1st implement the

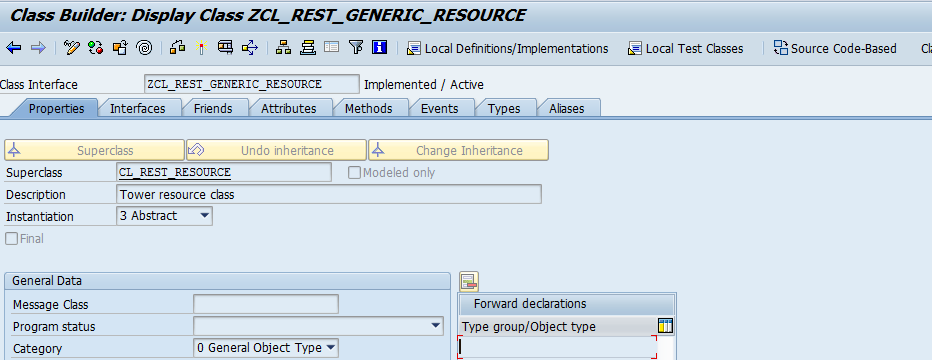
ZCL\_FM\_TO\_API in SE24

2nd below class.

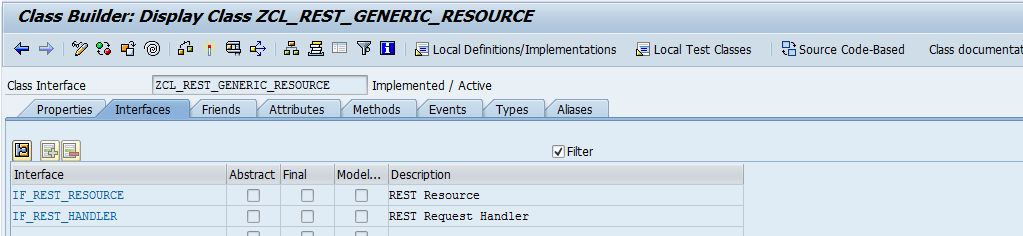
ZCL\_REST\_GENERIC\_RESOURCE- Abstract class

SUPERCLASS: CL\_REST\_RESOURCE

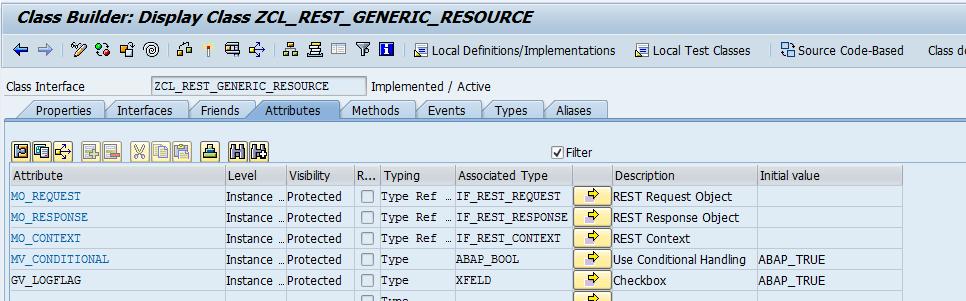
Step-1:



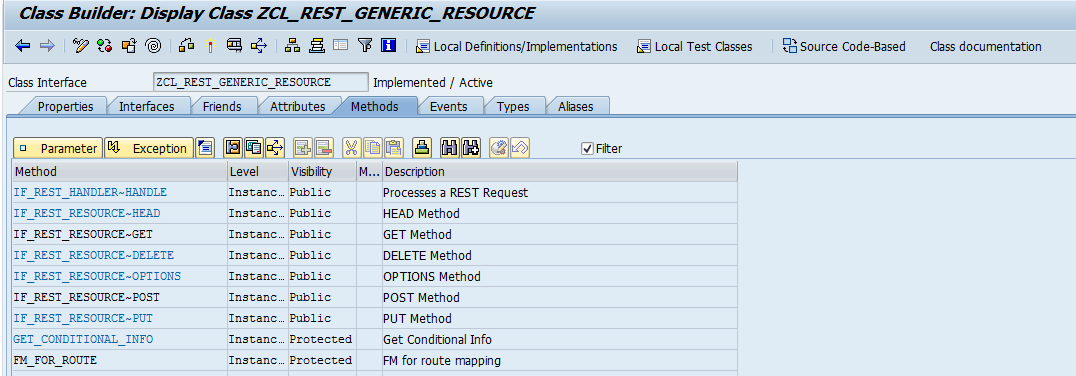
Step-2



Step-3

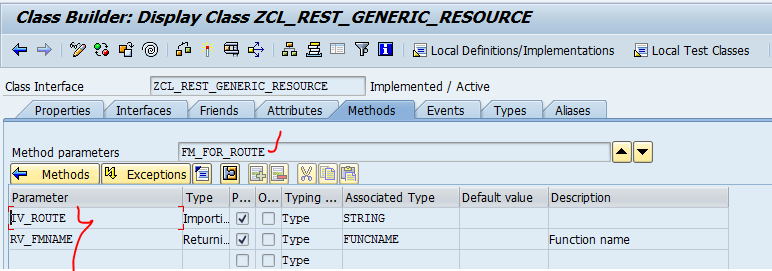


Step-4



Redefine the FM\_for\_route() method:

Parameters:



This method is abstract method this is implemented in sub classes only and it is taking the input as “route” like “/routename” and giving the FM name to the output.

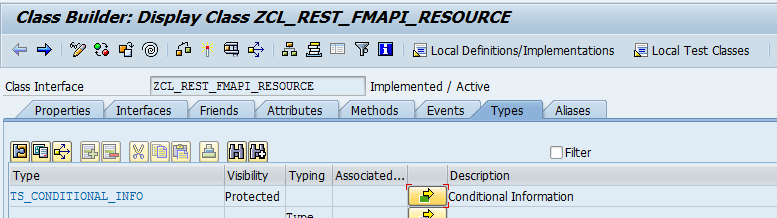
**IF\_REST\_RESOURCE~GET: redefine the “get” mehod.**

**Code:**

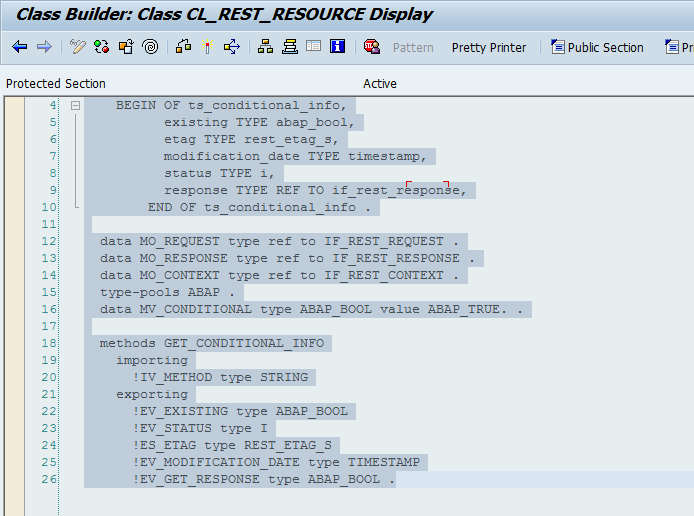
METHOD if\_rest\_resource~get.  
  
  DATA: lv\_json\_req       TYPE string,  
        lv\_json\_res       TYPE string,  
        lo\_fm2api         TYPE REF TO zcl\_fm\_to\_api,  
        lo\_root           TYPE REF TO cx\_root,  
        lv\_path           TYPE string,  
        lv\_route          TYPE string,  
        lv\_fmname         TYPE funcname,  
        lv\_meta           TYPE string,  
        lt\_nvp            TYPE tihttpnvp,  
        lv\_template       TYPE xfeld,  
        lv\_uri            TYPE string,  
        lv\_paramtype      TYPE char3,  
        lv\_content\_type   TYPE string.  
  
  lt\_nvp = mo\_request->get\_entity( )->get\_header\_fields( ).  
  lv\_route = mo\_request->get\_entity( )->get\_header\_field( iv\_name = '~path\_info' ).  
  IF lv\_route = '/fetch-tk'.  
    mo\_response->create\_entity( )->set\_string\_data('Token sent. Check response header.' ).  
  ELSE.  
  
    TRANSLATE lv\_route TO LOWER CASE.  
    lv\_fmname = fm\_for\_route( lv\_route ).  
  
