Technical Articles

pavan karaiya

April 26, 2019 7 minute read

Dynamic table data read and odata binding

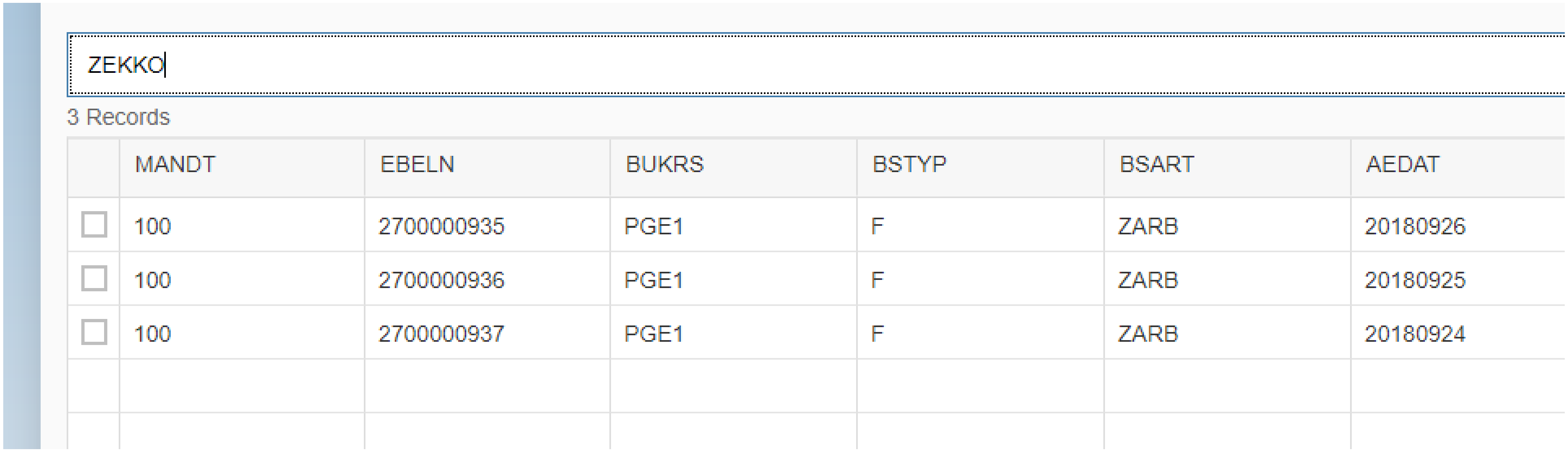
Follow RSS feed Like

1 Like 2,276 Views 5 Comments

Sometimes in our requirement we need to read data dynamically based on some condition, like columns, table/ internal table name may changes as per user input. During one such requirement we need replicate se16 like transaction for Fiori application. Here we need to show table and table data based on the table name given in input field.

In this blog I would explain the steps to achieve the same. We will go through the odata creation, logic to collect table config and data and then consume odata service on SAP ui5 application.

Output would be like as below. In input field provide any table name and on submitting, application should load table data.



ZEKKO

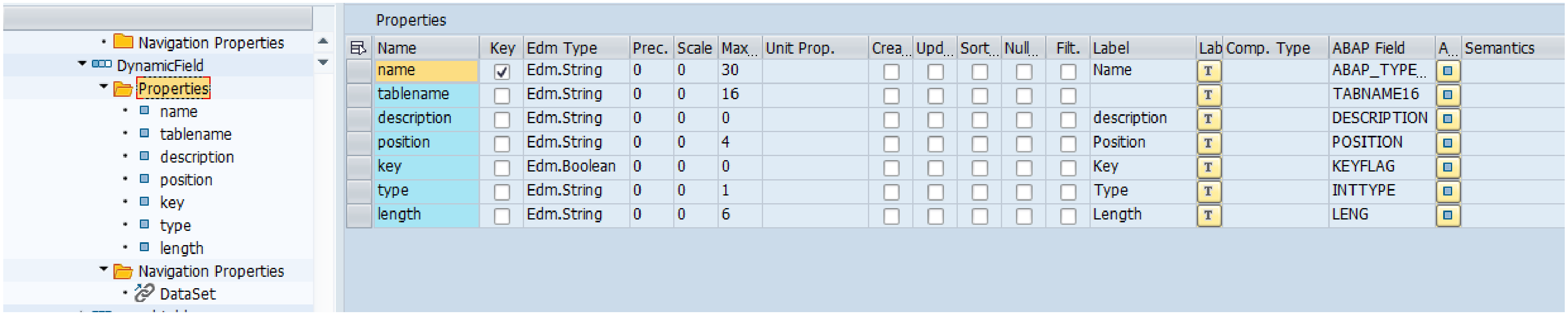
3 Records

	MANDT	EBELN	BUKRS	BSTYP	BSART	AEDAT
□	100	2700000935	PGE1	F	ZARB	20180926
□	100	2700000936	PGE1	F	ZARB	20180925
□	100	2700000937	PGE1	F	ZARB	20180924

First of all create 2 entities for collecting table field information and data like below.

