EB

**Übungen**

**zum Seminar 16962  
  
Tag 03 – Übungen zur Konsumierung von OData Services in UI5**

### Aufgabe 1 – OData Service erstellen

Erstellen Sie im SAP Backendsystem einen OData Service mit der Transaktion SEGW. Erstellen Sie eine Entität SFlight, basierend auf der Datenbanktabelle SFlight. Dazu die Entitätsmenge SFlightSet. Aktivieren Sie den OData Service. Überprüfen Sie in der Transaktion /IWFND/MAIN, ob der OData Service sichtbar ist, einen aktiven ICF Knoten hat und einen Systemeintrag hat. Rufen Sie dann die Transaktion /IWFND/GW\_SERVICE auf und testen Sie, ob der Aufruf des Metadatendokuments der Entitätsmenge SFlightSet des OData Service ein Ergebnis zurückgibt.

### Aufgabe 2 – Methoden in der OData Service implementieren

Für die einzelnen REST Befehle (Get, Post, Push, Delete) implementieren Sie nun die entsprechenden Methoden in der OData Serviceklasse \*\_DPC\_EXT damit die Daten konsumiert werden können.

Redefinieren Sie in der \*\_DPC\_EXT Klasse ihres OData Services folgende Methoden:

SFLIGHTSET\_GET\_ENTITY   
SFLIGHTSET\_GET\_ENTITYSET  
SFLIGHTSET\_CREATE\_ENTITY  
SFLIGHTSET\_UPDATE\_ENTITY  
SFLIGHTSET\_DELETE\_ENTITY

Implementieren Sie die einzelnen Methoden mit ABAP/OO wie folgt:

\*GET\_ENTITY: gibt zu einem Schlüssel genau einen Eintrag aus der Tabelle SFlight zurück

\*GET\_ENTITYSET: gibt alle Einträge aus der Tabelle SFlight zurück und filtert dabei auf die Filterwerte in der Tabelle it\_filter

\*CREATE\_ENTITY: Erzeugt einen neuen Eintrag in der Tabelle SFlight für die übergebenen Werte in der Struktur er\_entity (inklusive Schlüsselfeldern)

\*UPDATE\_ENTITY: Aktualisiert einen bestehenden Eintrag (dieser wird aus Tabelle SFlight anhand der übergebenen Schlüssel in Tabelle it\_key\_tab gelesen) in der Tabelle SFlight mit den übergebenen Werten

\*DELETE\_ENTITY: Löscht einen bestehenden Eintrag (wird aus SFlight anhand der übergebenen Schlüssel in Tabelle it\_key\_tab gelesen) aus der Tabelle SFlight

### Aufgabe 3 – Im OData Service Entitätsmenge basierend auf CDS View anlegen

Im OData Service aus Aufgabe 1: Erstellen Sie eine weitere Entität SFlight\_CDS, basierend nicht auf einer Datenbanktabelle sondern auf einem CDS View. Ordnen Sie unter dem Knoten „Serviceimplementierung“ diese Entitätsmenge einer Datenquelle zu und wählen Sie dabei den CDS View aus. Es findet ein Mapping statt. Was ist nun mit den Methoden in der OData Service Klasse \*DPC\_EXT passiert? Und zwar mit den Methoden \*GET\_ENTITY und \*GET\_ENTITYSET. Müssen diese noch implementiert werden?

### Aufgabe 4 – UI5 App mit List Report und Zugriff auf OData Service anlegen

Erstellen Sie eine UI5 App. Sie können dafür zum Beispiel auch den easy-ui5 Generator zur Hilfe nehmen, der eine neue App generiert. Die Doku dafür finden Sie hier:

https://github.com/ui5-community/generator-ui5-project

Im Visual Studio Code das Terminal öffnen und dort in das entsprechende Zielverzeichnis gehen und folgendes ausführen:

Für die Installation des easy-ui5 Generators zunächst folgenden Schritt ausführen:

npm install -g yo generator-easy-ui5

Danach die Generierung der UI5 App (als Project) folgendermaßen starten:

PS C:\..\Coding> yo easy-ui5 project

? What do you want to do? project         [project]

? Which namespace do you want to use? schulung.ui5

? How do you want to name this project? odataproject

? How do you want to name the first uimodule within your project? odataapp

? Do you want to enable the SAP Fiori elements flexible programming model? No

? On which platform would you like to host the application? SAP NetWeaver

? Where should your UI5 libs be served from? Content delivery network (SAPUI5)

