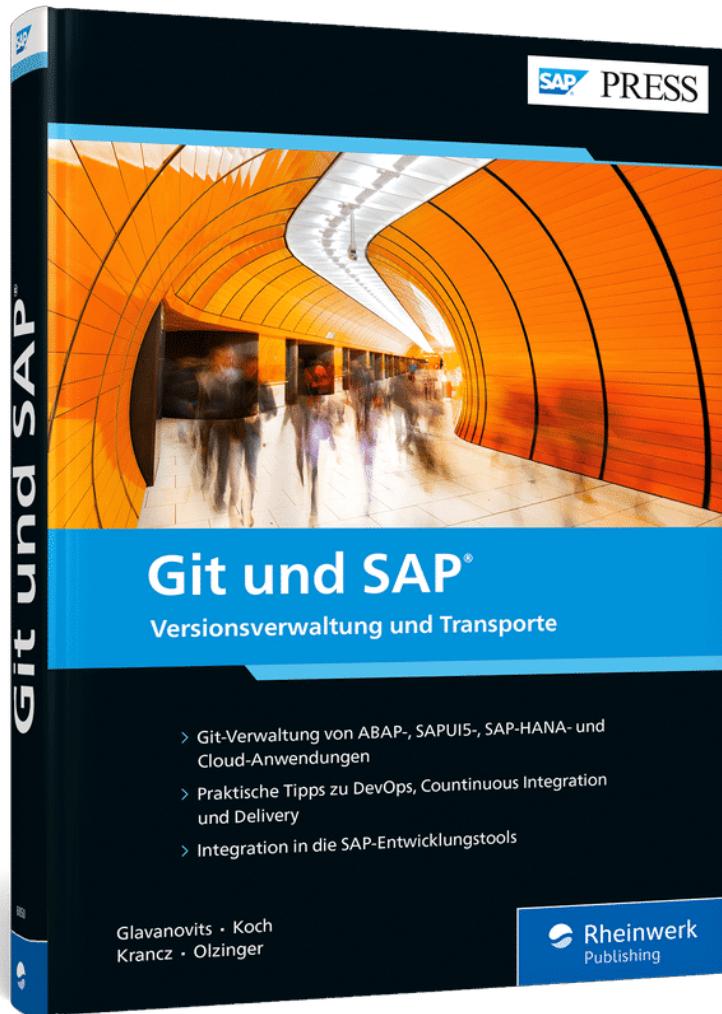
Sie werden sich wahrscheinlich schon denken, dass wenn man vor einem Push jedes Mal Fetch und Merge ausführen muss, dann treten diese zwei Befehle ja fast immer in Verbindung auf. Ja, das stimmt und Sie können sich mit dem Befehl Pull Abhilfe schaffen.

Ein Pull führt zuerst einen Fetch und danach einen Merge aus.

Grundlagen-Video | commit, fetch, merge, pull, push

Anbei finden Sie den ersten Teil unserer Videoreihe zu Git eingebettet. Dieses Video finden Sie auch auf unserem [YouTube-Kanal](#).





<https://www.rheinwerk-verlag.de/git-und-sap/>

Git und SAP

Nutzen Sie das moderne Versionsmanagement für Ihre SAP-Projekte! In diesem Buch erfahren Sie, wie Git in die Entwicklungsumgebungen für SAPUI5 oder ABAP integriert ist. Sie lernen, wie Sie Git-Befehle direkt in SAP Business Application Studio, ABAP Development Tools oder SAP Web IDE absetzen, Repositorys klonen und Branches verwalten. Ihre Entwicklungsteams werden nicht mehr anders arbeiten wollen!

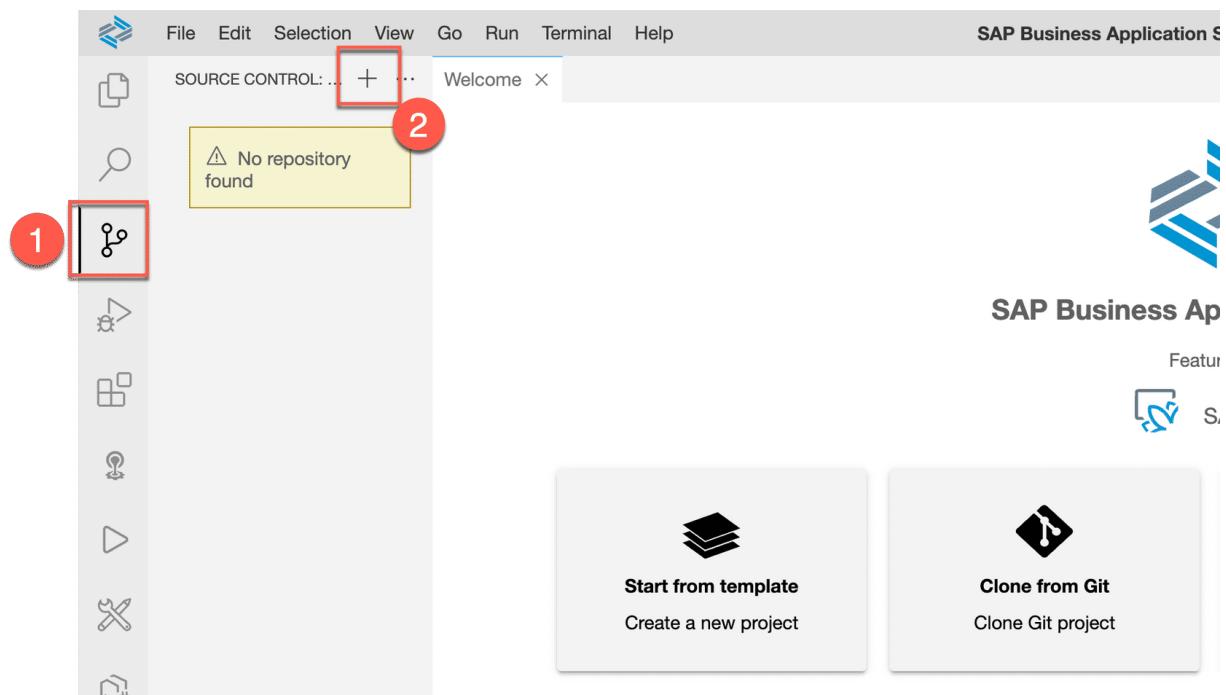
- Git-Verwaltung von Anwendungen in ABAP, SAPUI5 und in der Cloud
- Praktische Tipps zu DevOps, Continuous Integration und Delivery

- Einbindung in die SAP-Entwicklungstools

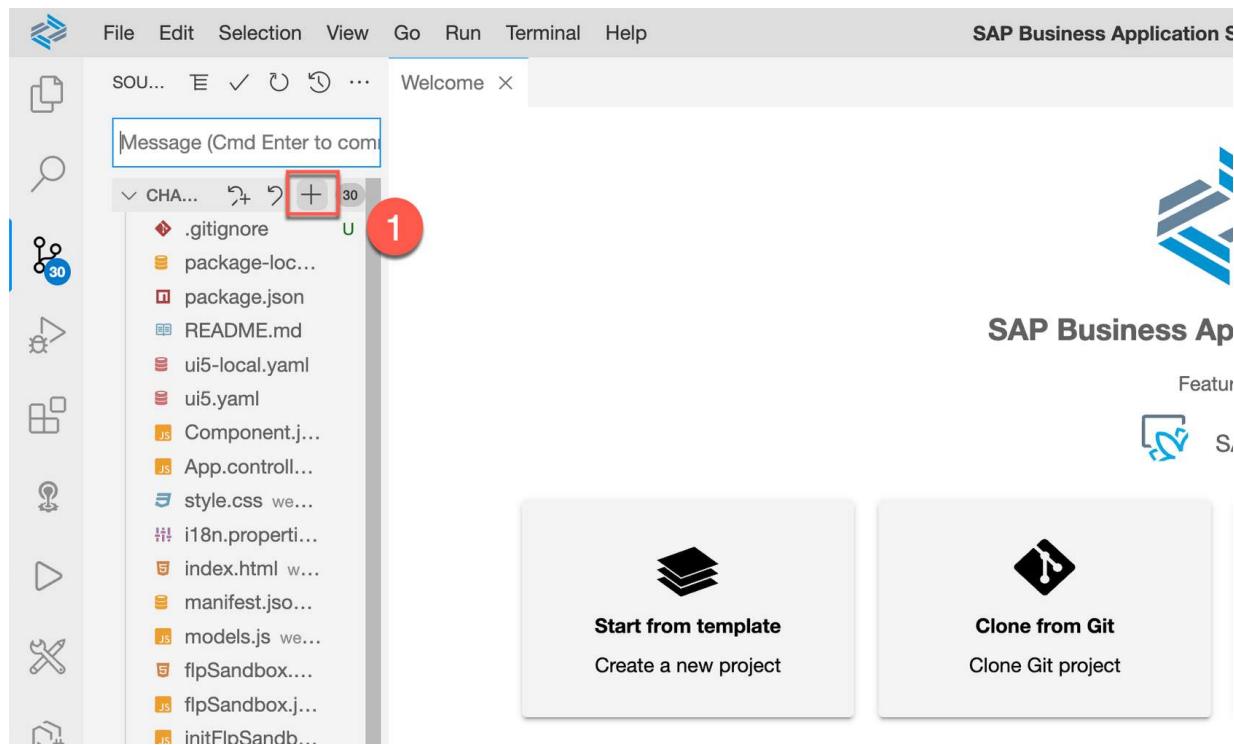
2.2 Git anbinden

MK Martin Koch

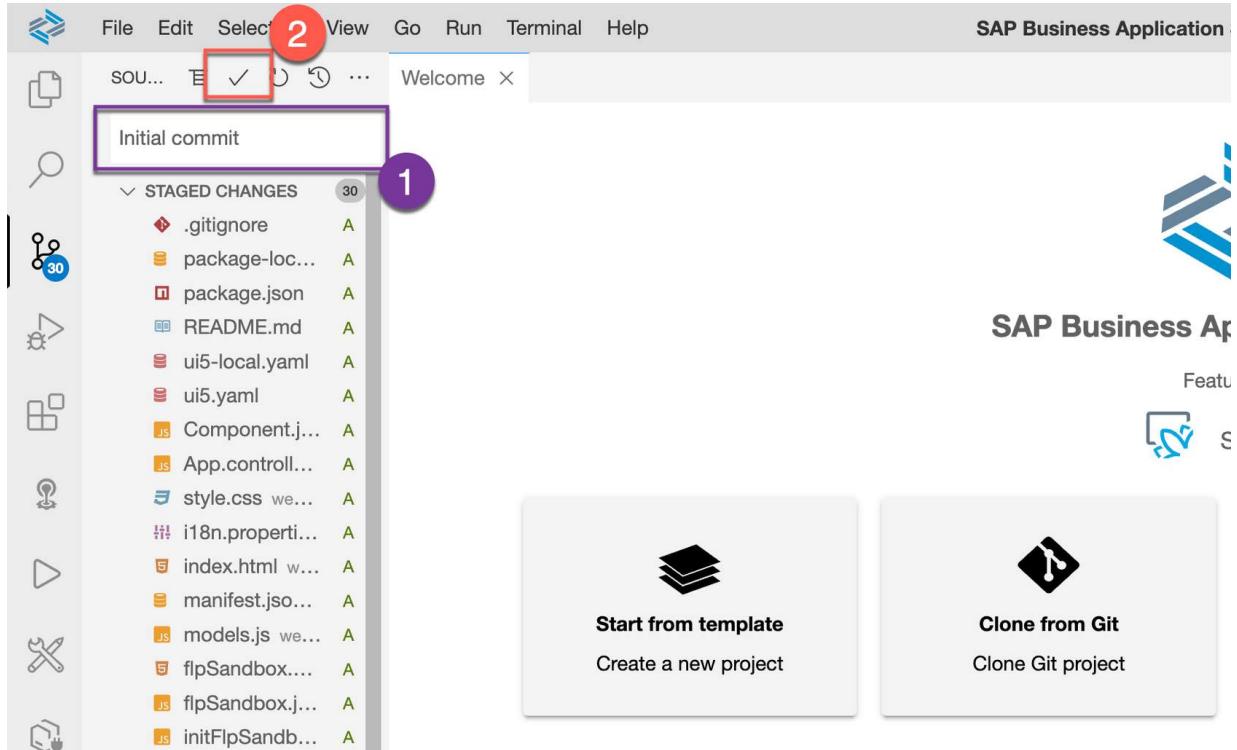
Das SAP Business Application Studio besitzt eine Git-Integration. Im Fenster Source Control muss zunächst ein lokales Repository angelegt werden, welches anschließend mit dem Remote-Repository verbunden werden muss.



1. Zu **Source Control** wechseln
2. Lokales Repository initialisieren

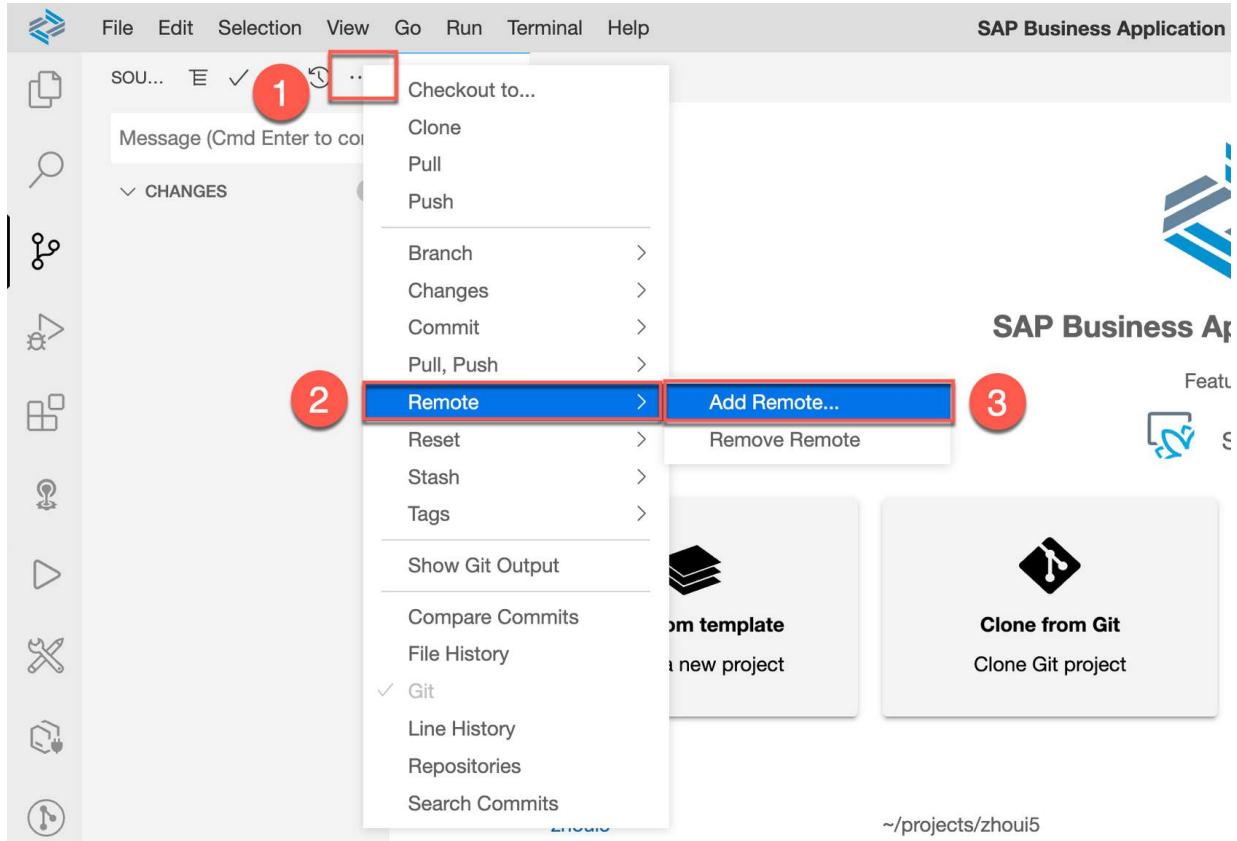


1. Alle Änderungen in die Staging Area bringen



1. Commit Nachricht eingeben (Beispiel: **Initial commit**)

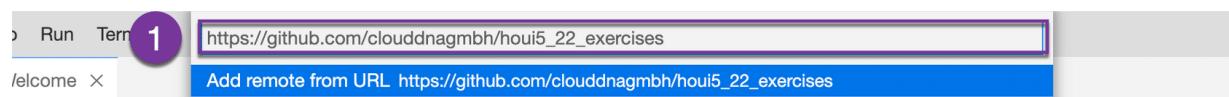
2. Commit ausführen



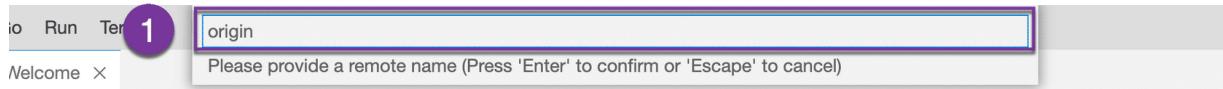
1. Git - Einstellungen öffnen

2. Remote auswählen

3. Add Remote... auswählen



1. URL des Remote-Repositorys hinzufügen



1. Namen für die Verbindung vergeben

A screenshot of the GitHub developer settings page. Step 1 highlights the 'Settings / Developer settings' link. Step 2 highlights the 'Personal access tokens' section. Step 3 highlights the 'Generate new token' button. The page shows two existing tokens: one for 'admin' with scopes 'admin:enterprise, admin:gg, key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, delete:packages, delete_repo, gist, notifications, repo, user, workflow, write:discussion, write:packages' and another for 'RiskManagement' with scopes 'admin:repo_hook, repo, user'. Both tokens were last used within the last 4 weeks. A note at the bottom explains that personal access tokens function like OAuth access tokens and can be used instead of a password for Git over HTTPS or to authenticate to the API over Basic Authentication.

Search or jump to... Pull requests Issues Marketplace Explore

1 Settings / Developer settings

2 Personal access tokens

3 Generate new token Revoke all

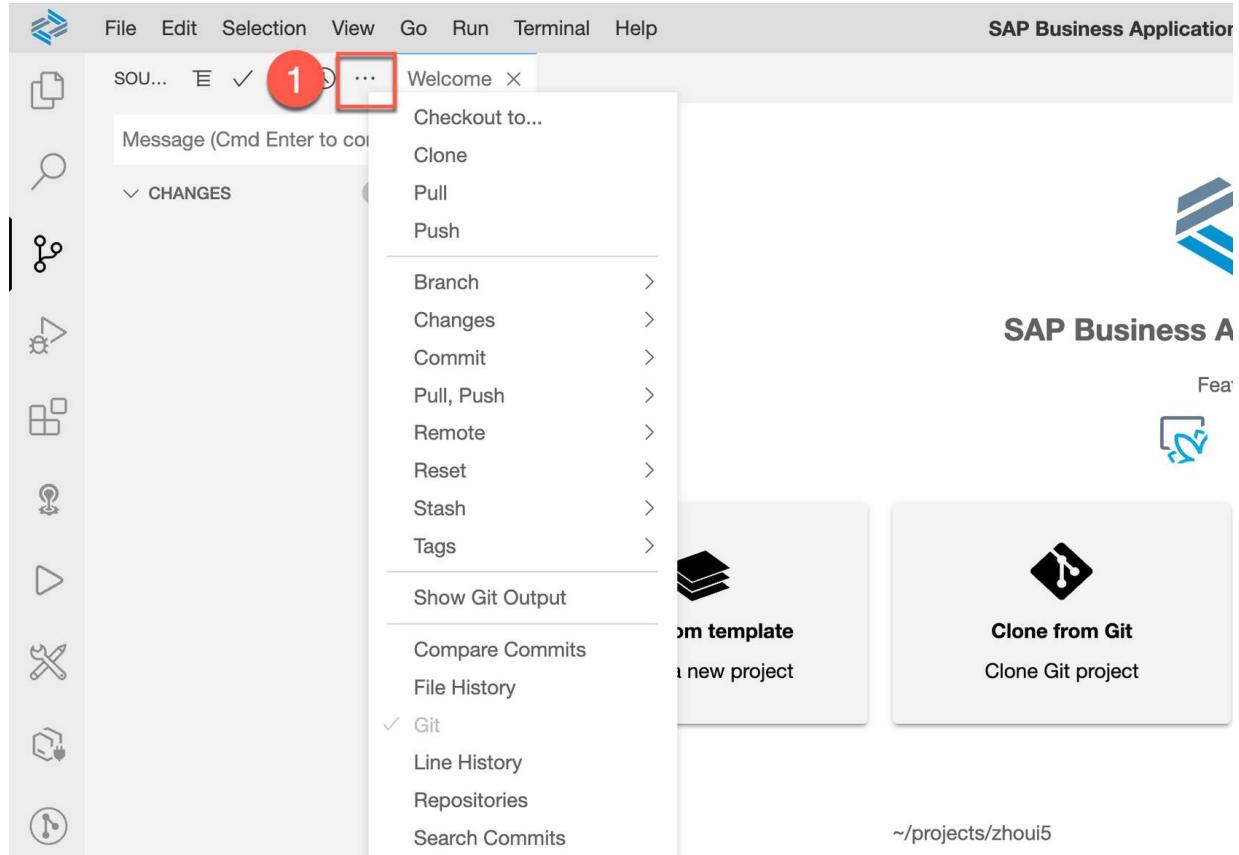
Tokens you have generated that can be used to access the GitHub API.

User	Scopes	Last used	Action
admin	admin:enterprise, admin:gg, key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, delete:packages, delete_repo, gist, notifications, repo, user, workflow, write:discussion, write:packages	Within the last 4 weeks	Delete
RiskManagement	admin:repo_hook, repo, user	Within the last 3 months	Delete

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

© 2022 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

1. Token für Login in GitHub erstellen



1. Bei den Git-Einstellungen den ersten Push ausführen

2.3 Zusammenfassung

 Martin Koch

In dieser Übung haben wir Git kennengelernt und ein lokales Repository angelegt. Das lokale Repository wurde mit einem Remote-Repository verknüpft. Über das Remote-Repository könnten wir unsere Entwicklungen innerhalb des Entwicklungsteams verteilen.

3 Einleitung

MK

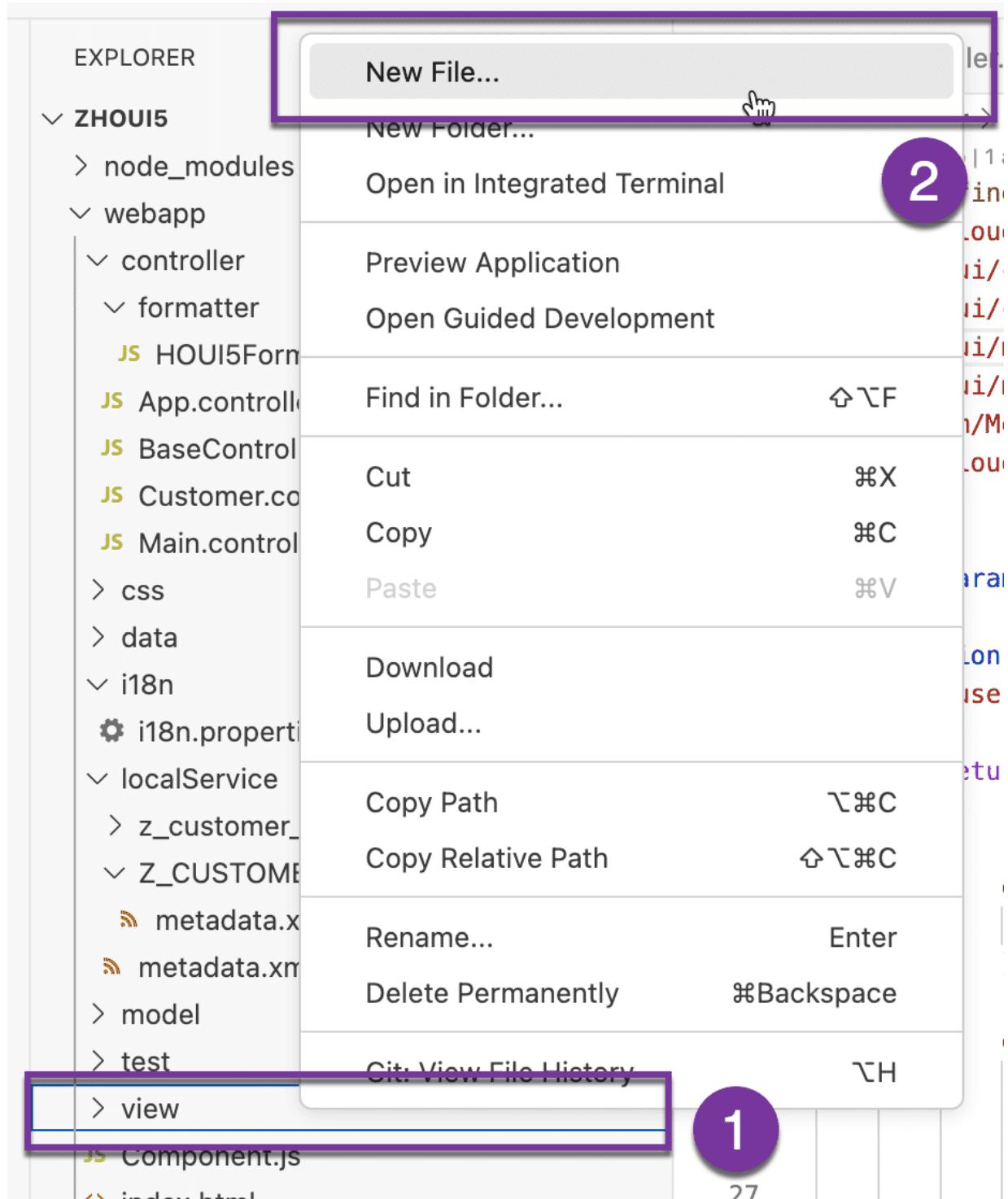
Martin Koch

Nach der erfolgreichen Erstellung und Einrichtung unserer Applikation beginnen wir in dieser Übung nun damit, unsere Anwendung mit Leben zu füllen. Es werden die ersten UI-Elemente kennengelernt und gezeigt, wie man mit den SAPUI5-Controls umgeht. Am Ende dieser Übung haben wir eine Übersichtsseite erstellt, die auf einfachste Art und Weise eine Tabelle mit Kundendaten anzeigt.

3.1 SemanticPage erstellen

 MK Martin Koch

Zuallererst erstellen wir in der Main-View ein **SemanticPage**. Dabei ersetzen wir das vorhandene Page-Control durch das **SemanticPage**-Control und fügen ebenfalls den Namespace dafür hinzu.

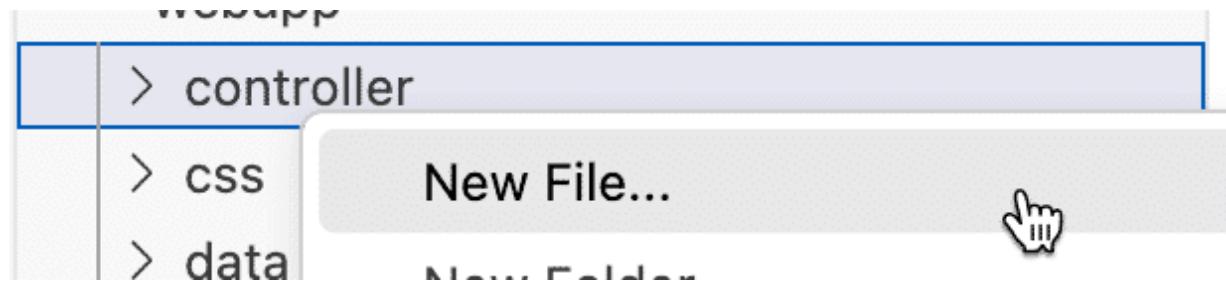


1. Rechtsklick auf Ordner **view**

2. Linksklick auf **new File**



File **Main.view.xml** benennen



Wie auch im vorherigen Schritt im Ordner **controller** ein neues File **Main.controller.js** erstellen

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

    return Controller.extend("at.clouddna.training##.zhoui5.controller.Main",
        {
            onInit: function() {

            },
        });
});
```

Controller mit folgendem Code befüllen

Die [SAPUI5 Semantic Page](#) ist ein spezielles Steuerelement für die Erstellung von benutzerfreundlichen Oberflächen. Sie bietet vordefinierte Abschnitte für Header, Inhalt und Fußzeile, um eine einheitliche Struktur zu gewährleisten. Durch flexible Spaltensysteme und responsives Design unterstützt sie die Fiori-Designprinzipien von SAP, wodurch Entwickler schnell ansprechende Anwendungen erstellen können. Der Header enthält den Seitentitel und zusätzliche Elemente, während der Inhalt für den Hauptbereich und die Fußzeile für ergänzende Informationen oder Steuerelemente vorgesehen sind.

The screenshot shows a code editor interface for a SAPUI5 application. The tabs at the top are: Main.view.xml (selected), Main.controller.js, manifest.json, and i18n.properties. The code in Main.view.xml is as follows:

```
1 <mvc:View controllerName="at.clouddna.training00.zhoui5.controller.Main"
2   xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3   xmlns:s="sap.f.semantic" 2
4   xmlns="sap.m">
5   <s:SemanticPage id="main_page"> 1
6     <s:titleHeading>
7       <Title text="{i18n>title}" />
8     </s:titleHeading>
9     <s:content>
10    </s:content>
11  </s:SemanticPage>
12
13 </mvc:View>
```

Annotations are present in the code:

- A red box surrounds the entire `<s:SemanticPage>` element, labeled with a red circle containing the number 1.
- A red box surrounds the `xmlns:s="sap.f.semantic"` declaration, labeled with a red circle containing the number 2.

1. SemanticPage einfügen

2. Namespace für **SemanticPage** hinzufügen

```
<mvc:View controllerName="at.clouddna.training00.zhoui5.controller.Main"  
    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"  
    xmlns:s="sap.f.semantic"  
    xmlns="sap.m">  
    <s:SemanticPage id="main_page">  
        <s:titleHeading>  
            <Title text="{i18n>title}" />  
        </s:titleHeading>  
        <s:content>  
  
        </s:content>  
    </s:SemanticPage>  
</mvc:View>
```

```

"routes": [
  {
    "name": "Main",
    "pattern": ":?query:",
    "target": [
      "TargetMain"
    ]
  }
],
"targets": {
  "TargetMain": {
    "viewType": "XML",
    "transition": "slide",
    "clearControlAggregation": false,
    "viewId": "Main",
    "viewName": "Main"
  }
}

```

Im **"manifest.json"** routes und targets auf **Main** umändern

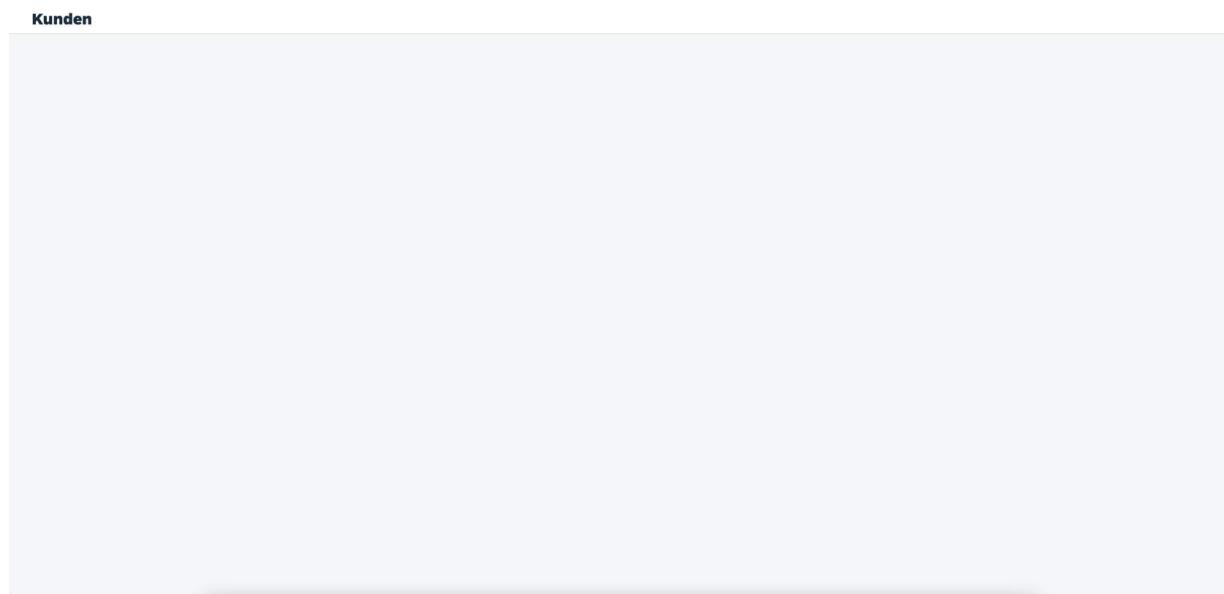
```

"routes": [
  {
    "name": "RouteMain",
    "pattern": ":?query:",
    "target": [
      "TargetMain"
    ]
  }
]

```

```
        ],
    },
],
"targets": {
    "TargetMain": {
        "viewType": "XML",
        "transition": "slide",
        "clearControlAggregation": false,
        "viewId": "Main",
        "viewName": "Main"
    },
}
```

Das Ergebnis sieht nun folgendermaßen aus:



Übersichtsseite mit SemanticPage

3.2 Tabelle erstellen

MK

Martin Koch

In diesem Abschnitt wird eine Tabelle eingeführt und mit Kundendaten befüllt. Zu Beginn werden die Daten nicht dynamisch aus einer externen Quelle geladen, sondern statisch direkt in die View integriert. Dieser statische Ansatz wird auch als "hart codiert" bezeichnet, da die Informationen fest im Programmcode eingebettet sind.

File tabs: Main.view.xml U X | JS Main.controller.js U | {} manifest.json M | i18n.properties M

webapp > view > Main.view.xml

```

1  <mvc:View controllerName="at.clouddna.training00.zhoui5.controller.Main"
2      xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3      xmlns:s="sap.f.semantic"
4      xmlns="sap.m">
5      <s:SemanticPage id="main_page">
6          <s:titleHeading>
7              <Title text="{i18n>title}" />
8          </s:titleHeading>
9          <s:content>
10             <Table id="main_table" headerText="{i18n>main.table.title}">
11                 <columns>
12                     <Column id="idMainColumn">
13                         <Text id="main_text_customerid" text="{i18n>app.customerid}" />
14                     </Column>
15                     <Column id="main_column_firstname">
16                         <Text id="main_text_firstname" text="{i18n>app.firstname}" />
17                     </Column>
18                     <Column id="main_column_lastname">
19                         <Text id="main_text_lastname" text="{i18n>app.lastname}" />
20                     </Column>
21                     <Column id="main_column_title" minScreenWidth="Desktop">
22                         <Text id="main_text_title" text="{i18n>app.title}" />
23                     </Column>
24                     <Column id="main_column_gender" minScreenWidth="Desktop">
25                         <Text id="main_text_gender" text="{i18n>app.gender}" />
26                     </Column>
27                     <Column id="main_column_email" minScreenWidth="Desktop">
28                         <Text id="main_text_email" text="{i18n>app.email}" />
29                     </Column>
30                     <Column id="main_column_phone" minScreenWidth="Desktop">
31                         <Text id="main_text_phone" text="{i18n>app.phone}" />
32                     </Column>
33                     <Column id="main_column_website" minScreenWidth="Desktop">
34                         <Text id="main_text_website" text="{i18n>app.website}" />
35                     </Column>
36                 </columns>
37                 <items>
38                     <ColumnListItem>
39                         <cells>
40                             <ObjectIdentifier title="1000001"/>
41                             <Text text="Max"/>
42                             <Text text="Mustermann"/>
43                             <Text text="Dr."/>
44                             <Text text="male"/>
45                             <Text text="max.mustermann@clouddna.at"/>
46                             <Text text="+43676123123"/>
47                             <Link text="www.clouddna.at" href="https://www.clouddna.at" target="_blank"/>
48                         </cells>
49                     </ColumnListItem>
50                 </items>
51             </Table>
52         </s:content>
53     </s:SemanticPage>
54 </mvc:View>

```

1. Tabellendefinition in die SemanticPage einfügen

```

<Table id="main_table" headerText="{i18n>main.table.title}">
    <columns>
        <Column id="idMainColumn">
            <Text id="main_text_customerid" text="{i18n>app.customerid}"/>
        </Column>
        <Column id="main_column_firstname">
            <Text id="main_text_firstname" text="{i18n>app.firstname}"/>
        </Column>
        <Column id="main_column_lastname">
            <Text id="main_text_lastname" text="{i18n>app.lastname}"/>
        </Column>
        <Column id="main_column_title" minScreenWidth="Desktop">
            <Text id="main_text_title" text="{i18n>app.title}"/>
        </Column>
        <Column id="main_column_gender" minScreenWidth="Desktop">
            <Text id="main_text_gender" text="{i18n>app.gender}"/>
        </Column>
        <Column id="main_column_email" minScreenWidth="Desktop">
            <Text id="main_text_email" text="{i18n>app.email}"/>
        </Column>
        <Column id="main_column_phone" minScreenWidth="Desktop">
            <Text id="main_text_phone" text="{i18n>app.phone}"/>
        </Column>
        <Column id="main_column_website" minScreenWidth="Desktop">
            <Text id="main_text_website" text="{i18n>app.website}"/>
        </Column>
    </columns>
    <items>
        <ColumnListItem>
            <cells>
                <ObjectIdentifier title="1000001"/>
                <Text text="Max"/>
                <Text text="Mustermann"/>
                <Text text="Dr."/>
                <Text text="male"/>
                <Text text="max.mustermann@clouddna.at"/>
                <Text text="+43676123123"/>
                <Link text="www.clouddna.at" href="https://www.clouddna.at"/>
            </cells>
        </ColumnListItem>
    </items>
</Table>

```

i18n pflegen

Um unsere Applikation auch in andere Sprachen übersetzen zu können, ist es wichtig "Internationalization"-Funktion (kurz **i18n**) zu berücksichtigen. Dazu fügen wir im zugehörigen File die zu übersetzenden Einträge ein.

