Linkedin Sales Systems Team - Technical Assignement

30th September 2016

Edgar Yucel Morán

OVERVIEW

In a no customized Salesforce.com instance, is required to manage a specific flow every time a contac leaves a company. Basically everytime the Company (Account) record changes, we need to convert the contact back to a Lead record with specific field mapping.

ARCHITECTURAL GOAL

The goal of this flow is to provide users a mechanisim that allows convert back Contact records in salesforce to a Leads unlinking the record from the roles assigned to one or many opportunities.

The solution could be used two ways:

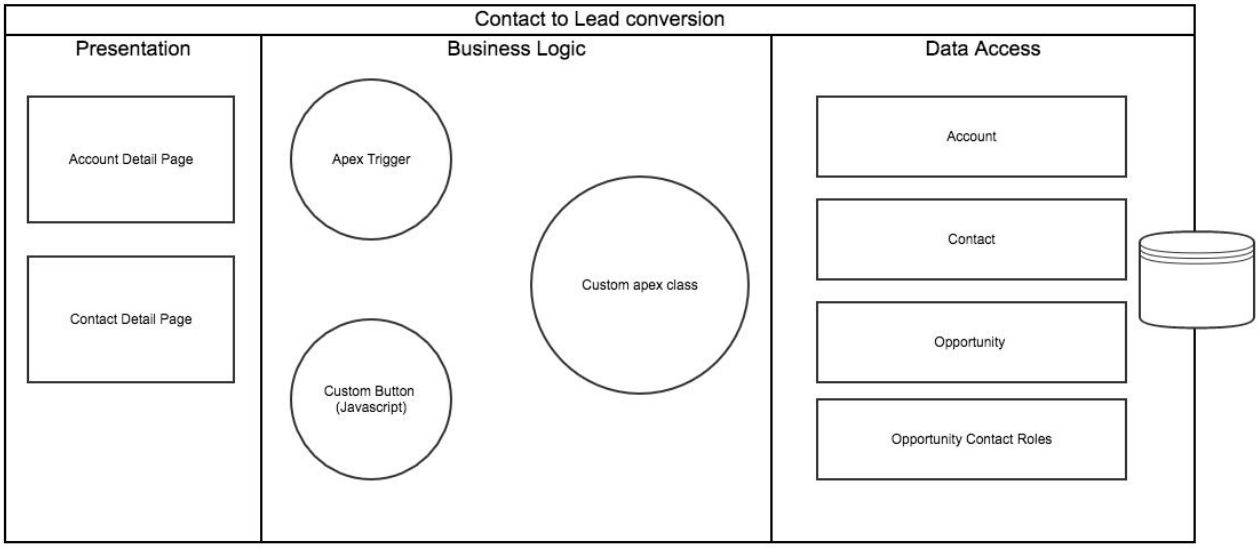
- Every time a Account record is changedd (updated it) the process is fired
- Every time the User in Salesforce clic on the "**Convert Contact to Lead**" button on the contact detail page.

SPECIFICATIONS

- Whenever the Account on Contact record changes, the contact should be converted back to a Lead. Mapping from Contact to Lead fields provided by Linkedin
- Delete any contact roles associated with open opportunities for the contact with the old account
- Send a plain text email to the Opportunity owner notifying the owner about the change unless the user who changed the Account value is the same as the Opportunity owner.
- Add a button on the Contact detail page to un-convert a contact back to lead on demand regardless of Account change. (If the Opportunity owner was the user who clicked the button, do not send any emails)
- Delete the contact

APPLICATION ARCHITECTURE

The contact convertion to a lead will be integrated to an actual working environment, that means we will use the Force.com platform to add functionallity non standard in Salesforce.



