Triggers for Recruitment App Scenario

(vkranjithkrishnan@gmail.com)

Exercise 1:
Write a trigger to avoid deleting a Job Application record, if the status on application is Approved.

How to approach?
Step1: Identify the event
Since it is required to prevent the user from deleting the Job Application, it would be "Before Delete" event.

Step2: Identify the Objects and fields to be evaluated
Object: Job_Application__c
Fields : Job_Application__c.Status__c is = "Approved"

Step3: Identify the trigger context variable to be used.
The existing records that is invoked by delete operation can only be available in trigger.old and hence the context variable to be used is trigger.old. Since it is a list of old version of Job_Application record, we can check the Job_Application__c.Status__c is = "Approved" for the record that user trying to delete.

Step4: High level logic

i. Assign the trigger.old[0] to a list variable (to process only one record at a time).
ii. Check the Status field for the list record saved above if it is equal to "Approved".
iii. if Status__c is approved, then terminate the transaction and display the user with error message to alert the approved job application cannot be deleted.

So the logic could be as follows

```
//Trigger.old[0] is list referring a single record using indice 0.
//Conditional check referring the stored list above.
//addError(errorMsg) - Marks a record with a custom error message and prevents any DML
// operation from occurring.

List<Job_Application__c> JobApp = Trigger.old[0];.
If(JobApp.Status__c = "Approved"){
   Jobapp.addError('Approved Positions can not be deleted!');
}
```

Note:Refer the url to know more about the other methods of sObjects - https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_methods_system_sobject.htm.
Basically every sObject in salesforce is a class and hence we can instantiate the sObject into an object (blue print of a class) and refer the methods of it. addError() is a type of sObject methods provided by force.com.

We are done with the trigger now, but what about if the trigger is invoked for more than one Job_Application record at a time. For example, if a user is trying to delete more than record at a time using Dataloader. Then the above trigger will work for just the first record since it is referred as trigger.old[0]. Hence we may need to bulkify the code to enable the trigger to process more than one record at a time (when we do a delete operation using dataloader using a csv file containing more number of Job application records which is considered as one transaction).

Triggers for Recruitment App Scenario

(vkranjithkrishnan@gmail.com)

The below code is bukyfied to process more records at a time by referring the complete list.
We know that trigger.old is list and hence can be referred in for loop directly like below and iterate.

i. Refer the trigger.old to iterate over the list of old versions of Job_Application__c records.

```
trigger notdeleteApprovedApplication on Job_Application__c (before delete) {
  for(Job_Application__c Jobapp: trigger.old){
    if(Jobapp.Status__c == 'Approved'){
      Jobapp.addError('Approved Positions can not be deleted!');
    }
  }
}
```

Triggers for Recruitment App Scenario
(vkranjithkrishnan@gmail.com)

Exercise 2:
Write a trigger to update the stage on Job Application to interviews completed when a new review is created.

How to approach?
Step1: Identify the event
after insert

Step2: Identify the Objects and fields to be evaluated
sObject: Review__c
Fields : Stage__c of parent object (Job_Application__c)

Step3: Identify the trigger context variable to be used.
trigger.new - which contains new versions of review records. From which, we can get the ID of newly inserted review records.

Step4: High level logic
Job_Application__c.Stage__c should be marked as “Interview Completed” for the corresponding child record (Review__c).

Now time to retrieve job application using the relationship field (id field) from child sObject and mark the job application stage as 'interview completed' . To get the relationship field name, refer the custom new field section on your review object detail page and perform the logic as below.

Example: if Job_Application__c is the kind of foreign key (similar to foriegn key) in Review__c object to denote master object record (Job_Application__c). Job_Application__c ID field is similar to your AccountId in contact object to related to parent object – Account. This AccountId is provided by the salesforce itself if related sObjects are standard ones (eg., contact, account, opportunity etc.,).

Triggers for Recruitment App Scenario
(vkranjithkrishnan@gmail.com)

1. Store the Job_Application__c id from review object into a collection. Use trigger.new to get the list of relationship ids and store into a set as below so that that set can later be used to retrieve related Job_Application (parent) record.

```
Set<id> JobIds = new Set<id>();   //empty set

for(Review Rw: Trigger.new){
    Jobids.add(Rw.Job_Application__c);   //storing relationship ids into the set
}
```

Query the Job Application object for the matching id collected above and update the corresponding Stage field to 'interview completed'.

```
list<Job_application__c> JobList = [SELECT  Id,Stage__c FROM Job_Application__c
                                    WHERE Id in :JobIds];
for(Job_application__c JA: JobList){
   JA.Stage__c = 'Interview Completed';
}
Update JobList;
```

Hence the complete code could be as follows.

```
trigger updateJobStagewhenReviewed on Review__c (after insert) {
    set<Id> RIds = new set<Id>();
    Map<Id, Job_application__c> JobApp = new Map<Id, Job_application__c>();
    for ( Review__c cc : Trigger.new ) {
        RIds.add(cc.Job_Application__c);
    }

    list<Job_application__c> JobList = [SELECT Id,Stage__c FROM Job_Application__c WHERE Id in :Rids];
    for(Job_application__c JA: JobList){
        JA.Stage__c = 'Interview Completed';
    }
    update JobList;
}
```