Salesforce Trigger Bulkification

Wed Jul 01 2020

What is Trigger/Apex bulkification?

Apex bulkification essentially means writing code in a way that allows it to process more than one record per execution i.e. the records should be able to process in bulk.

Why is this important? Well let's imagine a scenario where an admin updates a bunch of Account records in a Salesforce org using Data Loader. The admin will more than likely be updating these records in batches of 200(using the SOAP API).

Also imagine that in this Salesforce org that there is a Trigger on the Account object that must run when Accounts are being updated. If that Trigger is not bulkified then only one record per batch will be processed by the Trigger. Triggers in Salesforce should be written in a way that allow multiple records to be processed by the Trigger in a single execution.

So why can't Apex process the records 1 by 1? Well this would be a huge strain on Salesforce's servers. We want to keep our code fast and efficient and bulkifying is one of the ways we do this. Salesforce has implemented strict Governor limits to prevent orgs from taking up too many resources. For example each Apex transaction is limited to 100 SOQL queries per transaction. In non bulkified code this could be hit very quickly.

Okay, let's get to the good stuff, the examples!

Business Requirement

The business has asked you to set a description on an Account whenever the Account is being updated.

Here's what a non bulkified trigger might look like

```
trigger AccountTrigger on Account (before update) {

       //First We check if we are in the right Trigger Context BEFORE UPDATE of Account
       if(Trigger.isBefore && Trigger.isUpdate){
          Account account = Trigger.new[0]; //Grab an Account record from the trigger
          account.Description = 'This is a Business Account';
       }
    };
```

COPY CODE

The issue with the above code is that we access an Account from the Trigger.new List like so: Trigger.new[0]. This only gives us the first Account in the List. So if this were processing in bulk we would never update the description of other Accounts in that List.

Let's go ahead and bulkify the example above. We want to make sure that all Accounts have the description field updated

Bulkified Trigger

```
trigger AccountTrigger on Account (before update) {

     //First We check if we are in the right Trigger Context, BEFORE UPDATE of Account
     if(Trigger.isBefore && Trigger.isUpdate){
         for(Account account : Trigger.new){
             account.Description = 'This is a Business Account';
         }
     }
  };
```

COPY CODE

In the above example we have Bulkified the code by iterating(looping) over all the Accounts in the Trigger.new List. So now when the trigger fires and multiple records are being updated we set the Description field on all of the Accounts.