APEX - BATCH PROCESSING

Consider a scenario where you would like to process large number of records on daily basis, probably the cleaning of data or maybe deleting some unused data.

What is Batch Apex?

Batch Apex is asynchronous execution of Apex code, specially designed for processing the large number of records and has greater flexibility in governor limits than the synchronous code.

When to use Batch Apex?

- When you want to process large number of records on daily basis or even on specific time of interval then you could go for Batch Apex.
- Also, when you want an operation to be asynchronous then you could implement the Batch Apex. Batch Apex is exposed as an interface that must be implemented by the developer. Batch jobs can be programmatically invoked at runtime using Apex. Batch Apex operates over small batches of records, covering your entire record set and breaking the processing down to manageable chunks of data.

Using Batch Apex

When we are using the Batch Apex, we must implement the Salesforce-provided interface Database.Batchable, and then invoke the class programmatically.

You could monitor the class by following below steps:

To monitor or stop the execution of the batch Apex Batch job, go to Setup->Monitoring->Apex Jobs or Jobs-> Apex Jobs.

Database.Batchable interface has three methods which we must implement as given below:

- Start
- Execute
- Finish

Let's have look at each one of them.

Start:

Syntax:

```
global (Database.QueryLocator | Iterable) start(Database.BatchableContext bc) {}
```

This method will be called at the starting of the Batch Job and collects the data on which the Batch job will be operating.

Note the following points:

- Use the Database.QueryLocator object when you are using a simple query to generate the scope of objects used in the batch job. In this case, the SOQL data row limit will be bypassed.
- Use iterable object when you have complex criteria to process the records. Database.QueryLocator determines the scope of records which should be processed.

Execute:

Syntax:

```
global void execute(Database.BatchableContext BC, list<sobject<){}
```

where , list<sObject< is returned by the Database.QueryLocator method.

This method gets called after the Start method and does all the processing required for Batch Job.

Finish:

Syntax:

```
global void finish(Database.BatchableContext BC){}
```

This method gets called at the end and you could do some finishing activities like sending an email with information about the batch job records processed and status.

Batch Apex Example

Let's take an example of our existing Chemical Company and let's assume that we have requirement to update the Customer Status and Customer Description field of Customer Records which have been marked as Active and which have created Date as today. This should be done on daily basis and an email should be sent to a User about the status of the Batch Processing. Update the Customer Status as 'Processed' and Customer Description as 'Updated Via Batch Job'.

```
//Batch Job for Processing the Records
global class CustomerProessingBatch implements Database.Batchable<sobject>{
  global String [] email = new String[] {'test@test.com'};//Add here your email address
here

  //Start Method
  global Database.Querylocator start (Database.BatchableContext BC) {
    return Database.getQueryLocator('Select id, Name, APEX_Customer_Status__c,
APEX_Customer_Decscription__c From APEX_Customer__c WHERE createdDate = today AND
APEX_Active__c = true');//Query which will be determine the scope of Records fetching
the same
  }

  //Execute method
  global void execute (Database.BatchableContext BC, List<sobject> scope) {
    List<apex_customer__c> customerList = new List<apex_customer__c>();
    List<apex_customer__c> updtaedCustomerList = new List<apex_customer__c>();//List to
hold updated customer
    for (sObject objScope: scope) {
        APEX_Customer__c newObjScope = (APEX_Customer__c)objScope ;//type casting from
generic sOject to APEX_Customer__c
        newObjScope.APEX_Customer_Decscription__c = 'Updated Via Batch Job';
        newObjScope.APEX_Customer_Status__c = 'Processed';
        updtaedCustomerList.add(newObjScope);//Add records to the List
```

```
        System.debug('Value of UpdatedCustomerList '+updtaedCustomerList);
    }
        if (updtaedCustomerList != null && updtaedCustomerList.size()>0) {//Check if List
is empty or not
            Database.update(updtaedCustomerList); System.debug('List Size
'+updtaedCustomerList.size());//Update the Records
        }
  }

  //Finish Method
  global void finish(Database.BatchableContext BC){
  Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

  //Below code will fetch the job Id
  AsyncApexJob a = [Select a.TotalJobItems, a.Status, a.NumberOfErrors, a.JobType,
a.JobItemsProcessed, a.ExtendedStatus, a.CreatedById, a.CompletedDate From AsyncApexJob a
WHERE id = :BC.getJobId()];//get the job Id
  System.debug('$$$ Jobid is'+BC.getJobId());

  //below code will send an email to User about the status
  mail.setToAddresses(email);
  mail.setReplyTo('test@test.com');//Add here your email address
  mail.setSenderDisplayName('Apex Batch Processing Module');
  mail.setSubject('Batch Processing '+a.Status);
  mail.setPlainTextBody('The Batch Apex job processed  '+a.TotalJobItems+'batches with
'+a.NumberOfErrors+'failures'+'Job Item processed are'+a.JobItemsProcessed);
  Messaging.sendEmail(new Messaging.Singleemailmessage [] {mail});
  }
}
```

