DML and Callouts in Sequence

POSTED ON MAY 22, 2012

Today I wanted to discuss a use-case which most of the developers might have faced when integrating Salesforce to other systems.

Here is the Scenario –

- A Notification is received in Salesforce to pick up the Accounts from the Service deployed in another Integrating System.
- Callout from Salesforce to get the new Account Records.
- Insert the new Account Records that came on the Response
- Send the new Account Ids that are created to the Integrating System (Another callout).

In an ideal development, we write a After Insert Trigger on a Notification to perform a Callout, Save the Accounts and do the Callout Again.

But here we have couple of complications

- We cannot do a Synchronous Callout from a Trigger
- We cannot do a DML operation before a callout

Since Salesforce does not have a explicit Commit, if you try doing DML and then Callout, you will get '**You have uncommitted work pending. Please commit or rollback before calling out".** Its so painful because the error doesn't really give you anything,

I had to spend couple of hours to do the workaround and here is what I came up with.

Create a Schedulable Class & Batch Job, something similar to below –

```
global with sharing class ScheduleAccountService implements Schedulable{

    private String query;

    global ScheduleAccountService(String queryString){
        this.query = queryString;
    }

    global void execute(SchedulableContext sc){

      BatchAccountService bns = new BatchAccountService(query);
          Database.executeBatch(bns,1000);

    }

}
```

Batch Job stub can be similar to below

```
global with sharing class BatchAccountService implements Database.Batchable,Databa

    private String query;

    global BatchAccountService(String queryString){

        if(queryString != null)
            this.query = queryString;

    }

    global Database.Querylocator start(Database.BatchableContext BC){
        return Database.getQueryLocator(query);
    }

    global void execute(Database.BatchableContext BC, List scope){

        /* Do the Callout to send the new Account Records.*/

    }

    global void finish(Database.BatchableContext BC){

        AsyncApexJob a = [Select Id, Status, NumberOfErrors, JobItemsProcessed,Tot

        system.debug(' Job Processing finished with ' + a.TotalJobItems + ' batche

    }
```

1. Call the future method from the Trigger, that does the callout to get the Account Data
2. Save the Account Records into Database.
3. Execute the Schedulable Job to run after 5 seconds. (this job executes the Batch job and does the Callout)

The code will work if its only one callout from Batch job today, but with the new Summer '12, since the callout limits are increased, batch jobs can do upto 10 Callouts. (Is really useful if an intermediate Authentication is required.)