



BITI: Building Services for SAP Gateway

Graham Robinson

Yelcho Systems Consulting

INTRODUCTION

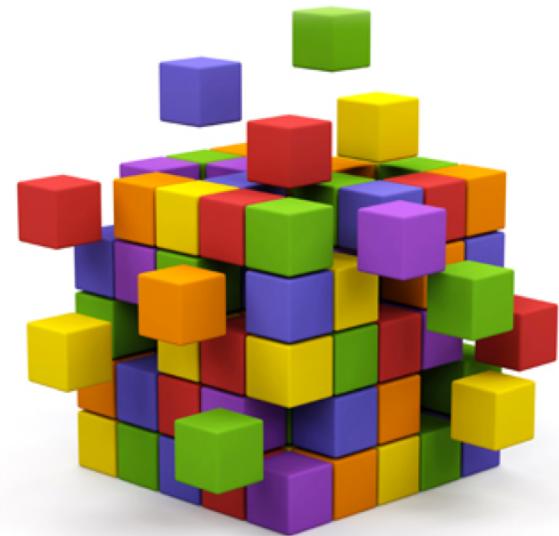


@grahamrobbo



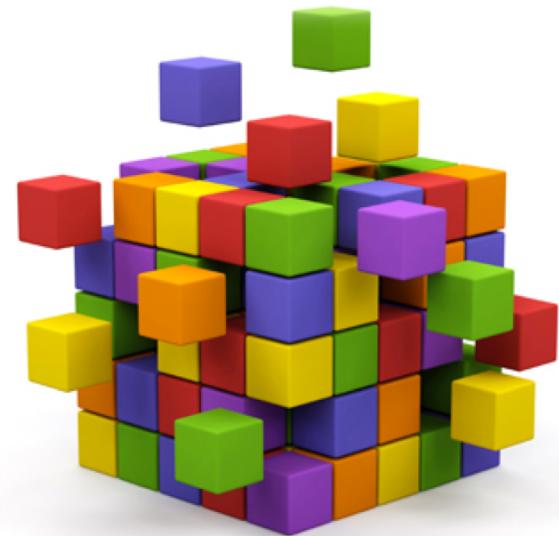
AGENDA

- Some Warnings
- Service Design
- Code patterns for reusability
- Code patterns for extensibility



AGENDA

- Some Warnings
- Service Design
- Code patterns for reusability
- Code patterns for extensibility



WARNING – THIS IS NOT FOR BEGINNERS



ASUG 25

WARNING - HOMEWORK

https://grahamrobbo.github.io/teched16_example/

The screenshot shows a presentation slide for SAP TECHED Las Vegas, Sept. 19-23, 2016. The title is "UX202 - Building Services for Gateway". Below the title are three buttons: "Slides", "Sample code", and "abapGit". The main content area starts with a "Welcome" message, followed by a note about the ABAP code being available for download. A section titled "*** Update 3 - April 2017 ***" mentions re-delivery at ASUG Webcast and SAUG Brisbane Conference. At the bottom, there are logos for ASUG Webcasts and the SAP Australian User Group (saug).

Welcome.

Thanks for your interest in my presentation UX202 - Building Services for Gateway. You can grab the ABAP code here and install it on your own system to look at it more closely and see it in action.

*** Update 3 - April 2017 ***

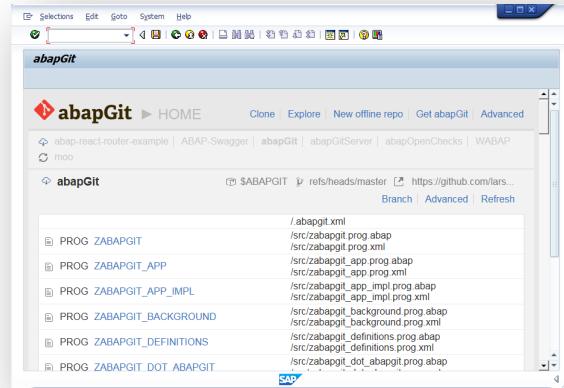
I was asked to re-deliver this session on a upcoming ASUG Webcast and also at the SAUG Brisbane Conference in May 2017.

I have therefore done some minor code refactoring - but most importantly I have updated the slides so that they now match the sample code.

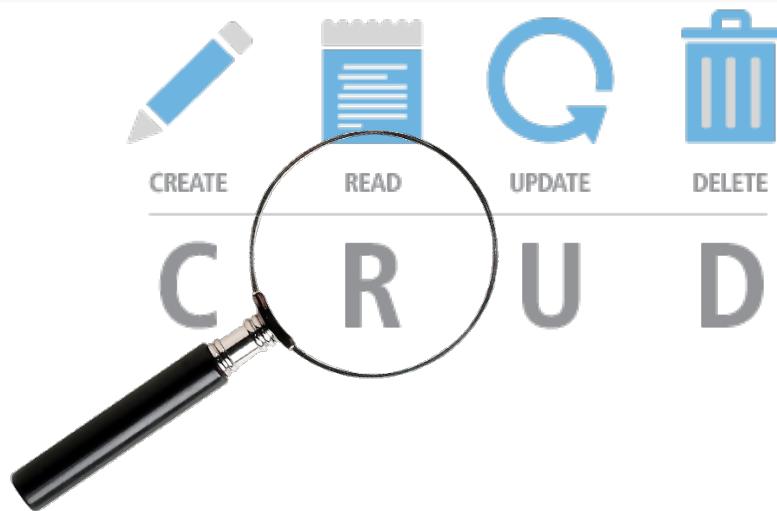
ASUG | Webcasts

saug SAP Australian User Group

To install you will also need abapGit
<https://abapGit.org>



WARNING – FOCUS ON CONSUMPTION



These code patterns can apply to all actions – but I will focus on consumption

GET_ENTITYSET is more complex because it needs to supports oData query options

CDS views (NW 7.40 SP8) provide other consumption options

WARNING – I DO NOT HAVE ALL THE ANSWERS

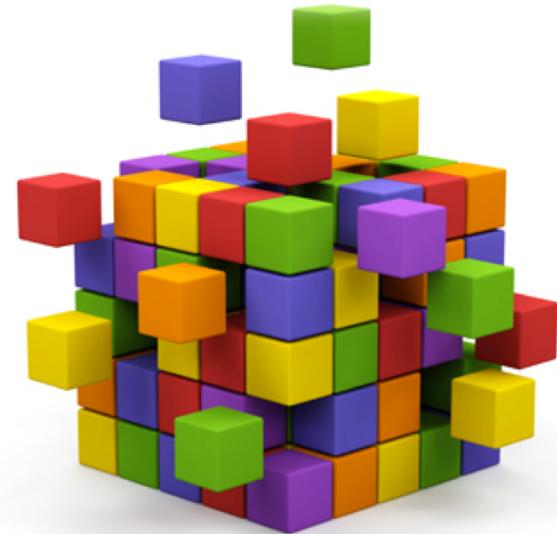
- \ (ツ) / -



ASUG 25

AGENDA

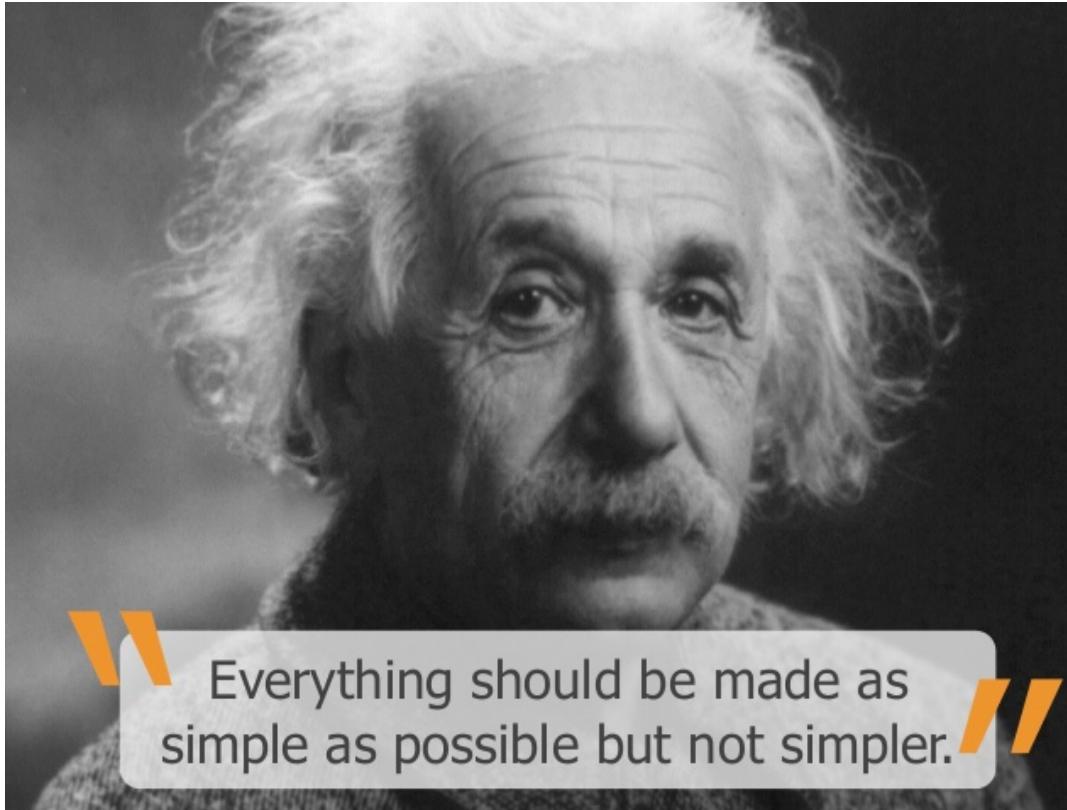
- Some Warnings
- Service Design
- Code patterns for reusability
- Code patterns for extensibility



SERVICE DESIGN PRINCIPLES

- Design for Consumers
- Think oData i.e. Entities, Associations, etc.
- Leave SAP domain knowledge at the door
- REST API's should be self-describing
- Function Imports should be for special cases

GENERAL THEORY OF API DESIGN



Everything should be made as
simple as possible but not simpler.



