End to End Guide to

Building OData Service from scratch using the

SAP NetWeaver Gateway Service Builder

&

Consuming it via SAPUI5 application

End to End Guide to building OData Service and Consuming it via SAPUI5 application

Hello There,

Thank you for downloading this step by step OData service application guide.

In this step by step tutorial guide I will walk you through the steps to create an OData service from scratch using the SAP Netweaver Gateway and also build an SAPUI5 application to use the created OData service

So accordingly this tutorial is split into 4 parts.
Part 1 – Create an OData service
Part 2 – Start an SAPUI5 application and connect it to SAP backend
Part 3 – Develop a Basic Hello World SAPUI5 Application
Part 4 – Develop a SAPUI5 Application to consume the OData service.

Through this tutorial I will walk you through the major aspects of the whole process. This quick guide will help you start quickly and help you enhance your own learning on this subject.

So without further delay let's get started.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

Part 1: Creating OData service using SAP Netweaver Gateway

Quick Introductions:

OData service: In SAP, OData service can be considered as a kind of web-service which can be used directly in any end user web application to access the module specific data. What data can be accessed through the service is decided during the development of such a service.

SAP Netweaver Gateway Service Builder: In SAP we use SAP Netweaver Gateway Service Builder (Tcode SEGW) to develop such OData services. This gateway is delivered by SAP in SAP Netweaver 7.2 version onwards. However it does not come automatically in built. To get it installed in your SAP version you need to install some add on packages.

I have developed this tutorial on the SAP Netweaver 7.4 version so when you try this tutorial on your system some of your screens may look a bit different.

ESPM Demo Model – Enterprise Sales and Procurement Model: SAP Netweaver delivers a sample data model which we can use to develop demo applications. This is known as the ESPM or EPM Model in short.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

This data model has a number of entities as shown below

EPM Data Status	
Products	115
Employees	33
Business Partners	45
Sales Orders	20
Goods Issue Records	0
Sales Order Invoices	0
Purchase Orders	20
Goods Receipt Records	0
Purchase Order Invoices	0

We will use this data model to base this tutorial on.

OK!! With this brief introduction we know some basics enough so start our adventure.

Moving on **Let's create an OData service based on the SAP Netweaver ESPM (Enterprise Sales and Procurement Model) Demo Model.**

Disclaimer: *I have developed this tutorial by reviewing various sample process from SCN to develop an OData service. This links are mentioned in the reference section at the end of this document.*

End to End Guide to building OData Service and Consuming it via SAPUI5 application

In this example we will develop an OData service to provide a list of products (product catalogue) stored in the SAP system. This product catalogue is part of the ESPM module of SAP.

In Simpler words, the final output of this exercise will be a web URL which could be used to get a list of products in SAP.

Later on we will use this service to develop an SAPUI5 application to design a beautiful application to display this product list.

So Let's begin

Pre-check Step

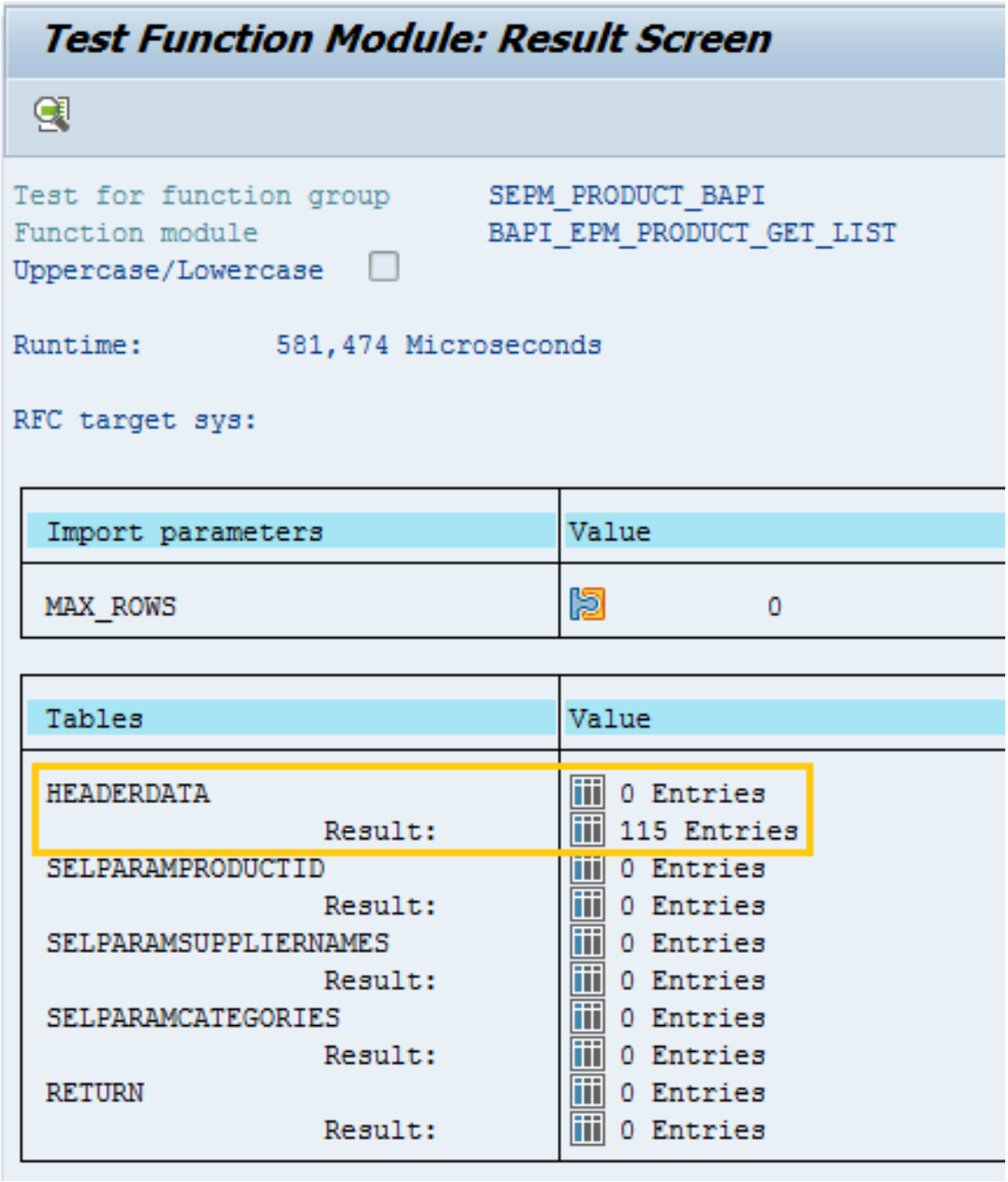
First check if the demo ESPM – Product data already exists in your SAP system.

For this execute the function module BAPI_EPM_PRODUCT_GET_LIST using Tcode SE37.

If the module does not return any data then,

Execute transaction **SEPM_DG**.

Try executing function module BAPI_EPM_PRODUCT_GET_LIST again in SE37. This time you should see the returned values in the HEADERDATA table.



In this case I can see 115 Product entries returned so we are all set to move with our actual OData service creation.

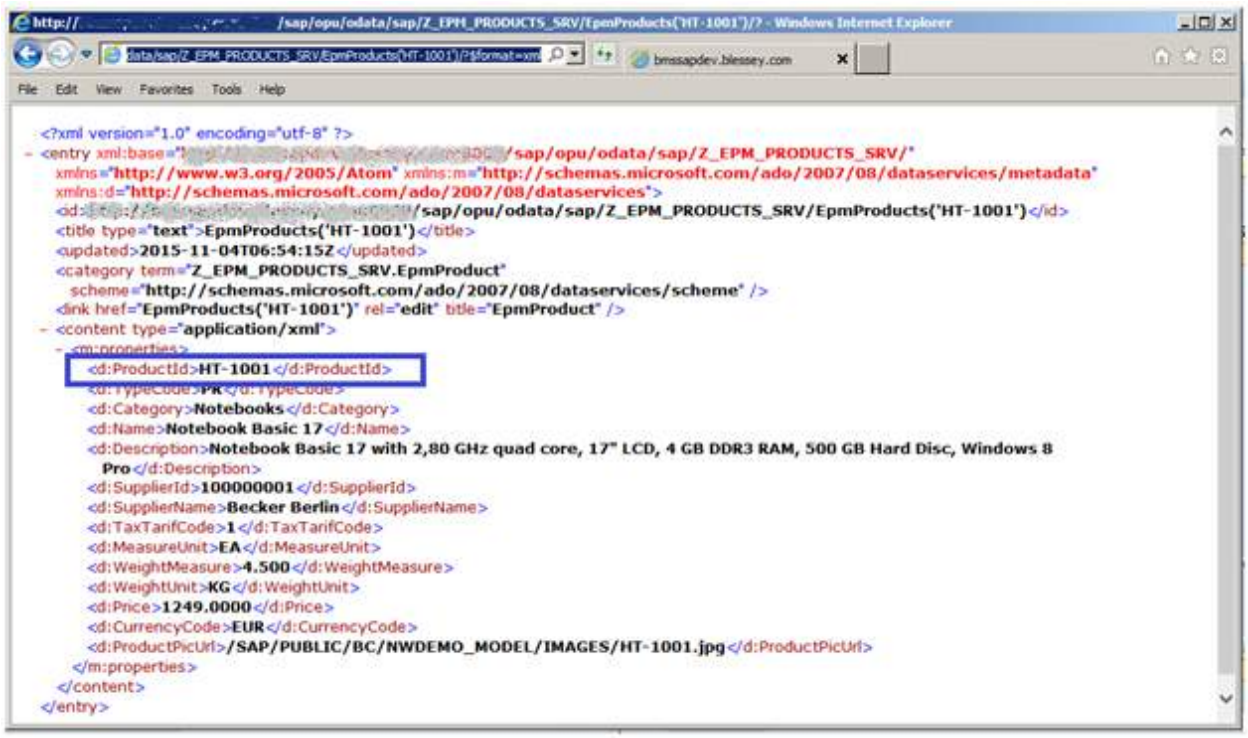
End to End Guide to building OData Service and Consuming it via SAPUI5 application

Usually when I go through tutorials I am tempted to scroll down to find out what is all this hard work finally going to look like.

So before we proceed further here I want to take a break and show you what you are finally going to have as your working product after all your hard work.

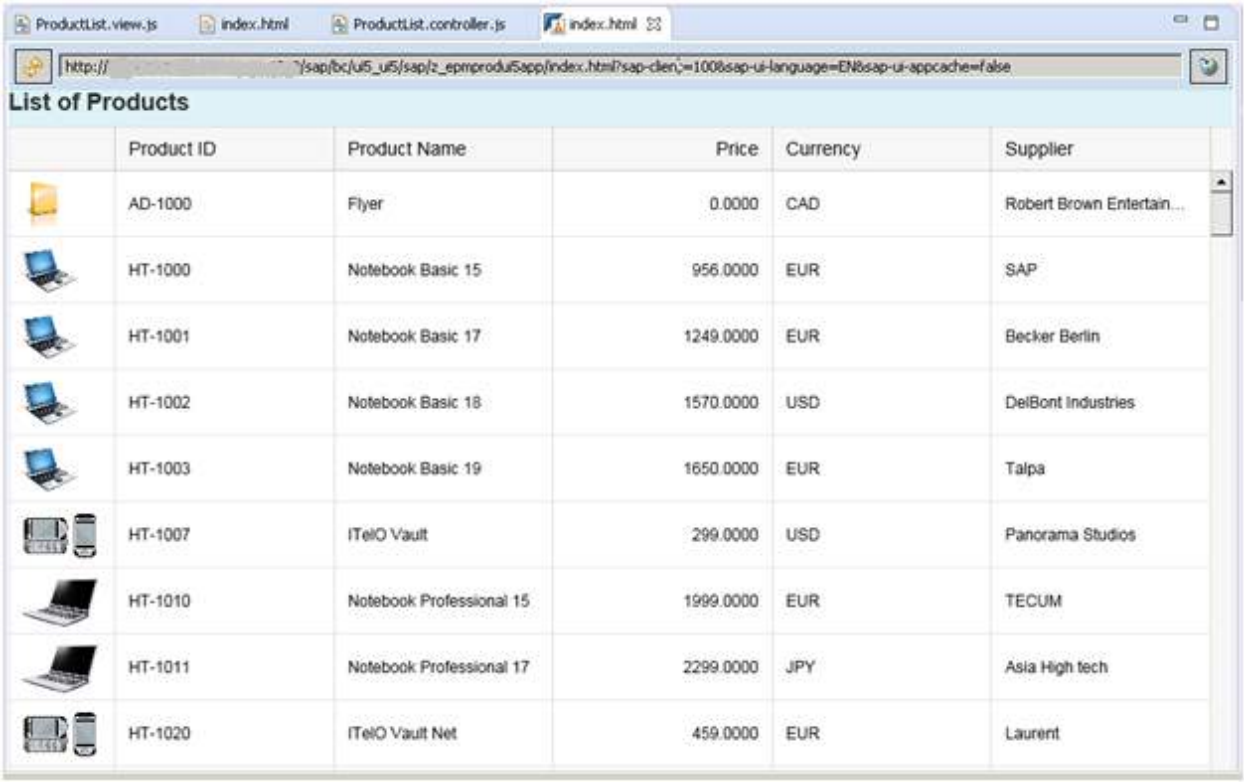
Steps we will follow

Output of Part 1.



End to End Guide to building OData Service and Consuming it via SAPUI5 application

Output of Part 2.



Product ID	Product Name	Price	Currency	Supplier
AD-1000	Flyer	0.0000	CAD	Robert Brown Entertain...
HT-1000	Notebook Basic 15	956.0000	EUR	SAP
HT-1001	Notebook Basic 17	1249.0000	EUR	Becker Berlin
HT-1002	Notebook Basic 18	1570.0000	USD	DelBont Industries
HT-1003	Notebook Basic 19	1650.0000	EUR	Talpa
HT-1007	ITelO Vault	299.0000	USD	Panorama Studios
HT-1010	Notebook Professional 15	1999.0000	EUR	TECUM
HT-1011	Notebook Professional 17	2299.0000	JPY	Asia High tech
HT-1020	ITelO Vault Net	459.0000	EUR	Laurent

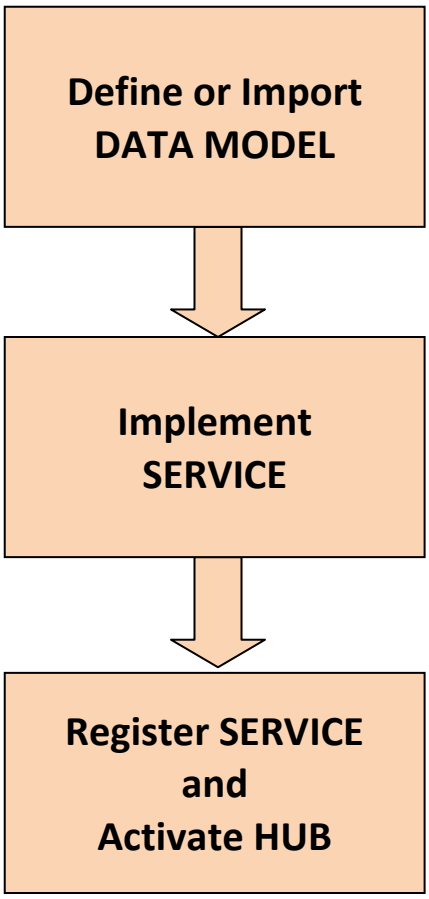
Now if you are interested to go ahead after looking at this lovely output, then let's continue.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

Building the OData Service

In SAP NetWeaver Gateway Service Builder you need to complete 3 main steps to build an OData service

1 Define or import the data model
2 Implement or generate the runtime logic for the service operations
3 Activate and run the service



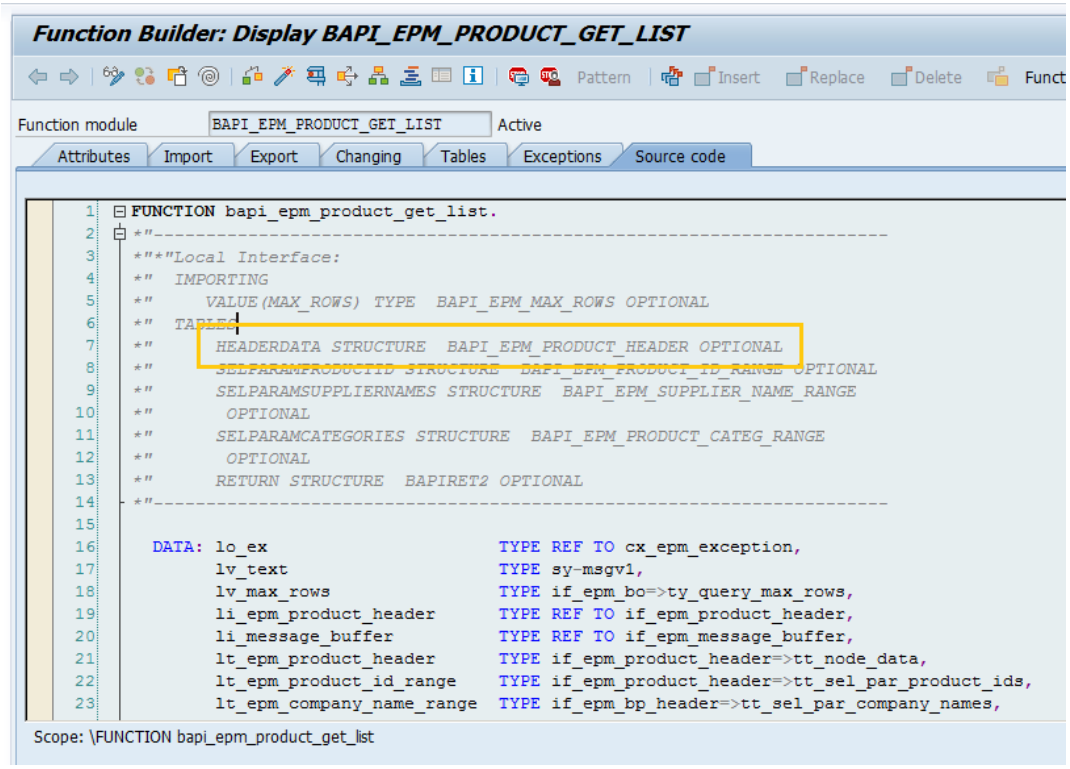
End to End Guide to building OData Service and Consuming it via SAPUI5 application

Here we will import a Data Model by using the SAP Netweaver ESPM (Enterprise Sales and Procurement Model). In this tutorial though, we will only use the Product Entity.

Step 1: Import Data Model from ABAP DDIC structure.

The final end result of all our hard work will be an Odata service which will return a product catalogue.

Since we want to finally show the product catalogue so before we import the data model into the NetWeaver Gateway Service Builder, let's look at the interface of the function module BAPI_EPM_PRODUCT_GET_LIST which had returned the list of products when we executed it in SE37.

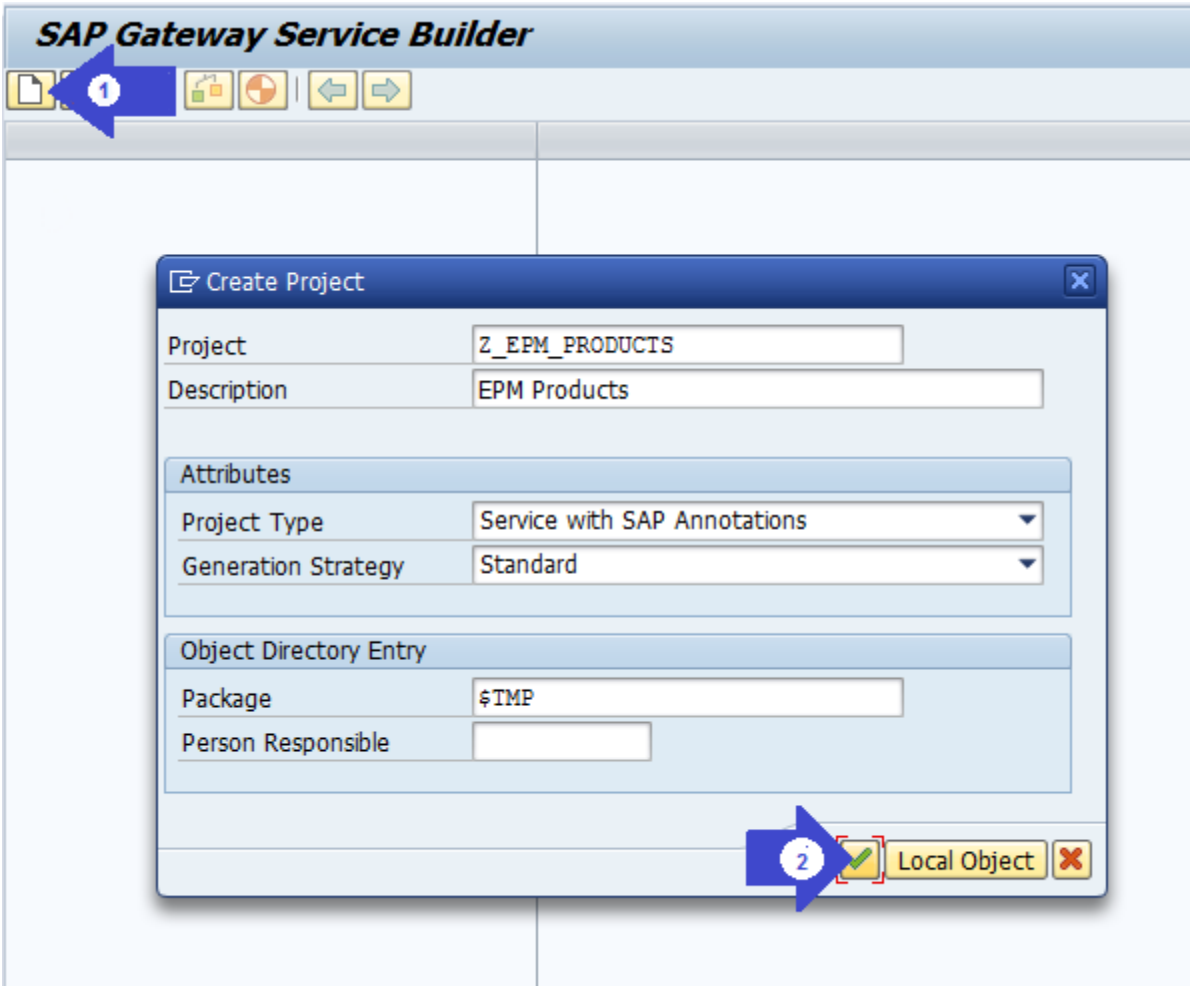


End to End Guide to building OData Service and Consuming it via SAPUI5 application

The values for the products (Product names, Category etc) were returned in the HEADERDATA Table and this is of type BAPI_EPM_PRODUCT_HEADER.

Hence we will need to refer this structure when we build our service.

Execute transaction SEGW and create a new project.



End to End Guide to building OData Service and Consuming it via SAPUI5 application

The result will be a project structure containing the data model, the service implementation, the runtime objects and the service information.
Save the generated project.

Within the service builder the data model can be imported from an external source (an EDMX file) or it can be generated from the Business Object Repository (BOR), RFC or ABAP DDIC information. Of course it can also be built up from scratch by defining all the properties, complex types, entity types and associations from scratch.

