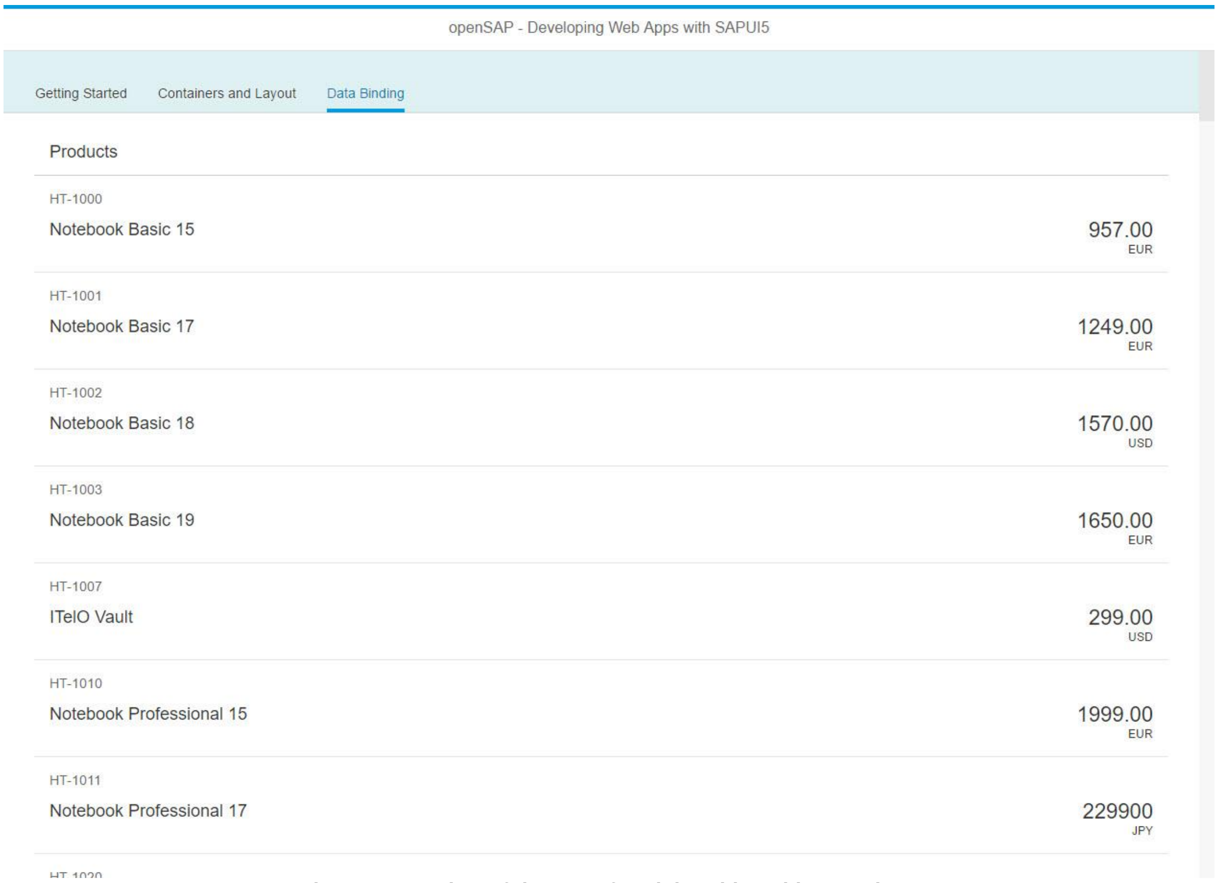
WEEK 2 UNIT 1
USING A REMOTE SERVICE WITH AGGREGATION BINDING

Please perform the exercises below in your app project as shown in the video.