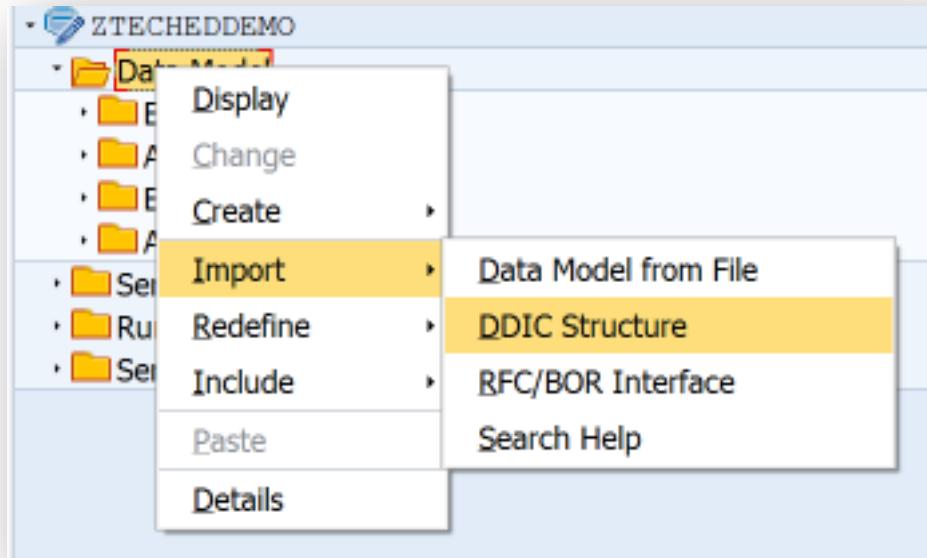
ASUG 25

CONSUMER VIEW OF API

```
<EntityType Name="Order">
  <Key>
    <PropertyRef Name="OrderID"/>
  </Key>
  <Property Name="OrderID" Type="Edm.Int32" Nullable="false" p6:StoreGeneratedPattern="Identity"
  xmlns:p6="http://schemas.microsoft.com/ado/2009/02/edm/annotation"/>
  <Property Name="CustomerID" Type="Edm.String" MaxLength="5" FixedLength="true" Unicode="true"/>
  <Property Name="EmployeeID" Type="Edm.Int32"/>
  <Property Name="OrderDate" Type="Edm.DateTime"/>
  <Property Name="RequiredDate" Type="Edm.DateTime"/>
  <Property Name="ShippedDate" Type="Edm.DateTime"/>
  <Property Name="ShipVia" Type="Edm.Int32"/>
  <Property Name="Freight" Type="Edm.Decimal" Precision="19" Scale="4"/>
  <Property Name="ShipName" Type="Edm.String" MaxLength="40" FixedLength="false" Unicode="true"/>
  <Property Name="ShipAddress" Type="Edm.String" MaxLength="60" FixedLength="false" Unicode="true"/>
  <Property Name="ShipCity" Type="Edm.String" MaxLength="15" FixedLength="false" Unicode="true"/>
  <Property Name="ShipRegion" Type="Edm.String" MaxLength="15" FixedLength="false" Unicode="true"/>
  <Property Name="ShipPostalCode" Type="Edm.String" MaxLength="10" FixedLength="false" Unicode="true"/>
  <Property Name="ShipCountry" Type="Edm.String" MaxLength="15" FixedLength="false" Unicode="true"/>
  <NavigationProperty Name="Customer" Relationship="NorthwindModel.FK_Orders_Customers" ToRole="Customers" FromRole="Orders"/>
  <NavigationProperty Name="Employee" Relationship="NorthwindModel.FK_Orders_Employees" ToRole="Employees" FromRole="Orders"/>
  <NavigationProperty Name="Order_Details" Relationship="NorthwindModel.FK_Order_Details_Orders" ToRole="Order_Details" FromRole="Orders"/>
  <NavigationProperty Name="Shipper" Relationship="NorthwindModel.FK_Orders_Shippers" ToRole="Shippers" FromRole="Orders"/>
</EntityType>
```



BIND ENTITY TO DDIC STRUCTURE



BIND ENTITY TO DDIC STRUCTURE

Wizard Step 1 of 3: Import from DDIC Structure

Create an Entity Type or Complex Type

Name: Customer

Entity Type Complex Type

Import from ABAP Structure

ABAP Structure: ZDEMO_CUSTOMER

Create Default Entity Set

 Next  Cancel



BIND ENTITY TO DDIC STRUCTURE

Wizard Step 3 of 3: Import from DDIC Structure

Modify Entity Type

IsEnti	Complex/Entity Type Name	ABAP Name	Is Key	Type	Name	Label
<input checked="" type="checkbox"/>	Customer	BP_ID	<input checked="" type="checkbox"/>	CHAR	CustomerId	Business Partner ID
<input checked="" type="checkbox"/>	Customer	COMPANY_NAME	<input type="checkbox"/>	CHAR	Name	Company
<input checked="" type="checkbox"/>	Customer	STREET	<input type="checkbox"/>	CHAR	Street	Street
<input checked="" type="checkbox"/>	Customer	CITY	<input type="checkbox"/>	CHAR	City	City
<input checked="" type="checkbox"/>	Customer	POSTAL_CODE	<input type="checkbox"/>	CHAR	PostalCode	Postal Code
<input checked="" type="checkbox"/>	Customer	COUNTRY	<input type="checkbox"/>	CHAR	CountryId	Country
<input checked="" type="checkbox"/>	Customer	COUNTRY_TEXT	<input type="checkbox"/>	CHAR	Country	Country
<input checked="" type="checkbox"/>	Customer	SEGMENTATION	<input type="checkbox"/>	CHAR	Segmentation	Segmentation

Back  Finish  Cancel



BIND ENTITY TO DDIC STRUCTURE

Wizard Step 2 of 3: Import from DDIC Structure

Select Parameter(s)

Data Source Parameter	Assign Structure	Description	Type	Length	Decimals	Import Search Help	Search Help	Output Length	Conv...
ZDEMO_CUSTOMER	<input type="checkbox"/>	ZDEMO_CUSTOMER				<input type="checkbox"/>			
NODE_KEY	<input type="checkbox"/>	Node Key	RAW	16		<input type="checkbox"/>		32	
BP_ID	<input checked="" type="checkbox"/>	Business Partner ID	CHAR	10		<input type="checkbox"/>	H_EPM_BP	10 ALPHA	
COMPANY_NAME	<input checked="" type="checkbox"/>	Company	CHAR	80		<input type="checkbox"/>	H_EPM_BP_NAME	80	
STREET	<input type="checkbox"/>	Street	CHAR	60		<input type="checkbox"/>		60	
CITY	<input type="checkbox"/>	City	CHAR	40		<input type="checkbox"/>		40	
POSTAL_CODE	<input type="checkbox"/>	Postal Code	CHAR	10		<input type="checkbox"/>		10	
COUNTRY	<input type="checkbox"/>	Country	CHAR	3		<input type="checkbox"/>	F4_INTCA	3	
COUNTRY_TEXT	<input type="checkbox"/>	Country	CHAR	50		<input type="checkbox"/>		50	
SEGMENTATION	<input type="checkbox"/>	Segmentation	CHAR	1		<input type="checkbox"/>		1	

Back Next Cancel



BIND ENTITY TO DDIC STRUCTURE

Wizard Step 3 of 3: Import from DDIC Structure

Modify Entity Type

IsEnti	Complex/Entity Type Name	ABAP Name	Is Key	Type	Name	Label
<input checked="" type="checkbox"/>	Customer	BP_ID	<input checked="" type="checkbox"/>	CHAR	CustomerId	
<input checked="" type="checkbox"/>	Customer	COMPANY_NAME	<input type="checkbox"/>	CHAR	Name	
<input checked="" type="checkbox"/>	Customer	STREET	<input type="checkbox"/>	CHAR	Street	
<input checked="" type="checkbox"/>	Customer	CITY	<input type="checkbox"/>	CHAR	City	
<input checked="" type="checkbox"/>	Customer	POSTAL_CODE	<input type="checkbox"/>	CHAR	PostalCode	
<input checked="" type="checkbox"/>	Customer	COUNTRY	<input type="checkbox"/>	CHAR	CountryId	
<input checked="" type="checkbox"/>	Customer	COUNTRY_TEXT	<input type="checkbox"/>	CHAR	Country	
<input checked="" type="checkbox"/>	Customer	SEGMENTATION	<input type="checkbox"/>	CHAR	Segmentation	

Remove
label
suggestions

This means labels are determined at runtime rather than design-time

Back Finish Cancel



\$METADATA

Properties

Name	Is Key	Edm Core Type	Prec.	Scale	Max L.	Unit	Property Name	Creatable	Updatable	Sortable	Nullable	Filt.	Label	Complex Type Name	ABAP Field Name
CustomerId	<input checked="" type="checkbox"/>	Edm.String	0	0	10			<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	I	BP_ID
Name	<input type="checkbox"/>	Edm.String	0	0	80			<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	I	COMPANY_NAME
Street	<input type="checkbox"/>	Edm.String	0	0	60			<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	I	STREET
City	<input type="checkbox"/>	Edm.String	0	0	40			<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	I	CITY
PostCode	<input type="checkbox"/>	Edm.String	0	0	10			<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	I	POSTAL_CODE
CountryId	<input type="checkbox"/>	Edm.String	0	0	3			<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	I	COUNTRY
Country	<input type="checkbox"/>	Edm.String	0	0	50			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	I	COUNTRY_TEXT
Segmentation	<input type="checkbox"/>	Edm.String	0	0	1			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	I	SEGMENTATION

```
<EntityType Name="Customer" sap:content-version="1">
  <Key>
    <PropertyRef Name="CustomerId"/>
  </Key>
  <Property Name="CustomerId" Type="Edm.String" Nullable="false" MaxLength="10" sap:label="Business Partner ID" sap:creatable="false" sap:updatable="false"/>
  <Property Name="Name" Type="Edm.String" Nullable="false" MaxLength="80" sap:label="Company" sap:creatable="false" sap:updatable="false"/>
  <Property Name="Street" Type="Edm.String" Nullable="false" MaxLength="60" sap:label="Street" sap:creatable="false" sap:updatable="false"/>
  <Property Name="City" Type="Edm.String" Nullable="false" MaxLength="40" sap:label="City" sap:creatable="false" sap:updatable="false"/>
  <Property Name="PostCode" Type="Edm.String" Nullable="false" MaxLength="10" sap:label="Postal Code" sap:creatable="false" sap:updatable="false"/>
  <Property Name="CountryId" Type="Edm.String" Nullable="false" MaxLength="3" sap:label="Country" sap:creatable="false" sap:updatable="false"/>
  <Property Name="Country" Type="Edm.String" Nullable="false" MaxLength="50" sap:label="Country" sap:creatable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false"/>
  <Property Name="Segmentation" Type="Edm.String" Nullable="false" MaxLength="1" sap:label="Segmentation" sap:creatable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false"/>
  <NavigationProperty Name="SalesOrders" Relationship="ZDEMO_SRV.Customer_SalesOrder" FromRole="FromRole_Customer_SalesOrder" ToRole="ToRole_Customer_SalesOrder"/>
</EntityType>
```



DATES & TIMES

The screenshot shows the SAP Studio Properties view. It lists two properties: SalesOrderId and SalesOrderDate. SalesOrderId is marked as a key and has type Edm.String. SalesOrderDate has type Edm.DateTimeOffset.

