}

Asynchronous Apex

Use Batch Apex

LeadProcessor Code:

```
global class LeadProcessor implements
Database.Batchable<sObject>, Database.Stateful {

// instance member to retain state across transactions
global Integer recordsProcessed = 0;

global Database.QueryLocator start(Database.BatchableContext bc) {
return Database.getQueryLocator('SELECT Id, LeadSource FROM Lead');
}

global void execute(Database.BatchableContext bc, List<Lead> scope){
// process each batch of records
List<Lead> leads = new List<Lead>();
for (Lead lead : scope) {

lead.LeadSource = 'Dreamforce';
// increment the instance member counter
recordsProcessed = recordsProcessed + 1;

}
update leads;
}

global void finish(Database.BatchableContext bc){
System.debug(recordsProcessed + ' records processed. Shazam!');

}
}
```

LeadProcessorTest:

```
@isTest
public class LeadProcessorTest {
@testSetup
static void setup() {
List<Lead> leads = new List<Lead>();
// insert 200 leads
for (Integer i=0;i<200;i++) {
leads.add(new Lead(LastName='Lead '+i,
Company='Lead', Status='Open - Not Contacted'));
}
insert leads;
}

static testmethod void test() {
```

```
Test.startTest();
LeadProcessor lp = new LeadProcessor();
Id batchId = Database.executeBatch(lp, 200);
Test.stopTest();

// after the testing stops, assert records were updated properly
System.assertEquals(200, [select count() from lead where LeadSource = 'Dreamforce']);
}
}
```