To retrieve the product catalogue returned by the BAPI into the structure of the BAPI - use the structure BAPI_EPM_PRODUCT_HEADER to import the DDIC structure.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

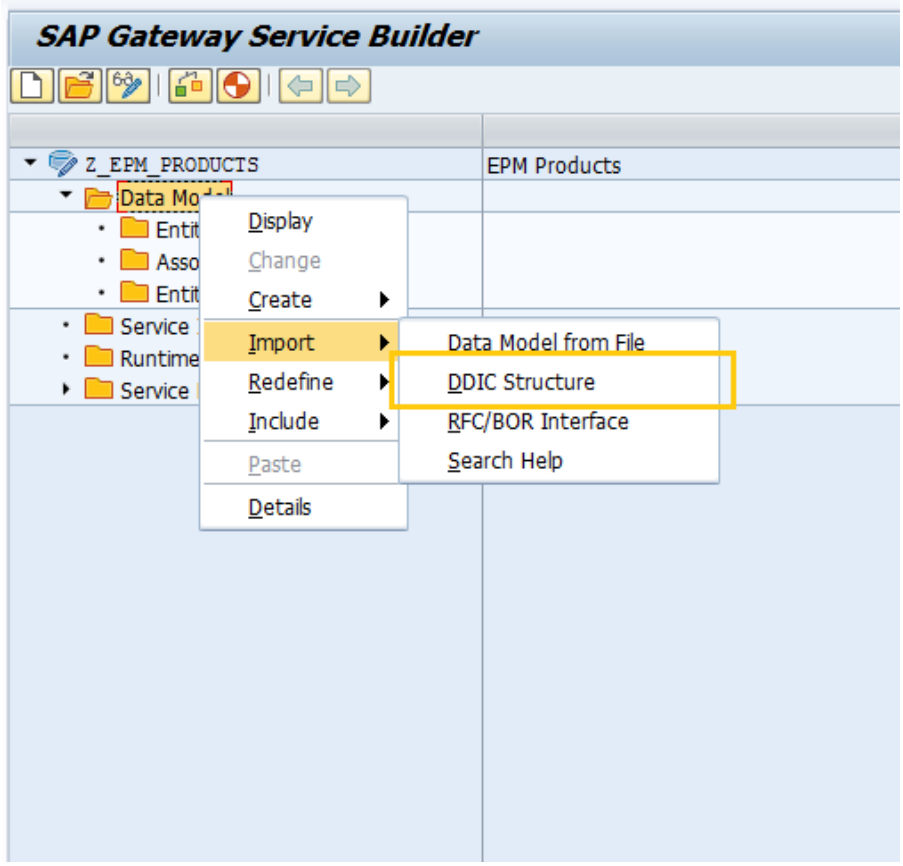
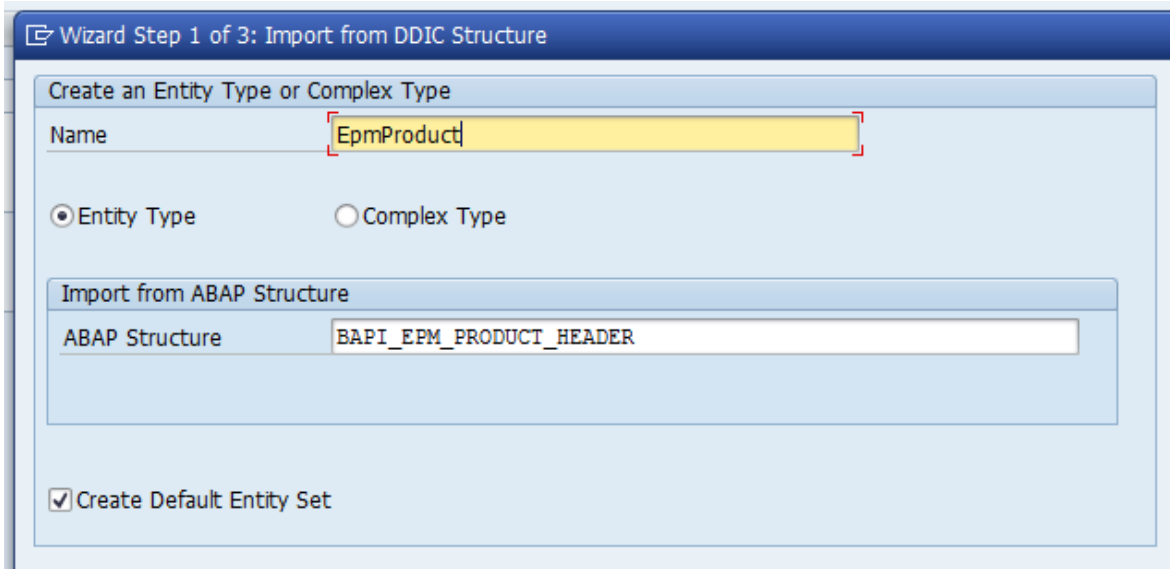
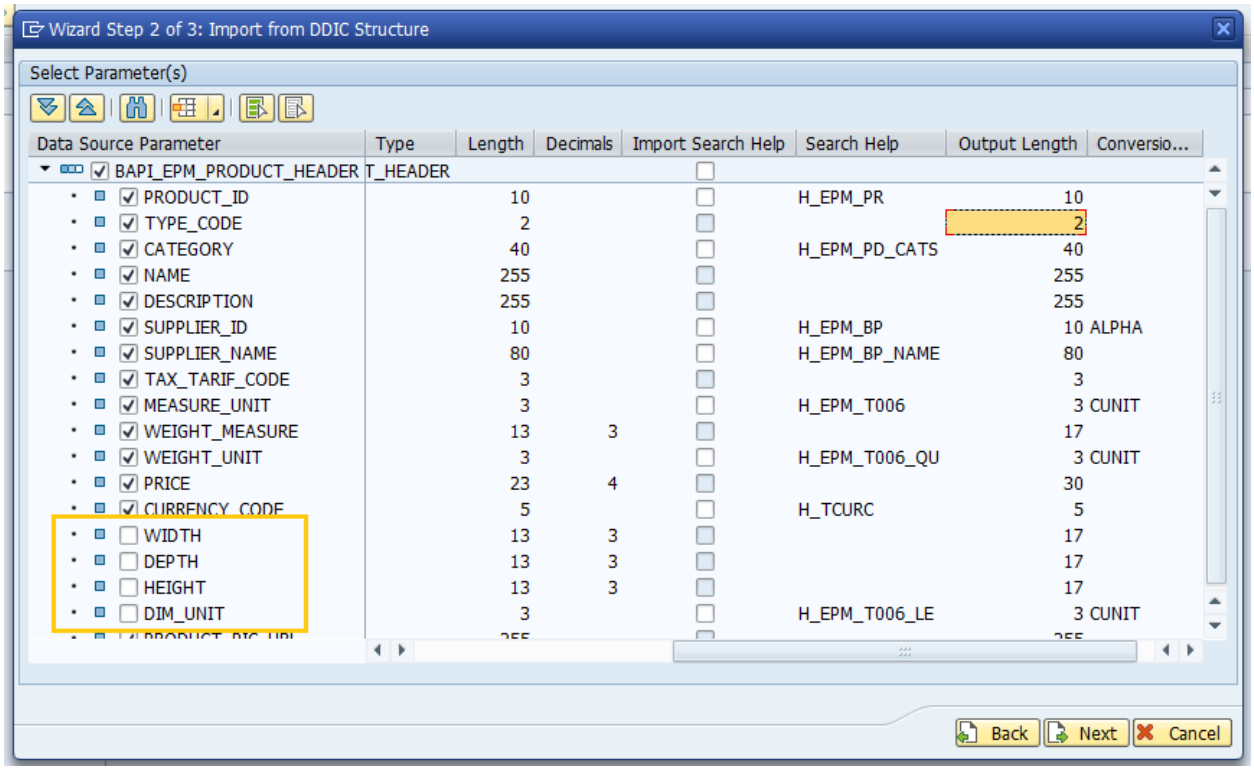


Figure 1: Import DDIC structure.



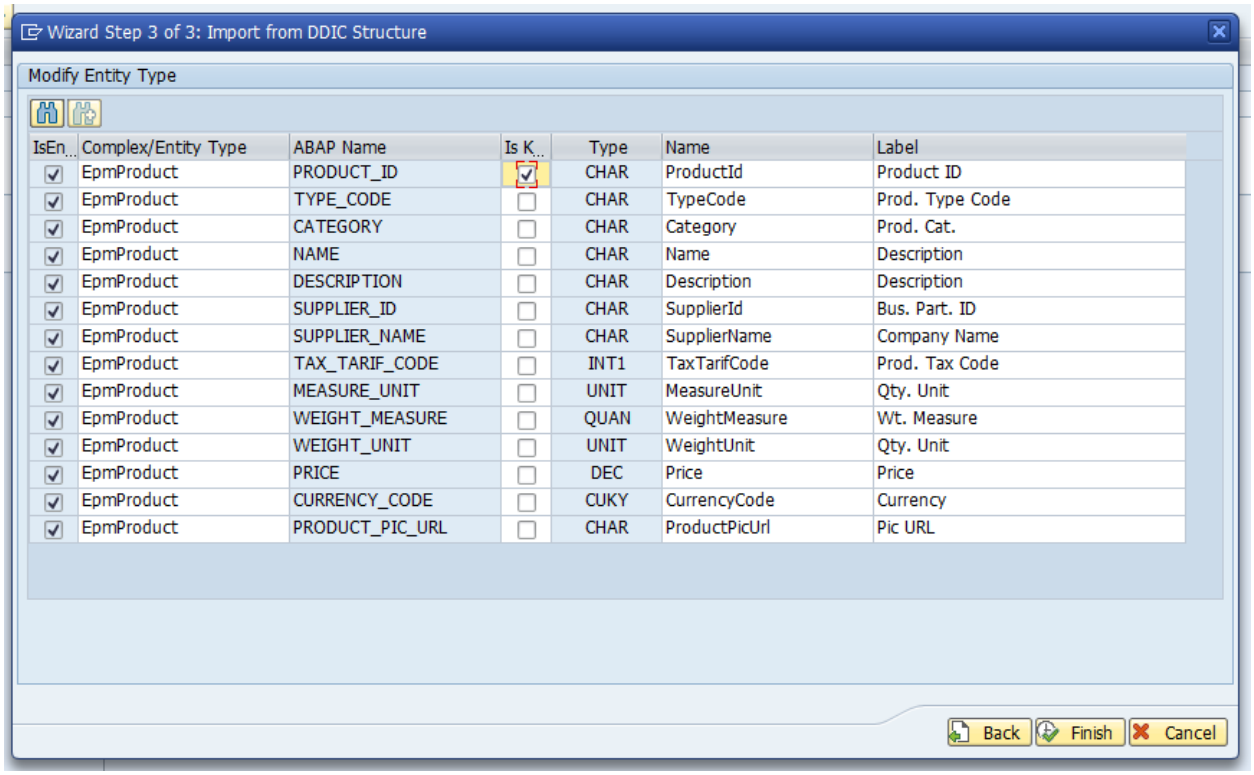
Step 1 Import DDIC Structure

End to End Guide to building OData Service and Consuming it via SAPUI5 application



Step 2 Deselect some fields you would not need in your service.

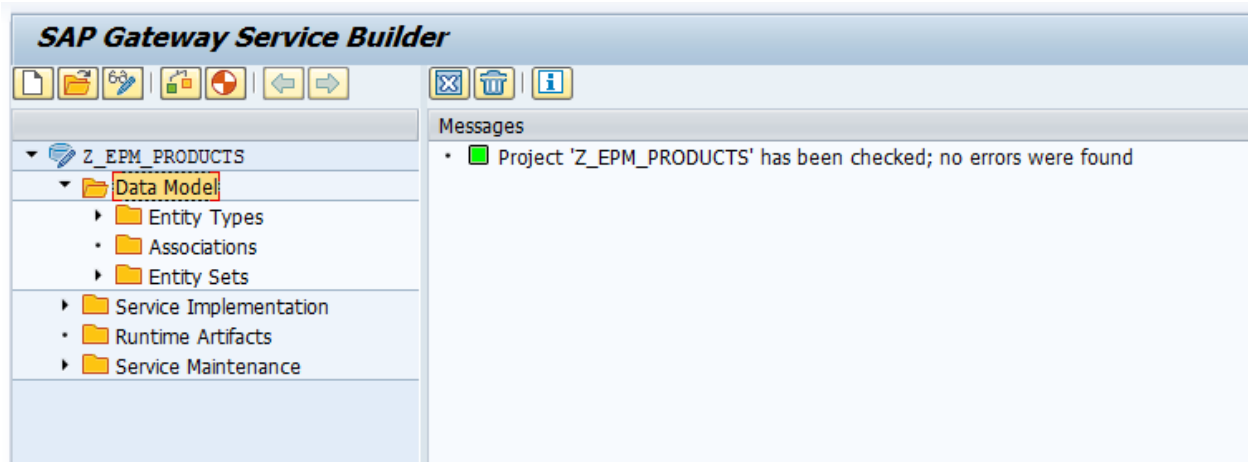
End to End Guide to building OData Service and Consuming it via SAPUI5 application



IsEn...	Complex/Entity Type	ABAP Name	Is K...	Type	Name	Label
☑	EpmProduct	PRODUCT_ID	☑	CHAR	ProductId	Product ID
☑	EpmProduct	TYPE_CODE	☐	CHAR	TypeCode	Prod. Type Code
☑	EpmProduct	CATEGORY	☐	CHAR	Category	Prod. Cat.
☑	EpmProduct	NAME	☐	CHAR	Name	Description
☑	EpmProduct	DESCRIPTION	☐	CHAR	Description	Description
☑	EpmProduct	SUPPLIER_ID	☐	CHAR	SupplierId	Bus. Part. ID
☑	EpmProduct	SUPPLIER_NAME	☐	CHAR	SupplierName	Company Name
☑	EpmProduct	TAX_TARIF_CODE	☐	INT1	TaxTarifCode	Prod. Tax Code
☑	EpmProduct	MEASURE_UNIT	☐	UNIT	MeasureUnit	Qty. Unit
☑	EpmProduct	WEIGHT_MEASURE	☐	QUAN	WeightMeasure	Wt. Measure
☑	EpmProduct	WEIGHT_UNIT	☐	UNIT	WeightUnit	Qty. Unit
☑	EpmProduct	PRICE	☐	DEC	Price	Price
☑	EpmProduct	CURRENCY_CODE	☐	CUKY	CurrencyCode	Currency
☑	EpmProduct	PRODUCT_PIC_URL	☐	CHAR	ProductPicUrl	Pic URL

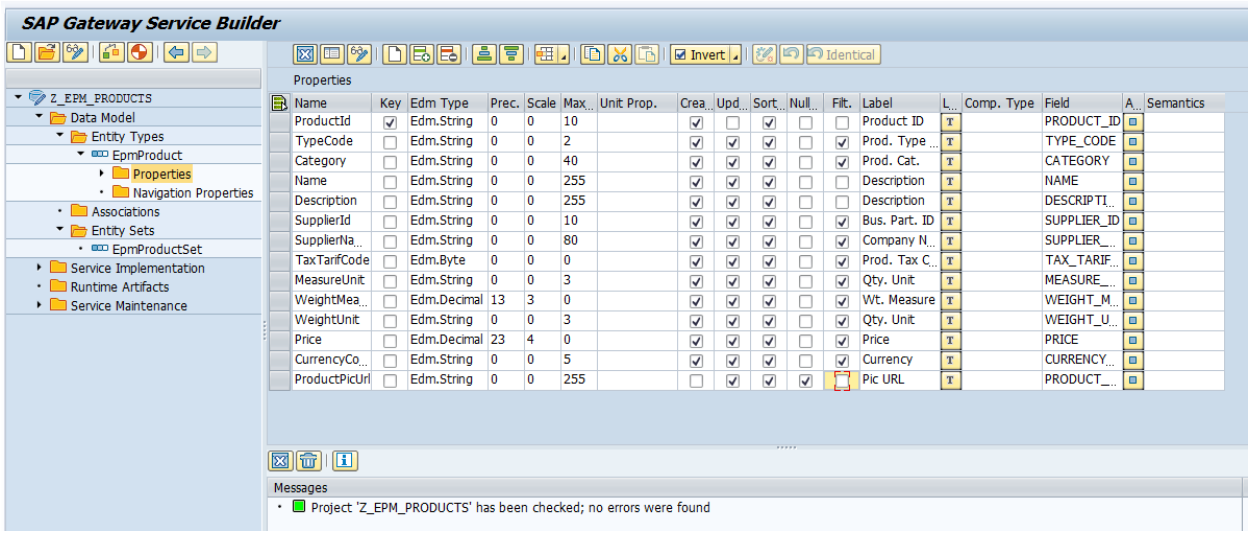
Step 3 Mark the PRODUCT_ID field as the Key for this service.

After you follow the wizard to add the DDIC structure, a new Entity Set - EmpProducSet will be added to your Project - Z_EPM_PRODUCTS Data Model.

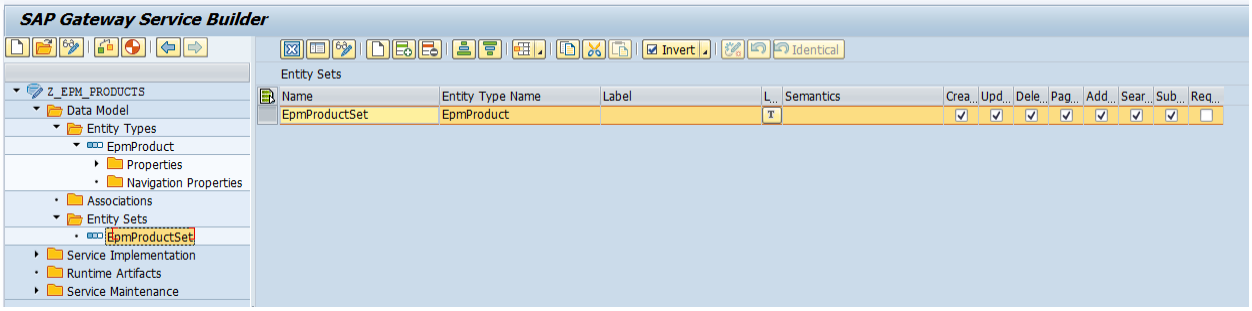


End to End Guide to building OData Service and Consuming it via SAPUI5 application

Double click the Entity Types ->EmpProduct->Properties to maintain annotations, labels and other properties of each of the entity types.



Check Entity Set 'EpmProducts' for the Entity Type 'EpmProduct'

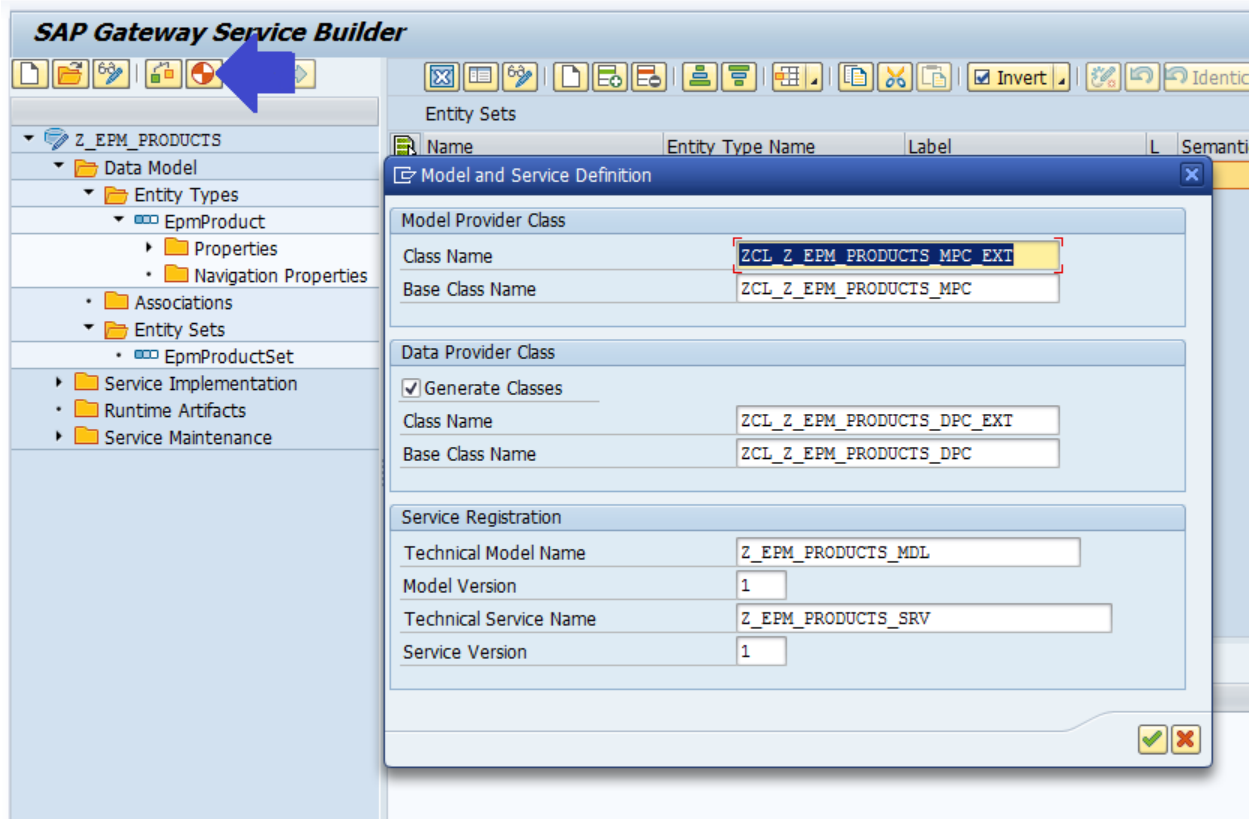


Step 2: Generate and Register the new Gateway Service

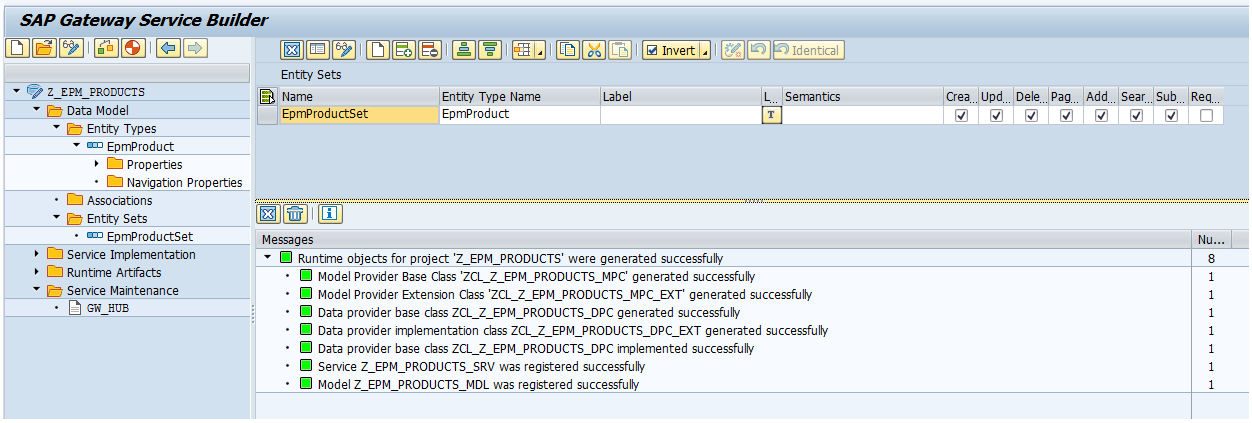
Now the service can already be generated, which means that the model and data provider classes are generated and the service is registered (names are suggested by the system, but can be overwritten):

Click the generate button to generate the service.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



Step 1 Click to generate the service



Step 2 Service generated successfully.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

Step 3: Activate and test your service.

After generating the runtime artifacts the service can be activated by double-clicking on the Service maintenance node.