Name	Is Key	Edm Core Type
SalesOrderId	<input checked="" type="checkbox"/>	Edm.String
SalesOrderDate	<input type="checkbox"/>	Edm.DateTimeOffset

```
<EntityType Name="SalesOrder" sap:content-version="1">
  <Key>
    <PropertyRef Name="SalesOrderId"/>
  </Key>
  <Property Name="SalesOrderId" Type="Edm.String" Nullable="false" MaxLength="10" sap:label="Sales Document" sap:creatable="false" sap:updatable="false"/>
  <Property Name="SalesOrderDate" Type="Edm.DateTimeOffset" Precision="7" sap:label="Document Date" sap:creatable="false" sap:updatable="false" sap:filterable="false"/>
  <NavigationProperty Name="Customer" Relationship="ZDEMO_SRV.Customer_SalesOrder" FromRole="ToRole_Customer_SalesOrder" ToRole="FromRole_Customer_SalesOrder"/>
  <NavigationProperty Name="SalesOrderItems" Relationship="ZDEMO_SRV.SalesOrder_Items" FromRole="FromRole_SalesOrder_Items" ToRole="ToRole_SalesOrder_Items"/>
</EntityType>
```

```
- <content type="application/xml">
  - <m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
    <d:SalesOrderId>60000008</d:SalesOrderId>
    <d:SalesOrderDate>2006-10-19T00:00:00Z</d:SalesOrderDate>
  </m:properties>
</content>
```

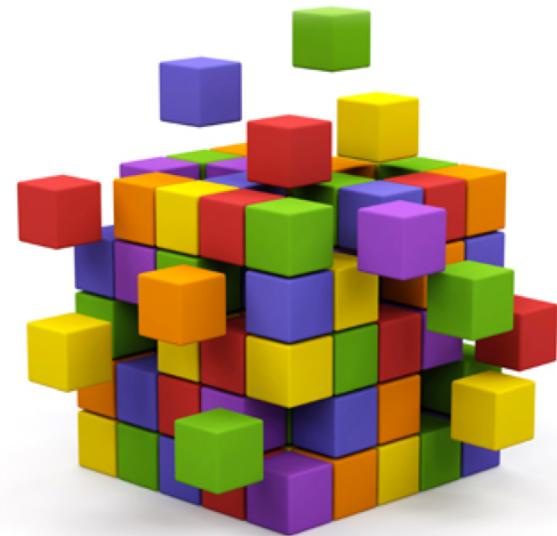
Use Edm.DateTimeOffset for date/time/timestamp properties

Don't be tempted to use Edm.String "for convenience"
Edm.DateTime replaced in oData V4 specification



AGENDA

- Some Warnings
- Service Design
- **Code patterns for reusability**
- Code patterns for extensibility



CODE PATTERNS FOR REUSABILITY

The first rule of reusability is
encapsulate all business
logic in
concrete classes



asug25

WHAT A CLASS IS NOT

An ABAP class that only contains static methods
is not really a class.....

... in all respects except object type it is a...

Function Group



CODE PATTERNS FOR REUSABILITY

The second rule of reusability
**ENCAPSULATE ALL BUSINESS
LOGIC IN
CONCRETE CLASSES**



asug 25

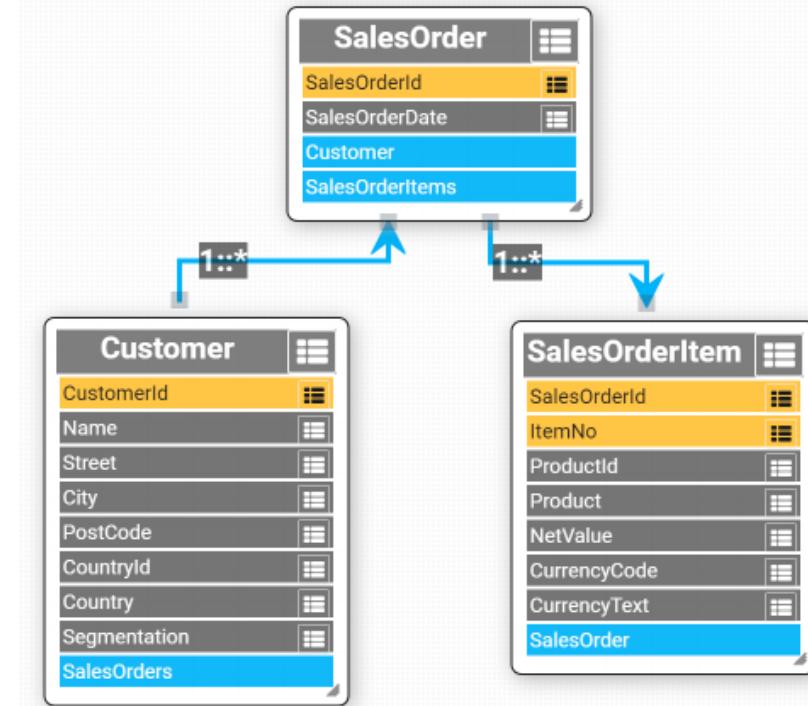
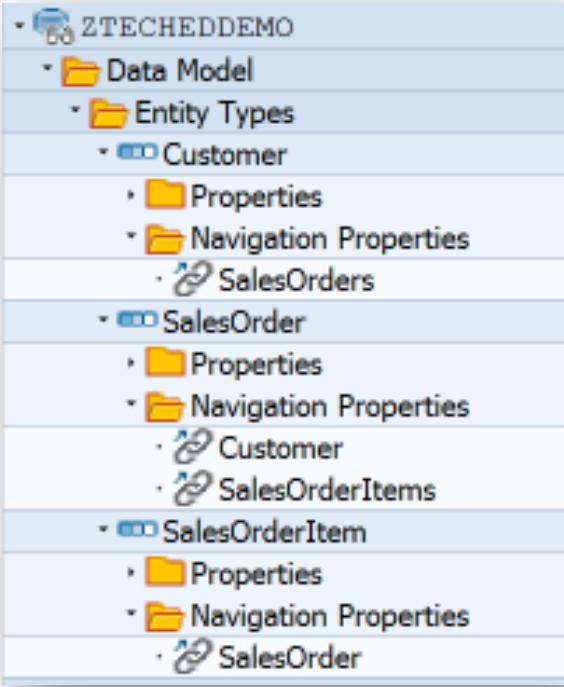
CODE PATTERNS FOR REUSABILITY

- I like the singleton design pattern in ABAP
 - Be aware that some people don't
 - I don't care – it works for me
- I like getters (& sometimes setters)
 - Be aware that some people don't
 - I don't care – it works for me



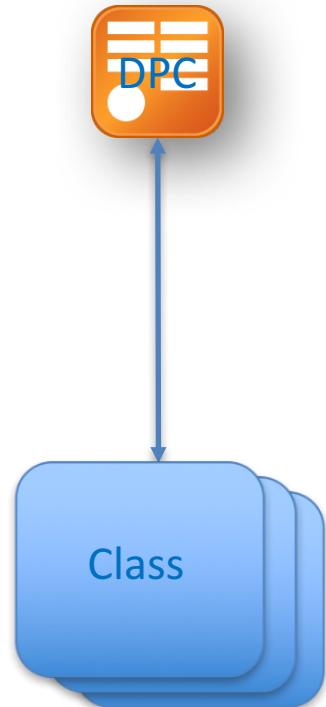
You can implement business logic however you like...
just remember the first & second rules of reusability

SAMPLE DATA MODEL

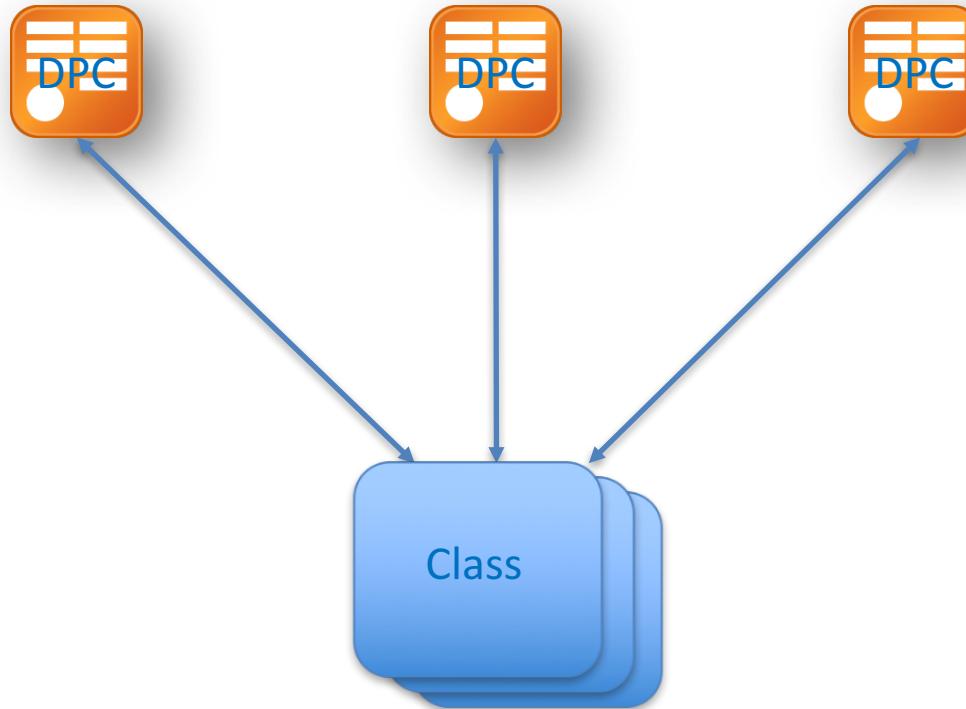


Based upon [SAP Enterprise Procurement Model](#)

