**Triggers:**

**Example:**

trigger HelloWorldTrigger on Account (before insert) {

System.debug('Hello World!');

}

**Execute:**

Account a = new Account(Name='Test Trigger');

insert a;

or

Directly Create Record in Object Record Page -New

Before Triggers:

are used to update or validate record values before they’re saved to the database.

After Trigger :

are used to access field values that are set by the system , and to affect changes in other records. The records that fire the after trigger are read-only.

**Basic Structure**:

ApexTrigger

Trigger AccountTrigger On Account(before Insert ,before Update, after Insert)

{

if(Trigger.isInsert)

{

if(Trigger.isBefore)

{

//code to execute

AccountClass.beforeInsert(Trigger.New); //example

}

else if(Trigger.isAfter)

{

//code

}

}

if(Trigger.isUpdate)

{

if(Trigger.isBefore)

{

//code

}

}

}

**Its very appropriate to create Handlerclass to write the code which is to be executed once trigger fired than writing the same inside the trigger class**

ApexClass:

Class AccountClass

{

Public static void beforeInsert(List <Account> list1)

{

for(Account ac:list1)

{

ac.fieldName=’Value’;

}

}

}

**Insert: (isInsert)**

* Trigger.old: null
* Trigger.new: record data .

1. **Before Insert: (isBefore)**

* No Id generated .
* To perform some manipulation before the record is inserted to object.
* No DML’s Required.
* Ex: To create a record for Account :

With name:Trigger Before Insert

Ratings: ‘Hot’ // Id should not be specified.

Apex Trigger:

if(Trigger.isInsert)

{

if(Trigger.isBefore)

{

AccountClass.beforeInsert(Trigger.new);

}

}

Apex Class:

public static void beforeInsert(List <Account> list1)

{

for(Account acn:list1)

{

if(acn.industry=='Media')

{

//acn.id='0012w00001Hj2PEAAZ'; it doesnt recieve it , it just take its own populated id.

acn.Rating='Hot';

acn.Description='My firstAccount';

}

}

}

**After Insert: (isAfter):**

* ID will be generated. || Record is Readonly.
* Here new record is created , we get record ID , Using this we can write logic to update records , or perform any task. (Can Use DML:Insert)

ApexTrigger:

if(Trigger.isInsert)

{

if(Trigger.isAfter)

{

AccountClass.afterInsert(Trigger.new);

}}

Apex Class:

Here after inserting the Account Record , We are going insert new record to opportunity.(We can perform any task).

public static void afterInsert(List <Account> list2)

{

List <Opportunity> opp=new List<Opportunity>();

for(Account acn:list2)

{

Opportunity op=new Opportunity();

op.AccountId=acn.id;

op.Amount=1000;

op.name=acn.name;

op.StageName='Prospecting';

op.Discount\_Percent\_\_c=10;

op.CloseDate=System.today();

opp.add(op);

}

insert opp;

}

**Update (isUpdate)**

* Trigger.old:record data
* Trigger.new : record data.

1. **Before Update (isBefore)**

* No DML’s
* To Perform some manipulations on other field on same obj.
* Ex.When Phone is equal to ‘92983’ , Set description as valid

**ApexTrigger:**

if(Trigger.isUpdate)

{

if(Trigger.isBefore)

{

AccountClass.beforeUpdate(Trigger.New);

}

}

**Apex Class:**

public static void beforeUpdate(List <Account> list3)

{

for(Account a:list3)

{

if(a.Phone!=null && a.Phone=='9148776818')

{

a.Description='Valid number';

}

}

**Ex:** To compare the current and previous Phone Number And Change Description.

To Do this we use oldMap: which contains previous version of record.

ApexTrigger:

AccountClass.beforeUpdateNew(Trigger.New,Trigger.oldMap);

Apexclass:

public static void beforeUpdateNew(List <Account> list4 , Map<Id,Account> oldMap)

{

for(Account a:list4)

{

if(oldMap!=null && a.Phone !=oldMap.get(a.Id).Phone)

{

a.Description='Phone Number Modified to ' + a.Phone;

}

}

}

* Map is used to store the old map record .
* In if:

------It checks The particular record has the previous version.

------the current value is not equal to previous value.

* Then updates Description.

1. **After Update:**

* We can’t make changes on the field of same record .
* We get AccountTrigger: execution of AfterUpdate caused by: System.FinalException: Record is read-only Error.
* We can perform action on Other Object .
* Ex: After the Phone Number Updated in Account , We can update the same information to Opportunity(Using Update DML :Update )

Invalid Scenario:

ApexClass:

if(a.Phone !=null)

{

a.Description='After Insertion Update'; //Error :Readonly

}

Valid Scenario:

When Account Phone number is updated The Description field in opportunity is updated.

ApexTrigger:

else if(Trigger.isAfter)

{

AccountClass.afterUpdate(Trigger.New, Trigger.oldMap);

}

ApexClass:

public static void afterUpdate(List<Account> list7, Map<Id,Account> oldMap)

{

Map<Id,Account> map1=new Map<Id,Account>();

for(Account a : list7)

{

if(oldMap!=null && a.Phone !=oldMap.get(a.Id).Phone)

{

map1.put(a.Id,a);

}

}

List <Opportunity> oppList=new List<Opportunity>();

for(Opportunity opp:[select id,AccountId,Description from Opportunity where AccountId IN : map1.keySet()])

{

if(map1.containsKey(opp.AccountId))

{

//opp.Description=map1.get(opp.AccountId).Phone;

opp.Description='Opp updated';

oppList.add(opp);

}

}

if(!oppList.isEmpty())

{

update oppList;

}}

**Delete:**

1. IsBefore and IsAfter:

* Both works same.
* Trigger.old: will have value
* Trigger.new : null

**After UnDelete**

* Triggers when restoring the value.
* Trigger.old: Null
* Trigger.new : Will have values.