Entity sets

Primary entity: DynamicField to contain configuration information of table

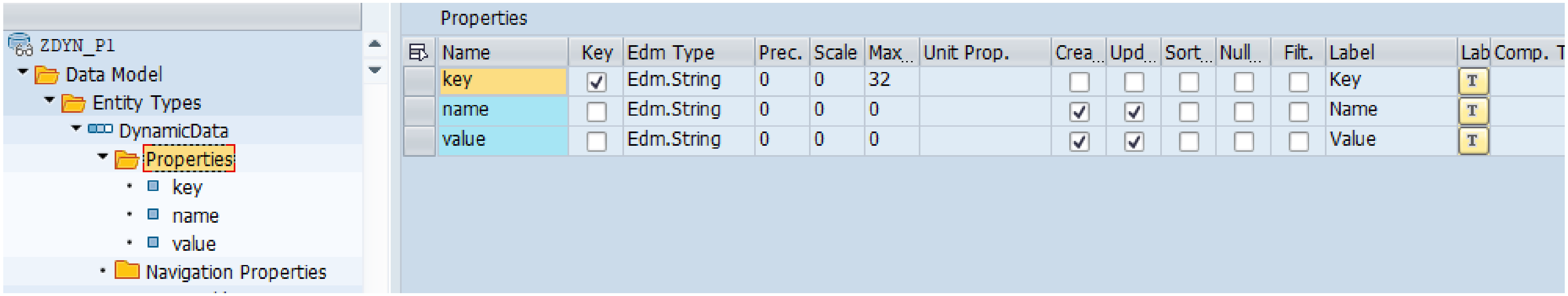


- Navigation Properties
- DynamicField
  - Properties
    - name
    - tablename
    - description
    - position
    - key
    - type
    - length
  - Navigation Properties
    - DataSet

Properties

Name	Key	Edm Type	Prec.	Scale	Max...	Unit Prop.	Crea...	Upd...	Sort...	Null...	Filt.	Label	Lab	Comp. Type	ABAP Field	A...	Semantics
name	☑	Edm.String	0	0	30		□	□	□	□	□	Name	T		ABAP_TYPE...	▣	
tablename	□	Edm.String	0	0	16		□	□	□	□	□		T		TABNAME16	▣	
description	□	Edm.String	0	0	0		□	□	□	□	□	description	T		DESCRIPTION	▣	
position	□	Edm.String	0	0	4		□	□	□	□	□	Position	T		POSITION	▣	
key	□	Edm.Boolean	0	0	0		□	□	□	□	□	Key	T		KEYFLAG	▣	
type	□	Edm.String	0	0	1		□	□	□	□	□	Type	T		INTTYPE	▣	
length	□	Edm.String	0	0	6		□	□	□	□	□	Length	T		LENG	▣	

And second entity: DynamicData to contain data for the table



Properties

Name	Key	Edm Type	Prec.	Scale	Max…	Unit Prop.	Crea…	Upd…	Sort…	Null…	Filt.	Label	Lab	Comp. T
key	☑	Edm.String	0	0	32		☐	☐	☐	☐	☐	Key	T	
name	☐	Edm.String	0	0	0		☑	☑	☐	☐	☐	Name	T	
value	☐	Edm.String	0	0	0		☑	☑	☐	☐	☐	Value	T	

Now create a method in DPC class to identify the table name, extract field information and data for table and save this data in entityset.

