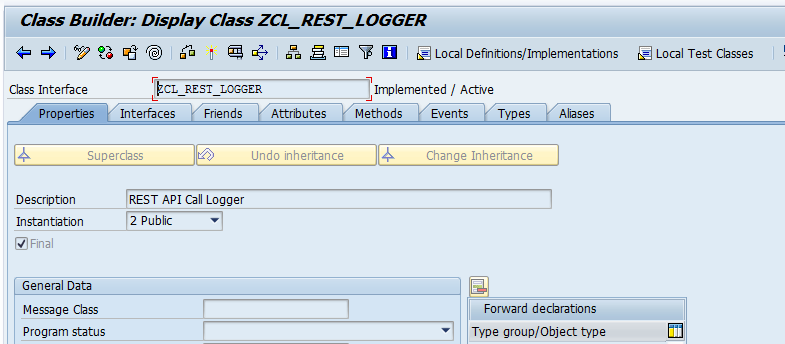
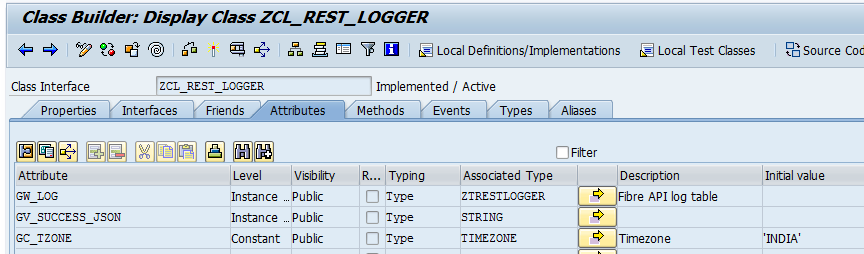
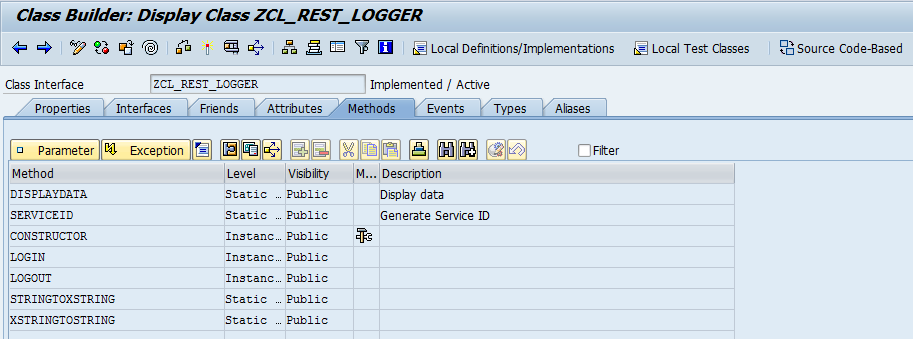
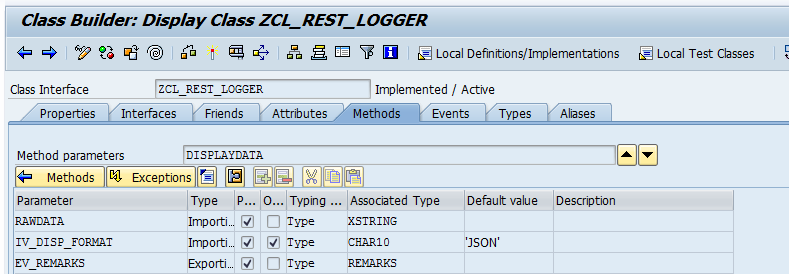
**ZCL\_REST\_LOGGER**





