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:

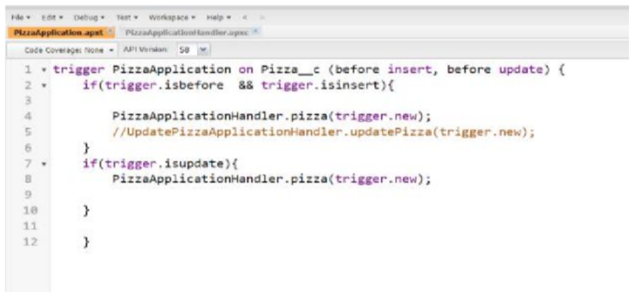
Before Trigger: This type of trigger in Salesforce is used either to update or validate the values of a record before they can be saved into the database. So, basically, the before trigger validates the record first and then saves it. Some criteria or code can be set to check data before it gets ready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

Activity- 1

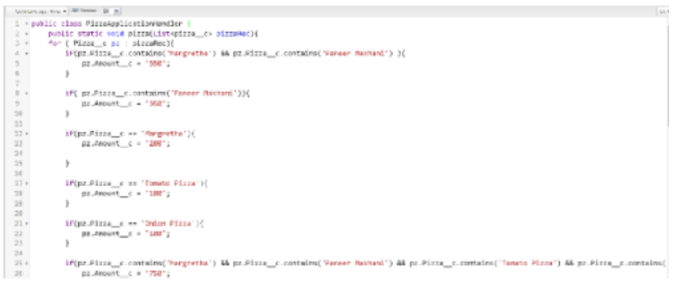
Use Case: This Trigger works for the Pizza Object where the scenario is like whenever the customer is selecting the Pizza whether it is veg Pizza or Non-veg Pizza According to the selection of Pizza The Amount will be reflected in the "Amount" Field.

Trigger:

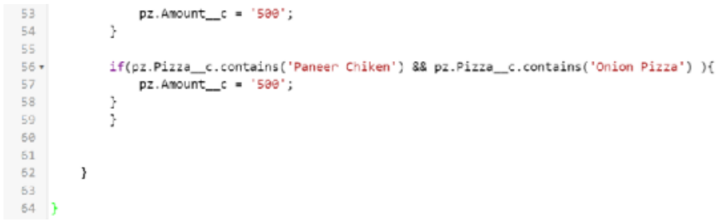


```
trigger PizzaApplication on Pizza__c (before insert, before update) {
    if(trigger.isbefore  && trigger.isinsert){

        PizzaApplicationHandler.pizza(trigger.new);
        //UpdatePizzaApplicationHandler.updatePizza(trigger.new);
    }
    if(trigger.isupdate){
        PizzaApplicationHandler.pizza(trigger.new);

    }

    }
```

Trigger Handler:



```
public class PizzaApplicationHandler {
    public static void pizza(List<pizza__c> pizzaRec){
    for ( Pizza__c pz : pizzaRec){
        if(pz.Pizza__c.contains('Margretha') && pz.Pizza__c.contains('Paneer Makhani') ){
            pz.Amount__c = '500';
        }

        if( pz.Pizza__c.contains('Paneer Makhani')){
            pz.Amount__c = '350';
        }

        if(pz.Pizza__c == 'Margretha'){
            pz.Amount__c = '200';

        }

        if(pz.Pizza__c == 'Tomato Pizza' ){
            pz.Amount__c = '100';
        }

        if(pz.Pizza__c == 'Onion Pizza' ){
            pz.Amount__c = '100';
        }

        if(pz.Pizza__c.contains('Margretha') && pz.Pizza__c.contains('Paneer Makhani') && pz.Pizza__c.contains('Tomato Pizza') && pz.Pizza__c.contains(
            pz.Amount__c = '750';
```



```
            pz.Amount__c = '500';
        }

        if(pz.Pizza__c.contains('Paneer Chiken') && pz.Pizza__c.contains('Onion Pizza') ){
            pz.Amount__c = '500';
        }
        }


    }

}
```

Trigger Code:

trigger PizzaApplication on Pizza__c (before insert, before update) {   if(trigger.isbefore && trigger.isinsert){

PizzaApplicationHandler.pizza(trigger.new);

