Apex based record sharing in Salesforce

 posted by Jitendra on July 27, 2012 under category Apex, Force.com, Salesforce and tagged as Apex, Salesforce with 22 Comments

Working with Apex based sharing in Salesforce

Last update : May 4 2020

There are situations where the business requirement is too complex and standard sharing rules provided by the Salesforce will not work.

Example: On Opportunity, you want to give access to record to some users which are in related list.
One way is to manually share the record which will need the interference of opportunity owner. But everyone will love automated solution.

Apex managed sharing provides developers with the ability to support an application's particular sharing requirements programmatically via Apex code.

Before you proceed, few points to note about Apex based sharing

- Share table available only when **Organization Wide Default** [OWD] sharing is not Public Read Write
- Apex sharing reasons can be created only for custom objects
- As **Apex Sharing Reason** not available for Standard objects only way to create Apex based sharing for Standard objects are using row cause Manual
- As row cause is Manual for Standard objects, **Apex based sharing would be lost once owner is changed** [Because its behavior of Manual Sharing]
- In case of custom object & custom Apex Sharing Reason, sharing would not be lost when owner changed
- Read more in detail from official documentation

Pre requisite:

The object's organization-wide default access level must not be set to the most permissive access level. For custom objects, this is Public Read/Write. For more information, see Access Levels. This object is used for creating Apex based sharing.

The user to which the record going to be shared must have the object level permission. If you are trying to share the record with edit permission but user does not have the edit permission on that object, then it will not work.

To access sharing programmatically, you must use the share object associated with the standard or custom object for which you want to share. For example, AccountShare is the sharing object for the Account object, ContactShare is the sharing object for the Contact object, and so on. In addition, all custom object sharing objects are named as follows, where MyCustomObject is the name of the custom object: "MyCustomObject__Share".

Objects on the detail side of a master-detail relationship do not have an associated sharing object. The detail record's access is determined by the master's sharing object and the relationship's sharing setting.

In this article I want to share the custom object "Test__c"☐ with students. Let's consider that every "Test"☐ has lookup to the "Student"☐.

Sharing Table

All objects that have a default sharing setting of the either "Private" or "Public Read Only" also have a related "Share" object that is similar to an access control list (ACL) found in other platforms. All share objects for custom objects are named as MyCustomObject__Share, where MyCustomObject__c is the name of the related custom object. A share object includes records supporting all three types of sharing: Force.com managed sharing, user managed sharing, and Apex managed sharing.

A custom object's share object allows four pieces of information to be defined:

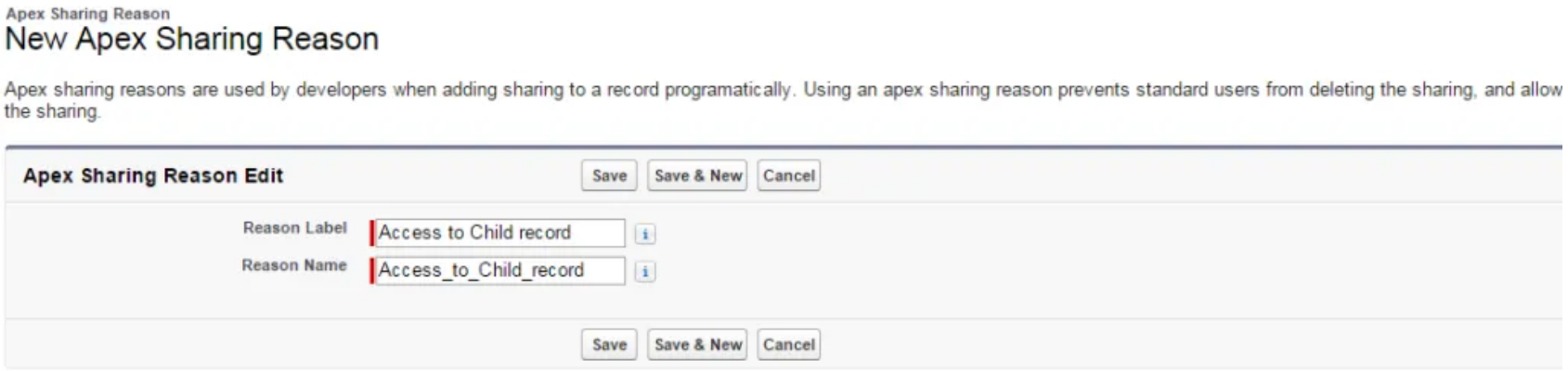
- The record being shared.
- The User or Group with whom the object is being shared.
- The permission level being granted to the User or Group.
- The reason why the User or Group has been granted sharing access.

Id	AccessLevel	IsDeleted	LastModifiedById	LastModifiedDate	ParentId	RowCause	UserOrGroupId

This information corresponds with the following fields in a share object:

ParentId	The Id of the record being shared. This field cannot be updated.
UserOrGroupId	The Id of the User to whom you are granting access. May also be a Public Group Id, Role Id, or Territory Id. This field cannot be updated.
AccessLevel	The level of access that the specified User or Group has been granted. Valid values for Apex managed sharing are: Edit, Read. This field must be set to an access level that is higher than the organization's default access level for the parent object. For more information, see Access Levels.
RowCause (aka Sharing Reasons)	The reason why the user or group is being granted access. The reason determines the type of sharing, which in turn controls who can alter the sharing record. This field cannot be updated.

Apex sharing reason



Salesforce Apex Sharing Reason

Before we can start writing any Apex managed sharing code, we must create an Apex sharing reason. To do that:

- Click “Setup | Create | Objects”.
- Select the custom object. (In this case, the “Test” Custom object.)
- Click New in the Apex Sharing Reasons related list. (If you don’t see this related list, Apex managed sharing has not been configured for your Org – please contact your support representative.)
- Enter a label for the Apex sharing reason.
- Enter a name for the Apex sharing reason.
- Click Save.

Now let’s start with Apex code:

```
trigger Test_Share on Test__c (after insert) {
    if(trigger.isInsert){

    /**
     * Test_Share is the "Share" table that was created when the Organization Wide Default
    *  sharing setting was set to "Private". Allocate storage for a list of Test_Share
    * records.
```

```
    **/
    List<Test_Share> jobShares = new List<Test_Share>();

    /** For each of the Test records being inserted, do the following: **/
    for(Test__c t : trigger.new){
        /** Create a new Test_Share record to be inserted in to the Test_Share table. **/
        Test_Share studentRecord = new Test_Share();
        /** Populate the Test_Share record with the ID of the record to be shared. **/
        studentRecord.ParentId = t.Id;
        /** Then, set the ID of user or group being granted access. In this case,
         * we're setting the Id of the student__c that was specified by the Test
         * Result in the student__c lookup field on the Test record.
         **/
        studentRecord.UserOrGroupId = t.student__c;
        /** Specify that the Student should have edit access for this particular Test record. **/
        studentRecord.AccessLevel = 'edit';
        /** Specify that the reason the Student can edit the record is because its his test result
         * (Student_Access__c is the Apex Sharing Reason that we defined earlier.)
         **/
        studentRecord.RowCause = Schema.Test_Share.RowCause.Student_Access__c;
        /** Add the new Share record to the list of new Share records. **/
        jobShares.add(studentRecord);
    }
    /** Insert all of the newly created Share records and capture save result **/
    Database.SaveResult[] jobShareInsertResult = Database.insert(jobShares,false);
    }
}
```