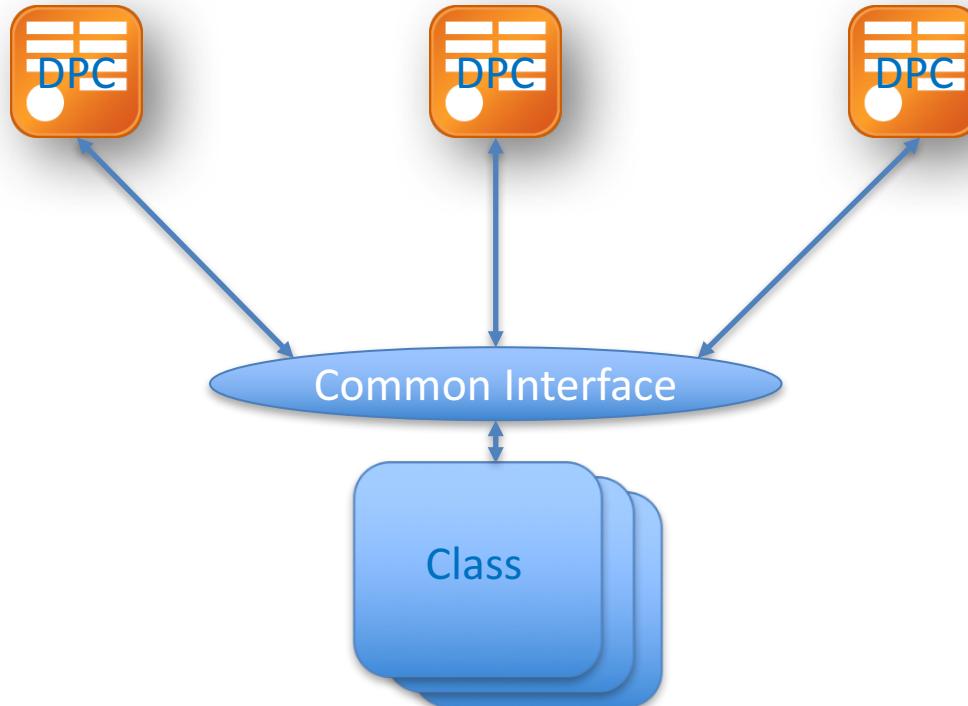
DATA PROVIDER CLASS



MULTIPLE CONSUMERS



MULTIPLE CONSUMERS



REPLICATE GW METHOD INTERFACE

Ty.	Parameter	Typing
IV	IV_ENTITY_NAME	TYPE STRING OPTIONAL
IV	IV_ENTITY_SET_NAME	TYPE STRING OPTIONAL
IV	IV_SOURCE_NAME	TYPE STRING OPTIONAL
IT	IT_FILTER_SELECT_OPTIONS	TYPE /IWBEPE/T_MGW_SELECT_OPTION OPTIONAL
IT	IT_ORDER	TYPE /IWBEPE/T_MGW_SORTING_ORDER OPTIONAL
IS	IS_PAGING	TYPE /IWBEPE/S_MGW_PAGING OPTIONAL
IT	IT_NAVIGATION_PATH	TYPE /IWBEPE/T_MGW_NAVIGATION_PATH OPTIONAL
IT	IT_KEY_TAB	TYPE /IWBEPE/T_MGW_NAME_VALUE_PAIR OPTIONAL
IV	IV_FILTER_STRING	TYPE STRING OPTIONAL
IV	IV_SEARCH_STRING	TYPE STRING OPTIONAL
IO	IO_TECH_REQUEST_CONTEXT	TYPE REF TO /IWBEPE/IF_MGW_REQ_ENTITYSET OPTIONAL
ER	ER_ENTITYSET	TYPE REF TO DATA
ES	ES_RESPONSE_CONTEXT	TYPE /IWBEPE/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_CONTEXT
/IWBEPE/CX	/MGW_BUSI_EXCEPTION	
/IWBEPE/CX	/MGW_TECH_EXCEPTION	

Method /IWBEPE/IF_MGW_APPL_SRV_RUNTIME~GET_ENTITYSET Active

```
1 | method /IWBEPE/IF_MGW_APPL_SRV_RUNTIME~GET_ENTITYSET.
```



AVOID USING DEPRECATED PARAMETERS

Ty.	Parameter	Typing	Description
IV	IV_ENTITY_NAME	TYPE STRING OPTIONAL	Obsolete
IV	IV_ENTITY_SET_NAME	TYPE STRING OPTIONAL	Obsolete
IV	IV_SOURCE_NAME	TYPE STRING OPTIONAL	Obsolete
IT	IT_FILTER_SELECT_OPTIONS	TYPE /IWBEPE/T_MGW_SELECT_OPTION OPTIONAL	table of select options - Obsolete
IT	IT_ORDER	TYPE /IWBEPE/T_MGW_SORTING_ORDER OPTIONAL	the sorting order - Obsolete
IS	IS_PAGING	TYPE /IWBEPE/S_MGW_PAGING OPTIONAL	paging structure - Obsolete
IT	IT_NAVIGATION_PATH	TYPE /IWBEPE/T_MGW_NAVIGATION_PATH OPTIONAL	table of navigation paths - Obsolete
IT	IT_KEY_TAB	TYPE /IWBEPE/T_MGW_NAME_VALUE_PAIR OPTIONAL	table for name value pairs - Obsolete
IV	IV_FILTER_STRING	TYPE STRING OPTIONAL	the filter as a string containing ANDs and ORs etc -Obsolete
IV	IV_SEARCH_STRING	TYPE STRING OPTIONAL	Obsolete
IO	IO_TECH_REQUEST_CONTEXT	TYPE REF TO /IWBEPE/IF_MGW_REQ_ENTITYSET OPTIONAL	
ER	ER_ENTITYSET	TYPE REF TO DATA	
ES	ES_RESPONSE_CONTEXT	TYPE /IWBEPE/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_CONTEXT	
/IWBEPE/CX	MGW_BUSI_EXCEPTION		business exception in mgw
/IWBEPE/CX	MGW_TECH_EXCEPTION		mgw technical exception

Method /IWBEPE/IF_MGW_APPL_SRV_RUNTIME~GET_ENTITYSET Active

```
1 | method /IWBEPE/IF_MGW_APPL_SRV_RUNTIME~GET_ENTITYSET.
```

<https://blogs.sap.com/2017/02/01/avoid-using-deprecated-sap-gateway-apis-in-your-odata-service-implementation/>



Nabi Zamani

COMMON INTERFACE

Interface ZIF_GW_METHODS Implemented / Active

Proper... Interfaces Attributes Metho... Events Types Aliases

Parameter Exception

Method Level M... Description

CREATE_ENTITY	Static Method	
CREATE_DEEP_ENTITY	Static Method	
DELETE_ENTITY	Static Method	
GET_ENTITY	Static Method	
GET_ENTITYSET	Static Method	
UPDATE_ENTITY	Static Method	
EXECUTE_ACTION	static Method	

Model

Message Container

Interface ZIF_GW_METHODS Implemented / Active

Property... Interfaces Attributes Methods Events Types Aliases

Parameters of Method GET_ENTITYSET

Properties

Parameter	Type	Pa... O...	Typing Method	Associated Type
IO_TECH_REQUEST_CONTEXT	Importing	<input type="checkbox"/>	Type Ref To	/IWBEPEP/IF_MGW_REQ_ENTITYSET
IO_MODEL	Importing	<input type="checkbox"/>	Type Ref To	ZCL_GW_MODEL
IO_MESSAGE_CONTAINER	Importing	<input checked="" type="checkbox"/>	Type Ref To	/IWBEPEP/IF_MESSAGE_CONTAINER
ET_ENTITYSET	Exporting	<input type="checkbox"/>	Type	DATA
ES_RESPONSE_CONTEXT	Exporting	<input type="checkbox"/>	Type	/IWBEPEP/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_CONTEXT

Untyped data object



ZCL_GW_MODEL

Class/Interface		ZCL_GW_MODEL		Implemented / Active	
Property...		Interfaces		Friends	
Attributes		Methods		Events	
Parameters		Exceptions		Sourcecode	
Method			Level	Visibility	M... Des
CONSTRUCTOR			Instance	Method	Public
GET_PROPERTY			Instance	Method	Public
GET_ENTITY_PROPERTIES			Instance	Method	Private

METHOD constructor.

```
DATA: lr_facade TYPE REF TO /iwbp/cl_mgw_dp_facade.  
lr_facade ?= runtime->get_dp_facade( ).  
  
mpc ?= lr_facade->/iwbp/if_mgw_dp_int_facade~get_model( ).
```

ENDMETHOD.

DPC->GET_MODEL

Class Builder: Class ZCL_DEMO_DPC_EXT Display

Parameter Type spec. Description

Ty.	Parameter	Type spec.	Description
	value(MODEL)	TYPE REF TO ZCL_GW_MODEL	
	/IWBEPE/CX_MGW_TECH_EXCEPTION		

Method GET_MODEL

```
1 METHOD get_model.
2
3   IF me->model IS NOT BOUND.
4     CREATE OBJECT me->model
5       EXPORTING
6         runtime = me.
7   ENDIF.
8
9   model = me->model.
10
11 ENDMETHOD.
```



CALL BO CLASS FROM DPC

Method	CUSTOMERS_GET_ENTITYSET	Active
1	METHOD customers_get_entityset.	
2	zcl_demo_customer=>zif_gw_methods~get_entityset(
3	EXPORTING	
4	io_tech_request_context = io_tech_request_context	
5	io_model = get_model()	
6	io_message_container = mo_context->get_message_container()	
7	IMPORTING	
8	et_entityset = et_entityset	
9	es_response_context = es_response_context).	
10	ENDMETHOD.	