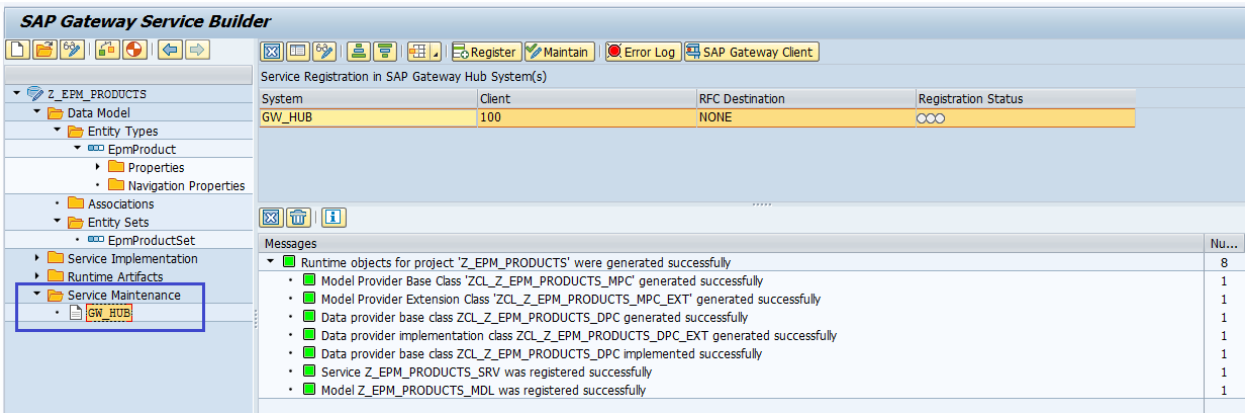
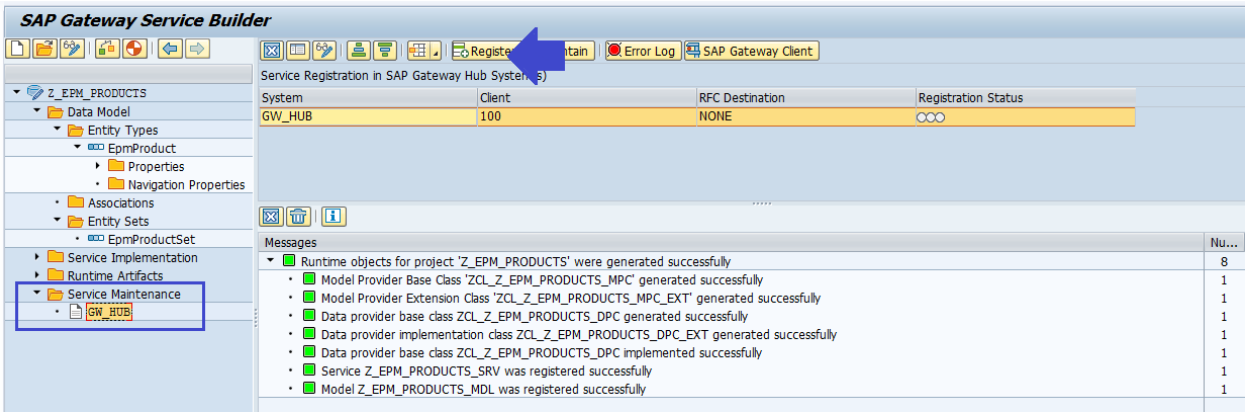


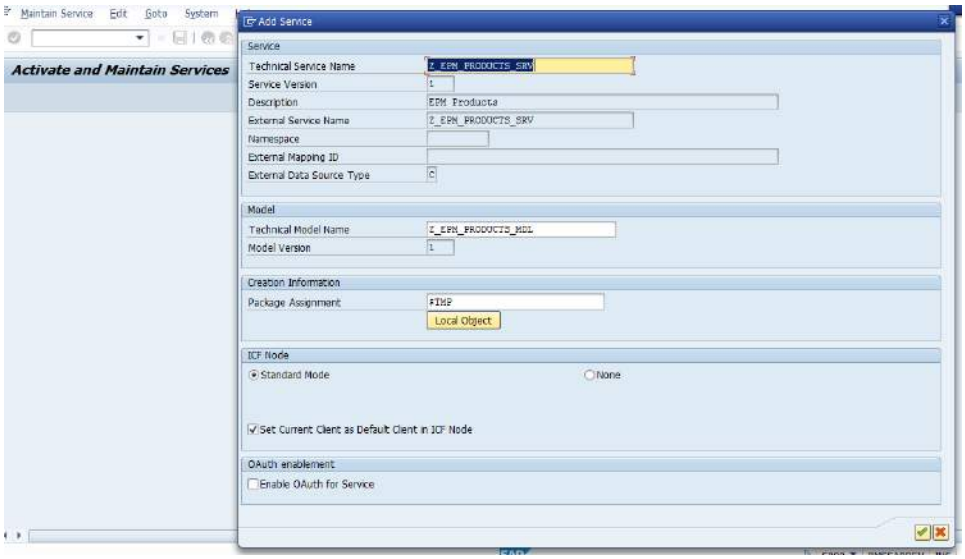
Fig: Clicking the Service Maintenance node brings up Service Registration options.

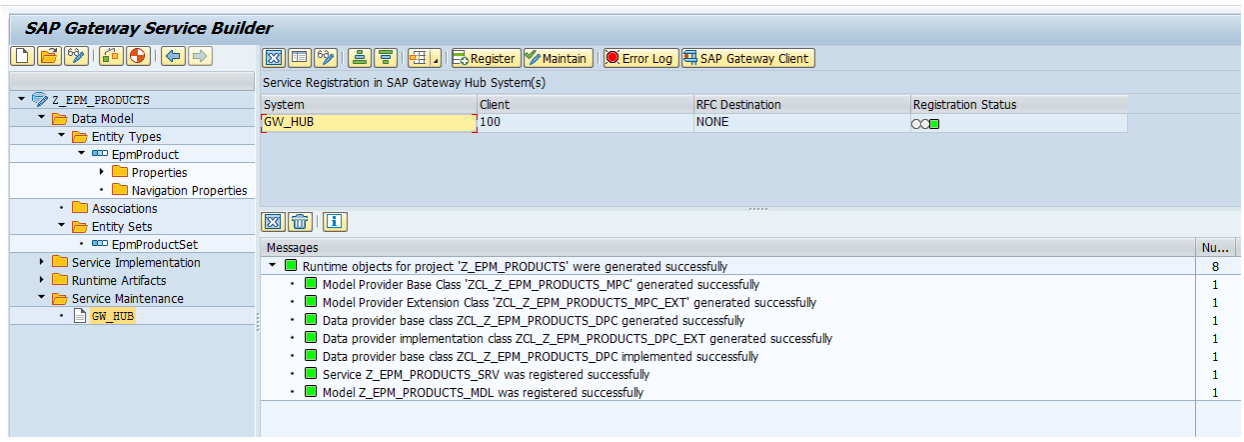
Here click on the Register button.



This will add and activate your service.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

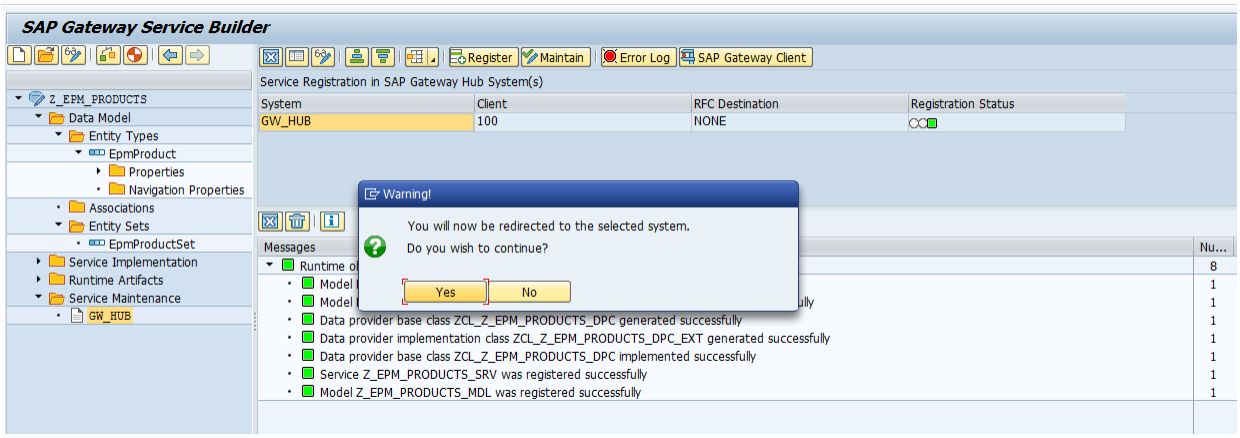




Service generated and Activated – check the registration status column is green.

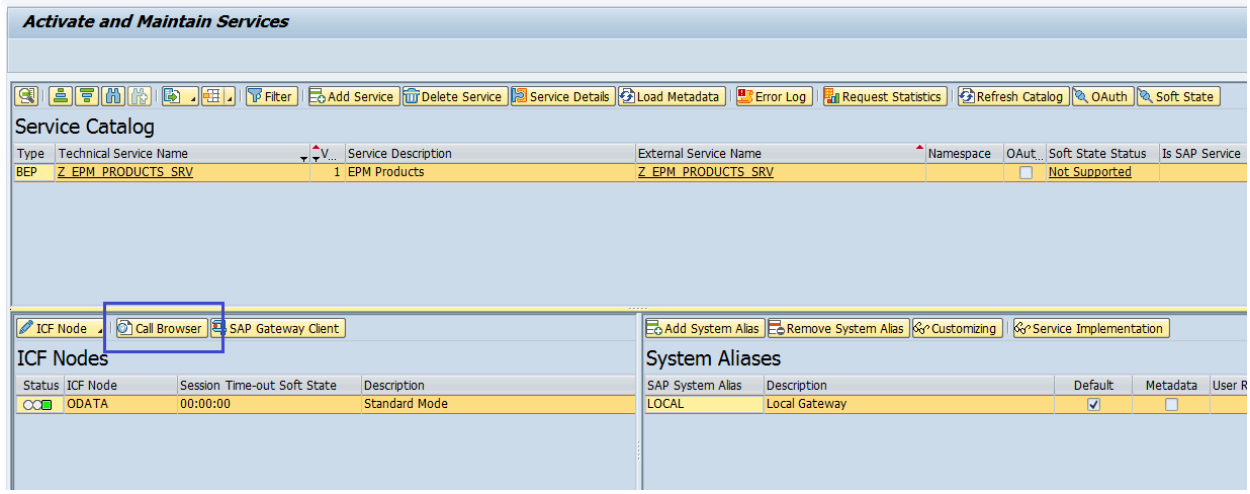
End to End Guide to building OData Service and Consuming it via SAPUI5 application

To test your service click the maintain button. In my case I have the SAP Netweaver Gateway installed in my current SAP Client hence the test will execute in my own server.

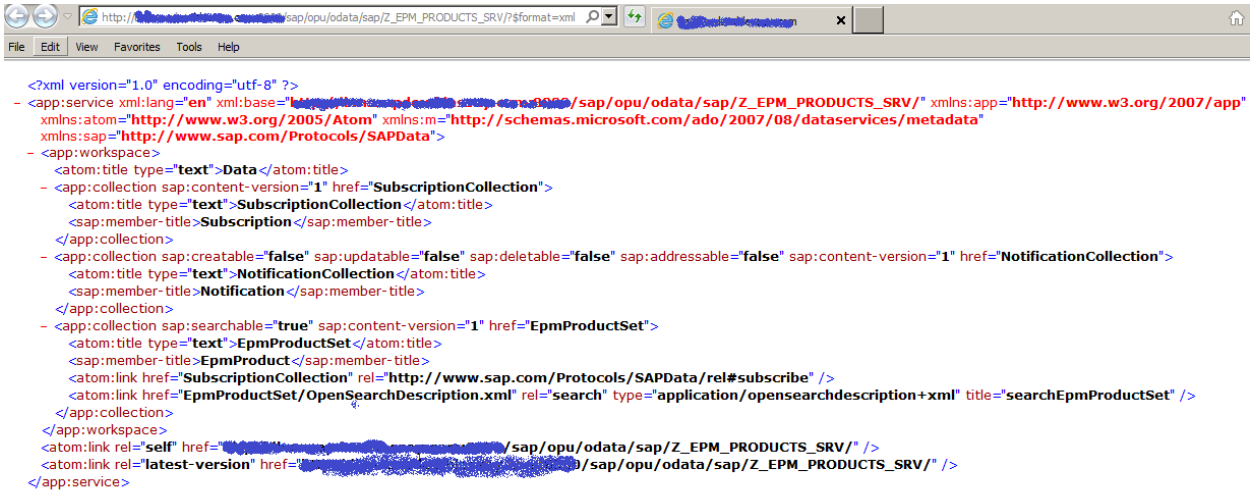


End to End Guide to building OData Service and Consuming it via SAPUI5 application

Clicking the Maintain button will get you to the Activate and Maintain Services screen



From here you can click the Call in Browser button to display the service document in the browser.



Service XML document or Gateway Service Z_EPM_PRODUCTS_SRV

To view the metadata of this service just append '/$metadata' to the service document URL.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

So now our service is ready.

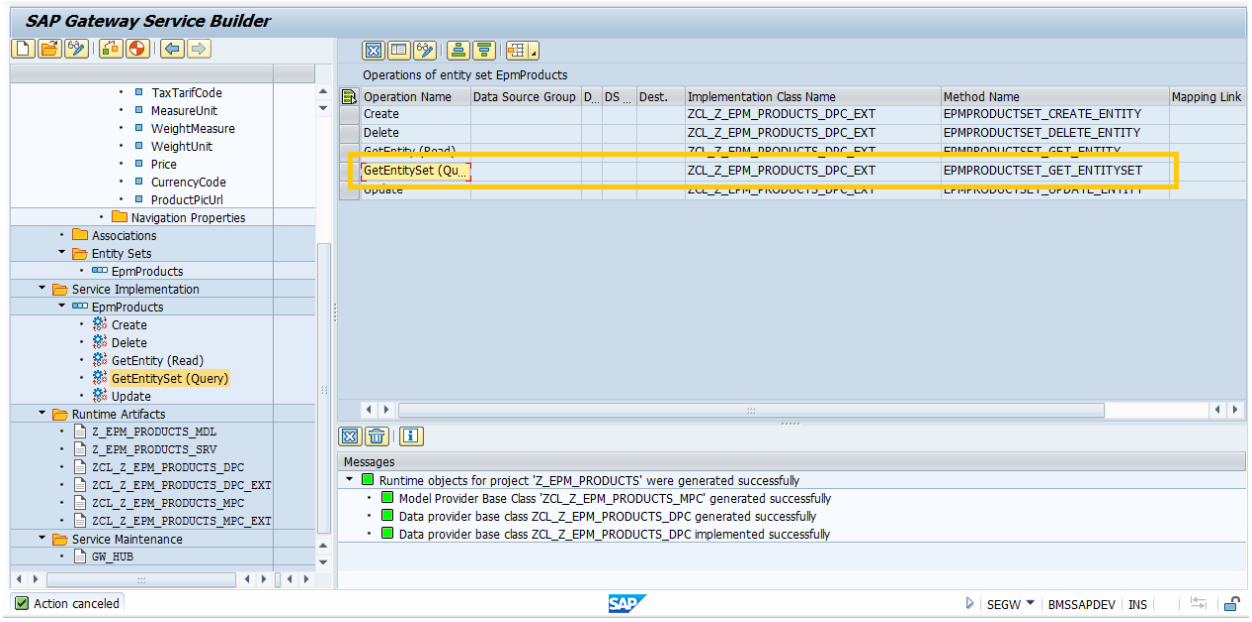
However it won't return any data until we add code, to the data provider class to fetch the EPM product data. We will do that in the next part of this tutorial.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

Implementing the Data Provider Class ZCL_Z_EPM_PRODUCTS_DPC_EXT with Paging and Sorting Logic

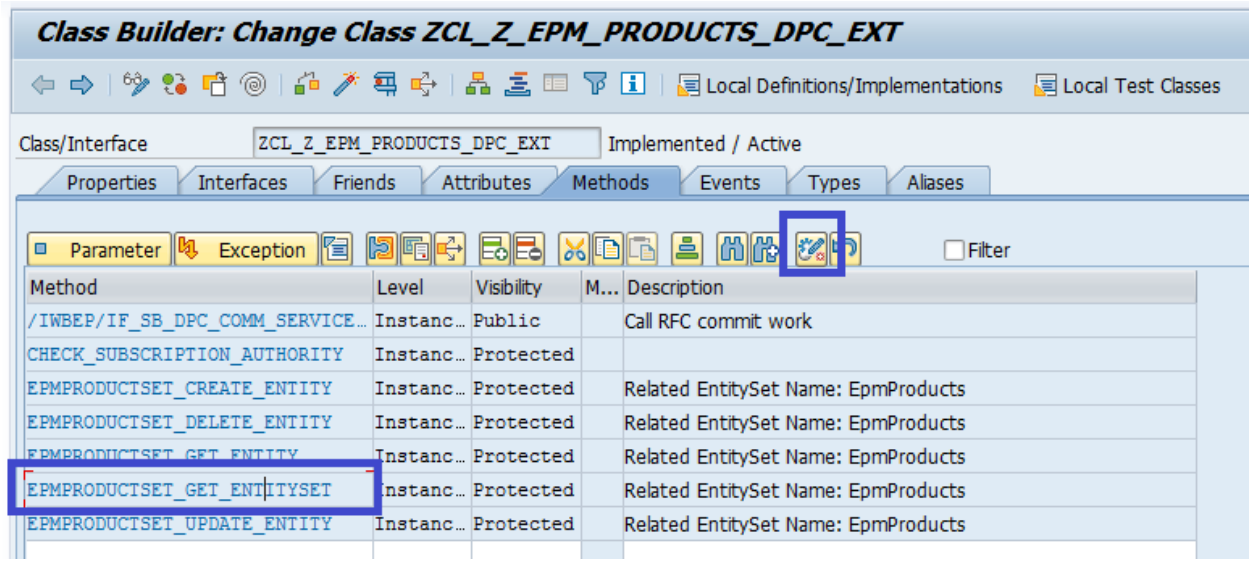
We now need to add code to fetch the product values.

So navigate back the Service builder and expand the service implementation node and locate and click the GetEntitySet (Query) operation. This will show you the implementation class and method for this operation.



We will need to add code to this class, into the method EPMPRODUCTSET_GET_ENTITYSET by redefinition. So in the class builder (Transaction SE24) open the class ZCL_Z_EPM_PRODUCTS_DPC_EXT in change mode, select the method EPMPRODUCTSET_GET_ENTITYSET and click the redefine method.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



Enter the following code into the EPMPRODUCTSET_GET_ENTITYSET Method

```
   DATA lt_products TYPE TABLE OF bapi_epm_product_header.
   DATA lt_return TYPE TABLE OF bapiret2.
   DATA lv_min TYPE i. DATA lv_max TYPE i.
   DATA lt_techorder TYPE /iwbep/t_mgw_tech_order.
   DATA lt_sortorder TYPE abap_sortorder_tab.
   FIELD-SYMBOLS <lf_order> TYPE /iwbep/s_mgw_tech_order.
   FIELD-SYMBOLS <lf_sortorder> TYPE abap_sortorder.
   FIELD-SYMBOLS <lf_products> TYPE bapi_epm_product_header.
   FIELD-SYMBOLS <lf_entityset> TYPE zcl_z_epm_products_mpc=>ts_epmproduct.

* Get technical request information
   IF io_tech_request_context IS BOUND.
      lt_techorder = io_tech_request_context->get_orderby( ).
   ENDIF.

* Get List of Products (to avoid a read with every call a cache could be
implemented)
   CALL FUNCTION 'BAPI_EPM_PRODUCT_GET_LIST'
     TABLES
       headerdata = lt_products
       return     = lt_return.

* Sorting
   LOOP AT lt_techorder ASSIGNING <lf_order>.
     APPEND INITIAL LINE TO lt_sortorder ASSIGNING <lf_sortorder>.
     <lf_sortorder>-name = <lf_order>-property.
```

```
      IF <lf_order>-order = `desc`.
        <lf_sortorder>-descending = abap_true.
      ENDIF.
       IF <lf_order>-property = `PRODUCT_ID`
       OR <lf_order>-property = `DESCRIPTION`
       OR <lf_order>-property = `NAME`
       OR <lf_order>-property = `CURRENCY_CODE`
       OR <lf_order>-property = `SUPPLIER_NAME`.
         <lf_sortorder>-astext = abap_true.
       ENDIF.

    ENDLOOP.

    SORT lt_products BY (lt_sortorder).

* Paging
    IF is_paging-skip IS NOT INITIAL.
      lv_min = is_paging-skip + 1.
    ELSE.
      lv_min = 1.
    ENDIF.
    IF is_paging-top IS NOT INITIAL.
      lv_max = is_paging-skip + is_paging-top.
    ELSE.
      lv_max = lines( lt_products ).
    ENDIF.
    LOOP AT lt_products FROM lv_min TO lv_max ASSIGNING <lf_products>.
      APPEND INITIAL LINE TO et_entityset ASSIGNING <lf_entityset>.
      MOVE-CORRESPONDING <lf_products> TO <lf_entityset>.
    ENDLOOP.
```

Save and Activate your code.

The code is simple to understand. All we are doing here is calling the BAPI 'BAPI_EPM_PRODUCT_GET_LIST' which we had executed in the Pre-Check step to fetch the values for PRODUCTS from the SAP system. Then we added code to be able to SORT order to our put if it is requested by the service caller in the io_tech_request_context and if Paging is requested then we add paging value as read from the is_paging Parameter.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

Thru this code you have now made your service able to fetch and return product values to your service caller.

