Salesforce GROW

Module : Batch Apex

Batch Apex

Batch Apex

Batch Apex operates over small batches of records, covering entire record set and breaking the processing down to manageable chunks.

Batch Apex allows us to define a job that can be divided into manageable chunks, where each chunk can be proceed separately. For example, if we want to make an field update of all records in any object which is having more number of records, then governor limits restricts us to process that operation. Because, In a single transaction we can only process 10,000 records.

Batch jobs can be programmatically scheduled to run at specific times using the Apex scheduler, or scheduled using the Schedule Apex page in the Salesforce user interface.

We can evaluate current count by viewing the Scheduled Jobs page in Salesforce or programmatically using SOAP API to query the AsyncApexJob object.

Batchable Interface

To use batch Apex, Apex class must implements the Salesforce-provided interface ”Database.Batchabl”e and then invoke the class programmatically..

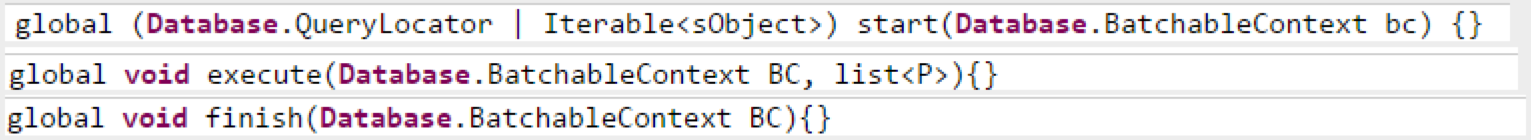
The Database.Batchable interface contains three methods that must be implemented:

1. Start
2. Execute
3. Stop

Start method is automatically called at the beginning of the apex job. This method will collect record or objects on which the operation should be performed. These record are divided into subtasks & passes those to execute method.

Execute method performs operation on the records fetched from start method.

Finish method executes after all batches are processed. Use this method to send confirmation email notifications.



```
global (Database.QueryLocator | Iterable<sObject>) start(Database.BatchableContext bc) {}
global void execute(Database.BatchableContext BC, list<P>){}
global void finish(Database.BatchableContext BC){}
```

Important Database Interfaces and Methods for Batch Apex

Database.BatchableContext

All the methods in the Database. Batchable interface require a reference to Database.BatchableContext object. Use this object to track the progress of the batch job.

Database.QueryLocator

The start method can return either a Database.QueryLocator object that contains the records to use in the batch job or an iterable.

Database.executeBatch

This method can use to Submit Batch Jobs.

Iterable in Batch Apex

The start method can return either a Database.QueryLocator object that contains the records to use in the batch job or an iterable.

System.scheduleBatch

This method can use to schedule a batch job to run once at a future time.

Database.AllowsCallouts

To use a callout in batch Apex, specify Database.AllowsCallouts in the class definition.

Submit, Schedule and Monitoring Batch Job

Submit Batch Job

ID batchprocessid = Database.executeBatch(<instance of a class> , <Optional: scope>);

Where:
Parameter1: An instance of a class that implements the Database.Batchable interface.
Parameter2: An optional parameter scope. This parameter specifies the number of records to pass into the execute method.

Schedule Batch Job

String cronID = System.scheduleBatch(**<instance of a class>**, 'job example', 1, **<Optional:scope>**);

Where:
Parameter1: An instance of a class that implements the Database.Batchable interface.
Parameter2: The job name.
Parameter3: The time interval, in minutes, after which the job starts executing.
Parameter4: An optional parameter *scope*. This parameter specifies the number of records to pass into the execute method.

Monitoring Batch Job

Check the status of job from **"Setup > Monitor > Jobs > Apex Job"**

Batch Apex Example and Steps

Create a apex class "**BatchDeleteLeads**" and implements "Database.Batchable" interface

Define below methods:

- "**start**" method into class with "global" access modifier. This method will collect the records, which are divided into subtasks & passes those to "execute" method.
- "**execute**" method into class with "global" access modifier. This method will performs operation on the records fetched from start method.
- "**end**" method into class with "global" access modifier. This method will executes after all batches are processed.

Execute below "B" Ap B code using Developer Console:

A

```
global class BatchDeleteLeads implements Database.Batchable<sObject> {
        public String query;
        global Database.QueryLocator start(Database.BatchableContext BC){
                return Database.getQueryLocator(query);
        }
        global void execute(Database.BatchableContext BC, List<sObject> scope){
                delete scope;
                DataBase.emptyRecycleBin(scope);
        }
        global void finish(Database.BatchableContext BC){
                //Do nothing
        }
}
```

B

```
BatchDeleteLeads batchDel= new BatchDeleteLeads();
// Query for selecting the Leads to delete
batchDel.query = 'SELECT Id FROM Lead';
// Invoke the batch job.
ID batchprocessid = Database.executeBatch(batchDel);
System.debug('Return Batch process ID: ' +
batchProcessId);
```

- Check the status of job from "**Setup** > **Monitor** > **Jobs** > **Apex Job**".

Batch Apex Governor Limits

Up to 5 batch jobs can be queued or active concurrently.

Up to 100 Holding batch jobs can be held in the Apex flex queue.

In a running test, you can submit a maximum of 5 batch jobs.

The batch Apex start method can have up to 15 query cursors open at a time per user.

A maximum of 50 million records can be returned in the Database.QueryLocator object. If more than 50 million records are returned, the batch job is immediately terminated and marked as Failed.

If the start method of the batch class returns a QueryLocator, the optional scope parameter of Database.executeBatch can have a maximum value of 2,000. If set to a higher value, Salesforce chunks the records returned by the QueryLocator into smaller batches of up to 2,000 records.

If no size is specified with the optional scope parameter of Database.executeBatch, Salesforce chunks the records returned by the start method into batches of 200.

The start, execute, and finish methods can implement up to 10 callouts each.

Only one batch Apex job's start method can run at a time in an organization. Batch jobs that haven’t started yet remain in the queue until they're started.

For more limits, please refer link “https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_batch_interface.htm”

Hands-On Exercise

Batch Apex Examples
Batch Apex Governor Limits
Batch Apex Best Practices

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_batch_interface.htm