```
webapp > i18n > i18n.properties
You, 2 minutes ago | 1 author (You)
1 # This is the resource bundle for at.clouddna.training00.zhoui5 You, 3 hours ago
2
3 #Texts for manifest.json
4
5 #XTIT: Application name
6 appTitle=HOUIS
7
8 #YDES: Application description
9 appDescription=A Fiori application.
10 #XTIT: Main view title
11 title=Kunden
12
13 main.table.title=Kundentabelle
14
15 app.customerid=Kundennummer
16 app.firstname=Vorname
17 app.lastname=Nachname
18 app.title=Titel
19 app.gender=Geschlecht
20 app.email=E-Mail
21 app.phone=Telefon
22 app.website=Website
```

i18n-Einträge:

1. neue hinzufügen

2. alte umbenennen

```
appTitle=HOUIS5

title=Kunden

main.table.title=Kundentabelle

app.customerid=Kundennummer
app.firstname=Vorname
app.lastname=Nachname
app.title=Titel
app.gender=Geschlecht
app.email=E-Mail
app.phone=Telefon
app.website=Website
```

Kunden

Kundentabelle								
Kundennummer	Vorname	Nachname	Titel	Geschlecht	E-Mail	Telefon	Website	
1000001	Max	Mustermann	Dr.	male	max.mustermann@clouddna.at	+43676123123	www.clouddna.at	

Übersichtsseite mit Tabelle

3.3 Zusammenfassung

 Martin Koch

Nach Abschluss dieser Übung haben wir nun eine SemanticPage, die ein Tabelle beinhaltet, erstellt. Die Tabelle wurde statisch mit Daten befüllt, um den ersten Kunden auf einfachste Art und Weise anzuzeigen.

4 Einleitung

 Martin Koch

Übung 4 Lokale Datenhaltung

Derzeit sind alle Werte in unserer Tabelle statisch und können nicht bearbeitet werden. In diesem Kapitel werden wir uns daher darauf konzentrieren, diese Werte dynamisch mit der Tabelle zu verbinden, indem wir ein JSON-Modell und Data Binding verwenden. Darüber hinaus planen wir die Integration eines Formatters, um sicherzustellen, dass die Werte zur Laufzeit verändert werden können, ohne das Data Binding zu unterbrechen.

Verwendete Controls: -

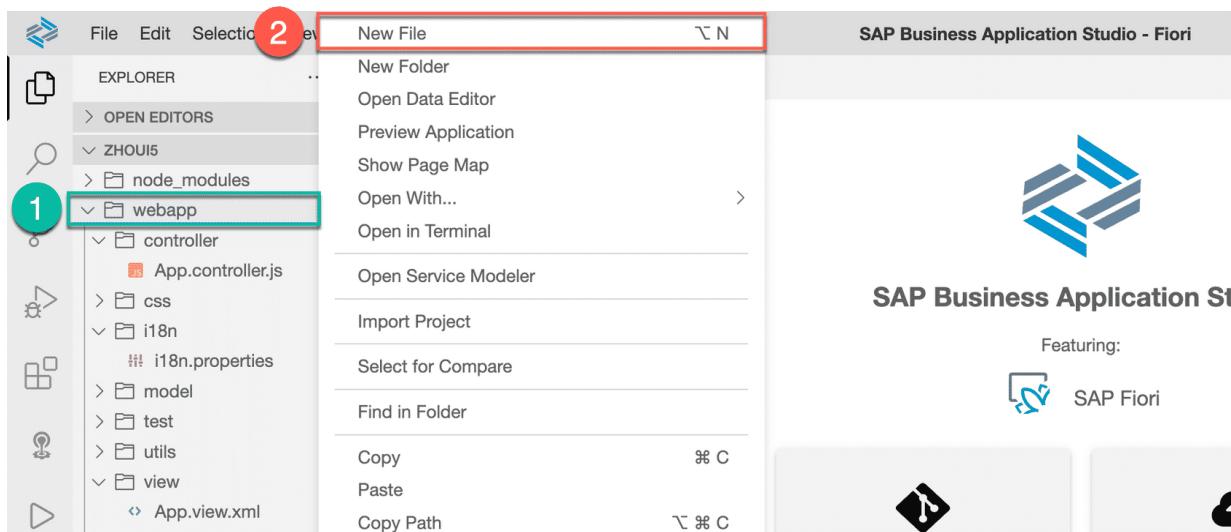
Verwendete Technologien: [JSONModel](#), [Data-Binding](#), Property Binding, Aggregation Binding, Expression Binding, [Formatter](#)

4.1 JSONModel

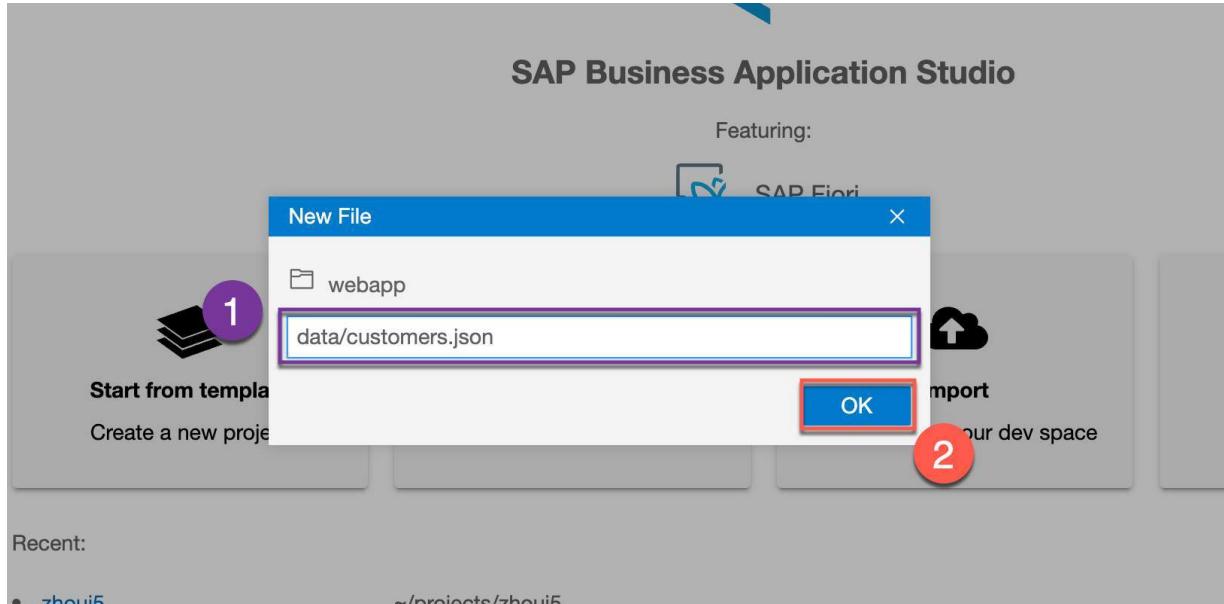
MK Martin Koch

JSON-Files anlegen

Zuerst müssen neue JSON-Dateien angelegt und anschließend mit Daten gefüllt werden.



1. Kontextmenü (=Rechtsklick) vom Verzeichnis webapp öffnen
2. Auf New File klicken



1. Dateinamen `data/customers.json` vergeben

The screenshot shows the SAP Studio IDE interface. On the left is the Explorer view, displaying the project structure under 'WORKSPACE (WORKSPACE)'. The 'customers.json' file is selected and highlighted with a blue background. In the center-right is the code editor with the 'customers.json' file open. The file contains JSON data for three customers, each with properties like Customerid, Firstname, Lastname, Title, Gender, Email, Phone, Website, and Birthdate. A red box surrounds the JSON content, and a red circle with the number '1' is located in the top right corner of the editor area.

```

1 {
  "customers": [
    {
      "Customerid": "1000001",
      "Firstname": "Max",
      "Lastname": "Mustermann",
      "Title": "Dr.",
      "Gender": "male",
      "Email": "max.mustermann@clouddna.at",
      "Phone": "+43676123121",
      "Website": "www.clouddna.at",
      "Birthdate": "20.04.1984"
    },
    {
      "Customerid": "1000002",
      "Firstname": "Margit",
      "Lastname": "Mustermann",
      "Title": "Dr.",
      "Gender": "female",
      "Email": "Anna.mustermann@clouddna.at",
      "Phone": "+43676123122",
      "Website": "www.clouddna.at",
      "Birthdate": "20.04.1990"
    },
    {
      "Customerid": "1000003",
      "Firstname": "Felix",
      "Lastname": "Mustermann",
      "Title": "",
      "Gender": "male",
      "Email": "felix.mustermann@clouddna.at",
      "Phone": "+43676123123",
      "Website": "www.clouddna.at",
      "Birthdate": "20.04.2004"
    }
  ]
}

```

1. Die JSON-Datei mit Kundendaten beleben

```
{
  "customers": [
    {
      "Customerid": "1000001",
      "Firstname": "Max",
      "Lastname": "Mustermann",
      "Title": "Dr.",
      "Gender": "male",
      "Email": "max.mustermann@clouddna.at",
      "Phone": "+43676123121",
      "Website": "www.clouddna.at",
      "Birthdate": "20.04.1984"
    },
    {
      "Customerid": "1000002",
      "Firstname": "Margit",
      "Lastname": "Mustermann",
      "Title": "Dr.",
      "Gender": "female",
      "Email": "Anna.mustermann@clouddna.at",
      "Phone": "+43676123122",
      "Website": "www.clouddna.at",
      "Birthdate": "20.04.1990"
    },
    {
      "Customerid": "1000003",
      "Firstname": "Felix",
      "Lastname": "Mustermann",
      "Title": "",
      "Gender": "male",
      "Email": "felix.mustermann@clouddna.at",
      "Phone": "+43676123123",
      "Website": "www.clouddna.at",
      "Birthdate": "20.04.2004"
    }
  ]
}
```

```
        "Customerid": "1000002",
        "Firstname": "Margit",
        "Lastname": "Mustermann",
        "Title": "Dr.",
        "Gender": "female",
        "Email": "Anna.mustermann@clouddna.at",
        "Phone": "+43676123122",
        "Website": "www.clouddna.at",
        "Birthdate": "20.04.1990"
    },
    {
        "Customerid": "1000003",
        "Firstname": "Felix",
        "Lastname": "Mustermann",
        "Title": "",
        "Gender": "male",
        "Email": "felix.mustermann@clouddna.at",
        "Phone": "+43676123123",
        "Website": "www.clouddna.at",
        "Birthdate": "20.04.2004"
    }
]
```

webapp > data > {} genders.json > ...

You, 2 months ago | 1 author (You)

```
1  {
2      "genders": [
3          {
4              "key": "1",
5              "text": "female"
6          },
7          {
8              "key": "2",
9              "text": "male"
10         }
11     ]
12 }
```

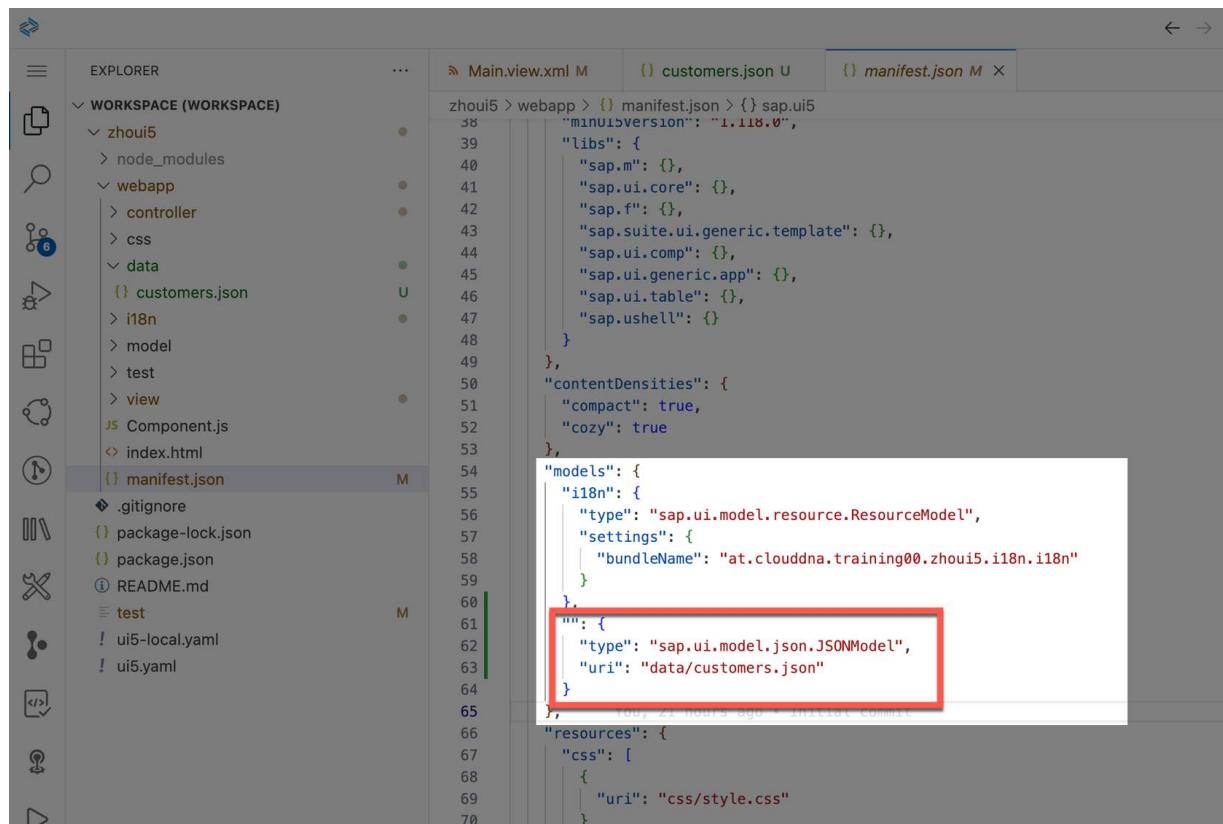
You, 2 months ago • ui5 bis baseControl

Die gleichen Schritte auch nochmal für ein "genders.json" File machen mit folgenden Daten:

```
{
    "genders": [
        {
            "key": "1",
            "text": "female"
        },
        {
            "key": "2",
            "text": "male"
        }
    ]
}
```

JSONModel anlegen

Die erstellten JSON-Files müssen als JSONModels eingerichtet werden. Für die Initialisierung des JSONModels stehen uns mehrere Wege zur Verfügung. Wir registrieren den JSONModel direkt im manifest.json, damit die Initialisierung vom Framework übernommen wird und dieser Model global in der ganzen Applikation zur Verfügung steht.



```
zhoi5 > webapp > manifest.json > sap.ui5
      "minui5version": "1.118.0",
      "libs": {
        "sap.m": {},
        "sap.ui.core": {},
        "sap.f": {},
        "sap.suite.ui.generic.template": {},
        "sap.ui.comp": {},
        "sap.ui.generic.app": {},
        "sap.ui.table": {},
        "sap.ushell": {}
      },
      "contentDensities": {
        "compact": true,
        "cozy": true
      },
      "models": {
        "i18n": {
          "type": "sap.ui.model.resource.ResourceModel",
          "settings": {
            "bundleName": "at.cloudna.training00.zhoi5.i18n.i18n"
          }
        },
        "": {
          "type": "sap.ui.model.json.JSONModel",
          "uri": "data/customers.json"
        }
      },
      "resources": {
        "css": [
          {
            "uri": "css/style.css"
          }
        ]
      }
    },
```

1. Datei manifest.json öffnen

2. Zum Attribut models folgende zwei JSONModels hinzufügen

```
"genderModel": {  
    "type": "sap.ui.model.json.JSONModel",  
    "uri": "data/genders.json"  
}
```

Darunter auch noch das "genderModel" einfügen

```
"models": {  
    "i18n": {  
        "type": "sap.ui.model.resource.ResourceModel",  
        "settings": {  
            "bundleName": "at.clouddna.training00.zhoui5.i18n.i18n"  
        }  
    },  
   "": {  
        "type": "sap.ui.model.json.JSONModel",  
        "uri": "data/customers.json"  
    },  
    "genderModel": {  
        "type": "sap.ui.model.json.JSONModel",  
        "uri": "data/genders.json"  
    }  
},
```

-  Ein Model ohne Namen, sprich ein Attribut in models mit Key "", wird als Default-Model verstanden. Es darf immer nur einen Default-Model geben. Alle anderen Models müssen

zwischen den Anführungszeichen einen Namen übergeben bekommen. Somit werden diese Models zu sogenannten Named Models.

4.2 DataBinding

MK

Martin Koch

Es gibt verschiedene Wege, um Daten gegen ein Control/eine View zu binden: **Aggregation Binding**, **Element Binding**, **Expression Binding**. Data-Binding wird in der XML-View immer mit {} signalisiert.

1. Aggregation Binding:

Aggregation Binding in SAPUI5 ermöglicht die dynamische Verknüpfung von UI-Elementen mit einer Liste von Daten. Dies betrifft beispielsweise Listen, Tabellen oder Container von UI-Elementen. Durch Aggregation Binding können mehrere Elemente gleichzeitig an ein UI-Element gebunden werden, wodurch komplexe Strukturen wie Listen von Elementen effizient umgesetzt werden können. Änderungen in der zugrunde liegenden Datenquelle führen zu automatischen Aktualisierungen in der UI-Ansicht.

2. Element Binding:

Element Binding ermöglicht die direkte Verknüpfung eines UI-Elements mit einem einzelnen Datenobjekt. Im Gegensatz zum Aggregation Binding, bei dem mehrere Elemente an ein UI-Element gebunden werden, wird beim Element Binding nur ein einzelnes Datenobjekt mit einem UI-Element verknüpft. Änderungen an den Daten führen zu Aktualisierungen im UI-Element.

3. Expression Binding:

Expression Binding in SAPUI5 ermöglicht die Auswertung von Ausdrücken direkt im Binding-Prozess. Dies erlaubt komplexe Berechnungen oder Formatierungen, bevor der Wert an das UI-Element gebunden wird. Expression Binding kann beispielsweise für die

Anzeige von formatierten Daten oder für die Berechnung von Abhängigkeiten zwischen verschiedenen Datenfeldern genutzt werden.

4. Property Binding:

Property Binding ist die grundlegende Form des Bindings in SAPUI5 und verknüpft die Eigenschaft (Property) eines UI-Elements mit einem Attribut eines Datenobjekts. Durch Property Binding können einfache Werte wie Zeichenketten oder Zahlen dynamisch mit Daten aus einer Datenquelle verbunden werden. Änderungen in den Daten führen zu automatischen Aktualisierungen in der Benutzeroberfläche.

Aufgabe zu Property Binding

Als value-Property der Input-Controls den dazugehörigen Attributnamen aus dem JSON-File nehmen und in diesem Format eintragen: {/Attributname}. Mit dem „/“ wird direkt per Property Binding auf das Attribut im Default-Model zugegriffen. Falls auf ein Named-Model zugegriffen werden möchte, sieht der Zugriff so aus: {modelName> /Attributname}.

```

<Table items="/customers" id="main_table" headerText="{i18n>main.table.title}>
  <columns>
    <Column id="idMainColumn">
      <Text id="main_text_customerid" text="{i18n>app.customerid}"/>
    </Column>
    <Column id="main_column_firstname">
      <Text id="main_text_firstname" text="{i18n>app.firstname}"/>
    </Column>
    <Column id="main_column_lastname">
      <Text id="main_text_lastname" text="{i18n>app.lastname}"/>
    </Column>
    <Column id="main_column_title" minScreenWidth="Desktop">
      <Text id="main_text_title" text="{i18n>app.title}"/>
    </Column>
    <Column id="main_column_gender" minScreenWidth="Desktop">
      <Text id="main_text_gender" text="{i18n>app.gender}"/>
    </Column>
    <Column id="main_column_birthdate" minScreenWidth="Desktop">
      <Text id="main_text_birthdate" text="{i18n>app.birthdate}"/>
    </Column>
    <Column id="main_column_email" minScreenWidth="Desktop">
      <Text id="main_text_email" text="{i18n>app.email}"/>
    </Column>
    <Column id="main_column_phone" minScreenWidth="Desktop">
      <Text id="main_text_phone" text="{i18n>app.phone}"/>
    </Column>
    <Column id="main_column_website" minScreenWidth="Desktop">
      <Text id="main_text_website" text="{i18n>app.website}"/>
    </Column>
  </columns>
  <items>
    <ColumnListItem>
      <cells>
        <ObjectIdentifier title="{Customerid}"/>
        <Text text="{Firstname}"/>
        <Text text="{Lastname}"/>
        <Text text="{Title}"/>
        <Text text="{Gender}"/>
        <Text text="{Birthdate}"/>
        <Text text="{Email}"/>
        <Text text="{Phone}"/>
        <Link text="{Website}" href="https://<{Website}>" target="_blank"/>
      </cells>
    </ColumnListItem>
  </items>
</Table>

```

1. Die customer-list bei den Items in der Tabelle einfügen

2. value-Property bei allen Input-Controls anpassen.

```
<Table items="/customers" id="main_table" headerText="{i18n>main.table.ti
    <columns>
        <Column id="idMainColumn">
            <Text id="main_text_customerid" text="{i18n>app.cus
        </Column>
        <Column id="main_column_firstname">
            <Text id="main_text_firstname" text="{i18n>app.firs
        </Column>
        <Column id="main_column_lastname">
            <Text id="main_text_lastname" text="{i18n>app.last
        </Column>
        <Column id="main_column_title" minWidth="Desktop"
            <Text id="main_text_title" text="{i18n>app.title}"
        </Column>
        <Column id="main_column_gender" minWidth="Desktop"
            <Text id="main_text_gender" text="{i18n>app.gender
        </Column>
        <Column id="main_column_birthdate" minWidth="Deskt
            <Text id="main_text_birthdate" text="{i18n>app.bir
        </Column>
        <Column id="main_column_email" minWidth="Desktop"
            <Text id="main_text_email" text="{i18n>app.email}
        </Column>
        <Column id="main_column_phone" minWidth="Desktop"
            <Text id="main_text_phone" text="{i18n>app.phone}
        </Column>
        <Column id="main_column_website" minWidth="Deskt
            <Text id="main_text_website" text="{i18n>app.websit
        </Column>
    </columns>
    <items>
        <ColumnListItem>
            <cells>
                <ObjectIdentifier title="{Customerid}"/>
                <Text text="{Firstname}"/>
                <Text text="{Lastname}"/>
                <Text text="{Title}"/>
                <Text text="{Birthdate}"/>
                <Text text="{Birthdate}"/>
                <Text text="{Email}"/>
            </cells>
        </ColumnListItem>
    </items>
</Table>
```

```
        <Text text="{Phone}" />
        <Link text="{Website}" href="https://'{Website}'"
      </cells>
    </ColumnListItem>
  </items>
</Table>
```

i18n im Detail

Das i18n-File ist für die automatische Übersetzung von Texten zuständig. Die Texte werden aus einem Named-Resource-Model ausgelesen. Falls eine andere Sprache als die Default-Sprache implementiert werden möchte, muss das i18n-File kopiert und mit der Namenskonvention „i18n_<xx>.properties“ eingefügt werden. Statt <xx> wird der jeweilige Sprachcode eingefügt, z.B.: de für Deutsch, hu für Ungarisch, ...

Sie wissen, dass auf Namend-Models wie folgt zugegriffen werden kann: {modelName}>pfad} und jetzt können wir auch nachvollziehen, warum mit {i18n>app.title} ein entsprechender Wert zurückgeliefert wird.



Achtung: Bei einem Resource-Named-Model muss beim Pfad kein "/" vorangestellt werden!

Unicodes

Es kann unter Umständen sein, dass manche Frontend-Server und/oder UTF-Einstellungen bei der Anzeige nicht mit Umlauten umgehen können. Bei Umlauten und sprachspezifischen Buchstaben, Zeichen, etc. kann der Unicode statt dem Character angegeben werden. Es muss

also \uXXXX geschrieben werden. So schreibt man statt „Kundenübersicht“ Kunden\u00FCbersicht“ ([siehe eine Liste der Unicodes auf Wikipedia](#)).

Meistverwendete Unicodes:

- \u00FC – ü
- \u00DC – Ü
- \u00E4 – ä
- \u00C4 – Ä
- \u00F6 – ö
- \u00D6 – Ö

4.3 Formatter im Controller

MK Martin Koch

Wenn Sie komplexe Berechnungen oder Abfragen zur Laufzeit durchführen möchten, empfehlen wir, Expression Binding zu vermeiden. In solchen Fällen können Formatter eine geeignete Alternative darstellen.

Unsere aktuelle Herausforderung besteht darin, anstelle von Expression Binding die korrekten sprachspezifischen Texte bei dem Gender-Text-Element zu laden.

```
<items>
    <ColumnListItem type="Navigation" press="onListItemClicked">
        <cells>
            <Text text="{Firstname}" />
            <Text text="{Lastname}" />
            <Text text="{Title}" />
            <Text text="{path:'Gender', formatter: '.genderFormatter'}"/>
            <Text text="{path: 'Birthdate', formatter: '.dateFormatter'}"/>
            <Text text="{Email}" />
            <Text text="{Phone}" />
            <Link text="{Website}" href="https://'{Website}" target="_blank"/>
        </cells>
    </ColumnListItem>
</items>
```

1. Formatter in der View hinzufügen

```

<items>
    <ColumnListItem>
        <cells>
            <ObjectIdentifier title="{Customerid}" />
            <Text text="{Firstname}" />
            <Text text="{Lastname}" />
            <Text text="{Title}" />
            <Text text="{path:'Gender', formatter: '.genderFormatter'}" />
            <Text text="{path: 'Birthdate', formatter: '.dateFormatter'}" />
            <Text text="{Email}" />
            <Text text="{Phone}" />
            <Link text="{Website}" href="https://'{Website}" target="_blank" />
        </cells>
    </ColumnListItem>
</items>

```

```

genderFormatter: function (sKey) {
    let oView = this.getView();
    let oI18nModel = oView.getModel("i18n");
    let oResourceBundle = oI18nModel.getResourceBundle();
    let sText = oResourceBundle.getText(sKey);
    return sText;
},
dateFormatter: function(date) {
    let dateObj = new Date(date);
    return dateObj.getDate() + "." + (dateObj.getMonth() + 1) + "." + dateObj.getFullYear();
}

```

1. Formatter-Funktion im Controller Main.controller.js implementieren

```

genderFormatter: function (sKey) {
    let oView = this.getView();
    let oI18nModel = oView.getModel("i18n");
    let oResourceBundle = oI18nModel.getResourceBundle();
    let sText = oResourceBundle.getText(sKey);
}

```

```

        return sText;
    }

dateFormatter: function(date) {
    let dateObj = new Date(date);
    return dateObj.getDate() + "." + (dateObj.getMonth() + 1) + "." + dateObj.getFullYear();
}

```

Kunden

Kundentabelle								
Kundennummer	Vorname	Nachname	Titel	Geschlecht	Geburtsdatum	E-Mail	Telefon	Website
1000001	Max	Mustermann	Dr.	männlich	20.04.1984	max.mustermann@clouddna.at	+43676123121	www.clouddna.at
1000002	Margit	Mustermann	Dr.	weiblich	20.04.1990	anna.mustermann@clouddna.at	+43676123122	www.clouddna.at
1000003	Felix	Mustermann		männlich	20.04.2004	felix.mustermann@clouddna.at	+43676123123	www.clouddna.at

Applikation inkl. Data-Binding und einem Formatter

4.4 Zusammenfassung

MK

Martin Koch

Nach Abschluss dieser Übung stammen die Kundendaten nicht mehr, wie im vorherigen Beispiel, aus statisch festgelegten Werten, sondern werden aus dem Default- und Named-Model bezogen, welche auf JSON-Dateien zugreifen. Um sicherzustellen, dass diese Daten direkt in der Ansicht angezeigt werden können, wurden verschiedene Data-Binding-Techniken angewendet. Zudem wurde ein Formatter eingesetzt, um sicherzustellen, dass die korrekten, sprachspezifischen Texte bei dem Geschlecht-Textfeld erscheinen.

Kunden

Kundentabelle

Kundennummer	Vorname	Nachname	Titel	Geschlecht	Geburtsdatum	E-Mail	Telefon	Website
1000001	Max	Mustermann	Dr.	männlich	20.04.1984	max.mustermann@clouddna.at	+43676123121	www.clouddna.at
1000002	Margit	Mustermann	Dr.	weiblich	20.04.1990	anna.mustermann@clouddna.at	+43676123122	www.clouddna.at
1000003	Felix	Mustermann		männlich	20.04.2004	felix.mustermann@clouddna.at	+43676123123	www.clouddna.at

5 Einleitung

MK

Martin Koch

In dieser Übung werden wir eine zweite Seite erstellen, die als Detailansicht für Kunden fungiert. Im ersten Schritt legen wir eine einfache Seite an und richten das Routing ein, um einen Aufruf zu ermöglichen. Anschließend präsentieren wir die Kundendaten für den Index 0. Um mehr Funktionalität zu bieten, ersetzen wir dann die herkömmliche Seite durch eine ObjectPage. Im vierten Schritt machen wir die Tabelleneinträge auf der Übersichtsseite anklickbar und verweisen auf die Detailseite des Kunden mit Index 0. Abschließend zeigen wir auf der Detailansicht die Informationen des ausgewählten Kunden an und implementieren eine Rückwärtsnavigation, um von der Detailansicht zur Übersichtsseite zurückzukehren.

5.1 Navigation erstellen

 Martin Koch

Nun wird die Navigation erstellt, zu der über die Route "/customer" hingavigiert werden kann.
In dieser werden später die Kundeninformationen angezeigt bzw. bearbeitet werden können.

Navigation hinzufügen

Nun werden wir die Navigation zur eben erstellen View in metadata.json erstellen.

Customer.view.xml U Customer.controller.js U manifest.json M X

```
webapp > {} manifest.json > {} sap.ui5 > {} routing > {} targets > {} TargetCustomer > viewName
67     "css": [
68         {
69             "uri": "css/style.css"
70         }
71     ],
72 },
73 "routing": {
74     "config": {
75         "routerClass": "sap.m.routing.Router",
76         "viewType": "XML",
77         "async": true,
78         "viewPath": "at.clouddna.training00.zhoui5.view",
79         "controlAggregation": "pages",
80         "controlId": "app",
81         "clearControlAggregation": false
82     },
83     "routes": [
84         {
85             "name": "RouteMain",
86             "pattern": ":?query:",
87             "target": [
88                 "TargetMain"
89             ]
90         },
91         {
92             "name": "RouteCustomer",
93             "pattern": "/customer",
94             "target": [
95                 "TargetCustomer"
96             ]
97         }
98     ],
99     "targets": {
100         "TargetMain": {
101             "viewType": "XML",
102             "transition": "slide",
103             "clearControlAggregation": false,
104             "viewId": "Main",
105             "viewName": "Main"
106         },
107         "TargetCustomer": {
108             "viewType": "XML",
109             "transition": "slide",
110             "clearControlAggregation": false,
111             "viewId": "Customer",
112             "viewName": "Customer"
113         }
114     },
115     "rootView": {
116         "viewName": "at.clouddna.training00.zhoui5.view.App",
117         "type": "XML",
118         "async": true,
119         "id": "App"
120     }
121 },
122 }
123
124
```

1

2

You, a few seconds ago • Uncommitted changes

1. Route hinzufügen

2. Target hinzufügen

```
{  
  "name": "RouteCustomer",  
  "pattern": "/customer",  
  "target": [  
    "TargetCustomer"  
  ]  
}
```

```
"TargetCustomer": {  
  "viewType": "XML",  
  "transition": "slide",  
  "clearControlAggregation": false,  
  "viewId": "Customer",  
  "viewName": "Customer"  
},
```

Kunden

Kunden

Kundennummer:	1000001
Vorname:	Max
Nachname:	Mustermann
Titel:	Dr.
Geschlecht:	weiblich
E-Mail:	max.mustermann@clouddna.at
Tel.nr.:	+43676123123
Homepage:	https://clouddna.at

Wird nun der URL der Pfad "/customer" angehängt, wird zur der vorherig erstellten Kundenseite verlinkt. Da noch keine UI-Elemente hinzugefügt wurden, handelt es sich dabei um die statische Ansicht der Kundeninformationen.

5.2 ElementBinding ohne Parameter

MK

Martin Koch

Nun werden wir in der erstellten Detailansicht den ersten Kunden anzeigen. Dabei wird über die Route "/customer/0" der Kunde mit dem Index 0 sichtbar sein.

```
{  
  "name": "RouteCustomer",  
  "pattern": "customer/{path}",  
  "target": [  
    "TargetCustomer"  
  ]  
}
```

Routeparameter path bei der **RouteCustomer** manifest.json hinzufügen

```
{  
  "name": "RouteCustomer",  
  "pattern": "customer/{path}",  
  "target": [  
    "TargetCustomer"  
  ]  
}
```

```
]
}
```

The screenshot shows the SAP Studio interface with the file Customer.controller.js open. The code defines a controller that extends the sap.ui.core.mvc.Controller. It includes an `onInit` function to attach a pattern match listener and an `_onPatternMatched` function to bind the view to the customer at index 0. A red box highlights the `_onPatternMatched` function.

```
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ],
4 /**
5  * @param {typeof sap.ui.core.mvc.Controller} Controller
6  */
7 function (Controller) {
8   "use strict";
9
10  return Controller.extend("at.cloudbees.training00_zhou5_controller.Customer", {
11    /**
12     * @param {sap.ui.core.mvc.Controller} Controller
13     */
14    _onInit: function () {
15      this.getOwnerComponent().getRouter().getRoute("RouteCustomer").attachPatternMatched(this._onPatternMatched, this);
16    },
17
18    _onPatternMatched: function(oEvent) {
19      this.getView().bindElement("/customers/0");
20    },
21
22    genderFormatter: function(sKey){
23      let oView = this.getView();
24      let oI18nModel = oView.getModel("i18n");
25      let oResourceBundle = oI18nModel.getResourceBundle();
26      let sText = oResourceBundle.getText(sKey);
27      return sText;
28    }
29  });
30});
```

Im Controller den Customer mit dem Index 0 an die View binden, nachdem das Routing ausgeführt wurde und den genderFormatter ebenfalls hinzufügen.

```
onInit: function () {
  this.getOwnerComponent().getRouter().getRoute("RouteCustomer").attachPatternMatched(this._onPatternMatched, this);
},
_onPatternMatched: function(oEvent) {
  this.getView().bindElement("/customers/0");
},
genderFormatter: function(sKey) {
  let oView = this.getView();
  let oI18nModel = oView.getModel("i18n");
  let oResourceBundle = oI18nModel.getResourceBundle();
  let sText = oResourceBundle.getText(sKey);
  return sText;
}
```

```
        return sText;  
    }  
  
}
```

Die Kundenseite kann nun über die Route "/customer/o" erreicht werden.



Vor der Route "/customer/0" muss ein "#" eingefügt werden ("#/customer/0")

Kundendetails

Kundennummer:	0000001
Vorname:	Max
Nachname:	Mustermann
Title:	Dr.
Geschlecht:	männlich
E-Mail:	max.mustermann@cluddna.at
Telefon:	+43761234521
Website:	www.cluddna.at
Geburtsdatum:	20.04.1984

Applikation mit SimpleForm

5.3 ObjectPageLayout

MK

Martin Koch

Nun wird in der Kundenansicht die normale Page durch eine ObjectPage ersetzt.

Das [**SAPUI5 ObjectPageLayout**](#) ist ein Layout-Steuerelement, das sich ideal für die Darstellung detaillierter Informationen zu einem einzelnen Datensatz eignet. Es ermöglicht die strukturierte Organisation von Inhalten in Abschnitten, unterstützt Lazy Loading für verbesserte Leistung, enthält einen Headerbereich mit Titel und Steuerelementen, bietet eine Navigationsleiste für den schnellen Zugriff auf verschiedene Abschnitte und unterstützt ein responsives Design. Der optionale Parallax-Effekt im Headerbereich bietet eine visuell ansprechende Scrollfunktionalität.

Customer.view.xml U X {} customers.json {} manifest.json M

```

1  <mvc:View controllerName="at.clouddna.training00.zhoui5.controller.Customer"
2    xmlns="sap.ui.core.mvc" displayBlock="true"
3    xmlns:u="sap.uxap"1
4    xmlns:i="sap.ui.layout.form"
5    xmlns:s="sap.f.semantic"
6    xmlns="sap.m">2
```

```

7      <u:ObjectPageLayout id="ObjectPageLayout"
8        showTitleInHeaderContent="true"
9        upperCaseAnchorBar="false">
10       <u:headerTitle>
11         <u:ObjectPageDynamicHeaderTitle>
12           <u:expandedHeading>
13             <Title text="Kundendetails" />
14           </u:expandedHeading>
15         </u:ObjectPageDynamicHeaderTitle>
16       </u:headerTitle>
17       <u:sections>
18         <u:ObjectPageSection>
19           <u:subSections>
20             <u:ObjectPageSubSection>
21               <u:blocks>
22                 <f:SimpleForm title="{Firstname} {Lastname}">1
23                   <Label text="{i18n>app.customerid}" labelFor="customerid" />
24                   <Text id="customerid" text="{Customerid}" />
25
26                   <Label text="{i18n>app.firstname}" labelFor="firstname" />
27                   <Text id="firstname" text="{Firstname}" />
28
29                   <Label text="{i18n>app.lastname}" labelFor="lastname" />
30                   <Text id="lastname" text="{Lastname}" />
31
32                   <Label text="{i18n>app.title}" labelFor="title" />
33                   <Text id="title" text="{Title}" />
34
35                   <Label text="{i18n>app.gender}" labelFor="gender" />
36                   <Text id="gender" text="{path: 'Gender', formatter: '.genderFormatter'}" />
37
38                   <Label text="{i18n>app.email}" labelFor="email" />
39                   <Text id="email" text="{Email}" />
40
41                   <Label text="{i18n>app.phone}" labelFor="phone" />
42                   <Text id="phone" text="{Phone}" />
43
44                   <Label text="{i18n>app.website}" labelFor="website" />
45                   <Text id="website" text="{Website}" />
46
47                   <Label text="{i18n>app.birthdate}" labelFor="birthdate" />
48                   <Text id="birthdate" text="{Birthdate}" />
49               </f:SimpleForm>1
50             </u:blocks>
51           </u:ObjectPageSubSection>
52         </u:subSections>
53       </u:ObjectPageSection>
54     </u:sections>
55   </u:ObjectPageLayout>
56 </mvc:View>
```

1. Namensraum für ObjectPageLayout hinzufügen

2. ObjectPageLayout erstellen und mit SimpleForm befüllen

```
xmlns:u="sap.uxap"
```

```
<u:ObjectPageLayout id="ObjectPageLayout"
    showTitleInHeaderContent="true"
    upperCaseAnchorBar="false">
    <u:headerTitle>
        <u:ObjectPageDynamicHeaderTitle>
            <u:expandedHeading>
                <Title text="Kundendetails" />
            </u:expandedHeading>
        </u:ObjectPageDynamicHeaderTitle>
    </u:headerTitle>
    <u:sections>
        <u:ObjectPageSection>
            <u:subSections>
                <u:ObjectPageSubSection>
                    <u:blocks>
                        <f:SimpleForm title="{Firstname} {Lastname}">
                            <Label text="{i18n>app.customerid}" labelFor="customerid">
                                <Text id="customerid" text="{Customerid}" />
                            <Label text="{i18n>app.firstname}" labelFor="firstname">
                                <Text id="firstname" text="{Firstname}" />
                            <Label text="{i18n>app.lastname}" labelFor="lastname">
                                <Text id="lastname" text="{Lastname}" />
                            <Label text="{i18n>app.title}" labelFor="title">
                                <Text id="title" text="{Title}" />
                            <Label text="{i18n>app.gender}" labelFor="gender">
                                <Text id="gender" text="{path: 'Gender', format: 'radio'}" />
                            <Label text="{i18n>app.email}" labelFor="email">
                                <Text id="email" text="{Email}" />
                            <Label text="{i18n>app.phone}" labelFor="phone">
                                <Text id="phone" text="{Phone}" />
                            <Label text="{i18n>app.website}" labelFor="website">
                                <Text id="website" text="{Website}" />
                            <Label text="{i18n>app.birthdate}" labelFor="birthdate">
                                <Text id="birthdate" text="{Birthdate}" />
                        </f:SimpleForm>
                    </u:blocks>
                </u:ObjectPageSubSection>
            </u:subSections>
        </u:ObjectPageSection>
    </u:sections>
</u:ObjectPageLayout>
```

```
</u:ObjectPageSubSection>
</u:subSections>
</u:ObjectPageSection>
</u:sections>
</u:ObjectPageLayout>
```

Kundendetails

Max Mustermann

Kundennummer:	1000001
Vorname:	Max
Nachname:	Mustermann
Title:	Dr.
Geschlecht:	männlich
E-Mail:	max.mustermann@clouddna.at
Telefon:	+497123121
Website:	www.clouddna.at
Geburtsdatum:	20.04.1984

Detailansicht mit ObjectPageLayout

5.4 Navigation erweitern

 Martin Koch

Wir erweitern nun die Navigation, um von der Übersichtsseite zur Kundenansicht zu wechseln.
Dabei wird im ersten Schritt jedoch immer der Kunde mit dem Index 0 angezeigt.

Customer.view.xml U Main.view.xml M Main.controller.js M Customer.controller.js U {}

webapp > view > Main.view.xml

You, a few seconds ago | 1 author (You)

```

1  <mvc:View controllerName="at.cloudna.training00.zhoui5.controller.Main"
2      xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3      xmlns:s="sap.f.semantic"
4      xmlns="sap.m">
5      <s:SemanticPage id="main_page">
6          <s:titleHeading>
7              <Title text="{i18n>title}" />
8          </s:titleHeading>
9          <s:content>
10             <Table items="/customers" id="main_table" headerText="{i18n>main.table.title}">
11                 <columns>
12                     <Column id="idMainColumn">
13                         <Text id="main_text_customerid" text="{i18n>app.customerid}" />
14                     </Column>
15                     <Column id="main_column_firstname">
16                         <Text id="main_text_firstname" text="{i18n>app.firstname}" />
17                     </Column>
18                     <Column id="main_column_lastname">
19                         <Text id="main_text_lastname" text="{i18n>app.lastname}" />
20                     </Column>
21                     <Column id="main_column_title" minWidth="Desktop">
22                         <Text id="main_text_title" text="{i18n>app.title}" />
23                     </Column>
24                     <Column id="main_column_gender" minWidth="Desktop">
25                         <Text id="main_text_gender" text="{i18n>app.gender}" />
26                     </Column>
27                     <Column id="main_column_birthdate" minWidth="Desktop">
28                         <Text id="main_text_birthdate" text="{i18n>app.birthdate}" />
29                     </Column>
30                     <Column id="main_column_email" minWidth="Desktop">
31                         <Text id="main_text_email" text="{i18n>app.email}" />
32                     </Column>
33                     <Column id="main_column_phone" minWidth="Desktop">
34                         <Text id="main_text_phone" text="{i18n>app.phone}" />
35                     </Column>
36                     <Column id="main_column_website" minWidth="Desktop">
37                         <Text id="main_text_website" text="{i18n>app.website}" />
38                     </Column>
39                 </columns>
40                 <items>
41                     <ColumnListItem type="Navigation" press="onListItemClicked">
42                         <cells>
43                             <ObjectIdentifier title="{Customerid}" />
44                             <Text text="{Firstname}" />
45                             <Text text="{Lastname}" />
46                             <Text text="{Title}" />
47                             <Text text="{path:'Gender', formatter: '.genderFormatter'}" />
48                             <Text text="{Birthdate}" />
49                             <Text text="{Email}" />
50                             <Text text="{Phone}" />
51                             <Link text="{Website}" href="https://{{Website}}" target="_blank" />
52                         </cells>
53                     </ColumnListItem>
54                 </items>
55             </Table>
56         </s:content>
57     </s:SemanticPage>
58 </mvc:View>
```

You, a few seconds ago | 1 author (You)

Type Navigation und onPressEvent im ColumnListItem hinzufügen

```
<ColumnListItem type="Navigation" press="onListItemClicked">
```

Customer.view.xml U Main.view.xml M Main.controller.js M Customer.controller.js U customers.json

webapp > controller > **JS** Main.controller.js > ...

You, a few seconds ago | 1 author (You)

```
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3 ],
4   /**
5    * @param {typeof sap.ui.core.mvc.Controller} Controller
6    */
7   function (Controller) {
8     "use strict";
9
10    return Controller.extend("at.cloudna.training00.zhoui5.controller.Main", {
11      /**
12       * Initialization code
13       */
14      /**
15       * @param {sap.ui.core.Control} oView View
16       */
17      /**
18       * @param {sap.ui.core.ValueFormatter} oFormatter Formatter
19       */
20      /**
21       * @param {string} sKey Key
22       */
23      /**
24       * @param {sap.ui.core.Item} oItem Item
25       */
26    });
27  });
28
29  /**
30   * @param {sap.ui.core.Item} oItem Item
31   */
32  onListItemClicked: function(oEvent) {
33    this.getOwnerComponent().getRouter().navTo("RouteCustomer", {path: oEvent.getParameter("list").getSelectedItem().getBindingContext().getPath()});
34  }
35});
```

Funktion **onListItemClicked** hinzufügen, die festlegt, was passieren soll, wenn eine Tabellenzeile angeklickt wird. Der Path (=Index des angeklickten Elements, in diesem Fall immer 0) wird als Parameter mitgeschickt

```

onListItemClicked: function(oEvent) {
    this.getOwnerComponent().getRouter().navTo("RouteCustomer", {path: 0}),
}

```

Screenshot of the SAP UI5 code editor showing the Customer.controller.js file. A red box highlights the `_onPatternMatched` method.