Table of Contents

Preview



openSAP - Developing Web Apps with SAPUI5

Products

HT-1000 Notebook Basic 15	957.00 EUR
HT-1001 Notebook Basic 17	1249.00 EUR
HT-1002 Notebook Basic 18	1570.00 USD
HT-1003 Notebook Basic 19	1650.00 EUR
HT-1007 ITelO Vault	299.00 USD
HT-1010 Notebook Professional 15	1999.00 EUR
HT-1011 Notebook Professional 17	229900 JPY

Figure 1 - Preview of the app after doing this unit's exercises

1 VERIFY YOUR DESTINATION TO BACKEND SYSTEM ES5

In this step we will verify the backend connection that you set up in week 0 unit 1.

Check User Access to the Service on System ES5

Explanation

1. Navigate to the following URL and log in with the user/pw create in week 0 unit 1:

   https://sapes5.sapdevcenter.com/sap/opu/odata/IWBEP/GWSAMPLE_BASIC/$metadata

   User: <P…> (Same as your SAP Hana Cloud Platform Developer Account)
   PW: <…>

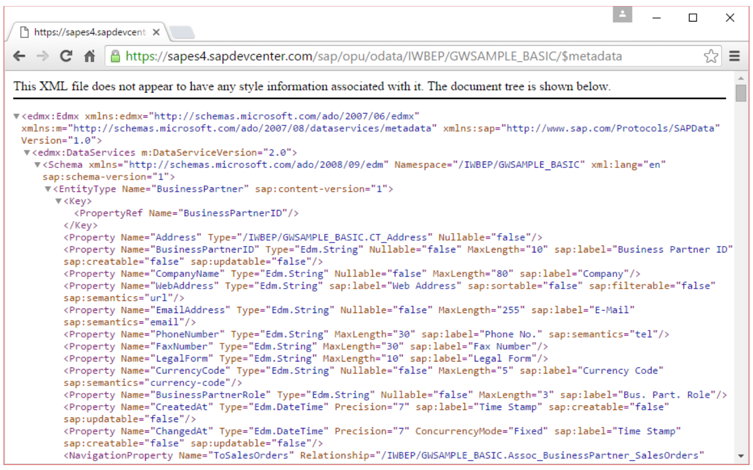
   You should see an XML document defining the service metadata that we will use throughout this course

With the change of the public demo system from ES4 to ES5 the sign-up process and the service URLs have been updated.

Please use ES5 whenever ES4 mentioned throughout this course.

Note: If you cannot access the metadata document or forgot your username/password you need to follow the sign up procedure from week 0 unit 1 – 2 CONNECT TO THE DEMO SYSTEM ES5. You will not be able to do the exercises for the remaining weeks of the course otherwise.

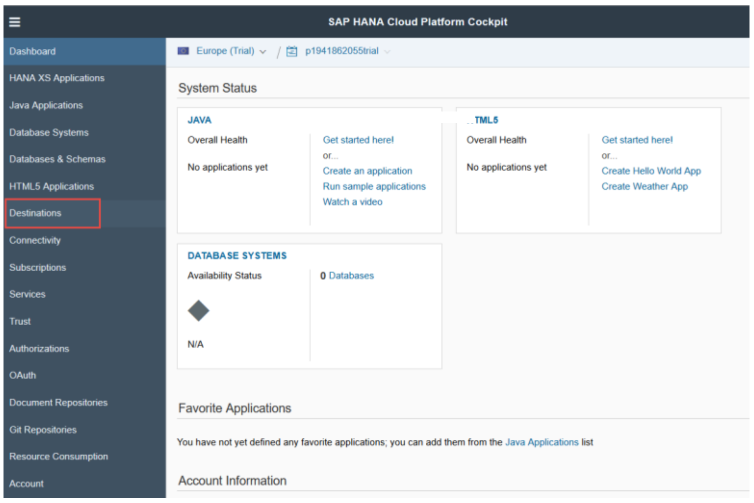
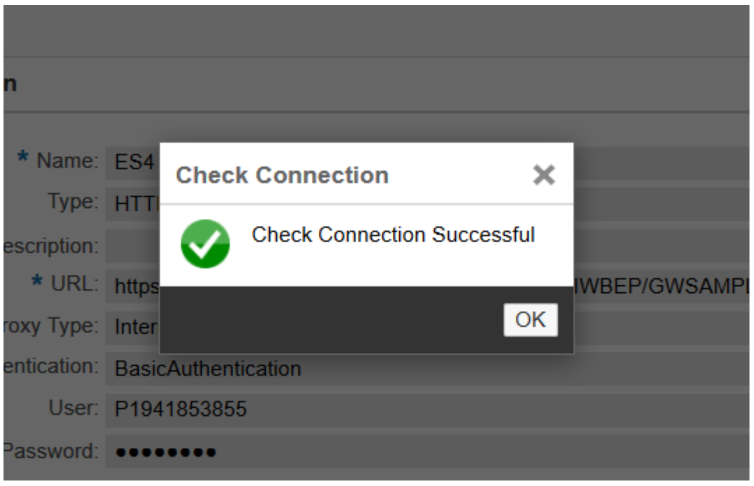
Screenshot