ODATA OPERATIONS

	Identify Primary Key(s)	Instantiate Class	Call class methods	Fill Entity
Create		✓	✓	✓
Read	✓	✓		✓
Update	✓	✓	✓	✓
Delete	✓	✓	✓	

GET_ENTITY

- Identify primary key(s)
- Instantiate BO class
- Fill entity



GET_ENTITY V1

```
DATA: lt_key_tab    TYPE /iwbept/mgw_tech_pairs,  
      ls_key        LIKE LINE OF lt_key_tab,  
      lv_bp_id      TYPE snwd_partner_id,  
      lo_customer   TYPE REF TO zif_demo_customer,  
      lr_entity     TYPE REF TO data.  
  
lt_key_tab = io_tech_request_context->get_keys( ).  
READ TABLE lt_key_tab  
  WITH KEY name = 'BP_ID'  
  INTO ls_key.  
  
lv_bp_id = ls_key-value.  
  
lo_customer = zcl_demo_customer->get_using_bp_id( lv_bp_id ).  
  
GET REFERENCE OF er_entity INTO lr_entity.  
lo_customer->zif_gw_methods~map_to_entity( lr_entity ).
```



GET_ENTITY V1 – IDENTIFY PRIMARY KEY

```
DATA: lt_key_tab  TYPE /iwbep/t_mgw_tech_pairs,  
      ls_key      LIKE LINE OF lt_key_tab,  
      lv_bp_id    TYPE snwd_partner_id,  
      lo_customer TYPE REF TO zif_demo_customer,  
      lr_entity   TYPE REF TO data.  
  
lt_key_tab = io_tech_request_context->get_keys( ).  
READ TABLE lt_key_tab  
  WITH KEY name = 'BP_ID'  
  INTO ls_key.  
  
lv_bp_id = ls_key-value.  
  
lo_customer = zcl_demo_customer->get_using_bp_id( lv_bp_id ).  
  
GET REFERENCE OF er_entity INTO lr_entity.  
lo_customer->zif_gw_methods~map_to_entity( lr_entity ).
```



GET_ENTITY V1 – INstantiate BO CLASS

```
DATA: lt_key_tab  TYPE /iwbep/t_mgw_tech_pairs,  
      ls_key      LIKE LINE OF lt_key_tab,  
      lv_bp_id    TYPE snwd_partner_id,  
      lo_customer TYPE REF TO zif_demo_customer,  
      lr_entity   TYPE REF TO data.  
  
lt_key_tab = io_tech_request_context->get_keys( ).  
READ TABLE lt_key_tab  
  WITH KEY name = 'BP_ID'  
  INTO ls_key.  
  
lv_bp_id = ls_key-value.  
  
lo_customer = zcl_demo_customer->get_using_bp_id( lv_bp_id ).  
  
GET REFERENCE OF er_entity INTO lr_entity.  
lo_customer->zif_gw_methods~map_to_entity( lr_entity ).
```



GET_ENTITY V1 – FILL ENTITY

```
DATA: lt_key_tab  TYPE /iwbp/t_mgw_tech_pairs,  
      ls_key      LIKE LINE OF lt_key_tab,  
      lv_bp_id    TYPE snwd_partner_id,  
      lo_customer TYPE REF TO zif_demo_customer,  
      lr_entity   TYPE REF TO data.  
  
lt_key_tab = io_tech_request_context->get_keys( ).  
READ TABLE lt_key_tab  
  WITH KEY name = 'BP_ID'  
  INTO ls_key.  
  
lv_bp_id = ls_key-value.  
  
lo_customer = zcl_demo_customer->get_using_bp_id( lv_bp_id ).  
  
GET REFERENCE OF er_entity INTO lr_entity.  
lo_customer->zif_gw_methods~map_to_entity( lr_entity ).
```



GET_ENTITY V2

```
DATA: lt_key_tab    TYPE /iwbept/mgw_tech_pairs,  
      ls_key        LIKE LINE OF lt_key_tab,  
      lv_bp_id      TYPE snwd_partner_id,  
      lo_customer   TYPE REF TO zif_demo_customer,  
      lr_entity     TYPE REF TO data.  
  
lt_key_tab = io_tech_request_context->get_keys( ).  
READ TABLE lt_key_tab  
  WITH KEY name = 'BP_ID'  
  INTO ls_key.  
  
lv_bp_id = ls_key-value.  
  
lo_customer = zcl_demo_customer->get_using_bp_id( lv_bp_id ).  
  
GET REFERENCE OF er_entity INTO lr_entity.  
lo_customer->zif_gw_methods~map_to_entity( lr_entity ).
```



GET_ENTITY V2

```
lt_key_tab = io_tech_request_context->get_keys( ).  
READ TABLE lt_key_tab  
  with KEY name = 'BP_ID'  
  into ls_key.  
  
lv_bp_id = ls_key-value.  
  
lo_customer = zcl_demo_customer->get_using_bp_id( lv_bp_id ).  
  
GET REFERENCE OF er_entity INTO lr_entity.  
lo_customer->zif_gw_methods~map_to_entity( lr_entity ).
```

GET_ENTITY V2 – TABLE EXPRESSION

```
lt_key_tab = io_tech_request_context->get_keys( ).  
  
lv_bp_id = lt_key_tab[ name = 'BP_ID' ]-value.  
  
lo_customer = zcl_demo_customer->get_using_bp_id( lv_bp_id ).  
  
GET REFERENCE OF er_entity INTO lr_entity.  
lo_customer->zif_gw_methods~map_to_entity( lr_entity ).
```

GET_ENTITY V2 – CONVERSION OPERATOR

```
lt_key_tab = io_tech_request_context->get_keys( ).
```

```
lo_customer = zcl_demo_customer->get_using_bp_id(  
    CONV #( lt_key_tab[ name = 'BP_ID' ]-value ) ).
```

```
GET REFERENCE OF er_entity INTO lr_entity.  
lo_customer->zif_gw_methods~map_to_entity( lr_entity ).
```

GET_ENTITY V2 – REFERENCE OPERATOR

```
lt_key_tab = io_tech_request_context->get_keys( ).
```

```
lo_customer = zcl_demo_customer->get_using_bp_id(  
    CONV #( lt_key_tab[ name = 'BP_ID' ]-value ) ).
```

```
lo_customer->zif_gw_methods~map_to_entity(  
    REF #( er_entity ) ).
```

GET_ENTITY V2

```
lt_key_tab = io_tech_request_context->get_keys( ).
```

```
lo_customer = zcl_demo_customer->get_using_bp_id(  
    CONV #( lt_key_tab[ name = 'BP_ID' ]-value ) ).
```

```
*-----*  
* For update operations we would call class methods here *  
*-----*
```

```
lo_customer->zif_gw_methods~map_to_entity(  
    REF #( er_entity ) ).
```

GET_ENTITY V2 – METHOD CHAINING

```
lt_key_tab = io_tech_request_context->get_keys( ).  
  
zcl_demo_customer->get_using_bp_id(  
    CONV #( lt_key_tab[ name = 'BP_ID' ]-value )  
)->zif_gw_methods~map_to_entity( REF #( er_entity ) ) .
```

ZIF_GW_METHODS~GET_ENTITY

```
METHOD zif_gw_methods~get_entity.  
  
TRY.  
  CASE io_tech_request_context->get_source_entity_type_name( ).  
    WHEN 'SalesOrder'.  
      DATA(source_keys) = io_tech_request_context->get_source_keys( ).  
      zcl_demo_salesorder=>get_using_so_id(  
        CONV #( source_keys[ name = 'SO_ID' ]-value )  
      )->get_customer( )->zif_gw_methods~map_to_entity( REF #( er_entity ) ).  
    WHEN OTHERS.  
      DATA(keys) = io_tech_request_context->get_keys( ).  
      zcl_demo_customer=>get_using_bp_id(  
        CONV #( keys[ name = 'BP_ID' ]-value )  
      )->zif_gw_methods~map_to_entity( REF #( er_entity ) ).  
  ENDCASE.  
  
  CATCH zcx_demo_bo cx_sy_itab_line_not_found INTO DATA(exception).  
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception  
    EXPORTING  
      textid = /iwbep/cx_mgw_busi_exception->business_error  
      previous = exception  
      message = |{ exception->get_text( ) }|.  
  ENDTRY.  
  
ENDMETHOD.
```



ZIF_GW_METHODS~GET_ENTITY

```
METHOD zif_gw_methods~get_entity.  
  
TRY.  
  CASE io_tech_request_context->get_source_entity_type_name( ).  
    WHEN 'SalesOrder'.  
      DATA(source_keys) = io_tech_request_context->get_source_keys( ).  
      zcl_demo_salesorder=>get_using_so_id(  
        CONV #( source_keys[ name = 'SO_ID' ]-value )  
      )->get_customer( )->zif_gw_methods~map_to_entity( REF #( er_entity ) ).  
    WHEN OTHERS.  
      DATA(keys) = io_tech_request_context->get_keys( ).  
      zcl_demo_customer=>get_using_bp_id(  
        CONV #( keys[ name = 'BP_ID' ]-value )  
      )->zif_gw_methods~map_to_entity( REF #( er_entity ) ).  
  ENDCASE.  
  
  CATCH zcx_demo_bo cx_sy_itab_line_not_found INTO DATA(exception).  
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception  
    EXPORTING  
      textid = /iwbep/cx_mgw_busi_exception=>business_error  
      previous = exception  
      message = |{ exception->get_text( ) }|.  
ENDTRY.  
  
ENDMETHOD.
```

Exception
handling



ZIF_GW_METHODS~GET_ENTITY

```
METHOD zif_gw_methods~get_entity.  
  
TRY.  
  CASE io_tech_request_context->get_source_entity_type_name( ).  
    WHEN 'SalesOrder'.  
      DATA(source_keys) = io_tech_request_context->get_source_keys( ).  
      zcl_demo_salesorder=>get_using_so_id(  
        CONV #( source_keys[ name = 'SO_ID' ]-value )  
      )->get_customer( )->zif_gw_methods~map_to_entity( REF #( er_entity ) ).  
    WHEN OTHERS.  
      DATA(keys) = io_tech_request_context->get_keys( ).  
      zcl_demo_customer=>get_using_bp_id(  
        CONV #( keys[ name = 'BP_ID' ]-value )  
      )->zif_gw_methods~map_to_entity( REF #( er_entity ) ).  
  ENDCASE.  
  
  CATCH zcx_demo_bo cx_sy_itab_line_not_found INTO DATA(exception).  
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception  
    EXPORTING  
      textid = /iwbep/cx_mgw_busi_exception->business_error  
      previous = exception  
      message = |{ exception->get_text( ) }|.  
  ENDTRY.  
  
ENDMETHOD.
```

Use salesorder
key to get
customer

MAP_TO_ENTITY

Ty.	Parameter	Typing	Description
ENTITY	ZCX_DEMO_BO	TYPE REF TO DATA	

Method	ZIF_GW_METHODS~MAP_TO_ENTITY	Active
1	METHOD zif_gw_methods~map_to_entity.	
2		
3	ENDMETHOD.	

MAP_TO_ENTITY

Ty.	Parameter	Typing	Description
ENTITY	ZCX_DEMO_BO	TYPE REF TO DATA	

Method	ZIF_GW_METHODS~MAP_TO_ENTITY	Active
1	METHOD zif_gw_methods~map_to_entity. 2 call_all_getters(entity). 3 ENDMETHOD.	



CALL_ALL_GETTERS

```
DATA: struct_descr TYPE REF TO cl_abap_structdescr,  
      parameter      TYPE abap_parmbind,  
      parameters     TYPE abap_parmbind_tab.  
  
ASSIGN entity->* TO FIELD-SYMBOL(<entity>).  
  
TRY.  
  struct_descr ?= cl_abap_structdescr->describe_by_data_ref( entity ).  
  LOOP AT struct_descr->components REFERENCE INTO DATA(component).  
    CLEAR parameters.  
    TRY.  
      ASSIGN COMPONENT component->name OF STRUCTURE <entity> TO FIELD-SYMBOL(<comp>).  
      IF sy-subrc = 0.  
        parameter-name = component->name.  
        parameter-kind = cl_abap_objectdescr->returning.  
        GET REFERENCE OF <comp> INTO parameter-value.  
        INSERT parameter INTO TABLE parameters.  
        DATA(method_name) = |GET_{ component->name }|.  
        CALL METHOD me->(method_name)  
          PARAMETER-TABLE parameters.  
      ENDIF.  
      CATCH cx_sy_dyn_call_error.  
    ENDTRY.  
  ENDLOOP.  
  CATCH cx_root.  
ENDTRY.
```

For each entity component we attempt
to call matching getter functional method

e.g <entity>-prop1 = me->get_prop1().