    IF lv\_fmname IS INITIAL.  
      CALL METHOD super->if\_rest\_resource~get( ).  
    ELSE.  
      TRY.  
          lv\_uri = mo\_request->get\_entity( )->get\_header\_field( iv\_name = '~request\_uri' ).  
          TRANSLATE lv\_uri TO LOWER CASE.  
          IF lv\_uri CS 'template=true'.  
            lv\_template = abap\_true.  
          ENDIF.  
          IF lv\_uri CS 'response=true'.  
            lv\_paramtype = 'OUT'.  
          ELSE.  
            lv\_paramtype = 'IN'.  
          ENDIF.  
  
          lv\_json\_res = zcl\_fm\_to\_api=>fmmetadata( iv\_fmname = lv\_fmname iv\_template = lv\_template iv\_paramtype = lv\_paramtype ).  
  
          IF lv\_uri CS 'format=xml'.  
            CALL TRANSFORMATION id SOURCE XML lv\_json\_res  
                 RESULT XML lv\_json\_res.  
            lv\_content\_type = 'application/xml'.  
          ELSE.  
            lv\_content\_type = 'application/json'.  
          ENDIF.  
  
        CATCH cx\_root INTO lo\_root.  
          lv\_json\_res = lo\_root->get\_text( ).  
      ENDTRY.  
  
      mo\_response->create\_entity( )->set\_header\_field(  iv\_name = 'Content-Type' iv\_value = lv\_content\_type ).  
      mo\_response->create\_entity( )->set\_string\_data( lv\_json\_res ).  
  
    ENDIF.  
  ENDIF.  
  
ENDMETHOD.

**Redfine the if\_rest\_resource~post.**

METHOD if\_rest\_resource~post.  
  
  DATA: lv\_json\_req       TYPE string,  
        lv\_json\_res       TYPE string,  
        lo\_fm2api         TYPE REF TO zcl\_fm\_to\_api,  
        lo\_root           TYPE REF TO cx\_root,  
        lo\_logger         TYPE REF TO zcl\_rest\_logger,  
        lv\_path           TYPE string,  
        lv\_route          TYPE string,  
        lv\_fmname         TYPE funcname,  
        lv\_meta           TYPE string,  
        lt\_nvp            TYPE tihttpnvp,  
        lv\_logging        TYPE string.  
  
  TRY.  
  
  
      lv\_logging = mo\_request->get\_entity( )->get\_header\_field( iv\_name = 'logging' ).  
      IF lv\_logging IS NOT INITIAL.  
        TRANSLATE lv\_logging TO LOWER CASE.  
        IF lv\_logging = 'true'.  
          gv\_logflag = abap\_true.  
        ELSEIF lv\_logging = 'false'.  
          CLEAR gv\_logflag.  
        ENDIF.  
      ENDIF.  
  
      IF gv\_logflag IS NOT INITIAL.  
        CREATE OBJECT lo\_logger  
          EXPORTING  
            io\_request = mo\_request.  
      ENDIF.  
      lv\_route = mo\_request->get\_entity( )->get\_header\_field( iv\_name = '~path\_info' ).  
      TRANSLATE lv\_route TO LOWER CASE.  
      lv\_fmname = fm\_for\_route( lv\_route ).  
      TRY.  
          CREATE OBJECT lo\_fm2api  
            EXPORTING  
              iv\_fmname = lv\_fmname.  
        CATCH cx\_root INTO lo\_root.  
          lv\_json\_res = lo\_root->get\_text( ).  
      ENDTRY.  
      IF lo\_fm2api IS BOUND.  
        lv\_json\_res = lo\_fm2api->reqtores( mo\_request->get\_entity( )->get\_string\_data( ) ).  
      ENDIF.  
      IF lo\_logger IS BOUND.  
        lo\_logger->logout( iv\_response = lv\_json\_res ).  
      ENDIF.  
  
    CATCH cx\_root INTO lo\_root.  
      lv\_json\_res = lo\_root->get\_text( ).  
  ENDTRY.  
  IF lv\_json\_res IS INITIAL.  
    lv\_json\_res = 'Unknown error occured'.  
  ENDIF.  
  mo\_response->create\_entity( )->set\_header\_field(  iv\_name = 'Content-Type' iv\_value = 'application/json' ).  
  mo\_response->create\_entity( )->set\_string\_data( lv\_json\_res ).  
  