https://sapes4.sapdevcent
https://sapes4.sapdevcenter.com/sap/opu/odata/IWBEP/GWSAMPLE_BASIC/$metadata

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<edmx:Edmx xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:sap="http://www.sap.com/Protocols/SAPData"
  Version="1.0">
  ▼<edmx:DataServices m:DataServiceVersion="2.0">
    ▼<Schema xmlns="http://schemas.microsoft.com/ado/2008/09/edm" Namespace="/IWBEP/GWSAMPLE_BASIC" xml:lang="en"
      sap:schema-version="1">
      ▼<EntityType Name="BusinessPartner" sap:content-version="1">
        ▼<Key>
            <PropertyRef Name="BusinessPartnerID"/>
          </Key>
          <Property Name="Address" Type="/IWBEP/GWSAMPLE_BASIC.CT_Address" Nullable="false"/>
          <Property Name="BusinessPartnerID" Type="Edm.String" Nullable="false" MaxLength="10" sap:label="Business Partner ID"
          sap:creatable="false" sap:updatable="false"/>
          <Property Name="CompanyName" Type="Edm.String" Nullable="false" MaxLength="80" sap:label="Company"/>
          <Property Name="WebAddress" Type="Edm.String" sap:label="Web Address" sap:sortable="false" sap:filterable="false"
          sap:semantics="url"/>
          <Property Name="EmailAddress" Type="Edm.String" Nullable="false" MaxLength="255" sap:label="E-Mail"
          sap:semantics="email"/>
          <Property Name="PhoneNumber" Type="Edm.String" MaxLength="30" sap:label="Phone No." sap:semantics="tel"/>
          <Property Name="FaxNumber" Type="Edm.String" MaxLength="30" sap:label="Fax Number"/>
          <Property Name="LegalForm" Type="Edm.String" MaxLength="10" sap:label="Legal Form"/>
          <Property Name="CurrencyCode" Type="Edm.String" Nullable="false" MaxLength="5" sap:label="Currency Code"
