Ensuring the Stability of Lease Transaction and Salesforce Batchjobs through Automation

Mr. Shankara Naryana Prasad S U[1] Dr. Deepamala N[2]
[1]PG Student [2]Associate Professor
[1,2]Department of Computer Science Engineering
[1,2]RV College of Engineering, Bengaluru

Abstract— A lease contract can be coined as an agreement between the lessor and the lessee where the lessee pays to the lessor for use of an asset or equipment such as buildings and vehicles. The lessor is the legal owner of the asset. Lease industry is expected to reach, US$1 billion industry by 2020 [1]. So the success of this industry or business depends on finding new ways to drive revenue through greater transaction volume and rapid time-to-market for new products. Lessors need online application engines and sophisticated tracking systems to facilitate the workflow of an application. Force.com is a Platform-as-a-Service (PaaS) offering that provides the customers to create and deploy applications. Batchjobs are periodic processes which is responsible for all transactions, based on the schedules entered. Apex tests can be used to automate unit tests for the Apex Code as well as functional tests that simulate user actions through the user interface (UI) or Force.com Web Services. This paper illustrates how test automation can be implemented in Force.com platform for developing lease management application.

Key words: Salesforce, Automation, Batchjobs

I. INTRODUCTION

Lease industry is expected to reach, US$1 billion industry by 2020 [1]. So to capture this market there is need of a sophisticated application, which can process any number of contracts simultaneously. Number of transactions are involved in this application such as bill generation, late charge, accruals, delinquency calculation, periodic charge, payment, payment reversal etc. Cloud platform is used for both automation and developing this application since it is impossible to determine the amount of data being generated and storage needed for the same [2]. Cloud platform used is Salesforce.com. Testing is critical step involved in any software development because it is responsible for the quality of the software developed by any organization. Industry is moving towards more and more automation since it is both cost effective and time effective. Automation of tests enables an easier and faster way to catch regressions during the software development cycle. There are various open source and commercial tools available for automation, but for all these solutions need to have dedicated resources to develop, maintain, and run tests. Salesforce.com provides on-demand platform for development, the importance of utilizing a scalable and robust testing infrastructure becomes even more important for ensuring quality of application. Batchjobs process thousands of records every day. Automation make sure that application is working as expected. The Apex test framework enables developers to focus on testing without worrying about any additional testing infrastructure.

A. Salesforce.com:

Salesforce.com pioneered the Software-as-a-Service(SaaS) industry by initially offering a Customer relationship management (CRM) service on demand. The infrastructure is based on an environment where customers share the same hardware resources for application functionality, but data is separated for each customer [3].

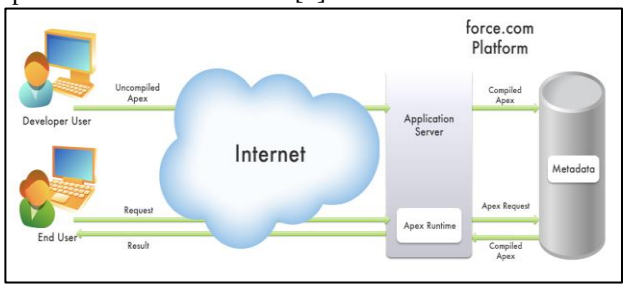

Fig. 1: Salesforce Architecture

This is also referred to as a multi-tenant environment. In this shared environment, customers are running on the same version of the application, and with each new release, have new features available for use. There is no waiting for CDs to ship and no binaries to download off from an FTP site; essentially, there is "No Software". Apex is the programming language used in Salesforce.

B. Force.com Platform:

The Force.com platform provides customers with features that allow them to customize their business solutions. This platform enables developers and partners to build and test applications without worrying about the software and hardware infrastructure. The platform supports a Model-View-Controller (MVC) framework which enables customers or partners to build or customize any of the MVC components:

sModel: Salesforce Objects (sObjects) represent the schema for the solution. There are standard sObjects like Account and Lead which are used by the CRM solutions. Customers can also define their own objects using the platform to represent almost any schema. For example, a customer can define objects like Quote for a quoting application. Listing 8 shows an XML representation for an sObject that represents an entity called Lead with two fields 'First Name' and 'Last Name' of type 'Text'.

View: Visualforce is the platform feature that enables customers to develop their own User Interface (UI) pages for the sObjects defined in their applications.

Controller: Customers can write controllers in Apex Code that define the business logic for some of the standard data manipulation language (DML) operations such as insert, update or delete on an sObject. The platform provides standard controllers as well as the capability to extend these controllers.

C. Salesforce Batchjobs:

Salesforce Batchjobs are periodic process, which is responsible for all the transactions in the application. When these Batchjobs are run for particular contract, all fields are updated based on the schedule.