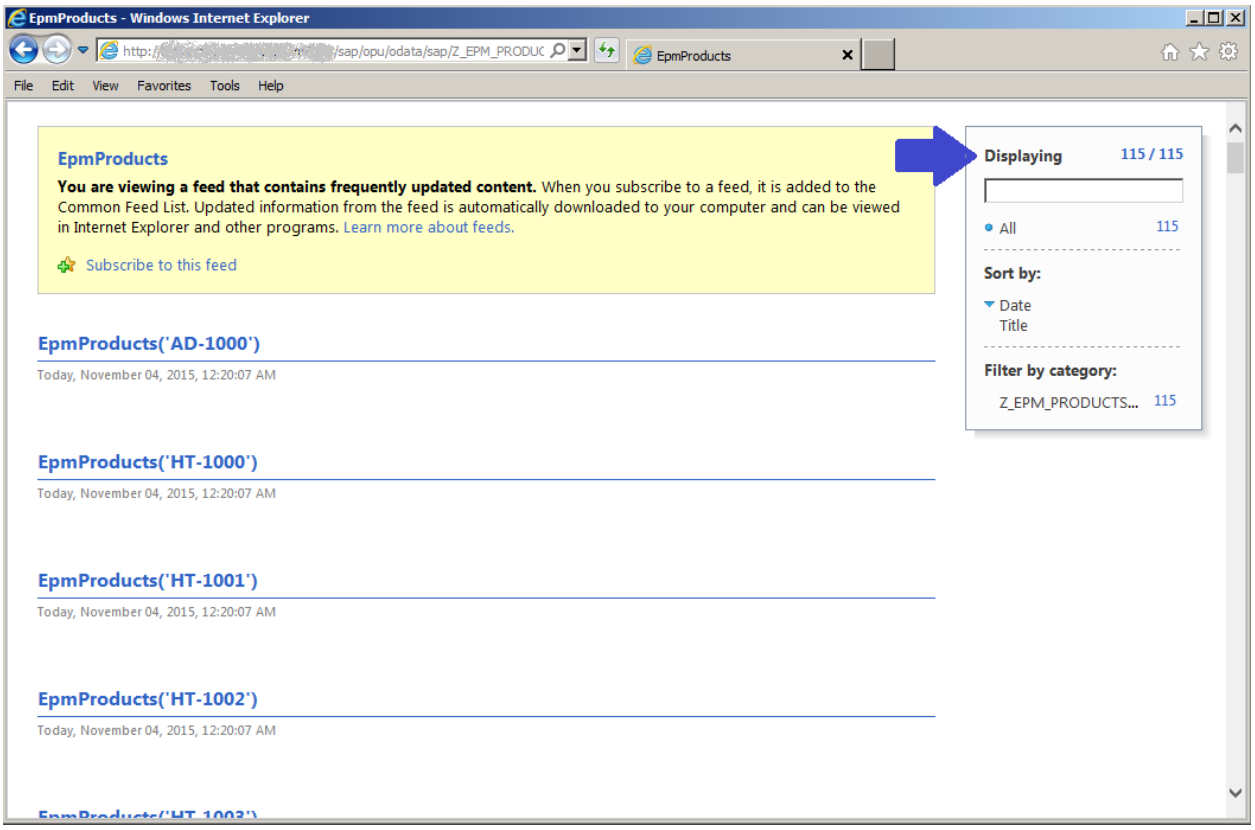
To ensure that the service will return you the values properly test the service by executing the service again in the browser


```
http://bmssapdev.blessey.com:8000/sap/opu/odata/sap/Z_EPM_PRODUCTS_SRV/EpmProducts/?$format=xml - Windows Internet Explorer
http://[redacted]/sap/opu/odata/sap/Z_EPM_PRODUC
bmssapdev.blessey.com
File Edit View Favorites Tools Help

- <feed xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xml:base="[redacted]/sap/opu/odata/sap/Z_EPM_PRODUCTS_SRV/">
    <id>http://[redacted]/sap/opu/odata/sap/Z_EPM_PRODUCTS_SRV/EpmProducts/</id>
    <title type="text">EpmProducts</title>
    <updated>2015-11-04T06:14:37Z</updated>
  - <author>
      <name />
    </author>
    <link href="EpmProducts/" rel="self" title="EpmProducts" />
  - <entry>
      <id>[redacted]/sap/opu/odata/sap/Z_EPM_PRODUCTS_SRV/EpmProducts('AD-1000')</id>
      <title type="text">EpmProducts('AD-1000')</title>
      <updated>2015-11-04T06:14:37Z</updated>
      <category term="Z_EPM_PRODUCTS_SRV.EpmProduct"
        scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
      <link href="EpmProducts('AD-1000')" rel="edit" title="EpmProduct" />
    - <content type="application/xml">
      - <m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
          xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
          <d:ProductId>AD-1000</d:ProductId>
          <d:TypeCode>AD</d:TypeCode>
          <d:Category>Computer system accessories</d:Category>
          <d:Name>Flyer</d:Name>
          <d:Description>Flyer for our product palette</d:Description>
          <d:SupplierId>100000015</d:SupplierId>
          <d:SupplierName>Robert Brown Entertainment</d:SupplierName>
          <d:TaxTarifCode>1</d:TaxTarifCode>
          <d:MeasureUnit>EA</d:MeasureUnit>
          <d:WeightMeasure>0.010</d:WeightMeasure>
          <d:WeightUnit>KG</d:WeightUnit>
          <d:Price>0.0000</d:Price>
          <d:CurrencyCode>CAD</d:CurrencyCode>
          <d:ProductPicUrl>/SAP/PUBLIC/BC/NWDEMO_MODEL/IMAGES/AD-1000.jpg</d:ProductPicUrl>
        </m:properties>
```


If you check the feed in Internet Explorer you will notice that there are exactly 115 records returned.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

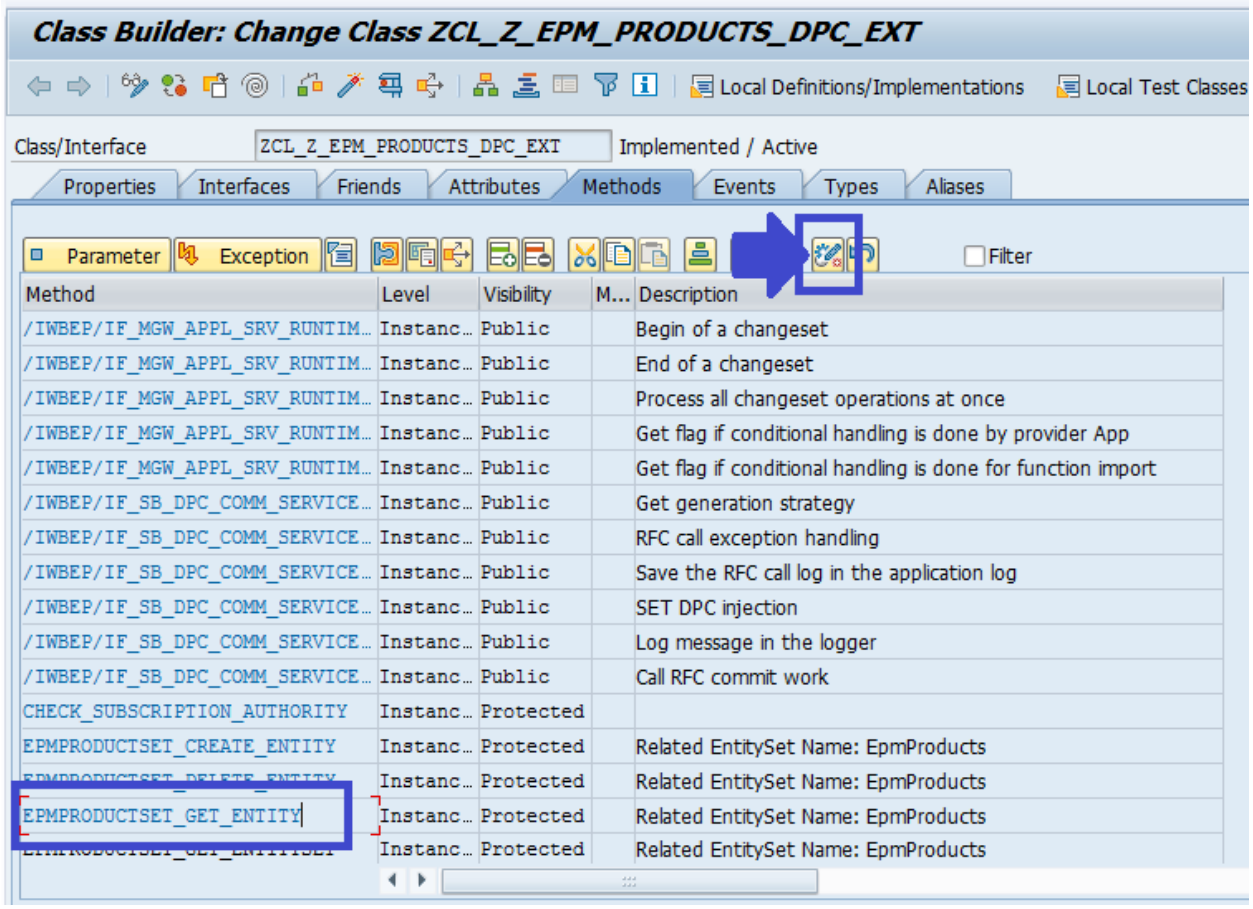


Tip: In internet explorer, to switch from the feed view to the raw xml, go to Internet Options -> Content -> Feeds and Web Slices Settings and uncheck the "Turn on feed reading view". Then revisit the URL, you should see the raw XML from the service.

The above implementation will enable the service to return the entire set of records. However we would also want out service to return specific product data when the product id is passed to the service. This can be done by redefining the EPMPRODUCTSET_GET_ENTITY method of the ZCL_Z_EPM_PRODUCTS_DPC_EXT class.

So again in the class builder (Transaction SE24) open the class ZCL_Z_EPM_PRODUCTS_DPC_EXT in change mode, select the method EPMPRODUCTSET_GET_ENTITY and click the redefine method.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



Enter the following code into the EPMPRODUCTSET_GET_ENTITY Method

```
DATA lv_product_id TYPE bapi_epm_product_id.
DATA ls_product TYPE bapi_epm_product_header.
DATA lt_return TYPE TABLE OF bapiret2.
FIELD-SYMBOLS <lf_key> TYPE /iwbep/s_mgw_name_value_pair.

READ TABLE it_key_tab INDEX 1 ASSIGNING <lf_key>.

IF sy-subrc = 0 AND <lf_key> IS ASSIGNED.
  lv_product_id = <lf_key>-value.
  CALL FUNCTION 'BAPI_EPM_PRODUCT_GET_DETAIL'
    EXPORTING
      product_id = lv_product_id
    IMPORTING
      headerdata = ls_product
    TABLES
      return     = lt_return.
```

End to End Guide to building OData Service and Consuming it via SAPUI5 application

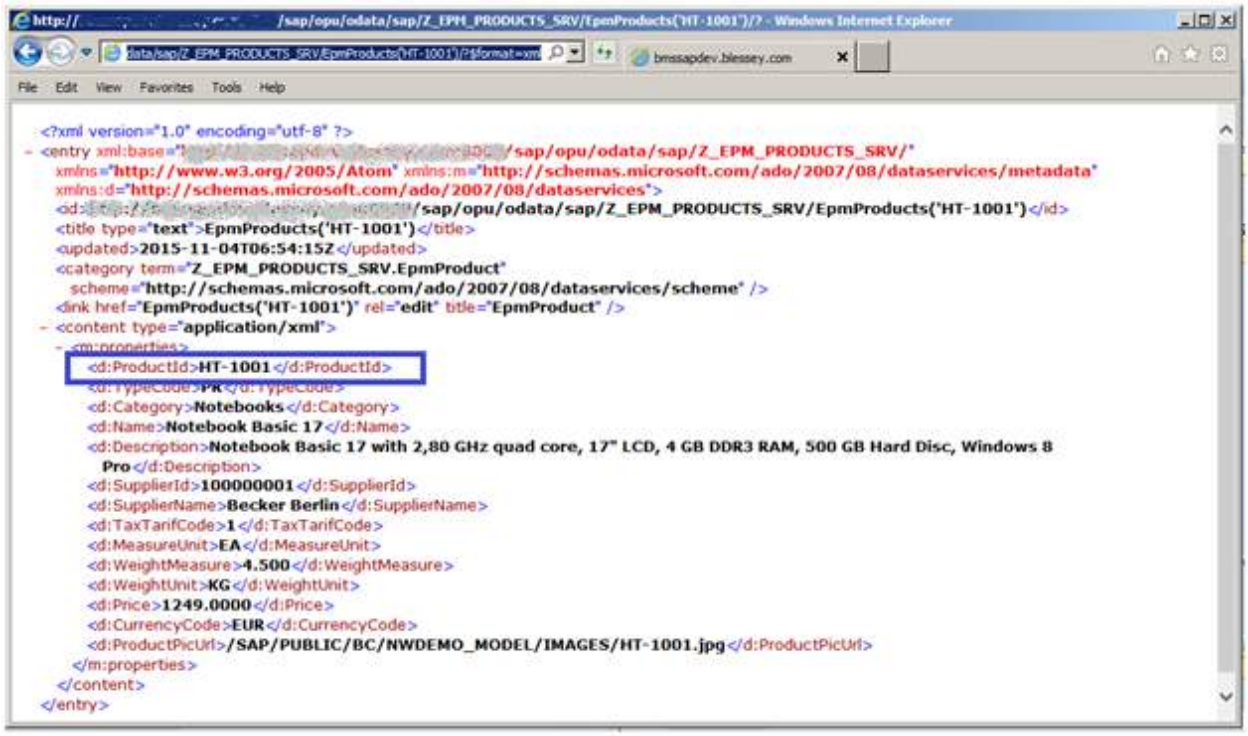
```
    MOVE-CORRESPONDING ls_product TO er_entity.
  ENDIF.
```

Save and activate the code.

In the above code we are simply passing the Product id from the import parameter it_key_tab of the method to 'BAPI_EPM_PRODUCT_GET_DETAIL' and returning back the product details into the export parameter ER_ENTITY.

Thru this code you have now made your service able to fetch and return a product values for a specific product based on the product id to your service caller.

To ensure that the service will return you the values properly test the service by executing the service again in the browser.



This means now we have a nicely working backend Odata service created and is responding as required.

This completes PART 1 of this tutorial.

In the following PART below we will now move focus to the front-end development part using SAPUI5 and consuming the data provided by this service to present it in a functional way to the end user.

END OF PART 1

PART 2

Developing an application using SAPUI5 to consume an OData Service.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

Introductions

SAPUI5 – UI Development Toolkit for HTML5

Why to use SAPUI5 :

The basic premise of OData is that any consumer application which can talk HTTP protocol should be able to consume the data. In other words any programming language which can understand how to make HTTP calls can be used to interact with OData.

However to that end SAP has provided its own set of library functions which provide a generic OData proxy object included in the SAPUI5 runtime libraries.

We will therefore use this SAPUI5 library to develop our application.

Pre Install steps

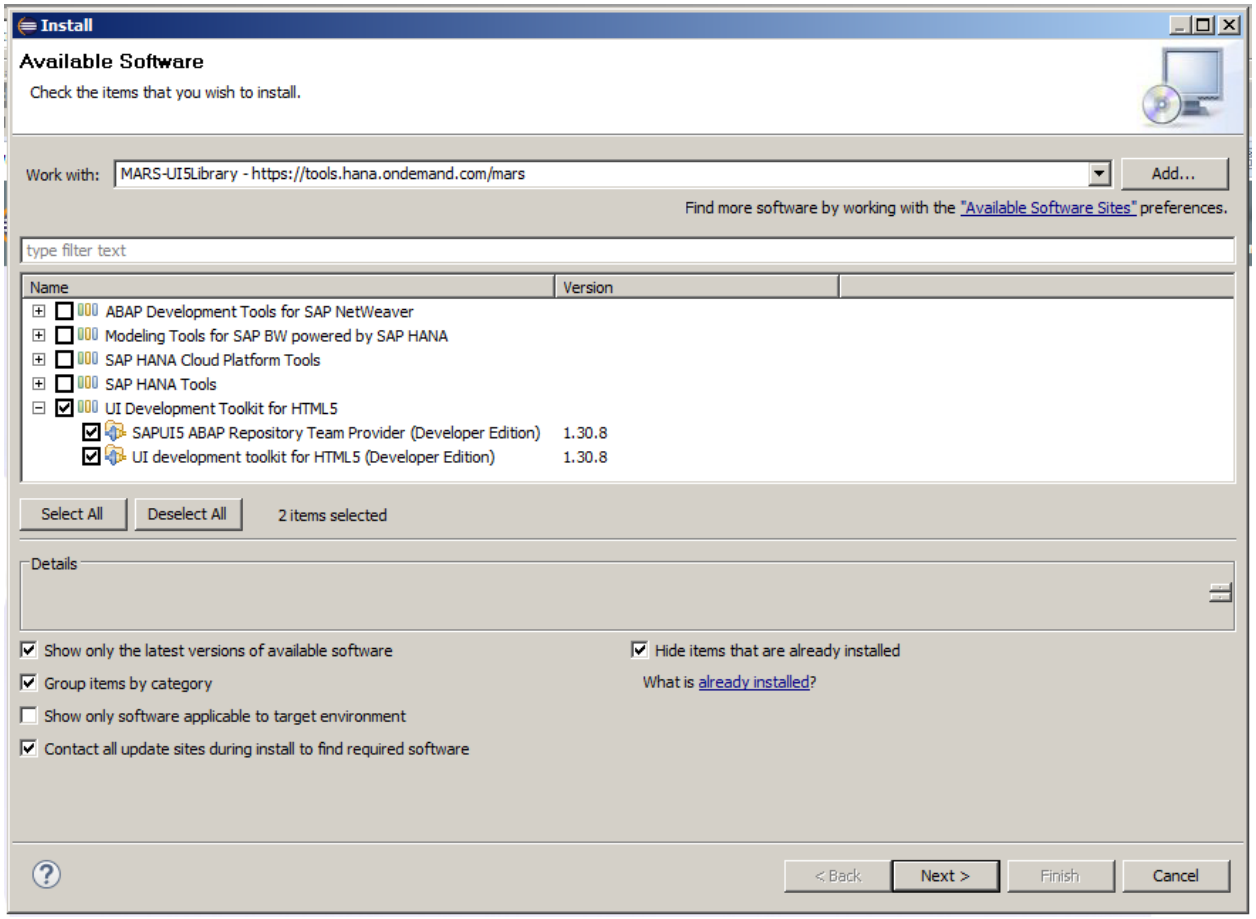
I am using the MARS version of Eclipse for this tutorial

You will find all the installation details about Eclipse-Mars and corresponding SAPUI5 Files for this version at this link

https://tools.hana.ondemand.com/mars/

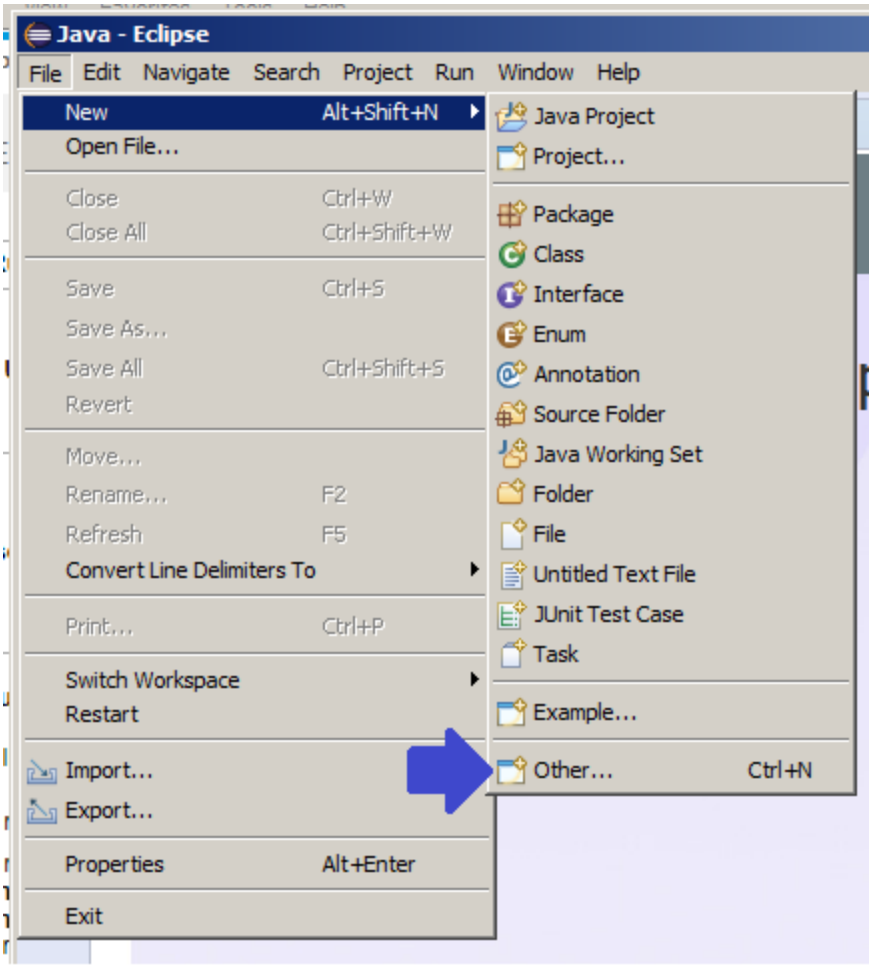
Refer to this tutorial blog to learn the steps to install MARS version of Eclipse and to install SAPUI5 on your local machine.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



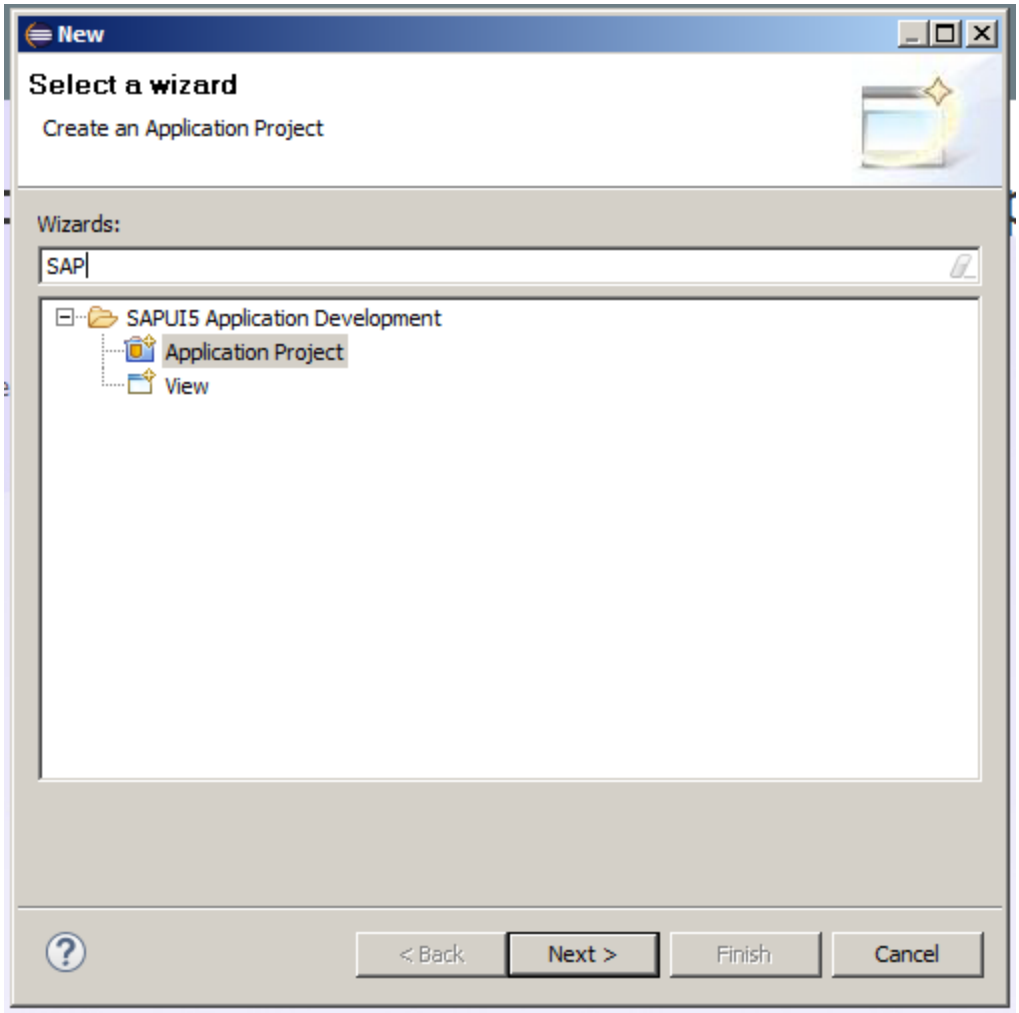
1. Start Eclipse
2. Navigate to File->New->Other

End to End Guide to building OData Service and Consuming it via SAPUI5 application



3. In the screen that appears, type SAP and under SAPUI5 Application Development folder select Application Project and hit the Next Button.