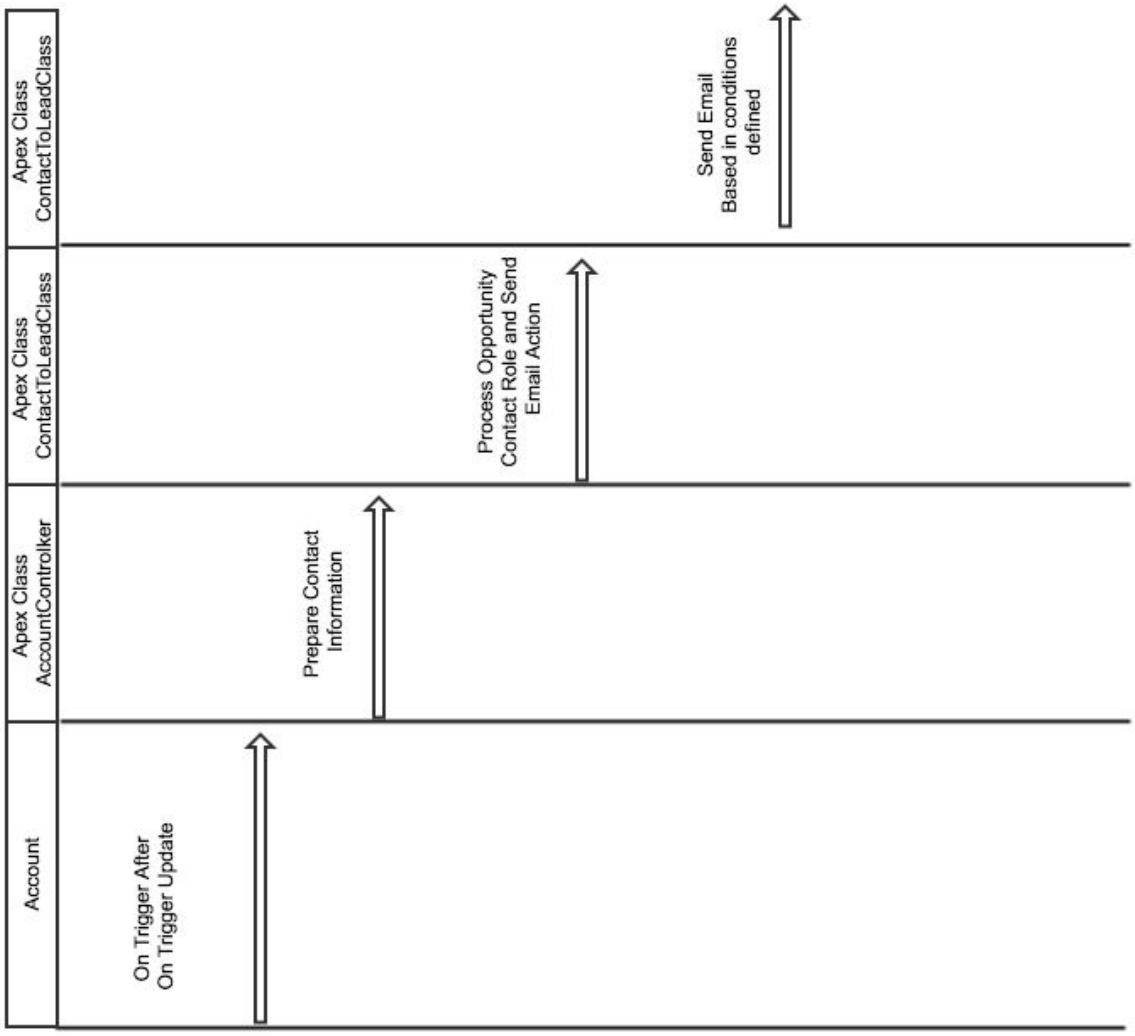
Presentation Layer: Allow users to execute the process having the actual representation of the data. It is the screen in the system that allows users to clic and fire the contact conversion.