```

Customer.view.xml U Main.view.xml M JS Main.controller.js M JS Customer.controller.js U X customers.json
webapp > controller > JS Customer.controller.js > sap.ui.define() callback > _onPatternMatched
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ],
4 /**
5  * @param {typeof sap.ui.core.mvc.Controller} Controller
6  */
7 function (Controller) {
8   "use strict";
9
10  return Controller.extend("at.cloudna.training00.zhoui5.controller.Customer", {
11    onInit: function () {
12      this.getOwnerComponent().getRouter().getRoute("RouteCustomer").attachPatternMatched(this._onPatternMatched, this);
13    },
14
15    _onPatternMatched: function(oEvent) {
16      let path = oEvent.getParameters().arguments["path"];
17      this.getView().bindElement("/customers/" + path);
18    },
19
20    genderFormatter: function(sKey){
21      let oView = this.getView();
22      let oI18nModel = oView.getModel("i18n");
23      let oResourceBundle = oI18nModel.getResourceBundle();
24      let sText = oResourceBundle.getText(sKey);
25      return sText;
26    }
27  });
28

```

Die Methode `_onPatternMatched` so ändern, dass das Element mit dem mitgeschickten Index gebunden wird.

```

_onPatternMatched: function(oEvent) {
  let path = oEvent.getParameters().arguments["path"];
  this.getView().bindElement("/customers/" + path);
},

```

Kunden

Kundentabelle										
Kundennummer	Vorname	Nachname	Titel	Geschlecht	Geburtsdatum	E-Mail	Telefon	Website	Q:	
1000001	Max	Mustermann	Dr.	männlich	20.04.1984	max.mustermann@clouddna.at	+43676123121	www.clouddna.at		
1000002	Margit	Mustermann	Dr.	weiblich	20.04.1990	anna.mustermann@clouddna.at	+43676123122	www.clouddna.at		
1000003	Felix	Mustermann		männlich	20.04.2004	felix.mustermann@clouddna.at	+43676123123	www.clouddna.at		

Nun sind die einzelnen Tabelleneinträge anklickbar geworden (erkennbar am Pfeil ganz rechts)

...

Kundendetails

Max Mustermann
Kundennummer: 1000001 Vorname: Max Nachname: Mustermann Titel: Dr. Geschlecht: männlich E-Mail: max.mustermann@clouddna.at Telefon: +43676123121 Website: www.clouddna.at Geburtsdatum: 20.04.1984

... und verlinken nach wie vor auf die Detailansicht des Kunden mit dem Index 0.

5.5 ElementBinding mit Parameter

MK

Martin Koch

Wir passen nun unsere Übersichtsseite an, sodass für jeden Kunden in der Liste automatisch die entsprechende Detailansicht geöffnet wird.

```
onListItemClicked: function(oEvent) {
    const sPath = oEvent.getSource().getBindingContext().getPath();

    this.getRouter().navTo("RouteCustomer", {
        path: encodeURIComponent(sPath)
    }, false);
},
```

Dazu passen wir im **Main.controller** die **onListItemClicked**-Methode so an, dass aus dem übergebenen **oEvent** der Index des geklickten Elements herausgelesen und dieser dann als Pfad übergeben wird.

```
onListItemClicked: function(oEvent) {
    const sPath = oEvent.getSource().getBindingContext().getPath();

    this.getRouter().navTo("RouteCustomer", {
        path: encodeURIComponent(sPath)
```

```
}, false);  
},
```

Nun können wir auch die Detailansicht des Kunden mit dem Index x öffnen:

Kundendetails

Margit Mustermann

Kundennummer:	1000002
Vorname:	Margit
Nachname:	Mustermann
Titel:	Dr.
Geschlecht:	weiblich
E-Mail:	Anna.mustermann@clouddna.at
Telefon:	+43676123122
Website:	www.clouddna.at
Geburtsdatum:	20.04.1990

Detailansicht des Kunden mit dem Index 1

5.6 Zusammenfassung

MK

Martin Koch

Wir haben nun eine Applikation mit zwei verschiedenen Seiten, die bereits ein beidseitiges Routing ermöglicht. Die Daten aus der Übersichtsseite werden erfolgreich in die Detailansicht geladen und korrekt angezeigt. Die nächste Herausforderung besteht darin, die Kundendaten auch editierbar zu gestalten.

6 Einleitung

 Martin Koch

Es soll über einen Button im Header steuerbar sein, ob sich die Ansicht im Anzeige- oder Bearbeitungsmodus befindet. Sowohl der Anzeige- als auch der Bearbeitungsmodus sollen jeweils in einem eigenen Fragment dargestellt werden. Zudem soll ein weiterer Button in die Fußzeile der Seite eingefügt werden, um das Abbrechen zu simulieren. Alle Buttons sollen entsprechend den Anzeigeeinstellungen ein- und ausgeblendet werden.

Verwendete Controls: -

Verwendete Technologien: -

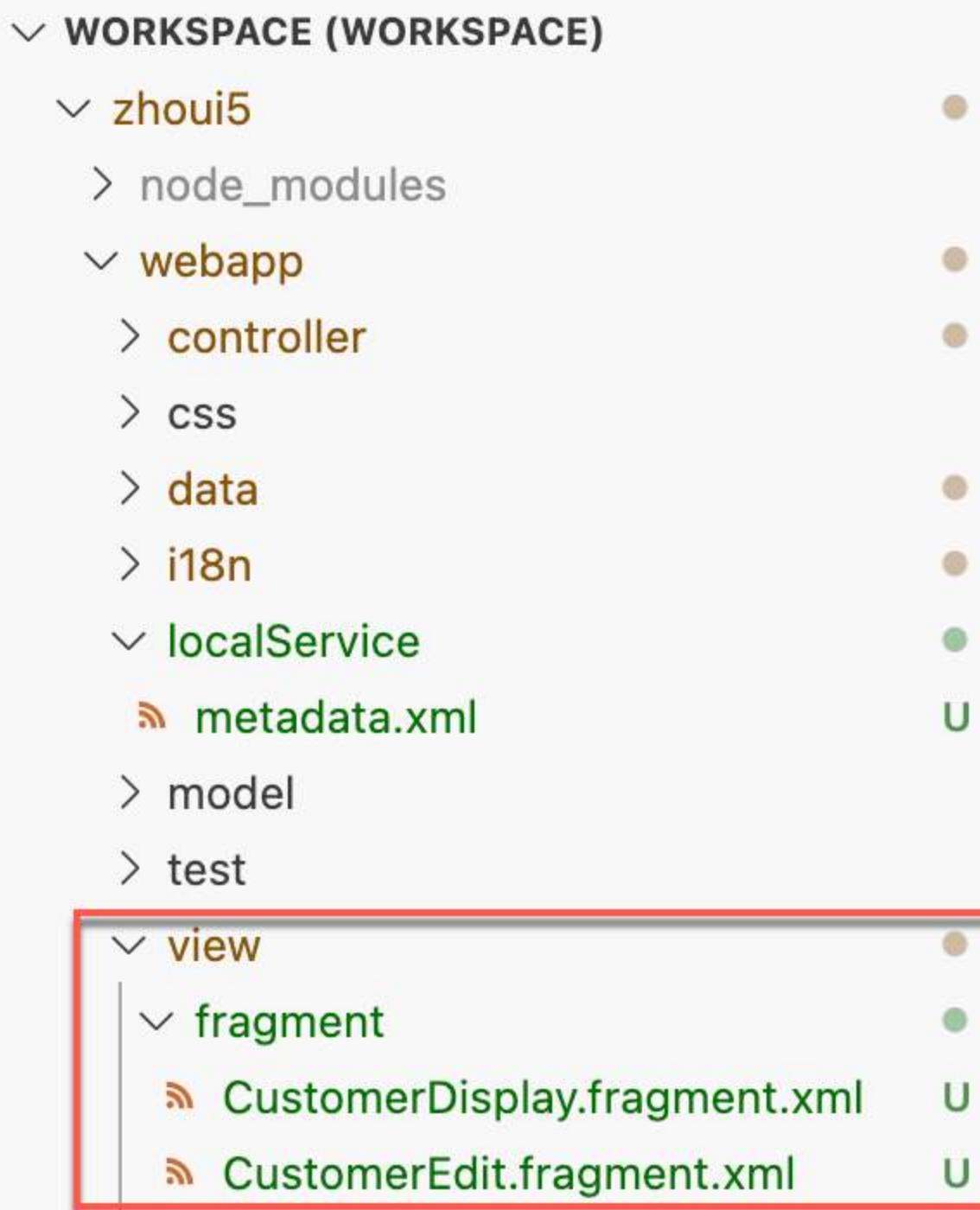
6.1 Display&Edit-Fragment anlegen

MK

Martin Koch

Zuerst muss ein Fragment angelegt werden, damit die SimpleForm ausgelagert werden kann.

SAPUI5 Fragmente sind wiederverwendbare und modulare UI-Komponenten, die dazu dienen, UI-Elemente und -Logik in einem isolierten Container zu gruppieren. Fragmente können in verschiedenen Ansichten oder Anwendungen wiederverwendet werden und bieten eine flexible Möglichkeit, UI-Komponenten zu strukturieren und zu organisieren. Sie können sowohl für die Definition von Dialogen als auch für Teile von Ansichten verwendet werden, um die Wartung und Entwicklung von Benutzeroberflächen zu erleichtern.



1. Im Ordner **view** einen Ordner **fragment** erstellen

2. Die files **CustomerDisplay.fragment.xml** und **CustomerEdit.fragment.xml** im neuen Ordner erstellen

webapp > view > fragment > CustomerDisplay.fragment.xml

You, 7 days ago | 1 author (You)

```
1  <core:FragmentDefinition
2      xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3      xmlns:core="sap.ui.core"
4      xmlns:u="sap.uxap"
5      xmlns:f="sap.ui.layout.form"
6      xmlns:s="sap.f.semantic"
7      xmlns="sap.m">
8  >
46
47  </core:FragmentDefinition>
```

You, 2 months ago • ui5 bis baseController

1. FragmentDefinition einfügen

```
<core:FragmentDefinition
    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
    xmlns:core="sap.ui.core"
    xmlns:u="sap.uxap"
    xmlns:f="sap.ui.layout.form"
    xmlns:s="sap.f.semantic"
    xmlns="sap.m">

</core:FragmentDefinition>
```

```

zhoui5 > webapp > view > fragment > CustomerDisplay.fragment.xml
1  <core:FragmentDefinition
2    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3    xmlns:core="sap.ui.core"
4    xmlns:u="sap.uxap"
5    xmlns:f="sap.ui.layout.form"
6    xmlns:ss="sap.f.semantic"
7    xmlns="sap.m">
8    <u:ObjectPageSection>
9      <u:subSections>
10        <u:ObjectPageSubSection>
11          <u:blocks>
12            <f:SimpleForm title="{Firstname} {Lastname}">
13              <Label text="{i18n>app.customerid}" labelFor="customerid" />
14              <Text id="customerid" text="{Customerid}" />
15
16              <Label text="{i18n>app.firstname}" labelFor="firstname" />
17              <Text id="firstname" text="{Firstname}" />
18
19              <Label text="{i18n>app.lastname}" labelFor="lastname" />
20              <Text id="lastname" text="{Lastname}" />
21
22              <Label text="{i18n>app.title}" labelFor="title" />
23              <Text id="title" text="{Title}" />
24
25              <Label text="{i18n>app.gender}" labelFor="gender" />
26              <Text id="gender" text="{${Gender} === '1' ? ${i18n>female} : ${i18n>male} }" />
27
28              <Label text="{i18n>app.email}" labelFor="email" />
29              <Text id="email" text="{Email}" />
30
31              <Label text="{i18n>app.phone}" labelFor="phone" />
32              <Text id="phone" text="{Phone}" />
33
34              <Label text="{i18n>app.website}" labelFor="website" />
35              <Text id="website" text="{Website}" />
36
37              <Label text="{i18n>app.birthdate}" labelFor="birthdate" />
38              <Text id="birthdate" text="{path: 'Birthdate', formatter: '.dateFormatter'}" />
39
40            </f:SimpleForm>
41          </u:blocks>
42        </u:ObjectPageSubSection>
43      </u:subSections>
44    </u:ObjectPageSection>
45  </core:FragmentDefinition>

```

Die **ObjectPageSubsection** aus der **CustomerView** in die Fragmente kopieren.

```

<u:blocks>
  <f:SimpleForm title="{Firstname} {Lastname}" editable="true">
    <Label text="{i18n>app.firstname}" labelFor="firstname" />
    <Input id="firstname" value="{Firstname}" />

    <Label text="{i18n>app.lastname}" labelFor="lastname" />
    <Input id="lastname" value="{Lastname}" />

```

Bei der Form gehört auch noch das Attribut ***editable*** auf ***true*** gesetzt. Dies bewirkt, dass sich die Zeilen automatisch auf die Höhe des Inputs anpassen, wenn ein solcher auftreten sollte

```
<core:FragmentDefinition
    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
    xmlns:core="sap.ui.core"
    xmlns:u="sap.uxap"
    xmlns:f="sap.ui.layout.form"
    xmlns:s="sap.f.semantic"
    xmlns="sap.m">
    <u:ObjectPageSection>
        <u:subSections>
            <u:ObjectPageSubSection>
                <u:blocks>
                    <f:SimpleForm title="{Firstname} {Lastname}" editable='true'
                        <Label text="{i18n>app.customerid}" labelFor="customerid">
                            <Text id="customerid" text="{Customerid}" />
                        </Label>
                    <Label text="{i18n>app.firstname}" labelFor="firstname">
                        <Text id="firstname" text="{Firstname}" />
                    </Label>
                    <Label text="{i18n>app.lastname}" labelFor="lastname">
                        <Text id="lastname" text="{Lastname}" />
                    </Label>
                    <Label text="{i18n>app.title}" labelFor="title" />
                    <Text id="title" text="{Title}" />
                    <Label text="{i18n>app.gender}" labelFor="gender" />
                    <Text id="gender" text="{= ${Gender} === '1' ? ${i18n>app.male} : ${i18n>app.female}}"/>
                    <Label text="{i18n>app.email}" labelFor="email" />
                    <Text id="email" text="{Email}" />
                    <Label text="{i18n>app.phone}" labelFor="phone" />
                    <Text id="phone" text="{Phone}" />
                    <Label text="{i18n>app.website}" labelFor="website" />
                    <Text id="website" text="{Website}" />
                    <Label text="{i18n>app.birthdate}" labelFor="birthdate" />
                    <Text id="birthdate" text="{path: 'Birthdate', format: 'DD/MM/YYYY'}" />
                </f:SimpleForm>
```

```

        </u:blocks>
    </u:ObjectPageSubSection>
</u:subSections>
</u:ObjectPageSection>
</core:FragmentDefinition>

```

zhoui5 > webapp > view > fragment > CustomerEdit.fragment.xml

```

1  <core:FragmentDefinition
2      xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3      xmlns:core="sap.ui.core"
4      xmlns:u="sap.uxap"
5      xmlns:f="sap.ui.layout.form"
6      xmlns:s="sap.f.semantic"
7      xmlns="sap.m">
8      <u:ObjectPageSection>
9          <u:subSections>
10         <u:ObjectPageSubSection>
11             <u:blocks>
12                 <f:SimpleForm title="{Firstname} {Lastname}">
13                     <Label text="{i18n>app.customerid}" labelFor="customerid" />
14                     <Input id="customerid" value="{Customerid}" />1
15
16                     <Label text="{i18n>app.firstname}" labelFor="firstname" />
17                     <Input id="firstname" value="{Firstname}" />1
18
19                     <Label text="{i18n>app.lastname}" labelFor="lastname" />
20                     <Input id="lastname" value="{Lastname}" />1
21
22                     <Label text="{i18n>app.title}" labelFor="title" />
23                     <Input id="title" value="{Title}" />1
24
25                     <Label text="{i18n>app.gender}" labelFor="gender" />
26                     <Select id="gender" selectedKey="{Gender}" items="{genderModel>/genders}">
27                         <items>
28                             <core:Item key="{genderModel>key}" text="{path: 'genderModel>text', formatter: '.genderFormatter'}" />
29                         </items>
30                     </Select>2
31
32                     <Label text="{i18n>app.email}" labelFor="email" />
33                     <Input id="email" type="Email" value="{Email}" />1
34
35                     <Label text="{i18n>app.phone}" labelFor="phone" />
36                     <Input id="phone" type="Tel" value="{Phone}" />1
37
38                     <Label text="{i18n>app.website}" labelFor="website" />
39                     <Input id="website" value="{Website}" />1
40
41                     <Label text="{i18n>app.birthdate}" labelFor="birthdate" />
42                     <Input id="birthdate" type="Date" value="{Birthdate}" />1
43                 </f:SimpleForm>
44             </u:blocks>
45         </u:ObjectPageSubSection>
46     </u:subSections>
47 </u:ObjectPageSection>
48 </core:FragmentDefinition>

```

Im **CustomerEdit.fragment.xml**:

1. Die **Textfelder** durch **Inputs** austauschen

2. Das **Gender-Textfeld** durch einen **Select** austauschen

```

<core:FragmentDefinition
    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
    xmlns:core="sap.ui.core"
    xmlns:u="sap.uxap"
    xmlns:f="sap.ui.layout.form"
    xmlns:s="sap.f.semantic"
    xmlns="sap.m">
    <u:ObjectPageSection>
        <u:subSections>
            <u:ObjectPageSubSection>
                <u:blocks>
                    <f:SimpleForm title="{Firstname} {Lastname}">
                        <Label text="{i18n>app.customerid}" labelFor="customerid" />
                        <Input id="customerid" value="{Customerid}" />

                        <Label text="{i18n>app.firstname}" labelFor="firstname" />
                        <Input id="firstname" value="{Firstname}" />

                        <Label text="{i18n>app.lastname}" labelFor="lastname" />
                        <Input id="lastname" value="{Lastname}" />

                        <Label text="{i18n>app.title}" labelFor="title" />
                        <Input id="title" value="{Title}" />

                        <Label text="{i18n>app.gender}" labelFor="gender" />
                        <Select id="gender" selectedKey="{Gender}" items="">
                            <items>
                                <core:Item key="{genderModel}>key" text="{label}" />
                            </items>
                        </Select>

                        <Label text="{i18n>app.email}" labelFor="email" />
                        <Input id="email" type="Email" value="{Email}" />

                        <Label text="{i18n>app.phone}" labelFor="phone" />
                        <Input id="phone" type="Tel" value="{Phone}" />

                        <Label text="{i18n>app.website}" labelFor="website" />
                        <Input id="website" value="{Website}" />

                        <Label text="{i18n>app.birthdate}" labelFor="birthdate" />
                        <Input id="birthdate" type="Date" value="{Birthdate}" />
                    </f:SimpleForm>
                </u:blocks>
            </u:ObjectPageSubSection>
        </u:subSections>
    </u:ObjectPageSection>
</core:FragmentDefinition>

```

```
</u:blocks>
</u:ObjectPageSubSection>
</u:subSections>
</u:ObjectPageSection>
</core:FragmentDefinition>
```

6.2 Fragmente im Controller definieren

MK Martin Koch

Im Customer-Controller müssen müssen zuerst JSONModel und Fragment inkludiert werden. Hinzufügen eines neuen JSONModels mit einer Property editmode, die false gesetzt wird. Setzen des neu erstellten Models mit this.getView().setModel(oEditModel, "editModel"). Dieses Model kann nun in der Customer-View mit dem Namen "editModel" angesprochen werden.

```
sap.ui.define([
    "sap/ui/model/json/JSONModel",
    "sap/ui/core/Fragment",
    "sap/m/MessageBox",
    ],
    /**
     * @param {typeof sap.ui.core.mvc.Controller} Controller
     */
    function (BaseController, JSONModel, Fragment, ) {
        "use strict";

        return BaseController.extend("at.cloodna.training00.zhoui5.controller.Customer", {

            onInit: function () {
                const oRouter = this.getOwnerComponent().getRouter();
                const oEditModel = new JSONModel({
                    editmode: false
                });

                this.setContentDensity();

                this.getView().setModel(oEditModel, "editModel");

                oRouter.getRoute("RouteCustomer").attachPatternMatched(this._onPatternMatched, this);
                oRouter.getRoute("CreateCustomer").attachPatternMatched(this._onCreatePatternMatched, this);
            },
        });
    }
);
```

Customer.controller:

- 1. Klassen/Module importieren**
- 2. Imports in einen Parameter speichern**
- 3. In der `onInit`-Methode einen JSONModel definieren und der View als Named-Model setzen**

```
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/m/MessageBox",
    "sap/ui/model/json/JSONModel",
    "sap/ui/core/Fragment"
],
/***
 * @param {typeof sap.ui.core.mvc.Controller} Controller
 */
function (Controller, MessageBox, JSONModel, Fragment) {
/* ... */
```

```
onInit: function () {
    const oRouter = this.getOwnerComponent().getRouter();
    const oEditModel = new JSONModel({
        editmode: false
    });

    this.setContentDensity();

    this.getView().setModel(oEditModel, "editModel");

    oRouter.getRoute("RouteCustomer").attachPatternMatched(this._onPatternMatched, this);
}
```

```
    oRouter.getRoute("CreateCustomer").attachPatternMatched(this._onCreate)
},
```

Anlegen eines leeren Objekts zur Speicherung der Fragments.

```
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/m/MessageBox",
4   "sap/ui/model/json/JSONModel",
5   "sap/ui/core/Fragment"
6 ],
7 /**
8  * @param {typeof sap.ui.core.mvc.Controller} Controller
9 */
10 function (Controller, MessageBox, JSONModel, Fragment) {
11   "use strict";
12
13   return Controller.extend("at.clouddna.training00.zhoui5.controller.", {
14     _fragmentList: {}, 1
15     OnInit: function () {
16       let oEditModel = new JSONModel({
17         editmode: false
18       });
19
20       this.getView().setModel(oEditModel, "editModel");
21     },
22   },
23 },
24
```

1. Eine Variable `_fragmentList` vorbereiten, welche die Fragmente hält

```
_fragmentList: {},
```

Um zwischen Edit- und Display-Fragment wechseln zu können, muss das entsprechende Fragment geladen werden.

Die Page wird zwischengespeichert, um leichter darauf zugreifen zu können.

Per `removeAllContent()`-Methode wird der gesamte Inhalt aus der content-Aggregation der Page gelöscht.

Wenn das anzuzeigende Fragment bereits geladen wurde, wird es aus dem Objekt ausgelesen und in die content-Aggregation der Page eingefügt.

Falls das Fragment noch nicht existiert, wird es über `Fragment.load()`-Methode per Promise gelesen. Diese Funktion bekommt die ID des Fragments, den Namen und der App-Controller wird dem Fragment als Controller zugeordnet. Wenn das Fragment geladen wurde, wird es in das Array hinzugefügt und angezeigt.

```
_showCustomerFragment: function(sFragmentName) {
    let page = this.getView().byId("ObjectPageLayout");
    page.removeAllSections();
    if(this._fragmentList[sFragmentName]) {
        page.addSection(this._fragmentList[sFragmentName]);
    } else {
        Fragment.load({
            id: this.getView().createId(sFragmentName),
            name: "at.clouddna.training00.zhoui5.view.fragment." + sFragmentName,
            controller: this
        }).then(function(oFragment) {
            this._fragmentList[sFragmentName] = oFragment;
            page.addSection(this._fragmentList[sFragmentName]);
            oFragment.bind(this);
        });
    }
},
```

The code is annotated with four purple circles numbered 1 through 4, corresponding to the following steps:

1. A call to `Fragment.load()` to load the fragment.
2. A call to `page.removeAllSections()` to clear the page's sections.
3. An `if` statement checking if the fragment exists in the list; if true, it adds the fragment to the page's sections.
4. The final closing brace of the `_showCustomerFragment` method definition.

1. Eine Methode `_showCustomerFragment` für das Fragment-Loading vorbereiten, welche als Parameter den Namen des Fragments entgegennimmt
2. Instanz zur Page in eine Variable speichern und den Inhalt der Page leeren
3. In der **if-Bedingung** prüfen, ob das Fragment schon existiert und **wenn ja**, dann als **Inhalt** der Page **setzen**

4. Wenn nein, muss das Fragment initialisiert, unserer Variable `_fragmentList` hinzugefügt und anschließend der Page als Inhalt gesetzt werden

```
_showCustomerFragment: function(sFragmentName) {
    let page = this.getView().byId("ObjectPageLayout");

    page.removeAllSections();

    if(this._fragmentList[sFragmentName]) {
        page.addSection(this._fragmentList[sFragmentName]);
    } else {
        Fragment.load({
            id: this.getView().createId(sFragmentName),
            name: "at.clouddna.training00.zhoui5.view.fragment
            controller: this
        }).then(function(oFragment) {
            this._fragmentList[sFragmentName] = oFragment;
            page.addSection(this._fragmentList[sFragmentName]);
        }.bind(this));
    }
},
```

Die `_toggleEdit`-Funktion implementieren. Diese toggled die editmode-Property des EditModels und zeigt das richtige Fragment an, in dem sie wiederum die Funktion `_showCustomerFragment` aufruft.

```
_toggleEdit: function(bEditMode){
    let oEditModel = this.getView().getModel("editModel");
    oEditModel.setProperty("/editmode", bEditMode);
    this._showCustomerFragment(bEditMode ? "CustomerEdit" : "CustomerDisplay");
},
```

1

1. Die Methode `_toggleEdit` nimmt einen Boolean entgegen und ändert je nach dem den Wert des Named-Models und fordert das Setzen des Fragments an

```
onEditPressed: function(){
    this._toggleEdit(true);
},
```

2

```
onSavePressed: function(){
    let oModel = this.getView().getModel();
    let oData = oModel.getData();
    MessageBox.success(JSON.stringify(oData));
```

```
    this._toggleEdit(false);
},
```

3

```
onCancelPressed: function(){
    this._toggleEdit(false);
},
```

4

1. Der Event-Handler `onEditPressed` ruft die Methode `_toggleEdit` auf
2. die Methode `onSavePressed` muss erweitert werden um dem Methodenaufruf `_toggleEdit`
3. Der Event-Handler `onCancelPressed` muss hinzugefügt werden. Diese Methode ruft ebenfalls `_toggleEdit` auf

```
onEditPressed: function() {
    this._toggleEdit(true);
}

_toggleEdit: function(bEditMode) {
    let oEditMode = this.getView().getModel("editMode");
```

```

14     oEditModel.setProperty("/editmode", bEditMode);
15
16     this._showCustomerFragment(bEditMode ? "CustomerEdit" : "CustomerDisplay");
17 },
18
19 onSavePressed: function(){
20     let oModel = this.getView().getModel();
21     let oData = oModel.getData();
22     MessageBox.success(JSON.stringify(oData));
23
24     this._toggleEdit(false);
25 },
26
27 onCancelPressed: function(){
28     this._toggleEdit(false);
29 },
30
31

```

```

14
15     _fragmentList: {},
16
17     _onInit: function () {
18         let oEditModel = new JSONModel({
19             editmode: false
20         });
21
22         this.getView().setModel(oEditModel, "editModel");
23         this._showCustomerFragment("DisplayCustomer"); ①
24     },
25
26
27     _showCustomerFragment: function(sFragmentName){
28         let oPage = this.getView().byId("page");
29
30         oPage.removeAllContent();
31

```

1. Abschließend soll in der **onInit**-Methode das Fragment für die **Anzeige initial** geladen werden

```
this._showCustomerFragment("CustomerDisplay");
```

Damit unsere Editierfunktion auch Funktioniert, brauchen wir dafür nur noch ein paar Buttons. Damit der Bearbeiten- und der Speicher- und Abbrechen-Button nicht zur gleichen Zeit angezeigt werden, muss in der Customer.view.xml die visible-Property bei den Buttons gesetzt werden. Hier wird per Expression-Binding auf das editModel, das in der onInit-Funktion erstellt wurde, zugegriffen.

```
<u:ObjectPageLayout id="ObjectPageLayout"
    showTitleInHeaderContent="true"
    showFooter="!= ${editModel}/editmode}"
    upperCaseAnchorBar="false">
    <u:headerTitle>
        <u:ObjectPageDynamicHeaderTitle>
            <u:expandedHeading>
                <HBox>
                    <Title text="{i18n>customerDetails}" />
                    <ToolbarSpacer />
                </HBox>
            </u:expandedHeading>
            <u:actions>
                <Button id="app_edit_button" text="{i18n>edit}" type="Emphasized" icon="sap-icon://edit"
                    press="onEditPressed" visible="!= ${editModel}/editmode}" />
            </u:actions>
        </u:ObjectPageDynamicHeaderTitle>
    </u:headerTitle>
    <u:footer>
        <Toolbar id="app_bar_footer">
            <ToolbarSpacer />
            <Button id="app_save_button" text="{i18n>save}" type="Accept" icon="sap-icon://save"
                press="onSavePressed" visible="!= ${editModel}/editmode}" />
            <Button id="app_cancel_button" text="{i18n>cancel}" type="Reject" icon="sap-icon://cancel"
                press="onCancelPressed" visible="!= ${editModel}/editmode}" />
        </Toolbar>
    </u:footer>
</u:ObjectPageLayout>
```

1

2

1. **Button für das Bearbeiten** soll nur dann sichtbar sein, wenn wir uns nicht im Editiermodus befinden

2. Die unteren zwei Buttons für das Speichern und Abbrechen sollen nur dann sichtbar sein, wenn wir uns im Editiermodus befinden