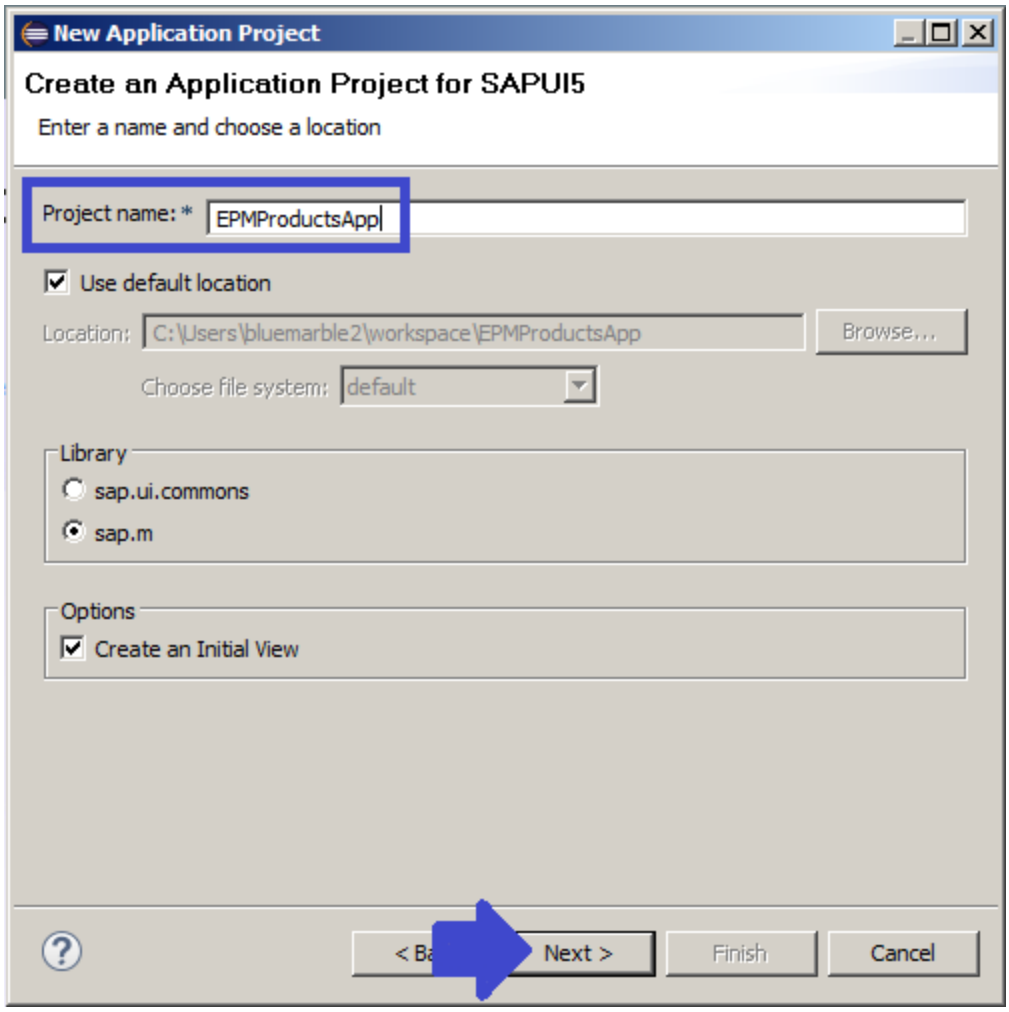
End to End Guide to building OData Service and Consuming it via SAPUI5 application



4. On the next Screen you need to provide a suitable project name for you APP

5. Here we are going to display the Products to the end user using the OData service we created in part 1, So enter a suitable name – EPMProductsApp

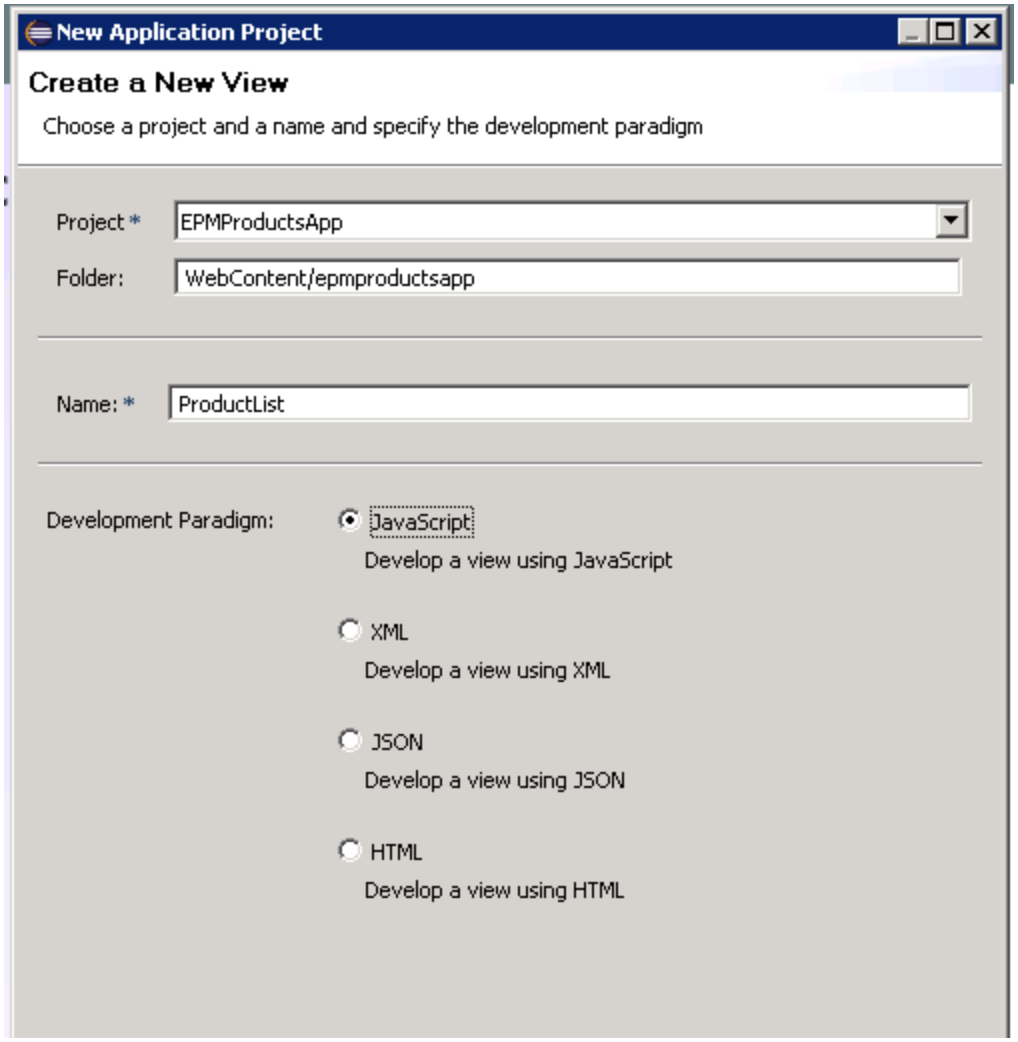
End to End Guide to building OData Service and Consuming it via SAPUI5 application



6. Keeping all other settings the same (or you can change the file location – your wish) click the next button. Notice in the above screenshot in Options, Create an Initial View is checked, hence in the next step we will need to provide details for the application's Initial view.

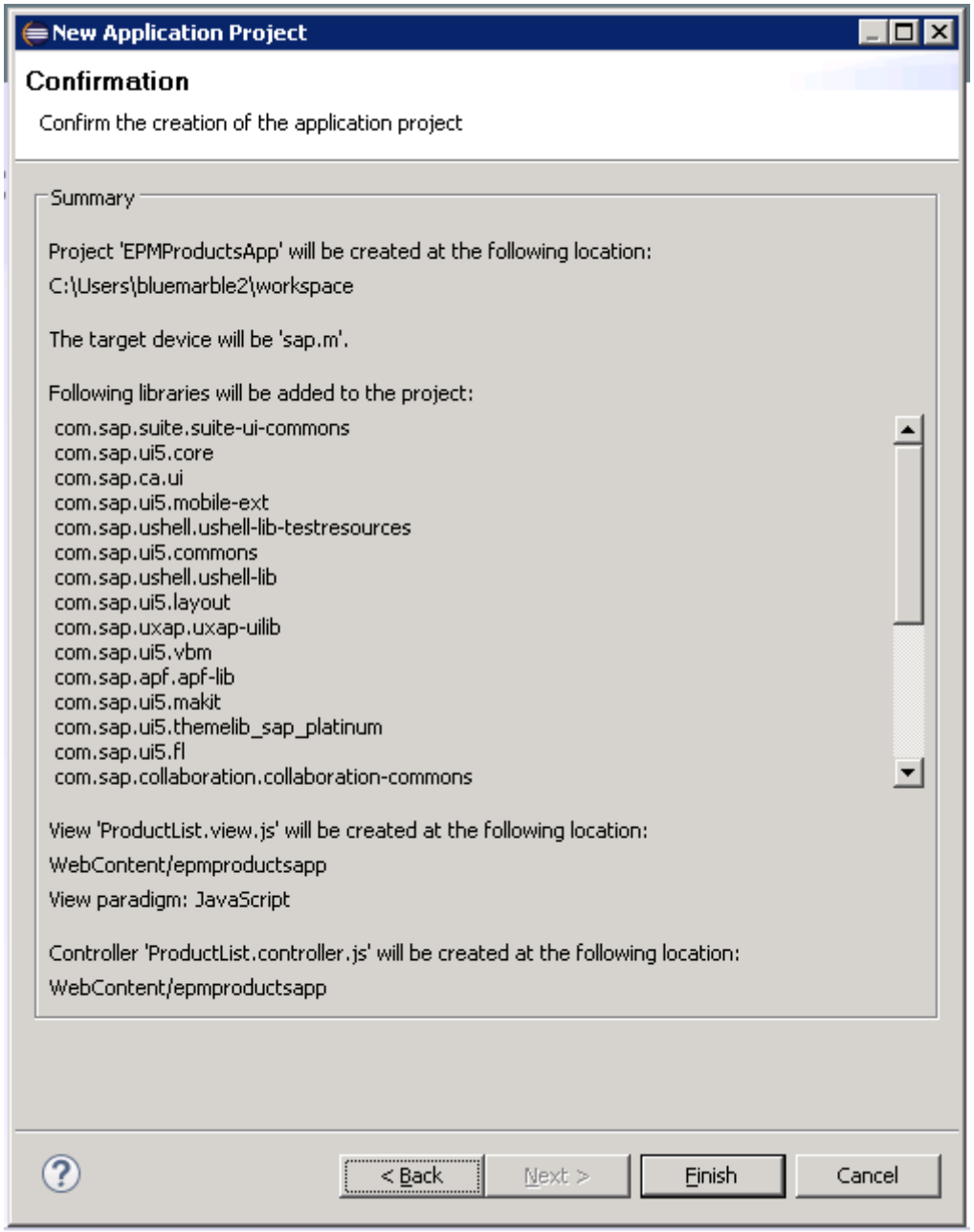
7. I have named it ProductList and kept Development Paradigm as Javascript and hit the next button.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



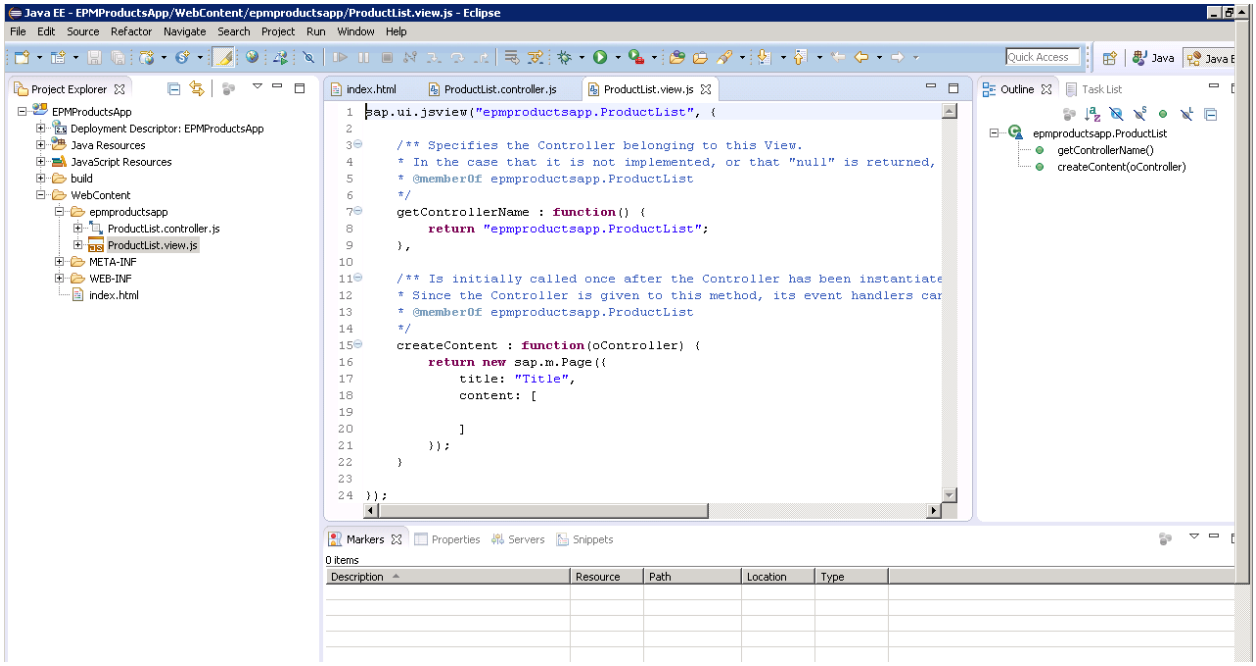
8. Finally you get a list of objects that will be automatically created for you. Click Finish.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



9. As a result an SAPUI5 Application component with the following parts is created.
10. Click Finish
11. You can then check the objects created for you.

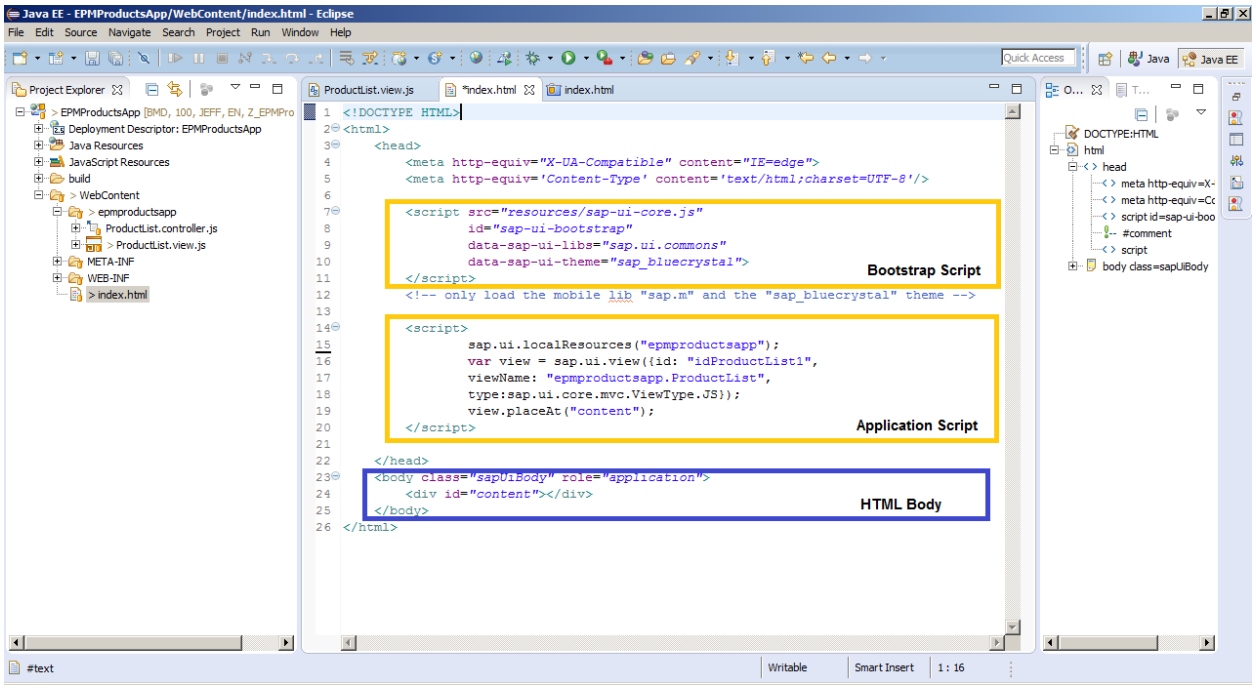
End to End Guide to building OData Service and Consuming it via SAPUI5 application



On close observation you will notice that the following parts have been created for you

1. ProductList.view.js: A view file of type JavaScript
2. **ProductList.controller.js**: A controller file of type JS
3. **index.html - The main file that contains the libaray references, the theme references, etc.**

End to End Guide to building OData Service and Consuming it via SAPUI5 application



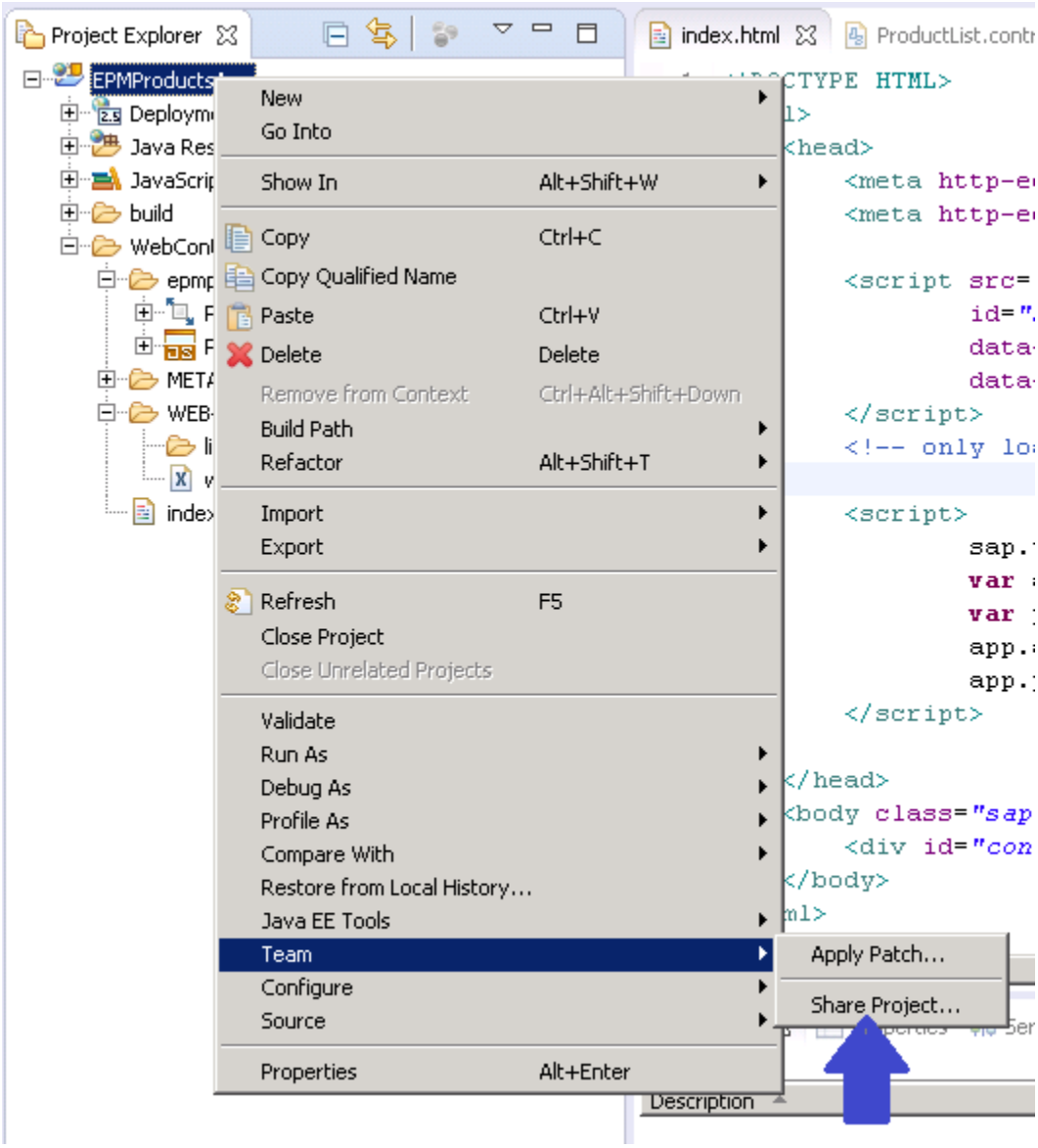
12. We now need to tie these parts together to create a smoothly running Product APP

End to End Guide to building OData Service and Consuming it via SAPUI5 application

How to deploy your SAPUI5 Application on the ABAP system

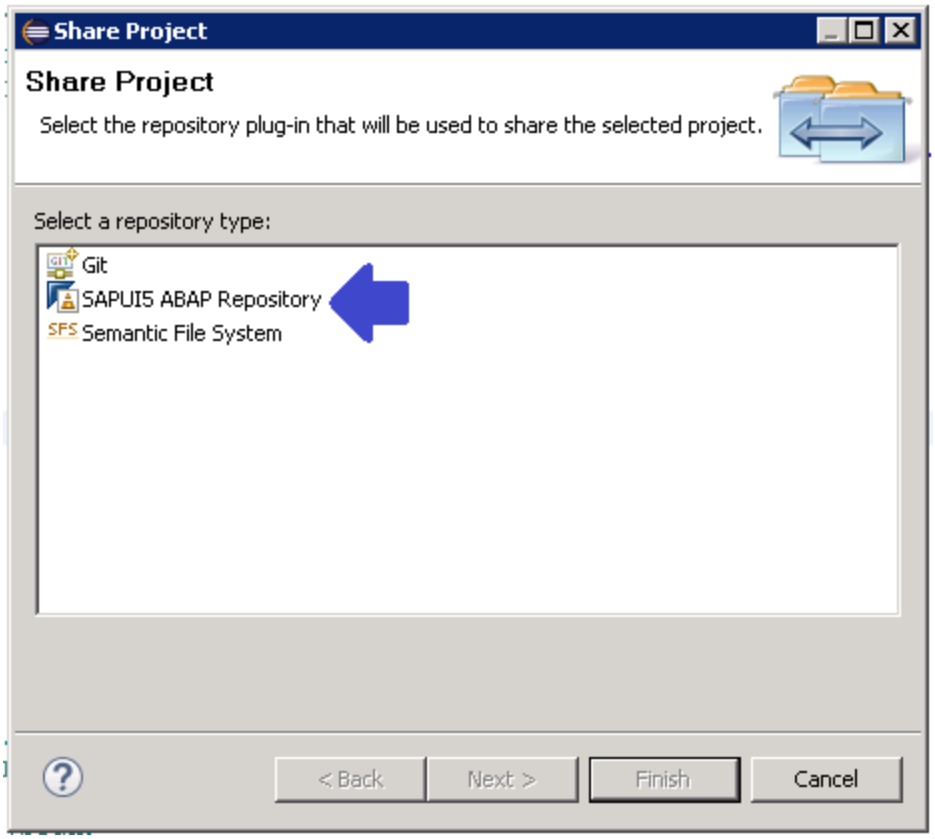
Follow the steps to connect your project to the Backend SAP system