          sap:semantics="currency-code"/>
          <Property Name="BusinessPartnerRole" Type="Edm.String" Nullable="false" MaxLength="3" sap:label="Bus. Part. Role"/>
          <Property Name="CreatedAt" Type="Edm.DateTime" Precision="7" sap:label="Time Stamp" sap:creatable="false"
          sap:updatable="false"/>
          <Property Name="ChangedAt" Type="Edm.DateTime" Precision="7" ConcurrencyMode="Fixed" sap:label="Time Stamp"
          sap:creatable="false" sap:updatable="false"/>
          <NavigationProperty Name="ToSalesOrders" Relationship="/IWBEP/GWSAMPLE_BASIC.Assoc_BusinessPartner_SalesOrders"
```

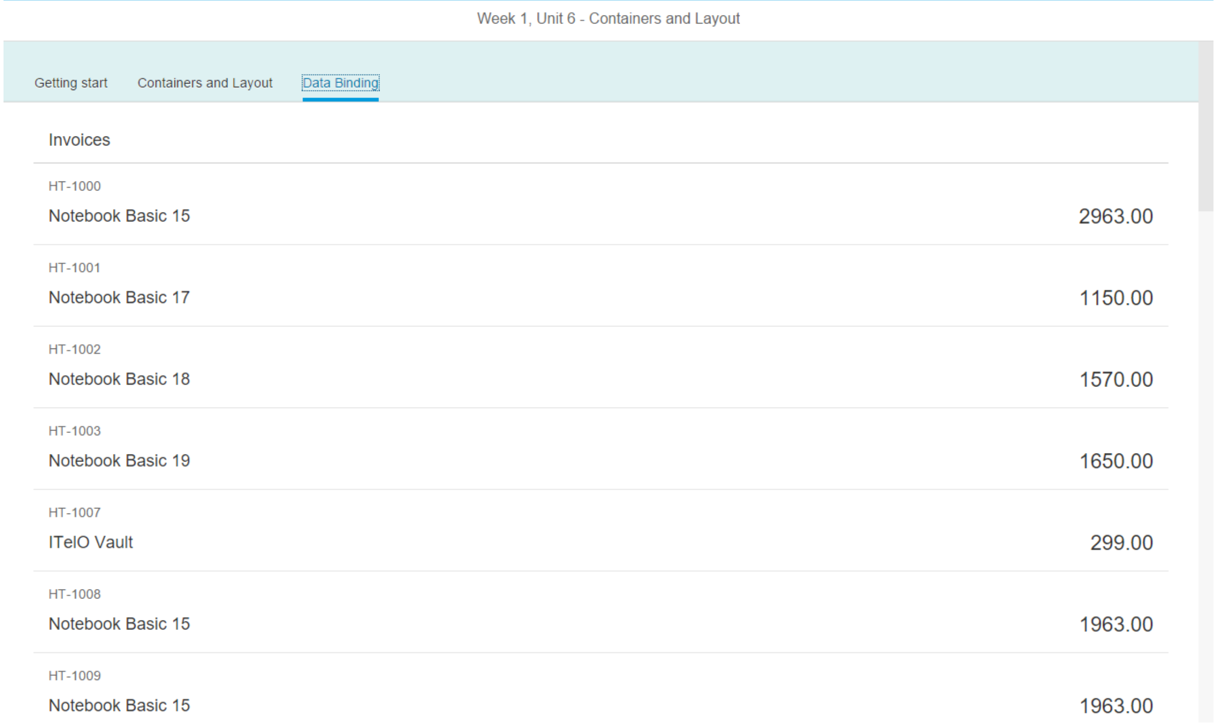
Check Destination to System ES5

Explanation	Screenshot
1. Go to destinations page of the SAP HCP Cockpit by opening the following URL and pressing the menu item "Destinations": https://account.hanatrial.ondemand.com/cockpit	 SAP HANA Cloud Platform Cockpit Dashboard HANA XS Applications Java Applications Database Systems Databases & Schemas HTML5 Applications Destinations Connectivity Subscriptions Services Trust Authorizations OAuth Document Repositories Git Repositories Resource Consumption Account Europe (Trial) / p1941862055trial System Status JAVA Overall Health No applications yet Get started here! or... Create an application Run sample applications Watch a video TML5 Overall Health No applications yet Get started here! or... Create Hello World App Create Weather App DATABASE SYSTEMS Availability Status N/A 0 Databases Favorite Applications You have not yet defined any favorite applications; you can add them from the Java Applications list Account Information