```
<u:ObjectPageLayout id="ObjectPageLayout"
    showTitleInHeaderContent="true"
    showFooter="{= ${editModel}/editmode }"
    upperCaseAnchorBar="false">
    <u:headerTitle>
        <u:ObjectPageDynamicHeaderTitle>
            <u:expandedHeading>
                <HBox>
                    <Title text="{i18n}customerDetails" />
                    <ToolbarSpacer />
                </HBox>
            </u:expandedHeading>
            <u:actions>
                <Button id="app_edit_button" text="{i18n}edit" type="Primary"
                    press="onEditPressed" visible="{= !${editModel}/editmode }"/>
            </u:actions>
        </u:ObjectPageDynamicHeaderTitle>
    </u:headerTitle>
    <u:footer>
        <Toolbar id="app_bar_footer">
            <ToolbarSpacer />
            <Button id="app_save_button" text="{i18n}save" type="Accept"
                press="onSavePressed" visible="{= ${editModel}/editmode }"/>
            <Button id="app_cancel_button" text="{i18n}cancel" type="Cancel"
                press="onCancelPressed" visible="{= ${editModel}/editmode }"/>
        </Toolbar>
    </u:footer>
</u:ObjectPageLayout>
```

Display-Mode

Kundendetails

 Bearbeiten

Maximilian Mustermensch

Kundennummer: 00000000-0000-0000-0000-000000000001
Vorname: Maximilian
Nachname: Mustermensch
Titel: Dr.
Geschlecht: männlich
E-Mail: MAXIMILIAN.MUSTERMANN@CLOUDDNA.AT
Telefon: +43 676 123 123 123
Website: www.clouddna.at
Geburtsdatum: 1.1.1970

Applikation im Display-Mode

Edit-Mode

Kundendetails

Maximilian Mustermensch

Vorname:	Maximilian
Nachname:	Mustermensch
Titel:	Dr.
Geschlecht:	männlich
E-Mail:	MAXIMILIAN.MUSTERMANN@CLOUDDNA.AT
Telefon:	+43 676 123 123 123
Website:	www.clouddna.at
Geburtsdatum:	Jan 1, 1970 

 Speichern  Abbrechen

Applikation im Edit-Mode

6.3 Zusammenfassung

MK

Martin Koch

In dieser Übung wurden zwei Fragmente erstellt, von denen eines ausschließlich für die Anzeige von Daten und das andere für die Bearbeitung zuständig ist. Diese Fragmente werden dynamisch zur Laufzeit gewechselt, um eine flexible Darstellung je nach Anforderung zu ermöglichen.

7 Einleitung

 Martin Koch

In dieser Übung werden wir unser JSONModel durch ein ODataModel ersetzen. Dieses hat den Vorteil, dass es direkt mit dem SAP-Backend zusammenarbeitet und man daher direkt die Originaldaten aus dem SAP-System verwenden kann. Dazu müssen wir die Übersichtsseite und die Detailansicht leicht anpassen.

7.1 OData Grundlagen (optional)

MK

Martin Koch

OData-Protokoll

Das **OData-Protokoll** beschreibt die Zugriffe auf REST APIs und definiert auch, wie die Antworten auszusehen haben. REST steht für Representational State Transfer und ist ein Paradigma mit sechs Prinzipien, welches ein einheitliches Konzept für die Kommunikation und für die Schnittstellen bei Webarchitekturen bringen soll. Es basiert grundsätzlich auf **HTTP-Abfragen**. In diesem Kontext sollten Sie immer im Hinterkopf behalten, dass diese Abfragen zustandslos sind. **Zustandslos** (engl. stateless) bedeutet in diesem Fall, dass kein Verbindungsaufbau und -abbau stattfindet. Jede Anfrage-Antwort-Zyklus muss alle notwendigen Informationen enthalten, damit diese Kommunikation von keinen weiteren Informationen abhängig ist, auf die sie ggf. warten muss.

Abfragen, die an eine OData-Schnittstelle gesendet werden, müssen einer bestimmten URI-Syntax (Uniform Resource Identifier) entsprechen. Diese Syntax wird von OData definiert (siehe die offizielle Dokumentation).

Servicedokument

Ein weiterer Teil der URI-Syntax ist die Angabe, auf welche Ressourcen man zugreift. Dafür wurden spezielle Abfragen definiert, die an der Stelle der Ressource im URI-Syntax mitgegeben werden können. Lässt man die Ressource komplett weg, ruft man also nur den Service Root URI auf, muss der OData Service ein sogenanntes Servicedokument zurückgegeben werden. Darin

erhält man als Antwort alle möglichen Ressourcen mit ihren URIs und den dazugehörigen sprechenden Namen.

Service-Metadatadokument

In den meisten Fällen reicht es nicht aus, nur die Namen dieser Ressourcen, die man auch Entitäten nennt, zu kennen. Man muss auch wissen, wie diese aufgebaut sind und welche Attribute und Navigationsattribute beziehungsweise Beziehungen sie besitzen. Für diese Zwecke gibt es das Service-Metadatendokument. Es wird aufgerufen, wenn Sie den Service Root URI um die URI-Option /\$metadata ergänzen.

Entität

Um dieses Dokument lesen zu können – und ganz allgemeinen, um das Konzept von OData zu verstehen –, muss man sich von den Grundsätzen der klassischen Datenbankmodellierung lösen und in Entitäten denken. Die Entitäten können Sie sich wie Objekte vorstellen. Sie besitzen bestimmte Eigenschaften (entweder Attribute oder auch Property genannt) und können Beziehungen untereinander haben (im Deutschen Navigationsattribute oder Assoziationen genannt).

Entitätstyp

Eine Entität wird immer von einem Entitätstyp beschrieben, der dem Service-Metadatendokument entnommen werden kann.

Entitätsmengen

Damit man mehrere Entitäten gebündelt anzeigen kann, werden Entitätsmengen (EntitySets) verwendet. Diese kann man mit Listen beziehungsweise Arrays in der objektorientierten Programmierung vergleichen. Auf jede Abfrage erhält man eine Menge an Entitäten zurück.

Diese Menge kann auch leer sein. Die Entitätsmengen finden Sie ziemlich am Ende des Service-Metadatendokuments im Bereich EntityContainer.

7.2 ODataModel einbinden

MK

Martin Koch

Das JSONModel soll durch ein ODataModel ersetzt werden. Die Daten in der ResponsiveTable sollen aus dem ODataModel bezogen werden.

sap.ui.model.odata.v2.ODataModel

Model, dessen Definition auf dem ODataV2 Protokoll beruht und Kommunikation mit diesem erlaubt.

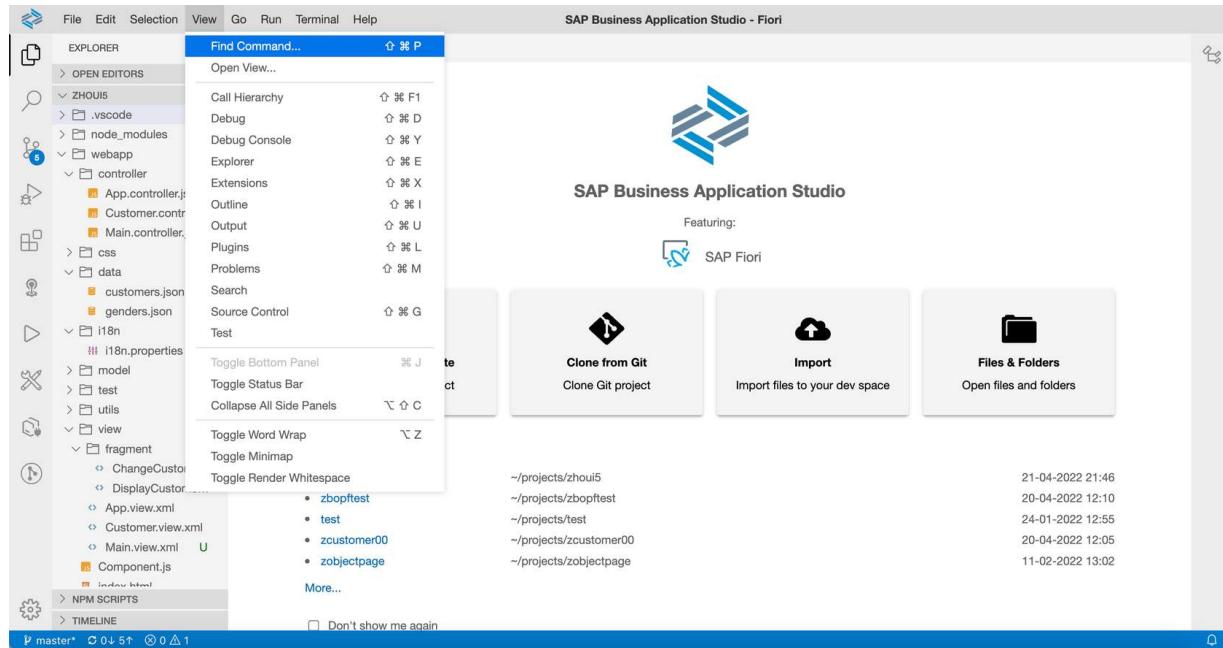
Most used Functions:

- `create` – Legt einen neuen, gefüllten Datensatz an.
- `createEntry` – Legt einen leeren Datensatz an, wird mit `submitChanges` gespeichert.
- `hasPendingChanges` – True, falls Datensätze geändert wurden.
- `submitChanges` – Speichert geänderte Datensätze.
- `remove` – Löscht einen Datensatz.
- `read` – Liest Datensätze.

- update – Ändert Datensätze

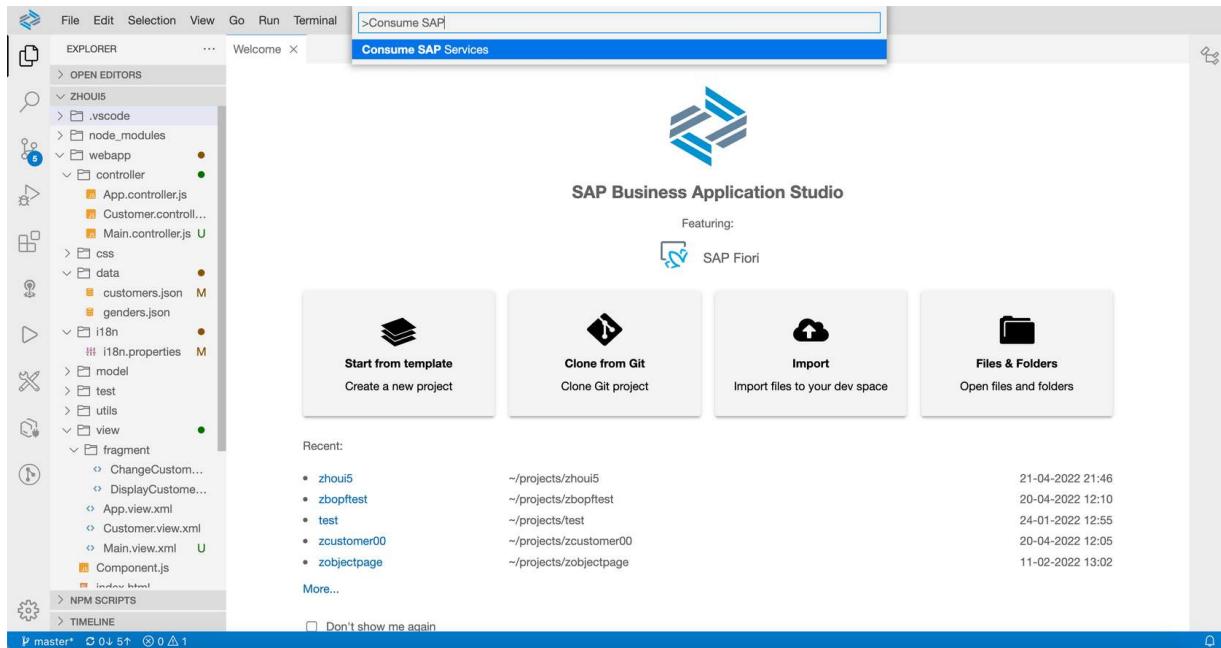
OData-Model anlegen

Ein OData-Model kann entweder aus dem Controller aus angelegt werden und, genauso wie ein JSONModel, der View registriert werden. Nachteil, auch in diesem Fall, dass der Model nur für die View sichtbar ist. Damit ein OData-Model global (in der ganzen App) verfügbar ist, kann dieser im manifest.json eingetragen werden. Ein Command kann für diesen Eintrag Abhilfe verschaffen.

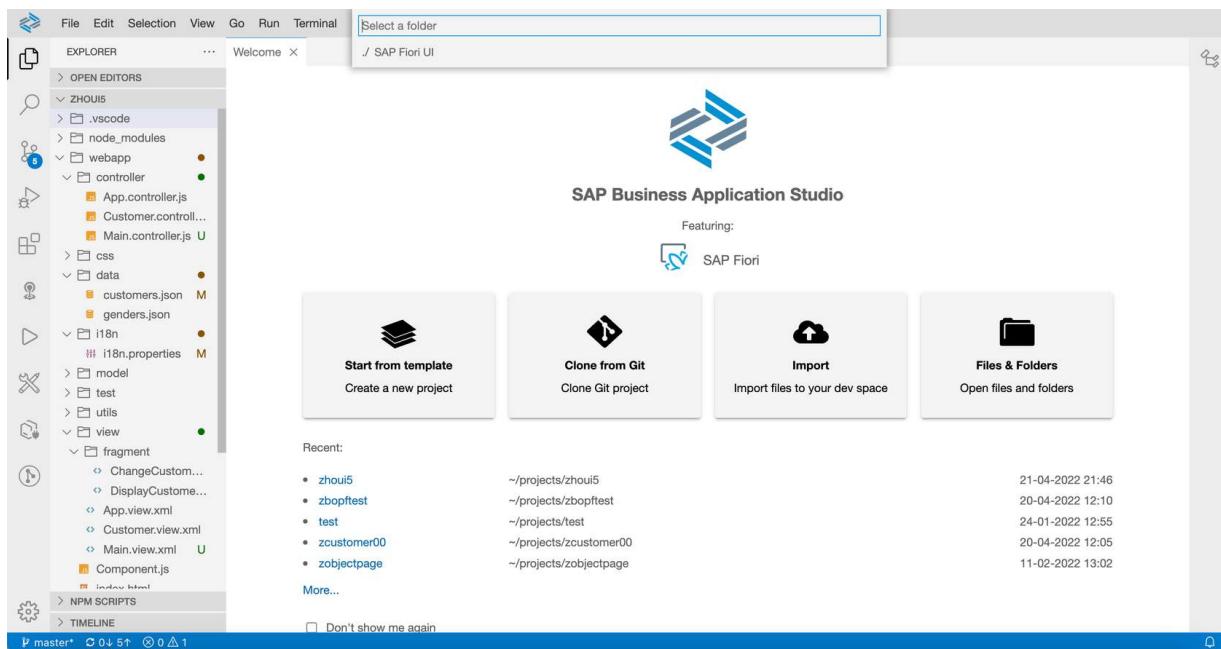


1. Auf View klicken

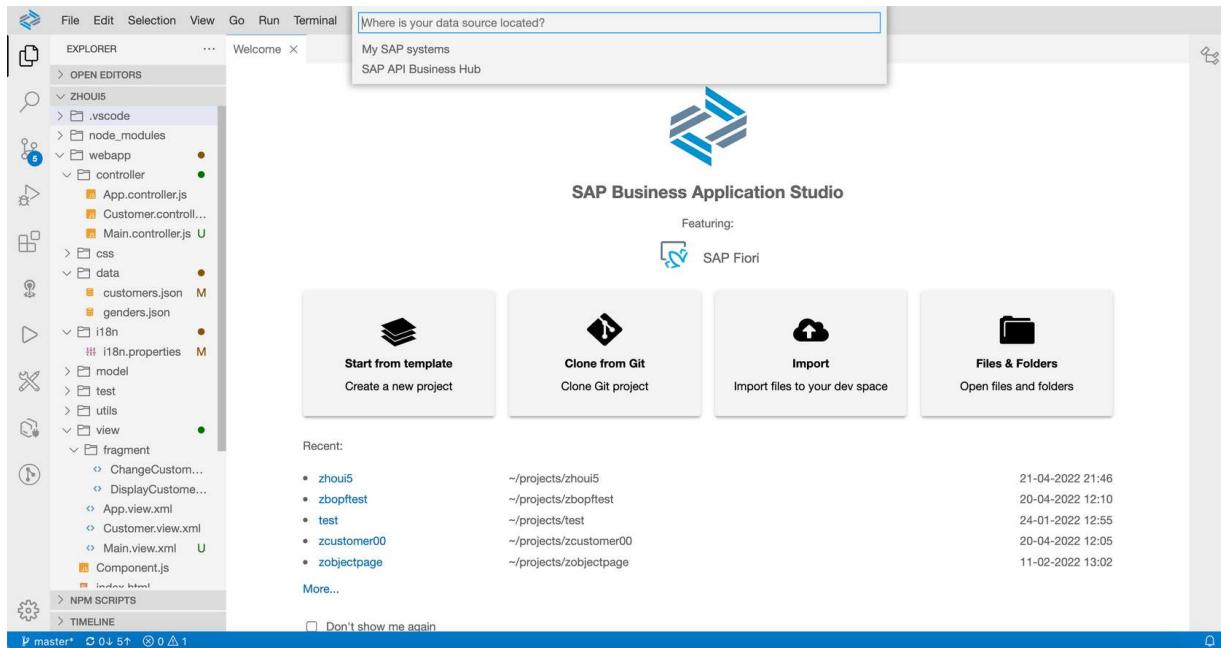
2. Find Command... auswählen



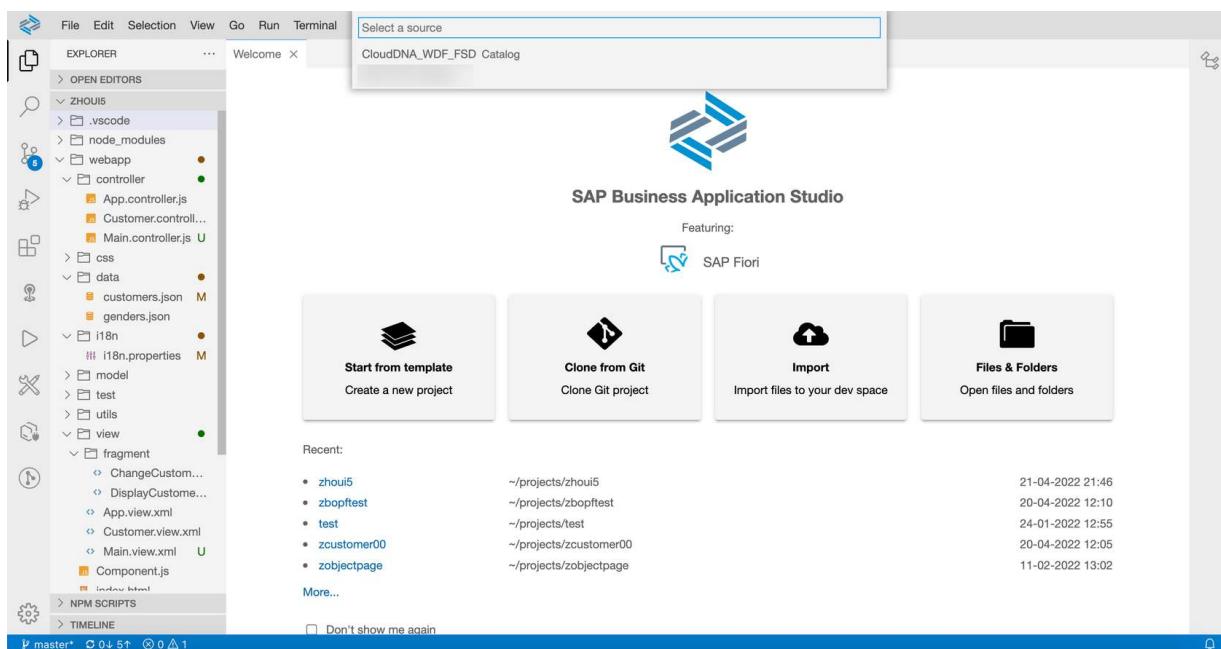
1. Nach Consume SAP Services suchen und auswählen



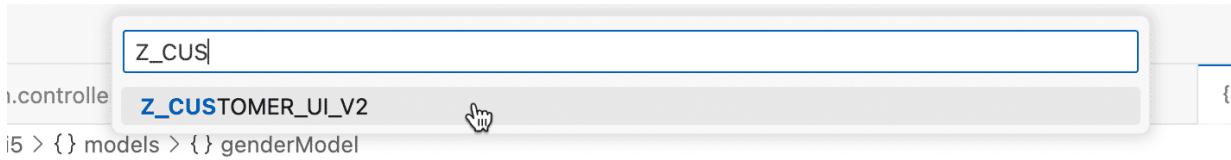
1. Projekt auswählen (in unserem Fall mit dem Pfad ./)



1. My SAP Systems auswählen



1. Destination CloudDNA_WDF_FSD Catalog auswählen



1. OData-Service Z_CUSTOMER_UI_V2 auswählen

A screenshot of the SAP Fiori Modeler interface, specifically the manifest.json editor. The code editor shows the manifest.json configuration. A purple box highlights the 'mainService' entry under the 'dataSources' section. A purple circle with the number '1' is positioned to the right of the highlighted code. The code snippet is as follows:

```
11 "description": "{{appDescription}}",
12 "resources": "resources.json",
13 "sourceTemplate": {
14   "id": "@sap-ux/fiori-freestyle-writer:basic",
15   "version": "0.11.6"
16 },
17 "dataSources": {
18   "mainService": {
19     "uri": "/sap/opu/odata/",
20     "type": "OData",
21     "settings": {
22       "annotations": [],
23       "localUri": "localService/metadata.xml",
24       "odataVersion": "4.0"
25     }
26 }
```

1. Standard Data-Source-Eintrag mit dem Namen **mainService** im **manifest.json** entfernen

```

  "models": {
    "i18n": {
      "type": "sap.ui.model.resource.ResourceModel",
      "settings": {
        "bundleName": "at.clouddna.training00.zhoui5.i18n.i18n"
      }
    },
   "": {
      "dataSource": "Z_CUSTOMER_UI_V2",
      "preload": true,
      "settings": {
        "defaultBindingMode": "TwoWay",
        "defaultCountMode": "Inline",
        "refreshAfterChange": false
      }
    },
  },

```

1. Default-Model prüfen

```

"": {
  "dataSource": "Z_CUSTOMER_UI_V2",
  "preload": true,
  "settings": {
    "defaultBindingMode": "TwoWay",
    "defaultCountMode": "Inline",
    "refreshAfterChange": false
  }
},

```

Metadata.xml inspizieren

Im Ordner localService wurde automatisch das metadata-file des OData-Services hinein kopiert.

Der Service liefert zwei Entities:

- Customer – EntitySet: CustomerSet
 - a. CustomerId – Key
 - b. Firstname
 - c. Lastname
 - d. AcademicTitle
 - e. Email
 - f. Gender
 - g. Phone
 - h. Website
 - i. Documents – Navigation Property auf CustomerDocument
- CustomerDocument – EntitySet: CustomerDocumentsSet
 - a. DocId – Key
 - b. CustomerId – Key
 - c. DokumentName
 - d. DokumentType
 - e. Content

```

metadata.xml X
webapp > localService > Z_CUSTOMER_UI_V2 > metadata.xml
  74   <Property Name="Content" Type="com.binary" sap:label="Binary String" sap:quickinfo="XString"/>
  75   <Property Name="Createdat" Type="Edm.DateTimeOffset" Precision="0" sap:label="Time Stamp" sap:quickinfo="UTC Time Stamp in Short Form (YYYYMMDDhhmmss)"/>
  76   <Property Name="Changedat" Type="Edm.DateTimeOffset" Precision="0" sap:label="Time Stamp" sap:quickinfo="UTC Time Stamp in Short Form (YYYYMMDDhhmmss)"/>
  77   <Property Name="Changedby" Type="Edm.String" MaxLength="12" sap:display-format="UpperCase" sap:label="User Name"/>
  78   <Property Name="to_Customer" Relationship="cds_z_customer_ui.assoc_9FE20F64318DBBDC74B6B643440E2F64" FromRole="ToRole_assoc_9FE20F64318DBBDC74B6B643440E2F64" ToRole="FromRole_assoc_9FE20F64318DBBDC74B6B643440E2F64" />
</EntityType>
<EntityType Name="Z_P_CUSTOMERType" sap:label="CDS Projection - Customer" sap:content-version="1">
  <Key>
    | <PropertyRef Name="Customerid"/>
  </Key>
  <Property Name="Delete_mc" Type="Edm.Boolean" sap:label="Dyn. Method Control" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false"/>
  <Property Name="Update_mc" Type="Edm.Boolean" sap:label="Dyn. Method Control" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false"/>
  <Property Name="to_CustomerDocument_oc" Type="Edm.Boolean" sap:label="Dynamic CBA-Control" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false"/>
  <Property Name="Customerid" Type="Edm.Guid" Nullable="false" sap:label="UUID" sap:quickinfo="16 Byte UUID in 16 Bytes (Raw Format)" sap:createable="false" sap:updatable="false" sap:filterable="false"/>
  <Property Name="Firstname" Type="Edm.String" MaxLength="40" sap:label="Firstname" sap:quickinfo="First Name"/>
  <Property Name="Lastname" Type="Edm.String" MaxLength="40" sap:label="Lastname" sap:quickinfo="Last Name"/>
  <Property Name="Title" Type="Edm.String" MaxLength="20" sap:label="Title" sap:quickinfo="Academic Title: Written Form"/>
  <Property Name="Phone" Type="Edm.String" MaxLength="30" sap:display-format="UpperCase" sap:label="Phone" sap:quickinfo="First Telephone No.: Dialing Code + Number"/>
  <Property Name="Email" Type="Edm.String" MaxLength="122" sap:display-format="UpperCase" sap:label="Email" sap:quickinfo="E-mail address"/>
  <Property Name="Gender" Type="Edm.String" MaxLength="1" sap:display-format="UpperCase" sap:label="Gender" sap:quickinfo="Gender of business partner"/>
  <Property Name="Website" Type="Edm.String" MaxLength="64" sap:label="Website" sap:quickinfo="OO : Website"/>
  <Property Name="Createdby" Type="Edm.String" MaxLength="12" sap:display-format="UpperCase" sap:label="Createdby" sap:quickinfo="User Name" sap:createable="false" sap:updatable="false" sap:filterable="false"/>
  <Property Name="Changedat" Type="Edm.DateTimeOffset" Precision="0" sap:label="Changedat" sap:quickinfo="UTC Time Stamp in Short Form (YYYYMMDDhhmmss)" sap:createable="false" sap:updatable="false" sap:filterable="false"/>
  <Property Name="Changedby" Type="Edm.String" MaxLength="12" sap:display-format="UpperCase" sap:label="Changedby" sap:quickinfo="User Name" sap:createable="false" sap:updatable="false" sap:filterable="false"/>
  <Property Name="Changedat" Type="Edm.DateTimeOffset" Precision="0" sap:label="Changedat" sap:quickinfo="UTC Time Stamp in Short Form (YYYYMMDDhhmmss)" sap:createable="false" sap:updatable="false" sap:filterable="false"/>
  <Property Name="Birthdate" Type="Edm.DateTime" Precision="0" sap:display-format="Date" sap:label="Birthdate" sap:quickinfo="Field of type DATS"/>
  <NavigationProperty Name="to_CustomerDocument" Relationship="cds_z_customer_ui.assoc_9FE20F64318DBBDC74B6B643440E2F64" FromRole="FromRole_assoc_9FE20F64318DBBDC74B6B643440E2F64" />
</EntityType>
<EntityType Name="SAP_Currency" sap:content-version="1">
  <Key>
    | <PropertyRef Name="CurrencyCode"/>
  </Key>
  <Property Name="CurrencyCode" Type="Edm.String" Nullable="false" MaxLength="5" sap:label="Currency" sap:semantics="currency-code"/>
  <Property Name="ISOCode" Type="Edm.String" Nullable="false" MaxLength="3" sap:label="ISO code"/>
  <Property Name="Text" Type="Edm.String" Nullable="false" MaxLength="15" sap:label="Short text"/>
  <Property Name="DecimalPlaces" Type="Edm.Byte" Nullable="false" sap:label="Decimals"/>
</EntityType>
<EntityType Name="SAP_UnitOfMeasure" sap:content-version="1">
  <Key>
    | <PropertyRef Name="UnitCode"/>
  </Key>
</EntityType>

```

metadata.xml

7.3 Main-View anpassen



Martin Koch

Nun werden wir unsere Übersichtsseite so anpassen, dass die Daten aus dem ODataModel verwendet werden.

Main.view.xml M X

JS Main.controller.js M

webapp > view > Main.view.xml

You, a few seconds ago | 1 author (You)

```
1 <mvc:View controllerName="at.clouddna.training00.zhou15.controller.Main" You, 7 days ago * example 3
2   xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3   xmlns:s="sap.f.semantic"
4   xmlns="sap.m">
5   <s:SemanticPage id="main_page">
6     <s:titleHeading>
7       <Title text="{i18n>title}" />
8     </s:titleHeading>
9     <s:content>
10    <Table items="/Z_P_CUSTOMER" id="table" headerText="{i18n>main.table.title}">1
11      <columns>
12        <Column id="idMainColumn">
13          <Text id="main_text_customerid" text="{i18n>app.customerid}" />
14        </Column>
15        <Column id="main_column_firstname">
16          <Text id="main_text_firstname" text="{i18n>app.firstname}" />
17        </Column>
18        <Column id="main_column_lastname">
19          <Text id="main_text_lastname" text="{i18n>app.lastname}" />
20        </Column>
21        <Column id="main_column_title" minWidth="Desktop">
22          <Text id="main_text_title" text="{i18n>app.title}" />
23        </Column>
24        <Column id="main_column_gender" minWidth="Desktop">
25          <Text id="main_text_gender" text="{i18n>app.gender}" />
26        </Column>
27        <Column id="main_column_birthdate" minWidth="Desktop">
28          <Text id="main_text_birthdate" text="{i18n>app.birthdate}" />
29        </Column>
30        <Column id="main_column_email" minWidth="Desktop">
31          <Text id="main_text_email" text="{i18n>app.email}" />
32        </Column>
33        <Column id="main_column_phone" minWidth="Desktop">
34          <Text id="main_text_phone" text="{i18n>app.phone}" />
35        </Column>
36        <Column id="main_column_website" minWidth="Desktop">
37          <Text id="main_text_website" text="{i18n>app.website}" />
38        </Column>
39        <Column id="main_column_search_dialog">
40          <Button press="onOpenDialog" type="Transparent" icon="sap-icon://search"/>
41        </Column>
42      </columns>
43      <items>
44        <ColumnListItem type="Navigation" press="onListItemClicked">
45          <cells>
46            <ObjectIdentifier title="{Customerid}" />
47            <Text text="{Firstname}" />
48            <Text text="{Lastname}" />
49            <Text text=" {Title} />
2
50            <Text text="={ ${Gender} === '1' ? ${i18n>female} : ${i18n>male} }"/>
51            <Text text="={path: 'Birthdate', formatter: '.dateFormatter' }"/>
52            <Text text="={Email}" />
53            <Text text="={Phone}" />
54            <Link text="={Website}" href="https:///{Website}" target="_blank"/>
55          </cells>
56        </ColumnListItem>
57      </items>
58    </Table>
59  </s:content>
60</s:SemanticPage>
61</mvc:View>
```

1. **items**-Property der Table anpassen

2. Geschlecht ans Backend anpassen (1...weiblich, 2...männlich)

```
items="/Z_P_CUSTOMER"
```

```
<Text text="={ ${Gender} === '1' ? ${i18n>female} : ${i18n>male} }"/>
```

Main.view.xml M Main.controller.js M genders.json U X

webapp > data > genders.json > ...

```
1  {
2   "genders": [
3     {
4       "key": "1",
5       "text": "female"
6     },
7     {
8       "key": "2",
9       "text": "male"
10    }
11  ]
12 }
```

Das GendersModel muss nun ebenfalls ans Backend angepasst werden.

```
{  
  "genders": [  
    {  
      "key": "1",  
      "text": "female"  
    },  
    {  
      "key": "2",  
      "text": "male"  
    }  
  ]  
}
```

Main.view.xml M JS Main.controller.js M X

webapp > controller > JS Main.controller.js > sap.ui.define() callback

You, 6 minutes ago | 1 author (You)

```

1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   'sap/ui/core/syncStyleClass',
4   'sap/ui/core/Fragment',
5   "sap/ui/model/Filter",
6   "sap/ui/model/FilterOperator"
7 ],
8 /**
9  * @param {typeof sap.ui.core.mvc.Controller} Controller
10 */
11 function (Controller, syncStyleClass, Fragment, Filter, FilterOperator) {
12   "use strict";
13
14   return Controller.extend("at.clouddna.training00.zhoui5.controller.Main", {
15     ...
16     ...
17     ...
18     ...
19     ...
20     ...
21     ...
22     ...
23     ...
24     ...
25     ...
26     ...
27     ...
28     ...
29     ...
30   });

```

You, 5 days ago

1

1. Da sich der Pfad des BindingContext geändert hat, muss nun die Übergabe an die Detailansicht ebenfalls angepasst werden.

```
let path = oEvent.getSource().getBindingContext().getPath().substring(1);
```

So sieht nun die Übersichtsseite aus, nachdem das SAP-Backend angebunden wurde, optisch hat sich nichts verändert:

Kunden

Kundentabelle

Kundennummer	Vorname	Nachname	Titel	Geschlecht	Geburtsdatum	E-Mail	Telefon	Website	Q
00000000-0000-0000-0000-000000000001	Maximilian	Mustermann	Dr.	männlich	1.1.1970	MAXIMILIAN.MUSTERMANN@CLOUDINA.AT	+43 676 123 123 123	www.cloudina.at	>
00000000-0000-0000-0000-000000000002	Maxima	Mustermann	Mag.	weiblich	1.7.1980	MAXIMA.MUSTERMANN@CLOUDINA.AT	+43 676 124 124 124	www.cloudina.at	>

Übersichtsseite nach Backend-Anbindung

7.4 Detail-View anpassen

MK

Martin Koch

Nun werden wir die Detailansicht so ändern, dass die Daten ebenfalls vom Backend verwendet werden.

```
_onPatternMatched: function(oEvent) {
    const sPath = oEvent.getParameters().arguments.path;

    this.sCustomerPath = decodeURIComponent(sPath);
    this.getView().bindElement(this.sCustomerPath);

    this._showCustomerFragment("CustomerDisplay");
},
```

1. **_onPatternMatched**-Methode im Customer.controller anpassen, da sich der übergebene Pfad geändert hat

```
_onPatternMatched: function(oEvent) {
    const sPath = oEvent.getParameters().arguments.path;

    this.sCustomerPath = decodeURIComponent(sPath);
    this.getView().bindElement(this.sCustomerPath);
```

```

        this._showCustomerFragment("CustomerDisplay");
    },

```

JS Customer.controller.js M CustomerDisplay.fragment.xml U X

webapp > view > fragment > CustomerDisplay.fragment.xml

```

1  <core:FragmentDefinition
2      xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3      xmlns:core="sap.ui.core"
4      xmlns:u="sap.uxap"
5      xmlns:f="sap.ui.layout.form"
6      xmlns:s="sap.f.semantic"
7      xmlns="sap.m">
8      <u:ObjectPageSection>
9          <u:subSections>
10             <u:ObjectPageSubSection>
11                 <u:blocks>
12                     <f:SimpleForm title="{Firstname} {Lastname}">
13                         <Label text="{i18n>app.customerid}" labelFor="customerid" />
14                         <Text id="customerid" text="{Customerid}" />
15
16                         <Label text="{i18n>app.firstname}" labelFor="firstname" />
17                         <Text id="firstname" text="{Firstname}" />
18
19                         <Label text="{i18n>app.lastname}" labelFor="lastname" />
20                         <Text id="lastname" text="{Lastname}" />
21
22                         <Label text="{i18n>app.title}" labelFor="title" />
23                         <Text id="title" text="{Title}" />
24
25                         <Label text="{i18n>app.gender}" labelFor="gender" />
26                         <Text id="gender" text="{${Gender} === '1' ? ${i18n>female} : ${i18n>male}}" />1
27
28                         <Label text="{i18n>app.email}" labelFor="email" />
29                         <Text id="email" text="{Email}" />
30
31                         <Label text="{i18n>app.phone}" labelFor="phone" />
32                         <Text id="phone" text="{Phone}" />
33
34                         <Label text="{i18n>app.website}" labelFor="website" />
35                         <Text id="website" text="{Website}" />
36
37                         <Label text="{i18n>app.birthdate}" labelFor="birthdate" />
38                         <Text id="birthdate" text="{path: 'Birthdate', formatter: '.dateFormatter'}" />
39                     </f:SimpleForm>
40                 </u:blocks>
41             </u:ObjectPageSubSection>
42         </u:subSections>
43     </u:ObjectPageSection>
44 </core:FragmentDefinition>

```

1. Geschlecht gleich wie bei der Übersichtsseite anpassen

```
<Text id="gender" text="${= ${Gender} === '1' ? ${i18n>female} : ${i18n>male}">
```

Augenscheinlich hat sich auch hier nichts verändert:

The screenshot shows a web-based application interface for viewing customer details. At the top, there's a header bar with a back arrow and the text "Kundendetails". On the right side of the header is a blue "Bearbeiten" button. Below the header, the customer's name "Maximilian Mustermann" is displayed in bold. A thin horizontal line separates this from the detailed information below. The detailed information is presented in a table-like format with two columns. The left column contains the field names, and the right column contains the corresponding values. The fields listed are: Kundennummer (Customer Number) with value 0000000-0000-0000-0000-0000000001; Vorname (First Name) with value Maximilian; Nachname (Last Name) with value Mustermann; Titel (Title) with value Dr.; Geschlecht (Gender) with value männlich; E-Mail (Email) with value MAXIMILIAN.MUSTERMANN@CLOUDONA.AT; Telefon (Phone) with value +43 670 123 123 123; Website (Website) with value www.cloudona.at; and Geburtsdatum (Birth Date) with value 1.1.1970.

Kundennummer	0000000-0000-0000-0000-0000000001
Vorname	Maximilian
Nachname	Mustermann
Titel	Dr.
Geschlecht	männlich
E-Mail	MAXIMILIAN.MUSTERMANN@CLOUDONA.AT
Telefon	+43 670 123 123 123
Website	www.cloudona.at
Geburtsdatum	1.1.1970

Übersichtsseite nach SAP-Anbindung

7.5 Zusammenfassung

MK

Martin Koch

In dieser Übung haben wir OData kennengelernt und ein ODataModel in unser SAPUI5-Frontend integriert. Zudem haben wir unsere Views und Controller angepasst, um Daten aus dem Backend zu konsumieren und möglicherweise geänderte Daten an das Backend zurückzusenden. Die nächste Herausforderung besteht nun darin, die Möglichkeit zur Erstellung neuer Daten zu implementieren.

8 Einleitung

MK

Martin Koch

In dieser Übung werden wir erkunden, wie ein neuer Kunde erstellt werden kann, um sicherzustellen, dass er korrekt im SAP-Backend gespeichert wird und anschließend im Frontend korrekt angezeigt wird.

8.1 Main-view anpassen

MK Martin Koch

Wir werden nun die Navigation so anpassen, dass es möglich ist, auf eine leere Detailansicht im Bearbeitungsmodus zu verlinken. Dies ermöglicht das Hinzufügen eines neuen Kunden direkt von dieser Ansicht aus.

```
{  
  "name": "CreateCustomer",  
  "pattern": "createcustomer",  
  "target": [  
    "TargetCustomer"  
  ]  
}
```

im **manifest.json** eine **neue Route** für die create funktion erstellen

```
{  
  "name": "CreateCustomer",  
  "pattern": "createcustomer",  
  "target": [  
    "TargetCustomer"  
  ]  
}
```

```
        "TargetCustomer"
    ]
}
```

```
<headerToolbar>
    <OverflowToolbar id="main_overflow_toolbar">
        <Title id="main_table_title" text="{i18n>main.table.title}" />
        <ToolbarSpacer/>
        <Button id="main_button_create" icon="sap-icon://add" type="Transparent" text="{i18n>main.create.button}" press="onCreatePressed"/>
    </OverflowToolbar>
</headerToolbar>
```

Im **Header** der Tabelle einen neuen **Button** für das erstellen einfügen mit der Funktion **onCreatePressed**

```
<Button id="main_button_create" icon="sap-icon://add" type="Transparent" te
```

```
onCreatePressed: function() {
    this.getOwnerComponent().getRouter().navTo("CreateCustomer", null, false);
}
```

Im Main.controller die Funktion, welche uns auf den Customer navigiert, implementieren

```
onCreatePressed: function() {
    this.getOwnerComponent().getRouter().navTo("CreateCustomer", null, false);
}
```

Main-View

Kunden

Kundentabelle								+ Hinzufügen
Firstname	Lastname	Title	Gender	Birthdate	Email	Phone	Website	
Maximilian	Mustermensch	Dr.	männlich	2.11.2023	MAXIMILIAN.MUSTEF	+43 676 123 123 124	www.clouddna.at	> X
Maxima	Mustermann	Mag.	weiblich	1.7.1980	MAXIMA.MUSTERMA	+43 676 124 124 124	www.clouddna.com	> X
Maxi	Spaceman	Dr Dr	männlich	1.1.1970	SPACYL@GMAIL.CO			> X

Main-view mit Hinzufügen-Button

8.2 Customer-view mit createEntry anpassen

MK

Martin Koch

Nach der Main-view ist es nun an der Zeit, die Customer-view anzupassen. Für die Erstellungsfunktionalität verwenden wir die Funktion [createEntry](#).

createEntry

Die Funktion "createEntry" im ODataModel von SAPUI5 wird verwendet, um eine temporäre, nicht persistente Entität (Datensatz) zu erstellen, die in der Anwendung bearbeitet und anschließend an den OData-Service gesendet werden kann. Dies ermöglicht das Vorabfüllen von Formularen oder das Erstellen von neuen Datensätzen, bevor sie tatsächlich in der Datenquelle gespeichert werden.

```

onInit: function () {
    let oEditModel = new JSONModel({
        editmode: false
    });

    this.getView().setModel(oEditModel, "editModel");

    let oRouter = this.getOwnerComponent().getRouter();

    oRouter.getRoute("RouteCustomer").attachPatternMatched(this._onPatternMatched, this); 1

    oRouter.getRoute("CreateCustomer").attachPatternMatched(this._onCreatePatternMatched, this);
},

```

```

_onCreatePatternMatched: function (oEvent) {
    this.bCreate = true;

    let oNewCustomerContext = this.getView().getModel().createEntry("/Z_P_CUSTOMER");
    this.getView().bindElement(oNewCustomerContext.getPath());

    this.getView().getModel("editModel").setProperty("/editmode", true);
    this._showCustomerFragment("CustomerEdit");
},
2

```

1. im **onInit** auf unsere neue Route mit der **_onCreatePatternMatched** reagieren

2. **_onCreatePatternMatched** implementieren

```

onInit: function () {
    let oEditModel = new JSONModel({
        editmode: false
    });

    this.getView().setModel(oEditModel, "editModel");

    let oRouter = this.getOwnerComponent().getRouter();

    oRouter.getRoute("RouteCustomer").attachPatternMatched(this._onPatternMatched, this);

    oRouter.getRoute("CreateCustomer").attachPatternMatched(this._onCreatePatternMatched, this);
}

_onCreatePatternMatched: function (oEvent) {
    this.bCreate = true;

    let oNewCustomerContext = this.getView().getModel().createEntry("/Z_P_CUSTOMER");
}

```

```
        this.getView().bindElement(oNewCustomerContext.getPath());  
  
        this.getView().getModel("editModel").setProperty("/editmode",  
        this._showCustomerFragment("CustomerEdit");  
    },
```

Jetzt fehlt nur noch die **onSavePressed**-Funktion anzupassen, damit unsere Daten auch im **Model gespeichert werden**. Dafür verwende wir die [submitChanges](#)-Funktion

submitChanges()

Die Funktion "submitChanges" im ODataModel von SAPUI5 wird verwendet, um die in der Anwendung vorgenommenen Änderungen an den Datensätzen an den OData-Service zu senden, um sie in der Datenquelle zu speichern. Dies ermöglicht das Aktualisieren, Hinzufügen oder Löschen von Datensätzen in der Datenquelle, basierend auf den in der Anwendung vorgenommenen Änderungen.

```

onSavePressed: function () {
    const oModel = this.getView().getModel();
    const sSuccessText = this.bCreate ? this.getLocalizedMessage("dialog.create.success") : this.getLocalizedMessage("dialog.edit.success");

    this.getView().setBusy(true);

    oModel.submitChanges({
        success: (oData, response) => {
            this.getView().setBusy(false);

            MessageBox.success(sSuccessText, {
                onClose: () => {
                    if (this.bCreate) {
                        this._toggleEdit(false);
                        oModel.refresh(true);
                        this.onNavBack();
                    } else {
                        this._toggleEdit(false);
                    }
                }
            });
        },
        error: (oError) => {
            this.getView().setBusy(false);
            MessageBox.error(oError.message);
        }
    });
},

```

submitChanges im onSavePressed hinzufügen

```

onSavePressed: function () {
    const oModel = this.getView().getModel();
    const sSuccessText = this.bCreate ? this.getLocalizedMessage("dialog.create.success") : this.getLocalizedMessage("dialog.edit.success");

    this.getView().setBusy(true);

    oModel.submitChanges({
        success: (oData, response) => {
            this.getView().setBusy(false);

            MessageBox.success(sSuccessText, {
                onClose: () => {
                    if (this.bCreate) {
                        this._toggleEdit(false);
                        oModel.refresh(true);
                        this.onNavBack();
                    } else {
                        this._toggleEdit(false);
                    }
                }
            });
        },
        error: (oError) => {
            this.getView().setBusy(false);
            MessageBox.error(oError.message);
        }
    });
},

```

```
        error: (oError) => {
            this.getView().setBusy(false);
            MessageBox.error(oError.message);
        }
    }) ;
},

```

resetChanges

Die resetChanges-Methode wird verwendet, um alle lokalen, aber noch nicht übermittelten Änderungen an OData-Modellen zurückzusetzen. Dies ermöglicht das Verwerfen von ungespeicherten Änderungen und die Wiederherstellung des ursprünglichen Zustands der Daten.

