Persistent Object Services using ABAP

The following steps illustrate the usage of ABAP - Persistent Object Services for Database Update:

Step 1. Go to t-code SE11.
Create a Z Table (Copy the Standard Flights Demo Table into ZTB_SFLIGHT).

Step 2. Go to t-code SE24.
Create a Persistent Class ZCL_PERSIST_SFLIGHT

Step 3. Click on "Persistence representation" - Ctrl+F4 in the GOTO menu to add the Z-Table

Step 4. Save and activate the class and also the Class Actor.

Step 5. Create a Custom report program and execute it.

Results: The DB CRUD (Create, Read/Update, Delete) will be done efficiently using the ABAP Persistent Object Services.

Program Name   : ZRPT_PERSIST_SFLIGHT
Author         : Subrahmanya Pindiproli
Requested By   : N.A
Date created   : 2012/04/12
SAP Release    : ECC6.0
Project Name   : N.A
Task #         :
Description    : Displays a Demo Usage of Persistent Object Services for DB CRUD Activities

REPORT zrpt_persist_sflight.

SELECTION-SCREEN BEGIN OF BLOCK blk1 WITH FRAME TITLE tit1.
PARAMETERS: carrid LIKE ztb_sflight-carrid OBLIGATORY,
            connid LIKE ztb_sflight-connid OBLIGATORY,
            fldate LIKE ztb_sflight-fldate OBLIGATORY,
            price LIKE ztb_sflight-price.
SELECTION-SCREEN END OF BLOCK blk1.

SELECTION-SCREEN BEGIN OF BLOCK blk2 WITH FRAME TITLE tit2.
PARAMETERS: r1 TYPE c RADIOBUTTON GROUP rad1,
            r2 TYPE c RADIOBUTTON GROUP rad1,
            r3 TYPE c RADIOBUTTON GROUP rad1.
SELECTION-SCREEN END OF BLOCK blk2.

CLASS lcl_class1 DEFINITION.
PUBLIC SECTION.
DATA: agent TYPE REF TO zca_persist_sflight,
     flights TYPE REF TO zcl_persist_sflight.
METHODS:
    fetch_persistent IMPORTING  im_carrid LIKE carrid
                               im_connid LIKE connid.
PUBLIC SECTION.

PROCEDURE EXPORTING.

CREATE_PERSISTENT IMPORTING.

DELETE_PERSISTENT IMPORTING.

PRIVATE SECTION.

DATA: carrid TYPE ztb_sflight-carrid,
     connid TYPE ztb_sflight-connid,
     fldate TYPE ztb_sflight-fldate,
     price TYPE ztb_sflight-price.

ENDCLASS. "lcl_class1 DEFINITION

*---------------------------------------------------------------------*
* CLASS lcl_class1 IMPLEMENTATION
*---------------------------------------------------------------------*
CLASS lcl_class1 IMPLEMENTATION.

METHOD fetch_persistent.

agent = zca_persist_sflight=>agent.
TRY.
    agent->get_persistent( EXPORTING i_carrid = im_carrid
                          i_connid = im_connid
                          i_fldate = im_fldate
                          RECEIVING result = flights ).
    carrid = flights->get_carrid( ).
    connid = flights->get_connid( ).
    fldate = flights->get_fldate( ).
    price = flights->get_price( ).
    IF r1 EQ 'X'.
        output( ).
    ENDIF.
CATCH cx_os_object_not_found.
    MESSAGE 'Object doesn''t exists' TYPE 'I' DISPLAY LIKE 'E'.
ENDTRY.
ENDMETHOD. "fetch_persistent

METHOD output.

WRITE:/ carrid, connid, fldate, price.
ENDMETHOD. "output

METHOD create_persistent.

fetch_persistent( EXPORTING im_carrid = im_carrid
                  im_connid = im_connid
                  im_fldate = im_fldate ).
TRY.
    agent->create_persistent( EXPORTING i_carrid = im_carrid
                              i_connid = im_connid
                              i_fldate = im_fldate
                              i_price = im_price
                              RECEIVING result = flights ).
    COMMIT WORK.
    WRITE 'Object Created'.
CATCH cx_os_object_already_exists .
    MESSAGE 'Object already exists' TYPE 'I' DISPLAY LIKE 'E'.
ENDTRY.
ENDMETHOD. "create_persistent

METHOD delete_persistent.

fetch_persistent( EXPORTING im_carrid = im_carrid
                  im_connid = im_connid
                  im_fldate = im_fldate ).
TRY. 
  agent->delete_persistent( EXPORTING i_carrid = im_carrid 
                          i_connid = im_connid 
                          i_fldate = im_fldate ).

COMMIT WORK.

WRITE 'Object Deleted'.
CATCH cx_os_object_not_existing .
  MESSAGE 'Object doesn''t exists' TYPE 'I' DISPLAY LIKE 'E'.
ENDTRY.
ENDMETHOD. "delete_persistent
ENDCLASS. "lcl_class1 IMPLEMENTATION

DATA ref_class1 TYPE REF TO lcl_class1.
  *---------------------------------------------------------------------*
  | Load-of-Program                                                  |
  *---------------------------------------------------------------------*
LOAD-OF-PROGRAM.
  tit1 = text-001. tit2 = text-001.
  *---------------------------------------------------------------------*
  | Start-of-Selection                                               |
  *---------------------------------------------------------------------*
START-OF-SELECTION.
CREATE OBJECT ref_class1.
IF r1 EQ 'X'.
  ref_class1->fetch_persistent( EXPORTING im_carrid = carrid 
                                 im_connid = connid 
                                 im_fldate = fldate ).
ELSEIF r2 EQ 'X'.
  ref_class1->create_persistent( EXPORTING im_carrid = carrid 
                                  im_connid = connid 
                                  im_fldate = fldate 
                                  im_price = price ).
ELSE.
  ref_class1->delete_persistent( EXPORTING im_carrid = carrid 
                                 im_connid = connid 
                                 im_fldate = fldate 
                                 im_price = price ).

ENDIF.

ENDIF.

ENDIF.

ENDIF.

ENDIF.
Advantages of using persistent object services for DB CRUD activities:
1. It allows smaller, autonomous units to be built so you can test each unit independently, before it is integrated into the larger application.
2. Very well defined modular approach, helps in efficient development, enhancement and bug fixing.
3. Overall maintainability of the code improves multifold.