2. Make sure that there is a destination "ES5" created and click on the check icon on the right side. Verify that the connection test is successful. **Note:** If there is no destination ES5 in your developer account you need to follow the sign up procedure from week 0 unit 1 – 2 CONNECT TO THE DEMO SYSTEM ES5. You will not be able to do the exercises for the remaining weeks of the course otherwise.	 n * Name: ES4 Type: HTT escription: * URL: https IWBEP/GWSAMPL roxy Type: Inter entication: BasicAuthentication User: P1941853855 Password: ●●●●●●●●● Check Connection Check Connection Successful OK

2 USE THE SERVICE IN YOUR APP

In this step we will add a new tab “Data Binding” to our app and display a list of products from the ES5 backend service.

Preview



Week 1, Unit 6 - Containers and Layout

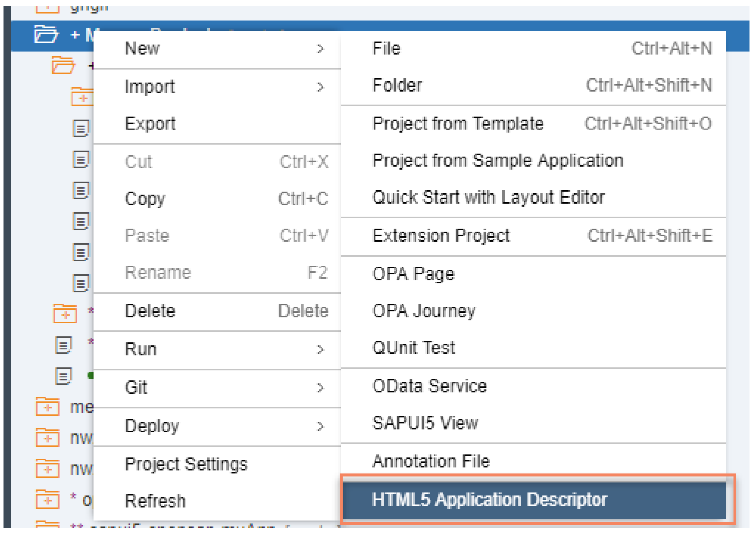
Getting start | Containers and Layout | Data Binding

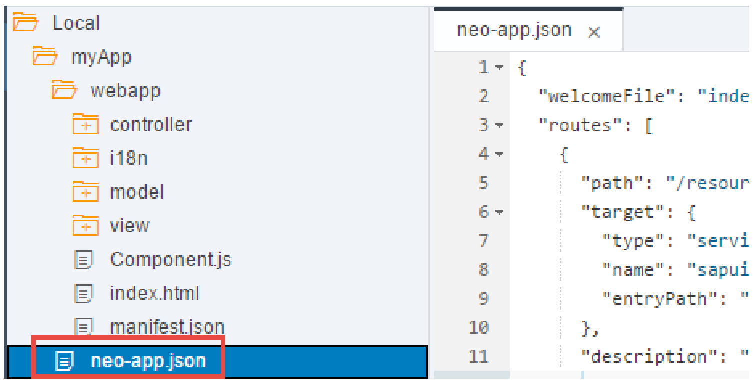
Invoices

HT-1000 Notebook Basic 15	2963.00
HT-1001 Notebook Basic 17	1150.00
HT-1002 Notebook Basic 18	1570.00
HT-1003 Notebook Basic 19	1650.00
HT-1007 ITelO Vault	299.00
HT-1008 Notebook Basic 15	1963.00
HT-1009 Notebook Basic 15	1963.00

Figure 2 - A list of products from the backend service

Create a neo-app.json file

Explanation	Screenshot
1. Open SAP Web IDE and right click on your project folder “myApp”. Select “New > HTML5 Application Descriptor”. **Note:** The menu option will only appear if the project folder is directly located below the root node “Workspace”.	

Explanation	Screenshot
2. Open the newly created file `neo-app.json`. It contains a JSON configuration file for your project with resource definitions. The default configuration will make sure SAPUI5 is registered under the path “`/resources`”.	