GET_ENTITYSET

Just like GET_ENTITY in a loop...except...



ASUG 25

ODATA QUERY OPTIONS

SAP The Best-Run Businesses Run SAP Sitemap Global Search Search within this release

Help Portal Analytics Data Management Human Capital Management Supply Chain Management
Content and Collaboration Enterprise Management Product Lifecycle Mgmt Technology Platform
Customer Relationship Mgmt Financial Management Supplier Relationship Mgmt Additional Information

Technology Platform > SAP Gateway > 2.0 SAP NetWeaver Gateway

SAP NetWeaver Gateway Feedback Fullscreen Print Download

OData Query Options

The OData query options supported by SAP NetWeaver Gateway belong to either of these categories:

- They are automatically available in the framework, no further actions needed by development.
- They are available, but they require implementation efforts from development.

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes



ODATA QUERY OPTIONS

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes



KNOWN CONSTRAINTS

SAP 1574568 - SAP NetWeaver Gateway 2.0 - Known Constraints

Version 54 Validity: 28.07.2016 - active Language English (Master) Compare versions SSCR Download

Content: Summary Header Data Validity References

Symptom

This SAP Note outlines the constraints applicable to SAP NetWeaver Gateway 2.0.

Other Terms

OPU-GW

Reason and Prerequisites

These constraints apply in addition to the requirements and constraints described in the SAP NetWeaver Gateway 2.0 documentation on <http://help.sap.com/ngateway>. For more information about SAP NetWeaver Gateway 2.0, see SAP Note 1560585.

Solution

Read carefully and act accordingly.

This SAP Note contains several sections:

- General Constraints: These apply to SAP Netweaver Gateway as a whole.
- Constraints for the software component versions IW_FND 240 and GW_CORE 190 that are available with SAP NetWeaver Gateway 2.0 SP4 are described in SAP Note 1735987.
- General Constraints for Generic Channel: These apply to the Generic Channel specifically.
 - BOR/RFC generator constraints: These constraints apply to the BOR/RFC generator if it leverages the generic runtime.
 - Screen Scraping constraints: These constraints apply to the Screen Scraping generator if it leverages the generic runtime.
 - Content: These constraints apply to some of the content that leverages the generic runtime.

General Constraints

The following constraints apply to SAP Netweaver Gateway as a whole.

- odata for SAP (standard mode): The implementation of the OData protocol "OData for SAP", provided with SAP Netweaver Gateway 2.0 SP3, has the following constraints.

SAP Note 1574568

ZIF_GW_METHODS~GET_ENTITYSET

Goal is to identify primary key(s) as quickly as possible and use them to fill entities.



ZIF_GW_METHODS~GET_ENTITYSET

- Create anonymous data object
- \$orderby
- \$filter
- \$inlinecount
- \$top & \$skip
- \$count
- \$skiptoken
- Fill entities



CREATE ANONYMOUS DATA OBJECT

```
" Use RTTS/RTTC to create anonymous object like line of et_entityset
DATA: entity          TYPE REF TO data.
TRY.
  DATA(struct_descr) = get_struct_descr( et_entityset ).
  CREATE DATA entity TYPE HANDLE struct_descr.
  ASSIGN entity->* TO FIELD-SYMBOL(<entity>).
CATCH cx_sy_create_data_error INTO DATA(cx_sy_create_data_error).
  RAISE EXCEPTION TYPE /iwbep/cx_mgw_tech_exception
    EXPORTING
      textid   = /iwbep/cx_mgw_tech_exception=>internal_error
      previous = cx_sy_create_data_error.
ENDTRY.
```



ASUG 25

\$ORDERBY

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes



\$ORDERBY

```
" $orderby query options
DATA: orderby_clause TYPE string.
LOOP AT io_tech_request_context->get_orderby( ) REFERENCE INTO DATA(orderby).
  DATA(property) =
    io_model->get_property(
      iv_entity_name = io_tech_request_context->get_entity_type_name( )
      iv_property_name = orderby->property ).
  IF property-sortable = abap_true.
    CASE orderby->property. " This is where we add table aliases for the SQL join
      WHEN 'BP_ID' OR 'COMPANY_NAME'.
        orderby_clause = orderby_clause &&
          |, BP~{ orderby->property } { orderby->order CASE = UPPER }ENDING |.
      WHEN OTHERS.
        orderby_clause = orderby_clause &&
          |, AD~{ orderby->property } { orderby->order CASE = UPPER }ENDING |.
    ENDCASE.
  ELSE.
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception
      EXPORTING
        textid = /iwbep/cx_mgw_busi_exception=>business_error
        message = |Order property '{ property-external_name }' is not supported|.
  ENDIF.
ENDLOOP.
SHIFT orderby_clause LEFT DELETING LEADING ', '.
```



\$FILTER

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes



\$FILTER

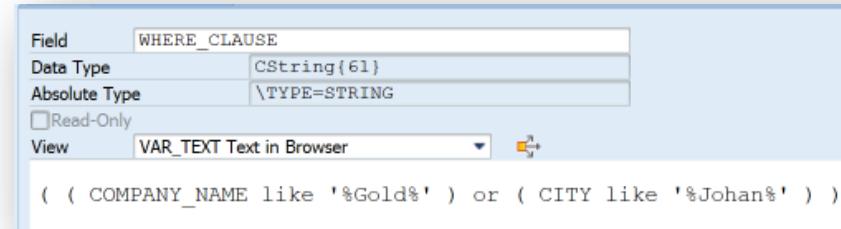
```
" $filterby query options
DATA: where_clause      TYPE string.
LOOP AT io_tech_request_context->get_filter( )->get_filter_select_options( )
      REFERENCE INTO DATA(filterby).
property =
  io_model->get_property(
    iv_entity_name = io_tech_request_context->get_entity_type_name( )
    iv_property_name = CONV #( filterby->property ) .
IF property-filterable = abap_true.
  CASE filterby->property.
    WHEN 'BP_ID'.
      DATA(bp_range) = filterby->select_options.
      where_clause = |{ where_clause } & BP~BP_ID IN @BP_RANGE|.
      ...
    ENDCASE.
  ELSE.
    RAISE EXCEPTION TYPE /iwbpex/cx_mgw_busi_exception
      EXPORTING
        textid      = /iwbpex/cx_mgw_busi_exception=>filter_not_supported
        filter_param = CONV #( property-external_name ).
  ENDIF.
ENDLOOP.
```



COMPLEX \$FILTER QUERIES

\$filter=substringof('Gold',Name) or substringof('Johan',City)

```
IF sy-subrc NE 0. " Catch complex $filter queries
  where_clause = io_tech_request_context->get_filter( )->get_filter_string( ).
ENDIF.
```



*Note where_clause may require modification e.g. Table aliases, etc.

\$INLINECOUNT

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes



\$INLINECOUNT

```
IF io_tech_request_context->has_inlinecount( ) = abap_true.  
DATA dbcount TYPE int4 .  
SELECT COUNT(*)  
  INTO @dbcount  
  FROM snwd_bpa AS bp  
    INNER JOIN snwd_ad AS ad  
      ON bp~address_guid = ad~node_key  
    WHERE (where_clause).  
es_response_context-inlinecount = dbcount.  
ENDIF.
```

\$INLINECOUNT

```
IF io_tech_request_context->has_inlinecount( ) = abap_true.  
DATA dbcount TYPE int4 .  
SELECT COUNT(*)  
  INTO @dbcount  
  FROM snwd_bp  
    INNER JOIN  
      ON bp~address  
 WHERE (where  
es_response_count  
ENDIF.
```

```
<feed xml:base="http://korora.yelcho.com.au:8000/sap/opu/odata/sap/ZDEMO_SRV/" xmlns="http://www.w3.org/2005/Atom"  
  <id>http://korora.yelcho.com.au:8000/sap/opu/odata/sap/ZDEMO_SRV/Customers</id>  
  <title type="text">Customers</title>  
  <updated>2017-04-06T07:31:33Z</updated>  
  <author>  
    <name/>  
  </author>  
  <link href="Customers" rel="self" title="Customers"/>  
  <m:count>47</m:count>  
  <entry>  
    <id>http://korora.yelcho.com.au:8000/sap/opu/odata/sap/ZDEMO_SRV/Customers('100000000')</id>  
    <title type="text">Customers('100000000')</title>  
    <updated>2017-04-06T07:31:33Z</updated>
```

\$TOP & \$SKIP

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes



\$TOP & \$SKIP

```
DATA: top  TYPE i,
      skip TYPE i.
top = io_tech_request_context->get_top( ).
skip = io_tech_request_context->get_skip( ).

" Get primary keys
SELECT bp~node_key
  FROM snwd_bpa AS bp
    INNER JOIN snwd_ad AS ad
      ON bp~address_guid = ad~node_key
        WHERE (where_clause)
          ORDER BY (orderby_clause)
            INTO CORRESPONDING FIELDS OF @<entity>.

CHECK sy-dbcnt > skip.
APPEND <entity> TO <entityset>.
IF top > 0 AND lines( <entityset> ) GE top.
  EXIT.
ENDIF.
ENDSELECT.
```

\$COUNT

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes



\$COUNT

```
CHECK io_tech_request_context->has_count( ) NE abap_true.
```



ASUG 25

\$SKIPTOKEN

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes



\$SKIPTOKEN

```
" $skiptoken
CONSTANTS: max_page_size TYPE i VALUE 50.
DATA: index_start TYPE i,
      index_end    TYPE i.

IF lines( <entityset> ) > max_page_size.
  index_start = io_tech_request_context->get_skiptoken( ).
  IF index_start = 0. index_start = 1. ENDIF.
  index_end = index_start + max_page_size - 1.
LOOP AT <entityset> REFERENCE INTO entity.
  IF index_start > 1.
    DELETE <entityset>.
    SUBTRACT 1 FROM index_start.
  ELSE.
    CHECK sy-tabix > max_page_size.
    DELETE <entityset>.
  ENDIF.
ENDLOOP.
IF lines( <entityset> ) = max_page_size.
  es_response_context-skiptoken = index_end + 1.
ENDIF.
ENDIF.
```