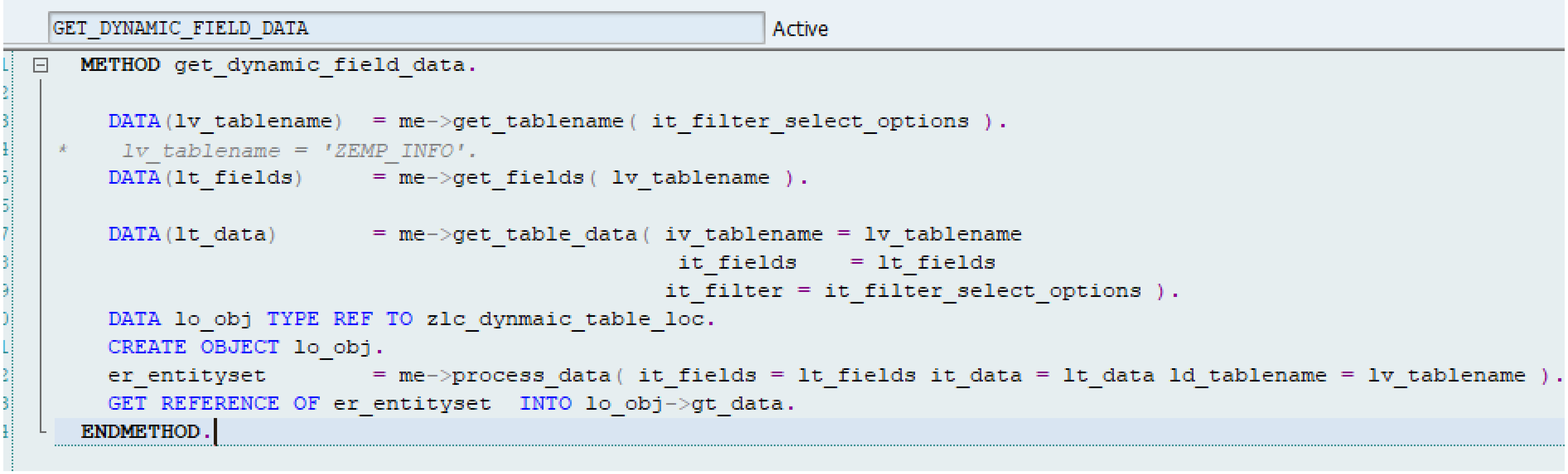
Here we have written logic to

Fetch table name coming from UI screen

Fetch Table fields

Fetch Table data

And then combine whole data in single entity



```
METHOD get_dynamic_field_data.

  DATA(lv_tablename)    = me->get_tablename( it_filter_select_options ).
*   lv_tablename = 'ZEMP_INFO'.
  DATA(lt_fields)       = me->get_fields( lv_tablename ).

  DATA(lt_data)         = me->get_table_data( iv_tablename = lv_tablename
                                               it_fields    = lt_fields
                                              it_filter = it_filter_select_options ).
  DATA lo_obj TYPE REF TO zlc_dynmaic_table_loc.
  CREATE OBJECT lo_obj.
  er_entityset          = me->process_data( it_fields = lt_fields it_data = lt_data ld_tablename = lv_tablename ).
  GET REFERENCE OF er_entityset  INTO lo_obj->gt_data.
ENDMETHOD.
```

Logic to fetch table fields and description

```
DATA: lo_table TYPE REF TO cl_abap_tabledescr,
lo_struc TYPE REF TO cl_abap_structdescr,
lo_eleme TYPE REF TO cl_abap_elemdescr,
lr_dref   TYPE REF TO data,
ls_fileds TYPE LINE OF zcl_zdyn_p1_mpc=>tt_dynamicfield,
lt_fields TYPE zcl_zdyn_p1_mpc=>tt_dynamicfield.
CREATE DATA lr_dref TYPE STANDARD TABLE OF (iv_tablename).ASSIGN lr_dref->* TO FIELD-SYMBOL(<fs_itab>).
lo_table ?= cl_abap_typedescr=>describe_by_data( <fs_itab> ).
lo_struc ?= lo_table->get_table_line_type( ).
DATA(lt_ddic) = lo_struc->get_ddic_object( ).
LOOP AT lt_ddic ASSIGNING FIELD-SYMBOL(<fs_ddic>).
lo_eleme      ?= cl_abap_elemdescr=>describe_by_name( <fs_ddic>-rollname ).
DATA(ls_dfies)  = lo_eleme->get_ddic_field( ).