II. PROBLEM DESCRIPTION

As with any application, Force.com platform lease application require testing to ensure that there are no regressions. How to ensure that developed application is stable after every change in the code. It is extremely difficult to manually check each and every feature in the application, since it is time consuming. Before the introduction of the Apex programming language, customers had to test using their own UI and API integrations. During this time, customers were required to maintain their own resources or infrastructure to perform any testing. This can be overcome by automation [4]. Some of the following issues illustrate the challenges with testing on the on-demand platform.

A. Regression Tests:

With so many transactions involved the application, developers need to validate that any customizations or changes they make in the code do not regress any functionality. For a large organization which might have many administrators, there needs to be a way to ensure that these collective changes do not break application functionality. A transaction can have hundreds of test case based on business scenario and every test case should pass when it run during the deployment to the Salesforce's main server. Even if single test case fails, which means software is not stable and error message will be displayed [5].

B. Acceptance Test for Lease Application:

When an organization installs an application from AppExchange, it's possible that the application will make assumptions on the configuration settings of the consuming organization which may or may not be correct. There is no easy way to determine this without installing the application and exercising all its functionality. This is not a scalable solution when the application is very large and may have many integration points. There is a need for exercising all the functionalities of the application before installing it. Therefore, as part of the Apex test framework, all developer written tests are executed during install, providing the necessary behavioral validation.

Because of the multi-tenant platform, any updates that Salesforce.com releases is automatically deployed to all customers. Salesforce.com needs to ensure that no existing behavior is regressed when moving to a new release. Considering the open ended nature of providing an entire programming language to customers, there is no way to effectively test all permutations in which the code is currently being used or will be used in the future. This is where tests written by developers of Apex applications can actually augment salesforce.com's already robust quality process [5].

III. SOLUTION

Apex Code provides a test framework to automate tests for validating functionality [6]. Since all Apex Code is on the platform itself, customers do not have to worry about the maintenance of these automation test suites.

A. Apex Test Methods:

Test methods (also known as unit tests) allow testers to ensure robust and error-free code. Apex tests are class methods which execute without committing any data. Below, an Apex test example is shown. HelloWorld is a regular class which has a static method. This class also includes a test method which provides basic behavioral validation and will be executed as part of the Apex test framework.

```
public class HelloWorld{
public static String getGreeting(){
return('Hello World');
}
static testmethod void testGreeting(){
String gr = getGreeting();
System.assertEquals('Hello World', gr);
}
}
```

The test method 'testGreeting' validates that the method 'getGreeting()' returns the correct value. The Apex methods which are tagged with 'testmethod' keyword are always treated as individual tests. These methods can run any Apex code like any other regular Apex methods. They are treated as tests when they are run within an Apex testing context. These test methods take no arguments. A major feature of these test methods is that they commit no data to the database. This guarantees that the tests can be run without changing any data related to the application.

Test methods can call methods in other classes just like any other regular static methods. All Apex tests written for an application can be run using the Salesforce.com UI or API. The UI provides the option to run all tests in a specific Apex class or for all classes. The API provides the same options as well. It also allows choosing the tests for a specific application as an organization could have several installed AppExchange applications.

The test results report the following after executing any tests:

1) Number of tests run
2) Number of test failures
3) Percentage of code coverage as well as the coverage details, such as which lines were covered, which lines were not, and so on
4) Profiling information for the test execution, which provides details of the number of expensive operations like SOQL and SOSL queries as well as the timings.

This information is helpful in determining the quality of the code. Here are the requirements for the tests in an application that must be met before the applications can be deployed or distributed to other organizations.

1) There should be no test failures
2) In order to deploy an application, 75 percent of the application's lines of code need to be covered by its tests. It is recommended by salesforce.com that all logic and logical branches of Apex Code are covered and executed in testmethods despite the minimum code coverage requirement.
3) All Apex triggers in the application should be invoked at least once. This is to ensure that all sObject database save operations continue to function as desired with the additional Apex Trigger customizations.