? Would you like to create a new directory for the project? Yes

? Would you like to initialize a local git repository for the project? Yes

In der manifest.json fügen Sie dann folgendes hinzu, um den OData Service als Service zu definieren und das UI5 Default Model auf diesen Service zu setzen:

Unter „sap.app“:

"dataSources": {

            "mainService": {

                "uri": "/sap/opu/odata/sap/ZZ\_MAL\_GET\_FLIGHT\_SRV/",

                "type": "OData",

                "settings": {

                    "odataVersion": "2.0",

                    "localUri": "localService/metadata.xml",

                    "annotations": [

                        "annotation0"

                    ]

                }

            }

        }

Unter „sap.ui5“:

        "models": {

            "": {

                "dataSource": "mainService",

                "preload": false,

                "settings": {

                    "defaultBindingMode": "TwoWay"

                }

            },

Damit bei der lokalen Ausführung der App korrekt auf das SAP Zielsystem gemapped werden kann, um den OData Service anzusprechen sollte noch folgendes in die ui5.yaml übernommen werden.

Unter server (falls schon ein Eintrag unter configuration existiert, die Angaben dort ergänzen):

  customMiddleware:

    - name: fiori-tools-proxy

      afterMiddleware: compression

      configuration:

        ignoreCertError: true

        backend:

          - path: /sap

            url: https://cot.rz.sys.aok.de:8433

            client: "099"

<https://cot.rz.sys.aok.de:8433> ist dabei die Zieladresse (in unserem Fall wohl der SAP ABAP Frontend Server / Netweaver), diese ist entsprechend zu ändern.

Fügen Sie jetzt im Main View eine sap.m.Table ein, die die Daten aus dem UI5 Default model liest, indem sie auf die Entitätsmenge (SFlightSet) zugreift . Fügen Sie Spalten (colums -> Column) und Zeilenwerte (items -> ColumnItem) mit den jeweiligen Feldnamen ein. Nutzen Sie dafür Beispiele aus der SAP UI5 Doku.

### Aufgabe 5a – UI5 App – Ersetzen der sap.m.Table

Die Verwendung von Smart Controls in UI5 hat viele Vorteile, da diese die Annotationen des OData Services zur Generierung von z.B. Tabellenspalten, Filtern und Formatierungen nutzen können. Ersetzen sie die sap.m.Table aus Aufgabe 4 durch eine sap.ui.comp.smarttable.SmartTable. Fügen Sie ebenfalls eine SmartFilterBar hinzu (sap.ui.comp.smartfilterbar.SmartFilterBar). Die SmartTable sollte auf die ID dieser SmartFilterBar verweisen.

Nutzen Sie dafür Beispiele aus der SAPUI5 Doku. Es müssen dafür keine separaten Spalten Zeilenwerte definiert werden, die SmartTable zieht sich das alles aus dem Backend. Ohne das Attribut der inital angezeigten Spalten sind aber zunächst keine Spalten sichtbar (da dafür Annotationen aus dem Backend fehlen), diese müssen über die Einstellungen extra eingeblendet werden.

### Aufgabe 5b – UI5 App – SmartTable – Hinzufügen von Spalten und Smart Filtern

Fügen Sie der SmartTable aus Aufgabe 5 initial sichtbare Spalten hinzu. Fügen Sie dem SmartFilter zwei Filterfelder für Carrid und Connid hinzu.

Fügen Sie außerdem der SmartTable eine sap.m.Table als Kindknoten hinzu und setzen Sie dort mit dem Attribut sticky die HeaderToolbar, InfoToolbar und ColumnHeaders auf sticky, so dass sie beim vertikalen Scrollen sichtbar bleiben.

Nutzen Sie dafür Beispiele aus der SAPUI5 Doku.

### Aufgabe 6a – UI5 App – Routing zur Detailseite

Erstellen Sie mit Hilfe des easy-ui5 Generators einen neuen View (Detail) und fügen Sie diesem Target und Routing hinzu.

Führen Sie dazu folgendes im Terminal aus:

PS C:\...\schulung.ui5.odataproject> yo easy-ui5 project view

✨ found existing uimodule odataapp

? How do you want to name your new view? Detail

? Do you want to set up a JavaScript controller for your new view? Yes

? Do you want to set up a route and target for your new view? Yes

creating new view for odataapp

   create odataapp\webapp\controller\Detail.controller.js

   create odataapp\webapp\view\Detail.view.xml

 conflict odataapp\webapp\i18n\i18n.properties

? Overwrite odataapp\webapp\i18n\i18n.properties? overwrite

    force odataapp\webapp\i18n\i18n.properties

 conflict odataapp\webapp\manifest.json

? Overwrite odataapp\webapp\manifest.json? overwrite

    force odataapp\webapp\manifest.json

Fügen Sie nun Code für den Detail Controller ein. Dieser überprüft u.a. den Hash in der URL und extrahiert die entsprechenden Parameter. Diese Parameter sind später die Schlüsselwerte für einen Satz aus SFlight. Dazu aber später mehr.