```

onCancelPressed: function () {
    const oModel = this.getView().getModel();

    if (oModel.hasPendingChanges()) {
        oModel.resetChanges()
    }

    this._toggleEdit(false);

    if (this.bCreate) {
        this.bCreate = !this.bCreate;
        this.onNavBack();
    }
},
_toggleEdit: function(bEditMode){
    const oEditModel = this.getView().getModel("editModel");

    oEditModel.setProperty("/editmode", bEditMode);

    this._showCustomerFragment(bEditMode ? "CustomerEdit" : "CustomerDisplay");
},
onNavBack: function () {
    var oHistory = History.getInstance();
    var sPreviousHash = oHistory.getPreviousHash();

    if (sPreviousHash !== undefined) {
        window.history.go(-1);
    } else {
        var oRouter = this.getOwnerComponent().getRouter();
        oRouter.navTo("Main", {}, true);
    }
},

```

Y

Die "onCancelPressed" und "_toggleEdit" verändern bzw. neu erstellen

```

onCancelPressed: function () {
    const oModel = this.getView().getModel();

    if (oModel.hasPendingChanges()) {
        oModel.resetChanges()
    }

    this._toggleEdit(false);
}

```

```
        if (this.bCreate) {
            this.bCreate = !this.bCreate;
            this.onNavBack();
        }
    },
    _toggleEdit: function(bEditMode) {
        const oEditModel = this.getView().getModel("editModel");

        oEditModel.setProperty("/editmode", bEditMode);

        this._showCustomerFragment(bEditMode ? "CustomerEdit" : "CustomerDisplay");
    },
    onNavBack: function () {
        var oHistory = History.getInstance();
        var sPreviousHash = oHistory.getPreviousHash();

        if (sPreviousHash !== undefined) {
            window.history.go(-1);
        } else {
            var oRouter = this.getOwnerComponent().getRouter();
            oRouter.navTo("Main", {}, true);
        }
    },
},
```

8.3 Zusammenfassung

 Martin Koch

In dieser Übung haben wir erfolgreich Daten über den OData-Service erstellt und gespeichert.
Dies wurde mithilfe der OData-Methoden **createEntry** und **submitChanges** umgesetzt.

9 Einleitung

 Martin Koch

Bisher konnten wir erfolgreich neue Daten erstellen, sie in der Tabelle anzeigen und auch bearbeiten. In diesem Kapitel setzen wir die Implementierung für das Löschen von Daten um.

9.1 Daten löschen

MK Martin Koch

Für das Löschen verwenden wir das "delete"-Attribut der **sap.m.Table**. Dadurch erhalten wir "X"-Symbole am Ende jeder Zeile, an die wir eine Funktion anhängen können. Im Controller entfernen wir dann das ausgewählte Element und aktualisieren die Tabelle.

Main-view anpassen

```
<s:content>
  <Table items="/Z_P_CUSTOMER" id="main_table" headerText="{i18n>main.table.title}"
    mode="Delete" delete="onDeletePressed">
    <headerToolbar>
      <OverflowToolbar id="_IDGenOverflowToolbar1">
        <Title id="_IDGenTitle1" text="{i18n>main.table.title}" />
        <ToolbarSpacer id="_IDGenToolbarSpacer1" />
        <Button id="main_button_create" icon="sap-icon://add" type="Accept" text="{i18n>create}" press="onCreatePressed"/>
        <Button press="onOpenDialog" type="Transparent" icon="sap-icon://search" />
      </OverflowToolbar>
    </headerToolbar>
```

In der Tabelle die Funktion **onDeletePressed** an das **delete-Attribut** übergeben,

```
<Table items="/Z_P_CUSTOMER" id="main_table" headerText="{i18n>main.table.title}"
  mode="Delete" delete="onDeletePressed">
```

Kunden

Kundentabelle									+ Hinzufügen
Kundennummer	Vorname	Nachname	Titel	Geschlecht	Geburtsdatum	E-Mail	Telefon	Website	
0000000-0000-0000-000-000000000001	Maximilian	Mustermensch	Dr.	männlich	1.1.1970	MAXIMILIAN.MU STERMANN@CL OUDDNA.AT	+43 676 123 123 123	www.clouddna.at	> X
0000000-0000-0000-000-000000000002	Maxima	Mustermann	Mag.	weiblich	1.7.1980	MAXIMA.MUSTE RMANN@CLOUD DNA.AT	+43 676 124 124 124	www.clouddna.com	> X
c15a92c4-6620-1ede-97d1-6b4a8d50a5e9	Paul	Proksch		männlich	16.9.2023				> X
c15a92c4-6620-1ede-98bc-376265e8eb6f	asdf	jklö		männlich	1.1.1970				> X
c15a92c4-6620-1eee-97a6-2bbf8634bc7a				weiblich	1.1.1970				> X
c15a92c4-6620-1eee-98a2-d3bf1ace5e10				weiblich	1.1.1970				> X

Tabellen mit löschen-Buttons

Controller-anpassen

```
webapp > controller > Customer.controller.js > ...
You, 5 minutes ago | 2 authors (You and others)
1 sap.ui.define([
2   "sap/m/MessageBox",
3
4 ],
5   /**
6    * @param {typeof sap.ui.core.mvc.Controller} Controller
7    */
8   function () {
9     var MessageBox = sap.m.MessageBox;
10
11   }
12 })
```

Message-Box zu den Imports hinzufügen

```

sap.ui.define([
    ...
    "sap/m/MessageBox",
    ...
],
/***
 * @param {typeof sap.ui.core.mvc.Controller} Controller
 */
function (MessageBox) {

```

```

onDeletePressed: function(oEvent){
    const oListItem = oEvent.getParameters().listItem;
    const oModel = this.getView().getModel();
    const sPath = oListItem.getBindingContext().getPath();
    const oResourceBundle = this.getView().getModel("i18n").getResourceBundle();

    MessageBox.warning(this.getLocalizedText("sureToDeleteQuestion"), {
        title: oResourceBundle.getText("sureToDeleteTitle"),
        actions: [MessageBox.Action.YES, MessageBox.Action.NO],
        emphasizedAction: MessageBox.Action.YES,
        onClose: (oAction)=>{
            if(MessageBox.Action.YES === oAction){
                this.getView().setBusy(true);

                oModel.remove(sPath, {
                    success: (oData, response) => {
                        this.getView().setBusy(false);

                        MessageBox.success(oResourceBundle.getText("dialog.delete.success"));
                        oModel.refresh(true);
                    },
                    error: (oError) => {
                        this.getView().setBusy(false);

                        MessageBox.error(oError.message);
                    }
                });
            }
        }
    });
}

```

1. **onDeletePressed** definieren, die bei Buttonpress aufgerufen

2. im **_onDelete** erstellen wir einen **MessageBox**, die den Löschvorgang für den Benutzer sichtbar macht.

3. Wenn der "OK"-Button gedrückt wird, so wird das ausgewählte **Element gelöscht** und das **Model aktualisiert**

```
onDeletePressed: function(oEvent) {
    const oListItem = oEvent.getParameters().listItem;
    const oModel = this.getView().getModel();
    const sPath = oListItem.getBindingContext().getPath();
    const oResourceBundle = this.getView().getModel("i18n").getResourceBund

    MessageBox.warning(this.getLocalizedText("sureToDeleteQuestion"), {
        title: oResourceBundle.getText("sureToDeleteTitle"),
        actions: [MessageBox.Action.YES, MessageBox.Action.NO],
        emphasizedAction: MessageBox.Action.YES,
        onClose: (oAction)=>{
            if(MessageBox.Action.YES === oAction) {
                this.getView().setBusy(true);

                oModel.remove(sPath, {
                    success: (oData, response) => {
                        this.getView().setBusy(false);

                        MessageBox.success(oResourceBundle.getText("dialog
                        oModel.refresh(true);
                    },
                    error: (oError) => {
                        this.getView().setBusy(false);

                        MessageBox.error(oError.message);
                    }
                });
            }
        });
    });
},
```

i18n pflegen

```
delete=Löschen  
sureToDeleteTitle=Löschvorgang  
sureToDeleteQuestion=Sind Sie sicher, dass Sie diesen Kunden löschen wollen?  
dialog.delete.success=Erfolgreich gelöscht!
```

9.2 Zusammenfassung

 Martin Koch

In dieser Übung haben wir unserer Tabelle mithilfe der "delete"-Funktion des OData Models eine Löschfunktion hinzugefügt. Damit sind nun alle CRUD-Operationen in unserem Programm umgesetzt.

9.5 UI Anpassung

MK Martin Koch

i18n-Einträge austauschen

Durch den OData-Service erhalten wir nicht nur den Datensatzwert, sondern auch weitere Eigenschaften wie beispielsweise das Property "sap:label". In der "metadata.xml" können wir sämtliche dieser Properties einsehen und sie anschließend über das [Metadata-Binding](#) an die UI anbinden.



```

<Property Name="Changedby" Type="Edm.String" sap:label="sap:label" sap:quickinfo="sap:label" sap:display-format="UpperCase" sap:label="User Name"/>
<Property Name="Changedby" Type="Edm.String" MaxLength="12" sap:display-format="UpperCase" sap:label="User Name"/>
<NavigationProperty Name="to_Customer" Relationship="cds_z_customer_ui.assoc_9FE20F64318DBBDC74B6B643440E2F64" FromRole="ToRole_assoc_9FE20F64318DBBDC74B6B643440E2F64" ToRole="FromRole_assoc_9FE20F64318DBBDC74B6B643440E2F64"/>
</EntityType>
<EntityType Name="Z_P_CUSTOMERType" sap:label="CDS Projection - Customer" sap:content-version="1">
  <Key>
    <PropertyRef Name="Customerid"/>
  </Key>
  <Property Name="Delete_mc" Type="Edm.Boolean" sap:label="Dyn. Method Control" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false" sap:label="Delete method control" sap:quickinfo="Delete method control" sap:display-format="Boolean" sap:label="Delete method control" sap:quickinfo="Delete method control"/>
  <Property Name="Update_mc" Type="Edm.Boolean" sap:label="Dyn. Method Control" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false" sap:label="Update method control" sap:quickinfo="Update method control" sap:display-format="Boolean" sap:label="Update method control" sap:quickinfo="Update method control"/>
  <Property Name="to_CustomerDocument_o" Type="Edm.Boolean" sap:label="Dynamic CbA-Control" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false" sap:label="Customer document association" sap:quickinfo="Customer document association" sap:display-format="Boolean" sap:label="Customer document association" sap:quickinfo="Customer document association"/>
  <Property Name="Customerid" Type="Edm.Guid" Nullable="false" sap:label="UUID" sap:quickinfo="16 Byte UUID in 16 Bytes (Raw Format)" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false" sap:label="Customer ID" sap:quickinfo="Customer ID" sap:display-format="String" sap:label="Customer ID" sap:quickinfo="Customer ID"/>
  <Property Name="Firstname" Type="Edm.String" MaxLength="40" sap:label="Firstname" sap:quickinfo="First Name"/>
  <Property Name="Lastname" Type="Edm.String" MaxLength="40" sap:label="Lastname" sap:quickinfo="Last Name"/>
  <Property Name="Title" Type="Edm.String" MaxLength="20" sap:label="Title" sap:quickinfo="Academic Title: Written Form"/>
  <Property Name="Phone" Type="Edm.String" MaxLength="30" sap:display-format="UpperCase" sap:label="Phone" sap:quickinfo="First Telephone No.: Dialing Order" sap:label="Phone" sap:quickinfo="First Telephone No.: Dialing Order"/>
  <Property Name="Email" Type="Edm.String" MaxLength="132" sap:display-format="UpperCase" sap:label="Email" sap:quickinfo="E-mail address"/>
  <Property Name="Gender" Type="Edm.String" MaxLength="1" sap:display-format="UpperCase" sap:label="Gender" sap:quickinfo="Gender of business partner"/>
  <Property Name="Website" Type="Edm.String" MaxLength="64" sap:label="Website" sap:quickinfo="ODD : Website"/>
  <Property Name="Createdby" Type="Edm.String" MaxLength="12" sap:display-format="UpperCase" sap:label="Createdby" sap:quickinfo="User Name" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false" sap:label="Created by" sap:quickinfo="Created by" sap:display-format="String" sap:label="Created by" sap:quickinfo="Created by"/>
  <Property Name="Createdat" Type="Edm.DateTimeOffset" Precision="0" sap:label="Createdat" sap:quickinfo="UTC Time Stamp in Short Form (YYYYMMDDhhmmss)" sap:label="Created at" sap:quickinfo="UTC Time Stamp in Short Form (YYYYMMDDhhmmss)"/>
  <Property Name="Changedby" Type="Edm.String" MaxLength="12" sap:display-format="UpperCase" sap:label="Changedby" sap:quickinfo="User Name" sap:createable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false" sap:label="Changed by" sap:quickinfo="Changed by" sap:display-format="String" sap:label="Changed by" sap:quickinfo="Changed by"/>
  <Property Name="Changedat" Type="Edm.DateTimeOffset" Precision="0" sap:label="Changedat" sap:quickinfo="UTC Time Stamp in Short Form (YYYYMMDDhhmmss)" sap:label="Changed at" sap:quickinfo="UTC Time Stamp in Short Form (YYYYMMDDhhmmss)"/>
  <Property Name="Birthdate" Type="Edm.DateTime" Precision="0" sap:label="Birthdate" sap:quickinfo="Field of type DATS"/>
  <NavigationProperty Name="to_CustomerDocument" Relationship="cds_z_customer_ui.assoc_9FE20F64318DBBDC74B6B643440E2F64" FromRole="FromRole_assoc_9FE20F64318DBBDC74B6B643440E2F64" ToRole="ToRole_assoc_9FE20F64318DBBDC74B6B643440E2F64"/>
</EntityType>
<EntityType Name="SAP__Currency" sap:content-version="1">
  <Key>
    <PropertyRef Name="CurrencyCode"/>
  </Key>
</EntityType>

```

0. Im "metadata.xml" zum Entitätstyp "Z_P_CustomerType" navigieren

1. Im Type sehen wir alle unsere Felder, die wir auch vorher im Programm benutzt haben

2. Jedes Feld hat auch ein "sap:label"-Property.

```
<columns>
    <Column id="main_column_firstname">
        <Text id="main_text_firstname" text="{/#Z_P_CUSTOMER/Firstname/@sap:label}"/>
    </Column>
    <Column id="main_column_lastname">
        <Text id="main_text_lastname" text="{/#Z_P_CUSTOMER/Lastname/@sap:label}"/>
    </Column>
    <Column id="main_column_title" minWidth="Desktop" demandPopin="true">
        <Text id="main_text_title" text="{/#Z_P_CUSTOMER>Title/@sap:label}"/>
    </Column>
    <Column id="main_column_gender" minWidth="Desktop">
        <Text id="main_text_gender" text="{/#Z_P_CUSTOMER/Gender/@sap:label}"/>
    </Column>
    <Column id="main_column_birthdate" minWidth="Desktop" demandPopin="true">
        <Text id="main_text_birthdate" text="{/#Z_P_CUSTOMER/Birthdate/@sap:label}"/>
    </Column>
    <Column id="main_column_email" minWidth="Desktop">
        <Text id="main_text_email" text="{/#Z_P_CUSTOMER>Email/@sap:label}"/>
    </Column>
    <Column id="main_column_phone" minWidth="Desktop">
        <Text id="main_text_phone" text="{/#Z_P_CUSTOMER/Phone/@sap:label}"/>
    </Column>
    <Column id="main_column_website" minWidth="Desktop">
        <Text id="main_text_website" text="{/#Z_P_CUSTOMER/Website/@sap:label}"/>
    </Column>
</columns>
```

In der Main.view.xml könne wir in der Tabelle bei jedem Column nun den "text"-Wert mit der Metadata Beschreibung austauschen

```
<columns>
    <Column id="main_column_firstname">
        <Text id="main_text_firstname" text="{/#Z_P_CUSTOMER/Firstname/@sap:label}"/>
    </Column>
    <Column id="main_column_lastname">
        <Text id="main_text_lastname" text="{/#Z_P_CUSTOMER/Lastname/@sap:label}"/>
    </Column>
    <Column id="main_column_title" minWidth="Desktop" demandPopin="true">
        <Text id="main_text_title" text="{/#Z_P_CUSTOMER>Title/@sap:label}"/>
    </Column>
```

```

<Column id="main_column_gender" minScreenWidth="Desktop">
    <Text id="main_text_gender" text="{/#Z_P_CUSTOMER/Gender/@sap:label}">
</Column>
<Column id="main_column_birthdate" minScreenWidth="Desktop" demandPopin="true">
    <Text id="main_text_birthdate" text="{/#Z_P_CUSTOMER/Birthdate/@sap:label}">
</Column>
<Column id="main_column_email" minScreenWidth="Desktop">
    <Text id="main_text_email" text="{/#Z_P_CUSTOMER/Email/@sap:label}">
</Column>
<Column id="main_column_phone" minScreenWidth="Desktop">
    <Text id="main_text_phone" text="{/#Z_P_CUSTOMER/Phone/@sap:label}">
</Column>
<Column id="main_column_website" minScreenWidth="Desktop">
    <Text id="main_text_website" text="{/#Z_P_CUSTOMER/Website/@sap:label}">
</Column>
</columns>

```

Kunden

Kundentabelle								+ Hinzufügen
Firstname	Lastname	Title	Gender	Birthdate	Email	Phone	Website	
Maximilian	Mustermensch	Dr.	männlich	2.11.2023	MAXIMILIAN.MUSTEF	+43 676 123 123 124	www.clouddna.at	> X
Maxima	Mustermann	Mag.	weiblich	1.7.1980	MAXIMA.MUSTERMA	+43 676 124 124 124	www.clouddna.com	> X
Maxi	Spaceman	Dr Dr	männlich	1.1.1970	SPACYL@GMAIL.CO			> X

Main-View nach Metadata-Binding

ObjectPageLayout erweitern

In erfahrenen Händen wird das **ObjectPageLayout** zu einem leistungsstarken Werkzeug. Im vorherigen Kapitel haben wir in der Kundenansicht die herkömmliche Page durch eine schlichte Object Page ersetzt. Jetzt ist es an der Zeit, zusätzliche Funktionen zu integrieren, um die Seite visuell ansprechender zu gestalten.

Header anpassen

Der Header wird nun auf- und zuklappbar sein, wobei der volle Name als Titel und das Geburtsdatum als Header-Content erscheinen. Darüber hinaus wird ein Avatar ebenfalls angezeigt.

```
xmlns:s="sap.t.semantic"
xmlns="sap.m">
<u:ObjectPageLayout id="cust_objectpagelayout"
    showTitleInHeaderContent="true"
    showFooter="={ ${editModel}/editmode}"
    upperCaseAnchorBar="false">
    <u:headerTitle>
        <u:objectPageDynamicHeaderTitle>
            <u:expandedHeading>
                <Title text="{Firstname} {Lastname}" />
            </u:expandedHeading>

            <u:snappedHeading>
                <FlexBox fitContainer="true" alignItems="Center">
                    <Avatar src="sap-icon://person-placeholder" class="sapUiTinyMarginEnd"/>
                    <Title text="{Firstname} {Lastname}" />
                </FlexBox>
            </u:snappedHeading>
        </u:objectPageDynamicHeaderTitle>
    </u:headerTitle>

    <u:headerContent>
        <FlexBox>
            <Avatar class="sapUiSmallMarginEnd" src="sap-icon://person-placeholder" displaySize="L" />

            <VBox>
                <Label text="{#Z_P_CUSTOMERType/Birthdate@sap:label}" labelFor="cust_header_birthdate"/>
                <Title id="cust_header_birthdate" text="{path: 'Birthdate', formatter: '.dateFormatter'}" titleStyle="H6"/>
            </VBox>
        </FlexBox>
    </u:headerContent>
</u:ObjectPageLayout>
```

1

2

1. "*expandedHeading*" und "*snappedHeading*" einfügen

2. "*headerContent*" einfügen.

```
<u:headerTitle>
    <u:objectPageDynamicHeaderTitle>
        <u:expandedHeading>
            <Title text="{Firstname} {Lastname}" />
        </u:expandedHeading>

        <u:snappedHeading>
```

```

<FlexBox fitContainer="true" alignItems="Center">
    <Avatar src="sap-icon://person-placeholder" class="sapUiTinyMarginEnd" />
    <Title text="{Firstname} {Lastname}" />
</FlexBox>
</u:snappedHeading>

<u:actions>
    <HBox>
        <Button id="cust_edit_button" text="{i18n>cust.edit.button}" press="onEditPressed" visible="={!${editModel>/editmode}}"/>
    </HBox>
</u:actions>
</u:ObjectPageDynamicHeaderTitle>
</u:headerTitle>

<u:headerContent>
    <FlexBox>
        <Avatar class="sapUiSmallMarginEnd" src="sap-icon://person-placeholder" />
        <VBox>
            <Label text="{#/Z_P_CUSTOMERType/Birthdate/@sap:label}" labelFor="cust_header_birthdate" />
            <Title id="cust_header_birthdate" text="{path: 'Birthdate', format: 'DD.MM.YYYY'}" />
        </VBox>
    </FlexBox>
</u:headerContent>

```

The screenshot shows the expanded header of an SAP Fiori object page. The header contains the customer's name, a placeholder icon, and the birthdate. A 'Bearbeiten' button is also present.

Header ausgeklappt

The screenshot shows the collapsed header of an SAP Fiori object page. The header only displays the customer's name, a placeholder icon, and a dropdown arrow.

Sektionsinhalt auf zwei Subsektionen aufteilen

Das Ziel ist es den Inhalt der Customer-view auf zwei Teilbereiche aufzuteilen. Einen für die Kundeninformationen (Firstname, Lastname, Title, Gender, Birthdate) und einen Für Kontaktmöglichkeiten (Email, Phone, Website). Dafür müssen wir nur eine zweite "ObjectPageSubsection" einfügen und die dafür notwendigen Felder einfügen

```
<u:objectPageSection>
  <u:subSections>
    <u:objectPageSubSection titleUppercase="true" title="Kundeninformationen">
      <u:blocks>
        <f:SimpleForm title="Grunddaten">
          <Label text="#{Z_P_CUSTOMERType/Firstname/@sap:label}" labelFor="firstname" />
          <Text id="firstname" text="#{Firstname}" />

          <Label text="#{Z_P_CUSTOMERType/Lastname/@sap:label}" labelFor="lastname" />
          <Text id="lastname" text="#{Lastname}" />

          <Label text="#{Z_P_CUSTOMERType>Title/@sap:label}" labelFor="title" /> You, an hour ago
          <Text id="title" text="#{Title}" />

          <Label text="#{Z_P_CUSTOMERType/Gender/@sap:label}" labelFor="gender" />
          <Text id="gender" text="#{Gender} == '1' ? ${i18n>female} : ${i18n>male}" />

          <Label text="#{Z_P_CUSTOMERType/Birthdate/@sap:label}" labelFor="birthdate" />
          <Text id="birthdate" text="#{path: 'Birthdate', formatter: '.dateFormatter'}" />
        </f:SimpleForm>
      </u:blocks>
    </u:objectPageSubSection>
    <u:objectPageSubSection>
      <u:blocks>
        <f:SimpleForm title="Kontaktmöglichkeiten">
          <Label text="#{Z_P_CUSTOMERType>Email/@sap:label}" labelFor="email" />
          <Text id="d" text="#{Email}" />

          <Label text="#{Z_P_CUSTOMERType/Phone/@sap:label}" labelFor="phone" />
          <Text id="g" text="#{Phone}" />

          <Label text="#{Z_P_CUSTOMERType/Website/@sap:label}" labelFor="website" />
          <Text id="e" text="#{Website}" />
        </f:SimpleForm>
      </u:blocks>
    </u:objectPageSubSection>
  </u:subSections>
</u:objectPageSection>
```

CustomerDisplay.fragment.xml mit zwei ObjectPageSubsections und den zuvor gelernten
Metadata Binding

```
<u:ObjectPageSection title="Kundeninformationen">
    <u:subSections>
        <u:ObjectPageSubSection titleUppercase="true">
            <u:blocks>
                <f:SimpleForm title="Grunddaten">
                    <Label text="#{Z_P_CUSTOMERType/Firstname/@sap:label}" />
                    <Text id="firstname" text="#{Firstname}" />

                    <Label text="#{Z_P_CUSTOMERType/Lastname/@sap:label}" />
                    <Text id="lastname" text="#{Lastname}" />

                    <Label text="#{Z_P_CUSTOMERType>Title/@sap:label}" />
                    <Text id="title" text="#{Title}" />

                    <Label text="#{Z_P_CUSTOMERType/Gender/@sap:label}" />
                    <Text id="gender" text="#{Gender} == '1' ? ${i18n}: ${Gender}" />

                    <Label text="#{Z_P_CUSTOMERType/Birthdate/@sap:label}" />
                    <Text id="birthdate" text="#{path: 'Birthdate', formatter: 'date'}" />
                </f:SimpleForm>
            </u:blocks>
        </u:ObjectPageSubSection>

        <u:ObjectPageSubSection>
            <u:blocks>
                <f:SimpleForm title="Kontaktmöglichkeiten">
                    <Label text="#{Z_P_CUSTOMERType>Email/@sap:label}" />
                    <Text id="d" text="#{Email}" />

                    <Label text="#{Z_P_CUSTOMERType/Phone/@sap:label}" />
                    <Text id="g" text="#{Phone}" />

                    <Label text="#{Z_P_CUSTOMERType/Website/@sap:label}" />
                    <Text id="e" text="#{Website}" />
                </f:SimpleForm>
            </u:blocks>
        </u:ObjectPageSubSection>
    </u:subSections>
</u:ObjectPageSection>
```

```

    </u:subSections>
</u:ObjectPageSection>

```

```

<u:subSections>
<u:ObjectPageSubSection titleUppercase="true" >
<u:blocks>
<f:SimpleForm title="{i18n>fragment.edit.info}">
    <Label text="{#/Z_P_CUSTOMERType/Firstname/@sap:label}" labelFor="fragment_edit_firstname" />
    <Input id="fragment_edit_firstname" value="{Firstname}" />

    <Label text="{#/Z_P_CUSTOMERType/Lastname/@sap:label}" labelFor="fragment_edit_lastname" />
    <Input id="fragment_edit_lastname" value="{Lastname}" />

    <Label text="{#/Z_P_CUSTOMERType/Title/@sap:label}" labelFor="fragment_edit_title" />
    <Input id="fragment_edit_title" value="{Title}" />

    <Label text="{#/Z_P_CUSTOMERType/Gender/@sap:label}" labelFor="fragment_edit_gender" />
    <Select id="fragment_edit_gender" selectedKey="{Gender}" items="{genderModel>genders}">
        <items>
            <core:Item key="{genderModel>key}" text="{path: 'genderModel>text', formatter: '.genderFormatter'}" />
        </items>
    </Select>

    <Label text="{#/Z_P_CUSTOMERType/Birthdate/@sap:label}" labelFor="fragment_edit_birthdate" />
    <DatePicker id="fragment_edit_birthdate" value="{path: 'Birthdate', type: 'sap.ui.model.type.Date', formatOptions: {UTC: true}}" />
</f:SimpleForm>
</u:blocks>
</u:ObjectPageSubSection>

<u:ObjectPageSubSection>
<u:blocks>
<f:SimpleForm title="{i18n>fragment.edit.contact}">
    <Label text="{#/Z_P_CUSTOMERType/Email/@sap:label}" labelFor="fragment_edit_email" />
    <Input id="fragment_edit_email" type="Email" value="{Email}" />

    <Label text="{#/Z_P_CUSTOMERType/Phone/@sap:label}" labelFor="fragment_edit_phone" />
    <Input id="fragment_edit_phone" type="Tel" value="{Phone}" />

    <Label text="{#/Z_P_CUSTOMERType/Website/@sap:label}" labelFor="fragment_edit_website" />
    <Input id="fragment_edit_website" value="{Website}" />
</f:SimpleForm>
</u:blocks>
</u:ObjectPageSubSection>
</u:subSections>
</u:ObjectPageSection>

```

CustomerEdit.fragment.xml mit zwei ObjectPageSubsections und den zuvor gelernten Metadata Binding

```

<u:ObjectPageSection title="{i18n>fragment.edit.title}">
    <u:subSections>
        <u:ObjectPageSubSection titleUppercase="true" >
            <u:blocks>
                <f:SimpleForm title="{i18n>fragment.edit.info}">
                    <Label text="{#/Z_P_CUSTOMERType/Firstname/@sap:label}" labelFor="fragment_edit_firstname" />
                    <Input id="fragment_edit_firstname" value="{Firstname}" />

                    <Label text="{#/Z_P_CUSTOMERType/Lastname/@sap:label}" labelFor="fragment_edit_lastname" />
                    <Input id="fragment_edit_lastname" value="{Lastname}" />

```

```

<Input id="fragment_edit_lastname" value="{Lastname}" />

<Label text="{/#Z_P_CUSTOMERType/Title/@sap:label}" label="Title" />
<Input id="fragment_edit_title" value="{Title}" />

<Label text="{/#Z_P_CUSTOMERType/Gender/@sap:label}" label="Gender" />
<Select id="fragment_edit_gender" selectedKey="{Gender}" type="radio">
    <items>
        <core:Item key="{genderModel}key" text="{path:label}" />
    </items>
</Select>

<Label text="{/#Z_P_CUSTOMERType/Birthdate/@sap:label}" label="Birthdate" />
<DatePicker id="fragment_edit_birthdate" value="{birthdate}" type="date" />
</f:SimpleForm>
</u:blocks>
</u:ObjectPageSubSection>

<u:ObjectPageSubSection>
    <u:blocks>
        <f:SimpleForm title="{i18n>fragment.edit.contact}">
            <Label text="{/#Z_P_CUSTOMERType/Email/@sap:label}" label="Email" />
            <Input id="fragment_edit_email" type="Email" value="{Email}" />

            <Label text="{/#Z_P_CUSTOMERType/Phone/@sap:label}" label="Phone" />
            <Input id="fragment_edit_phone" type="Tel" value="{Phone}" />

            <Label text="{/#Z_P_CUSTOMERType/Website/@sap:label}" label="Website" />
            <Input id="fragment_edit_website" value="{Website}" />
        </f:SimpleForm>
    </u:blocks>
</u:ObjectPageSubSection>
</u:subSections>
</u:ObjectPageSection>

```

URL-Helper einbindung

Im nächsten Abschnitt integrieren wir den **URLHelper**, um automatisch E-Mails zu versenden oder Telefonanrufe zu initiieren.

Der [URLHelper](#) in SAPUI5 ist ein Hilfsmodul, das Funktionen bereitstellt, um die Handhabung von URLs in einer Anwendung zu erleichtern. Mit dem URLHelper können Entwickler auf einfache Weise URLs generieren oder Aktionen wie das Öffnen von Links oder das Senden von E-Mails durchführen. Es bietet Methoden wie normalize, parse, buildQuery, redirectTo und triggerEmail für eine effiziente Verwaltung von URLs und unterstützt so eine verbesserte Benutzererfahrung in SAPUI5-Anwendungen.

```
_getVal: function(evt) {
    return evt.getSource().getText();
},
handleTelPress: function (evt) {
    sap.m.URLHelper.triggerTel(this._getVal(evt));
},
handleEmailPress: function (evt) {
    sap.m.URLHelper.triggerEmail(this._getVal(evt), "Info Request", false, false, false, true);
});
```

Nötige Funktionen für URL Helper. In den **Customer.controller.js** und **Main.controller.js** einbinden

Wird direkt über den Namensraum angesprochen, muss also nicht importiert werden

```

_getVal: function(evt) {
    return evt.getSource().getText();
},
handleTelPress: function (evt) {
    sap.m.URLHelper.triggerTel(this._getVal(evt));
},
handleEmailPress: function (evt) {
    sap.m.URLHelper.triggerEmail(this._getVal(evt), "Info Request", false,
}

```

```

<items>
    <ColumnListItem type="Navigation" press="onListItemClicked">
        <cells>
            <Text text="{Firstname}" />
            <Text text="{Lastname}" />
            <Text text="{Title}" />
            <Text text="{= ${Gender} === '1' ? ${i18n>female} : ${i18n>male} }" />
            <Text text="{path: 'Birthdate', formatter: '.dateFormatter'}" />
            <Link text="{Email}" press="handleEmailPress"/>
            <Link text="{Phone}" press="handleTelPress"/>
            <Link text=" {website}" href="https:// {website}" target="_blank" />
        </cells>
    </ColumnListItem>
</items>

```

URL Helper Funktionen in Main.view.xml

```

<items>
    <ColumnListItem type="Navigation" press="onListItemClicked">
        <cells>
            <Text text="{Firstname}" />
            <Text text="{Lastname}" />
            <Text text="{Title}" />
            <Text text="{= ${Gender} === '1' ? ${i18n>female} : ${i18n>male} }" />
            <Text text="{path: 'Birthdate', formatter: '.dateFormatter'}" />
            <Link text="{Email}" press="handleEmailPress"/>

```

```

        <Link text="{Phone}" press="handleTelPress"/>
        <Link text="{Website}" href="https://'{Website}" target="_blank"
    </cells>
</ColumnListItem>
</items>

```

```

<u:ObjectPageSubSection>
    <u:blocks>
        <f:SimpleForm title="{i18n}fragment.edit.contact">
            <Label text="{/#Z_P_CUSTOMERType/Email/@sap:label}" labelFor="fragment_display_email" />
            <Link id="fragment_display_email" text="{Email}" press="handleEmailPress"/>

            <Label text="{/#Z_P_CUSTOMERType/Phone/@sap:label}" labelFor="fragment_display_phone" />
            <Link id="fragment_display_phone" text="{Phone}" press="handleTelPress"/>

            <Label text="{/#Z_P_CUSTOMERType/Website/@sap:label}" labelFor="fragment_display_website" />
            <Link id="fragment_display_website" text="{Website}" href="https://'{Website}" target="_blank"/>
        </f:SimpleForm>
    </u:blocks>
</u:ObjectPageSubSection>
,:subSections

```

URL Helper Funktionen in CustomerDisplay.fragment.xml

```

<u:ObjectPageSubSection>
    <u:blocks>
        <f:SimpleForm title="{i18n}fragment.edit.contact">
            <Label text="{/#Z_P_CUSTOMERType/Email/@sap:label}" labelFor="fragment_display_email" />
            <Link id="fragment_display_email" text="{Email}" press="handleEmailPress"/>

            <Label text="{/#Z_P_CUSTOMERType/Phone/@sap:label}" labelFor="fragment_display_phone" />
            <Link id="fragment_display_phone" text="{Phone}" press="handleTelPress"/>

            <Label text="{/#Z_P_CUSTOMERType/Website/@sap:label}" labelFor="fragment_display_website" />
            <Link id="fragment_display_website" text="{Website}" href="https://'{Website}" target="_blank"/>
        </f:SimpleForm>
    </u:blocks>
</u:ObjectPageSubSection>

```

Kundentabelle					+ Hinzufügen		
Firstname	Lastname	Title	Gender	Birthdate	Email	Phone	Website
Maximilian	Mustermensch	Dr.	männlich	2.11.2023	MAXIMILIAN.MUSTEF	+43 676 123 123 124	www.clouddna.at
Maxima	Mustermann	Mag.	weiblich	1.7.1980	MAXIMA.MUSTERMA	+43 676 124 124 124	www.clouddna.com
Maxi	Spaceman	Dr Dr	männlich	1.1.1970	SPACYL@GMAIL.COM		

URL Hepler in Main-view

```

cust.edit.button=Bearbeiten
cust.attachments=Anhänge
cust.cancel.button=Abbrechen
cust.save.button=Speichern

```

```

fragment.edit.title=Kundeninformationen
fragment.edit.info=Grunddaten
fragment.edit.contact=Kontaktmöglichkeiten

```

i18n Pflegen

```

cust.edit.button=Bearbeiten
cust.attachments=Anhänge
cust.cancel.button=Abbrechen
cust.save.button=Speichern

fragment.edit.title=Kundeninformationen
fragment.edit.info=Grunddaten
fragment.edit.contact=Kontaktmöglichkeiten

```

Maximilian Mustermensch

 Bearbeiten



Birthdate
2.11.2023



KUNDENINFORMATIONEN

Grunddaten

Firstname: Maximilian
Lastname: Mustermensch
Title: Dr.
Gender: männlich
Birthdate: 2.11.2023

Kontaktmöglichkeiten

Email: MAXIMILIAN.MUSTERMANN@CLOUDDNA.AT
Phone: +43 676 123 124
Website: www.clouddna.at

Display-Page

Maximilian Mustermensch



Birthdate
2.11.2023



KUNDENINFORMATIONEN

Grunddaten

Firstname:
Lastname:
Title:
Gender: ▼
Birthdate: 

Kontaktmöglichkeiten

Email:
Phone:
Website:

 Speichern

 Abbrechen

Edit-Page

10 Einleitung

MK

Martin Koch

Der Gender-Formatter soll als ein eigenständiges Modul ausgelagert und geladen werden. Die Applikation soll um einen Base-Controller erweitert werden. Im Base Controller sollen die folgenden Convenience-Functions gesammelt werden:

- Backnavigation
- GetRouter
- Auslesen von Texten aus dem i18n mit und ohne Parameter
- Logging
- Lesen von Named- und Default-Models
- Setzen von Named- und Default-Models
- Setzen der Content Density

Verwendete Controls: -

Verwendete Technologien: Modularisierung, Vererbung, Refactoring

10.1 Modul erstellen und Formatter auslagern (optional)

MK Martin Koch

The screenshot shows a SAP UI5 application structure in the Explorer view of a code editor. The project tree includes 'ZHOUIS', 'vscode', 'dist', 'node_modules', 'webapp', 'controller', 'css', 'data', 'i18n', 'localService', 'model', 'test', 'utils', 'view', and 'fragment'. In the 'fragment' folder, there are files for 'ChangeCustomer.fragment.xml', 'DisplayCustomer.fragment.xml', 'App.view.xml', 'Customer.view.xml', 'Main.view.xml', 'Component.js', 'index.html', and 'manifest.json'. A file named '.gitignore' is also present. The main editor window displays the XML content of 'ChangeCustomer.fragment.xml'. A modal dialog titled 'New File' is open, prompting for a file name. The input field contains 'formatter/HOUI5Formatter.js'. Below the input field are 'OK' and 'Cancel' buttons. The code in the editor window is as follows:

```
<core:FragmentDefinition xmlns:core="sap.ui.core" xmlns:m="sap.m" xmlns:f="sap.ui.layout.form">
  <f:SimpleForm id="edit_simpleform">
    <!-- Content of the form -->
  </f:SimpleForm>
</core:FragmentDefinition>
```

1. Datei anlegen

```
You, 2 days ago | 1 author (You)
1 sap.ui.define([], 
2   function () {
3     "use strict";
4     return {
5       genderFormatter: function (sKey) {
6         let oView = this.getView();
7         let oI18nModel = oView.getModel("i18n");
8         let oResourceBundle = oI18nModel.getResourceBundle();
9         let sText = oResourceBundle.getText(sKey);
10        return sText;
11      },
12
13      dateFormatter: function(date) {
14        let dateObj = new Date(date);
15        return dateObj.getDate() + "." + (dateObj.getMonth() + 1) + "." + dateObj.getFullYear();
16      }
17    }
18); You, a month ago * commit until smartTables
```

1. Modul ausprogrammieren

```
sap.ui.define([], 
  function () {
    "use strict";
    return {
      genderFormatter: function (sKey) {
        let oView = this.getView();
        let oI18nModel = oView.getModel("i18n");
        let oResourceBundle = oI18nModel.getResourceBundle();
        let sText = oResourceBundle.getText(sKey);
        return sText;
      },
      dateFormatter: function(date) {
        let dateObj = new Date(date);
        return dateObj.getDate() + "." + (dateObj.getMonth() + 1) + "."
      }
    })
  );
```

```

    "at/clouddna/training00/zhoui5/controller/formatter/HOUISFormatter"
],
1
/**
 * @param {typeof sap.ui.core.mvc.Controller} Controller
 */
function (Controller, syncStyleClass, Fragment, Filter, FilterOperator, MessageBox, HOUISFormatter) {
    "use strict";

    return Controller.extend("at.clouddna.training00.zhoui5.controller.Main", {
        ...HOUISFormatter,
2
3

```

1. Import hinzufügen

2. Import in einen Parameter speichern

3. Modul über spread-Operator in die Klasse anfügen

```

<items>
    <ColumnListItem type="Navigation" press="onListItemClicked">
        <cells>
            <Text text="{Firstname}" />
            <Text text="{Lastname}" />
            <Text text="{Title}" />
            <Text text="{= ${Gender} === '1' ? ${i18n>female} : ${i18n>male} }" />
            <Text text="{path: 'Birthdate', formatter: '.dateFormatter'}" />
            <Link text="{Email}" press="handleEmailPress"/> You, 2 days ago · h
            <Link text="{Phone}" press="handleTelPress"/>
            <Link text="{Website}" href="https://'{Website}" target="_blank"/>
        </cells>
    </ColumnListItem>

```

1. Formatter im Main.view.xml anpassen

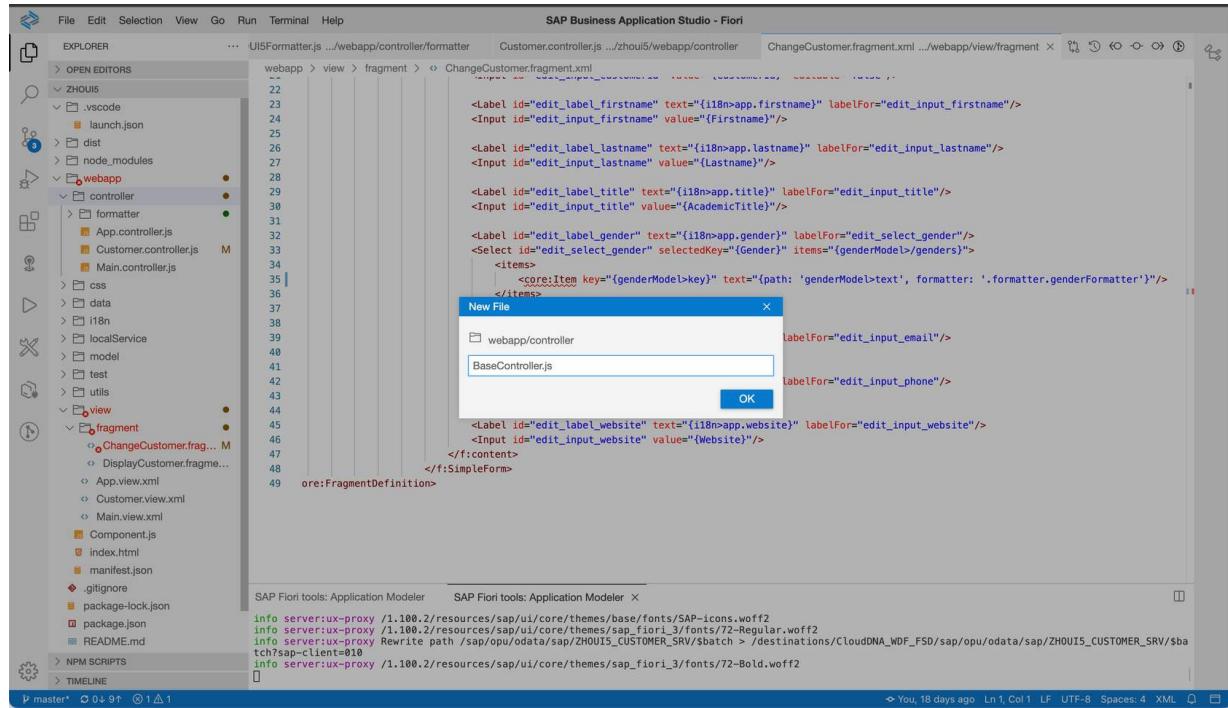
```
<Text text="{path: 'Birthdate', formatter: '.dateFormatter'}" />
```

10.2 BaseController erstellen

MK Martin Koch

BaseController

Der BaseController muss angelegt und mit den jeweiligen Funktionen erweitert werden, die später in den weiteren Controllern wiederverwendet werden sollen.



The screenshot shows the SAP Business Application Studio interface for a Fiori application named ZHOUIS. The left sidebar displays the project structure under 'EXPLORER'. In the center, the code editor shows a fragment XML file named 'ChangeCustomer.fragment.xml'. A modal dialog titled 'New File' is open, prompting the user to enter the file name 'BaseController.js'. The code editor contains XML fragments for editing customer details like first name, last name, title, gender, and website. At the bottom of the code editor, there is a log window showing various server logs related to the application's runtime environment.

1. Datei BaseController.js anlegen



The screenshot shows a code editor window in SAP Studio. The active tab is 'BaseController.js'. The code is as follows:

```
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller"
3 ],
4   /**
5    * @param {typeof sap.ui.core.mvc.Controller} Controller
6    */
7   function (Controller) {
8     "use strict";
9
10    return Controller.extend("at.clouddna.training00.zhoui5.controller.BaseController", [
11      /**
12       * @param {sap.ui.core.routing.History} History
13       */
14      {
15        /**
16         * @param {sap/base/Log} Log
17         */
18        constructor: function (History, Log) {
19          "use strict";
20
21          this._history = History;
22          this._log = Log;
23        }
24      }
25    });
26  };
27
```

1. Leere Implementierung dieses Moduls einfügen

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

    return Controller.extend("at.clouddna.training00.zhoui5.controller
      BaseController", {

    }) ;
  }) ;
```

The screenshot shows a code editor with the file `Customer.controller.js` open. The code is a SAPUI5 controller definition. Three specific parts of the code are highlighted with purple circles and numbers:

- 1.** A call to `sap.ui.define` with parameters including `"sap/ui/core/routing/History"`. This highlights the import of the History module.
- 2.** The declaration of the controller's constructor, which takes `Controller` and `History` as parameters. This highlights the storage of the History object in a parameter.
- 3.** The implementation of the `onNavBack` function, which checks if `sPreviousHash` is defined and either performs a back navigation or navigates to the "Main" route depending on the result.

```

Customer.controller.js .../zhou15/webapp/controller      ChangeCustomer.fragment.xml .../webapp/view/fragment      BaseController.js ×      Main.controller.js .../zhou15/webapp/controller
webapp > controller > BaseController.js > sap.ui.define() callback > ...
1  sap.ui.define([
2    "sap/ui/core/mvc/Controller",
3    "sap/ui/core/routing/History"1
4  ],
5  function (Controller, History) {
6    "use strict";
7
8    return Controller.extend("at.clouddna.training00.zhou15.controller.BaseController", {
9
10   onNavBack: function () {
11     var oHistory = History.getInstance();
12     var sPreviousHash = oHistory.getPreviousHash();
13
14     if (sPreviousHash !== undefined) {
15       window.history.go(-1);
16     } else {
17       var oRouter = this.getOwnerComponent().getRouter();
18       oRouter.navTo("Main", {}, true);
19     }
20   }
21 });
22 });
23
24

```

1. History-Import einfügen

2. Import in ein Parameter speichern

3. onNavBack-Funktion implementieren

```

onNavBack: function () {
  var oHistory = History.getInstance();
  var sPreviousHash = oHistory.getPreviousHash();

  if (sPreviousHash !== undefined) {
    window.history.go(-1);
  } else {
    var oRouter = this.getOwnerComponent().getRouter();
    oRouter.navTo("Main", {}, true);
  }
}

```

The screenshot shows the SAPUI5 IDE interface with the file `BaseController.js` open. The code defines a BaseController that extends another BaseController and includes logging functions. Annotations are overlaid on the code:

- Annotation 1:** Points to the import statement `sap/base/Log`.
- Annotation 2:** Points to the declaration of the BaseController constructor.
- Annotation 3:** Points to the block containing the logging methods (`logDebug`, `logError`, etc.).

```

Customer.controller.js .../zhoui5/webapp/controller      ChangeCustomer.fragment.xml .../webapp/view/fragment      BaseController.js ×      Main.controller.js .../zhoui5/webapp/controller
webapp > controller > BaseController.js > sap.ui.define() callback > ...
1  sap.ui.define([
2    "sap/ui/core/mvc/Controller",
3    "sap/ui/core/routing/History",
4    "sap/base/Log"
5  ],
6  function (Controller, History, Log) {
7    "use strict";
8
9    return Controller.extend("at.cloudDNA.training00.zhoui5.controller.BaseController", {
10
11      logDebug: function(sMessage) {
12        let oLogger = Log.getLogger(this.getView().getControllerName());
13        oLogger.debug("DEBUG - " + sMessage);
14      },
15
16      logError: function(sMessage) {
17        let oLogger = Log.getLogger(this.getView().getControllerName());
18        oLogger.error("ERROR - " + sMessage);
19      },
20
21      logFatal: function(sMessage) {
22        let oLogger = Log.getLogger(this.getView().getControllerName());
23        oLogger.fatal("FATAL - " + sMessage);
24      },
25
26      logInfo: function(sMessage) {
27        let oLogger = Log.getLogger(this.getView().getControllerName());
28        oLogger.info("INFO - " + sMessage);
29      },
30
31      logTrace: function(sMessage) {
32        let oLogger = Log.getLogger(this.getView().getControllerName());
33        oLogger.trace("TRACE - " + sMessage);
34      },
35
36      logWarning: function(sMessage) {
37        let oLogger = Log.getLogger(this.getView().getControllerName());
38        oLogger.warning("WARNING - " + sMessage);
39      },

```

1. Log-Import einfügen

2. Import in ein Parameter speichern

3. Funktionen für die einzelnen Log-Einträge anlegen

```

logDebug: function (sMessage) {
  let oLogger = Log.getLogger(this.getView().getControllerName());
  oLogger.debug("DEBUG - " + sMessage);
}

logError: function (sMessage) {
  let oLogger = Log.getLogger(this.getView().getControllerName());
  oLogger.error("ERROR - " + sMessage);
}

logFatal: function (sMessage) {
  let oLogger = Log.getLogger(this.getView().getControllerName());
  oLogger.fatal("FATAL - " + sMessage);
}

logInfo: function (sMessage) {
  let oLogger = Log.getLogger(this.getView().getControllerName());
  oLogger.info("INFO - " + sMessage);
}

logTrace: function (sMessage) {
  let oLogger = Log.getLogger(this.getView().getControllerName());
  oLogger.trace("TRACE - " + sMessage);
}

logWarning: function (sMessage) {
  let oLogger = Log.getLogger(this.getView().getControllerName());
  oLogger.warning("WARNING - " + sMessage);
}

```

```

    } ,

    logInfo: function (sMessage) {
        let oLogger = Log.getLogger(this.getView().getControllerName());
        oLogger.info("INFO - " + sMessage);
    } ,

    logTrace: function (sMessage) {
        let oLogger = Log.getLogger(this.getView().getControllerName());
        oLogger.trace("TRACE - " + sMessage);
    } ,

    logWarning: function (sMessage) {
        let oLogger = Log.getLogger(this.getView().getControllerName());
        oLogger.warning("WARNING - " + sMessage);
    } ,

```

The screenshot shows a browser developer tools interface with the code editor open to the index.html file. The file contains the following code:

```

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     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <title>App Title</title>
8     <style>
9         html, body, body > div, #container, #container-uiarea {
10             height: 100%;
11         }
12     </style>
13     <script id="sap-ui-bootstrap"
14         src="resources/sap-ui-core.js"
15         data-sap-ui-theme="sap_fiori_3"
16         data-sap-ui-resourceroots='{
17             "at.clouddna.training00.zhoui5": "./"
18         }'
19         data-sap-ui-compatVersion="edge"
20         data-sap-ui-async="true"
21         data-sap-ui-frameOptions="trusted"
22         data-sap-ui-loglevel="debug" You a few seconds ago + Uncommitted changes
23     </script>
24     <script id="locate-reuse-libs" src="./utils/locate-reuse-libs.js"
25         data-sap-ui-manifest-uri=".manifest.json"
26         data-sap-ui-componentName="at.clouddna.training00.zhoui5">
27     </script>
28 </head>

```

A red box highlights the line 'data-sap-ui-loglevel="debug"'. The status bar at the bottom of the editor shows 'You a few seconds ago + Uncommitted changes'.

1. Debugging in der index.html aktivieren

```
data-sap-ui-loglevel="debug"
```

The screenshot shows a code editor with several tabs at the top: Customer.controller.js, ChangeCustomer.fragment.xml, BaseController.js (which is the active tab), index.html, and Main.controller.js. The BaseController.js code is as follows:

```
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/core/routing/History",
4   "sap/base/Log"
5 ],
6 function (Controller, History, Log) {
7   "use strict";
8
9   return Controller.extend("at.clouddna.training00.zhoui5.controller.BaseController", {
10
11     _sContentDensityClass: "",
12
13     _getContentDensity: function(){
14       if(!this._sContentDensityClass){
15         if(sap.ui.Device.support.touch){
16           this._sContentDensityClass = "sapUiSizeCozy";
17         }else{
18           this._sContentDensityClass = "sapUiSizeCompact";
19         }
20       }
21
22       return this._sContentDensityClass;
23     },
24
25     setContentDensity: function(){
26       this.getView().addStyleClass(this._getContentDensity());
27     },
28
29     getRouter: function(){
30       return this.getOwnerComponent().getRouter();
31     },
32
33   });
34 }
```

Two sections of the code are highlighted with purple boxes and numbered circles:

- Annotation 1 highlights the `_getContentDensity` method, which checks if `_sContentDensityClass` is defined. If not, it uses the `sap.ui.Device.support.touch` property to determine the content density class: "sapUiSizeCozy" for touch devices and "sapUiSizeCompact" for others.
- Annotation 2 highlights the `setContentDensity` method, which adds the style class determined by `_getContentDensity` to the view.

1. Content-Density verwalten

2. Funktion für Router anlegen

```
_sContentDensityClass: "",

_getContentDensity: function () {
  if (!this._sContentDensityClass) {
    if (sap.ui.Device.support.touch) {
      this._sContentDensityClass = "sapUiSizeCozy";
    } else {
      this._sContentDensityClass = "sapUiSizeCompact";
    }
  }
}
```

```

        return this._sContentDensityClass;
    },

setContentDensity: function () {
    this.getView().addStyleClass(this._getContentDensity());
},
getRouter: function () {
    return this.getOwnerComponent().getRouter();
},

```

Customer.controller.js .../zhoui5/webapp/controller ChangeCustomer.fragment.xml .../webapp/view/fragment BaseController.js X index.html Main.controller.js .../zhoui5/ ⓘ ⌂ ⌃

webapp > controller > BaseController.js > ...

```

1  sap.ui.define([
2      "sap/ui/core/mvc/Controller",
3      "sap/ui/core/routing/History",
4      "sap/base/Log"
5  ],
6  function (Controller, History, Log) {
7      "use strict";
8
9      return Controller.extend("at.clouddna.training00.zhoui5.controller.BaseController", {
10
11         _sContentDensityClass: "",
12
13         getModel: function(sName){
14             return this.getView().getModel(sName);
15         },
16
17         setModel: function(oModel, sName){
18             return this.getView().setModel(oModel, sName);
19         },
20

```

1. Logik für das Laden und Setzen eines Models implementieren

```

getModel: function (sName) {
    return this.getView().getModel(sName);
},
setModel: function (oModel, sName) {
    return this.getView().setModel(oModel, sName);
},

```



```
Customer.controller.js .../zhoui5/webapp/controller ChangeCustomer.fragment.xml .../webapp/view/fragment BaseController.js x index.html Main.controller.js .../zhoui5/ ...
1 sap.ui.define([
2   "sap/ui/core/mvc/Controller",
3   "sap/ui/core/routing/History",
4   "sap/base/Log"
5 ],
6   function (Controller, History, Log) {
7     "use strict";
8
9     return Controller.extend("at.clouddna.training00.zhoui5.controller.BaseController", {
10       _sContentDensityClass: "",
11
12       getLocalizedText: function(sId, aParams){
13         let oBundle = this.getOwnerComponent().getModel("i18n").getResourceBundle();
14         return oBundle.getText(sId, aParams);
15       },
16
17     });
18   }
19 }
```

1. Funktion für das Lesen der übersetzbaren Texte implementieren

```
getLocalizedText: function (sId, aParams) {
  let oBundle = this.getOwnerComponent().getModel("i18n").getResourceBundle();
  return oBundle.getText(sId, aParams);
},
```

Anpassungen in den Controllern

Der BaseController gehört in den jeweiligen Controllern eingefügt und erweitert. Die definierten Funktionen im BaseController können danach in den jeweiligen Controllern wiederverwendet werden.

webapp > controller > **JS** Main.controller.js > ...

Maximilian Olzinger, 5 days ago | 2 authors (You and others)

```
1 sap.ui.define([ "at/clouddna/training00/zhoui5/controller/BaseController",  
2   "sap/m/MessageBox",  
3   "at/clouddna/training00/zhoui5/controller/formatter/HOUISFormatter"  
4 ],  
5   /**  
6    * @param {typeof sap.ui.core.mvc.Controller} Controller  
7    */  
8   function BaseController, MessageBox, HOUISFormatter) {  
9     "use strict";  
10  
11     return BaseController.extend("at.clouddna.training00.zhoui5.controller.Main", {  
12       ...HOUISFormatter,  
13  
14       onInit: function () {  
15         this.setContentDensity();  
16       }  
17     });  
18   }
```

1. BaseController importieren (muss erster import in der Liste sein!)
2. Import in ein Parameter speichern
3. BaseController mit dem eigenen Controller erweitern
4. In der onInit-Funktion Content-Density setzen

```

_delete: function(oListItem){
    let oModel = this.getView().getModel();
    //let oResourceBundle = this.getView().getModel("i18n").getResourceBundle();
    let sPath = oListItem.getBindingContext().getPath();

    MessageBox.warning(oResourceBundle.getText("sureToDeleteQuestion"), {
        title: oResourceBundle.getText("sureToDeleteTitle"),
        actions: [MessageBox.Action.YES, MessageBox.Action.NO],
        emphasizedAction: MessageBox.Action.YES,
        onClose: function(oAction){
            if(MessageBox.Action.YES === oAction){
                oModel.remove(sPath, {
                    success: (oData, response) => {
                        MessageBox.success(this.getLocalizedMessage("dialog.delete.success"));
                        oModel.refresh(true);
                    },
                    error: (oError) => {
                        MessageBox.error(oError.message);
                    }
                });
            }
        }
    });
},
onListItemClicked: function(oEvent) {
    let path = oEvent.getSource().getBindingContext().getPath().substring(1);
    this.getRouter().navTo("RouteCustomer", {path: path});
}

```

1

2

3

1. Variablendeclaration für ResourceBundle auskommentieren
2. Übersetzbaren Texte vom BaseController liefern lassen
3. Variablendeclaration für Router auskommentieren und den Router direkt laden

```

webapp > controller > JS Customer.controller.js > ...
You, 11 minutes ago | 2 authors (You and others)
1 sap.ui.define(["at/clouddna/training00/zhoui5/controller/BaseController",
2   "sap/ui/model/json/JSONModel",
3   "sap/ui/core/Fragment",
4   "sap/m/MessageBox",
5   "at/clouddna/training00/zhoui5/controller/formatter/HOUISFormatter",
6   "sap/ui/core/Item"
7 ],
8 /**
9  * @param {typeof sap.ui.core.mvc.Controller} Controller
10 */
11 function (BaseController, JSONModel, Fragment, MessageBox, HOUISFormatter, Item) {
12   "use strict";
13
14   return BaseController.extend("at.clouddna.training00.zhoui5.controller.Customer", {
15     formatter: HOUISFormatter,
16     _fragmentList: {},
17     bCreate: false,
18
19     /**
20      * @param {sap.ui.core.mvc.Controller} Controller
21      */
22     /**
23      * @param {sap.ui.core.mvc.Controller} Controller
24      */
25     /**
26      * @param {sap.ui.core.mvc.Controller} Controller
27      */
28     /**
29      * @param {sap.ui.core.mvc.Controller} Controller
30      */
31     /**
32      * @param {sap.ui.core.mvc.Controller} Controller
33      */
34     /**
35      * @param {sap.ui.core.mvc.Controller} Controller
36      */
37     /**
38      * @param {sap.ui.core.mvc.Controller} Controller
39      */
40     /**
41      * @param {sap.ui.core.mvc.Controller} Controller
42      */
43     /**
44      * @param {sap.ui.core.mvc.Controller} Controller
45      */
46     /**
47      * @param {sap.ui.core.mvc.Controller} Controller
48      */
49     /**
50      * @param {sap.ui.core.mvc.Controller} Controller
51      */
52     /**
53      * @param {sap.ui.core.mvc.Controller} Controller
54      */
55     /**
56      * @param {sap.ui.core.mvc.Controller} Controller
57      */
58     /**
59      * @param {sap.ui.core.mvc.Controller} Controller
60      */
61     /**
62      * @param {sap.ui.core.mvc.Controller} Controller
63      */
64     /**
65      * @param {sap.ui.core.mvc.Controller} Controller
66      */
67     /**
68      * @param {sap.ui.core.mvc.Controller} Controller
69      */
70     /**
71      * @param {sap.ui.core.mvc.Controller} Controller
72      */
73     /**
74      * @param {sap.ui.core.mvc.Controller} Controller
75      */
76     /**
77      * @param {sap.ui.core.mvc.Controller} Controller
78      */
79     /**
79      * @param {sap.ui.core.mvc.Controller} Controller
80      */
81     /**
81      * @param {sap.ui.core.mvc.Controller} Controller
82      */
83     /**
83      * @param {sap.ui.core.mvc.Controller} Controller
84      */
85     /**
85      * @param {sap.ui.core.mvc.Controller} Controller
86      */
86     /**
86      * @param {sap.ui.core.mvc.Controller} Controller
87      */
87     /**
87      * @param {sap.ui.core.mvc.Controller} Controller
88      */
88     /**
88      * @param {sap.ui.core.mvc.Controller} Controller
89      */
89     /**
89      * @param {sap.ui.core.mvc.Controller} Controller
90      */
90     /**
90      * @param {sap.ui.core.mvc.Controller} Controller
91      */
91     /**
91      * @param {sap.ui.core.mvc.Controller} Controller
92      */
92     /**
92      * @param {sap.ui.core.mvc.Controller} Controller
93      */
93     /**
93      * @param {sap.ui.core.mvc.Controller} Controller
94      */
94     /**
94      * @param {sap.ui.core.mvc.Controller} Controller
95      */
95     /**
95      * @param {sap.ui.core.mvc.Controller} Controller
96      */
96     /**
96      * @param {sap.ui.core.mvc.Controller} Controller
97      */
97     /**
97      * @param {sap.ui.core.mvc.Controller} Controller
98      */
98     /**
98      * @param {sap.ui.core.mvc.Controller} Controller
99      */
99     /**
99      * @param {sap.ui.core.mvc.Controller} Controller
100    */
101   });
102 }

```

1. BaseController importieren
2. Import in ein Parameter speichern
3. BaseController mit dem eigenen Controller erweitern
4. In der `onInit`-Funktion Content-Density setzen
5. In der `onInit`-Funktion die Router direkt laden

10.3 Zusammenfassung

 Martin Koch

In dieser Übung wurde der "BaseController" implementiert. In diesem Controller wurden Funktionen wie Navigation, Setzen/Abfragen von Modellen, Logging, Routing, Content Density und i18n-Texte ausgelagert. Zudem wurde ein eigenständiges Modul für die Formatierung definiert, das an verschiedenen Stellen eingebunden und wiederverwendet werden kann.

11 Einleitung

 Martin Koch

In der Applikation ist geplant, die bestehende Tabelle durch eine SmartTable zu ersetzen. Hierfür werden lokale Annotations genutzt. Zusätzlich erfolgt eine Anpassung der Views, um den Fiori Design Guidelines zu entsprechen.

Verwendete Controls: SmartTable , SmartFilterBar

Verwendete Technologien: Smart Controls, UI Annotations (Lokale Annotationen)

11.1 Grundlagen Low-Code-Entwicklung

MK

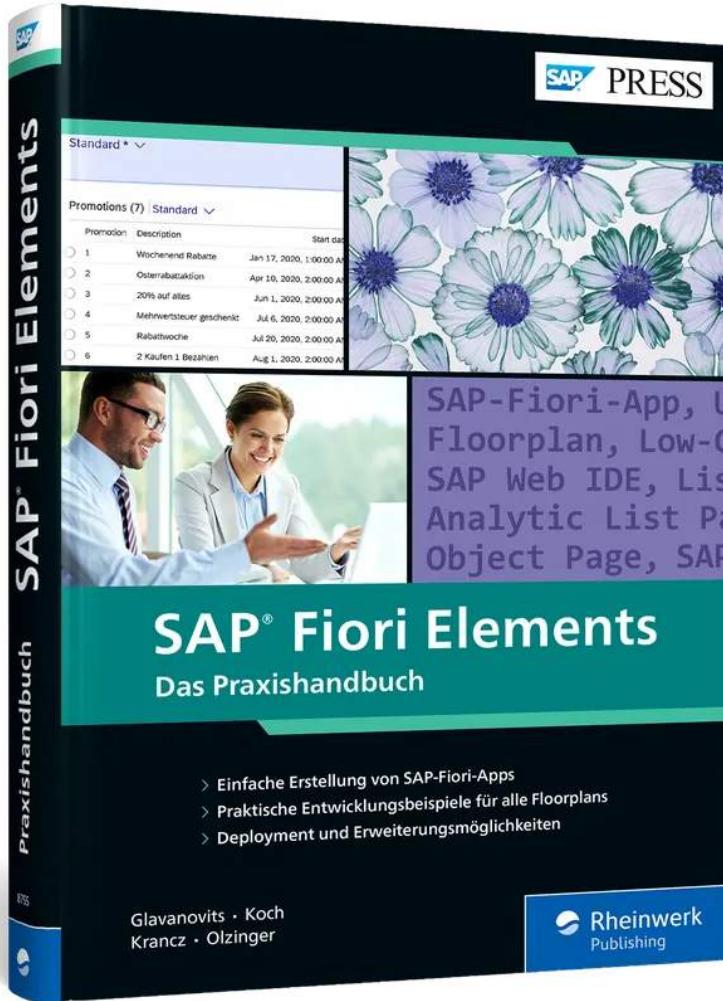
Martin Koch

Smart Controls

Mit der Version 1.30 des SAPUI5 Frameworks wurden die Smart Controls eingeführt. Sie ermöglichen Ihnen den Aufbau und das Verhalten von Tabellen, Formularen und Charts vollständig über Metadaten zu steuern, ohne eine einzige Zeile JavaScript zu entwickeln. Smart Controls interpretieren die OData-Metadaten. In bestimmten Fällen persistieren Smart Control die vom Benutzer*in personalisierte Version der Benutzeroberfläche. Die Metadaten werden in Form von Annotationen bereitgestellt.

SAP Fiori Elements

SAP Fiori Elements ist ein Framework, das die am häufigsten verwendeten sogenannten Floorplans umfasst. Es ist einerseits darauf ausgelegt, die Entwicklung signifikant zu beschleunigen und andererseits eine konsistente User Experience zu gewährleisten. SAP Fiori Elements bietet Ihnen dennoch eine hohe Flexibilität. Eine Konfiguration der Anwendung ist in allen Fällen zwingend erforderlich. Zusätzlich haben Sie die Möglichkeit, die erforderlichen Annotation entweder im Backend oder innerhalb der Anwendung zu erstellen. SAP Fiori Elements Anwendungen können unter Verwendung von SAP-Standardtechnologien erweitert werden. Damit können Sie beispielsweise eine Basisapplikation erstellen und für bestimmte Länder oder Geschäftsbereiche abweichende Anforderungen in eigene Anwendungen abdecken, die diese Anforderungen abbilden. In der objektorientierten Programmierung würden man an dieser Stelle von Vererbung sprechen.



<https://www.rheinwerk-verlag.de/sap-fiori-elements-das-praxishandbuch/>

SAP Fiori Elements - Das Praxishandbuch

Erstellen Sie SAP-Fiori-Apps ganz einfach und ohne JavaScript-Programmierung! Dieses Buch zeigt Ihnen, wie Sie sich den Low-Code-Ansatz von SAP Fiori Elements zunutze machen, um Entwicklungszeiten und -aufwand zu reduzieren. In praktischen Beispielen werden Sie durch die Entwicklung von Übersichts- und Detailseiten, analytischen Anwendungen und Objektseiten geführt. Und wenn doch noch individuelle Anpassungen an Ihren Apps erforderlich sind, finden Sie auch dazu hilfreiche Tipps und Anleitungen.

- Einfache Erstellung von SAP-Fiori-Apps
- Praktische Entwicklungsbeispiele für alle Floorplans

- Deployment und Erweiterungsmöglichkeiten

11.2 SmartTable mit Remote-Annotationen

MK

Martin Koch

In diesem Kapitel werden wir die grundlegendste Variante von SmartTable und SmartFilterBar implementieren. Dabei werden wir uns auf die einfache Konfiguration und Verwendung dieser SAPUI5-Steuerelemente konzentrieren, um eine effiziente Tabellendarstellung in Kombination mit Filtermöglichkeiten zu ermöglichen.

sap.ui.comp.smarttable.SmartTable

Eine [SmartTable](#) generiert eine Tabelle basierend auf den OData-Metadaten, die mit Hilfe eines Annotation-Files geregelt werden. Sie kann mit eigenen Spalten, Header, etc... erweitert werden. Die Basis-Columns und -Rows kommen jedoch aus dem OData-Service und über ein Annotation-File wird gesteuert, welche Columns wie angezeigt werden.

Most used Aggregations:

- customToolbar – Toolbar für eigene Funktionalitäten.
- customData – Wird benutzt, um spezifische Daten hinzuzufügen.

Most used Properties:

- entitySet – EntitySet aus dem ODataService.
- smartFilterId – Gibt eine dazugehörige SmartFilterBar an.

- tableType – ResponsiveTable, TreeTable, ...

sap.ui.comp.smartfilterbar.SmartFilterBar

Die [SmartFilterBar](#) legt aufgrund der OData-Metadaten Filtereingabefelder an.

Voraussetzung ist, dass die Property des OData-Attributs sap:filterable=true existiert.

```
webapp > view > Main.view.xml
You, 2 minutes ago | 1 author (You)
1  <mvc:View controllerName="at.cloudna.training00.zhoui5.controller.Main"
2    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
3    xmlns:s="sap.f.semantic"
4    xmlns:core="sap.ui.core"
5    xmlns="sap.m"
6    xmlns:smartTable="sap.ui.comp.smarttable"
7    xmlns:smartFilterBar="sap.ui.comp.smartfilterbar">
8    <s:SemanticPage id="main_page" showFooter="false">
9      <s:titleHeading>
10        <title text="{i18n>main.page.title}" />
11      </s:titleHeading>
12      <s:content>
13        <VBox fitContainer="true">
14          <smartFilterBar:SmartFilterBar id="main_smartFilterBar" entitySet="Z_P_CUSTOMER" persistencyKey="SmartFilter_Explored"/>1
15
16          <smartTable:SmartTable entitySet="Z_P_CUSTOMER" initiallyVisibleFields="Firstname,Lastname,Title,Phone" smartFilterId="main_smartFilterBar"
17            tableType="ResponsiveTable" enableExport="true" useVariantManagement="true" useTablePersonalisation="true" header="{i18n>main.table.title}"
18            showRowCount="true" enableAutoBinding="true" class="sapUiResponsiveContentPadding" enableAutoColumnWidth="true"/>2
19
20        </VBox>3
21      </s:content>
22    </s:SemanticPage>
23  </mvc:View>
```

You, 2 months ago + example 3

1. Inhalt des "content"-Feldes in der SemanticPage mit neuen Inhalt befüllen

2. Code für die "SmartFilterBar"

3. Code für die "SmartTable"

```
<mvc:View controllerName="at.clouddna.training00.zhoui5.controller.Main"
    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"
    xmlns:s="sap.f.semantic"
    xmlns:core="sap.ui.core"
    xmlns="sap.m"
    xmlns:smartTable="sap.ui.comp.smarttable"
    xmlns:smartFilterBar="sap.ui.comp.smartfilterbar">
  <s:SemanticPage id="main_page" showFooter="false">
    <s:titleHeading>
      <Title text="{i18n>main.page.title}" />
    </s:titleHeading>
    <s:content>
      <VBox fitContainer="true">
        <smartFilterBar:SmartFilterBar id="main_smartFilterBar" eni
          <smartTable:SmartTable entitySet="Z_P_CUSTOMER" initiallyV:
            tableType="ResponsiveTable" enableExport="true" useVariantL
            showRowCount="true" enableAutoBinding="true" class="sapUiRe
          </smartTable:SmartTable>
        </smartFilterBar:SmartFilterBar>
      </VBox>
    </s:content>
  </s:SemanticPage>
</mvc:View>
```

Kunden

Standard ▾

[Go](#) Hide Filter Bar Filters

To show filters here, add them to the filter bar in Filters

Kundentabelle (3)

Firstname	Lastname	Title	Phone
Maximilian	Mustermensch	Dr.	+43 676 123 123 124
Maxima	Mustermann	Mag.	+43 676 124 124 124
Maxi	Spaceman	Dr Dr	

Main-Page mit SmartTable & SmartFilterBar

11.3 Zusammenfassung

 Martin Koch

In dieser Übung wurden SmartControls verwendet. Die Main-View wurde auf eine SmartTable und SmartFilterBar umgebaut. Diese Controls bekommen ihre Daten basierend auf den OData-Metadaten und haben viele Funktionalitäten „out of the box“. Sie sind frei erweiterbar, jedoch müssen Annotations gesetzt werden, um die SmartControls zu definieren.

12 Einleitung

 Martin Koch

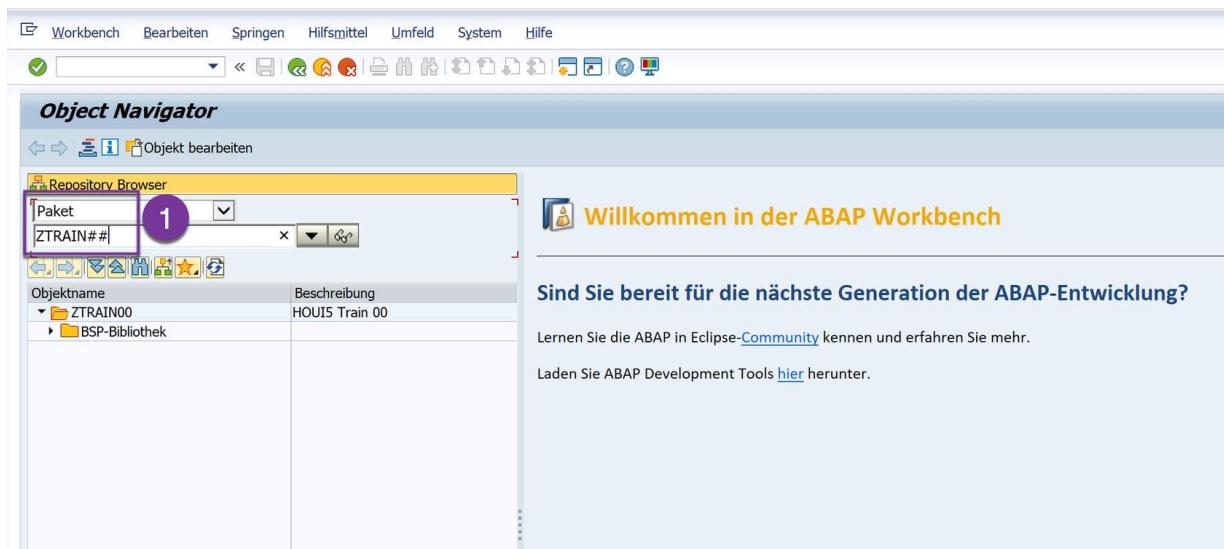
Deployen der Applikation in das ABAP System und starten der Applikation als Standalone App im ABAP System

Verwendete Controls: -

Verwendete Technologien: Cloud Connector

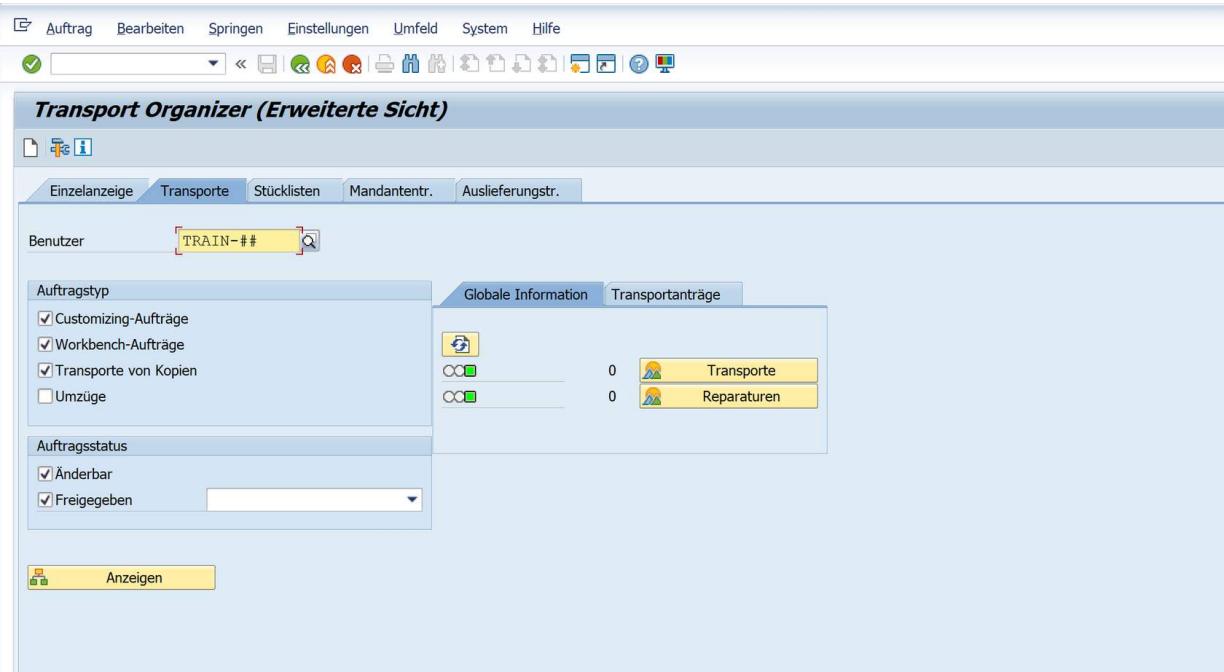
12.1 Deployment durchführen

MK Martin Koch



1. In der Transaktion SE80:

- a. Paket ZTRAIN##
- b. und Transportauftrag anlegen



1. Transportauftrag suchen und kopieren

In SAP Business Application Studio den Terminal starten.

```
~/projects/zhoui5 ×  
user: zhoui5 $ npm install
```

1. Node Packages installieren

```
~/projects/zhoui5 ×  
  
user: zhoui5 $ npx fiori add deploy-config  
Adding deploy-config to the project.  
info Add:Deploy-config Using: @sap/fiori:deploy-config  
? Please choose the target ABAP  
? Destination CloudDNA_WDF_FSD - http://s4dev.virtual:8080  
? Name (Maximum length: 15 characters) Z00HOUIS  
? Package ZTRAIN00  
? Transport Request S4DK900189  
? Generate standalone index.html during deployment Yes  
? Deployment Description HOUIS App Student00
```

1. Deploy-Config hinzufügen:

- a. Target **ABAP** auswählen
- b. Destination **CloudDNA_WDF_FSD** auswählen
- c. BSP-Namen vergeben: **ZHOUIS_##** (muss im Kundennamensraum sein!)
- d. Paket **ZTRAIN##** angeben
- e. **Transport** angeben
- f. **index.html** generieren lassen
- g. Deployment-Beschreibung hinzufügen

```
~/projects/zhoui5 ×  
  
user: zhoui5 $ npm run deploy █
```

1. Deployment starten

```
~/projects/zhoui5 ×

Confirmation is required to deploy the app:

Application Name: Z00HOUIS
Package: ZTRAIN00
Transport Request: S4DK900189
Destination: CloudDNA_WDF_FSD
SCP: false
Target System SAPUI5 version: 1.84.18

Target system's SAPUI5 version is lower than the local minUI5Version. Testing locally with different Run Configuration/17d50220bcd848aa854c9c182d65b699/Latest/en-US/09171c8bc3a64ec7848f0ef31770a793.html

? Start deployment (Y/n)?
> (Y/n)
```

1. Details prüfen und Deployment bestätigen

```
~/projects/zhoui5 ×

info builder:custom deploy-to-abap UPLOAD FILE  : controller/App.controller.js (Text)
info builder:custom deploy-to-abap UPLOAD FILE  : controller/BaseController-dbg.js (Text)
info builder:custom deploy-to-abap {
  name: at.cloudDNA.training00.zhoui5,
  version: 0.0.1,
  buildTimestamp: 202205130633,
  scmRevision: '',
  libraries: []
}
info builder:custom deploy-to-abap }
info builder:custom deploy-to-abap ... 49 messages omitted ...
info builder:custom deploy-to-abap * Updating the Application Index *
info builder:custom deploy-to-abap SAPUI5 Application has been uploaded and registered successfully
info builder:custom deploy-to-abap * Done *
info builder:custom deploy-to-abap App available at http://s4dev.virtual:8080/sap/bc/ui5_ui5/sap/z00houi5
info builder:custom deploy-to-abap Deployment Successful.
user: zhoui5 $ █
```

1. Fertigstellung bestätigen

The screenshot shows the SAP Web Application Builder interface. The title bar reads "Web Application Builder: BSP-Applikation anzeigen". The left side features a "Repository Browser" with a tree view of packages and application components. The selected item is "Z00HOU15 App Student00" under "BSP-Bibliothek > Z00HOU15". The main right panel displays the properties for the selected application. The "Eigenschaften" tab is active, showing the following details:

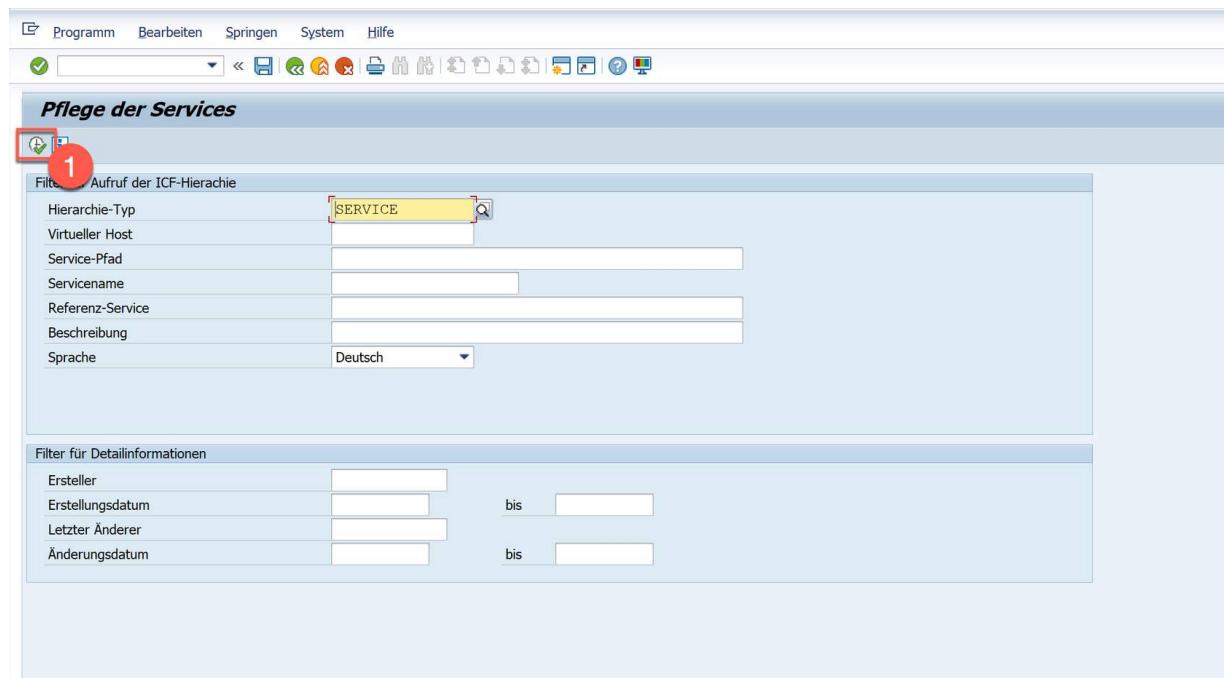
BSP-Applikation	Z00HOU15	aktiv	
Kurzbeschreibung	HOU15 App Student00		
Anleger	KRANCZ	Erstellungsdatum	13.05.2022
letzter Änderer	KRANCZ	Änderungsdatum	13.05.2022
Paket	ZTRAIN00		
Originalsprache	EN		
Applikation	Z00HOU15		

Below these fields are sections for "Einstiegs-BSP", "Anwendungsklasse" (/UI5/CL_UI5_BSP_APPLICATION), and "Thema". Under "Thema", there are three checkboxes: "Zustandsbehaftet" (unchecked), "Unterstützt Portal-Integration" (unchecked), and "XSRF-Schutz" (checked).

