Creating Triggers for Salesforce

Problem

Salesforce provides a bunch of standard objects and gives the ability to customize them or create new ones as required for an organization.

The fields of these objects can contain references to other objects. Example: “Case” is having references to “Account”, “Contact” and Child relationships to “CaseComments”, “Attachments” etc.,

Since these data are segregated into different entities based on references, the update dates of these entities will not change when their references change. Simply put, when you add an Attachment or Create/Update, a ‘CaseComment’ for a case, the Update date of a ‘Case’ will not change as they are separate entities.

When ConnectALL tries to sync records, It will check for the “SystemModifiedTimeStamp” field of the entity under sync, and since this will not be changed when the updates happen on “Reference” objects, we will need a way to update the record when the “Reference” objects changed or modified.

Solution

Salesforce provides a clean and optimized solution for this problem using “Apex Triggers”.

We will be using this approach to perform a “Reverse Update” of the parent record when a “Reference” object is modified.

Sample Trigger on Case Comments Object

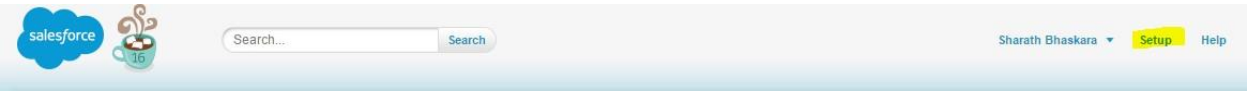
```
trigger ReverseUpdateCaseFromCaseComment on CaseComment (after u
pdate, after insert, after delete){
   for(CaseComment
comment:Trigger.new){
   String caseId = comment.ParentId;
   Datetime updateDate = comment.SystemModstamp;
   Case caseOb = [SELECT Id,Description FROM Case WHERE Id=:case
Id];
   update caseOb;
 }
}
```

Setup

Triggers on Standard Objects Ex: “Account”