ENDMETHOD.



protected section.  
  
  types:  
    BEGIN OF ts\_conditional\_info,  
          existing TYPE abap\_bool,  
          etag TYPE rest\_etag\_s,  
          modification\_date TYPE timestamp,  
          status TYPE i,  
          response TYPE REF TO if\_rest\_response,  
        END OF ts\_conditional\_info .  
  
  data MO\_REQUEST type ref to IF\_REST\_REQUEST .  
  data MO\_RESPONSE type ref to IF\_REST\_RESPONSE .  
  data MO\_CONTEXT type ref to IF\_REST\_CONTEXT .  
  type-pools ABAP .  
  data MV\_CONDITIONAL type ABAP\_BOOL value ABAP\_TRUE. .  
  
  methods GET\_CONDITIONAL\_INFO  
    importing  
      !IV\_METHOD type STRING  
    exporting  
      !EV\_EXISTING type ABAP\_BOOL  
      !EV\_STATUS type I  
      !ES\_ETAG type REST\_ETAG\_S  
      !EV\_MODIFICATION\_DATE type TIMESTAMP  
      !EV\_GET\_RESPONSE type ABAP\_BOOL .



REST API SERVICE CREATION USING GENERIC Z CLASSES

# Introduction

We have created a ready-to-use solution for creating REST APIs in 5 easy steps. This document demonstrates these steps.

# Requirement

If the requirement is to create and expose wrapper REST APIs for existing/newly created function modules, this solution can be used. Please note, this solution is only available in D91 and yet to be created in DER/D90/HD1.

# Features

On using this solution, the following features are inherited automatically.

1. Logging and data tracking
2. X-csrf token validation
3. Implicit data conversion
4. Metadata generation
5. Template generation

# Pre-requisites

* SICF Service name should be decided in advance.
* Function module(s) should exist to perform the core task of the required API.
* Unique route name should be decided for mapping to each of the function modules.

# Steps with Example

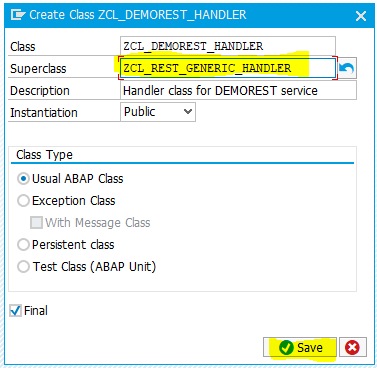
For example, let us assume there is requirement to create a new REST API service named DEMOREST for the following 3 function modules with routes as mentioned below:

|  |  |
| --- | --- |
| **FM Name** | **Route** |
| Z\_TOWER\_PR\_CREATE | /pr-create |
| Z\_TOWER\_PR\_CHANGE | /pr-change |
| Z\_TOWER\_PR\_READ | /pr-read |

The solution can be created using the following 4 steps:

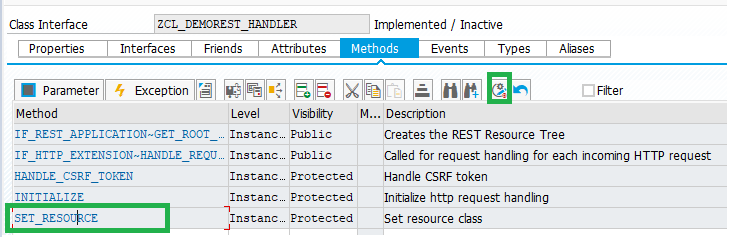
## Step 1🡺 Inherit generic handler class to create your handler class

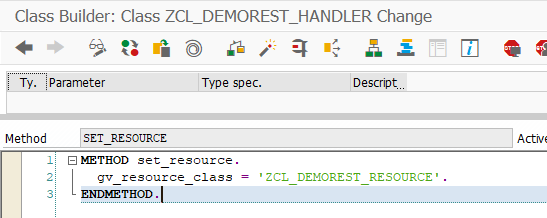
Create a child final class by inheriting abstract class **ZCL\_REST\_GENERIC\_HANDLER** as below (T-code SE24):



## Step 2🡪 Re-Define abstract method ‘SET\_RESOURCE’

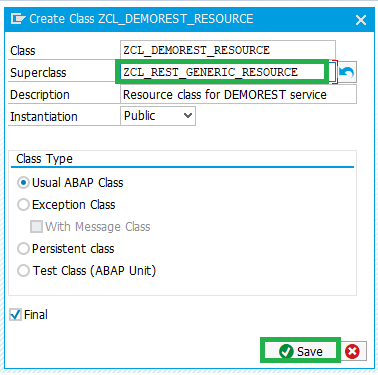
In this step, we need to specify the RESOURCE class name (Created in step 3), by setting global attribute GV\_RESOURCE\_CLASS:





## Step 3: Inherit generic resource class to create your resource class

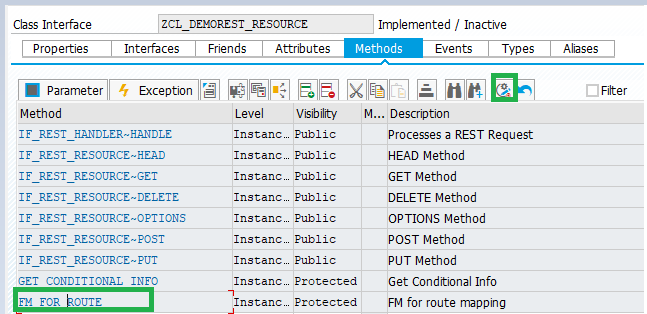
Create a child final class by inheriting abstract class **ZCL\_REST\_GENERIC\_RESOURCE** as below (T-code SE24):

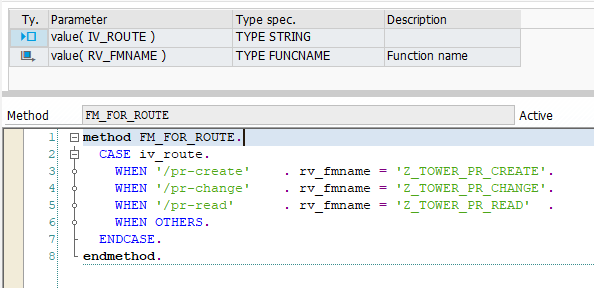


Note: Make sure that the name of the new class is same as specified in step 2.

## Step 4🡪 Re-Define abstract method ‘FM\_FOR\_ROUTE’

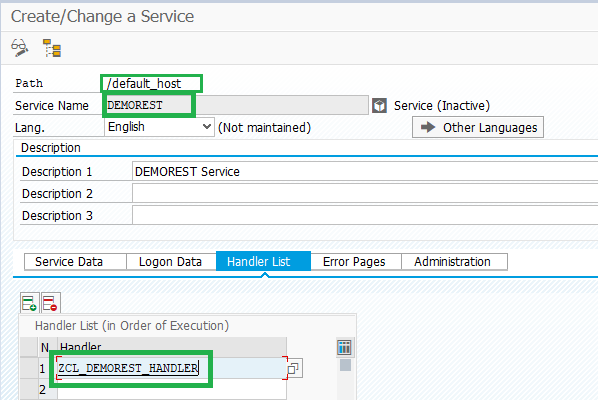
In this step, we need to specify route and Function Module mapping:



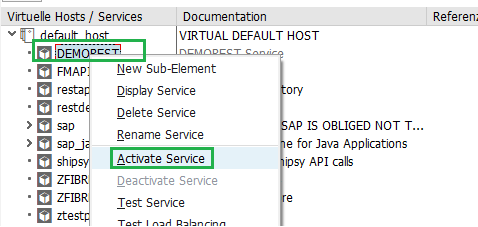


## Step 5 🡪 Create SICF Node and map Handler class created

Go to t-code SICF and create new sub element at your desired path and map Handler class name as below:



Note: Ensure to activate the node after creation:



# Conclusion

That’s all required from development end. Once the service is activated, your rest service is ready for testing, loaded with in-built features mentioned.

# Generated URLs

The service node link can be found by right clicking the service in SICF and then clicking on ‘Test service’. This action should fire-up your browser and you can copy the link.

Please note, to access the core service, you still need to append the route for the required FM in the generated URL at end.

For example, in this case Following URLs are active for FM ‘Z\_TOWER\_PR\_READ’:

|  |  |  |
| --- | --- | --- |
| Method-Type | URL | Result |
| GET | <http://deverpd91.bss.dev.jio.com:8000/demorest/pr-read?sap-client=112> | {      "SIGNATURE": {          "NAME": "Z\_TOWER\_PR\_READ",          "DESCRIPTION": "PR creation",          "IMPORTING": [              {                  "PARAMETER": "INPUT",                  "REQUIRED": "false"              }          ],          "EXPORTING": [              {                  "PARAMETER": "OUTPUT",                  "REQUIRED": "false"              }          ],          "CHANGING": [],          "TABLES": []      },      "METADATA": {          "INPUT": {              "UNIQUE\_ID": {                  "DATATYPE": "CHAR",                  "LENGTH": 18,                  "DESCRIPTION": "Length 18 text"              },              "PR\_NUM": {                  "DATATYPE": "CHAR",                  "LENGTH": 10,                  "DESCRIPTION": "Purchase Requisition Number"              }          }      }  } |