Apex Trigger

Apex can be invoked by using triggers. Apex triggers enable you to perform custom actions
before or after changes to Salesforce records, such as insertions, updates, or deletions.

A trigger is Apex code that executes before or after the following types of operations:

- insert
- update
- delete
- merge
- upsert
- undelete

For example, you can have a trigger run before an object's records are inserted into the database, after records have been deleted, or even after a record is restored from the Recycle Bin.

You can define triggers for top-level standard objects that support triggers, such as a Contact or an Account, some standard child objects, such as a CaseComment, and custom objects. To define a trigger, from the object management settings for the object whose triggers you want to access, go to Triggers.

There are primarily two types of Apex Triggers:

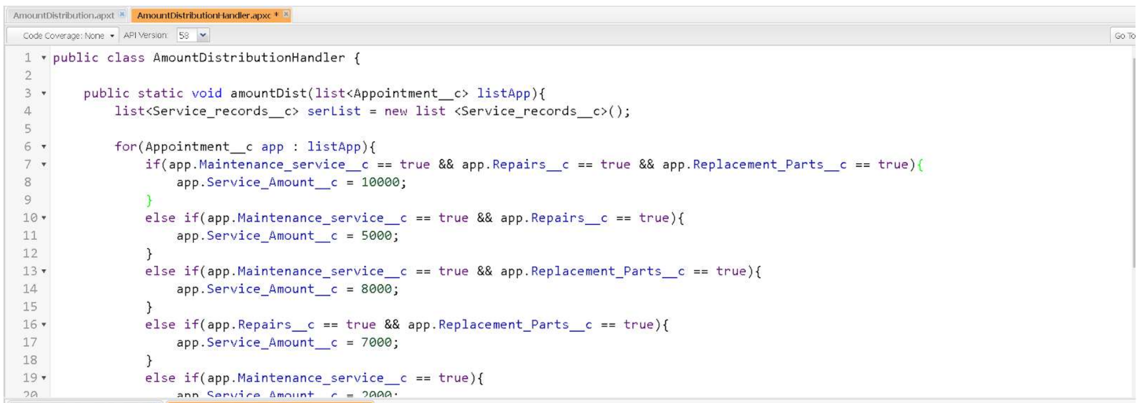
Before Trigger: This type of trigger in Salesforce is used either to update or validate the values of a record before they can be saved into the database. So, basically, the before trigger validates the record first and then saves it. Some criteria or code can be set to check data before it gets ready to be inserted into the database.

After Trigger: This type of trigger in Salesforce is used to access the field values set by the system and affect any change in the record. In other words, the after trigger makes changes to the value from the data inserted in some other record.