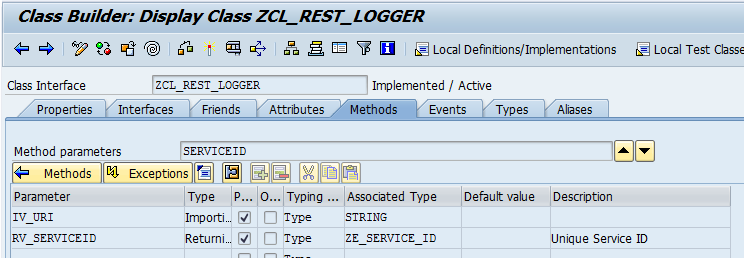


**DISPLAYDATA~Method implementations.**



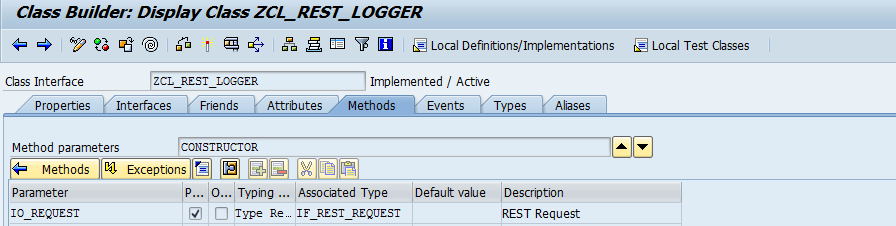
METHOD displaydata.  
  
  DATA:  
  lv\_length   TYPE i,  
  lt\_binary   TYPE STANDARD TABLE OF x255,  
  l\_json      TYPE string,  
  l\_xml       TYPE string,  
  lo\_json\_dsr TYPE REF TO cl\_trex\_json\_deserializer,  
  lo\_root     TYPE REF TO cx\_root,  
  lo\_tree     TYPE REF TO cl\_salv\_tree,  
  lo\_nodes    TYPE REF TO cl\_salv\_nodes,  
  l\_html      TYPE xstring.  
  
  FIELD-SYMBOLS: <lfs\_data> TYPE any,  
                 <lfs\_datatab> TYPE table.  
  
  
  
  IF rawdata IS NOT INITIAL.  
  
    CALL FUNCTION 'SCMS\_XSTRING\_TO\_BINARY'  
      EXPORTING  
        buffer        = rawdata  
      IMPORTING  
        output\_length = lv\_length  
      TABLES  
        binary\_tab    = lt\_binary.  
  
    IF lt\_binary IS NOT INITIAL.  
      CALL FUNCTION 'SCMS\_BINARY\_TO\_STRING'  
        EXPORTING  
          input\_length = lv\_length  
        IMPORTING  
          text\_buffer  = l\_json  
        TABLES  
          binary\_tab   = lt\_binary  
        EXCEPTIONS  
          failed       = 1  
          OTHERS       = 2.  
      IF sy-subrc <> 0.  
        ev\_remarks = 'Falied to convert binary ro string'.  
        RETURN.  
      ENDIF.  
      IF l\_json IS NOT INITIAL.  
        TRY.  
            CASE iv\_disp\_format.  
              WHEN 'JSON'.  
\*            cl\_demo\_output=>display\_json( l\_json ).  
  
                CALL TRANSFORMATION sjson2html SOURCE XML l\_json  
                                         RESULT XML l\_html.  
  
                cl\_abap\_browser=>show\_html( html\_string =  
                  cl\_abap\_codepage=>convert\_from( l\_html ) ).  
  
  
              WHEN 'XML'.  
                CALL TRANSFORMATION id SOURCE XML l\_json  
                               RESULT XML l\_xml.  
                cl\_abap\_browser=>show\_xml( xml\_string = l\_xml ).  
\*            cl\_demo\_output=>display\_xml( l\_xml ).  
  
              WHEN OTHERS.  
            ENDCASE.  
          CATCH cx\_root.  
        ENDTRY.  
      ENDIF.  
    ENDIF.  
  ENDIF.  
  
ENDMETHOD.

**SERVICEID~Implementation**



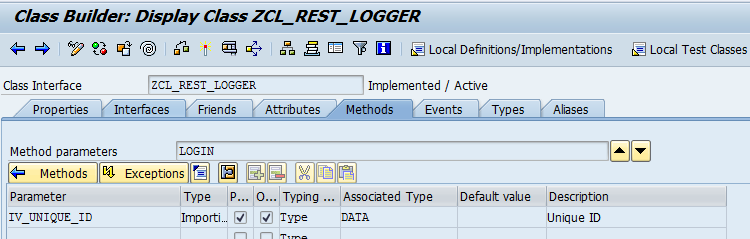
method SERVICEID.  
  