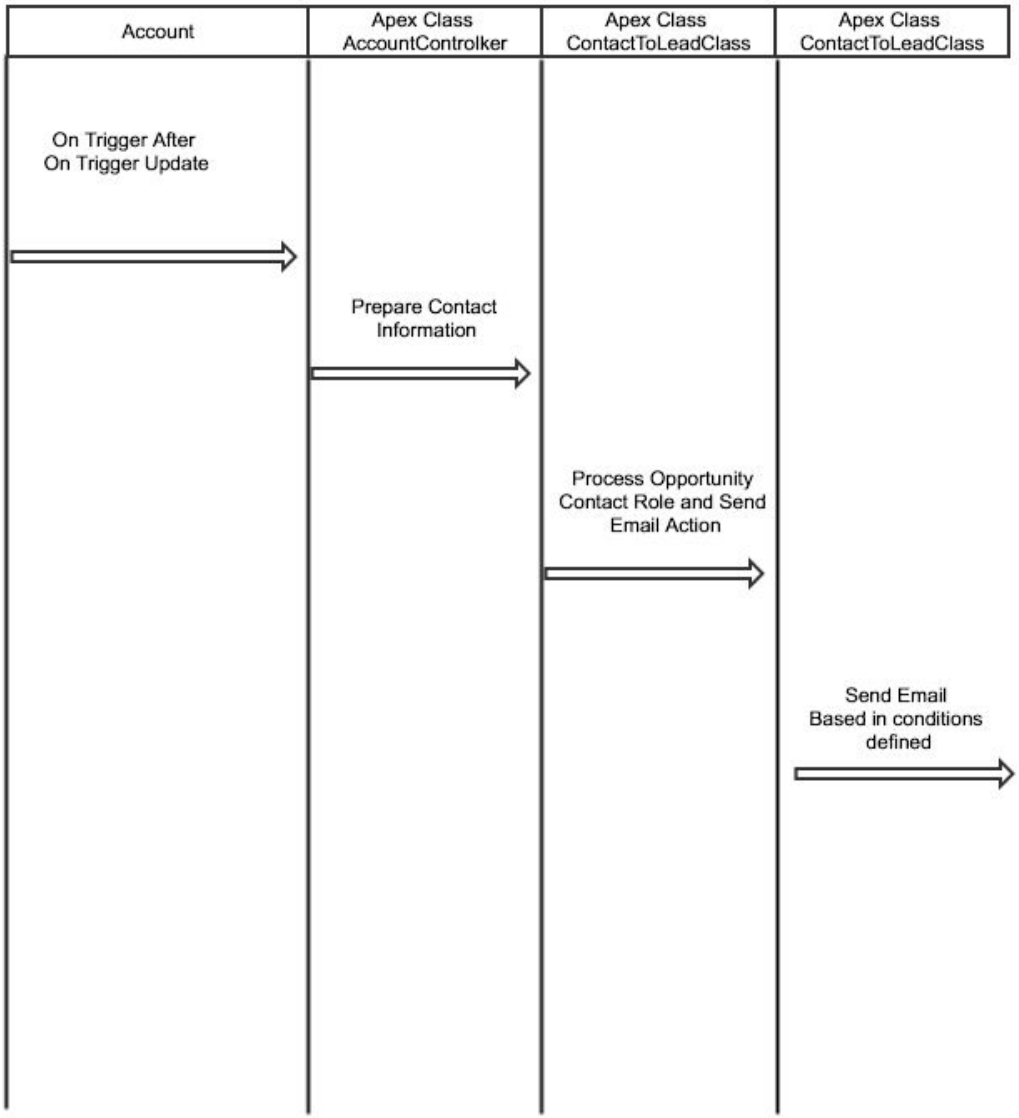
Business Logic: Allows to execute the core functionallity, grab all information allows to validate and transform any data type and proceed to executed the DML actions to the data base.

