Week 3 Unit 1

Creating Apps with Templates

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

Table of Contents

[1 Extend the ES5 Destination for Usage in SAP Web IDE 2](#_Toc449111388)

[2 Create New Project from the Worklist Template 3](#_Toc449111389)

[3 Run the App in Your Workspace 8](#_Toc449111390)

[4 (Optional) Configure Local Mock Data 11](#_Toc449111391)

[5 Run the App in HANA Cloud Platform 14](#_Toc449111392)

[6 (Optional) Where to Find What in the Project? 18](#_Toc449111393)

## Preview

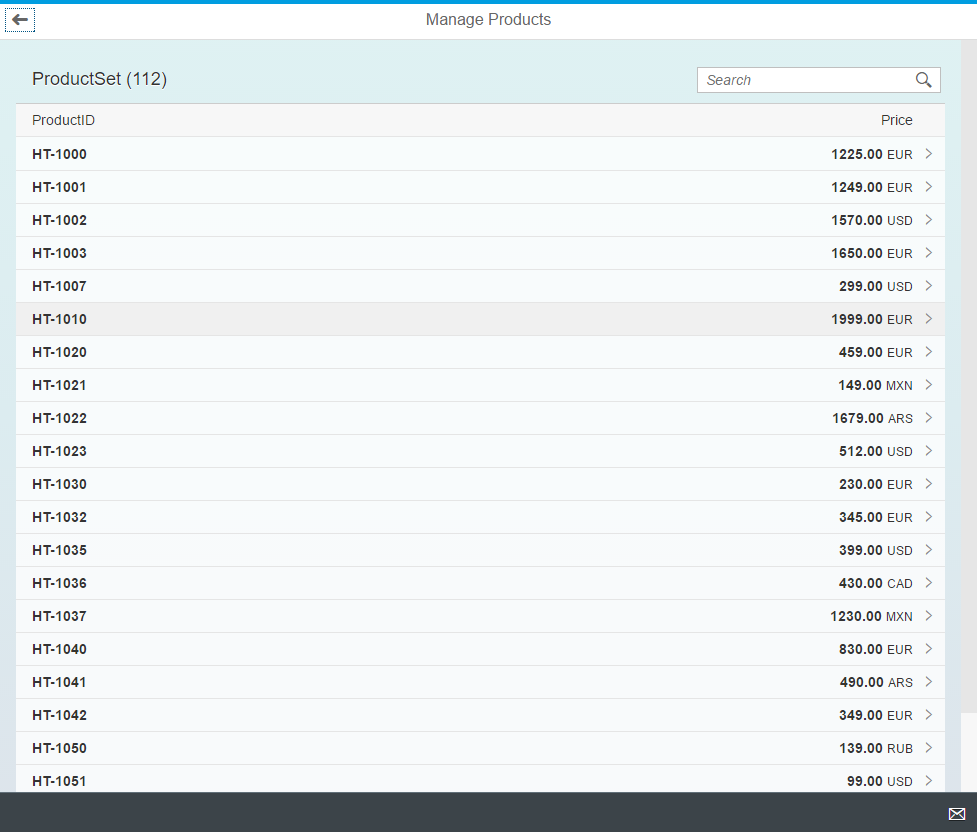


Figure 1 - The initial worklist template

# Extend the ES5 Destination for Usage in SAP Web IDE

In the first step of this unit we will enhance the destination to the ES5 system with properties for SAP Web IDE. This is required for the next steps to use the destination in the template generation.

| Explanation | **Screenshot** |
| --- | --- |
| 1. Open the **SAP HANA Cloud Platform Cockpit**   [https://account.hanatrial.ondemand. com/cockpit](https://account.hanatrial.ondemand.com/cockpit)   1. Select **Destinations** in the navigation menu on the left 2. Select the **ES5** destination created in week 0 – unit 1 3. Press the **Edit** button |  |
| 1. Add 3 additional properties with the help of the **New Property** button:  * **WebIDEEnabled** = **true** * **WebIDESystem** = **ES5** * **WebIDEUsage** = **odata\_gen**  1. **Save** the Destination |  |

# Create New Project from the Worklist Template

In this step we will use the “SAP Fiori Worklist Application” template to generate an application that is already bound to the ES5 OData service.

**Important Note about Templates**: During the template generation a set of files is generated into your workspace that serve as a starting point for developing a productive application. These files are a copy and as such decoupled from the template code. There is not automatic approach to receive updates when the template code is changed later by SAP.