//UpdatePizzaApplicationHandler.updatePizza(trigger.new);
}
if(trigger.isupdate){
PizzaApplicationHandler.pizza(trigger.new);
}
}

Trigger Handler

public class PizzaApplicationHandler.pizza {   public static void pizza(List<pizza__c> pizzaRec){   for (Pizza__c pz : pizzaRec){

```
         if(pz.Pizza__c.contains('Margretha') &&
 pz.Pizza__c.contains('Paneer Makhani') ){
     pz.Amount__c = '550';
}
 if( pz.Pizza__c.contains('Paneer
 Makhani')){ pz.Amount__c = '350';
  }
  if(pz.Pizza__c ==
 'Margretha'){  pz.Amount__c = '200';
  }
 if(pz.Pizza__c == 'Tomato Pizza'){
          pz.Amount__c = '100';
      }  if(pz.Pizza__c ==
 'Onion Pizza'){
          pz.Amount__c =
 '100';       }

  if(pz.Pizza__c.contains('Margretha') &&
 pz.Pizza__c.contains('Paneer
 Makhani') && p z.Pizza__c.contains('Tomato
 Pizza') && pz.Pizza__c.contains('Onion Pizza')
  ){  pz.Amount__c = '750';
   }
 if(pz.Pizza__c.contains('Margretha') &&
 pz.Pizza__c.contains('Paneer
 Makhani') && p z.Pizza__c.contains('Tomato Pizza'))
 pz.Amount__c = '750';
  }
 if(pz.Pizza__c == 'Chicken
 Pizza'){ pz.Amount__c = '400';
  }
 if(pz.Pizza__c == 'Paneer
 Chicken'){  pz.Amount__c = '400';
}
```

```
if(pz.Pizza__c.contains('Paneer
Chicken')                                      &&
pz.Pizza__c.contains('Chicken
Pizza') ){ pz.Amount__c = '800';
} if(pz.Pizza__c.contains('Paneer
          Chicken') &&
pz.Pizza__c.contains('Paneer Makhani')
 ){ pz.Amount__c = '750';
 }
    if(pz.Pizza__c.contains('Paneer
Chicken') &&
pz.Pizza__c.contains('Margretha')
 ){            pz.Amount__c = '750';
      }
if(pz.Pizza__c.contains('Paneer Chicken') &&
pz.Pizza__c.contains('Tomato Pizza') ){
             pz.Amount__c = '500';
}
if(pz.Pizza__c.contains('Paneer              Chicken')              &&
pz.Pizza__c.contains('Onion Pizza') ){            pz.Amount__c =
'500';
}
}
}
}
```

Schedule Apex

Scheduled Apex in Salesforce is a feature that allows you to schedule the execution of Apex classes to run at specified times. This capability is useful for automating repetitive tasks, such as data updates or calculations, on a regular basis To create a scheduled Apex job, you need to define an Apex class that implements the Schedulable interface. This interface requires you to implement a single method called execute, which contains the logic you want to run at the scheduled time

```
public class MyScheduledClass implements
Schedulable { public void
execute(SchedulableContext context) {
}
}
```

To schedule the execution of this class, you can use the Salesforce Developer Console or the Salesforce user interface. Here's an example of scheduling this class to run every day at 2 PM

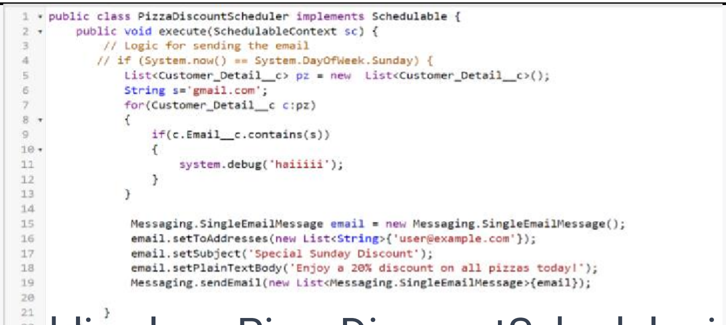
```
String cronExp = '0 0 14 * * ?'; // Cron expression for 2 PM
every day

System.schedule('My Scheduled Job', cronExp, new
MyScheduledClass());
```

In the example above, System.schedule is used to create a new scheduled job. The first parameter is the name of the job, the second parameter is the cron expression that defines the schedule, and the third parameter is the instance of the Apex class that should be executed at the scheduled time.