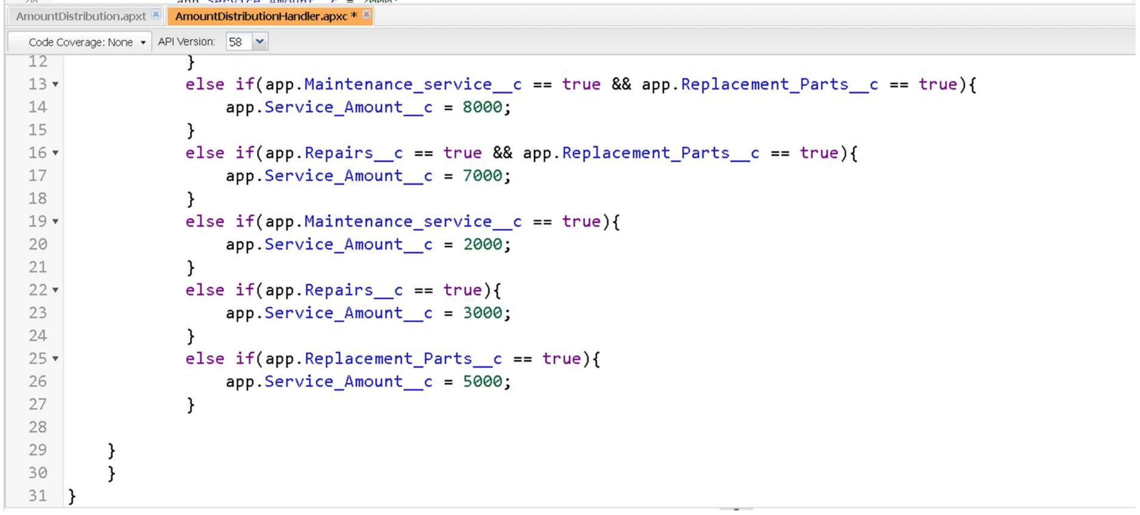
Apex handler

UseCase : This use case works for Amount Distribution for each Service the customer selected for there Vehicle.

1. Login to the respective trailhead account and navigate to the gear icon in the top right corner.
2. Click on the Developer console. Now you will see a new console window.
3. In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.
4. Name the class as “AmountDistributionHandler ”.



```
public class AmountDistributionHandler {

    public static void amountDist(list<Appointment__c> listApp){
        list<Service_records__c> serList = new list <Service_records__c>();

        for(Appointment__c app : listApp){
            if(app.Maintenance_service__c == true && app.Repairs__c == true && app.Replacement_Parts__c == true){
                app.Service_Amount__c = 10000;
            }
            else if(app.Maintenance_service__c == true && app.Repairs__c == true){
                app.Service_Amount__c = 5000;
            }
            else if(app.Maintenance_service__c == true && app.Replacement_Parts__c == true){
                app.Service_Amount__c = 8000;
            }
            else if(app.Repairs__c == true && app.Replacement_Parts__c == true){
                app.Service_Amount__c = 7000;
            }
            else if(app.Maintenance_service__c == true){
                app.Service_Amount__c = 2000;
```



```
            }
            else if(app.Maintenance_service__c == true && app.Replacement_Parts__c == true){
                app.Service_Amount__c = 8000;
            }
            else if(app.Repairs__c == true && app.Replacement_Parts__c == true){
                app.Service_Amount__c = 7000;
            }
            else if(app.Maintenance_service__c == true){
                app.Service_Amount__c = 2000;
            }
            else if(app.Repairs__c == true){
                app.Service_Amount__c = 3000;
            }
            else if(app.Replacement_Parts__c == true){
                app.Service_Amount__c = 5000;
            }

        }
    }
}
```

Code:

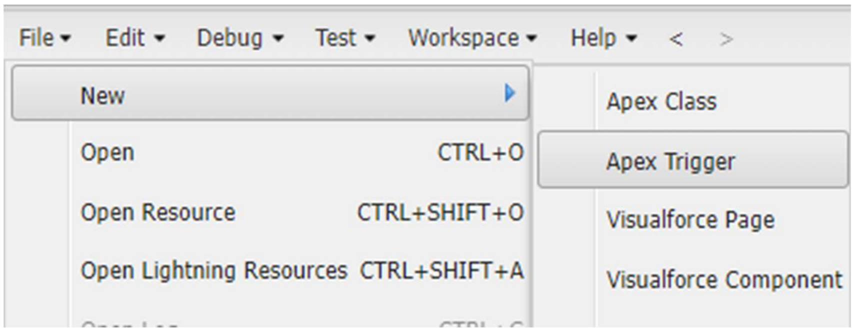
public class AmountDistributionHandler {