\$SKIPTOKEN

```
" $skiptoken
CONSTANTS: max_page_size TYPE i VALUE 50.
DATA: index_start TYPE i,
      index_end    TYPE i.

IF lines( <entityset> ) > max_page_size.
  index_start = io_tech_request_index_start.
  IF index_start = 0. index_start = 1.
  index_end = index_start + max_page_size.
  LOOP AT <entityset> REFERENCE.
    IF index_start > 1.
      DELETE <entityset>.
      SUBTRACT 1 FROM index_start.
    ELSE.
      CHECK sy-tabix > max_page_size.
      DELETE <entityset>.
    ENDIF.
  ENDLOOP.
  IF lines( <entityset> ) = max_page_size.
    es_response_context-skiptoken = index_end + 1.
  ENDIF.
ENDIF.
```

```
<entry>
  <id>http://mwsapdev2k8.mayfairs.local:8010/sap/opu/odata/sap/ZDEMO_SRV/Customers('9147')</id>
  <title type="text">Customers('9147')</title>
  <updated>2016-09-21T15:58:12Z</updated>
  <category term="ZDEMO_SRV.Customer" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/schemas/microsoft.com/ado/2007/08/dataservices/structure#Customer"></category>
  <link href="Customers('9147')" rel="self" title="Customer"/>
  <link href="Customers('9147')/SalesOrders" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/schemas/microsoft.com/ado/2007/08/dataservices/structure#Customer(SalesOrders)" type="application/json;odata=verbose"/>
  <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/structure#Customer">
      <d:CustomerId>9147</d:CustomerId>
      <d:Name>SOUTHWEST AUTOMOTIVE GROUP PTY LTD</d:Name>
      <d:Street>CNR MT LINDESAY HWY & CORP PLACE</d:Street>
      <d:City>HILLCREST</d:City>
      <d:RegionId>QLD</d:RegionId>
      <d:Region>Queensland</d:Region>
      <d:PostCode>4118</d:PostCode>
      <d:CountryId>AU</d:CountryId>
      <d:Country>Australia</d:Country>
      <d:Segmentation>A</d:Segmentation>
    </m:properties>
  </content>
</entry>
<link rel="next" href="Customers?$skiptoken=51%20"/>
</feed>
```

FILL ENTITIES

```
" Fill entities
LOOP AT <entityset> REFERENCE INTO entity.
  ASSIGN entity->* TO <entity>.
  ASSIGN COMPONENT 'NODE_KEY' OF STRUCTURE <entity> TO FIELD-SYMBOL(<node_key>).
  CHECK <node_key> IS ASSIGNED.
  TRY.
    zcl_demo_customer=>get( <node_key> )->zif_gw_methods~map_to_entity( entity ).
    CATCH zcx_demo_bo.
  ENDTRY.
ENDLOOP.
```



WHERE IS THE CODE?

https://grahamrobbo.github.io/teched16_example/

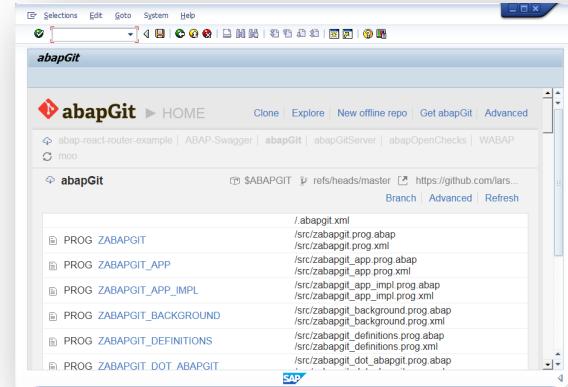
The screenshot shows a presentation slide with the following details:

- Title:** SAP TECHED | Las Vegas Sept. 19-23, 2016
- Section:** UX202 - Building Services for Gateway
- Navigation:** Slides, Sample code, abapGit
- Content:**
 - Welcome.
 - Thanks for your interest in my presentation UX202 - Building Services for Gateway. You can grab the ABAP code here and install it on your own system to look at it more closely and see it in action.
 - Update 3 - April 2017:** I was asked to re-deliver this session on a upcoming ASUG Webcast and also at the SAUG Brisbane Conference in May 2017.
 - I have therefore done some minor code refactoring - but most importantly I have updated the slides so that they now match the sample code.

ASUG | Webcasts



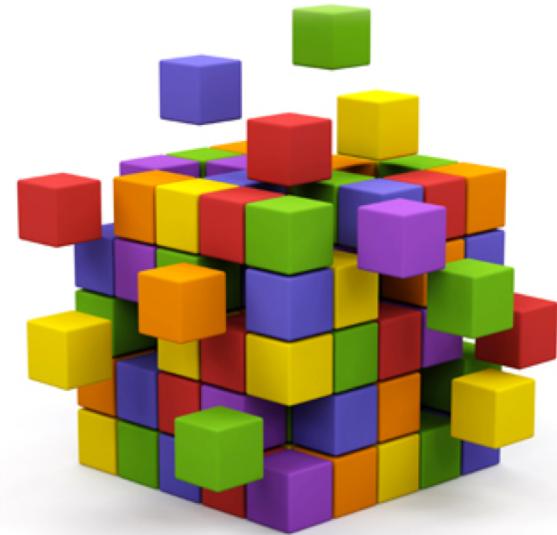
To install you will also need abapGit
<https://abapGit.org>



ASUG 25

AGENDA

- Some Warnings
- Service Design
- Code patterns for reusability
- Code patterns for extensibility



CODE PATTERNS FOR EXTENSIBILITY

- Mainline code built on central development server
- Mainline code deployed to customer system via abapGit 
- Need to be able to add customer-specific code without changing mainline code

BO CLASS SUBCLASS

Class/Interface	ZCL_DEMO_CUSTOMER_CUST	Implemented / Active					
Proper...	Interfaces	Friends	Attributes	Metho...	Events	Types	Aliases
Method		Level	Visibility	M...	Description		
<ZIF_DEMO_CUSTOMER>							
GET		Static	M.Public				
GET_KUNNR		Instance	.Public				
GET_NAME1		Instance	.Public				
GET_STRAS		Instance	.Public				
GET_ORT01		Instance	.Public				
GET_REGIO		Instance	.Public				
GET_REGION_TEXT		Instance	.Public				
GET_PSTLZ		Instance	.Public				
GET_LAND1		Instance	.Public				
GET_LAND_TEXT		Instance	.Public				

New methods
or redefine
existing
methods



BO CLASS INSTANTIATION

```
METHOD zif_demo_customer~get.
```

```
TRY.
```

```
    DATA(inst) = zif_demo_customer~instances[ node_key = node_key ].  
    CATCH cx_sy_itab_line_not_found.
```

```
    inst-node_key = node_key.
```

```
    DATA(class_name) = get_subclass( 'ZCL_DEMO_CUSTOMER' ).
```

```
    CREATE OBJECT inst-instance
```

```
        TYPE (class_name)
```

```
        EXPORTING
```

```
            node_key = inst-node_key.
```

```
    APPEND inst TO zif_demo_customer~instances.
```

```
ENDTRY.
```

```
instance ?= inst-instance.
```

```
ENDMETHOD.
```

BO CLASS INSTANTIATION

```
METHOD zif_demo_customer~get.
```

```
TRY.
```

```
    DATA(inst) = zif_demo_customer~instances[ node_key = node_key ].  
    CATCH cx_sy_itab_line_not_found.
```

```
    inst-node_key = node_key.
```

```
    DATA(class_name) = get_subclass( 'ZCL_DEMO_CUSTOMER' ).
```

```
    CREATE OBJECT inst-instance
```

```
        TYPE (class_name)
```

```
        EXPORTING
```

```
            node_key = inst-node_key.
```

```
    APPEND inst TO
```

```
ENDTRY.
```

```
instance ?= inst-in
```

```
ENDMETHOD.
```

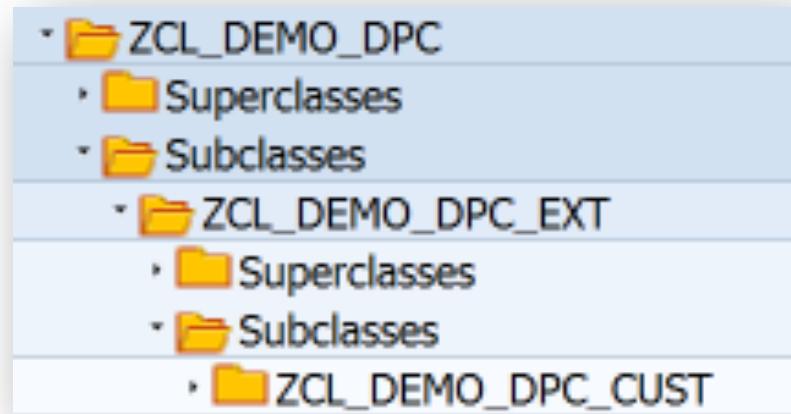
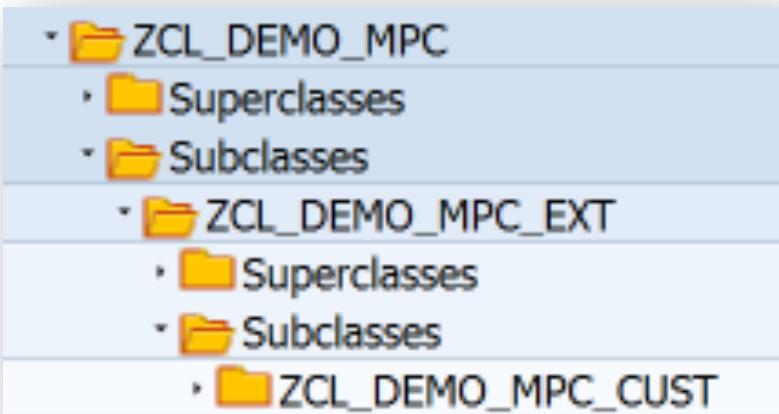
Class/Interface		ZCL_DEMO_CUSTOMER_CUST	Implemented / Active				
Property...	Interfaces	Friends	Attributes	Methods	Events	Types	Aliases
 		Friend	ZCL_DEMO_CUSTOMER				