ls_fileds-abap_typekind   =  <fs_ddic>-fieldname.
ls_fileds-description     = ls_dfies-scrtext_l.
```

```
ls_fileds-position        = ls_dfies-position.
ls_fileds-keyflag         =  ls_dfies-keyflag.
ls_fileds-inttype         = ls_dfies-inttype.
ls_fileds-leng            = ls_dfies-leng.
APPEND ls_fileds TO lt_fields.ENDLOOP.
CALL METHOD me-/iwbep/if_mgw_conv_srv_runtime~copy_data_to_ref
EXPORTING
is_data = lt_fields
CHANGING cr_data = rt_fields.
```

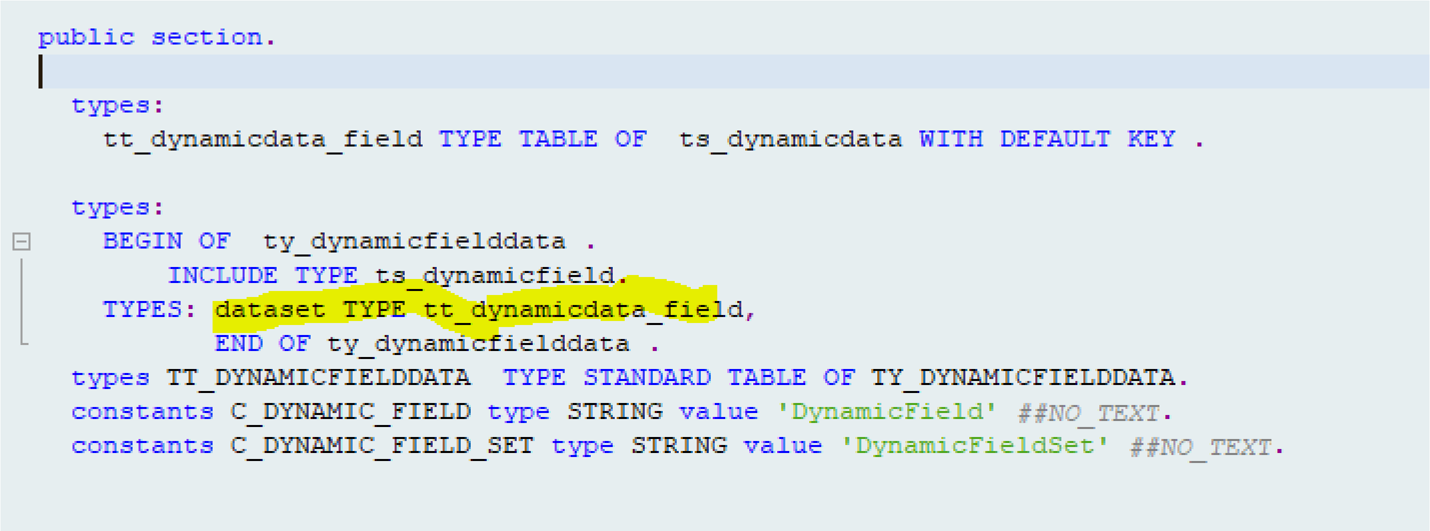
Fetch table data

```
METHOD get_table_data.
DATA : ld_data  TYPE REF TO data,
ld_where TYPE edpline,
lt_where TYPE TABLE OF edpline.
FIELD-SYMBOLS: <ft_data>   TYPE ANY TABLE,
<ft_data_w> TYPE ANY TABLE,
<ft_fields> TYPE zcl_zdyn_p1_mpc=>tt_dynamicfield.
ASSIGN it_fields->* TO <ft_fields>.
DATA(ld_search) = me->get_search( it_filter ).
CREATE DATA ld_data TYPE  TABLE OF (iv_tablename).
ASSIGN ld_data->* TO <ft_data>.
ASSIGN ld_data->* TO <ft_data_w>.
IF ld_search IS INITIAL.
SELECT * FROM (iv_tablename) INTO TABLE <ft_data>.
ELSE.
LOOP AT <ft_fields> ASSIGNING FIELD-SYMBOL(<fs_fields>).
IF <fs_fields>-inttype EQ ‘P’ OR <fs_fields>-inttype EQ ‘b’ OR <fs_fields>-inttype EQ ‘I’.
*             ld_where = <fs_fields>-abap_typekind  && ‘ = ‘ && ld_search  .
ELSE.
ld_where = <fs_fields>-abap_typekind  && ‘ =’ && | ‘| && ld_search && |’| .
APPEND ld_where TO lt_where.
SELECT * FROM (iv_tablename) INTO TABLE  <ft_data_w>  WHERE (ld_where).
```

```
ENDIF.
CLEAR ld_where.
refresh <ft_data_w> .
ENDLOOP.
ENDIF.
CALL METHOD me->/iwbep/if_mgw_conv_srv_runtime~copy_data_to_ref
EXPORTING
is_data = <ft_data>
CHANGING
cr_data = rt_data.
ENDMETHOD.
```