DATA: lt\_string TYPE STANDARD TABLE OF string,  
      lv\_service TYPE string,  
      lv\_route   TYPE string,  
      lv\_params   TYPE string.  
SPLIT iv\_uri at '?' INTO iv\_uri lv\_params.  
SPLIT iv\_uri at '/' INTO TABLE lt\_string.  
  
READ TABLE lt\_string into lv\_service INDEX lines( lt\_string ) - 1.  
READ TABLE lt\_string into lv\_route   INDEX lines( lt\_string ).  
  
rv\_serviceid = |/{ lv\_service }/{ lv\_route }|.  
  
endmethod.

**CONSTRUCTOR~Method Implementation**



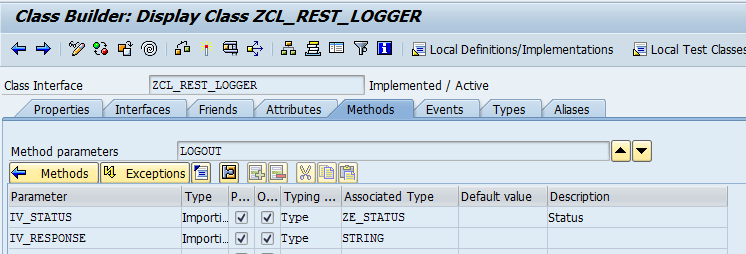
METHOD constructor.  
  TYPES:  
  BEGIN OF ty\_data,  
  unique\_id TYPE ze\_unique\_id,  
  END OF   ty\_data.  
  
  DATA: lw\_data     TYPE ty\_data,  
        lv\_json\_req TYPE string,  
        lo\_root     TYPE REF TO cx\_root,  
        lt\_log      TYPE STANDARD TABLE OF ztrestlogger,  
        lw\_log      TYPE ztrestlogger.  
  
  gw\_log-serviceid = serviceid( io\_request->get\_uri( ) ).  
  lv\_json\_req      = io\_request->get\_entity( )->get\_string\_data( ).  
  
  gw\_log-indata = stringtoxstring( lv\_json\_req ).  
  
\*  TRANSLATE lv\_json\_req TO UPPER CASE. "PWC\_SHAFIQUL 29.01.2021 11:59:55  
  TRY .  
      zui\_cl\_json=>deserialize(  
        EXPORTING  
          json             = lv\_json\_req  
        CHANGING  
          data             =  lw\_data  
      ).  
      gw\_log-unique\_id = lw\_data-unique\_id.  
  
  
      IF gw\_log-unique\_id IS NOT INITIAL.  
  
        SELECT  \*  
          FROM ztrestlogger  
          INTO TABLE lt\_log  
          WHERE serviceid = gw\_log-serviceid  
          AND   unique\_id = gw\_log-unique\_id.  
  
        IF sy-subrc IS INITIAL.  
  
          SORT lt\_log BY outtime DESCENDING.  
          READ TABLE lt\_log INTO lw\_log INDEX 1.  
          IF sy-subrc IS INITIAL AND lw\_log-status = 'S'.  
            gw\_log = lw\_log.  
            gv\_success\_json = xstringtostring( gw\_log-outdata ).  
          ENDIF.  
        ENDIF.  
      ENDIF.  
      me->login( gw\_log-unique\_id ).  
    CATCH cx\_root INTO lo\_root.  
  
  ENDTRY.  
  
ENDMETHOD.