Example Trigger on Account to Update the Parent Account when Child is Updated

1. Login to Salesforce and click on “Setup” link next to the username.

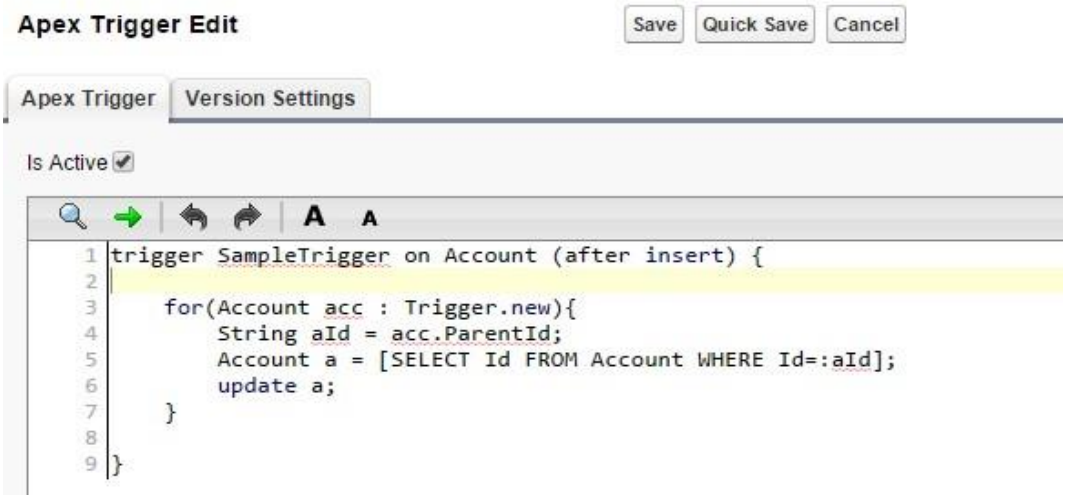


2. On the left-hand side navigation menu, “Go to Build -> Customize”.

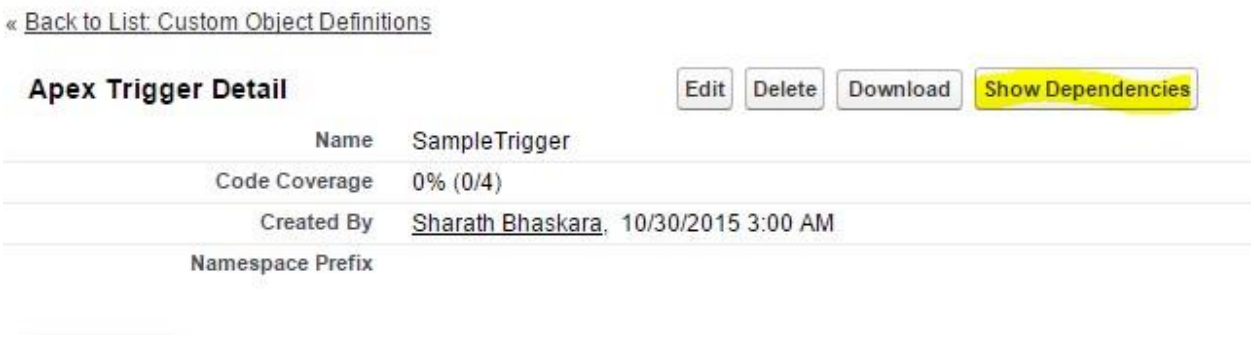
3. Select the Object you wanted to create the trigger on and click on the “triggers” link.

4. Click the “New” button to start coding your trigger.

5. Code your trigger, as shown below and click Save.



6. After Saving, you can view the supposed changes by clicking on “Show Dependencies” button.



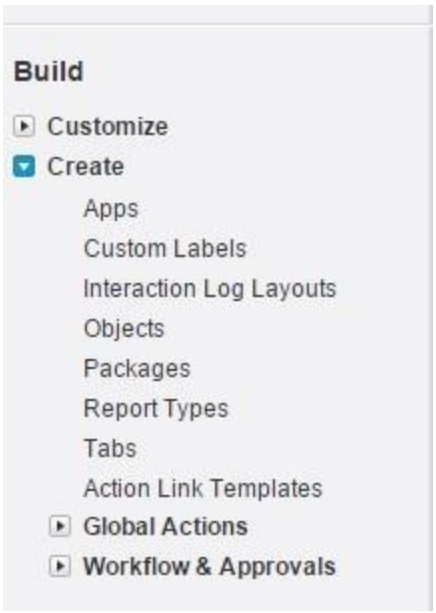
7. Notice the “Update” selected on the “Account” entity specifying the record will be updated.

Triggers on Custom Objects

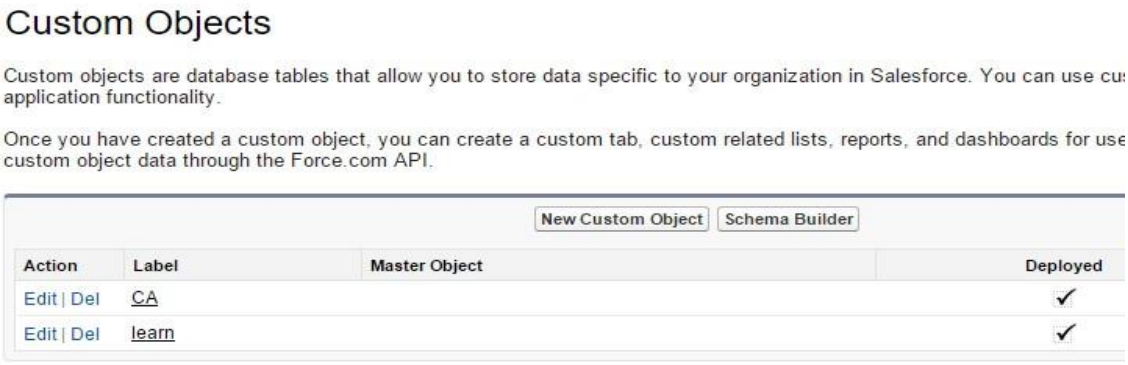
For Creating triggers on custom objects,

1. Login to Salesforce and Click on “Setup” link next to username.

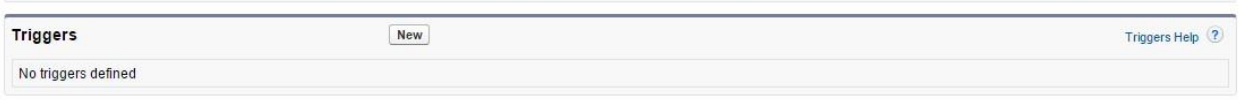
2. On the left-hand side navigation menu, “Go to Build -> Create - > Objects”.