To execute this code, first save it and then paste the below code in execute anonymous. This will create the object of class and Database.execute method will execute the Batch job. Once the job is completed then an email will be sent on the specified email address. Make sure that you have a customer record which has Active as checked.

```
//Paste in Developer Console
CustomerProessingBatch objClass = new CustomerProessingBatch();
Database.executeBatch (objClass);
```

Once this class is executed, then check the email address you have provided where you will receive the email with information. Also, you could check the status of the batch job via the Monitoring page and steps as provided above.

If you check the debug logs then you could find the List size which indicates how many records have been processed.

Limitations

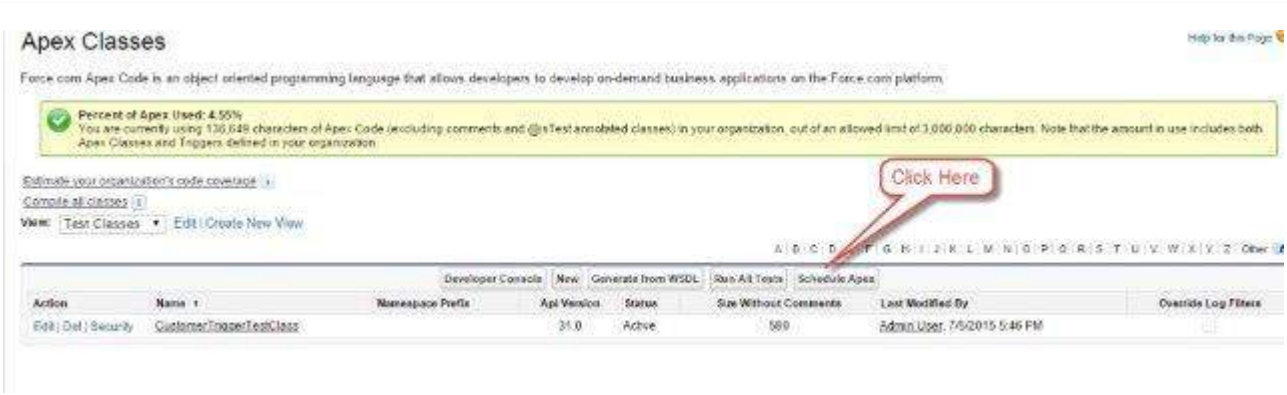
We could only have 5 batch job processing at a time. This is one of the limitations of Batch Apex.

Scheduling the Apex Batch Job using Apex Detail Page

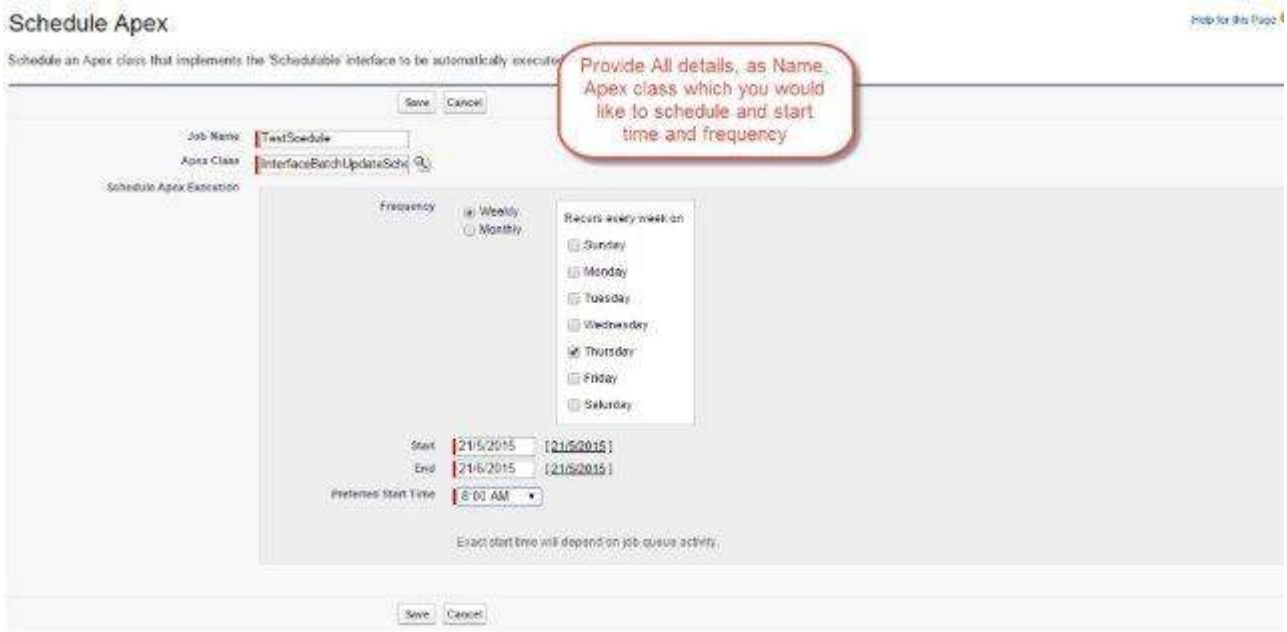
You could schedule the Apex class via Apex detail page as well as give below:

Step 1: Go to Setup=>Apex Classes, Click on Apex Classes.

Step 2: Click on Schedule Apex button:



Step 3: Provide details:



Step 4: Then Click on Save button and your class will be scheduled.

Scheduling the Apex Batch Job using Schedulable Interface

```
//Batch Job for Processing the Records
global class CustomerProessingBatch implements Database.Batchable<sobject>{
  global String [] email = new String[] {'test@test.com'};//Add here your email address
here

  //Start Method
  global Database.Querylocator start (Database.BatchableContext BC) {
    return Database.getQueryLocator('Select id, Name, APEX_Customer_Status__c,
APEX_Customer_Decscription__c From APEX_Customer__c WHERE createdDate = today AND
APEX_Active__c = true');//Query which will be determine the scope of Records fetching
the same
  }

  //Execute method
  global void execute (Database.BatchableContext BC, List<sobject> scope) {
    List<apex_customer__c> customerList = new List<apex_customer__c>();
    List<apex_customer__c> updtaedCustomerList = new List<apex_customer__c>();//List to
hold updated customer
    for (sObject objScope: scope) {
        APEX_Customer__c newObjScope = (APEX_Customer__c)objScope ;//type casting from
generic sOject to APEX_Customer__c
        newObjScope.APEX_Customer_Decscription__c = 'Updated Via Batch Job';
        newObjScope.APEX_Customer_Status__c = 'Processed';
        updtaedCustomerList.add(newObjScope);//Add records to the List
        System.debug('Value of UpdatedCustomerList '+updtaedCustomerList);
    }
```

```
    if (updtaedCustomerList != null && updtaedCustomerList.size()>0) {//Check if List is
empty or not
        Database.update(updtaedCustomerList); System.debug('List Size
'+updtaedCustomerList.size());//Update the Records
    }
  }

  //Finish Method
  global void finish(Database.BatchableContext BC){
      Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

      //Below code will fetch the job Id
      AsyncApexJob a = [Select a.TotalJobItems, a.Status, a.NumberOfErrors, a.JobType,
a.JobItemsProcessed, a.ExtendedStatus, a.CreatedById, a.CompletedDate From AsyncApexJob a
WHERE id = :BC.getJobId()];//get the job Id
      System.debug('$$$ Jobid is'+BC.getJobId());

      //below code will send an email to User about the status
      mail.setToAddresses(email);
      mail.setReplyTo('test@test.com');//Add here your email address
      mail.setSenderDisplayName('Apex Batch Processing Module');
      mail.setSubject('Batch Processing '+a.Status);
      mail.setPlainTextBody('The Batch Apex job processed  '+a.TotalJobItems+'batches with
'+a.NumberOfErrors+'failures'+'Job Item processed are'+a.JobItemsProcessed);

      Messaging.sendEmail(new Messaging.Singleemailmessage [] {mail});
  }

  //Scheduler Method to scedule the class
  global void execute(SchedulableContext sc)
    {
        CustomerProessingBatch conInstance = new CustomerProessingBatch();
        database.executebatch(conInstance,100);
    }
}

//Paste in Developer Console
CustomerProessingBatch objClass = new CustomerProessingBatch();
Database.executeBatch (objClass);
```