```
{
  "welcomeFile": "inde
  "routes": [
    {
      "path": "/resour
      "target": {
        "type": "servi
        "name": "sapui
        "entryPath": "
      },
      "description": "
```

neo-app.json

```
{
   welcomeFile: "index.html",
   routes: [
      {
         …
      },
      {
         "path": "/destinations/ES5",
         "target": {
            "type": "destination",
               "name": "ES5"
               },
         "description": "ES5 Demo Service"
      }
   ]
}
```

Add the destination “ES5” to the `neo-app.json` configuration file. It contains all project settings for SAP Web IDE and is created in the root folder of your project. It is a JSON format file consisting of multiple configuration keys. The most important setting for you to configure is the path where the OpenUI5 runtime is located when starting the app.

Below “routes”, add a new configuration entry for the destination ES5. The destination itself is configured inside the SAP HANA Cloud Platform Cockpit. With this configuration entry in the `neo-app.json` file you are able to use it inside your application project.

webapp/manifest.json

```
{
  "_version": "1.1.0",
  "sap.app": {
    …
    "applicationVersion": {
      "version": "1.0.0"
   },
    "dataSources": {
      "ES5": {
        "uri": "/destinations/ES5/sap/opu/odata/IWBEP/GWSAMPLE_BASIC/",
        "type": "OData",
        "settings": {
          "odataVersion": "2.0"
        }
      }
    }
  },
  "sap.ui": {
  …
  },
  "sap.ui5": {
  …
  "models": {
    …
    "helloPanel": {
      "type": "sap.ui.model.json.JSONModel",
      "uri": "model/HelloPanel.json"
    },
    "" : {
      "dataSource": "ES5"
    }
  }
}
```

Create a Model Configuration in the App Descriptor. In the `sap.app` section we add the data source configuration for the OData service. With the key `ES5` we specify a configuration object that allows automatic instantiation of this model. We specify the URI by combining the name of the destination we configured in the `neo-app.json` file above with the path to the service. So `/destinations/ES5` references the destination that we have configured in the SAP HANA Cloud Cockpit and `/sap/opu/odata/IWBEP/GWSAMPLE_BASIC/` is the path to the service endpoint. In addition, we specify the type of the service (OData) and the model version (2.0).

The complete URI at run time looks like this:
https://sapes5.sapdevcenter.com/sap/opu/odata/IWBEP/GWSAMPLE_BASIC/

In the `models` section under `sap.ui5`, we add a second entry with an empty key. The value `ES5` is a reference to the data source section that we specified above. This configuration allows the component to retrieve the technical information for this model during the start-up of the app.

Our component will then automatically create an instance of `sap.ui.model.odata.v2.ODataModel` according to the settings we specified above, and make it available as the default model on the component.

webapp/view/App.view.xml

```
…
<Page title="OpenSAP - Developing with SAPUI5">
  <IconTabBar
    id="idTopLevelIconTabBar"
    selectedKey="db"
    …>
    <items>
      …
      <IconTabFilter
        text="{i18n>dataBindingFilter}" key="db">
        <content>
          <List
            headerText="{i18n>productListTitle}"
            items="{/ProductSet}">
            <items>
              <ObjectListItem
                title="{Name}"
                number="{Price}"
                intro="{ProductID}" />
            </items>
          </List>
        </content>
      </IconTabFilter>
    </items>
  </IconTabBar>
</Page>
…
```

We add a new `IconTabFilter` with the key `db` and the translated text `dataBindingFilter` to the view of our app. additionally, we set the `selectedkey` property on the `IconTabBar` control to `db` so that this tab is opened by default when we start the app.

The content of the `IconTabFilter` is a list control with a custom header text. The item aggregation of the list is bound to the root path `ProductSet` of the OData service. And since we defined a default model, we do not have to prefix the binding definitions with a model name and can just write the path.

In the `items` aggregation, we define the template for the list that will be automatically repeated for each product of our service data. More precisely, we use an `ObjectListItem` to create a control for each aggregated child of the `items` aggregation. The `title` property of the list item is bound to properties of a single product. This is achieved by defining a relative path (without `/` in the beginning). This works because we have bound the `items` aggregation absolutely via `items={/ProductSet}` to the collection of products.

webapp/i18n/i18n.properties

```
…
# Tabs
dataBindingFilter = Data Binding
…
# Data Binding Content
productListTitle = Products
```

Add the new translation texts to your resource bundle file.

You should now see a list of products fetched from the ES5 backend system in your app!

3 (OPTIONAL) USE MOCK SERVER IF YOU HAVE A SLOW INTERNET CONNECTION

Note:
If the service connection is too slow for you during these exercises, you can manually switch to a mock server and simulate the backend requests by doing the following changes in your project. This is a workaround that will create random test data in your app. The `MockServer` will be explained later in the course in more detail and would be set up a bit differently in a real project.

webapp/index.html

