Triggers in Salesforce – Apex Developer Guide

In this Salesforce tutorial, we will learn about triggers in Salesforce and different types of triggers in Salesforce. So what is a trigger in Salesforce? Is it a class, an Object or apex code ? Most of the people will answer Trigger is an apex code ,yes it is correct , However rest two are also correct .

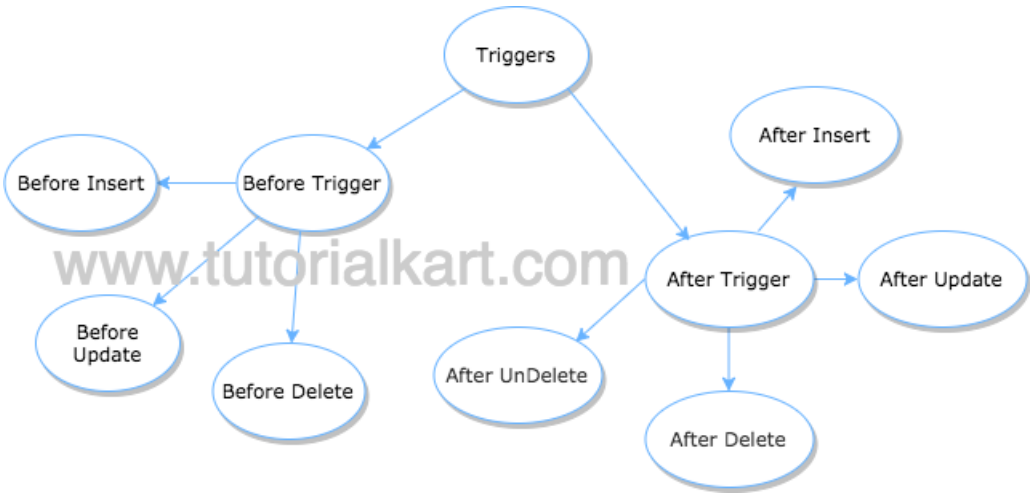
What are triggers in salesforce?

Triggers in Salesforce are programmatic event handlers which is an Apex code that gets executed when a record is saved. Trigger is an object where for each trigger we have written, Salesforce will create a record in **`ApexTrigger`** object.

- Apex Trigger is also a class which contains twelve static context variables.
- Triggers in Salesforce gets executed when a DML operation occurs on an sObject record.
- Apex triggers run on Server side, not on client side.
- Triggers can handle Database manipulation language (DML) operations, can execute SOQl and can call custom Apex methods.

Different Triggers in Salesforce.

Triggers in Salesforce are two types, Before Triggers and After Triggers. Now will learn about basic syntax of a trigger in SFDC.



Salesforce trigger Syntax.

```
trigger triggerName
on ObjectName
```

```
trigger triggerName on ObjectName (trigger_events) {
   //code_block
}

trigger TestTrigger on Case (Before Insert,After Insert,Before Update ,After Update ,Before Delete ,After
Delete,After Undelete) {

 //Code Block

}
```

1. Apex trigger is always started with a keyword`trigger`.
2. Next we have to enter Trigger name.
3. Enter the condition.
4. To execute trigger on a case like before insert, after insert, before update, after update, before delete, after delete, after undelete, we must specify trigger events in a comma separated list as shown above.

We can declare more than one trigger event in one trigger ,but each should be separated by comma. The events we can specify in an Apex Trigger are as follows.

1. Before Insert.
2. Before Update.
3. Before Delete.
4. After Insert.
5. After Updat.
6. After Delete.
7. After Undelete.

There is a System defined class called Trigger which contains 12(twelve) implicit variable,which we can access at run time. All Trigger Context variables are prefixed with**"Trigger"**. Ex : (`trigger.isinsert`).

Below are the varibales with description.

1. **isExecuting :** It returns true, if the current apex code is trigger.
2. **isBefore :** It returns true, if the code inside trigger context is executed before the record is saved.
3. **isAfter :** It returns true, if the code inside trigger context is executed after the record is saved.
4. **isInsert** : It returns true, if the code inside trigger context is executed due to an insert operation.
5. **isUpdate :** It returns true, if the code inside trigger context is executed due to an update operation.
6. **isDelete :** It returns true, if the code inside trigger context is executed due to delete operation.
7. **isUnDelete :** It returns true, if the code inside trigger context is executed due to undelete operation. i,e when we recovered data from recycle bin .
8. **new :** It returns the new version of the object records. Suppose when we inserted/updated 10 records trigger.new will contain 1o records .
9. **newMap :** It returns a map of new version of sObject which contains an ID's as a *key* and the old versions of the sObject records as *value.* This map is only available in before update, after insert, and after update triggers.

10. **old :** It returns the old version of the object records.
11. **oldMap :** It returns a map of old version of sObject which contains an IDs as a key and the new versions of the sObject records as value. This map is available for only update and delete trigger.
12. **size :** It return the size of the manipulated record. It will return one if you will insert one record, It will return the size of the record you are inserting ,updating or deleting or undeleting.

Trigger Event	Trigger.New	Trigger.Old
Before Insert	Yes	No
Before Update	Yes	Yes
Before Delete	No	Yes
Before UnDelete	No	Yes
After Insert	Yes	No

Conclusion : In our next Salesforce tutorial, we will create our first Apex trigger in Salesforce and will learn how to call a class method from a trigger with an example.