Laden Sie den Quellcode für den DetailController aus dem GitHub und fügen Sie ihn ein.

Fügen Sie nun im MainView in der sap.m.Table (ist ein Kind der SmartTable und überschreibt definierte Eigenschaften dieser) in der items Aggregation ein ColumnListItem vom Typ „Navigation“ hinzu, welches auf den press Event reagiert und damit eine EvenHandler-Funktion: „onPress“ im MainView Controller aufruft.

In der onPress EventHandler Funktion im MainView Controller lesen Sie nun aus dem BindingContext des Items der angeklickten Zeile die Schlüsselwerte (Carrid, Connid, Fldate) aus und übergeben diese an den Router. Routen Sie damit zur Detailseite und übergeben Sie die Schlüsselwerte als Parameter.

Beim Klick auf eine Tabellenzeile sollte nun die Detailseite aufgerufen werden.

Nutzen Sie dafür auch Beispiele aus der SAPUI5 Doku.

### Aufgabe 6b – UI5 App – Formular auf Detailseite und Binding

Fügen Sie dem DetailView ein Formular hinzu (SimpleForm) mit Eingabefeldern für die Felder CarrId, ConnId, Fldate und Price.

Beim Ausführen werden in den Feldern noch nicht die eigentlichen Werte angezeigt. Das liegt daran, dass sie noch nicht an den entsprechenden Datensatz im OData Model gebunden sind.

Das erreichen wir durch folgendes Vorgehen:

* + 1. Ändern des Patterns für die Detail Route, so dass die 3 Schlüsselparameter carrid, connid und fldate mit übergeben werden
    2. Im Detail Controller: Nach dem PatternMatch; Auslesen der Parameter aus dem Pattern (URL)
    3. Erstellen eines Schlüssels (mit diesen 3 Parametern) im OData Model für die Entitätsmenge
    4. Ermitteln des Pfads des Schlüssels
    5. Binden des gesamten DetailViews an den Pfad (mit this.getView().bindElement)

Das Binding des Views an den Pfad im OData Model sorgt dafür dass das Binding an alle Kinder im View weitergerreicht wird. Dadurch bekommen auch die Formularfelder die entsprechenden Werte aus dem Datensatz.

Nutzen Sie für das Einfügen des Formulars (SImpleForm) auch Beispiele aus der SAPUI5 Doku.

### Aufgabe 7a – UI5 App –Detailseite – Flug aktualisieren

Fügen Sie dem DetailView unter dem Formular 2 Buttons hinzu: Speichern und Abbrechen.

Verknüpfen Sie die press Events der Buttons jeweils mit 2 EventHandler Funktionen im Detail Controller. Beim Abbrechen soll zurück auf den MainView navigiert werden (siehe Funktion: onNavBack).

Beim Speichern sollen mittels der Update Funktion des OData Models die Werte für Price und Currency im Backend für den an den View gebundenen Modelpfad (mit den 3 Schlüsseln) aktualisiert werden.

Das erreichen wir durch folgendes Vorgehen:

* + 1. Im Detail Controller: Auslesen des an den View gebunden Modelpfads mittels getBindingContext
    2. Auslesen der Werte für Price und Currency aus dem BindingContext und übernehmen der Werte (mit Key/Value) in ein Object
    3. Aufruf der Update Funktion des OData Models und Übergabe des Modelspfads und des Objects mit den geänderten Werten
    4. Nach Aufruf der Update Funktion: Ausgabe von Nachrichten im Erfolgsfall und im Fehlerfall (mittels sap.m.MessageBox).
    5. Da die Felder Carrid, Connid und Fldate Schlüsselfelder sind, sollen diese nicht geändert werden, deshalb sollten die entsprechenden Eingabefelder auf enabled=false gesetzt werden.

### Aufgabe 7b – UI5 App –Detailseite – Flug hinzufügen

Fügen Sie dem MainView in einer Toolbar der Tabelle einen Button hinzu (nur mit einem Plus Icon). Verknüpfen Sie den Button mit einer EventHandler Funktion in MainView Controller. Dort wird dann ein Dialog erzeugt und geöffnet.

