A trigger is the piece of code that executed before and after a record is Inserted/Updated/Deleted from the force.com database.

Types of Triggers:

– Before Triggers

– After Triggers

Before Trigger: Before triggers are used to perform the logic on the same object and specifically we cannot use the DML operation (Insert, update, delete) on these triggers. These triggers fired before the data saved into the database.

After Trigger: After triggers are used to perform the logic on the related objects and these triggers are used access the fields values that are created by system (Ex: CreatedBy, LasteModifiedBy , Record Id etc..).

Bulk Triggers:

By default, every trigger is a bulk trigger which is used to process the multiple records at a time as a batch. For each batch of 200 records.

Executing the Trigger

Following are the events on which we can fir the trigger −

insert

update

delete

merge

upsert

undelete

Syntax

Trigger <trigger name> on <Object name> (trigger Events) {

// Implement the Logic here

}

Trigger Context Variables:

All the trigger context variables prefixed with “Trigger.” (Ex: Trigger.isInsert, etc..)

isInsert: Returns true if the trigger fired due to insert operation

isUpdate: Returns true if the trigger fired due to the update operation.

isDelete: Returns true if the trigger fired due to delete operation.

isBefore: Returns true if the trigger fired before the record saved.

isAfter: Returns true if the trigger fired after the record saved.

New: Returns a list of the new version of sObject records.

Old: Returns a list of an old version of sObject records.

NewMap: Returns a map of a new version of sObject records. (map stored in the form of map<id,newRecords>)

OldMap: Returns a map of an old version of sObject records. (map stored in the form of map<id, oldRecords>)

Size: Returns an integer (total number of records invoked due to trigger invocation for the both old and new)

isExecuting: Returns true if the current apex code is a trigger.

The below table tells about the events we can use in the new trigger and old trigger

Trigger Event Trigger.New Trigger.Old

Before Insert Yes No

Before Update Yes Yes

Before Delete No Yes

After UnDelete No Yes

After Insert Yes No

After Update Yes Yes

After Delete No Yes

Trigger Context Variable considerations:

– Trigger.Old is always read-only

– We cannot delete trigger.new

– In before triggers, trigger.new can be used to update the fields on the same object.

– In After trigger, we get a runtime exception when the user tries to modify the fields in the same object.

Trigger Example 1

Suppose we received a business requirement that we need to create an Invoice Record when Customer's 'Customer Status' field changes to Active from Inactive. For this, we will create a trigger on APEX\_Customer\_\_c object by following these steps −

Step 1 − Go to sObject

Step 2 − Click on Customer

Step 3 − Click on 'New' button in the Trigger related list and add the trigger code as give below.

// Trigger Code

trigger Customer\_After\_Insert on APEX\_Customer\_\_c (after update) {

List InvoiceList = new List();

for (APEX\_Customer\_\_c objCustomer: Trigger.new) {

if (objCustomer.APEX\_Customer\_Status\_\_c == 'Active') {

APEX\_Invoice\_\_c objInvoice = new APEX\_Invoice\_\_c();

objInvoice.APEX\_Status\_\_c = 'Pending';

InvoiceList.add(objInvoice);

}

}

// DML to insert the Invoice List in SFDC

insert InvoiceList;

}

Explanation

Trigger.new − This is the context variable which stores the records currently in the trigger context, either being inserted or updated. In this case, this variable has Customer object's records which have been updated.

There are other context variables which are available in the context – trigger.old, trigger.newMap, trigger.OldMap.

rigger Example 2

The above trigger will execute when there is an update operation on the Customer records. Suppose, the invoice record needs to be inserted only when the Customer Status changes from Inactive to Active and not every time; for this, we can use another context variable trigger.oldMap which will store the key as record id and the value as old record values.

// Modified Trigger Code

trigger Customer\_After\_Insert on APEX\_Customer\_\_c (after update) {

List<apex\_invoice\_\_c> InvoiceList = new List<apex\_invoice\_\_c>();

for (APEX\_Customer\_\_c objCustomer: Trigger.new) {

// condition to check the old value and new value

if (objCustomer.APEX\_Customer\_Status\_\_c == 'Active' &&

trigger.oldMap.get(objCustomer.id).APEX\_Customer\_Status\_\_c == 'Inactive') {

APEX\_Invoice\_\_c objInvoice = new APEX\_Invoice\_\_c();

objInvoice.APEX\_Status\_\_c = 'Pending';

InvoiceList.add(objInvoice);

}

}

// DML to insert the Invoice List in SFDC

insert InvoiceList;

}