3. You can select an existing object or create a new object as needed.



4. Click on the custom object “Label” and navigate to the “Triggers” section.

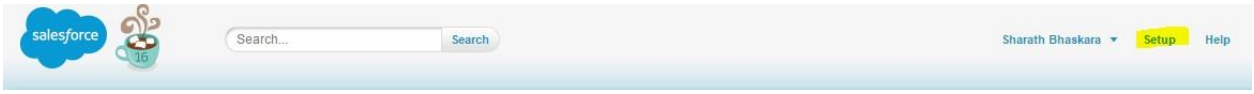


5. Click on “New” and follow the steps 5, 6, 7 discussed in [Triggers on Standard Objects Ex: “Account”]

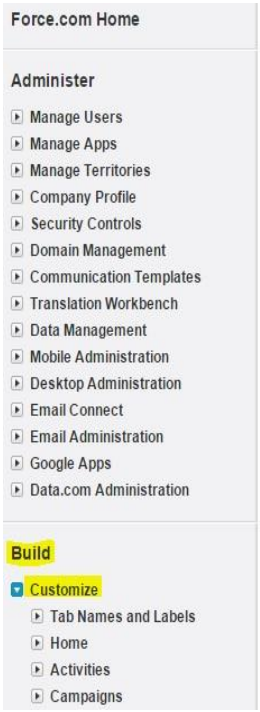
Appendix

Enabling ConnectALL to sync comments on Case when there is a update/insert in ‘CaseComments’.

