METHOD if\_os\_ca\_persistency~get\_persistent\_by\_query.  
\*Espresso Tutorials Schnelleinstieg ABAP S.  152 -159  
\*\*TRY.  
\*CALL METHOD SUPER->IF\_OS\_CA\_PERSISTENCY~GET\_PERSISTENT\_BY\_QUERY  
\*  EXPORTING  
\*    I\_QUERY         =  
\*\*    I\_PARAMETER\_TAB =  
\*\*    I\_PAR1          =  
\*\*    I\_PAR2          =  
\*\*    I\_PAR3          =  
\*\*    I\_SUBCLASSES    = OSCON\_FALSE  
\*\*    I\_UPTO          = 0  
\*  RECEIVING  
\*    RESULT          =  
\*    .  
\*\* CATCH CX\_OS\_OBJECT\_NOT\_FOUND .  
\*\* CATCH CX\_OS\_QUERY\_ERROR .  
\*\*ENDTRY.  
  DATA: flag\_not\_loaded TYPE os\_boolean,  
        bkey\_string TYPE string,  
        temp\_object\_ref TYPE typ\_object\_ref,  
        business\_key TYPE typ\_business\_key,  
        object\_data\_tab TYPE typ\_db\_data\_tab,  
        query TYPE REF TO if\_os\_qry\_service,  
        where\_clause TYPE string,  
        order\_by\_clause TYPE string.  
  
  DATA: temp\_current\_object\_iref TYPE REF TO object.  
  FIELD-SYMBOLS: <fs\_object\_data> TYPE typ\_db\_data.  
  TRY.  
      CALL METHOD map\_initialize\_metadata.  
      query ?= i\_query.  
      CALL METHOD query->map  
        EXPORTING  
          i\_class\_agent = me  
          i\_map\_att\_tab = map\_att\_info\_tab.  
  
      CALL METHOD query->get\_order\_by\_clause  
        RECEIVING  
          result = order\_by\_clause.  
      IF ( i\_parameter\_tab IS SUPPLIED ).  
        IF ( ( i\_par1 IS SUPPLIED ) OR  
             ( i\_par2 IS SUPPLIED ) OR  
          ( i\_par3 IS SUPPLIED ) ).  
  
          RAISE EXCEPTION TYPE cx\_os\_query\_parameter\_error  
          EXPORTING  
            textid = cx\_os\_query\_parameter\_error=>call\_error.  
        ENDIF.  
        CALL METHOD query->get\_where\_clause  
          EXPORTING  
            i\_parameter\_tab = i\_parameter\_tab  
          RECEIVING  
            result          = where\_clause.  
        CALL METHOD map\_load\_from\_database  
          EXPORTING  
            i\_where\_clause    = where\_clause  
            i\_order\_by\_clause = order\_by\_clause  
            i\_subclasses      = i\_subclasses  
            i\_upto            = i\_upto  
          RECEIVING  
            result            = object\_data\_tab.  
      ELSE.  
        CALL METHOD query->get\_where\_clause  
          RECEIVING  
            result = where\_clause.  
        CALL METHOD map\_load\_from\_database  
          EXPORTING  
            i\_where\_clause    = where\_clause  
            i\_order\_by\_clause = order\_by\_clause  
            i\_subclasses      = i\_subclasses  
            i\_upto            = i\_upto  
            par1              = i\_par1  
            par2              = i\_par2  
            par3              = i\_par3  
          RECEIVING  
            result            = object\_data\_tab.  
      ENDIF.  
    CATCH cx\_os\_db\_select.  
      CLEAR object\_data\_tab.  
  ENDTRY.  
  TRY.  
      APPEND internal\_next\_undo\_info  
              TO internal\_transaction\_stack.  
      internal\_current\_transaction = sy-tabix.  
      LOOP AT object\_data\_tab  
        ASSIGNING <fs\_object\_data>.  
        CALL METHOD map\_extract\_identifier  
          EXPORTING  
            i\_db\_data      = <fs\_object\_data>  
          IMPORTING  
            e\_business\_key = business\_key.  
  
        CLEAR current\_object\_iref.  
  
        READ TABLE special\_bkey\_tab INTO current\_special\_object\_info  
        WITH TABLE KEY business\_key = business\_key.  
        IF ( sy-subrc = 0 ).  
  
          READ TABLE special\_object\_info  
          INTO current\_special\_object\_info  
          WITH TABLE KEY  
          object\_id = current\_special\_object\_info-object\_id.  
          CALL METHOD os\_load\_and\_validate\_current  
            EXPORTING  
              i\_index = sy-tabix.  
        ENDIF.  
        IF ( NOT current\_object\_iref IS INITIAL ).  
          CASE current\_object\_info-pm\_status.  
            WHEN oscon\_ostatus\_deleted.  
              "  object marked for deletion  
              temp\_current\_object\_iref = current\_object\_iref.  
              bkey\_string = map\_serialize\_bkey( business\_key ).  
              CALL METHOD os\_clear\_current.  
              CLEAR current\_special\_object\_info.  
              CLASS cx\_os\_object\_not\_found DEFINITION LOAD.  
              RAISE EXCEPTION TYPE cx\_os\_object\_not\_found  
              EXPORTING  
                object = temp\_current\_object\_iref  
                bkey = bkey\_string  
                textid = cx\_os\_object\_not\_found=>deleted\_by\_bkey.  
            WHEN oscon\_ostatus\_transient.  
              temp\_current\_object\_iref = current\_object\_iref.  
              bkey\_string = map\_serialize\_bkey( business\_key ).  
              CALL METHOD os\_clear\_current.  
              CLEAR current\_special\_object\_info.  
              CLASS cx\_os\_object\_not\_found DEFINITION LOAD.  
              RAISE EXCEPTION TYPE cx\_os\_object\_not\_found  
              EXPORTING  
                object = temp\_current\_object\_iref  
                bkey = bkey\_string  
                textid =  
cx\_os\_object\_not\_found=>is\_transient\_by\_bkey.  
            WHEN oscon\_ostatus\_not\_loaded.  
              flag\_not\_loaded = oscon\_true.  
            WHEN OTHERS.  
              flag\_not\_loaded = oscon\_false.  
          ENDCASE.  
        ELSE.  
          flag\_not\_loaded = oscon\_true.  
        ENDIF.  
        temp\_object\_ref ?= current\_object\_iref.  
        IF ( flag\_not\_loaded = oscon\_true ).  
          CALL METHOD pm\_check\_and\_set\_attributes  
            EXPORTING  
              i\_object\_data = <fs\_object\_data>  
              i\_id\_provided = id\_status\_none.  
          temp\_object\_ref ?= current\_object\_iref.  
          CALL METHOD os\_pm\_loaded\_persistent.  
        ENDIF.  
        APPEND temp\_object\_ref TO result.  
        CALL METHOD os\_clear\_current.  
      ENDLOOP.  
  
      DELETE internal\_transaction\_stack  
      INDEX internal\_current\_transaction.  
      ADD -1 TO internal\_current\_transaction.  
      IF ( internal\_transaction\_stack IS INITIAL ).  
        clear internal\_undo\_info.  
        internal\_next\_undo\_info = 1.  
      ENDIF.  
    CLEANUP.  
      CALL METHOD os\_internal\_undo.  
      CALL METHOD os\_clear\_current.  
      CLEAR current\_special\_object\_info.  
  ENDTRY.  
  
  CLEAR current\_special\_object\_info.  
  
ENDMETHOD.