Das Hinzufügen eines neuen Fluges erreichen wir durch folgendes Vorgehen:

* + 1. Anlegen eines Views für den Dialog. Der Dialog enthält ein Formular (es kann dieselbe SimpleForm aus dem DetailView genommen werden). Außerdem einen Button zum Speichern und zum Abbrechen
    2. Im MainView Controller: In der EventHandler Funktion zum Hinzufügen wird der Dialog geladen und geöffnet.
    3. Nach dem Öffnen des Dialogs wird (ähnlich wie im Detail) mit der OData Model Funktion createKey ein Datensatz erzeugt (ohne Schlüssel da diese ja bei Neuanlage noch nicht feststehen). Dann wird der Pfad dieses Keys ermittelt und der Dialog per bindElement Funktion an diesen Pfad gebunden
    4. Der Abbrechen Dialog wird mit einer EventHandler Funktion verknüpft, die den Dialog schließt.
    5. Der Speichern Button im Dialog wird mit einer EventHandler Funktion (z.B. onAddFlight) verknüpft. In der EventHandler Funktion wird zunächst ein Object mit den Key/Value Werten angelegt. Die Werte für die Felder werden mittels getBindingContext aus dem Dialog gelesen. Dann wird mit der OData Model Funktion create (Übergabe des Modelspfads und des Objects ) der Datensatz im Backend erzeugt. Danach wird der Dialog geschlossen.
    6. Nach Aufruf der Create Funktion: Ausgabe von Nachrichten im Erfolgsfall und im Fehlerfall (mittels sap.m.MessageBox).

#### Element Binding

Nutzen Sie Element Binding um ein UI5 Control an ein Model zu binden.

Erstellen Sie eine XML-View und einen Controller, die ein sap.m.Panel Control verwenden, um die Details eines Produkts anzuzeigen. Binden Sie das sap.m.Panel Control an einen bestimmten Pfad im Model.

Nutzen Sie dazu ein JSON Model mit folgenden Daten:

{

                product: {

                    name: "Laptop",

                    price: "1200 EUR"

                }  
}

            }

#### Expression Binding

Nutzen Sie Expression Binding um ein UI5 Control an ein Model zu binden.

Erstellen Sie eine XML-View und einen Controller, die ein sap.m.Text Control verwenden, um den Lagerstatus eines Produkts anzuzeigen. Verwenden Sie ein Expression Binding, um den Text basierend auf der Menge im Lager zu berechnen.

Nutzen Sie dazu ein JSON Model mit folgenden Daten:

{

                product: {

                    quantity: 15

                }

}

#### Aggregation Binding

Nutzen Sie Aggregation Binding um ein UI5 Control an ein Model zu binden.

Erstellen Sie eine XML-View und einen Controller, die ein sap.m.List Control verwenden, um eine Liste von Produkten anzuzeigen. Binden Sie die items-Aggregation des sap.m.List Controls an eine Sammlung von Daten im Modell.

Nutzen Sie dazu ein JSON Model mit folgenden Daten:

{

                products: [

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

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

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

                ]

            }

#### Composite Binding

Nutzen Sie Composite Binding um ein UI5 Control an ein Model zu binden.

Erstellen Sie eine XML-View und einen Controller, die ein sap.m.Text Control verwenden, um den vollständigen Namen eines Benutzers anzuzeigen. Verwenden Sie ein Composite Binding, um den vollständigen Namen aus den Vor- und Nachnamen-Eigenschaften im Modell zu berechnen.

Nutzen Sie dazu ein JSON Model mit folgenden Daten:

{

                user: {

                    firstName: "John",

                    lastName: "Doe"

                }

}

#### Einfache Navigation zwischen Views

Implementieren Sie ein einfaches Routing.

Erstellen Sie eine einfache Navigation von einer Startseite zu einer Detailseite. Konfigurieren Sie das Routing in der manifest.json und implementieren Sie die Navigation im Controller. Die Navigation kann zum Beispiel durch Klick auf einen Button ausgelöst werden.

#### Navigation mit Parameterübergabe

Nutzen Sie Composite Binding um ein UI5 Control an ein Model zu binden.

Erweitern Sie die Navigation aus der vorherigen Übung, um einen Parameter (z.B. eine Produkt-ID) von der Startseite zur Detailseite zu übergeben und den Parameter auf der Detailseite anzuzeigen.

#### Navigation zurück zur Startseite

Erweitern Sie die Navigation aus der vorherigen Übung und fügen Sie einen Button zur Detailseite hinzu, um zur Startseite zurückzukehren. Implementieren Sie die Navigation im Controller.