Now we will combine whole data into single entity that is lt_field_data

This field is complex structure which will combine our earlier declared entity types. So we will expose data in nested structure to UI5 application



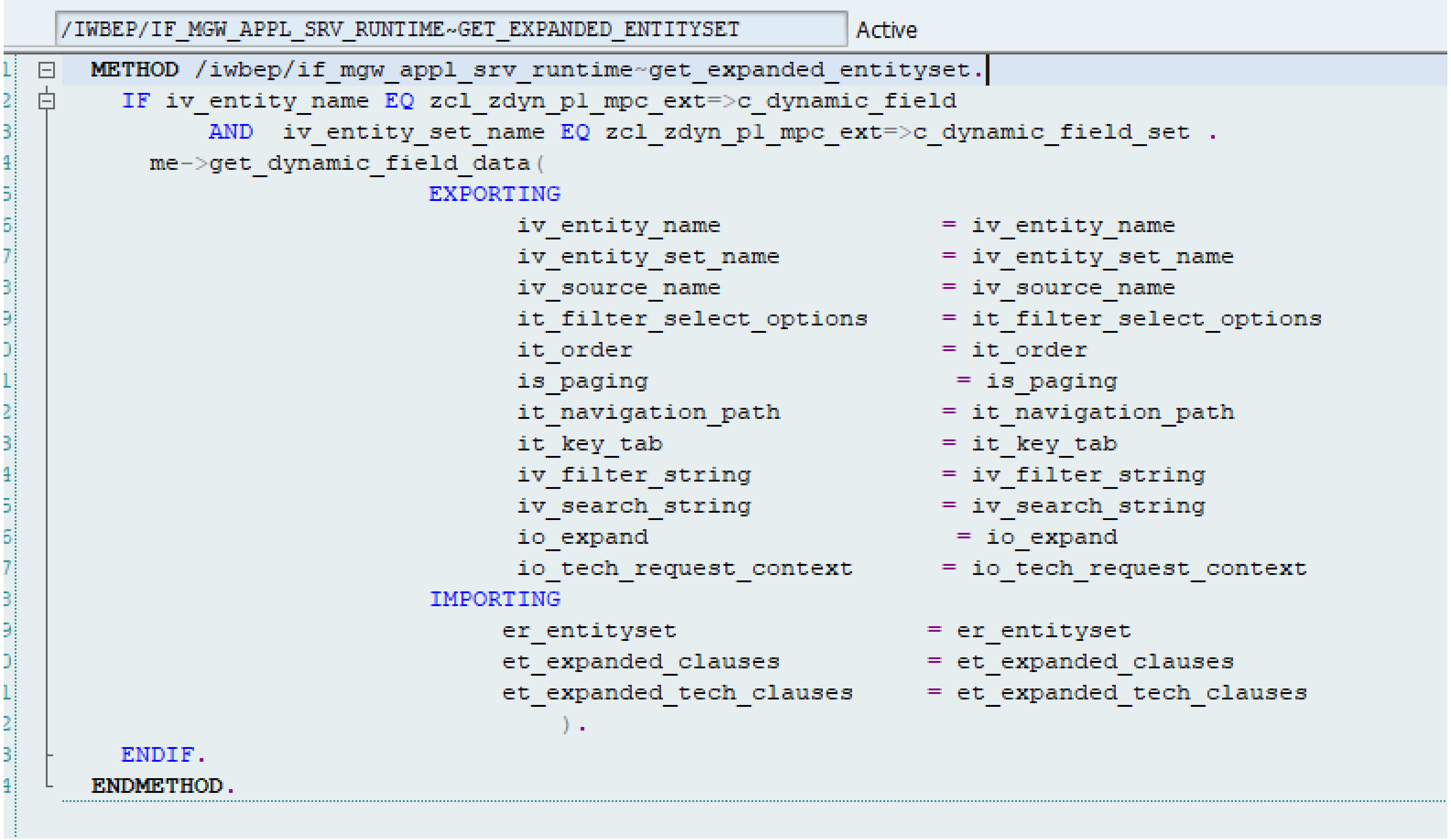
```
public section.

  types:
    tt_dynamicdata_field TYPE TABLE OF  ts_dynamicdata WITH DEFAULT KEY .

  types:
    BEGIN OF  ty_dynamicfielddata .
        INCLUDE TYPE ts_dynamicfield.
    TYPES: dataset TYPE tt_dynamicdata_field,
           END OF ty_dynamicfielddata .
  types TT_DYNAMICFIELDDATA  TYPE STANDARD TABLE OF TY_DYNAMICFIELDDATA.
  constants C_DYNAMIC_FIELD type STRING value 'DynamicField' ##NO_TEXT.
  constants C_DYNAMIC_FIELD_SET type STRING value 'DynamicFieldSet' ##NO_TEXT.
```

```
METHOD process_data.
DATA : lt_field_data TYPE STANDARD TABLE OF zcl_zdyn_p1_mpc_ext=>ty_dynamicfielddata,
ls_field_data TYPE  zcl_zdyn_p1_mpc_ext=>ty_dynamicfielddata,
ls_data       TYPE LINE OF zcl_zdyn_p1_mpc_ext=>tt_dynamicdata_field,
lo_sysuuid    TYPE REF TO cl_system_uuid,
ld_counter    type char2.
FIELD-SYMBOLS:<ft_fields> TYPE table,
<ft_data>   TYPE table.
ASSIGN it_fields->* TO <ft_fields>.
LOOP AT <ft_fields> ASSIGNING FIELD-SYMBOL(<fs_fields>).
me->move_corresponding(
EXPORTING
id_source_data = <fs_fields>
IMPORTING
ed_target_data = ls_field_data
) .
ld_counter = ld_counter + 1.
ls_field_data-TABNAME16 = ld_tablename.
ls_field_data-position = ld_counter.
APPEND ls_field_data TO lt_field_data.
ENDLOOP.
ASSIGN  it_data->* TO <ft_data>.
CREATE OBJECT lo_sysuuid .
ld_counter = 0.
LOOP AT <ft_data> ASSIGNING FIELD-SYMBOL(<fs_data>).
TRY.
DATA(lv_uuid) = lo_sysuuid->if_system_uuid~create_uuid_c32( ).
CATCH cx_uuid_error .
ENDTRY.
LOOP AT lt_field_data ASSIGNING FIELD-SYMBOL(<fs_field_data>).
ASSIGN COMPONENT <fs_field_data>-abap_typekind  OF STRUCTURE <fs_data> TO FIELD-SYMBOL(<fv_data>).
ls_data-key       = lv_uuid.
ls_data-fieldname = <fs_field_data>-abap_typekind.
```

```
ls_data-value     = <fv_data>.
append ls_data to <fs_field_data>-dataset.
ENDLOOP  .
ENDLOOP.
CALL METHOD me->/iwbep/if_mgw_conv_srv_runtime~copy_data_to_ref
EXPORTING
is_data = lt_field_data
CHANGING
cr_data = tr_data.
ENDMETHOD.
```

At last when you have finished with data processing call expanded entity set and call the above primary method i.e.