1. Right click on your Project name in the Project Explorer and Select the option Team -> Share Project.

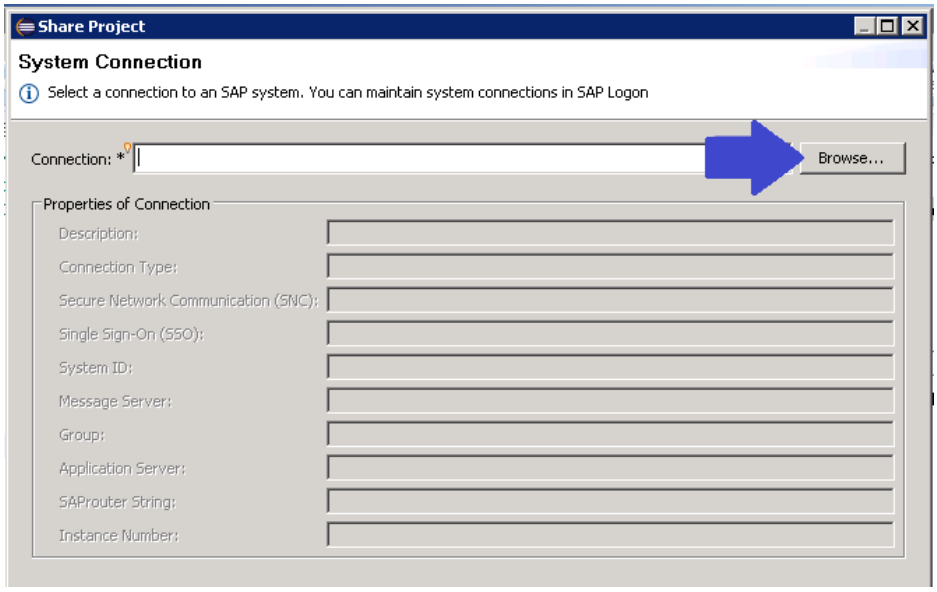


2. On the next screen select a repository plugin which in our case is the SAPUI5 ABAP Repository and hit next button

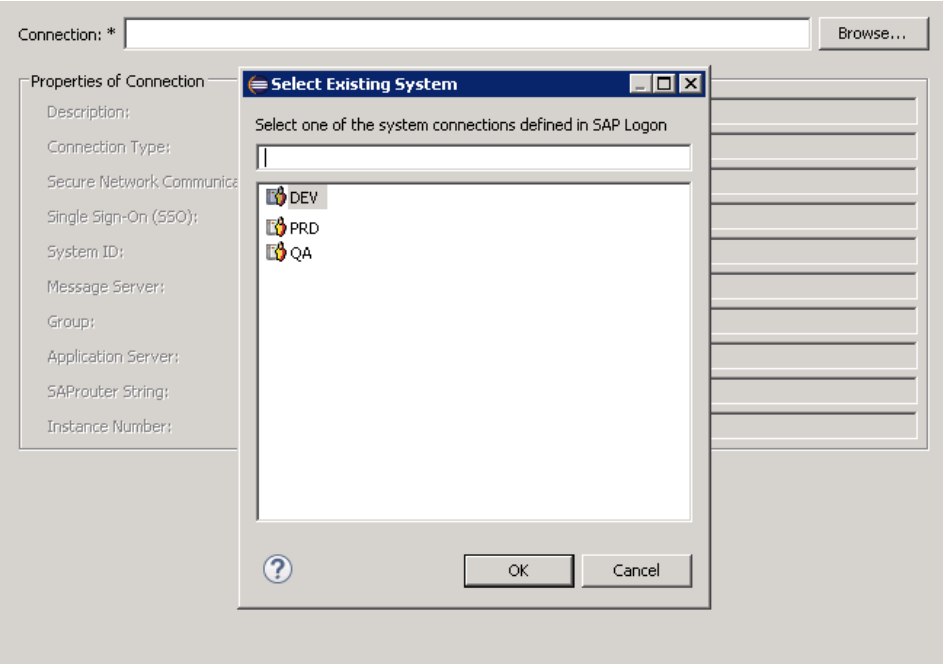
End to End Guide to building OData Service and Consuming it via SAPUI5 application



3. In the screen that follows you need to select your SAP server instance that you want to connect to.



End to End Guide to building OData Service and Consuming it via SAPUI5 application

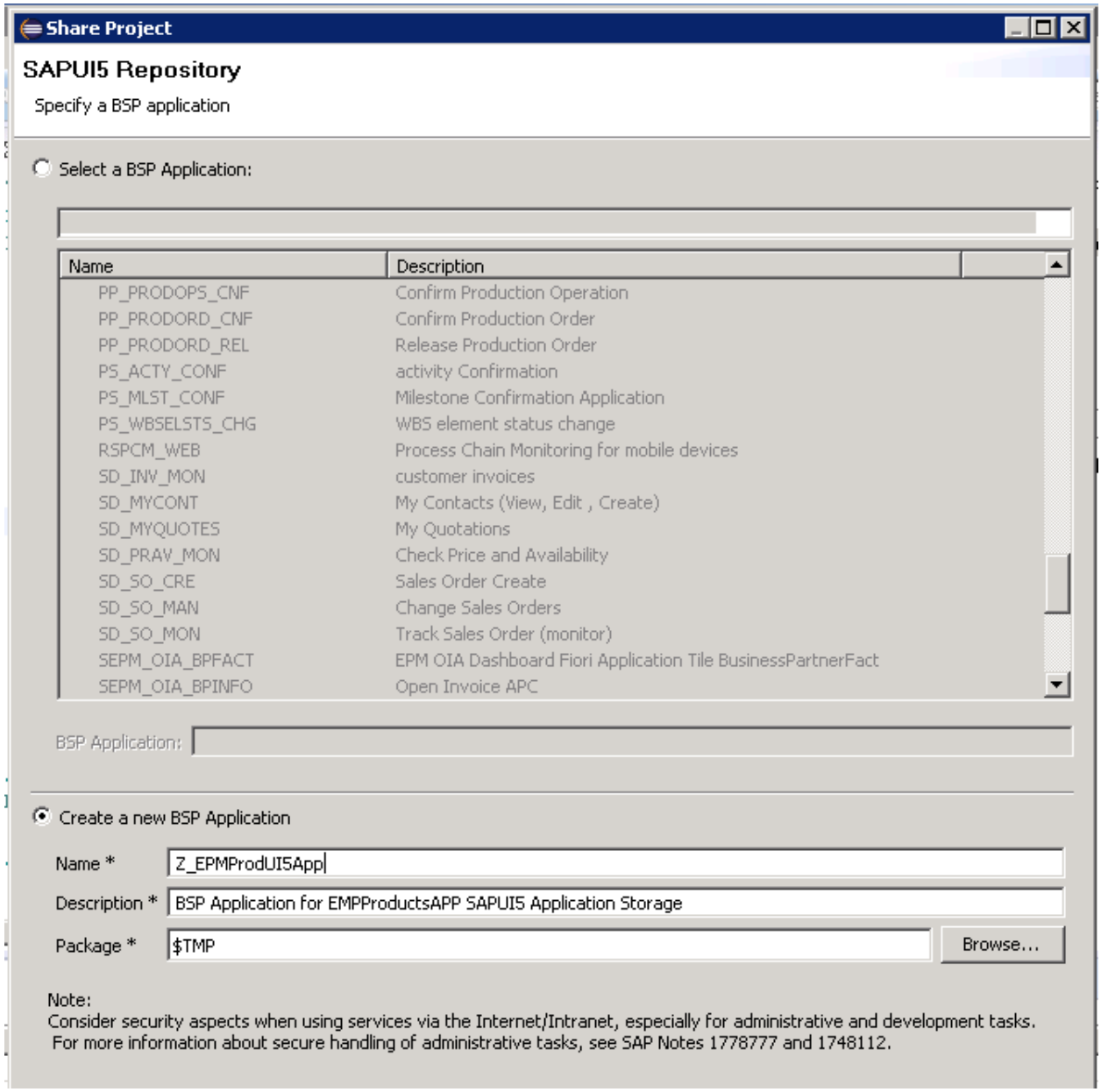


4. And then click ok. The system automatically populates all the data related to this connection. Click the Next button.

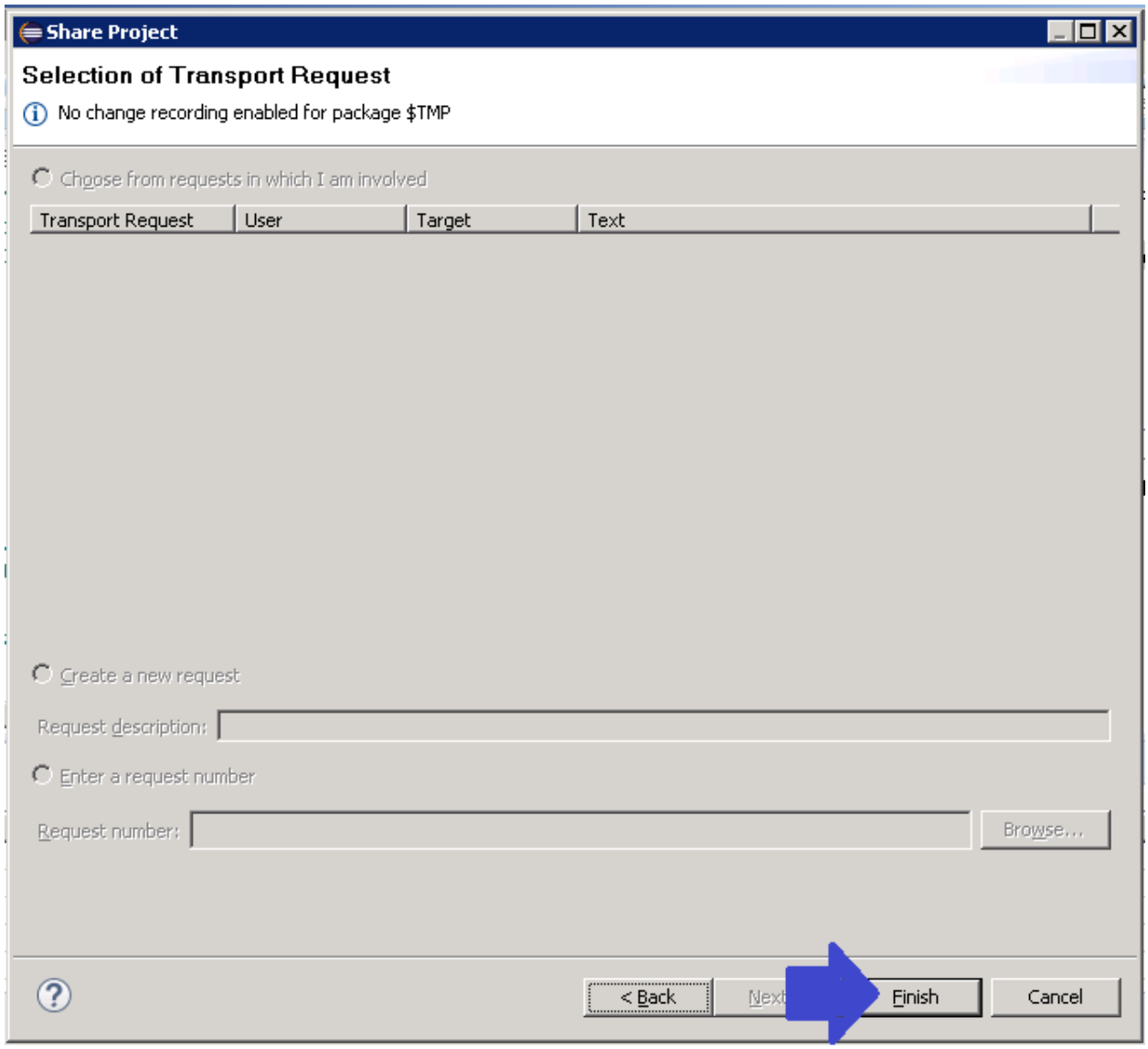
5. On the next screen – Provide the Client, Userid and Password of the SAP Instance you want to connect to and hit the Next button.
6. Eclipse will then try to connect to your SAP system.

7. Once the connection is established, you then have to specify the name, a description and the development package of the BSP Application (in our example Z_EPMPRODUI5APP) which will be created as the corresponding repository object (if a transportable package is selected, in the next step a transport request has to be selected or created).
8. So Select the Create a new BSP Application check box and provide the name Z_EPMProductsApp and a description as EMP Product Catalgue Display App
9. For Package – I have selected a local object $TMP.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

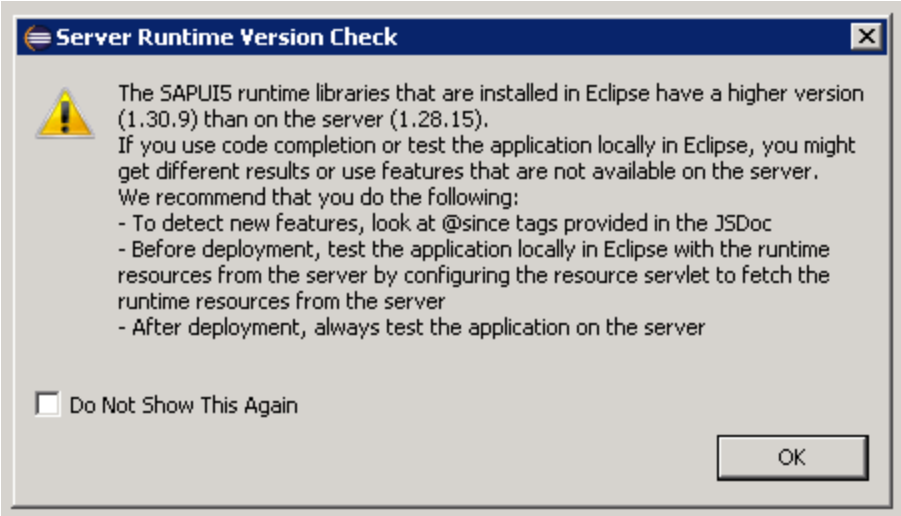


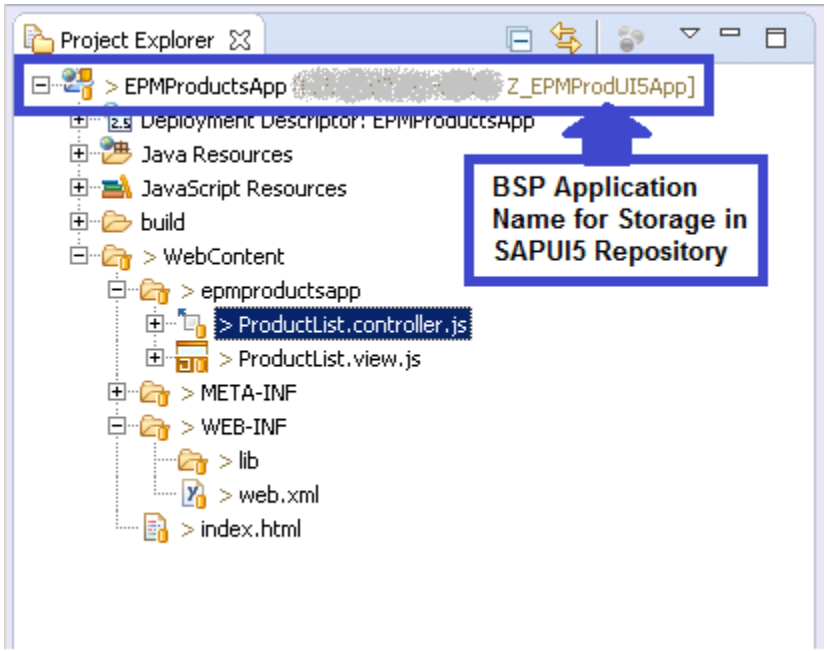
10. On the next screen you may need to provide a transport request but since this is a local object ($TMP) hence no transport request is necessary. Simply hit the finish button on the next screen.



11. You may get this warning message

End to End Guide to building OData Service and Consuming it via SAPUI5 application



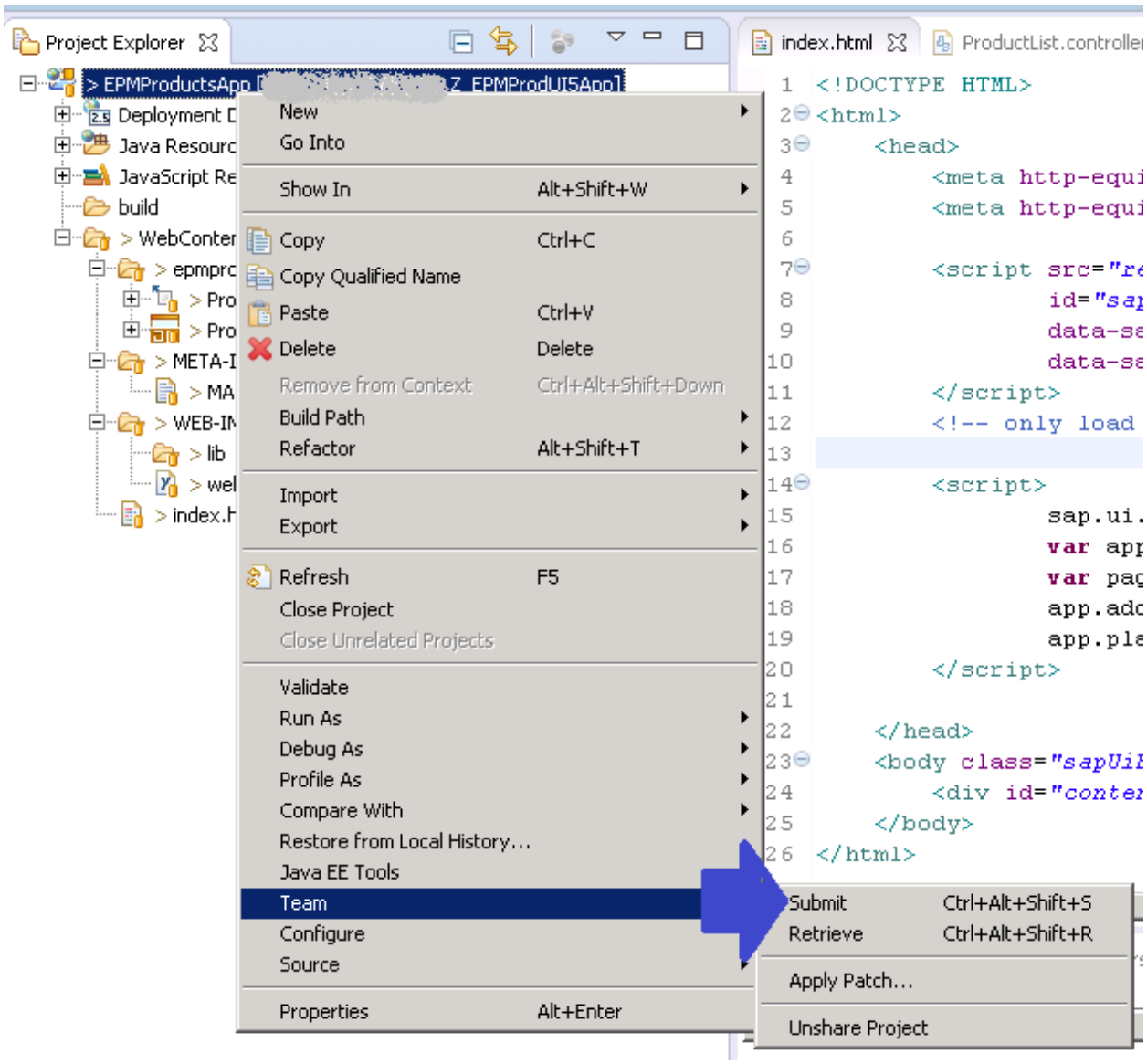


12. Now you can see that our SAPUI5 application in Eclipse is connected to the Backend BSP application in SAP. Which means once our development in Eclipse is complete; we can simply transport the whole application to Quality and Production as a single transport request.

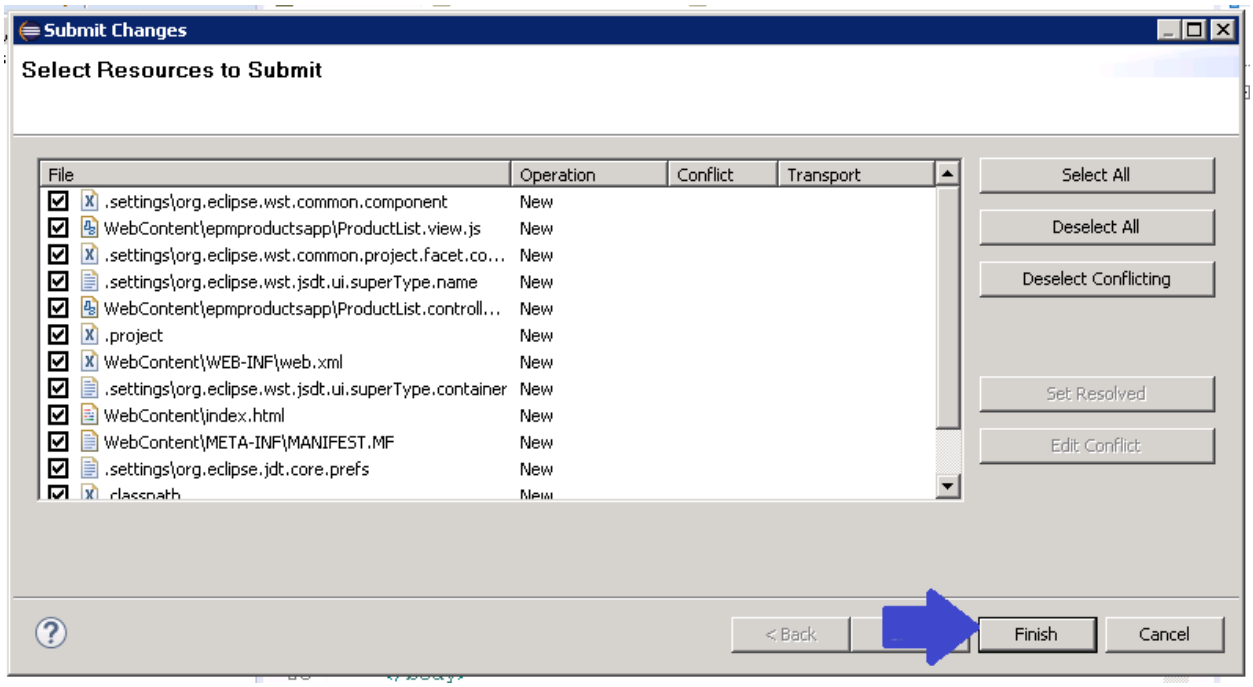
13. However note that we have still not transferred any content from Eclipse to Backend SAP system. To do that follow the steps as outlined below.

14. Right click on the Project name and select Team/Submit

End to End Guide to building OData Service and Consuming it via SAPUI5 application

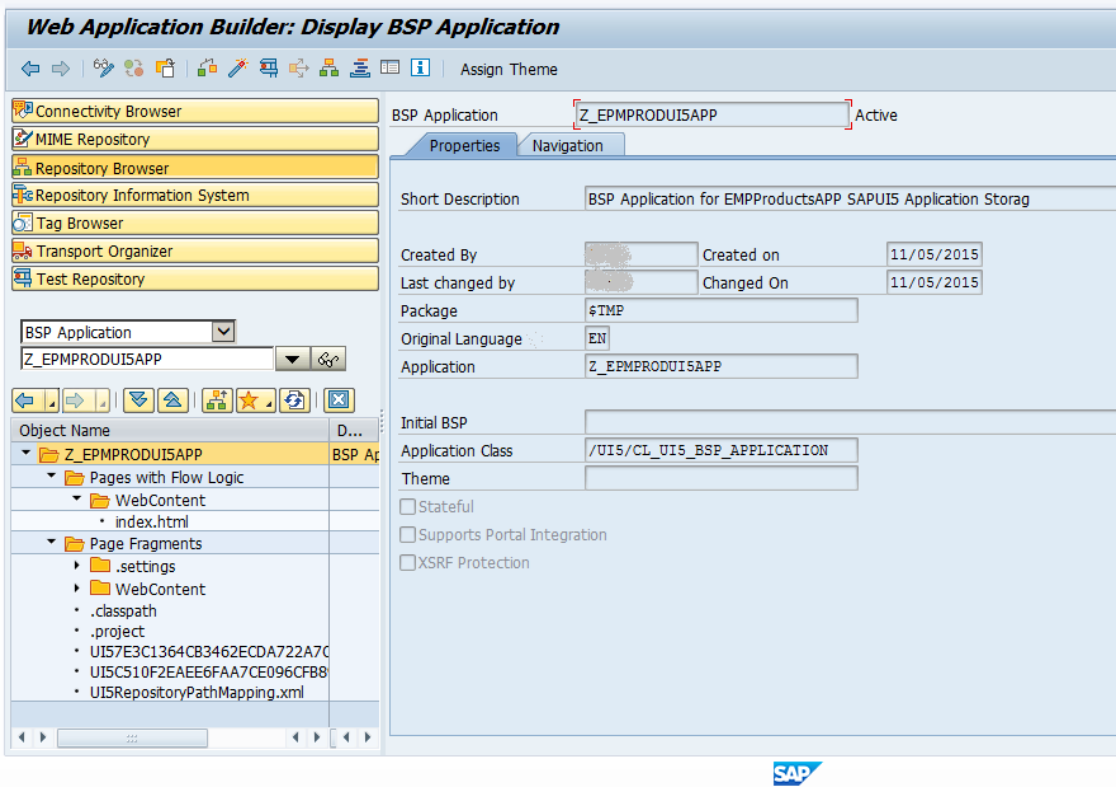


15. The system shows a list of all the resources which you have developed in your application. Select all the ones which are relevant. Since we are deploying this application on to server for the first time hence we have to select all the resources. Click the Finish button.