```
…
  <script>
    sap.ui.getCore().attachInit(function () {
      // workaround begin: start a mock server
      jQuery.sap.require("sap.ui.core.util.MockServer");
      var oMockServer = new sap.ui.core.util.MockServer({
        rootUri: "/destinations/ES5/sap/opu/odata/IWBEP/GWSAMPLE_BASIC/"
      });
      oMockServer.simulate("metadata.xml", {
        sMockdataBaseUrl: "mockdata",
        bGenerateMissingMockData: true
      });
      oMockServer.start();
      // workaround end

      new sap.ui.core.ComponentContainer({
        name: "sap.ui.demo.wt"
      }).placeAt("content");
    });
  </script>
…
```

Add the code above to your index file to start a local mock server that will intercept and simulate requests to your backend service. Don’t worry, we will explain it later in the course. This is just a workaround for now.

webapp/metadata.xml (NEW)

```
<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="1.0"
  xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
  xmlns:sap="http://www.sap.com/Protocols/SAPData">
  <edmx:DataServices m:DataServiceVersion="2.0">
    <Schema
      xmlns="http://schemas.microsoft.com/ado/2008/09/edm"
      Namespace="/IWBEP/GWSAMPLE_BASIC"
      xml:lang="en"
      sap:schema-version="1">
      <EntityType Name="Product">
        <Key>
          <PropertyRef Name="ProductID"/>
        </Key>
        <Property Name="ProductID" Type="Edm.String" Nullable="false"
MaxLength="10"/>
        <Property Name="TypeCode" Type="Edm.String" Nullable="false"
MaxLength="2"/>
        <Property Name="Category" Type="Edm.String" Nullable="false"
MaxLength="40"/>
        <Property Name="Name" Type="Edm.String" Nullable="false"
MaxLength="255"/>
        <Property Name="NameLanguage" Type="Edm.String" MaxLength="2"/>
        <Property Name="Description" Type="Edm.String" MaxLength="255"/>
        <Property Name="DescriptionLanguage" Type="Edm.String"
MaxLength="2"/>
        <Property Name="SupplierID" Type="Edm.String" Nullable="false"
MaxLength="10"/>
        <Property Name="SupplierName" Type="Edm.String" MaxLength="80"/>
        <Property Name="TaxTarifCode" Type="Edm.Byte" Nullable="false"/>
        <Property Name="MeasureUnit" Type="Edm.String" Nullable="false"
MaxLength="3"/>
        <Property Name="WeightMeasure" Type="Edm.Decimal" Precision="13"
Scale="3"/>
        <Property Name="WeightUnit" Type="Edm.String" MaxLength="3"/>
        <Property Name="CurrencyCode" Type="Edm.String" Nullable="false"
MaxLength="5"/>
        <Property Name="Price" Type="Edm.Decimal" Precision="16"
Scale="3"/>
        <Property Name="Width" Type="Edm.Decimal" Precision="13"
Scale="3"/>
        <Property Name="Depth" Type="Edm.Decimal" Precision="13"
Scale="3"/>
        <Property Name="Height" Type="Edm.Decimal" Precision="13"
Scale="3"/>
        <Property Name="DimUnit" Type="Edm.String" MaxLength="3"
sap:label="Dimension Unit"/>
        <Property Name="CreatedAt" Type="Edm.DateTime" Precision="7"/>
        <Property Name="ChangedAt" Type="Edm.DateTime" Precision="7"
ConcurrencyMode="Fixed"/>
      </EntityType>
      <EntityContainer Name="/IWBEP/GWSAMPLE_BASIC_Entities"
m:IsDefaultEntityContainer="true">
```

```
        <EntitySet Name="ProductSet"
EntityType="/IWBEP/GWSAMPLE_BASIC.Product"/>
      </EntityContainer>
    </Schema>
  </edmx:DataServices>
</edmx:Edmx>
```

Add the file `metadata.xml` that contains the service interface with the product entity to be mocked.

Related Information
Walkthrough Tutorial - Step 26: Remote OData Service