implement get_expanded_entityset and call get_dynamic_field_data

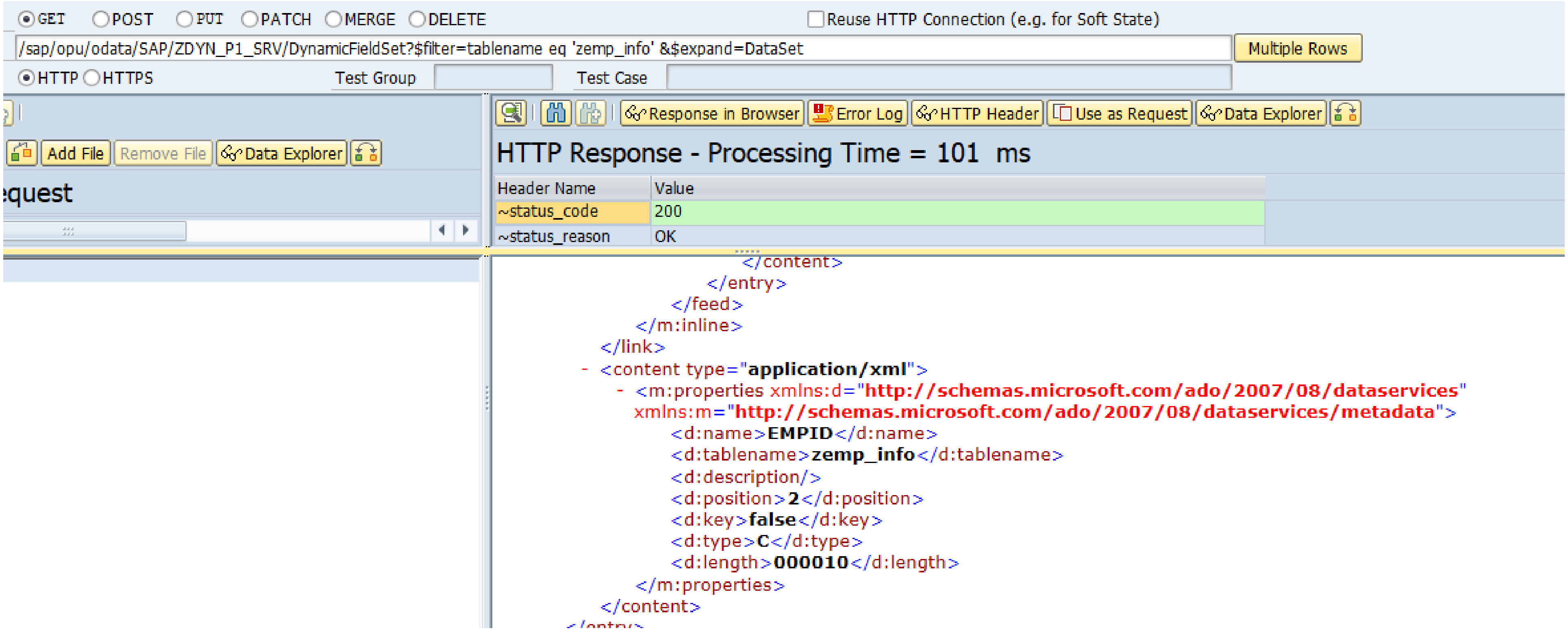


/IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_EXPANDED_ENTITYSET Active

```
METHOD /iwbep/if_mgw_appl_srv_runtime~get_expanded_entityset.
  IF iv_entity_name EQ zcl_zdyn_pl_mpc_ext=>c_dynamic_field
        AND  iv_entity_set_name EQ zcl_zdyn_pl_mpc_ext=>c_dynamic_field_set .
    me->get_dynamic_field_data(
                       EXPORTING
                             iv_entity_name              = iv_entity_name
                             iv_entity_set_name          = iv_entity_set_name
                             iv_source_name              = iv_source_name
                             it_filter_select_options    = it_filter_select_options
                             it_order                    = it_order
                             is_paging                    = is_paging
                             it_navigation_path          = it_navigation_path
                             it_key_tab                  = it_key_tab
                             iv_filter_string            = iv_filter_string
                             iv_search_string            = iv_search_string
                             io_expand                    = io_expand
                             io_tech_request_context     = io_tech_request_context
                       IMPORTING
                            er_entityset                = er_entityset
                            et_expanded_clauses         = et_expanded_clauses
                            et_expanded_tech_clauses    = et_expanded_tech_clauses
                                ).
  ENDIF.
ENDMETHOD.
```

Now if you register and execute the service:

You will get the table structure and data returned in nested structure as below



● GET ○ POST ○ PUT ○ PATCH ○ MERGE ○ DELETE

□ Reuse HTTP Connection (e.g. for Soft State)

/sap/opu/odata/SAP/ZDYN_P1_SRV/DynamicFieldSet?$filter=tablename eq 'zemp_info' &$expand=DataSet

● HTTP ○ HTTPS

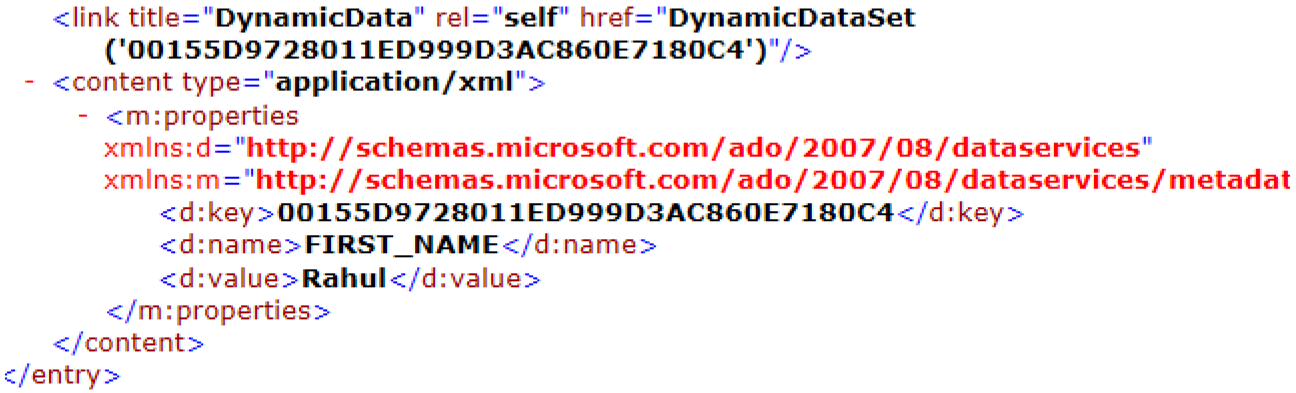
Test Group

Test Case

HTTP Response - Processing Time = 101 ms

Header Name	Value
~status_code	200
~status_reason	OK

```
                </content>
              </entry>
            </feed>
          </m:inline>
        </link>
      - <content type="application/xml">
        - <m:properties xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
            xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
            <d:name>EMPID</d:name>
            <d:tablename>zemp_info</d:tablename>
            <d:description/>
            <d:position>2</d:position>
            <d:key>false</d:key>
            <d:type>C</d:type>
            <d:length>000010</d:length>
          </m:properties>
        </content>
      </entry>
```



```
    <link title="DynamicData" rel="self" href="DynamicDataSet
       ('00155D9728011ED999D3AC860E7180C4')"/>
  - <content type="application/xml">
    - <m:properties
       xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
       xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadat
          <d:key>00155D9728011ED999D3AC860E7180C4</d:key>
          <d:name>FIRST_NAME</d:name>
          <d:value>Rahul</d:value>
       </m:properties>
    </content>
</entry>
```