DPC & MPC CLASSES

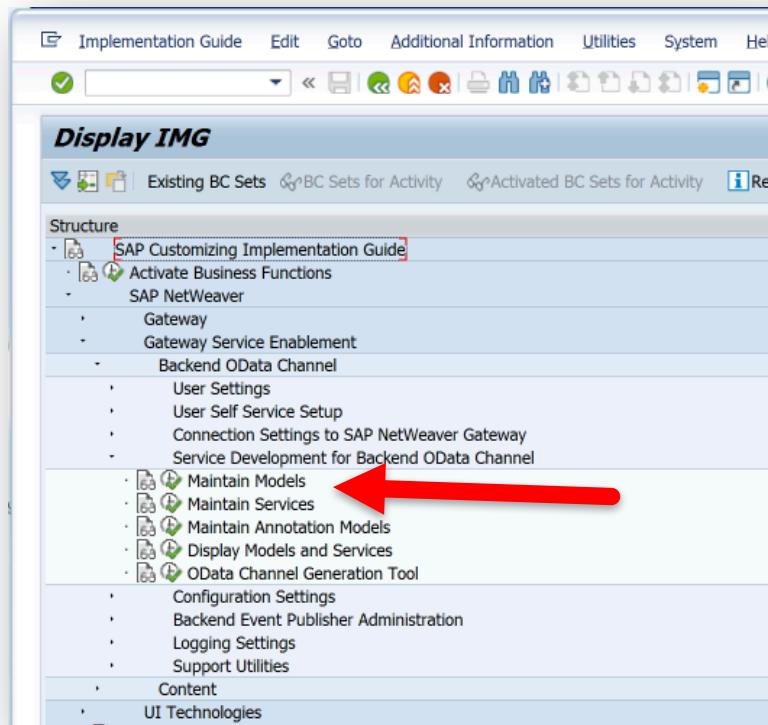
- ZCL_DEMO_MPC
 - Superclasses
 - Subclasses
 - ZCL_DEMO_MPC_EXT

- ZCL_DEMO_DPC
 - Superclasses
 - Subclasses
 - ZCL_DEMO_DPC_EXT

DPC & MPC SUBCLASSES

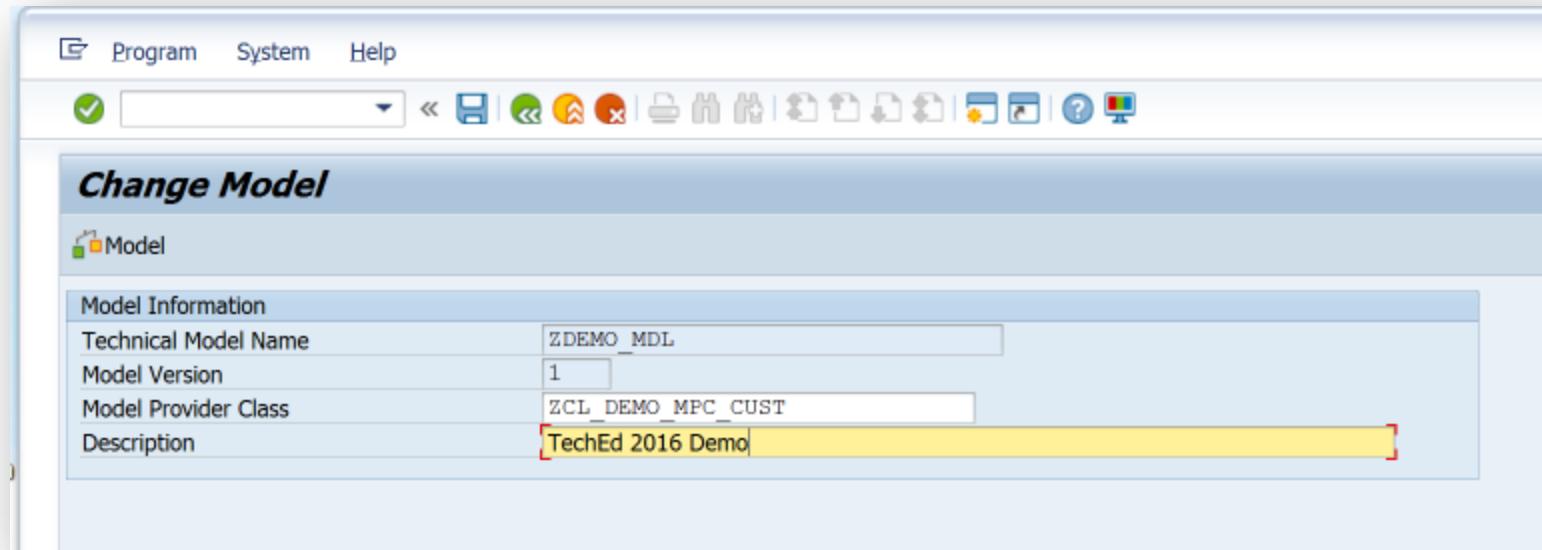


MAINTAIN GW MODELS & SERVICES



ASUG 25

/IWBEPR_DST_MODEL_BUILDER



ASUG 25

/IWBEPR_DST_SERVICE_BUILDER

Program Edit Goto System Help

Cleanup Cache Configuration

Change Service

Service Information	ZDEMO_SRV
Technical Service Name	1
Service Version	TechEd 2016 Demo
Description	ZDEMO_SRV
External Service Name	ZCL DEMO DPC CUST
Namespace	DEVELOPER
Data Provider Class	DEVELOPER
Created By	Changed By
Package	\$ROBBO

Extension for Service	
Technical Service Name	
Service Version	

Model Information	ZDEMO_MDL
Technical Model Name	1
Model Version	TechEd 2016 Demo
Description	ZCL DEMO MPC CUST
Model Provider Class	DEVELOPER
Created By	Changed By
Package	\$ROBBO

Create Model Unassign Model Assign Model Annotation Model



EXTEND GW SERVICE

- Create or Enhance DDIC structure
- Enhance model definition in MPC
- Add code to fill new entities/properties



ENHANCE DDIC STRUCTURE

Structure	ZDEMO_CUSTOMER	Active	
Short Description	Customer		
Attribu... Components Entry help/check Currency/quantity fields			
Predefined Type		1 / 11	
Component	Typing Method	Component Type	Data Type Length Decim... Short Description
KUNNR	1 Types	▼ KUNNR	CHAR 10 0 Customer Number
NAME1	1 Types	▼ NAME1 GP	CHAR 35 0 Name 1
STRAS	1 Types	▼ STRAS GP	CHAR 35 0 House number and street
ORT01	1 Types	▼ ORT01 GP	CHAR 35 0 City
REGIO	1 Types	▼ REGIO	CHAR 3 0 Region (State, Province, County)
REGION_TEXT	1 Types	▼ ZDEMO REGION_TEXT	CHAR 20 0 Region text
PSTLZ	1 Types	▼ PSTLZ	CHAR 10 0 Postal Code
LAND1	1 Types	▼ LAND1 GP	CHAR 3 0 Country Key
LAND_TEXT	1 Types	▼ ZDEMO COUNTRY_TEXT	CHAR 50 0 Country Text
.APPEND	1 Types	▼ ZDEMO CUSTOMER APPEND	EDO 0 Customer append
SEGMENTATION	1 Types	▼ ZDEMO SEGMENTATION	CHAR 1 0 Customer segmentation value



ENHANCE MODEL

```
| METHOD extend_customer.  
  
DATA:  
  lo_entity_type  TYPE REF TO /iwbp/if_mgw_odata_entity_typ, "#EC NEEDED  
  lo_property     TYPE REF TO /iwbp/if_mgw_odata_property. "#EC NEEDED  
  
*****  
*   ENTITY - Customer  
*****  
  
  lo_entity_type = model->get_entity_type( 'Customer' ). "#EC NOTEXT  
  
*****  
*Properties  
*****  
  
  lo_property = lo_entity_type->create_property(  
    iv_property_name = 'Segmentation'  
    iv_abap_fieldname = 'SEGMENTATION' ). "#EC NOTEXT  
  lo_property->set_type_edm_string( ).  
  lo_property->set_maxlength( iv_max_length = 1 ). "#EC NOTEXT  
  lo_property->set_creatable( abap_false ).  
  lo_property->set_updatable( abap_false ).  
  lo_property->set_sortable( abap_false ).  
  lo_property->set_nullable( abap_false ).  
  lo_property->set_filterable( abap_false ).  
  
ENDMETHOD.
```

FILL NEW PROPERTY

Class Builder Class ZCL_DEMO_CUSTOMER_CUST Display

MIME Repository Repository Browser Repository Information System

Package: \$TECHED16

Object Name: ZCL_DEMO_CUSTOMER_CUST

Method: GET_SEGMENTATION

Ty.	Parameter	Typing	Descript.
value(SEGMENTATION)		TYPE ZDEMO_SEGMENTATION	
ZCX_DEMO_BO			

```
1 METHOD get_segmentation.  
2 *-----  
3 * Sample extension of BO class with new method  
4 *-----  
5 DATA(name) = zif_demo_customer~get_company_name().  
6 segmentation = name(1).  
7 ENDMETHOD.
```



SUMMARY

- Service Design
 - Design for consumers
- Code patterns for reusability
 - The first & second rule
- Code patterns for extensibility



WHERE IS THE CODE?

https://grahamrobbo.github.io/teched16_example/

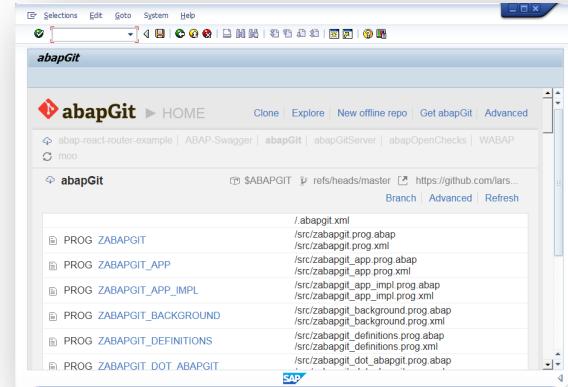
The screenshot shows a presentation slide with the following details:

- Title:** SAP TECHED | Las Vegas Sept. 19-23, 2016
- Section:** UX202 - Building Services for Gateway
- Navigation:** Slides, Sample code, abapGit
- Content:**
 - Welcome.
 - Thanks for your interest in my presentation UX202 - Building Services for Gateway. You can grab the ABAP code here and install it on your own system to look at it more closely and see it in action.
 - Update 3 - April 2017:** I was asked to re-deliver this session on a upcoming ASUG Webcast and also at the SAUG Brisbane Conference in May 2017.
 - I have therefore done some minor code refactoring - but most importantly I have updated the slides so that they now match the sample code.

ASUG | Webcasts



To install you will also need abapGit
<https://abapGit.org>



ASUG 25



 Yelcho
Systems Consulting

Graham Robinson
Principal Consultant

Mobile +61 412 402 441

Email graham@yelcho.com.au



@grahamrobbo

Photo by Frank Hurley – used with permission

THANK YOU

Thank you for your time

asUG | Webcasts
