1.user command event in classes in oops

User command in reports

**SET USER-COMMAND**

[**Quick Reference**](javascript:call_link('abapset_user-command_shortref.htm'))

**Syntax**

**SET USER-COMMAND fcode.**

**Effect**

**Raises a list event with a function code specified in fcode. fcode must be a**[**character-like data object**](javascript:call_link('abencharlike_data_object_glosry.htm'))**. This statement can be used when creating a list. After completion, but before display of the current list, the runtime environment responds as if a user action were performed in the displayed list using the function code specified in fcode.**

**The assignment of list events to function codes corresponds to the description under**[**AT USER-COMMAND**](javascript:call_link('abapat_user-command.htm'))**:**

* **The predefined function codes of the tables specified under AT USER-COMMAND are caught by the runtime environment**
* **The function codes "PICK" and "PFnn" raise the events**[**AT LINE-SELECTION**](javascript:call_link('abapat_line-selection.htm'))**or**[**AT PFnn**](javascript:call_link('abapat_pfnn.htm'))
* **all other functions raise the event**[**AT USER-COMMAND**](javascript:call_link('abapat_user-command.htm'))

**If the corresponding event block is implemented, the value of sy-lsind is increased by one and the event block is executed.**

**If multiple SET USER-COMMAND statements are used when creating a list, all are ignored except the last one.**

**Notes**

* **The function code "PICK" only raises an event if the cursor is positioned on a list line.**
* **If a function code is assigned to the "Return key in the current**[**GUI status**](javascript:call_link('abengui_status_glosry.htm'))**, this function code is used instead of the one specified in fcode.**

**Example**

**Creates (program-driven) one basic list and two details lists and displays a search dialog box in the second details list using the predefined function code "%SC". The statement SET CURSOR is used to position the cursor in a list line in the event block AT LINE-SELECTION to enable the function code "PICK".**

**START-OF-SELECTION.  
  SET USER-COMMAND 'MYCOMM'.  
  WRITE 'Basic List'.  
  
AT USER-COMMAND.  
  CASE sy-ucomm.  
    WHEN 'MYCOMM'.  
      WRITE 'Details List from USER-COMMAND,'.  
      WRITE: 'SY-LSIND', sy-lsind.  
      SET CURSOR LINE 1.  
      SET USER-COMMAND 'PICK'.  
  ENDCASE.  
  
AT LINE-SELECTION.  
  WRITE 'Details List from LINE-SELECTION,'.  
  WRITE: 'SY-LSIND', sy-lsind.  
  SET USER-COMMAND '%SC'.**

2. events in class

**SAP ABAP - Object Events**

An **event** is a set of outcomes that are defined in a class to trigger the event handlers in other classes. When an event is triggered, we can call any number of event handler methods. The link between a trigger and its handler method is actually decided dynamically at run-time.

In a normal method call, a calling program determines which method of an object or a class needs to be called. As fixed handler method is not registered for every event, in case of event handling, the handler method determines the event that needs to be triggered.

An event of a class can trigger an event handler method of the same class by using the RAISE EVENT statement. For an event, the event handler method can be defined in the same or different class by using the FOR EVENT clause, as shown in the following syntax −

FOR EVENT <event\_name> OF <class\_name>.

Similar to the methods of a class, an event can have parameter interface but it has only output parameters. The output parameters are passed to the event handler method by the RAISE EVENT statement that receives them as input parameters. An event is linked to its handler method dynamically in a program by using the SET HANDLER statement.

When an event is triggered, appropriate event handler methods are supposed to be executed in all the handling classes.

Example

REPORT ZEVENT1.

CLASS CL\_main DEFINITION.

PUBLIC SECTION.

DATA: num1 TYPE I.

METHODS: PRO IMPORTING num2 TYPE I.

EVENTS: CUTOFF.

ENDCLASS.

CLASS CL\_eventhandler DEFINITION.

PUBLIC SECTION.

METHODS: handling\_CUTOFF FOR EVENT CUTOFF OF CL\_main.

ENDCLASS.

START-OF-SELECTION.

DATA: main1 TYPE REF TO CL\_main.

DATA: eventhandler1 TYPE REF TO CL\_eventhandler.

CREATE OBJECT main1.

CREATE OBJECT eventhandler1.

SET HANDLER eventhandler1→handling\_CUTOFF FOR main1.

main1→PRO( 4 ).

CLASS CL\_main IMPLEMENTATION.

METHOD PRO.

num1 = num2.

IF num2 ≥ 2.

RAISE EVENT CUTOFF.

ENDIF.

ENDMETHOD.

ENDCLASS.

CLASS CL\_eventhandler IMPLEMENTATION.