UI5 application

Create a sapui5 project and connect your odata service to the application

Create tableview.xml as below, create table with basic attributes only

```
<mvc:View xmlns:html=”http://www.w3.org/1999/xhtml” xmlns:mvc=”sap.ui.core.mvc” xmlns:u=”sap.ui.unified” xmlns:c=”sap.ui.core”

xmlns:m=”sap.m” xmlns:t=”sap.ui.table” controllerName=”dynDynamicTab.controller.DynTabView” displayBlock=”true”>

<m:Page id = “page” showHeader=”true” enableScrolling=”false” class=”sapUiContentPadding” title=”Data Browser”>

<m:Input placeholder=”Enter table name” submit = “onSubmit” value=”zemp_info” id=”tablename” showValueHelp=”true” valueHelpRequest=”onSearch”></m:Input>

<m:Button id = “search” text=”Load” press=’onLoad’ visible = “false” ></m:Button>

<m:Label id = “count” text=”” ></m:Label>

<m:content>

<t:Table id=”tableid”  visibleRowCount=”20″ enableSelectAll=”false” threshold=”15″ enableBusyIndicator=”true” ariaLabelledBy=”title”></t:Table>

</m:content>

</m:Page>

</mvc:View>
```

Inside **tableview.controller**

Load odatamodel

```
var oModelVariantO1 = new sap.ui.model.odata.ODataModel(“/sap/opu/odata/SAP/ZDYN_P1_SRV/”, true);

var oJsonModel1 = new sap.ui.model.json.JSONModel();

sap.ui.core.BusyIndicator.show(0);
```

On click of button or submit event of input field read data from backend as below

fetch table name frominput control and concatenate with url for reading

```
var url = “DynamicFieldSet?$filter=tablename eq ‘” + this.byId(“tablename”).getValue() + “‘ &$expand=DataSet”;


oModelVariantO1.read(url, {


success: function(oData, response) {

//          console.log(oData);
sap.ui.core.BusyIndicator.hide();

that.odata = oData;

that.writedata(that.odata);

},

error: function(err) {

sap.ui.core.BusyIndicator.hide();

}

});
```

Under function writedata we have received all the results from back end

Get rows and columns separately from results array. Bind rows and columns and then you can see the output in your screen

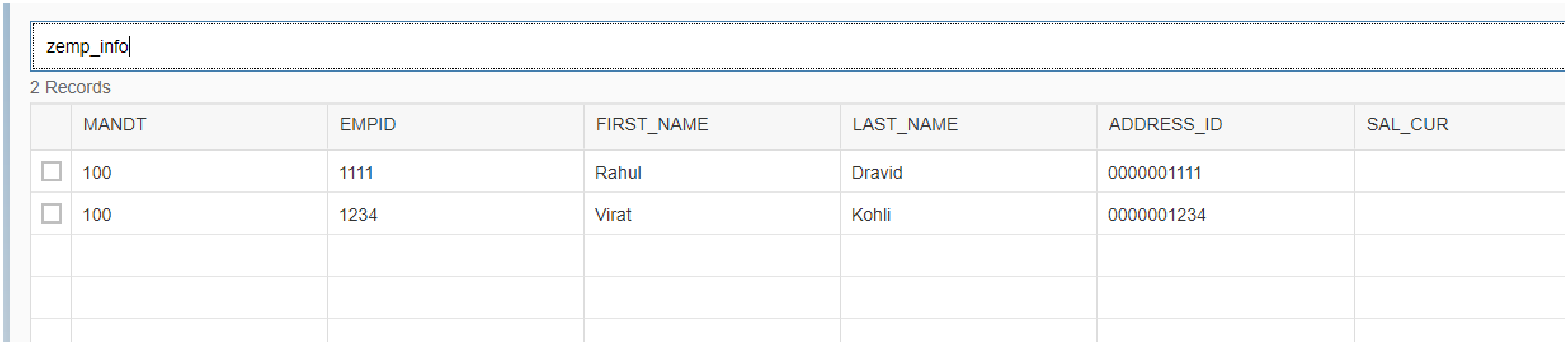
```
writedata: function(results) {

var row_array = [];

var array = results.results;

var no_of_rows = results.results[0].DataSet.results;

this.byId(“count”).setText(no_of_rows.length + ‘ Records’);

no_of_rows.forEach(function(entry1, j) {
```

```
var obj = {};
var j_value = j;
array.forEach(function(entry, i) {

var field = results.results[i].DataSet.results[j_value];
var fieldvalue = field.value;
//var fieldname = results.results[i].description;
var fieldname = field.name;

Object.defineProperty(obj, fieldname, {
value: fieldvalue //set data1 and data 2 accordingly
});
});
row_array.push(obj);
});
```

```
var jsonmodel = new sap.ui.model.json.JSONModel();
jsonmodel.setData({
rows: row_array,
columns: results.results
});
var oTable = this.getView().byId(“tableid”);
oTable.setModel(jsonmodel);
oTable.bindColumns(“/columns”, function(sId, oContext) {
var columnName = oContext.getObject().name;
//          var columntemp = oContext.getObject().name;
return new sap.ui.table.Column({
label: columnName,

template: columnName,
sortProperty: columnName,
filterProperty: columnName
});

});

oTable.bindRows(“/rows”);

});

},
```