16. You can now log into our SAP backend and location our SAPUI5 Application as a BSP Application in the backend

End to End Guide to building OData Service and Consuming it via SAPUI5 application



17. Our Application is fully functional now, however we still haven't added any functional code to this application.
18. To test that our deployment works correctly lets add a simple "Hello World" statement to the code.

This code we will add in the View of the Application so that this Text will be displayed when we execute out application.

So in eclipse, locate the ProductList.view.js File and here in the createContent Function add the following code

```
return new sap.ui.commons.TextView({text:"Hello World"});
```

End to End Guide to building OData Service and Consuming it via SAPUI5 application

index.html | ProductList.controller.js | *ProductList.view.js

```
1  sap.ui.jsview("epmproductsapp.ProductList", {
2
3      /** Specifies the Controller belonging to this View.
4      * In the case that it is not implemented, or that "null" is re
5      * @memberOf epmproductsapp.ProductList
6      */
7      getControllerName : function() {
8          return "epmproductsapp.ProductList";
9      },
10
11     /** Is initially called once after the Controller has been ins
12     * Since the Controller is given to this method, its event hand
13     * @memberOf epmproductsapp.ProductList
14     */
15     createContent : function(oController) {
16         return new sap.ui.commons.TextView({text:"Hello World"});
17     }
18
19 });
```

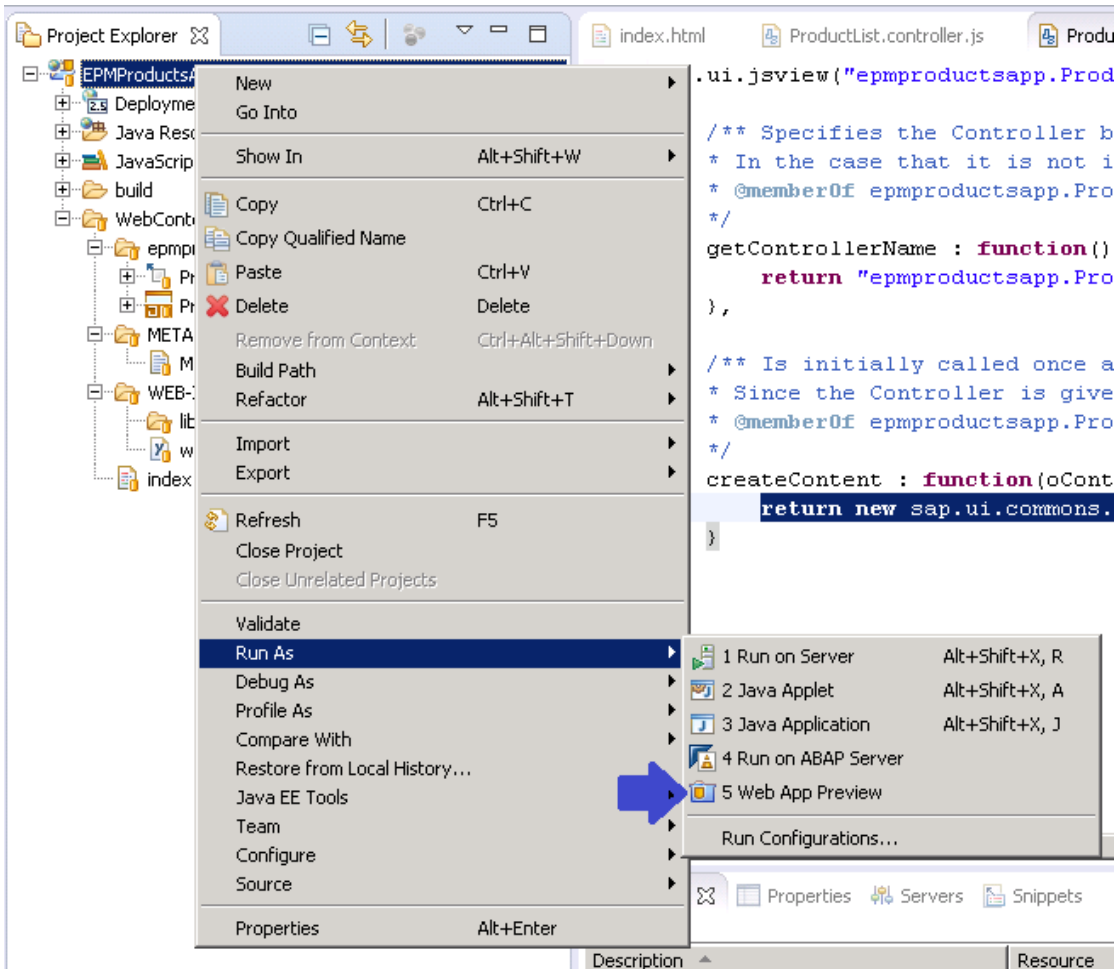
Hit Save in Eclipse and then Submit your change to SAP through Team/Submit.

Now it's time to test our changes.

To do so follow the steps below.

1. In the Project Explorer, Right click on the Application Name and Select – Run As/ Web App Preview.

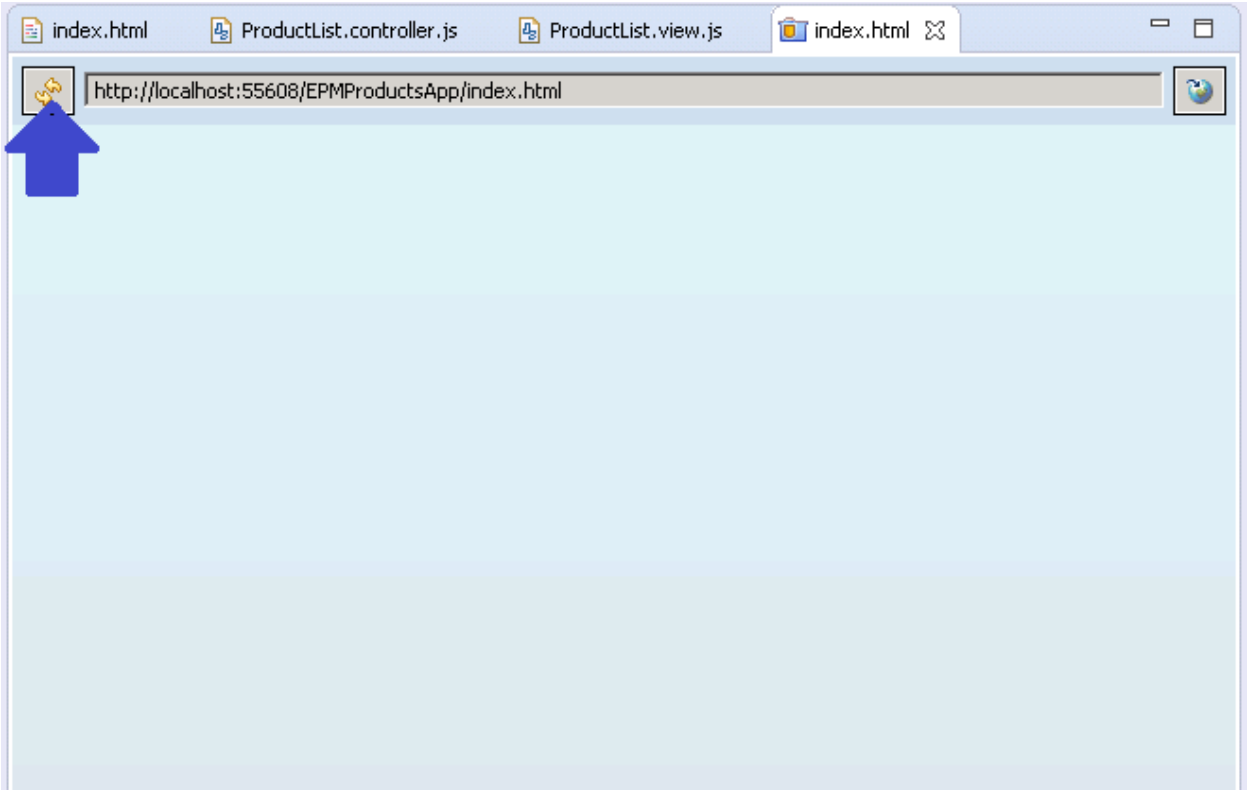
End to End Guide to building OData Service and Consuming it via SAPUI5 application



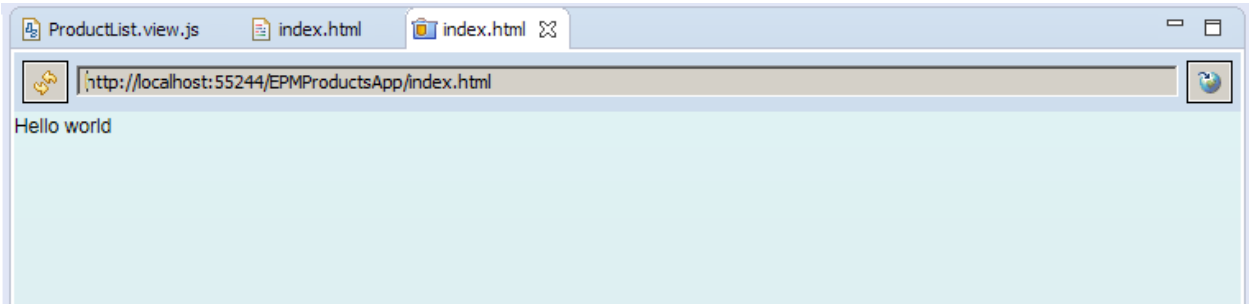
2. This will launch your application in an inbuilt preview-editor in Eclipse

3. Click the Refresh button here to see your changes to the application.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



And here is our output

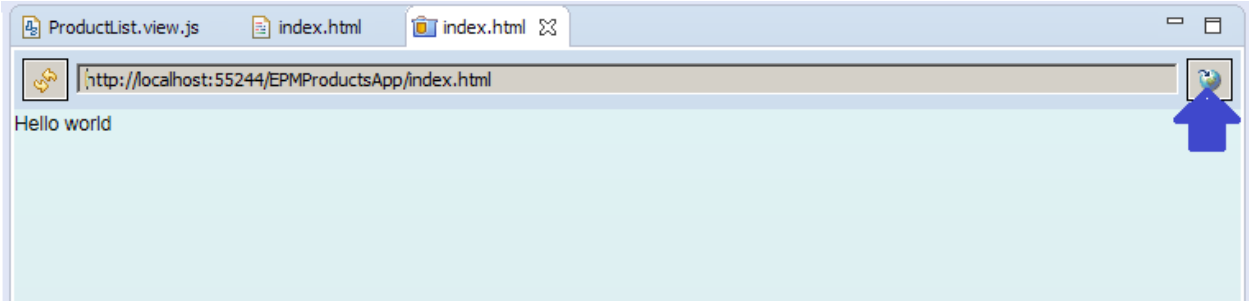


Note: In-case you do not see your output "Hello world" here, then most probably there is a miss-match in the library used in the bootstrap script. I had this issue which I resolved by changing

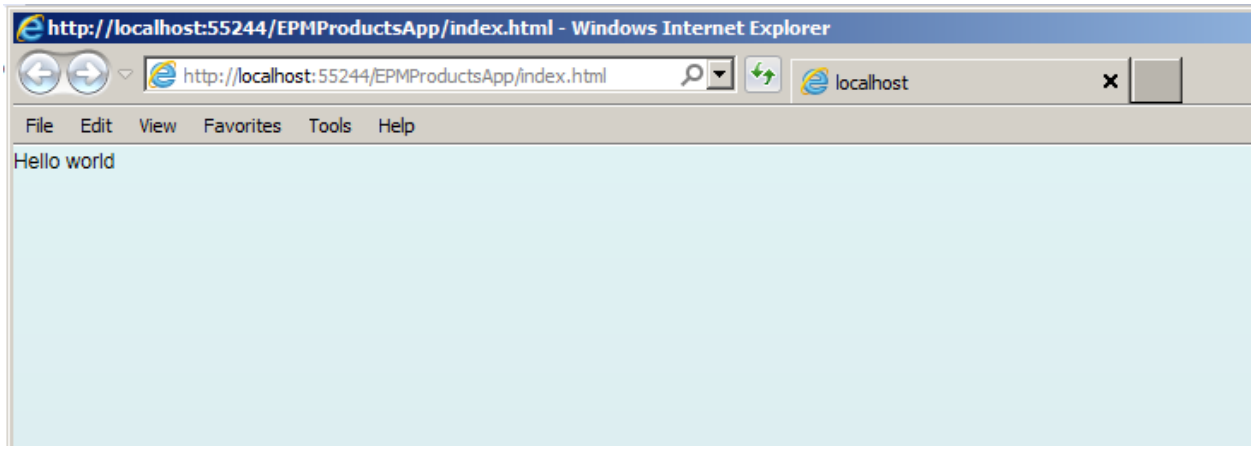
1. *The Resource library file path in the index.html*
2. *And changing the file reference from sap.m to sap.ui.commons*

This becomes clearer to identify such issues when you understand debugging SAPUI5 applications.

Debugging the application becomes easier in an external browser. You can launch the external browser by clicking the open in external browser button.



This will launch your default browser which in my case is Internet Explorer. If you want you can even copy this url and open the page in your favorite browser. Firefox and Chrome are two browsers which are preferred by developers because of a number of developer extensions the provide.



The above was a test to check that the application is functioning properly on the frontend local development front.

Testing the SAPUI5 Application on the ABAP Backend Server.

The next part is a quick test to confirm this application will work fine at the Application server.

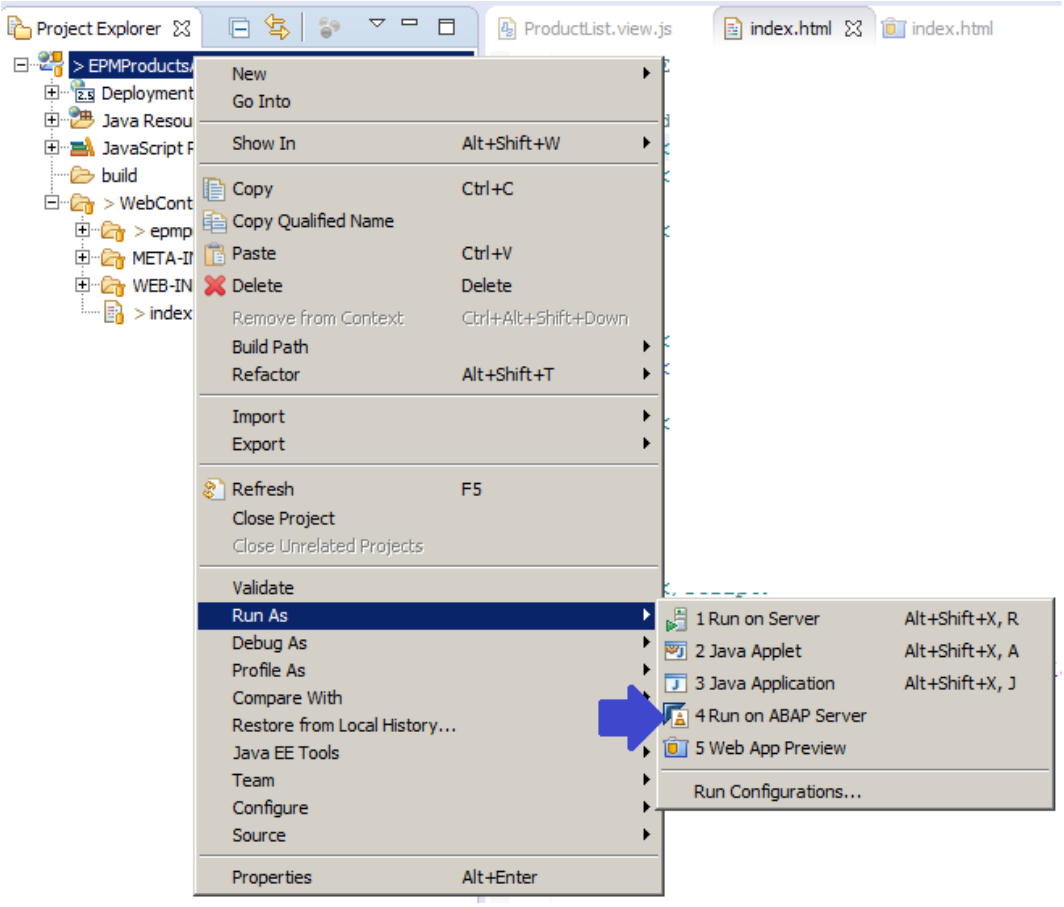
One point to note here is that when you are developing SAPUI5 applications with the SAPUI5 Tools the code completion and application preview features are based on the SAPUI5 runtime libraries, which has been installed in your Eclipse installation. However after you have deployed the application to the ABAP server and execute it there, it will use the SAPUI5 runtime libraries, which have been installed on the ABAP server.

There are a couple of ways to test your application on the ABAP App server. We will take a simpler route here.

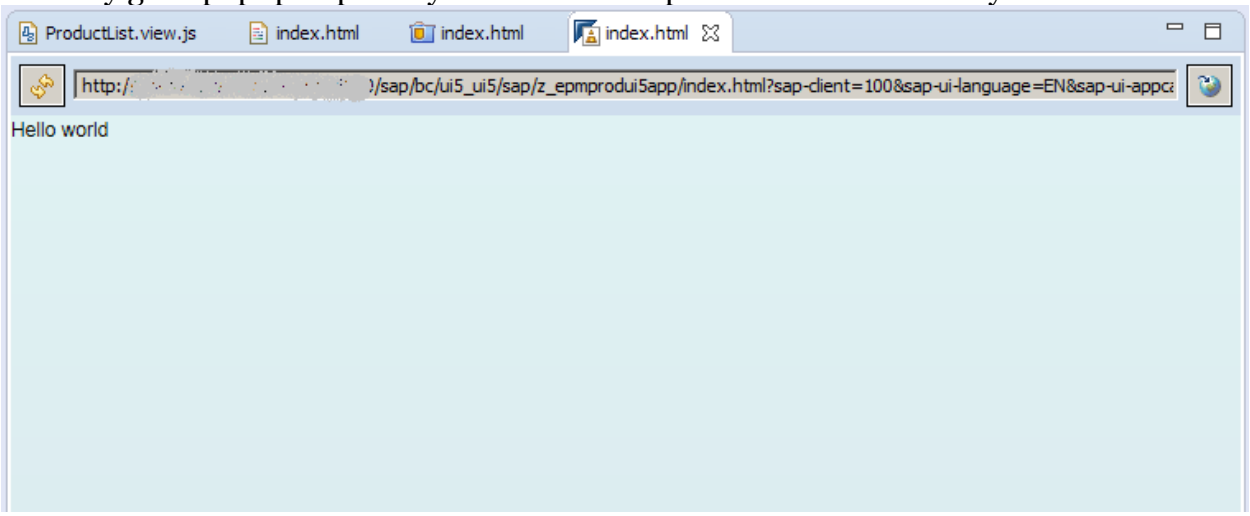
To do so follow the steps below.

1. In the Project Explorer, Right click on the Application Name and Select – Run As/Run on Application server.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



2. You may get a popup to put in your user id and password for the SAP system.



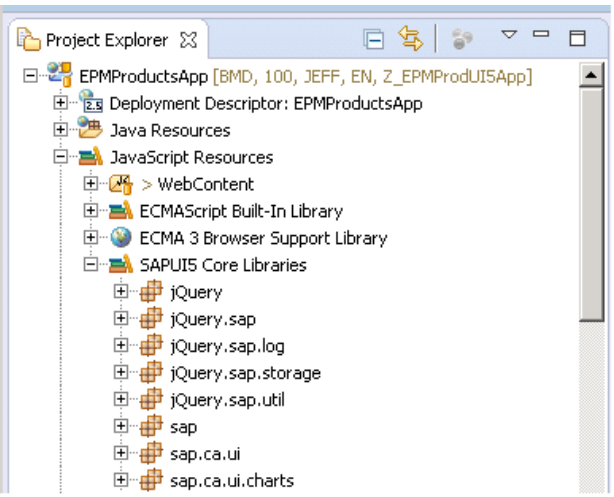
3. So here we see the same output as the one we got from our local Eclipse.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

Developing SAPUI5 Application to consume OData Gateway Service.

The SAPUI5 application follows a model-view-controller architecture.

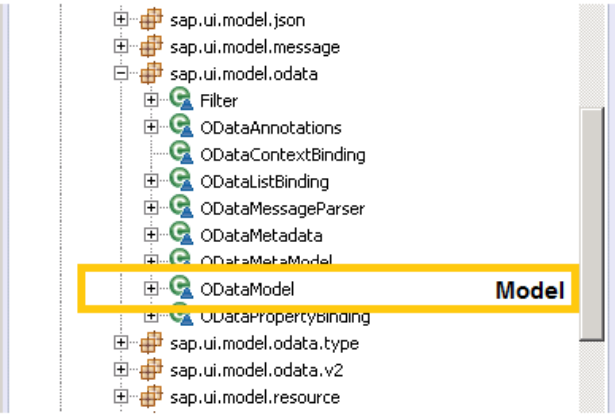
If you are uncertain about this architecture you can get a simpler understanding from the blog post here.



In the case of an SAPUI5 application it is farily easy to identify each of these components.

The Controller and View are developed as JavaScript files while the Model is our OData.

So the Controller in our application is the ProductList.controller.js file and the View is the Productlist.view.js file. The Model is the OData backed service which is going to provide the underlying data to the application.

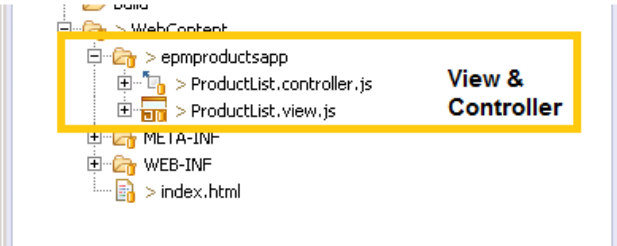


The program flow is as follows.

At the start of the application the SAPUI5 initially calls the view controller's

We will first connect to out backend OData service and retrieve data from it.

Once we get the data model we can then play around with the data in the <view/controller> and control how it will be displayed to the end user.



We can then use various field elements or form controls given you use by the SAPUI Library like the Text Field, Data Table, Dropdown Box etc.

So to First step, in our cod, lets connect to our backend OData model.

We will leverage the onInit method of the controller as that is what controls the application.

End to End Guide to building OData Service and Consuming it via SAPUI5 application

ProductList.view.js | index.html | index.html | *ProductList.controller.js

```
1  sap.ui.controller("epmproductsapp.ProductList", {
2
3  /**
4  * Called when a controller is instantiated and its View controls (if available) are already created.
5  * Can be used to modify the View before it is displayed, to bind event handlers and do other one-time initia
6  * @memberOf epmproductsapp.ProductList
7  */
8      onInit: function() {
9  // URL of the OData service - IMPORTANT: relative to the server
10     var sServiceUrl = "/sap/opu/odata/sap/z_epm_products_srv/";
11
12 // create OData model instance with service URL and JSON format
13     var oModel = new sap.ui.model.odata.ODataModel(
14             sServiceUrl, true, "[illegible]", "[illegible]");
15     sap.ui.getCore().setModel(oModel);
16     },
17
18 /**
19 * Similar to onAfterRendering, but this hook is invoked before the controller's View is re-rendered
20 * (NOT before the first rendering! onInit() is used for that one!).
21 * @memberOf epmproductsapp.ProductList
22 */
```

1 Service URL for OData Service

2 Model instantiation refering to OData service

So where did we get the URL value for the ServiceURL.