**LOGIN!Method Implementation**



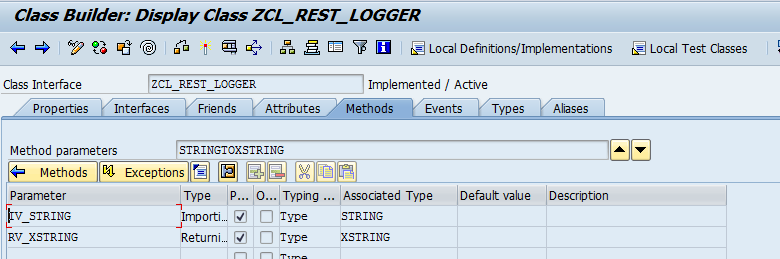
METHOD login.  
  
  gw\_log-changed\_by = sy-uname.  
  GET TIME STAMP FIELD gw\_log-intime.  
  IF iv\_unique\_id IS NOT INITIAL.  
    gw\_log-unique\_id = iv\_unique\_id.  
  ELSE.  
    gw\_log-unique\_id = gw\_log-intime.  
  ENDIF.  
  CONDENSE gw\_log-unique\_id.  
ENDMETHOD.

**LOGOUT~Method Implementation**



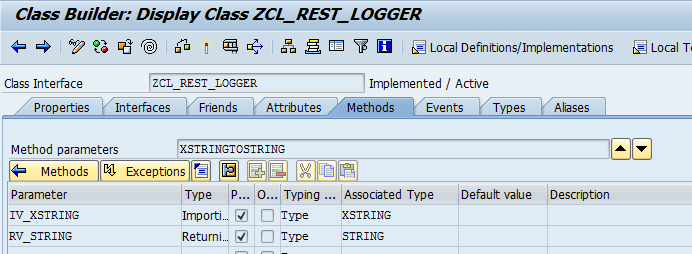
METHOD logout.  
  TYPES:  
  BEGIN OF ty\_resp,  
  status TYPE ze\_status,  
  END OF   ty\_resp.  
  
  DATA: lw\_log  TYPE ztrestlogger,  
        lw\_resp TYPE ty\_resp.  
  
  IF iv\_status IS INITIAL.  
  
    CALL METHOD zui\_cl\_json=>deserialize  
      EXPORTING  
        json = iv\_response  
      CHANGING  
        data = lw\_resp.  
  
    gw\_log-status = lw\_resp-status.  
  ELSE.  
    gw\_log-status = iv\_status.  
  ENDIF.  
  
  gw\_log-outdata = stringtoxstring( iv\_response ).  
  GET TIME STAMP FIELD gw\_log-outtime.  
  INSERT ztrestlogger FROM gw\_log.  
  COMMIT WORK.  
  
ENDMETHOD.

**STRINGTOXSTRING~Method Implementation**



METHOD stringtoxstring.  
IF iv\_string IS NOT INITIAL.  
  
  CALL FUNCTION 'SCMS\_STRING\_TO\_XSTRING'  
    EXPORTING  
      text   = iv\_string  
    IMPORTING  
      buffer = rv\_xstring  
    EXCEPTIONS  
      failed = 1  
      OTHERS = 2.  
  IF sy-subrc <> 0.  
\* Implement suitable error handling here  
  ENDIF.  
  
ENDIF.  
ENDMETHOD.

**XSTRINGTOSTRING~Method Implementation**



METHOD xstringtostring.  
  DATA:  
  lv\_length   TYPE i,  
  lt\_binary   TYPE STANDARD TABLE OF x255.  
  
  CALL FUNCTION 'SCMS\_XSTRING\_TO\_BINARY'  
    EXPORTING  
      buffer        = iv\_xstring  
    IMPORTING  
      output\_length = lv\_length  
    TABLES  
      binary\_tab    = lt\_binary.  
  
  IF lt\_binary IS NOT INITIAL.  
    CALL FUNCTION 'SCMS\_BINARY\_TO\_STRING'  
      EXPORTING  
        input\_length = lv\_length  
      IMPORTING  
        text\_buffer  = rv\_string  
      TABLES  
        binary\_tab   = lt\_binary  
      EXCEPTIONS  
        failed       = 1  
        OTHERS       = 2.  
    IF sy-subrc <> 0.  
    ENDIF.  
  ENDIF.  
  
ENDMETHOD.