Data Access: After all information has been prepared, this layer store the that information in the Salesforce.com database storing the records as result of a DML requested action (Insert / Delete / Update / Upsert)

SEQUENCE OF EVENTS

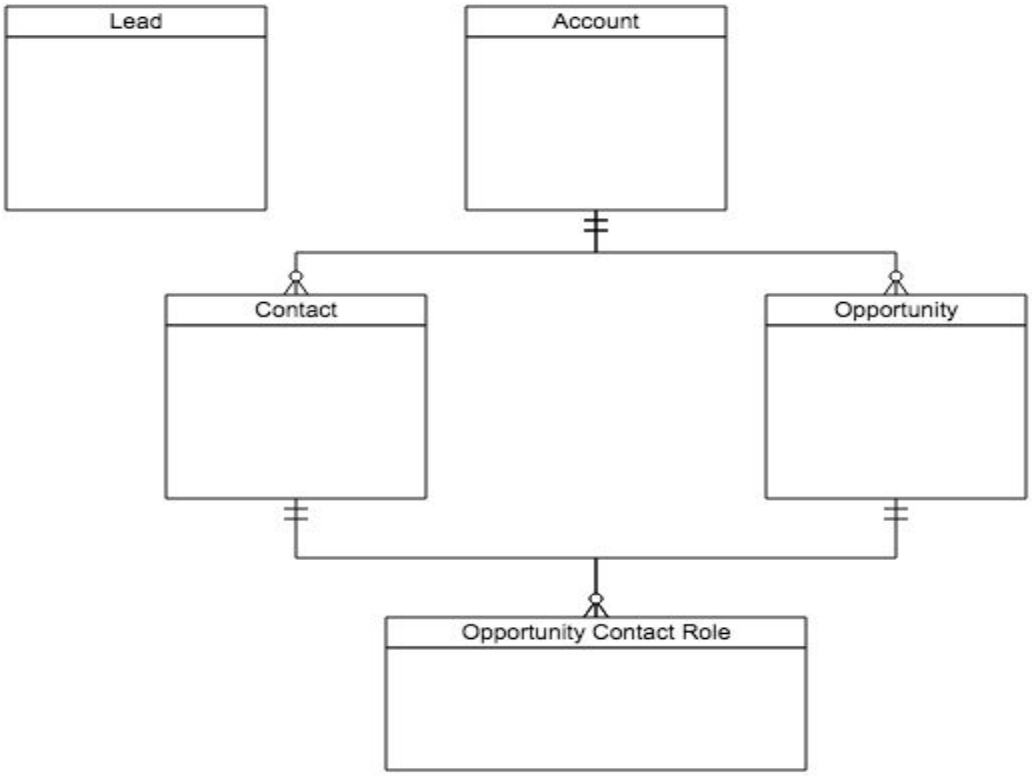
This section contains the flow events in the interaction between the user screens and the behind actions that are executed in order to complete circle of validation and execution in the conversion.




Account
Apex Class
AccountControlker
Apex Class
ContactToLeadClass
Apex Class
ContactToLeadClass
On Trigger After
On Trigger Update
Prepare Contact
Information
Process Opportunity
Contact Role and Send
Email Action
Send Email
Based in conditions
defined