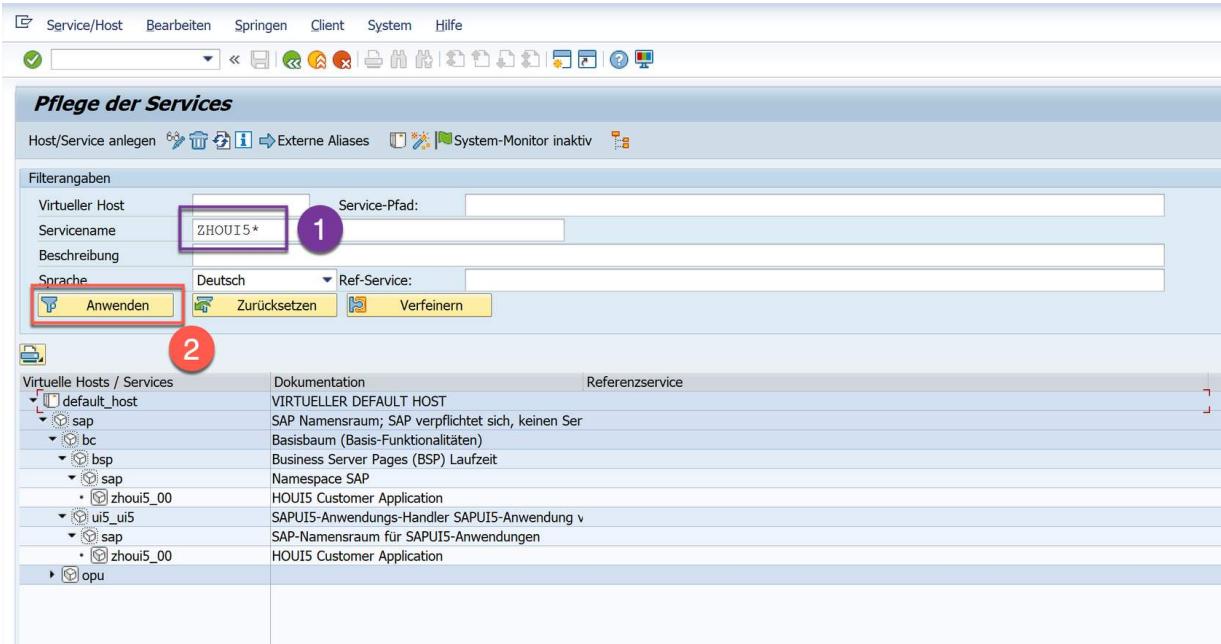
1. Paket mit dem neuen Inhalt prüfen

12.2 Applikation Standalone starten

MK Martin Koch



1. Transaktion SICF starten und Suche ausführen



1. Nach dem Knoten ZHOU15* suchen

2. Suche starten

SAP GUI - Pflege der Services

Host/Service anlegen Externe Aliases System-Monitor inaktiv

Filterangaben

Virtueller Host: ZHOU15* Service-Pfad:

Servicename: ZHOU15* Beschreibung: Sprache: Deutsch Ref-Service:

Anwenden Zurücksetzen Verfeinern

Virtuelle Hosts / Services

- default_host
 - sap
 - bc
 - bsp
 - zou15_00
 - ui5_ui5
 - sap
 - zou15_00
 - opu

Dokumentation: VIRTUELLE DEFAULT HOST
Referenzservice: SAP-Namensraum; SAP verpflichtet sich, keinen Ser Basisbaum (Basis-Funktionalitäten)
Business Server Pages (BSP) Laufzeit Namespace SAP
HOU15 Customer Application SAPUI5-Anwendungs-Handler SAPUI5-Anwendung v SAP-Namensraum für SAPUI5-Anwendungen HOU15 Customer Application

Neues Subelement
Service Anzeigen
Service Löschen
Service Umbenennen
Service aktivieren
Service deaktivieren **1** **Service testen**
Referenzen auf Service
Objektkatalogeintrag
Ausschneiden
Kopieren
Einsetzen

1. Service testen

12.3 Zusammenfassung

MK

Martin Koch

In dieser Übung wurde die Applikation in ein ABAP Repository deployed. Die Applikation kann nach ausgewähltem Paket einem Transportauftrag zugewiesen werden und so in ein ABAP System geladen werden. In der nächsten Übung wird die Applikation in das SAP Fiori Launchpad gehängt.

13 Einleitung

MK

Martin Koch

In dieser Übung wird die Applikation in das SAP Fiori Launchpad mittels eines Kacheln zur Verfügung gestellt. Eine Kachel kann mittels eines Clicks genutzt werden damit, die Applikation dahinter gestartet wird. Kacheln werden in Katalogen ausgeliefert. Zusätzlich können diese in Gruppen zusammengefasst und als eine logische Einheit abgebildet werden. Kacheln aus Katalogen müssen manuell im App Finder auf die Startseite gepinnt werden, hingegen Gruppen automatisch sichtbar werden. Beide Objekte erfordern die Zuweisung der entsprechenden Rolle.

Verwendete Controls: -

Verwendete Technologien: -

Verwendete Transaktionen:

- SICF – Service-Definitionen
- SE80 – Object Navigator
- /UI2/FLPD_CUST – Launchpad Designer
- UI2/FLP – Fiori Launchpad
- PFCG – Role Maintenance

Fiori Launchpad Content Manager

Anstelle der Transaktion /UI2/FLPD_CUST können Sie auch folgende Transaktion verwenden:
/UI2/FLPCM_CUST

Der Vorteil hierbei ist, dass Sie einerseits bessere Such- und Filtermöglichkeiten haben und andererseits bei einer Änderung den bekannten Dialog zum Customizing-Auftrag bekommen.

Fiori Launchpad Checks and Intent Analysis

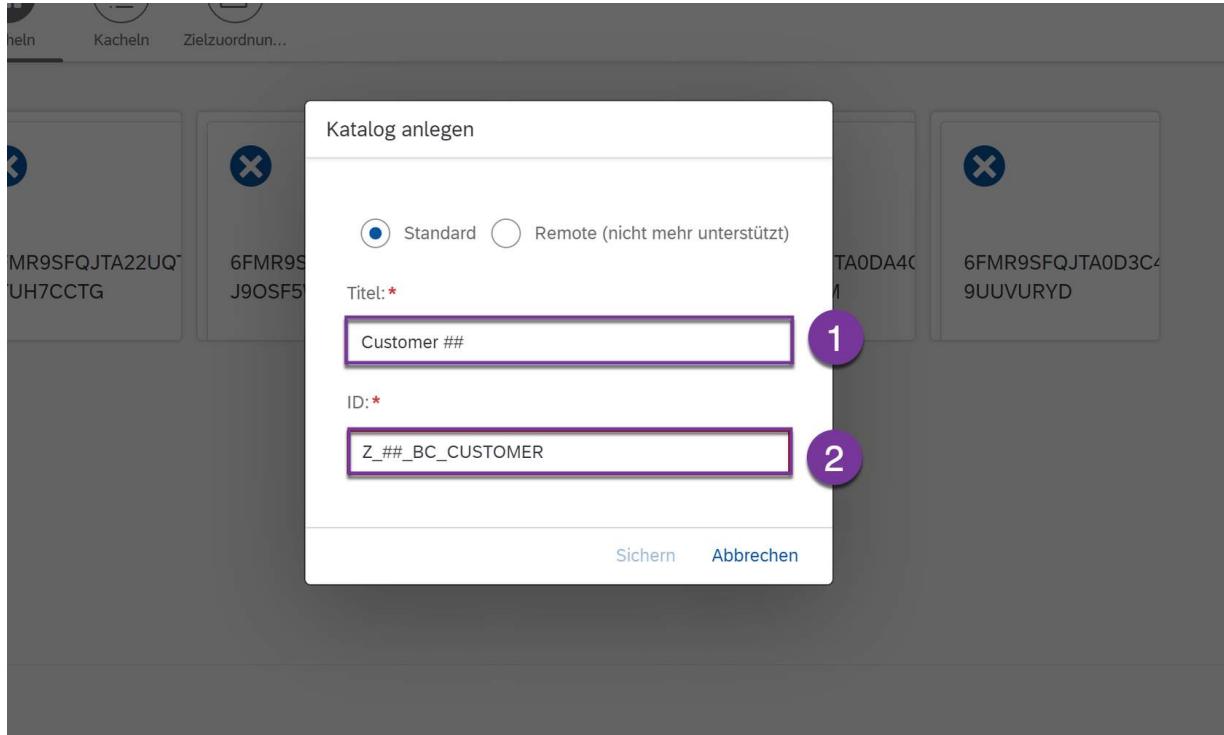
Die folgenden zwei Transaktionen (/UI2/FLIA und /UI2/FLC) können Ihnen einerseits dabei helfen, herauszufinden, ob Benutzer*innen die richtige Rolle zugewiesen haben und bestimmte Zielzuordnungen sehen. Andererseits können einige Fehlermeldungen im Fiori Launchpad Umfeld angesehen werden, um herausfinden zu können, warum Benutzer*innen zum Beispiel die Applikation nicht starten können.

13.1 Katalog anlegen

MK Martin Koch

The screenshot shows the SAP Fiori Catalog creation interface. The URL is /sap/bc/ui5/sap/arsvc_upb_admn/main.html?scope=CUST&sap-client=010&sap-language=DE#/Catalog/X-SAP-UI2-CATALOGPAGE:SAP_TC_ISPSCA_TRM_COMMON. The page title is "SAP_TC_ISPSCA_TRM_COMMON". The top navigation bar includes "Kataloge" (selected), "Gruppen", "Suchen", and "Mandant: 010". On the left, there's a sidebar with a search field for "Kataloge suchen" containing "SAP_TC_ISPSCA_TRM_CO..." (10). Below it are lists of other catalogs: "SAP_TC_RFQ_ASSORTMENT...", "SAP_TC_SCM_MPE_COMM...", "SAP_FICA_BC_RECO_REQ", and "SAP_FICA_BC_RECO_KEY_DISP". At the bottom of the sidebar is a red-bordered button with a plus sign and a red circle containing the number 1. The main area displays four catalog items with "X" icons: "6FMR9SFQJTA22UQFTUH7CCTG", "6FMR9SFQJTA22UQJ9OSF5WT7", "Formularbündel", and "6FMR9SFQJTA0DA4CW5PQ3F38M". There are also buttons for "Kacheln" (5), "Kacheln" (5), and "Zielzuordnung" (5).

1. Transaktion **/UI2/FLPD_CUST** öffnen und Katalog anlegen



1. Titel **Customer ##** vergeben

2. Katalog ID vergeben: **Z_##_BC_CUSTOMER**

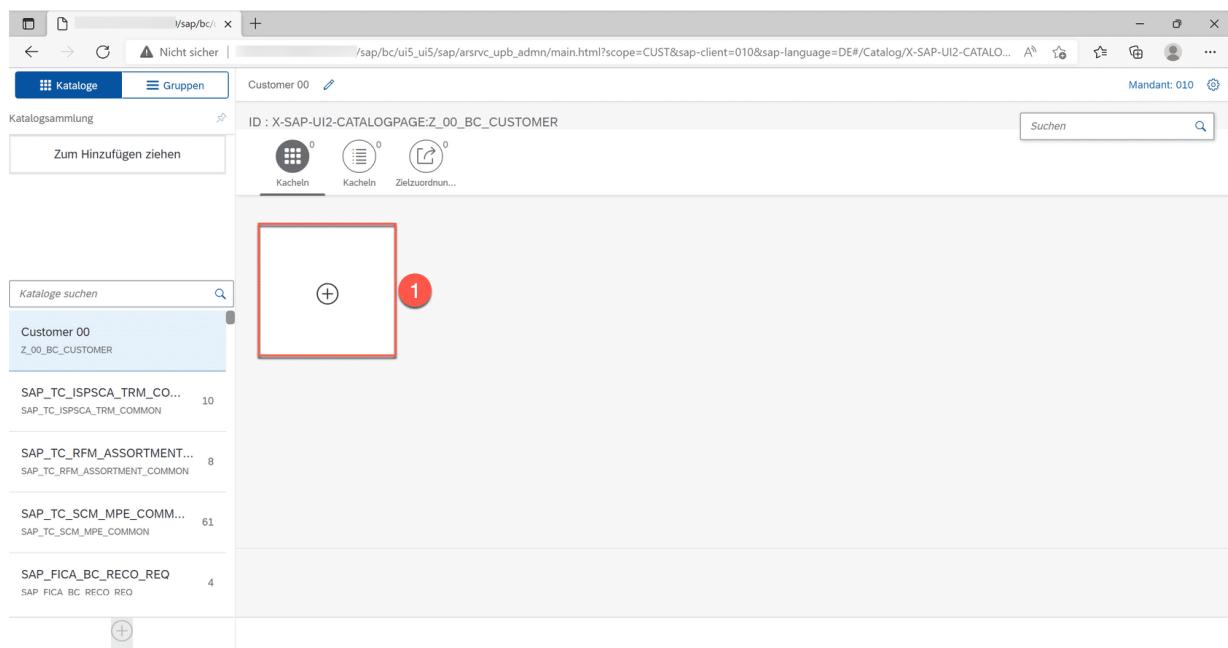
A screenshot of the SAP Fiori Catalog overview page. The top navigation bar shows the URL '/sap/bc/ui5/_ui5/sap/arsvc_upb_admn/main.html?scope=CUST&sap-client=010&sap-language=DE#/Catalog/X-SAP-UI2-CATALOG'. The main area displays a catalog entry for 'Customer 00' with ID 'Z_00_BC_CUSTOMER'. Below this, there is a search bar and a list of other catalog entries:

Katalog	Count
SAP_TC_ISPSCA_TRM_CO...	10
SAP_TC_ISPSCA_TRM_COMMON	
SAP_TC_RF_M_ASSIGNMENT...	8
SAP_TC_RF_M_ASSIGNMENT_COMMON	
SAP_TC_SCM_MPE_COMM...	61
SAP_TC_SCM_MPE_COMMON	
SAP_FICA_BC_RECO_REQ	4
SAP FICA BC RECO REQ	

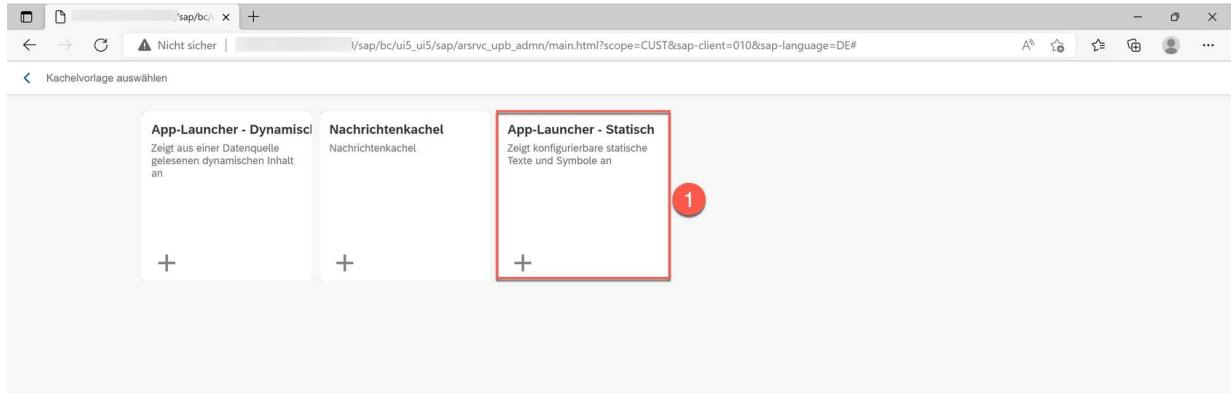
Ein leerer Katalog wurde angelegt

13.2 Kachel und Target-Mapping anlegen

MK Martin Koch



1. Kachel hinzufügen



1. Statische Kachel auswählen

The screenshot shows the configuration details for the selected static tile. The configuration fields are numbered as follows:

- Titel:** Manage Customers (Step 1)
- Untertitel:** Customer Overview (Step 2)
- Symbol:** sap-icon://Fiori2/F0004 (Step 3)
- Semantische Objektnavigation verwenden:** checked (Step 4)
- Semantisches Objekt:** Customer (Step 4)
- Aktion:** display## (Step 5)
- Hinzufügen** and **Entfernen** buttons (Step 6)
- Sichern** button (Step 6)

1. Titel vergeben

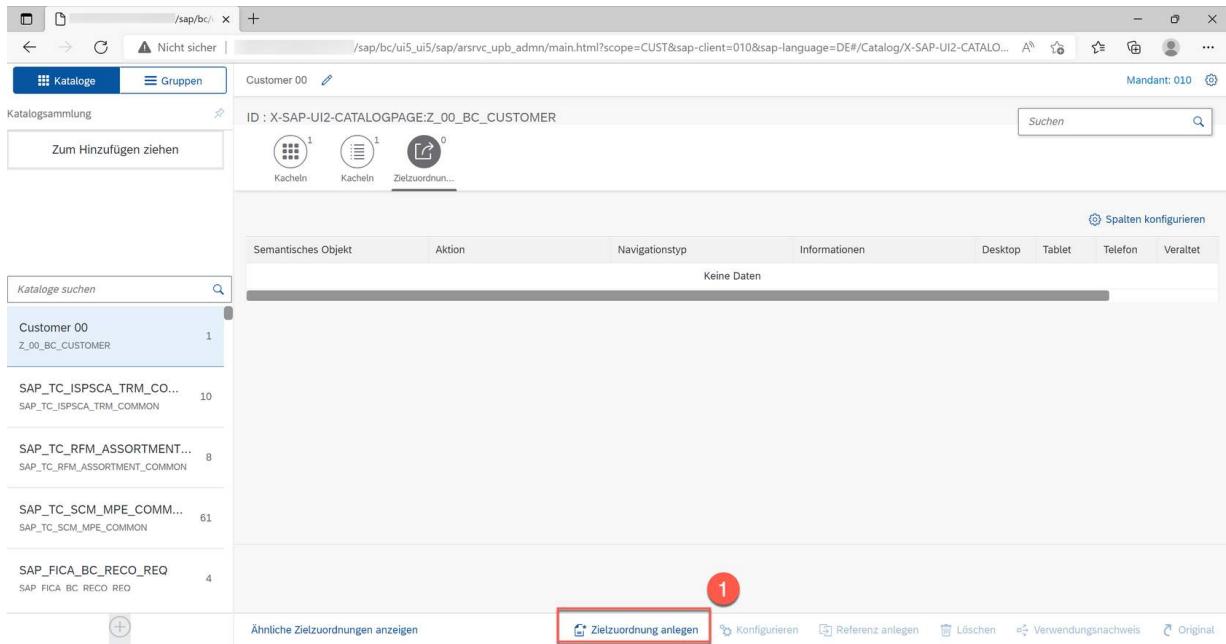
2. Untertitel vergeben

3. Icon vergeben

4. Semantisches Object **Customer** eintragen

5. Aktion **display##** eintragen

6. Sichern



1. Im dritten Tab Zielzuordnung (engl. Target Mapping) anlegen

The screenshot shows a code editor with tabs for 'Welcome', 'manifest.json', and 'ui5-deploy.yaml'. The 'manifest.json' tab is active, showing the following JSON code:

```
1 {  
2   "_version": "1.12.0",  
3   "sap.app": {  
4     "id": "at.cloudDNA.training00.zhoui5",  
5     "type": "application",  
6     "i18n": "i18n/i18n.properties",  
7     "applicationVersion": {  
8       "version": "0.0.1"  
9     },  
10    "title": "{{appTitle}}",  
11    "description": "{{appDescription}}",  
12    "resources": "resources.json",  
13    "sourceTemplate": {  
14      "id": "@sap-ux/fiori-freestyle-writer:basic",  
15      "version": "0.11.6"  
16    }  
17  }  
18}
```

A red box highlights the 'id' field in the 'sap.app' object.

1. Eindeutige ID der Applikation aus dem manifest.json kopieren

Konfiguration: "Ziel-Mapping"

Instanz-ID: B04BVCVUI5PM4XEC9Z1OGJ3D

Absicht	Ziel
Semantisches Objekt:	<input type="text" value="Customer"/> 1
Aktion:	<input type="text" value="display00"/> 2
Anwendungstyp:	<input type="text" value="SAPUI5-Fiori-App"/> 3
Titel:	<input type="text" value="Manage Customers"/> 4
URL:	<input type="text"/>
ID:	<input type="text" value="at.clouddna.training#0_zhoui5"/> 5

Allgemein

Informationen:

Gerätetypen: Desktop Tablet Telefon

Parameter:

Name	Obligatorisch	Wert	Ist regulärer Ausdruck	Standardwert	Zielname
<input type="checkbox"/> Namen einge...	<input type="checkbox"/>		<input type="checkbox"/>		

Zusätzliche Parameter zulassen

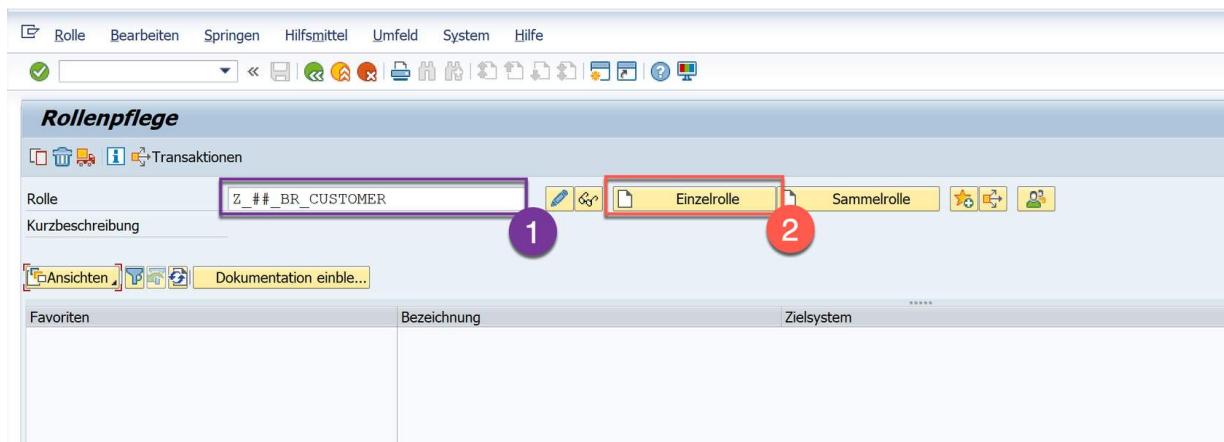
Hinzufügen Löschen 6

Sichern Abbrechen

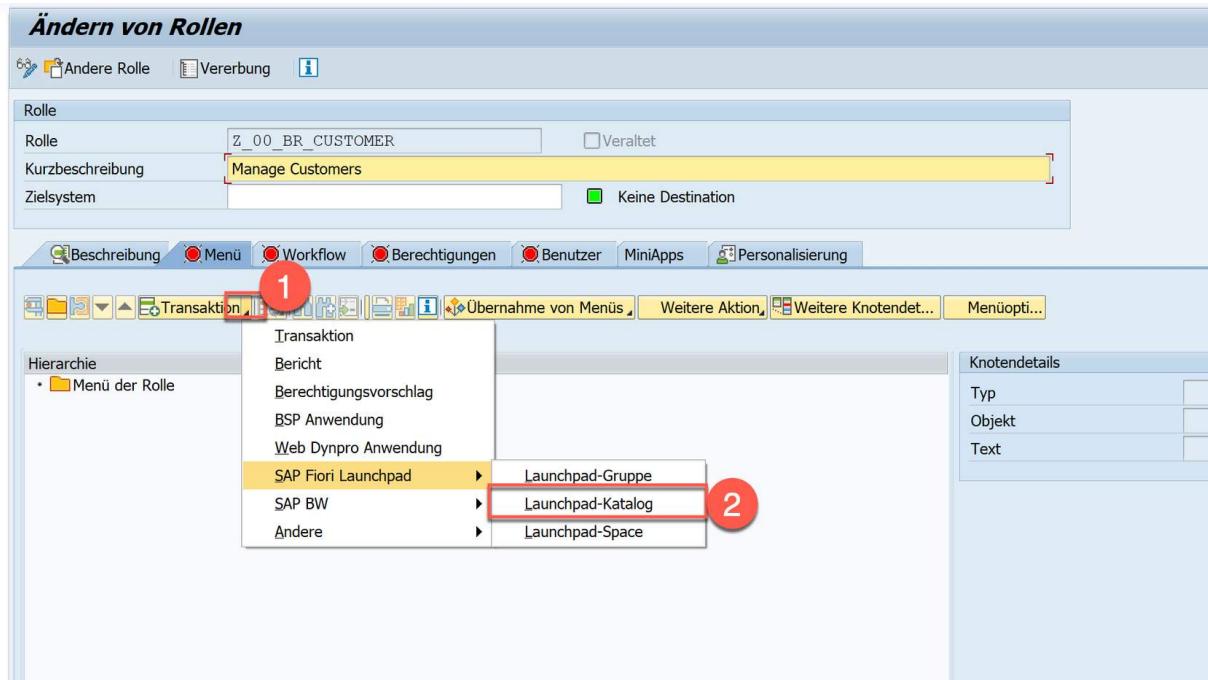
1. Semantisches Objekt **Customer** eintragen
2. Aktion **display##** eintragen
3. Sicherstellen, dass als Anwendungstyp SAPUI5-Fiori-App ausgewählt ist
4. Titel vergeben
5. Eindeutige ID der Applikation einfügen
6. **Sichern**

13.3 Rolle anlegen und Katalog zuweisen

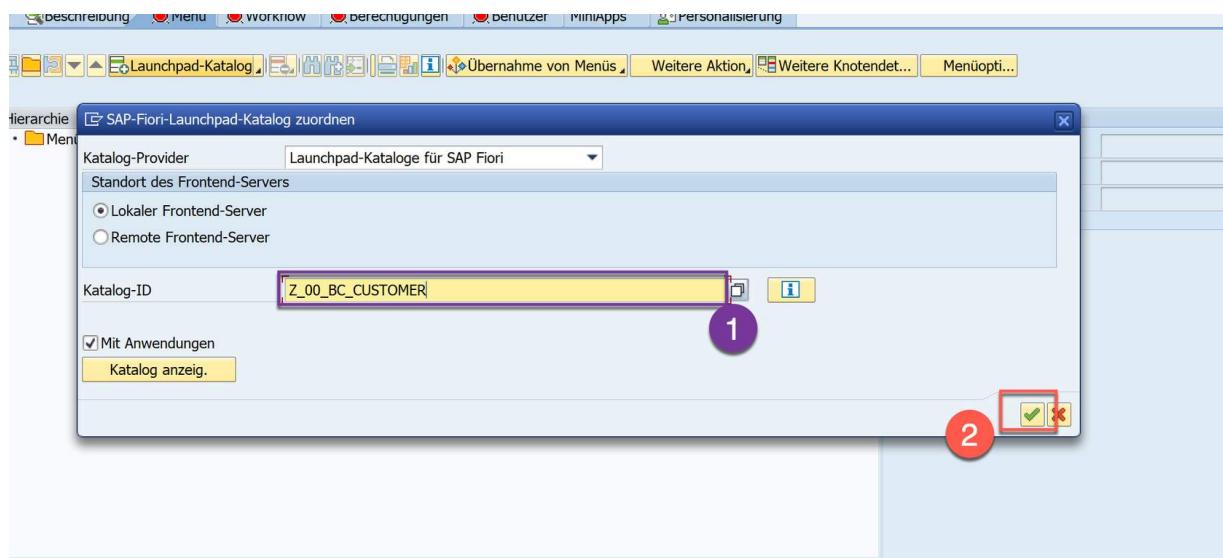
MK Martin Koch



1. In der Transaktion PFCG eine Rolle **Z##_BR_CUSTOMER** eintragen
2. **Einzelrolle** anlegen

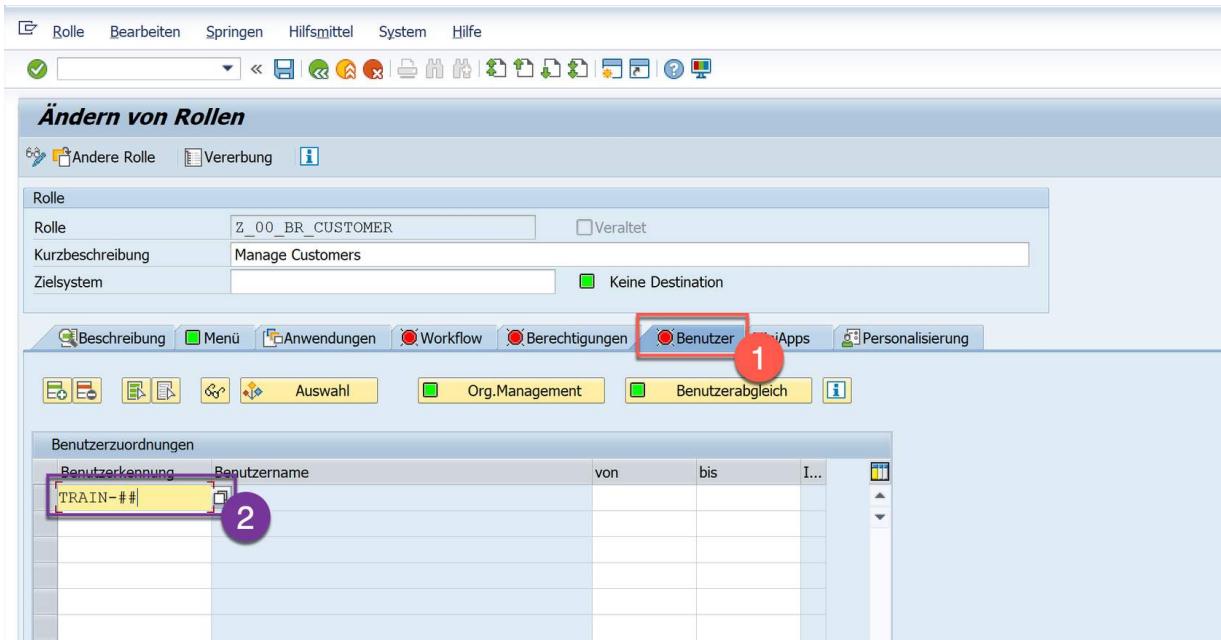


1. Im Menü weitere Elemente zum Hinzufügen auswählen
2. Unter SAP Fiori Launchpad **Launchpad-Katalog** auswählen



1. Eindeutige ID vom Katalog einfügen

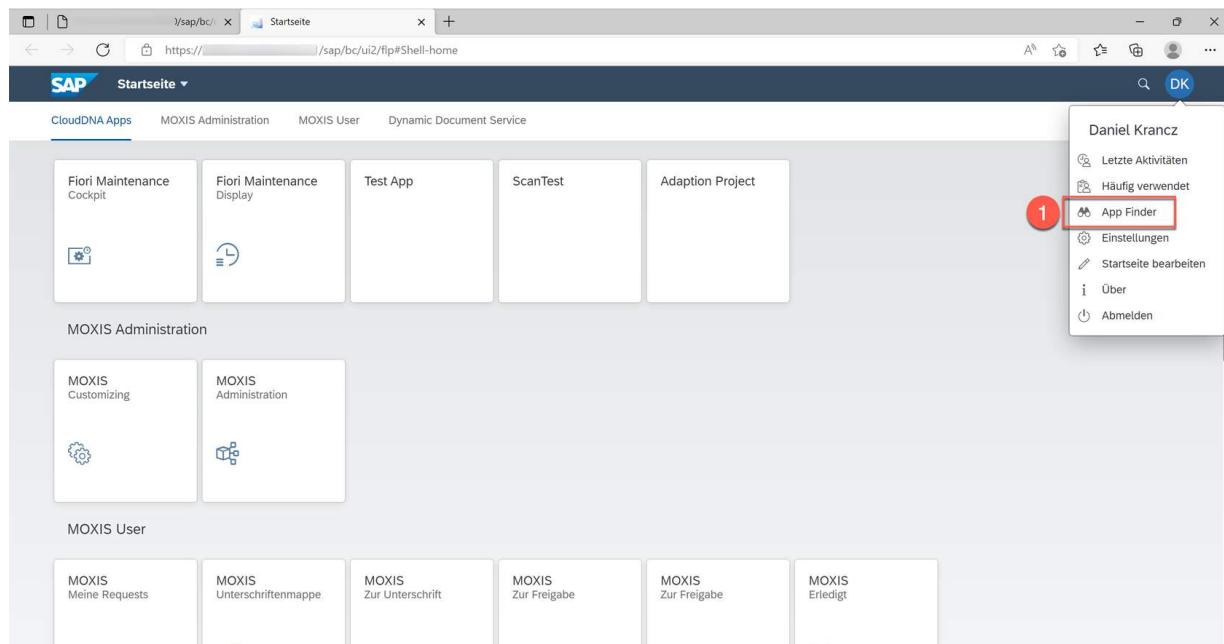
2. Sichern



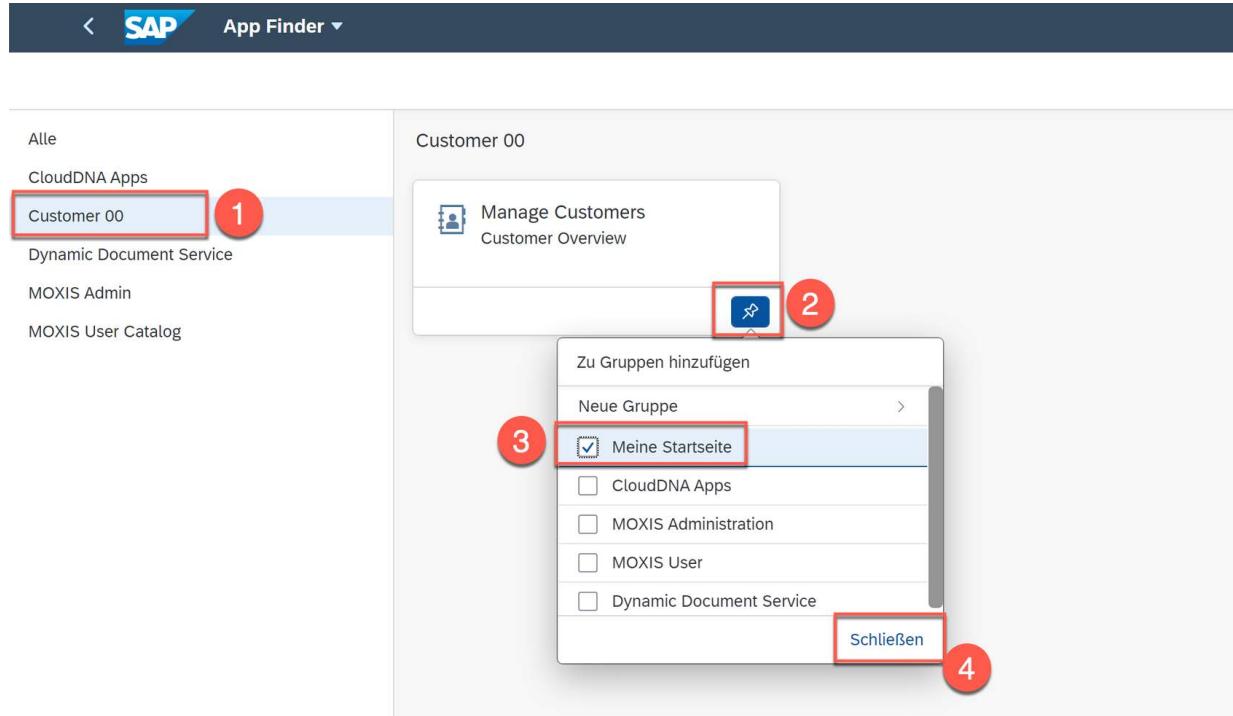
1. Zum Tab **Benutzer** wechseln
2. Benutzer **TRAIN-##** der Rolle hinzufügen

13.4 Kachel auf die Startseite pinnen

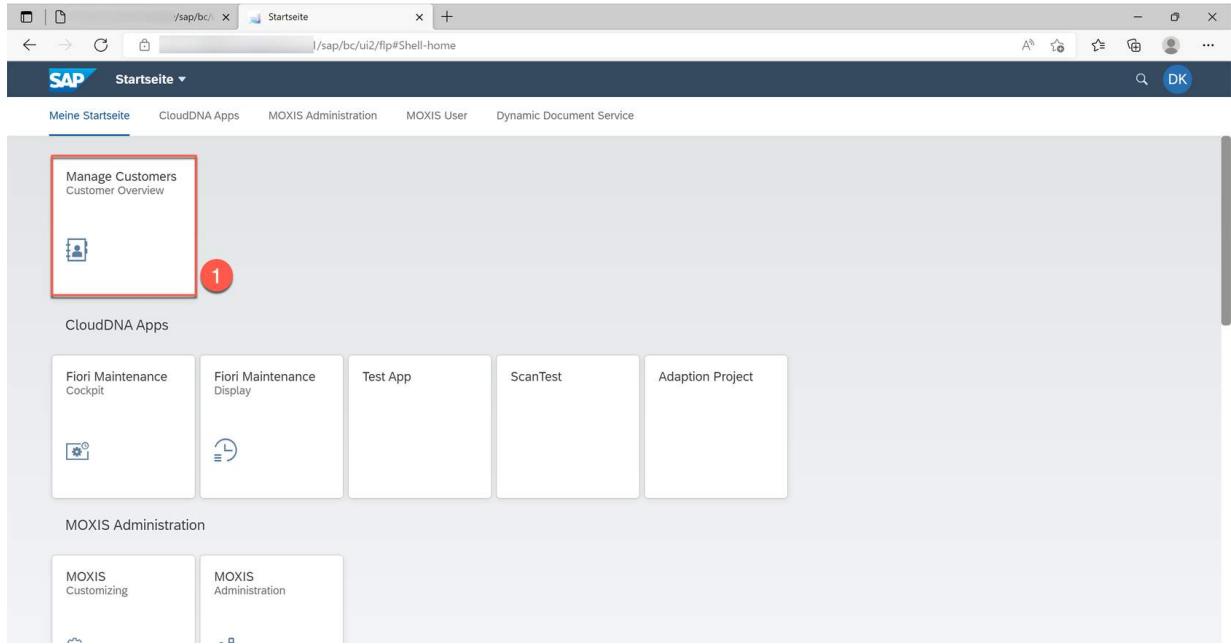
MK Martin Koch



1. Im SAP Fiori Launchpad (Transaktion /UI2/FLP) im Benutzermenü den **App Finder** starten



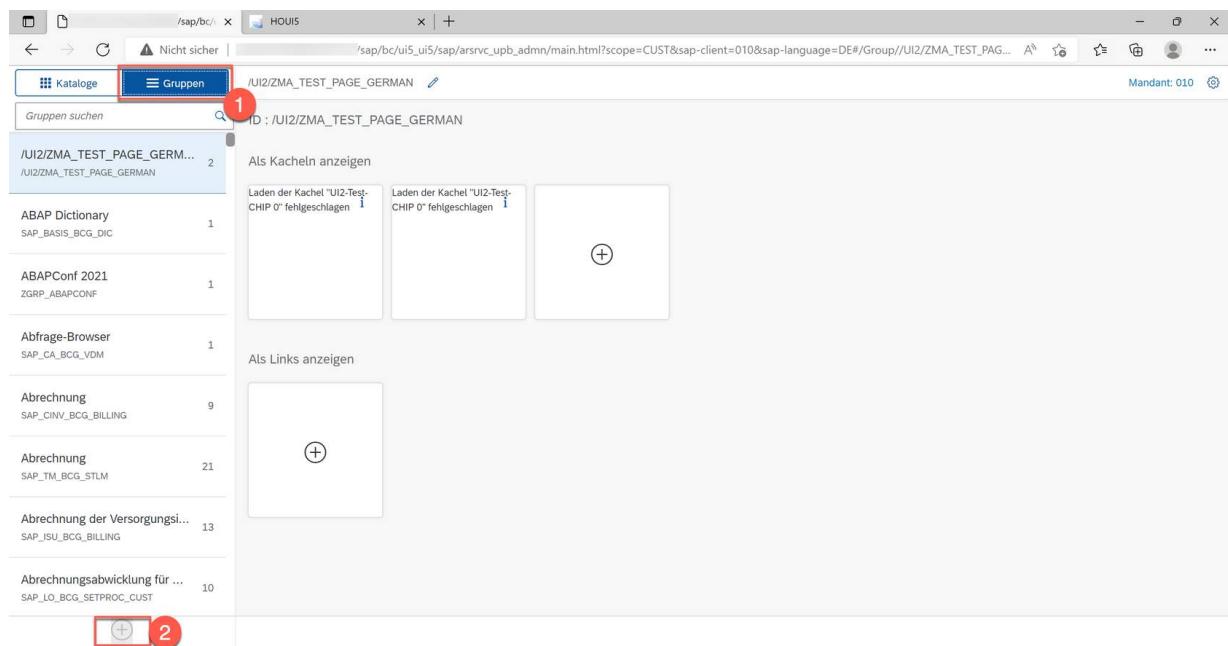
1. Katalog **Customer ##** auswählen
2. Toggle-Button betätigen
3. **Meine Startseite** (engl. My Home) auswählen
4. Dialog schließen