| Explanation | **Screenshot** |
| --- | --- |
| 1. If you have **SAP Web IDE** already open you need to restart it now by pressing the **Refresh** button in your chrome browser |  |
| 1. If creating and starting the Web IDE as new:   - create it for Fiori app development  - choose “Start from template” |  |
| 3. Choose “SAP Fiori application” type |  |
| 4. Choose “Worklist Page” |  |
| 5. fill the input fields for Data Source and Service Selection  connect to a system  ES5  /sap/opu/odata/IWBEP/GWSAMPLE\_BASIC/ |  |
| 6. Choose entity “ProductSet” without navigation entity |  |
| **7.** fill project attributes  Observe that latest SAPUI5 version is automatically selected for you.  **Note:**  In this openSAP course you should work with the **“Recommended” version** (i.e. the latest released version)  But when developing productive applications later on you have to choose the UI5 version available in your target system. |  |
| 8. deployment configuration |  |
| On the **Template Customization** step enter this data   1. **Application Settings**  * **Type = Standalone App**   **Note:** In this course, we will not build SAP Fiori launchpad apps, this setting will enable the app to run standalone.   * **Title** = **Manage Products** * **Namespace** = **opensap.manageproducts**      1. **Data Binding**  * **Object Collection** = **ProductSet** * **Object Collection ID** = **ProductID** * **Object Title** = **ProductID** * **Object Numeric Attribute** = **Price** * **Object Unit of Measure** = **Currency Code** |  |
| 1. Check that a new folder **ManageProducts** is added to your **Workspace** and contains these folders and files |  |

# Run the App in Your Workspace

In this step we will run the generated app from the Web IDE workspace. There are two options:

1. Run the app against the real OData service on the ES5 system
2. Run the app with the SAPUI5 mock server. In this case the server requests are intercepted on the client and their response simulated with faked data.

**Note:**Ideally you develop against the real backend system ES5. But if you have no or very slow network connection to the ES5 system - hosted in Europe - you can also run the app with mock data by selecting the run configuration “App (Mock Server)”.

| Explanation | **Screenshot** |
| --- | --- |
| **Run the app against the real OData service (default)**   1. Select the folder **ManageProducts** |  |
| 1. Open the **Run** menu and choose the **App** entry to run the application |  |
| 1. Now the application opens up in a new browser tab. This is the screen that you should see |  |
| 1. Press on the search field and enter the search term (e.g. “**20”**) 2. Press **Enter** on your keyboard 3. Observe that the product list is filtered |  |
| 1. Navigate to the product details by pressing on the table row with your mouse. 2. Observe that only basic data of the product is being displayed so far. |  |
| **Run the app with the SAPUI5 mock server (optional)?**   1. Select the folder **ManageProducts** |  |
| 1. Open the **Run** menu and choose the **App (Mock Server)** entry to run the application without reading data from a server |  |
| 1. Observe that the app is now filled with auto-generated mock data   **Note:** Not all features of the app can be simulated with auto-generated mock data. You can add local mock data (see next step) to improve the behavior. |  |

# (Optional) Configure Local Mock Data

To run the app with local mock data you must update the metadata and place mocked JSON data in your app project. These files will be loaded by the mock server upon startup of the app.

**Note:**Some features that we use throughout the course cannot be simulated with auto-generated mock data. If you want to use the mock server you must perform the steps in this chapter. If you work with the real back end system you can skip them.

**webapp/localService/metadata.xml**

…

<EntityType Name="Product" sap:content-version="1">

<NavigationProperty Name="ToSalesOrderLineItems" …/>

**<NavigationProperty Name="ToSupplier" Relationship="/IWBEP/GWSAMPLE\_BASIC.Assoc\_Products\_BusinessPartner" FromRole="FromRole\_Assoc\_Products\_BusinessPartner" ToRole="ToRole\_Assoc\_Products\_BusinessPartner"/>**

</EntityType>

…

<Association Name="Assoc\_BusinessPartner\_Products" sap:content-version="1">…</Association>

**<Association Name="Assoc\_Products\_BusinessPartner" sap:content-version="1">**

**<End Type="/IWBEP/GWSAMPLE\_BASIC.BusinessPartner" Multiplicity="1" Role="FromRole\_Assoc\_BusinessPartner\_Products"/>**

**<End Type="/IWBEP/GWSAMPLE\_BASIC.Product" Multiplicity="\*" Role="ToRole\_Assoc\_BusinessPartner\_Products"/>**

**<ReferentialConstraint>**

**<Principal Role="FromRole\_Assoc\_Products\_BusinessPartner">**

**<PropertyRef Name="SupplierID"/>**

**</Principal>**

**<Dependent Role="ToRole\_Assoc\_Products\_BusinessPartner">**

**<PropertyRef Name="BusinessPartnerID"/>**

**</Dependent>**

**</ReferentialConstraint>**

**</Association>**

…

<EntityContainer Name="/IWBEP/GWSAMPLE\_BASIC\_Entities" … >

**…**

<AssociationSet Name="Assoc\_BusinessPartner\_Products\_AssocSet" … >…</AssociationSet>

**<AssociationSet Name="Assoc\_Products\_BusinessPartner\_AssocSet" Association="/IWBEP/GWSAMPLE\_BASIC.Assoc\_Products\_BusinessPartner" sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:content-version="1">**

**<End EntitySet="ProductSet" Role="FromRole\_Assoc\_Products\_BusinessPartner"/>**

**<End EntitySet="BusinessPartnerSet" Role="ToRole\_Assoc\_Products\_BusinessPartner"/>**

**</AssociationSet>**

**…**

</EntityContainer>

…

The mock server does not support bi-directional associations. To establish a link between the product and supplier entity for the mock server we need to add a new Association and AssociationSet to the metadata and update the NavigationProperty from the product to the supplier.