Note that the cron expression follows a specific syntax that allows you to define complex schedules with minute, hour, day, month, and day-of-week precision

Once the scheduled Apex job is created, it will run automatically according to the specified schedule. You can monitor the execution and view any debug logs generated by the scheduled job in the
Salesforce user interface

Schedule Apex For Frequently Visited Customer:



```
public class PizzaDiscountScheduler implements Schedulable {
    public void execute(SchedulableContext sc) {
        // Logic for sending the email
       // if (System.now() == System.DayOfWeek.Sunday) {
            List<Customer_Detail__c> pz = new  List<Customer_Detail__c>();
            String s='gmail.com';
            for(Customer_Detail__c c:pz)
            {
                if(c.Email__c.contains(s))
                {
                    system.debug('haiiiii');
                }
            }

             Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
             email.setToAddresses(new List<String>{'user@example.com'});
             email.setSubject('Special Sunday Discount');
             email.setPlainTextBody('Enjoy a 20% discount on all pizzas today!');
             Messaging.sendEmail(new List<Messaging.SingleEmailMessage>{email});

        }
```

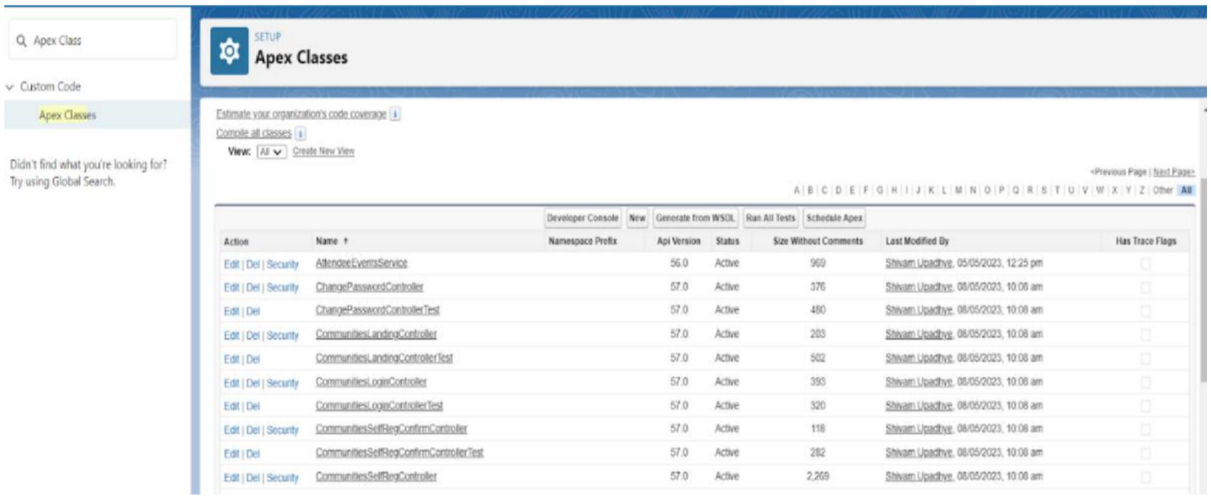
```
public class PizzaDiscountScheduler implements
Schedulable {     public void
execute(SchedulableContext sc) {
      // Logic for sending the email
     // if (System.now() == System.DayOfWeek.Sunday)
{
 List<Customer_Detail__c> pz = String s='gmail.com';
for(Customer_Detail__c c:pz)
         {
            if(c.Email__c.contains(s))
            {
               system.debug('haiiiii');
            }
         }
          Messaging.SingleEmailMessage   email        =
new
Messaging.SingleEmailMessage();
email.setToAddresses(new
List<String>{'user@example.com'});
        email.setSubject('Special      Sunday      Discount');
        email.setPlainTextBody('Enjoy  a  20%  discount  on
        all
pizzas today!');
                  Messaging.sendEmail(new
List<Messaging.SingleEmailMessage>{email});
                   } }
```

Activity-2

For Making the Schedule to send Mail To the Customer
Follow the steps below:

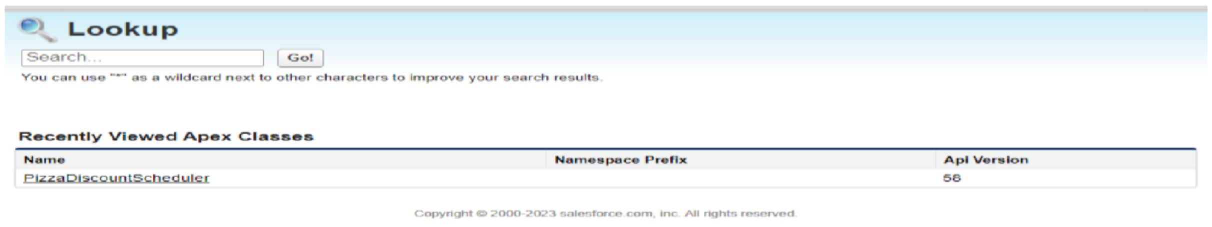
1.Click on the **Gear Icon** **>>** Go to the **Home Tab** >> In the Quick Find Box >> Search for **Apex Clas**s

2. Click on the **Schedule Apex** >> Give Job Name As >> **Schedule Pizza**.


Apex Class
Custom Code
Apex Classes
Didn't find what you're looking for? Try using Global Search.
SETUP
Apex Classes
Estimate your organization's code coverage
Compile all classes
View: All
Create New View
<Previous Page | Next Page>
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All
Developer Console
New
Generate from WSDL
Run All Tests
Schedule Apex

Action	Name ↑	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit | Del | Security	AttendeeEventsService		56.0	Active	969	Shivam Upadhye, 05/05/2023, 12:25 pm	☐
Edit | Del | Security	ChangePasswordController		57.0	Active	376	Shivam Upadhye, 08/05/2023, 10:08 am	☐
Edit | Del	ChangePasswordControllerTest		57.0	Active	480	Shivam Upadhye, 08/05/2023, 10:08 am	☐
Edit | Del | Security	CommunitiesLandingController		57.0	Active	203	Shivam Upadhye, 08/05/2023, 10:08 am	☐
Edit | Del	CommunitiesLandingControllerTest		57.0	Active	502	Shivam Upadhye, 08/05/2023, 10:08 am	☐
Edit | Del | Security	CommunitiesLoginController		57.0	Active	393	Shivam Upadhye, 08/05/2023, 10:08 am	☐
Edit | Del	CommunitiesLoginControllerTest		57.0	Active	320	Shivam Upadhye, 08/05/2023, 10:08 am	☐
Edit | Del | Security	CommunitiesSelfRegConfirmController		57.0	Active	118	Shivam Upadhye, 08/05/2023, 10:08 am	☐
Edit | Del	CommunitiesSelfRegConfirmControllerTest		57.0	Active	282	Shivam Upadhye, 08/05/2023, 10:08 am	☐
Edit | Del | Security	CommunitiesSelfRegController		57.0	Active	2,269	Shivam Upadhye, 08/05/2023, 10:08 am	☐

3.Click on **Apex Class Lookup** **>>** Select **PizzaDiscountScheduler**
In Recently Viewed Apex Class



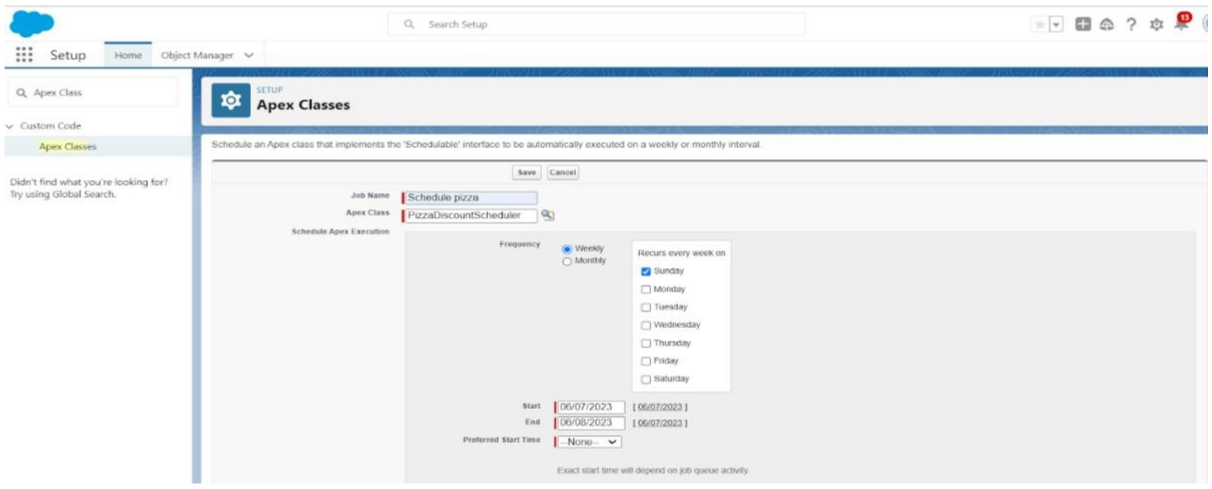
You can use "*" as a wildcard next to other characters to improve your search results.

