(b) Account, and Opportunity/
Account with opportunity/
Separate opp in two parts

{ OppIdWAcctSet : loop through Acct Records, if owner of opp is same as acct owner, that is 'the userId' put in this set. If owner is different in acct and opp, which means opp owner is not 'the userId,' don't care about these records.

OppIdWoAcctSet: loop through opp Records, comparing with oppIdWAcctSet, if oppid not in that set, put it in this set.

Separate opp in two parts, due to if Acct has opps and opps owner is the same as account owner, after update the Account owner in batch, in batchClass; finish method, update these opps, set the owner same as account owner.

for Acct: BatchWorkloadUpdate batch = new BatchWorkloadUpdate
(userRec.Id, acctIdsSet, oppIdWAcctSet, query, 'Acct')

opp BatchWorkloadUpdate batch = new BatchWorkloadUpdate
(userRec.Id, oppIdWoAcctSet, '', 'opportunity')

BatchWorkloadUpdate:

In finish method. update opps whose owner id is the same as associated account id.

Visualforce Page: UserWorkloadReassign.page

checkBatchStatus

↳ only set pollerBool = false when batch job status = completed

Add counter < objTypeCt
objTypeCtInBatch

in constructor, counter to total type of obj
if lead + Acct = 2
if lead + Acct + opp = 3

objTypeCtInBatch in execute

these two counter, is for checkBatchStatus, previously, if lead Batch is finished, in UI, complete will shown. This way, it will wait for all Batch complete, then complete shown

In BatchWorkloadUpdate $\overset{\text{class}}{\frown}$ add stateful, in order to send email regarding exception

If not using stateful, exception will be {} empty

user.

Best Answer chosen by safal soni

Amit Chaudhary 8

Please check below post for Batch job

1) http://amitsalesforce.blogspot.com/2016/02/batch-apex-in-salesforce-test-class-for.html

Batch Apex

A Batch class allows you to define a single job that can be broken up into manageable chunks that will be processed separately.

When to use Batch Apex

One example is if you need to make a field update to every Account in your organization. If you have 10,001 Account records in your org, this is impossible without some way of breaking it up. So in the start() method, you define the query you're going to use in this batch context: 'select Id from Account'. Then the execute() method runs, but only receives a relatively short list of records (default 200). Within the execute(), everything runs in its own transactional context, which means almost all of the governor limits only apply to that block. Thus each time execute() is run, you are allowed 150 queries and 50,000 DML rows and so on. When that execute() is complete, a new one is instantiated with the next group of 200 Accounts, with a brand new set of governor limits. Finally the finish() method wraps up any loose ends as necessary, like sending a status email.

Sample Batch Apex

1) **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.

2) **Execute Method** performs operation which we want to perform on the records fetched from start method.

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

NOTE:- All Three are required method. You can not remove any one method as it is a interface

Ravi Kumar 259

Batch apex class ,in finish method can we call the dml operations?

May 14, 2015 Reply Like 0

Manojjena

Hi Ravi ,

Yes you can do the DML in finish method. But only thing is that it will execute only when your batch process complete.It is just like house keeping .It will execute once .

May 14 2015 0 0

▲

6

▼

✓

Yes, you can certainly send an email to the user; in fact, this is my preferred method to avoid spamming users. You can do this by way of the Database.Stateful.

```
public class MyBatchable implements Database.Batchable<SObject>, Database.Stateful {
    Exception[] errors = new Exception[0];
    public Iterable<SObject> start(...) {
        ...
    }
    public void execute(...) {
        try {
            ...
        } catch(Exception e) {
            errors.add(e);
        }
    }
    public void finish(...) {
        if(!errors.isEmpty()) {
            Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
            mail.setSubject('Errors occurred during batch process.');
            mail.setTargetObjectId(UserInfo.getUserId());
            mail.setSaveAsActivity(false);
            mail.setPlainTextBody(buildBodyFor(errors));
            Messaging.sendEmail(new Messaging.Email[] { mail });
        }
    }
}
```

You might also include a way to send emails if too many errors have stacked up, or limit yourself to some number of exceptions (say, the first 50 or 100). Keep in mind that you do have a limited amount of memory available, so if you're concerned about having a large number of errors, consider logging them to the database and then querying them back in your finish method.

7

Some things to do:

1. Have your batch class implement Database.stateful
2. Declare some variables that are initialized in the constructor to 0. These variables count successes and errors; also a string variable that remembers the records that failed and why (initialized to empty string).
3. Use `Database.update` with `allOrNothing = false` instead of `update` within `execute()`. Interrogate each member of `Database.SaveResult[]` for `isSuccess()` and count succcesses and failures in your stateful variables from #2. Log in the stateful variable all the errors (id of record, name of record, and error message/exception)
4. In the `finish` method, send an email to the sysad of count of successes/failures + string variable of all the failures.
5. In `finish()` method, write your batch results to a custom `Log__c` record(s)

share improve this answer

answered Dec 20 '15 at 18:27
cropredy
30.2k 3 31 97

1 Decent answer but two additions: query the job Id in the finish method to check for blown governor