1. Login to Salesforce and Click on “Setup” link next to username.



2. On the left-hand side navigation menu, “Go to Build -> Customize”.

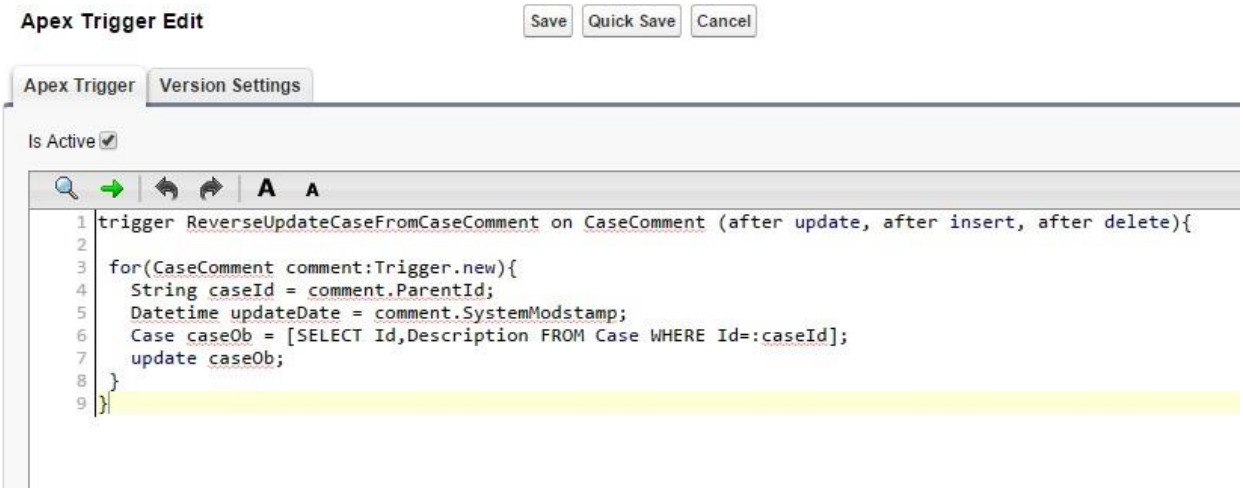


3. Navigate to “Case” and then “CaseComments” and select “CaseCommentTriggers”.

4. Click the “New” button to create a trigger.

5. Paste the below code in “Trigger” definition.

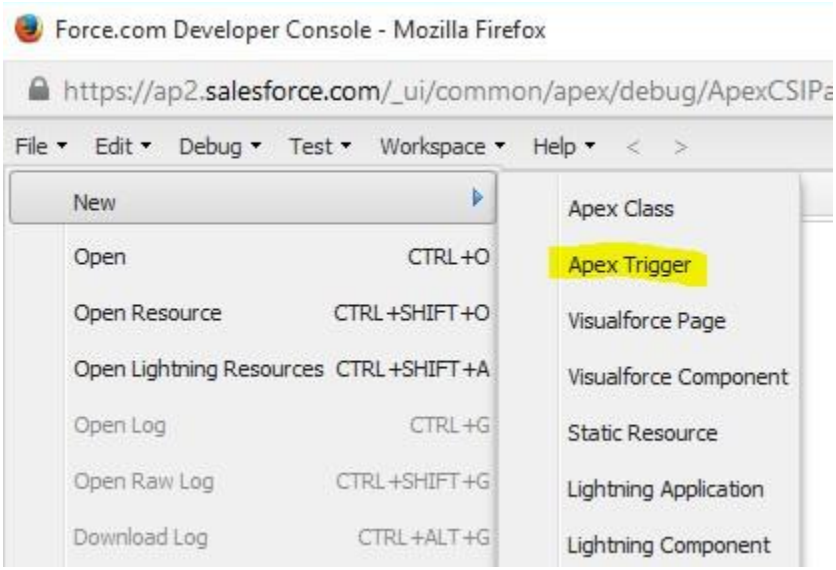
```
trigger ReverseUpdateCaseFromCaseComment on CaseComment (after
update, after insert, after delete){
   for(CaseComment
comment:Trigger.new){
   String caseId = comment.ParentId;
   Datetime updateDate = comment.SystemModstamp;
   Case caseOb = [SELECT Id,Description FROM Case WHERE
Id=:caseId];    update caseOb;
 }
}
```



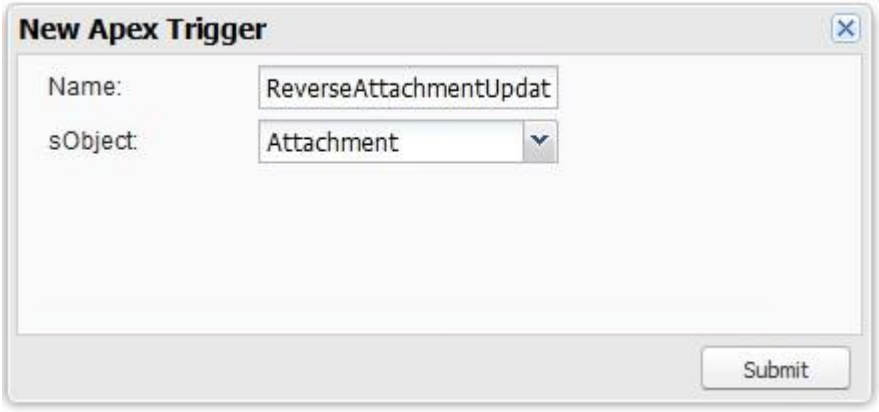
6. Click on Save, to finish creating a trigger.

Enabling ConnectALL to Sync Attachments on Cases

1. Login to Application and click on the “Developer Console” link under the username.
2. In the opened “development editor”, navigate to “File” -> “New” -> “Apex Trigger”.



3. In the below pop-up, enter a name for the trigger and select the “Attachment” object from dropdown and click on Submit.



4. Now type the below displayed “Trigger definition” (below image) in the editor. The below provided sample is for the type ‘Case’.

```
trigger ReverseAttachmentUpdateOnCase on Attachment (after insert, after update, after delete) {
    List<Schema.SObjectType> ObjectType = new List<Schema.SObjectType>();
    ObjectType.add(Case.SObjectType);

    for(Attachment att : Trigger.new){
        if(att.ParentId.getSObjectType() == Case.SobjectType){ // Update the parent when it's of type Case
            String parentId = att.ParentId;
            Case caseOb = [SELECT Id FROM Case WHERE Id=:parentId];
            if(caseOb != null){
                update caseOb;
            }
        }
    }
}
```

5. Save the Trigger Definition and close the editor to finish the setup.