Recently Viewed Apex Classes

Name	Namespace Prefix	Api Version
PizzaDiscountScheduler		58

Copyright © 2000-2023 salesforce.com, inc. All rights reserved.

4. In Frequency Click on the **Weekly Radio button You can Select the Start Date and Enddate As Per You Requirement and then Click on the Save Button.**



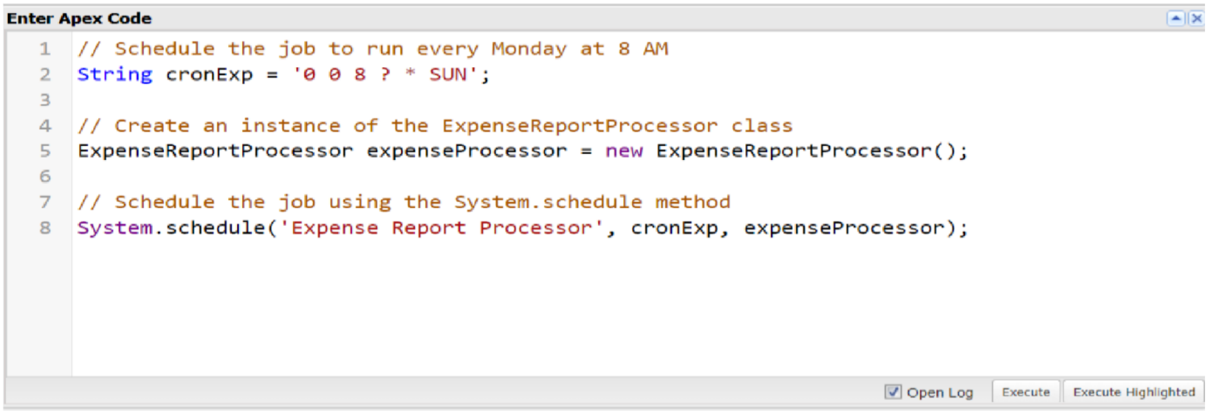
Schedule an Apex class that implements the 'Schedulable' interface to be automatically executed on a weekly or monthly interval.

Job Name	Schedule pizza
Apex Class	PizzaDiscountScheduler
Schedule Apex Execution	
Frequency	[✓ Weekly] Monthly
Recurs every week on	[✓ Sunday] Monday Tuesday Wednesday Thursday Friday Saturday
Start	06/07/2023 [06/07/2023]
End	06/08/2023 [06/07/2023]
Preferred Start Time	--None--

Exact start time will depend on job queue activity.

Alternate Option

6. Click on the **debug** besides file >> Click on the **Open Execute Anonymous Windows** Execute the below code .



```
// Schedule the job to run every Monday at 8 AM
String cronExp = '0 0 8 ? * SUN';

// Create an instance of the ExpenseReportProcessor class
ExpenseReportProcessor expenseProcessor = new ExpenseReportProcessor();

// Schedule the job using the System.schedule method
System.schedule('Expense Report Processor', cronExp, expenseProcessor);
```

// Schedule the job to run every Monday at 8 AM
String cronExp = '0 0 8 ? * SUN';

// Create an instance of the ExpenseReportProcessor class
ExpenseReportProcessor expenseProcessor = new
ExpenseReportProcessor();

// Schedule the job using the System.schedule method
System.schedule('Expense Report Processor', cronExp,
expenseProcessor);