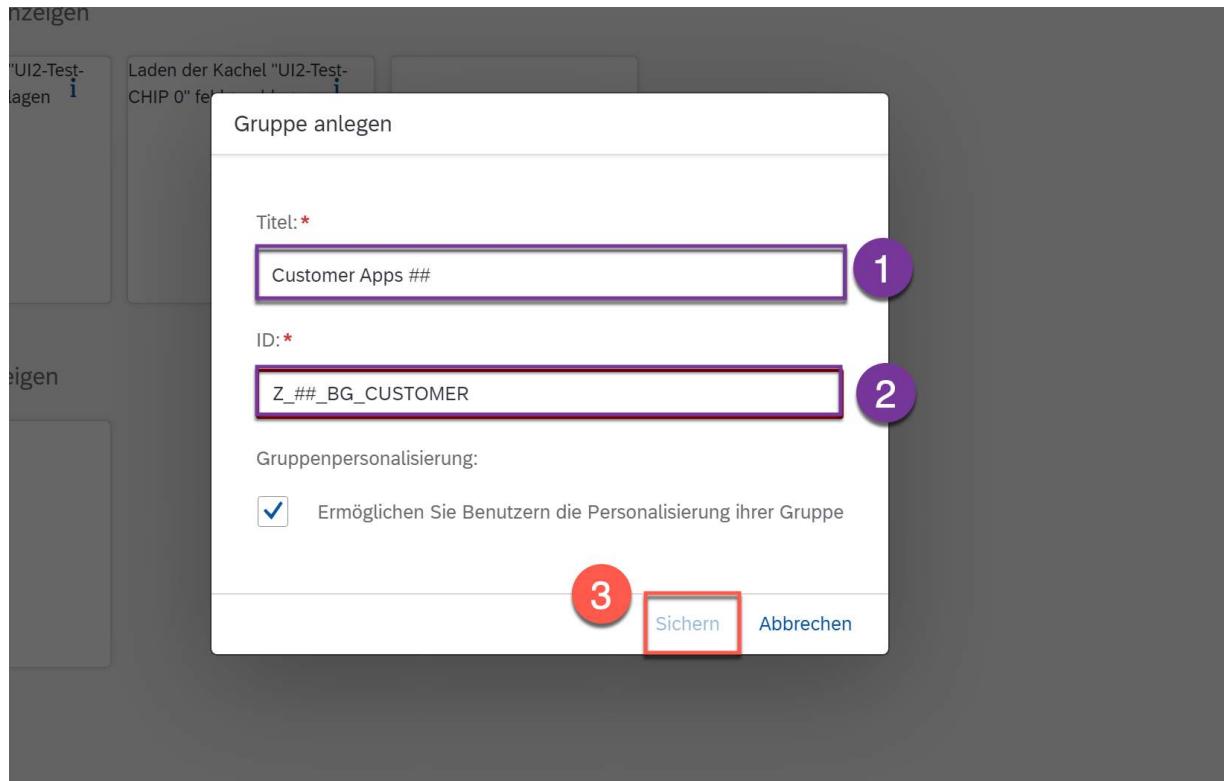
1. Kachel befindet sich nun auf der Startseite und die App kann gestartet werden

13.5 Gruppe anlegen und der Rolle zuweisen (optional)

MK Martin Koch



1. Zu den Gruppen wechseln
2. Eine Gruppe anlegen



1. Als Titel der Gruppe **Customer Apps ##** vergeben
2. Die ID der Gruppe soll **Z_##_BG_CUSTOMER** lauten
3. Sichern

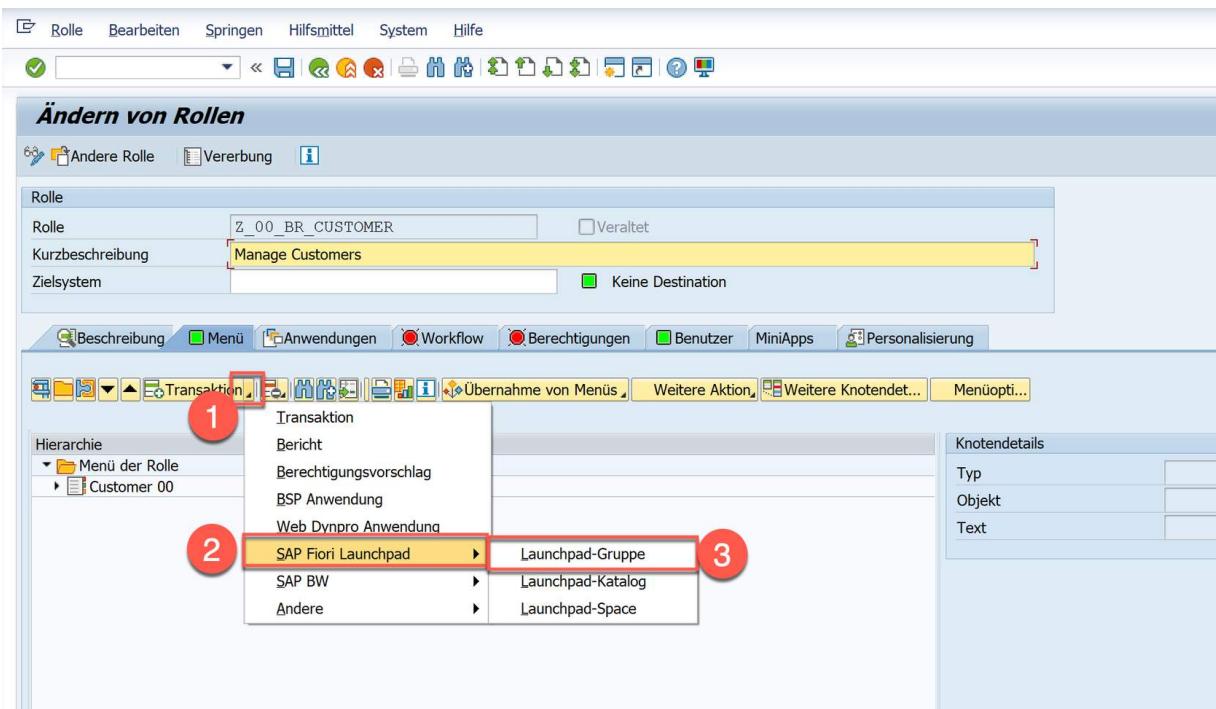
The screenshot shows the SAP Fiori Launchpad interface. On the left, there's a sidebar with various application icons and their names. In the center, under the heading 'Customer Apps 00' (ID: Z_00_BG_CUSTOMER), there are two sections: 'Als Kacheln anzeigen' (displayed) and 'Als Links anzeigen'. A red box highlights the 'Als Kacheln anzeigen' section, and a red circle with the number '1' is placed near the bottom right corner of this section.

1. Eine Kachel hinzufügen

The screenshot shows the configuration screen for the 'Customer Apps 00' group. At the top, it says 'Kachel an Gruppe "Customer Apps 00" anfügen'. Below this, a card labeled 'Customer 00' is selected, indicated by a purple border and a red circle with the number '1'. In the bottom right corner of the card, there is a red box containing a plus sign (+). A red circle with the number '2' is placed near this plus sign.

1. Katalog auswählen

2. Aus diesem Katalog die Kachel hinzufügen



1. Weitere Optionen

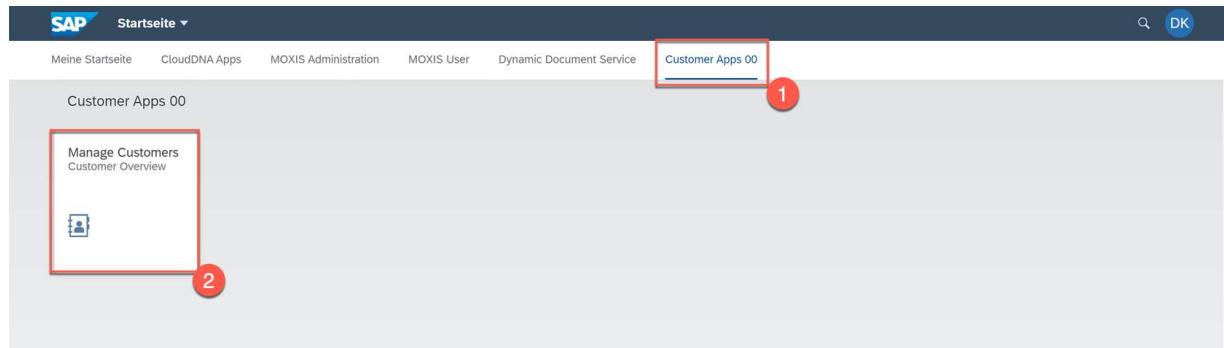
2. SAP Fiori Launchpad auswählen

3. Launchpad-Gruppe auswählen



1. ID der Gruppe eintragen

2. Sichern



Nach einem Refresh sollte die Kachel in einer separaten Gruppe automatisch sichtbar sein

13.7 Zusammenfassung

MK

Martin Koch

In dieser Übung wurde die Applikation über ein Launchpad im ABAP gestartet. Mit dem Launchpad-Designer wurde ein Katalog erstellt, der die Applikation in Form eines Tiles beinhaltet. Dieser Katalog wurde einer Gruppe zugewiesen, die den Katalog benutzt. Anschließend wurde eine neue Rolle angelegt, die auf diesen Katalog und diese Gruppe zugreifen darf. Beim Starten des Launchpads ist nun die erstellte Gruppe mit der eigenen Applikation ersichtlich, die über das Launchpad gestartet werden kann.

14 Einleitung

 Martin Koch

Die Applikation soll um einen "UploadSet" für Attachments erweitert werden. Dadurch wird Benutzern die Möglichkeit gegeben, Dateien hochzuladen und mit den entsprechenden Datensätzen zu verknüpfen. Dieser erweiterte Funktionsumfang ermöglicht eine verbesserte Handhabung von Anhängen und eröffnet neue Möglichkeiten für die Datenerfassung und -verwaltung innerhalb der Anwendung.

Verwendete Controls: UploadSet

Verwendete Technologien: Handling von Binärdaten

14.1 UploadSet implementieren

MK

Martin Koch

Um Dateien up- und down-zuladen gibt es mehrere Möglichkeiten. Die meistbenutzten UI-Controls sind **FileUploader** und **UploadSet**.

sap.ui.unified.FileUploader

Ein einfaches UI-Element, um einzelne Dateien hochzuladen.

Most used Properties:

- fileType – Gibt an, welche Files hochgeladen werden dürfen.
- uploadUrl – Url des Remtote-Servers
- value – Pfad der hochzuladenden Datei

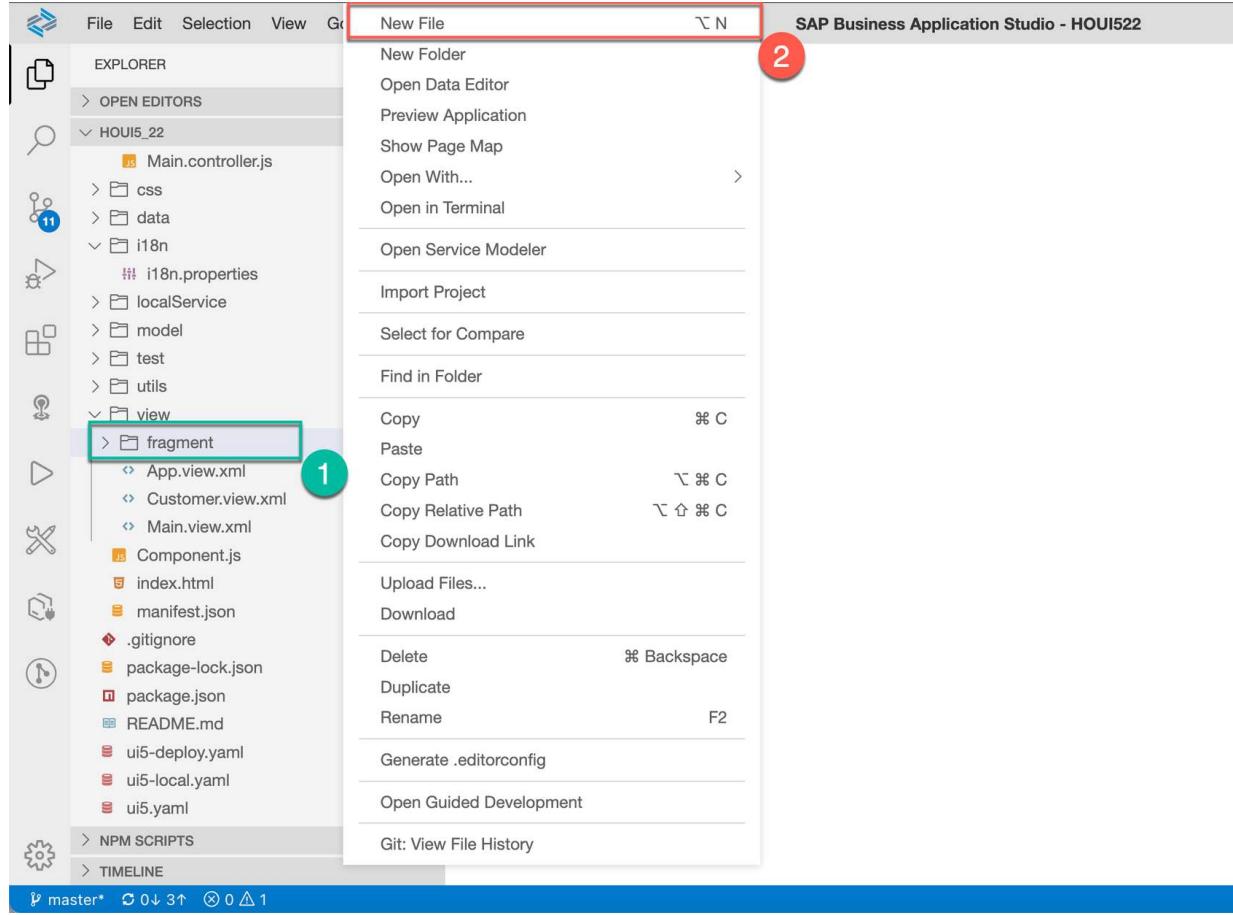
sap.m.upload.UploadSet

Ein vielseitiges Control, um einzelne oder mehrere Dateien hochzuladen. Bietet ein breites Spektrum an Zusatzfunktionen an, unter anderem eine Liste der hochgeladenen Dateien mit Editierfunktion, etc..

Most used Properties:

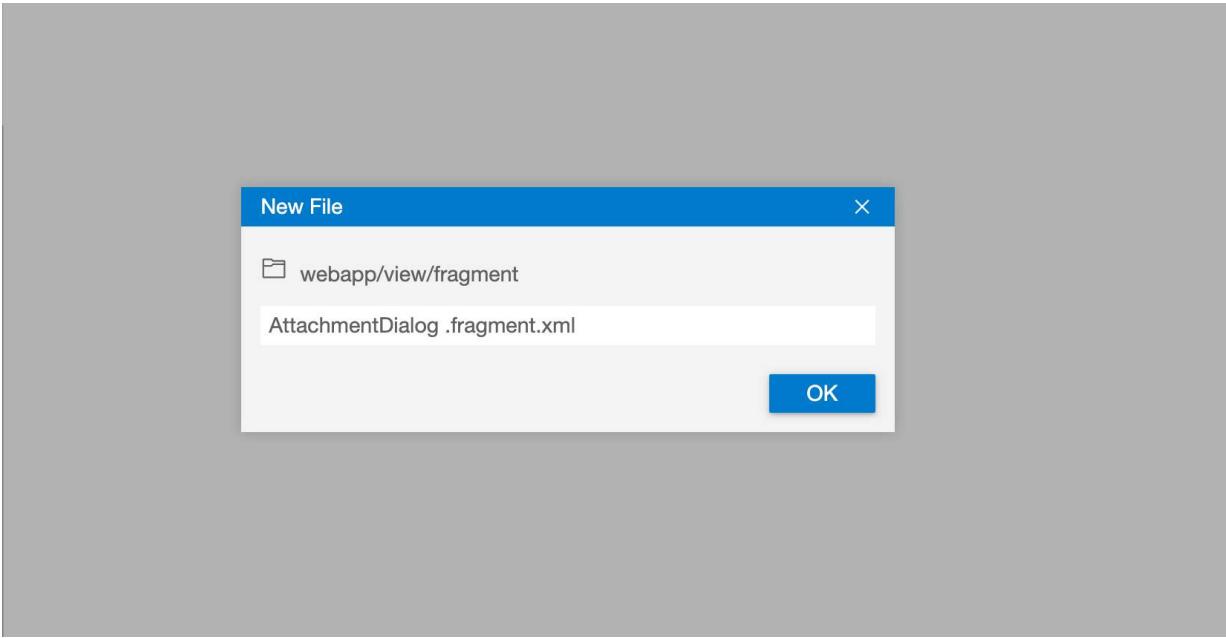
- fileTypes – Erlaubte Dateitypen
- uploadUrl – Url des Remote-Servers
- beforeUploadStarts – Aktion die vor dem Starten des Uploads ausgeführt wird
- items – Items der Liste aller Dateien
- incompleteItems – Items die zum Hochladen markiert worden sind

Wir werden einen UploadSet verwenden, welcher in ein Dialog ausgelagert ist und pro Customer in der Detail-View geöffnet werden kann.



1. Kontextmenü vom Verzeichnis **fragment** öffnen

2. Auf **New File** klicken



1. Namen AttachmentDialog.fragment.xml vergeben

```
<core:FragmentDefinition xmlns:core="sap.ui.core" xmlns="sap.m"
    xmlns:upload="sap.m.upload" xmlns:f="sap.ui.layout.form">
    <Dialog id="attachments dialog" title="{i18n>attachments title}" resizable="true">
        <upload:UploadSet
            id="attachments_uploadset"
            showIcons="true"
            items="{path: 'to_CustomerDocument', templateShareable: false, sorter: { path: 'Createdat', descending: true}}"
            mode="None"
            instantUpload="false"
            afterItemRemoved="onAfterUploadItemRemoved"
            afterItemAdded="onAfterItemAdded"
            beforeUploadStarts="onBeforeUploadStarts"
            uploadCompleted="onUploadCompleted">
            <upload:items>
```

1. FragmentDefinition mit einem Dialog einfügen

2. In dem Dialog einen UploadSet platzieren - die items Aggregation wird von der Navigation-Property `to_CustomerDocument` befüllt

```
<core:FragmentDefinition xmlns:core="sap.ui.core" xmlns="sap.m"
    xmlns:upload="sap.m.upload" xmlns:f="sap.ui.layout.form">
    <Dialog id="attachments_dialog" title="{i18n>attachments_title}" resizable="false">
        <upload:UploadSet
            id="attachments_uploadset"
            showIcons="true"
            items="{path: 'to_CustomerDocument', templateShareable: false,
            mode="None"
            instantUpload="false"
            afterItemRemoved="onAfterUploadItemRemoved"
            afterItemAdded="onAfterItemAdded"
            beforeUploadStarts="onBeforeUploadStarts"
            uploadCompleted="onUploadCompleted">
            <upload:items>

                </upload:items>
            </upload:UploadSet>
            <buttons>
                <Button id="attachment_dialog_close_button"
                    text="{i18n>attachment_close}" press="onAttachmentsDialogClose">
            </buttons>
        </Dialog>
    </core:FragmentDefinition>
```

AttachmentDialog.fragment.xml

```

1  <core:FragmentDefinition xmlns:core="sap.ui.core" xmlns="sap.m"
2   |   xmlns:upload="sap.m.upload" xmlns:f="sap.ui.layout.form">
3   |       <Dialog id="attachments_dialog" title="{i18n>attachments_title}" resizable="true">
4   |           <upload:UploadSet
5   |               id="attachments_uploadset"
6   |               showIcons="true"
7   |               items="{path: 'to_CustomerDocument', templateShareable: false, sorter: { path: 'Createdat', descending: true}}"
8   |               mode="None"
9   |               instantUpload="false"
10  |               afterItemRemoved="onAfterUploadItemRemoved"
11  |               afterItemAdded="onAfterItemAdded"
12  |               beforeUploadStarts="onBeforeUploadStarts"
13  |               uploadCompleted="onUploadCompleted">
14  |                   <upload:items>
15  |                       <upload:UploadSetItem
16  |                           id="attachments_uploadset_item"
17  |                           fileName="{Documentname}"
18  |                           visibleRemove="true"
19  |                           visibleEdit="false"
20  |                           removePressed="onRemovePressed"
21  |                           url="{path: 'Documentid', formatter: '.formatUrl'}/>
22  |                   </upload:items>
23  |               </upload:UploadSet>
24  |           <buttons>
25  |               <Button id="attachment_dialog_close_button"
26  |                   text="{i18n>attachment_close}" press="onAttachmentsDialogClose"/>
27  |           </buttons>
28  |       </Dialog>
29  </core:FragmentDefinition>
```

You, 3 days ago · houis all without uploadSet&smartTable

1

1. UploadsetItem einfügen

```

<upload:items>
    <upload:UploadSetItem
        id="attachments_uploadset_item"
        fileName="{Documentname}"
        visibleRemove="true"
        visibleEdit="false"
        removePressed="onRemovePressed"
        url="{path: 'Documentid', formatter: '.formatUrl'}/>
</upload:items>
```

```

<xmlns="sap.m">
<u:ObjectPageLayout id="cust_objectpagelayout"
    showTitleInHeaderContent="true"
    showFooter="{$editModel}/editmode}"
    upperCaseAnchorBar="false">
<u:headerTitle>
    <u:ObjectPageDynamicHeaderTitle>
        <u:expandedHeading>
            <Title text="Firstname {Lastname}" />
        </u:expandedHeading>

        <u:snappedHeading>
            <FlexBox fitContainer="true" alignItems="Center">
                <Avatar src="sap-icon://person-placeholder" class="sapUiTinyMarginEnd"/>
                <Title text="Firstname {Lastname}" />
            </FlexBox>
        </u:snappedHeading>

        <u:actions>
            <HBox>
                <Button id="cust_edit_button" text="{$i18n>cust.edit.button}" type="Emphasized" icon="sap-icon://edit"
                    press="onEditPressed" visible="{$editModel}/editmode}" class="sapUiTinyMarginEnd"/>
                <Button id="cust_button_attachments" text="{$i18n>cust.attachments}" visible="{$editModel}/editmode}" icon="sap-icon://attachment"
                    press=".onOpenAttachments"/>
            </HBox>
        </u:actions>
    </u:ObjectPageDynamicHeaderTitle>
</u:headerTitle>

```

1

```

<u:headerContent> You, 3 days ago * houis all without uploadSet&smartTable
    <FlexBox>
        <Avatar class="sapUiSmallMarginEnd" src="sap-icon://person-placeholder" displaySize="L" />

```

1. In der Customer-View im Header einen Button hinzufügen - mit diesem Button wird der Dialog geöffnet

```

<Button id="cust_button_attachments" text="{$i18n>cust.attachments}" visible="{$editModel}/editmode}" icon="sap-icon://attachment"
    press=".onOpenAttachments"/>

```

```

onOpenAttachments: function(oEvent) {
    if (!this._pDialog) {
        this._pDialog = Fragment.load({
            id: this.getView().getId(),
            name: "at.clouddna.training00.zhoui5.view.fragment.AttachmentDialog",
            controller: this
        }).then((oDialog)=>{
            this.getView().addDependent(oDialog);
            return oDialog;
        });
    }

    this._pDialog.then(function (oDialog) {
        oDialog.open();
    }.bind(this));
},

```

1

2

3

1. Die Funktion **onOpenAttachments** implementieren
2. Im ersten Teil wird geprüft, ob der **Dialog** schon Mal instanziert worden ist - wenn nicht, dann wird an dieser Stelle instanziert
3. Danach wird der **Dialog** in jedem Fall **geöffnet**

```

onOpenAttachments: function(oEvent) {
    if (!this._pDialog) {
        this._pDialog = Fragment.load({
            id: this.getView().getId(),
            name: "at.clouddna.training00.zhoui5.view.fragment.AttachmentDialog",
            controller: this
        }).then((oDialog)=>{
            this.getView().addDependent(oDialog);
            return oDialog;
        });
    }

    this._pDialog.then(function (oDialog) {
        oDialog.open();
    }.bind(this));
},

```

```

        } .bind(this));
    },
}

onAfterItemAdded: function(oEvent){
    const oUploadSet = this.getView().byId("attachments_uploadset");
    const oUploadSetItem = oEvent.getParameters().item;
    const sPath = this.getView().getBindingContext().sPath;

    oUploadSet.removeAllHeaderFields();

    this.getView().setBusy(true);

    this.getModel().create(sPath + "/to_CustomerDocument", {}, {
        success: (oData, response)=>{
            this.getView().setBusy(false);

            oUploadSet.addHeaderField(new Item({
                key: "X-CSRF-Token",
                text: this.getView().getModel().getSecurityToken()
            }));

            oUploadSet.addHeaderField(new Item({
                key: "Content-Disposition",
                text: `filename=${oUploadSetItem.getFileName()}` 
            }));

            oUploadSet.setUploadUrl(` ${this.getModel().sServiceUrl}/Z_C_CUSTOMERDOCUMENT(guid'${oData.Documentid}')/value`);

            oUploadSet.setHttpRequestMethod("PUT");

            oUploadSet.uploadItem(oUploadSetItem);
        },
        error: (oError)=>{
            this.getView().setBusy(false);
            MessageBox.error(oError.message);
        }
    });
},
}

```

1. In der Funktion **onAfterItemAdded** wird der **Name des Dokuments** extrahiert und als **HeaderParameter** gesetzt. Zusätzlich wird der **x-csrf-token** als **Header** bzgl. Authentifizierung gesetzt und **Upload-Url** sowie **Upload-Methode** festgelegt

i Achtung: Import von **sap.ui.core.Item** nicht vergessen! In dieser Methode **onAfterItemAdded** wird auf eine Variable mit dem Namen **Item** zugegriffen, welcher diesen Import beinhalten sollte.

```

sap.ui.define([
    "at/clouddna/training00/zhoui5/controller/BaseController",
    "sap/ui/model/json/JSONModel",
    "sap/ui/core/Fragment",
    "sap/m/MessageBox",
    "at/clouddna/training00/zhoui5/controller/formatter/HOU15Formatter",
    "sap/ui/core/Item"
],
/***
 * @param {typeof sap.ui.core.mvc.Controller} Controller
 */
function (BaseController, JSONModel, Fragment, MessageBox, HOU15Formatter)
...

```

```

onAfterItemAdded: function(oEvent) {
    const oUploadSet = this.getView().byId("attachments_uploadset");
    const oUploadsetItem = oEvent.getParameters().item;
    const sPath = this.getView().getBindingContext().sPath;

    oUploadSet.removeAllHeaderFields();

    this.getView().setBusy(true);

    this.getView().getModel().create(sPath + "/to_CustomerDocument", {}, {
        success: (oData, response)=>{
            this.getView().setBusy(false);

            oUploadSet.addHeaderField(new Item({
                key: "X-CSRF-Token",
                text: this.getView().getModel().getSecurityToken()
            }));

            oUploadSet.addHeaderField(new Item({
                key: "Content-Disposition",
                text: `filename=${oUploadsetItem.getFileName()}`
            }));

            oUploadSet.setUploadUrl(` ${this.getModel().sServiceUrl}/Z_C_CU`+

```

```

        oUploadSet.setHttpRequestMethod("PUT");

        oUploadSet.uploadItem(oUploadsetItem);
    },
    error: (oError)=>{
        this.getView().setBusy(false);
        MessageBox.error(oError.message);
    }
));
},

```

```

formatUrl: function(sDocumentid){
    const sPath = this.getView().getModel().createKey("/Z_C_CUSTOMERDOCUMENT", {
        Documentid: sDocumentid
    });

    return this.getView().getModel().sServiceUrl + sPath + "/$value";
},

```

1. Die Methode "formatUrl" erstellt und formatiert eine URL für ein Dokument anhand seiner Dokumenten-ID. Dabei wird der spezifische Pfad im OData-Modell mithilfe von "createKey" generiert, und die vollständige URL wird durch Verknüpfung mit dem Service-URL des Modells sowie der Ergänzung von "\$value" erstellt. Diese Methode wird für den Zugriff auf das Dokument, beispielsweise zum Download oder zur Anzeige im Frontend, verwendet.

```

formatUrl: function(sDocumentid){
    const sPath = this.getView().getModel().createKey("/Z_C_CUSTOMERDOCUMENT",
        Documentid: sDocumentid
    );
}

```

```
        return this.getView().getModel().sServiceUrl + sPath + "/$value";  
    },
```

```
    onUploadCompleted: function(){  
        this.getView().setBusy(false);  
        this.getView().getModel().refresh(true);  
    },
```

1

```
    onRemovePressed: function(oEvent){  
        oEvent.preventDefault();  
  
        const oModel = this.getView().getModel();  
        const sPath = oEvent.getSource().getBindingContext().getPath();  
  
        this.getView().setBusy(true);  
        this.getView().getModel().remove(sPath, {  
            success: (oData, response)=>{  
                this.getView().setBusy(false);  
                MessageBox.success(this.getLocalizedMessage("dialog.delete.success"));  
                oModel.refresh(true);  
            },  
            error: (oError)=>{  
                this.getView().setBusy(false);  
                MessageBox.error(oError.message);  
            }  
        });  
    },
```

2

```
    onAttachmentsDialogClose: function(){  
        this._pDialog.then(function(oDialog){  
            oDialog.close();  
        }.bind(this));  
    },
```

3

1. In der Funktion **onUploadCompleted** wird das OData-Model aktualisiert
2. Im Event-Handler **onRemovePressed** wird das Löschen eines Dokuments initiiert
3. Die Funktion **onAttachmentsDialogClose** schließt der Dialog

```
onUploadCompleted: function() {  
    this.getView().setBusy(false);  
    this.getView().getModel().refresh(true);
```

```
,  
  
onRemovePressed: function(oEvent) {  
    oEvent.preventDefault();  
  
    const oModel = this.getView().getModel();  
    const sPath = oEvent.getSource().getBindingContext().getPath();  
  
    this.getView().setBusy(true);  
    this.getView().getModel().remove(sPath, {  
        success: (oData, response)=>{  
            this.getView().setBusy(false);  
            MessageBox.success(this.getLocalizedText("dialog.delete.success"));  
            oModel.refresh(true);  
        },  
        error: (oError)=>{  
            this.getView().setBusy(false);  
            MessageBox.error(oError.message);  
        }  
    }) ;  
},  
  
onAttachmentsDialogClose: function(){  
    this._pDialog.then(function(oDialog){  
        oDialog.close();  
    }).bind(this));  
},
```

The screenshot shows a customer profile page with the following details:

Maximilian Mustermensch

Birthdate: 2.11.2023

KUNDENINFORMATIONEN

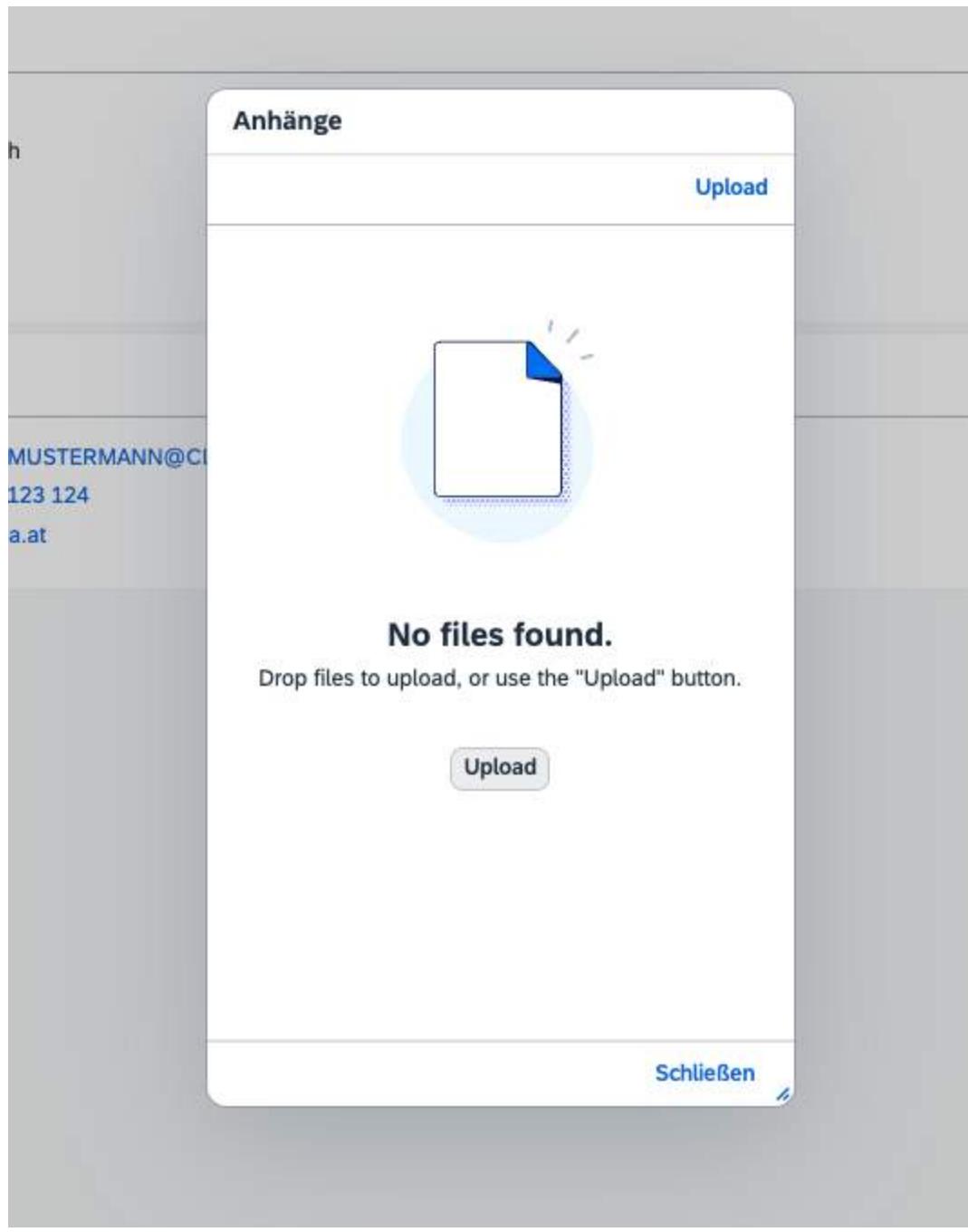
Grunddaten

Firstname:	Maximilian
Lastname:	Mustermensch
Title:	Dr.
Gender:	männlich
Birthdate:	2.11.2023

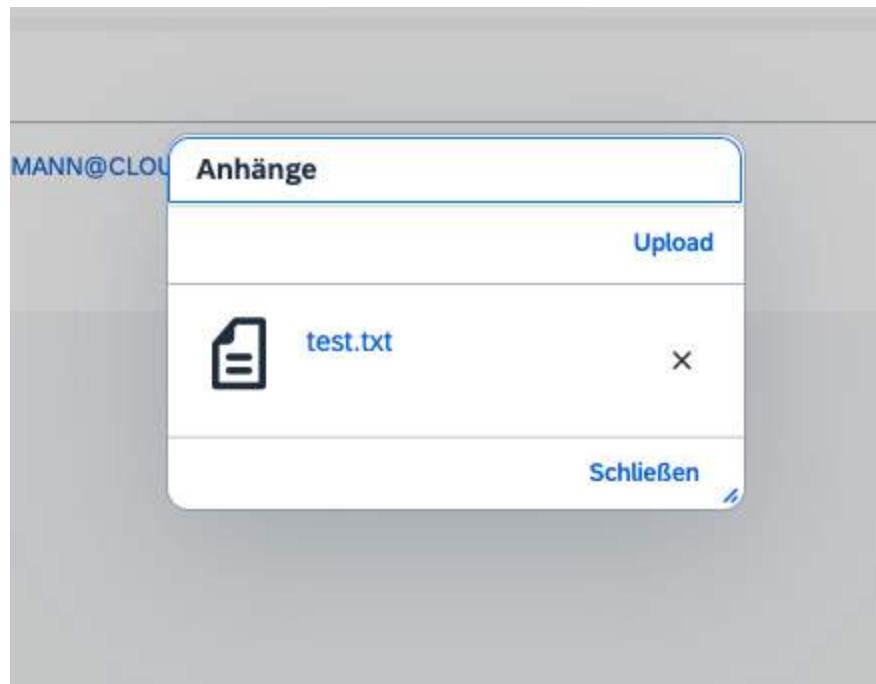
Kontaktmöglichkeiten

Email:	MAXIMILIAN.MUSTERMANN@CLOUDDNA.AT
Phone:	+43 676 123 123 124
Website:	www.clouddna.at

Button für das Öffnen des Dialogs erscheint



Noch keine Dateien hochgeladen



Die hochgeladene Datei

14.2 Zusammenfassung

MK

Martin Koch

Das "**UploadSet**" ist ideal für das Hochladen von Dokumenten geeignet. Die hochgeladenen Dokumente werden vom OData-Service akzeptiert und im SAP-System im Archiv abgelegt. Über eine spezifische URL kann die Datei dann im Binärformat gelesen und heruntergeladen werden. Diese Funktion ermöglicht eine effiziente Verwaltung von hochgeladenen Dokumenten innerhalb der Anwendung.

Weitere Tutorials

 Martin Koch

Anbei finden Sie ein paar Tutorials, die sich für Sie als nützlich erweisen könnten.

UploadCollection

Die UploadCollection ist zwar seit der SAPUI5-Version 1.88 deprecated, dennoch begegnet man ihr noch in vielen Systemen und Anwendungen. Anbei ein Tutorial, wie man mit der UploadCollection arbeitet.



Upload Collection.pdf
2.3 MB



Busy- und Dirty-State-Handling

Damit die Anwender*innen bezüglich länger andauernde Operationen bewusst werden und auf nicht gespeicherte Daten hingewiesen werden können, existieren diese zwei State-Handlings. Anbei finden Sie ein Tutorial, in dem diese zwei States ausprogrammiert werden.



Busy- und Dirty-State-Handling.pdf

1.2 MB



Extension Points und Extension Projects

Extension Points sind vordefinierte Erweiterungspunkte, die von Entwickler*innen zur Verfügung gestellt werden können, damit andere Entwickler*innen eine Applikation erweitern können.

Damit Applikation erweitert werden können, wurden früher Extension Projects angelegt. Diese sind am Auslaufen. Stattdessen sollten durch Adaption Projects verwendet werden.

In diesem Tutorial finden Sie dennoch ein Beispiel aus der alten Welt, zumindest was das Extension Project betrifft.



Extension Points und Extension Projects.pdf

2.8 MB