When we get the odata from back end system, we identify the row and column for the output table and bind the rows and column to out put table.

Output Screen



zemp_info

2 Records

	MANDT	EMPID	FIRST_NAME	LAST_NAME	ADDRESS_ID	SAL_CUR
□	100	1111	Rahul	Dravid	0000001111	
□	100	1234	Virat	Kohli	0000001234	

Here is in this example we have bound the row and columns with fieldname properties, additionally we can bind field description, position as well to the output table. Please let me know if your views on this approach and suggestion. Thanks for reading the blog. Cheers.

Alert Moderator

Assigned tags

NW ABAP Gateway (OData) | SAP Fiori | SAPUI5 |

Related Blog Posts

Create Dynamic Entity Model in SAP Gateway

By **Pranay Patel** , Feb 11, 2018

Create ODATA service and implement CRUD methods using ABAP 7.4
By **Former Member** , Mar 08, 2017

'Field Control' Feature in SAP OData Project
By **Amit Singh** , Apr 22, 2018

Related Questions

Odata : Creating dynamic Table columns based on dynamic structure
By **Former Member** , Mar 22, 2017

Single odata service for multiple tables
By **Soumya Nandi** , Jul 01, 2019

Data Binding - Odata Navigation is empty
By **Jochen Harder** , Feb 25, 2019

5 Comments

You must be Logged on to comment or reply to a post.

Alexander K

April 29, 2019 at 2:44 am

Thanks,

pavan karaiya
Very usefull blog!!!

Like (0)

pavan karaiya | Post author

April 29, 2019 at 2:38 pm

Thanks Alexander K

Like (0)

Emily Rascons

May 1, 2019 at 8:34 am

Good Idea!!

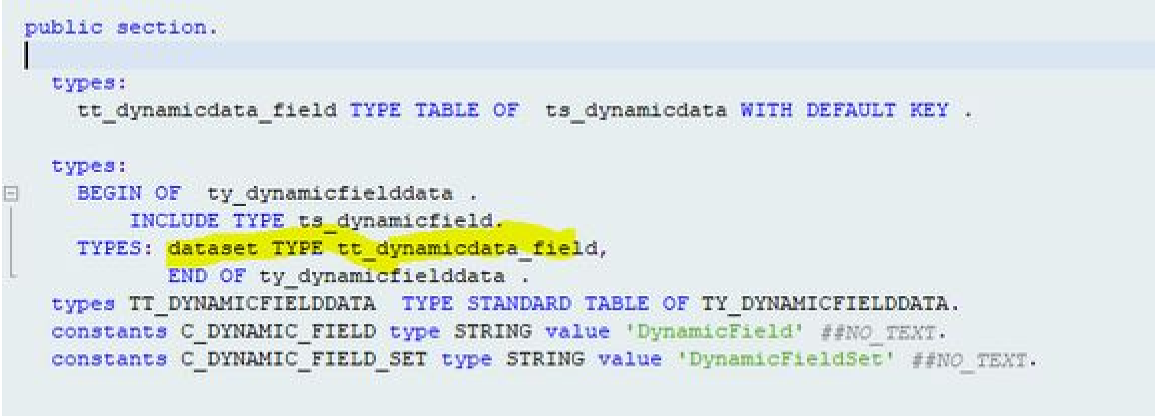
Like (0)

Alexander K

July 12, 2019 at 3:06 am

Hi Pavan.

How did you do that? In segw or it is hardcoded in _MPC class?



```
public section.

  types:
    tt_dynamicdata_field TYPE TABLE OF  ts_dynamicdata WITH DEFAULT KEY .

  types:
    BEGIN OF  ty_dynamicfielddata .
        INCLUDE TYPE ts_dynamicfield.
    TYPES: dataset TYPE tt_dynamicdata_field,
           END OF ty_dynamicfielddata .
  types TT_DYNAMICFIELDDATA  TYPE STANDARD TABLE OF TY_DYNAMICFIELDDATA.
  constants C_DYNAMIC_FIELD type STRING value 'DynamicField' ##NO_TEXT.
  constants C_DYNAMIC_FIELD_SET type STRING value 'DynamicFieldSet' ##NO_TEXT.
```

Like (0)

Jean-François Parmentier

August 8, 2019 at 1:21 pm

Hi Pavan,

very interesting indeed.

I succeeded in using this.

what if I need to use different model in the output table ? not only textfields but datepicker too.

how do I do this ?

Jean-François Parmentier.

Like (0)

Find us on

Privacy	Terms of Use
Legal Disclosure	Copyright
Trademark	Cookie Preferences
Newsletter	Support