METHOD handling\_CUTOFF.

WRITE: 'Handling the CutOff'.

WRITE: / 'Event has been processed'.

ENDMETHOD. ENDCLASS.

The above code produces the following output −

Handling the CutOff

Event has been processed

unit testing

st05 - is there any object that tell about the execution time of a statement and purpose of the st05

when we have to use cds views than abap views

difference between badi and user exit

hana amdp

about cds view

why it has to use cds views

badi filters

if I want to Implement badi implementations sequentially is it possible

best coding practices

types of reports

lock objects and how to create lock objects

cds view session variables

tmg events

cds view session variables

types of joins

Associations and its cardinality

how to print data from 3 different table in 3 different alv's in output

pre requisites to check when using for all entries

pre requisites when using select queries

differences between delivery class and data class

coding best practices

parallel cursor usage - nested loops what it will do

few open sql syntaxes

differences between cds view and extend view

steps to create odata services

difference between sy-tabix and sy-index

tcode sccm

types of joins and how it will work

default events will call when we run alv report

functions used in alv

how form will be called - driver program

debugging in forms

function modules used to call forms

events in tmg and its uses

event in classical report and interactive report

use of at selection screen on field

differences between at selection screen and at selection screen output

In loop , transporting values ------ ?

Answers:

1. The main difference between FOR ALL ENTRIES and INNER JOIN is that FOR ALL ENTRIES removes duplicate results, while INNER JOIN only returns records that match in both tables:

* **FOR ALL ENTRIES**

Eliminates duplicates from results. When using FOR ALL ENTRIES, the system selects records that meet the where condition and then removes duplicates.

* **INNER JOIN**

Combines records from two tables when there are matching values in a field common to both tables. INNER JOIN is the most common type of join.

Here are some other things to know about FOR ALL ENTRIES and INNER JOIN:

* When using FOR ALL ENTRIES, you generally end up with at least two internal tables.
* Avoid using INNER JOIN when you want to retrieve rows from one table that may or may not have corresponding rows in another table.

**Joins are used to fetch the data from more than one table**.

**1. Inner join**: This statement is used to extract the data from the multiples tables by

joining them. Up to 2 tables can only be joined. For more than 3 tables is

not advisable as it puts heavy load on database

**2. For all entries**: This statement is used to extract the data from the data base table

based on the entries of the Base internal table. The load on data base

will be very less.

Join statement is used to combine data from multiple tables based on a common field, while select for all entries is used to retrieve data from multiple tables based on a list of entries.

* Join statement is used when we want to fetch data from multiple tables based on a common field.
* Select for all entries is used when we want to retrieve data from multiple tables based on a list of entries.
* Join statement can be used with inner join, left outer join, right outer join, and full outer join.
* Select for all entries is more efficient when dealing with large datasets.
* Join statement requires a common field between the tables, while select for all entries requires a list of entries.

2. Explanation of each join type:

* **Inner Join:**

Returns only the rows where the join condition is met in both tables, essentially combining matching records from each table.

* **Left Outer Join:**

Retrieves all rows from the "left" table and matching rows from the "right" table, filling in any missing values from the right table with nulls.

* **Right Outer Join:**

Similar to a left outer join, but instead retrieves all rows from the "right" table and matching rows from the "left" table, filling in nulls for missing left table values.

Important points to remember:

* **Syntax:**

To specify a join type in ABAP SQL, use the relevant keyword like "INNER JOIN", "LEFT OUTER JOIN", or "RIGHT OUTER JOIN" within the SELECT statement.

* **Cross Join: full outer join**

While not as commonly used, a "CROSS JOIN" also exists in ABAP, which results in a Cartesian product, returning all possible combinations of rows from both tables.

3. To use the "AT USER-COMMAND" event within a function module in SAP ABAP, you need to create a screen within the function module and define a custom function code, which will then trigger the "AT USER-COMMAND" event when the user selects that function code on the screen; essentially, allowing you to execute specific logic based on the user's screen interaction.

4. at user command will be used in reports as follows:

**REPORT demo\_at\_user\_command.  
  
START-OF-SELECTION.  
  SET PF-STATUS 'MYLIST'.  
  WRITE 'List line'.  
  
AT USER-COMMAND.  
  IF sy-lsind = 20.  
    SET PF-STATUS 'MYLIST' EXCLUDING 'MY\_SELECTION'.  
  ENDIF.  
  CASE sy-ucomm.  
    WHEN 'MY\_SELECTION'.  
      WRITE: / 'You worked on list', sy-listi,  
             / 'You are on list', sy-lsind.  
    ...  
  ENDCASE.**

Similar to at line selection.

5.