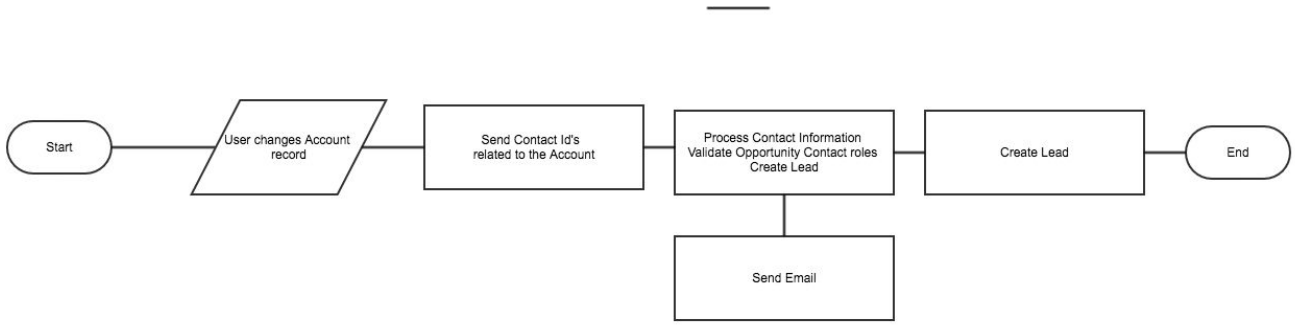
DATA MODEL

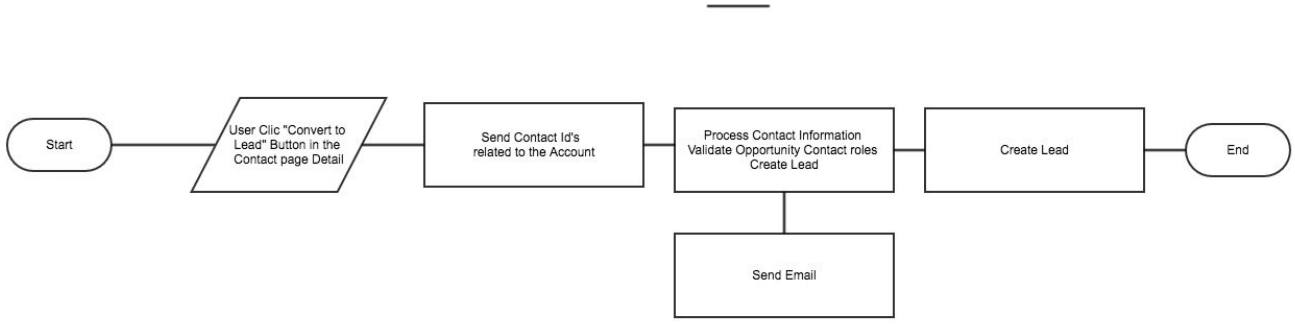
This section describes the data model that will be used in the implementation of thie contact to lead conversion.



BUSINESS FLOW

Here is a genaral flow representation of the flow execution in both cases the flow to convert the lead is called.





ASSUMPTIONS

Some assumptions around this requerimient.

- Every contact has a Account record related.
- No Cases are related to the Contact we want to convert bacj to Lead
- There’s no triggers around Account
- There’s no triggers around Contact
- The flow is executed manually per record (Even the code works in Bulk operations)
- There’s no specific field change validation on Account in the process is executed always the account is updated (LastModified Date is the field that always change).
- The email template not contain merge fields (Even now is using opportunity information and it is merged with the EmailController class)

FORCE.COM COMPONENTS IN THE SOLUTION

Here the list of every component used in order to achieve the requerimient:

- Custom Button
    - Contact
        - Name: Convert Contact to Lead
        - API Name: Convert_Contact_to_Lead
        - Type: Detail Page Button
        - Content Source: OnClick JavaScript
- Email Template
    - Name: Notify Opportunity Owner
    - API Name: Notify_Opportunity_Owner
- Apex Trigger
    - Account
        - Account_trigger
            - On After
            - On Update
- Apex Classes
    - AccountController.cls: sends the contact Id's based on the related contact records
    - CheckRecursive: Allows to check if there's a execution in progress (Avoid recursive Apex)
    - ContactToLeadClass: Verify Opportunity Contac Role records, validate the current user executing the process, send the email notification if it the criteria match.
    - CustomExceptionController: Create an instance of the Exception class to provide an custom error message when querying to email template
    - EmailNotificationsController: Send an email notification to the Opportunity Owner, it takes the email template specified above.
- TestMethod classes
    - AccountControllerTest
    - ContactToLeadClassTest
    - EmailNotificationsControllerTest
    - UnitTesDataSetup