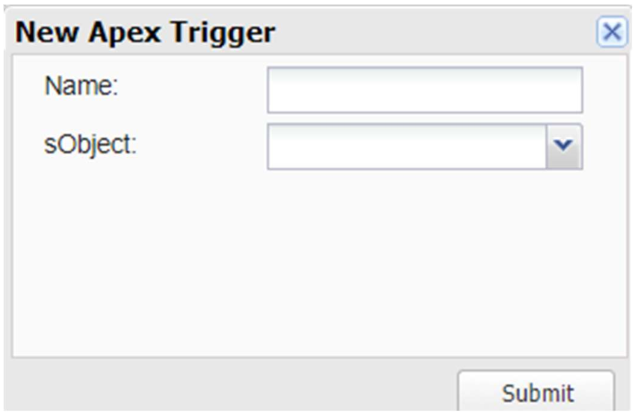
```
    public static void amountDist(list<Appointment__c> listApp){
        list<Service_records__c> serList = new list <Service_records__c>();
        for(Appointment__c app : listApp){
            if(app.Maintenance_service__c == true && app.Repairs__c == true
&& app.Replacement_Parts__c == true){
                app.Service_Amount__c = 10000;
            }
            else if(app.Maintenance_service__c == true && app.Repairs__c ==
true){
                app.Service_Amount__c = 5000;
            }
            else if(app.Maintenance_service__c == true &&
app.Replacement_Parts__c == true){
                app.Service_Amount__c = 8000;
            }
            else if(app.Repairs__c == true && app.Replacement_Parts__c ==
true){
                app.Service_Amount__c = 7000;
            }
            else if(app.Maintenance_service__c == true){
                app.Service_Amount__c = 2000;
            }
            else if(app.Repairs__c == true){
                app.Service_Amount__c = 3000;
            }
            else if(app.Replacement_Parts__c == true){
                app.Service_Amount__c = 5000;
            }

    }
    }
}
```

Trigger Handler :
How to create a new trigger :

1. While still in the trailhead account, navigate to the gear icon in the top right corner.
2. Click on developer console and you will be navigated to a new console window.
3. Click on File menu in the tool bar, and click on new? Trigger.
4. Enter the trigger name and the object to be triggered.
5. Name : AmountDistribution

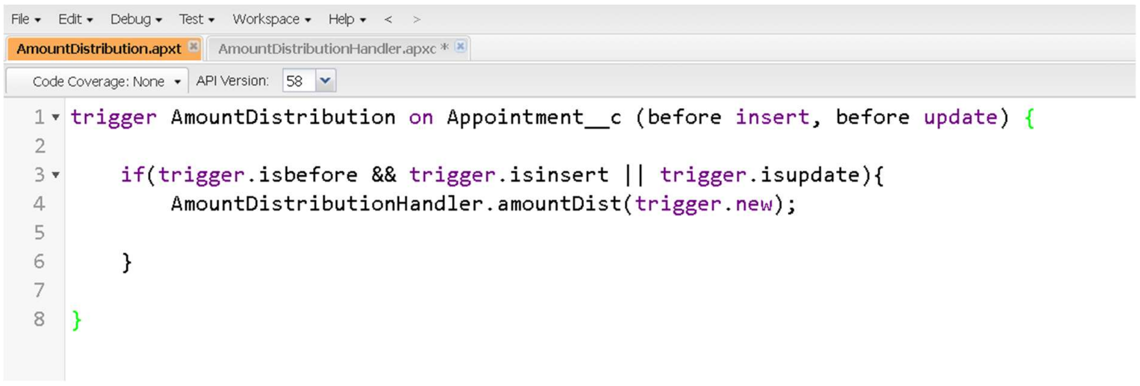
6. sObject : Appointment__c





Syntax For creating trigger :
The syntax for creating trigger is :
Trigger [trigger name] on [object name](Before/After event)
{
}
In this project , trigger is called whenever the particular records sum exceed the threshold i.e minimum business requirement value. Then the code in the trigger will get executed.

1. Handler for the Appointment Object



```
trigger AmountDistribution on Appointment__c (before insert, before update) {

    if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
        AmountDistributionHandler.amountDist(trigger.new);

    }

}
```

Code:

```
trigger AmountDistribution on Appointment__c (before insert, before
update) {

    if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
        AmountDistributionHandler.amountDist(trigger.new);

    }


}
```