Asynchronous Apex

Asynchronous Processing Basics

Use Future Methods

AccountProcessor

```
public class AccountProcessor {
    @future
    public static void countContacts(List<Id> accountIds){

        List<Account> accountsToUpdate = new List<Account>();

        List<Account> accounts = [Select Id, Name, (Select Id from contacts) from Account Where Id in
:accountIds];

        For(Account acc:accounts){
            List<Contact> contactList = acc.Contacts;
            acc.Number_Of_Contacts__c = contactList.size();
            accountsToUpdate.add(acc);
        }
        update accountsToUpdate;
    }

}
```

AccountProcessorTest

```
@IsTest
public class AccountProcessorTest {
    @IsTest
    private static void testCountContacts(){
        Account newAccount = new Account(Name='Test Account');
        insert newAccount;

        Contact newContact1 = new Contact(FirstName='John',LastName='Doe',AccountId =
newAccount.Id);
        insert newContact1;

        Contact newContact2 = new Contact(FirstName='Jane',LastName='Doe',AccountId =
newAccount.Id);
        insert newContact2;

        List<Id> accountIds = new List<Id>();
        accountIds.add(newAccount.Id);

        Test.startTest();
        AccountProcessor.countContacts(accountIds);
        Test.stopTest();



    }

}
```

Use Batch Apex

LeadProcessor

```
global class LeadProcessor implements Database.Batchable<sObject> {
    global Integer count = 0;

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

    global void execute (Database.BatchableContext bc, List<Lead> L_list){
        List<lead> L_list_new = new List<lead>();

        for(lead L:L_list){
            L.leadsource = 'Dreamforce';
            L_list_new.add(L);
            count += 1;
        }
         update L_list_new;
    }

     global void finish(Database.BatchableContext bc){
         system.debug('count = ' + count);


     }
}
```

LeadProcessorTest

```
@isTest
public class LeadProcessorTest {

    @isTest
    public static void testit(){
        List<lead> L_list = new List<lead>();

        for(Integer i=0; i<200; i++){
            Lead L = new lead();
            L.LastName = 'name' + i;
            L.Company  = 'Company';
            L.Status = 'Random Status';
            L_list.add(L);
        }
        insert L_list;

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

Control Processes with Queueable Apex

AddPrimaryContact

```
public class AddPrimaryContact implements Queueable{

    private Contact con;
    private String state;

    public AddPrimaryContact(Contact con , String state){
        this.con = con ;
        this.state = state ;
    }

    public void execute(QueueableContext context){
        List<Account> accounts = [Select Id, Name , (Select FirstName , LastName , Id from contacts )
                                  from Account where BillingState = :state Limit 200] ;
        List<Contact> primaryContacts = new List<Contact>();

        for( Account acc:accounts ){
            Contact c = con.clone ( );
            c.AccountId = acc.Id;
            primaryContacts.add(c);


            if(primaryContacts.size ( ) > 0 ) {
                insert primaryContacts ;
            }
        }
    }
}
```

AddPrimaryContactTest

```
@isTest
public class AddPrimaryContactTest {

    static testmethod void testQueueable ( ){
        List <Account> testAccounts = new List<Account>( ) ;
        for (Integer i=0; i <50; i++) {
            testAccounts.add(new Account ( Name= ' Account'  +i,BillingState = ' CA ' ) ) ;
        }
        for ( Integer j=0; j<50; j++) {
            testAccounts.add(new Account ( Name = ' Account ' +j ,BillingState =' NY ' ));

        }

        insert testAccounts ;

        Contact testContact = new Contact(FirstName = ' John ' , LastName='Doe ' ) ;
        insert testContact ;

        AddPrimaryContact addit = new addPrimaryContact(testContact , 'CA') ;

        Test.startTest();
        System.enqueueJob(addit) ;
        Test.stopTest( ) ;
        System.assertEquals ( 50 , [ Select count ( ) from Contact where accountId in ( Select Id from
Account where Billingstate = ' CA ' ) ] ) ;
    }
}
```

Schedule Jobs Using the Apex Scheduler

DailyLeadProcessor

```
global class DailyLeadProcessor implements Schedulable{
    global void execute(Schedulablecontext sc){
        List<Lead> lstofLead = [SELECT Id FROM Lead WHERE LeadSource = null LIMIT 200];

        List<Lead> lstofupdatedLead = new List<Lead>();
        if(!lstofLead.isEmpty()){
            for(Lead ld : lstofLead){
                ld.LeadSource = 'Dreamforce';
                lstofUpdatedLead.add(ld);
            }

            UPDATE lstofupdatedLead;
        }
    }
}
```

DailyLeadProcessorTest

```
@isTest
private class DailyLeadProcessorTest {
    @testSetup
    static void setup(){
        List<Lead> lstofLead = new List<Lead>();
        for(Integer i = 1; i<=200 ; i++){
            Lead ld = new Lead(Company = 'Comp' + i, LastName = 'LN' + i, Status = 'Working -
contacted');
            lstofLead.add(ld);
        }

        Insert lstofLead;
    }

    static testmethod void testDailyLeadProcessorScheduledJob(){
        String sch = '0 5 12 * * ?';
        Test.startTest();
        String jobId = System.Schedule('ScheduledApexText', sch, new DailyLeadProcessor());

        List<Lead> lstofLead = [ SELECT Id FROM Lead WHERE LeadSource = null LIMIT 200];
        system.assertEquals(200, lstOfLead.size());

        Test.stopTest();
    }
}
```

Monitor Asynchronous Apex