Remember this URL was generated when we had registered our OData service in the first part of this document.

If you don't remember this path then you can quickly find it by navigating to the service path

1. Execute transaction SICF in your SAP instance and navigate to the service.

Sap/opu/odata/sap/z_emp_products_srv

Virtuelle Hosts / Services	Documentation	Referenz Service
default_host	VIRTUAL DEFAULT HOST	
sap	SAP NAMESPACE; SAP IS OBLIGED NOT T...	
option	RESERVED SERVICES AVAILABLE GLOBALLY	
public	PUBLIC SERVICES	
0001_langes_feld		
ap	Application Platform	
A_NEW_INTAL1		/default_host/sap/A_FRISCH
opu	OData for SAP Products	
odata	Standard Mode	
iwfnd	Namespace	
sap	Namespace	
daag_mnggrp	DAAG_MNGGRP	
daag_monobj	DAAG_MONOBJ	
epm_lanes_demo_srv	EPM_LANES_DEMO_SRV	
epm_oia_apps_gw_service_s	EPM_OIA_APPS_GW_SERVICE_SRV	
upstream_def	UPSTREAM_DEF	
zgw_product_srv	ZGW_PRODUCT_SRV	
z_epm_products_srv	Z_EPM_PRODUCTS_SRV	

End to End Guide to building OData Service and Consuming it via SAPUI5 application

So now in the Applications core, we have set the model. We cannot add a UI Element on the View and bind it to this model.

To do this

1. In the Application View ProductList.view.js, in the createContent method we will
   a. instantiate a table control
   b. define columns for this table
   c. Bind the columns of this table to the Model (OData entity set)

Remember in the earlier part we had added a Hello World statement here. But this time we will instantiate a Table Object here

2. Comment the Hello world statement
```
//return new sap.ui.commons.TextView({text:"Hello world"});
```

3. And the following code (Below I will walk through the code step by step here so that you understand the main points)

```
// load table module, alternatively add sap.ui.table to
// the central bootstrap
jQuery.sap.require("sap.ui.table.Table");

// create table control with properties
var oTable = new sap.ui.table.Table({
      width : "100%",
      rowHeight : 50,
      title : "List of Products",
      selectionMode : sap.ui.table.SelectionMode.None
      });

// define the columns, which should be displayed
oTable.addColumn(new sap.ui.table.Column({
      width : "80px",
      flexible : false,
      template : new sap.ui.commons.Image({
      height : "45px", src : "{ProductPicUrl}"
      })
}));

oTable.addColumn(new sap.ui.table.Column({
      label : new sap.ui.commons.Label({
      text : "Product ID"
      }),
      template : new sap.ui.commons.TextView({
      text : "{ProductId}"
            }),
      sortProperty : "ProductId"
            }
```

```
// alternatively (instead of curly braces syntax for property binding):
// sap.ui.commons.TextView().bindText("ProductID")
// alternatively:
// sap.ui.commons.TextView().bindProperty("text","ProductID")
      ));

oTable.addColumn(new sap.ui.table.Column({
      label : new sap.ui.commons.Label({
      text : "Product Name"
      }),
      template : new sap.ui.commons.TextView({
      text : "{Name}"
      }),
      sortProperty : "Name"
}));

oTable.addColumn(new sap.ui.table.Column({
      label : new sap.ui.commons.Label({
      text : "Price",
      textAlign : sap.ui.core.TextAlign.End
      }),
      template : new sap.ui.commons.TextView({
      text : "{Price}",
      textAlign : sap.ui.core.TextAlign.End
      }),
      sortProperty : "Price"
      }));

oTable.addColumn(new sap.ui.table.Column({
      label : new sap.ui.commons.Label({
      text : "Currency"
      }),
      template : new sap.ui.commons.TextView({
      text : "{CurrencyCode}"
      }),
      sortProperty : "CurrencyCode"
      }));

oTable.addColumn(new sap.ui.table.Column({
      label : new sap.ui.commons.Label({
      text : "Supplier"
      }),
      template : new sap.ui.commons.TextView({
      text : "{SupplierName}"
      }),
      sortProperty : "SupplierName"
      }));

// bind table rows to the OData entity set
oTable.bindRows("/EpmProducts");
return oTable;
// alternatively:
// oTable.bindAggregation(sName, oBindingInfo)
// oTable.bindAggregation( "rows", "/EpmProducts" );
```

End to End Guide to building OData Service and Consuming it via SAPUI5 application

The code is clear enough, but for further understanding go through the screen shots below

```
16      //return new sap.ui.commons.TextView({text:"Hello world"});
17
18      // load table module, alternatively add sap.ui.table to
19      // the central bootstrap
20      jQuery.sap.require("sap.ui.table.Table");
```

Load the Table ui component

```
21
22      // create table control with properties
23      var oTable = new sap.ui.table.Table({
24      width : "100%",
25      rowHeight : 50,
26      title : "List of Products",
27      selectionMode : sap.ui.table.SelectionMode.None
28      });
29
```

Instantiate a Table Object

```
30      // define the columns, which should be displayed
31      oTable.addColumn(new sap.ui.table.Column({
32          width : "80px",
33          flexible : false,
34          template : new sap.ui.commons.Image({
35          height : "45px", src : "{ProductPicUrl}"
36          })
37      }));
```

Instantiate and add individual columns to the Table

```
38
39          oTable.addColumn(new sap.ui.table.Column({
40              label : new sap.ui.commons.Label({
41              text : "Product ID"
42              }),
43              template : new sap.ui.commons.TextView({
44              text : "{ProductId}"
45              }),
46              sortProperty : "ProductId"
47              }
48          // alternatively (instead of curly braces syntax for property binding):
49          // sap.ui.commons.TextView().bindText("ProductID")
50          // alternatively:
51          // sap.ui.commons.TextView().bindProperty("text","ProductID")
52          });
53
```

Adding Product ID Columns

```
96      // bind table rows to the OData entity set
97      oTable.bindRows("/EpmProducts");
98      return oTable;
99      // alternatively: // oTable.bindAggregation(sName, oBindingInfo)
100     // oTable.bindAggregation( "rows", "/EpmProducts" );
```

Bind Table to OData Entity

End to End Guide to building OData Service and Consuming it via SAPUI5 application

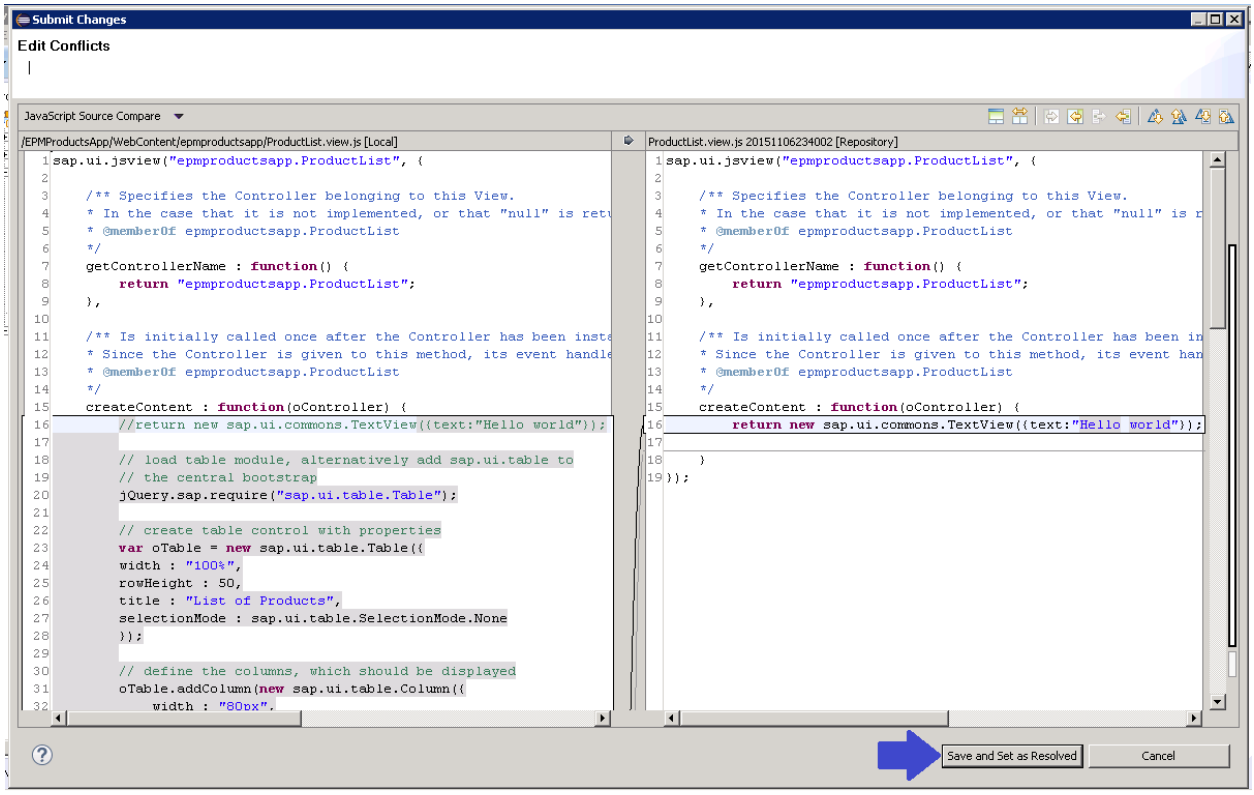
Let's test our application now.

Before that we need to transfer out code to the server.

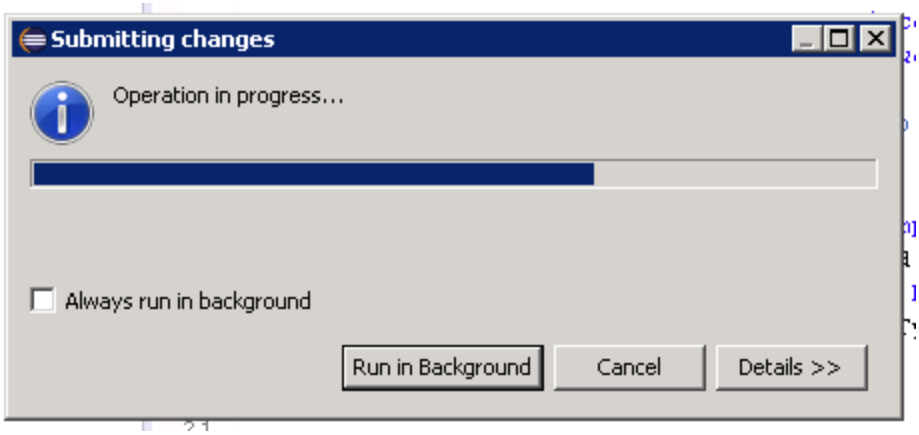
So right click on the Application and select Team/Submit.

When you submit the code to the backend, the system will do a compare with the old and new code and give you a conflict message. You can click Edit Conflicts button to verify your changes. This will highlight the changes for you.

Once you are convinced that the changes are all correct you can then click the Save and Set as Resolved.



End to End Guide to building OData Service and Consuming it via SAPUI5 application

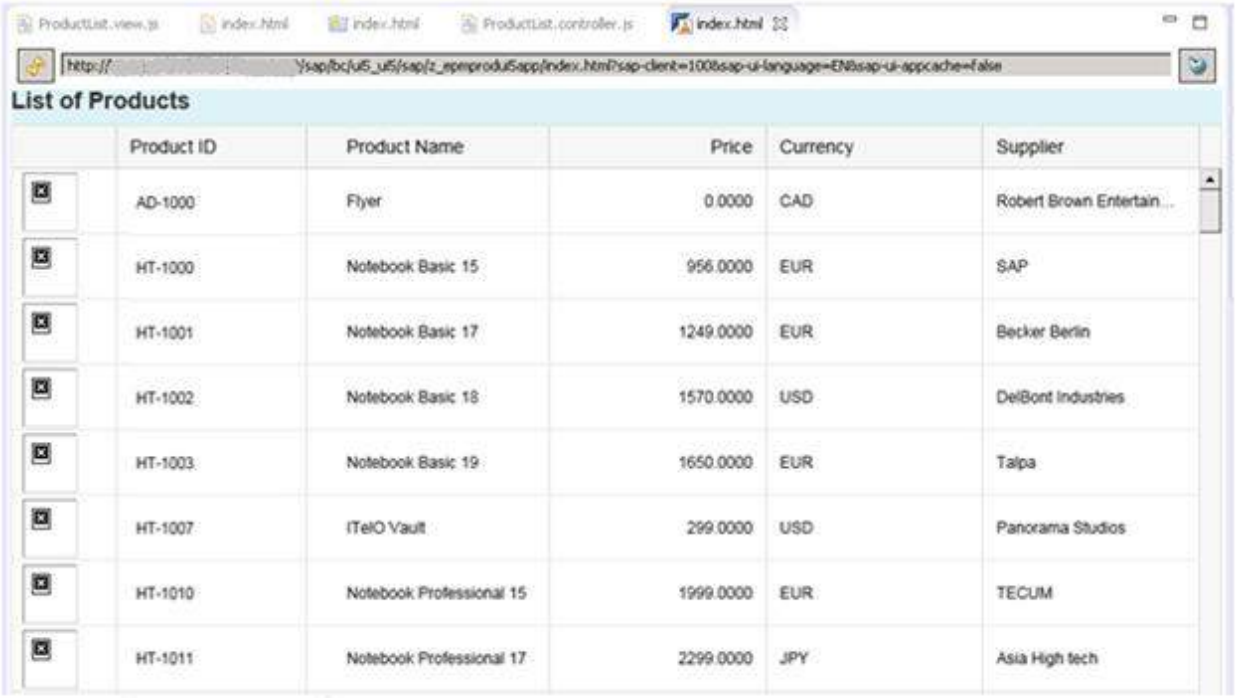


Now Test your application.

If you have the test browser open then you can just hit the refresh button to see your changes

After refresh.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



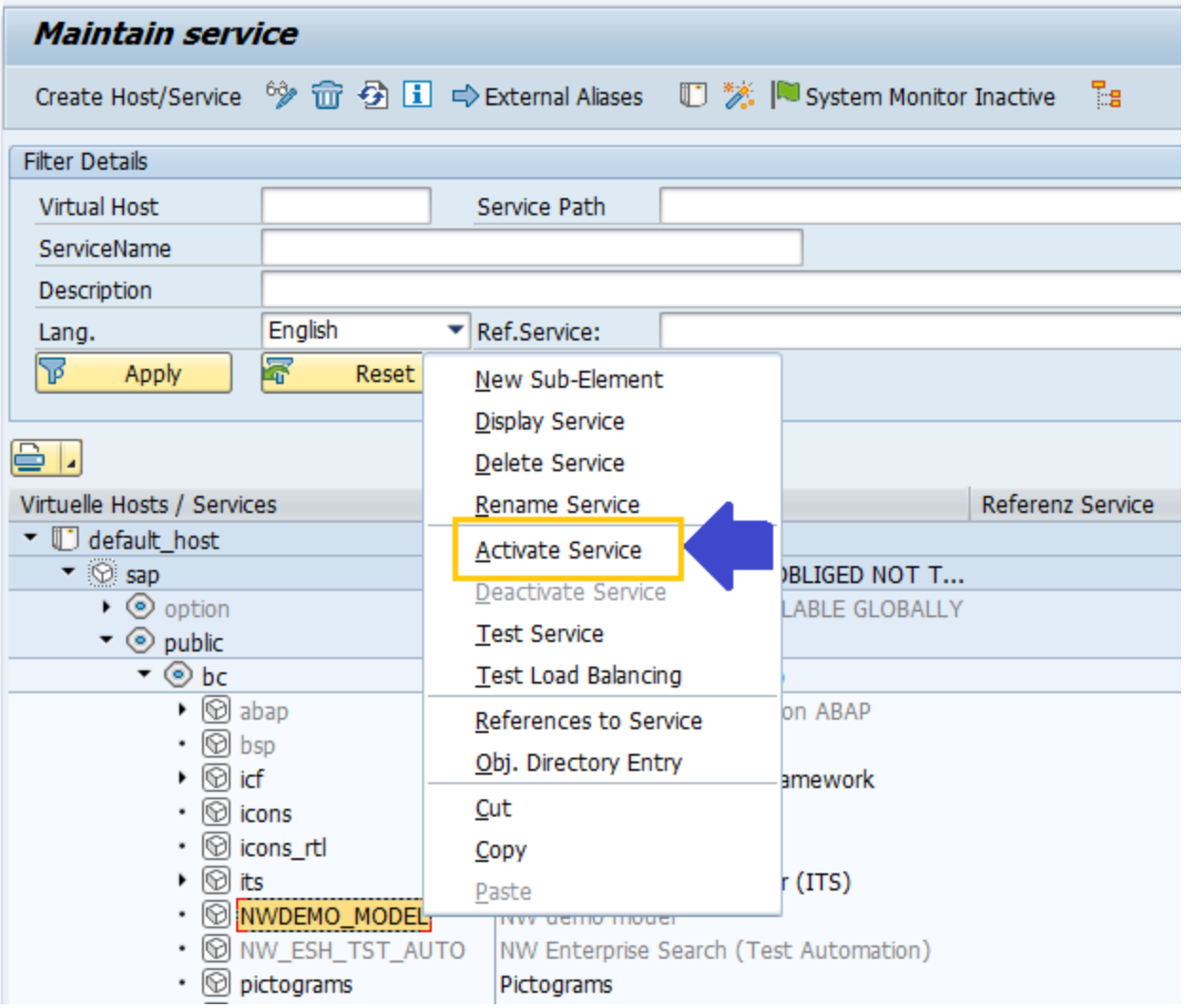
Product ID	Product Name	Price	Currency	Supplier
AD-1000	Flyer	0.0000	CAD	Robert Brown Entertain...
HT-1000	Notebook Basic 15	956.0000	EUR	SAP
HT-1001	Notebook Basic 17	1249.0000	EUR	Becker Berlin
HT-1002	Notebook Basic 18	1570.0000	USD	DelBont Industries
HT-1003	Notebook Basic 19	1650.0000	EUR	Talpa
HT-1007	ITelO Vault	299.0000	USD	Panorama Studios
HT-1010	Notebook Professional 15	1999.0000	EUR	TECUM
HT-1011	Notebook Professional 17	2299.0000	JPY	Asia High tech

The Images are not appearing here because the appropriate service for Images is inactive in my server. Usually you will find this service at the following location.

/sap/public/bc/NWDEMO_MODEL/

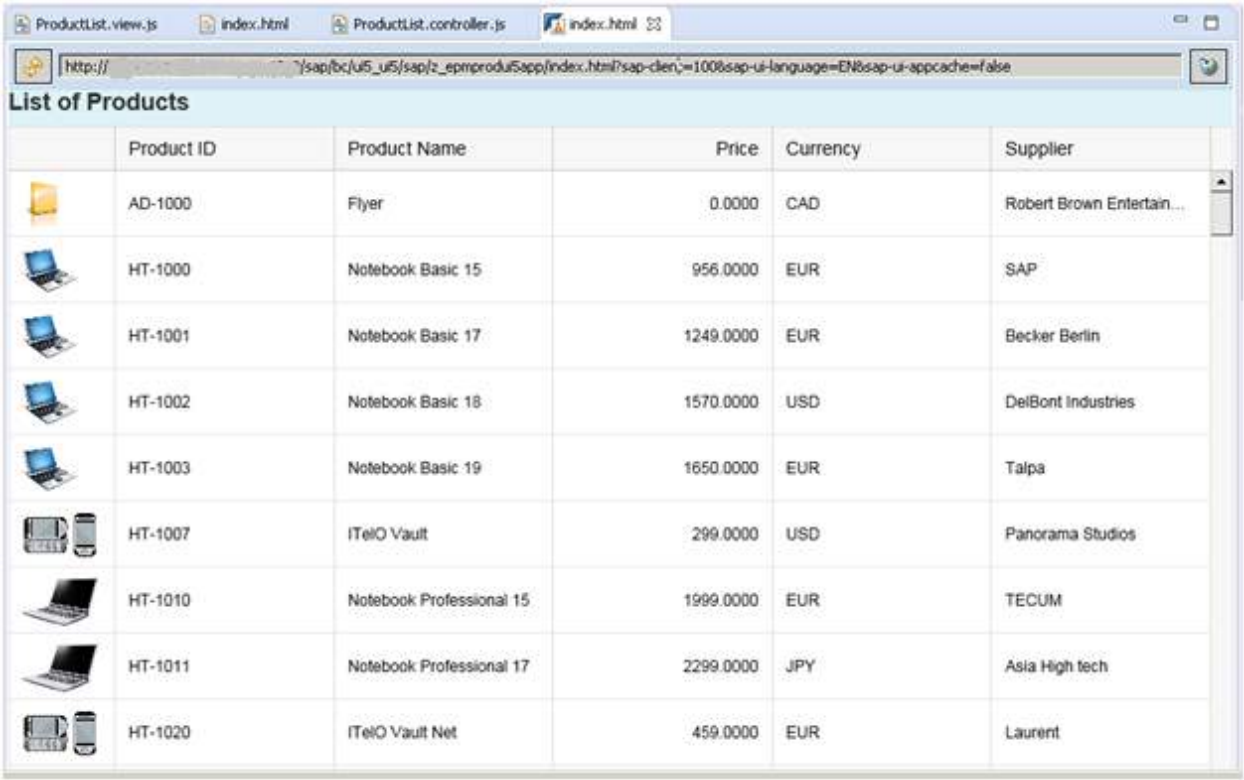
TO enable this service execute the transaction SICF and navigate to the path and activate the service.

End to End Guide to building OData Service and Consuming it via SAPUI5 application



Now we can refresh the application again. This time the images appear in the table

End to End Guide to building OData Service and Consuming it via SAPUI5 application



Product ID	Product Name	Price	Currency	Supplier
AD-1000	Flyer	0.0000	CAD	Robert Brown Entertain...
HT-1000	Notebook Basic 15	956.0000	EUR	SAP
HT-1001	Notebook Basic 17	1249.0000	EUR	Becker Berlin
HT-1002	Notebook Basic 18	1570.0000	USD	DelBont Industries
HT-1003	Notebook Basic 19	1650.0000	EUR	Talpa
HT-1007	ITelO Vault	299.0000	USD	Panorama Studios
HT-1010	Notebook Professional 15	1999.0000	EUR	TECUM
HT-1011	Notebook Professional 17	2299.0000	JPY	Asia High tech
HT-1020	ITelO Vault Net	459.0000	EUR	Laurent

Our Basic application to display the list of products is complete.

As a further step to this we can implement event handling methods to handle click events and drill down navigation on the application.

If you are interested to learn that then I have explained that process in my blog post.

This application is now fully functional and your company employees (who have a valid user id and password) can access this application and checkout the items available in the product catalogue.

Thank you for Referring to this guide. I hope it was helpful for you. In case you have any feedback or issues regarding the use of this guide –

Contact me on my blog www.beginners-sap.com

End to End Guide to building OData Service and Consuming it via SAPUI5 application

References Materials used in making this guide:

ODATA Services

- Step by Step Guide on OData Services
- End to End Development Example in SAP Netweaver 7.4 & SAP HANA
- Building new ODATA Services in 3 Quick Steps
- Rebooting your User Interfaces with SAP Netweaver Gateway
- Take Advantage of Cross-Platform, Cross-Device Access While Keeping Your Data Secure with SAP NetWeaver Gateway

UI Development Toolkit of HTML5 (SAPUI5)

- UI5 Developer API Documentation Reference

UI Add-on for SAP NetWeaver

- UI Add-On for SAP NetWeaver Product Documentation:
- SAP note 1759682, central note for the UI add-on for SAP NetWeaver with up-to-date information
- And off course – Numerous blogs referred to at SCN.SAP.COM