B. Apex Test Setup:

Since Apex Code runs in a multi-tenant world, there are restrictions called governor limits which ensure that no application blocks any other application by monopolizing the service. For example, the platform cannot allow an application to perform too many database queries as that may take up too many database connections thus causing a degradation in service for other customers. If any governor limit is reached during an Apex request, then the entire transaction is rolled back. There is no way to handle a governor exception; once one is thrown, it causes the entire request to fail. Apex tests should be written to ensure that applications never hit these limits. The governor limits could lead to issues when doing setup for Apex tests as tests are also bound by these limits. Doing both setup and testing within the same request may cause the governor limits to be exceeded depending upon how complex the functionality being tested is. Apex Code provides a way to define the start and end of a test so the governor limits are reset accordingly. Below shows a test which makes use of the test setup feature.

```
@isTest
private class AccountTestSuite {
static testmethod void
testInsertAccount(){
List<Account> ls = AccountUtil.
createAccounts(100);
insert(ls);
Test.startTest();
for(Integer i=0;i<ls.size();i++){
ls.get(i).Description = 'Account
Update';
}
update(ls);
Test.stopTest();
}
}
```

The Test.startTest and Test.stopTest methods help to clearly define the start and end of a test, whereas the lines of code prior to Test.startTest() set up the test data. There is a limit to the number of SOQL queries that can be done within a request. The method 'Limits. getQueries()' returns the number of SOQL queries executed so far. In this example, the governor limit for SOQL queries gets reset to zero after calling 'Test.startTest()'.

Overall Code Coverage

Class	Percent	Lines
ContractPaymentAdjTxnTrigger	100%	13/13
ContractPaymentTransactionTrigger	100%	14/14
ContractProcessor	0%	0/99
ContractTrigger	100%	16/16
CopyContractAction	65%	103/157
CopyContractCtrl	87%	14/16
CreditACHFileGenerator	85%	102/120
CreditACHProcessingAction	92%	25/27
CreditACHProcessingHandler	35%	26/74
CreditACHProcessingJob	77%	27/35
CustomLogicUtil	90%	29/32
CustomSettingsUtil	80%	32/40
DateUtil	96%	131/136
DayProcessController	36%	15/41
DayProcessTrigger	100%	4/4
DBAction	19%	4/21
DealerFundCtrl	42%	14/33
DealerFundingJob	90%	30/33
DealerFundingJobHandler	46%	21/45

Fig. 2: Line Coverage while Automation code is running

Status	Test Run	Failures	Total
✔	TestRun @ 8:59:05 pm	0	38
✔	TestRun @ 9:05:30 pm	0	29
✔	TestRun @ 9:08:08 pm	0	16
✔	TestRun @ 9:13:31 pm	0	16

Fig. 3: Test Run

The execution overview graph is used to analyze and rectify the code such that performance bottleneck is achieved.

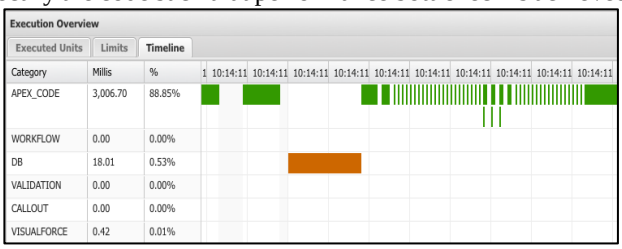

Fig. 4: Execution Overview

IV. FUTURE WORK AND CONCLUSION

In this paper we have seen how automation can be carried out for the lease management application.

The framework will continue to be enhanced by salesforce.com. Below are some of the enhancements planned for the Apex test framework:

1) Enhanced UI for displaying the test results when executing the test from the UI.
2) Enhanced UI for line-base code coverage results.
3) 3. Ability to emulate more security restrictions in the tests based on the running user
4) Scheduled/asynchronous tests

With the significant move of application development to hosted platforms, salesforce.com needs to provide a reliable and robust way to develop tests on the SaaS platform. Apex Code provides not only a development

environment for building applications but also a fully featured test framework on the Force.com platform that will become more robust as time goes on. The Apex test framework ensures the quality of a customer's own application as well as applications provided by other developers on the platform.

REFERENCES

[1] Facts and figures on the German leasing market 2015, BDL Annual Report, Leasing 2016.
[2] Cloud SaaS Applications for Business, Cloud Strategy Partners, IEEE eLearning Library Cloud SaaS Applications for Business Transcript, 2016, pp. 1-16.
[3] Cloud Service and Deployment Models, Cloud Strategy Partners, IEEE eLearning Library Cloud SaaS Applications for Business Transcript, 2016, pp. 1-15.
[4] Jigar Patel, Ankit Chouhan, "An Approach to Introduce Basics of Salesforce.com: A Cloud Service Provider", 2016 International Conference on Communication and Electronics Systems, 2016, pp. 560-572.
[5] Juee Daryapurkar, Anjali Raut, "The Multitenant Application Based on Salesforce.com", IJCSMC, Vol. 3, Issue. 12, December 2014, pp. 555 – 558.
[6] Reena Mathew, Ryan Spraetz, "Test Automation on a SaaS Platform", 2009 International Conference on Software Testing Verification and Validation, January 2009, pp. 317-325.

