ASYNCHRONOUS APEX. WHAT'S UNDER THE HOOD?

Friday 17 January, 2020 Published in Articles

Do you know what is going on when you call Async Apex? It just creates an AsyncApexJob record and then picks up by Salesforce queue to run asynchronously when resources are available. I think we could experiment a little, let's create some classes for our needs.

```
// Apex Async method example
public class AsyncExecutionExample implements Queueable {
    public void execute(QueueableContext context) {
          // do nothing
    }
}
```

```
// Callout example
public class CalloutExample {
    public static String doCallout() {
        HttpRequest request = new HttpRequest();
        request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals');
        request.setMethod('GET');

        Http http = new Http();
        HttpResponse response = http.send(request);
        return response.getBody();
    }
}
```

Experiment #1: Database.rollback and Queueable method

If we rollback a queueable call, what happens? It rollbacks all database changes and you do not find it in the Apex Flex Queue because of the AsyncApexJob record rollback from the database.

```
Savepoint sp = Database.setSavepoint();
ID jobID = System.enqueueJob(new AsyncExecutionExample());
Database.rollback(sp);
```

Experiment #2: Queueable method and Callout

As you know you cannot make callouts with pending transactions in the request context. As a result, you will get the System.CalloutException exception. It means that when you call the Async Apex method, the Salesforce does DML operation on the background. Don't worry it does not count against your Limits.

```
System.enqueueJob(new AsyncExecutionExample());
CalloutExample.doCallout();
```

Conclusion

1. Do not use Async Apex methods before callout methods.
2. Know that Async Apex methods can be rollbacked.
3. Salesforce does not immediately add your async method to the queue, it happens only when salesforce commits all DML operations to the database.