In the entity type Product modify the navigation property ToSupplier to use a new association Assoc\_Products\_BusinessPartner. Add a new association to the metadata as specified above. In the EntityContainer section add a new AssociationSet to complete the configuration.

**Adding the local mock data**

| Explanation | **Screenshot** |
| --- | --- |
| 1. In SAP Web IDE, go to **File > git > Clone Repository** and enter only the following URL in the configuration dialog:   [**https://github.com/SAP/openSAP-ui5-course.git**](https://github.com/SAP/openSAP-ui5-course.git)  **Note: Authentication** Leave all other fields of the dialog empty. No authentication is needed for this Git repository.   1. The course repository will be added to your local workspace |  |
| 1. Select the folder **mockdata** below the root folder 2. Right-click on the folder and select **copy** |  |
| 1. Inside your app project select the folder **webapp/localService** 2. Right-click on the folder and select **paste** |  |
| 1. Run the app with mock data:    * Open the **Run** menu and choose the **App (Mock Server)**  - or -    * Run the file **webapp/test/mockServer.html** in your app project |  |
| 1. Observe that the app is now filled with the mock data from the JSON files.   **Note:** The mock data in the JSON files is a local copy of the server’s data. The mock server reads it automatically if the files have the same name as the OData entities specified in the metadata.xml file. |  |

# Run the App in HANA Cloud Platform

| Explanation | **Screenshot** |
| --- | --- |
| 1. Select the folder **ManageProducts** |  |
| 1. Press the right mouse button on the **ManageProducts** folder 2. Choose **Deploy** 3. Choose **Deploy to SAP HANA Cloud Platform** |  |
| 1. Enter your SAP Hana Cloud Platform developer **UserID** and **Password** and press the **Login** button |  |
| 1. Only press the **Deploy** button | C:\Users\D050079\AppData\Local\Temp\SNAGHTML503505.PNG |
| After some time a success dialog opens:   1. Press on the **Open the active version of the application** link |  |
| 1. Observe that your apps opens up in a new browser tab   **Note:** This version of your application is now deployed to SAP HANA Cloud Platform and can be accessed by anyone having the URL. For example, you can share it in the forum if you have a question about your app coding. |  |
| 1. Bookmark the app’s URL in your chrome browser for easier access. |  |
| In case you have lost the app’s URL:   1. Open **HCP Cockpit** 2. Select **HTML5 Applications** 3. Select **manageproducts** |  |
| 1. Press the **Application URL** | C:\Users\D050079\AppData\Local\Temp\SNAGHTML555304.PNG |
| 1. Observe that a **dist** folder has been added to the app’s folder in Web IDE.   **Note:** SAP Web IDE automatically generates a performance optimized version of your application that gets deployed to SAP Hana Cloud Platform.  Do not modify the files in this folder but work on the original files in the **webapp** folder. |  |
| Deploy the app again:   1. Press the right mouse button on the **ManageProducts** folder 2. Choose **Deploy** 3. Choose **Deploy to SAP HANA Cloud Platform** |  |
| 1. Observe that on deploying the app a second time a new version will be generated and activated by default   **Note:** This is just fine. Versions of your application are automatically created so that you can decide which version you want to activate in your SAP Hana Clould Platform developer account | C:\Users\D050079\AppData\Local\Temp\SNAGHTML582742.PNG |

You have now successfully completed this unit. The next chapter just contains some background information on the generated files and can be read optionally.

# (Optional) Where to Find What in the Project?

* The file webapp/index.html is launched on choosing Run > App
* The file webapp/test/mockServer.html is launched on choosing Run > App (Mock Server)
* The ES5 destination has been added to the neo-app.json

{

…

}, {

"path": **"/destinations/ES5"**,

"target": {

"type": "destination",

"name": "ES5"

},

"description": "ES5"

}],

* The service URL has been added to the webapp/manifest.json

{

"\_version": "1.2.0",

"sap.app": {

…

"dataSources": {

"mainService": {

"uri": **"/destinations/ES5/sap/opu/odata/IWBEP/GWSAMPLE\_BASIC/"**,

"type": "OData",

"settings": {

"odataVersion": "2.0",

"localUri": "localService/metadata.xml"

}

}

},

* The build settings are in the .project.json

…

**"build"**: {

"targetFolder": "dist",

"sourceFolder": "webapp",

"excludedFolders": [

"test"

],

"excludedFiles": [

"test.html"

],

"buildRequired": false,

"lastBuildDateTime": "Tue, 17 May 2016 08:15:30 GMT"

},

…

* The link to the template’s documentation can be found in the readme.txt

#### WELCOME ####

This is your copy of the SAPUI5 Worklist Freestyle Application Template.

You can find the template version in the .project.json - file in your workspace

Standalone runnable files (\*.html) are located in the test-folder

This application is ready for client-side build in the SAP Web IDE and deployment to ABAP/HCP repositories

Documentation of all template-app features can be found in the SAPUI5 demokit here:

https://sapui5.hana.ondemand.com/#docs/guide/a460a7348a6c431a8bd967ab9fb8d918.html

#### Happy Development! ####