THE CUSTOMER SUCCESS PLATFORM
SALES SERVICE MARKETING COMMUNITY ANALYTICS APPS

Testing Strategy

January 2017

Version 1.2

Table of Contents

Whitepaper Series

This document is one of a series of Whitepaper on the Salesforce Software Development Lifecycle. The complete series is:

1. Introduction to Salesforce Software Development Lifecycle.
2. Agile development – how to carry out agile development with Salesforce technology, including the basics of scrum.
3. Environmental Management – This cover roadmap planning and Sandbox architecture.
4. Technical Governance – Development, Configuration and coding best practices.
5. Release Management – the processes and tools to migrate from Development, Test, UAT, Production, including source control and Continuous delivery.
6. Testing Strategy – the processes and tools need to carry out comprehensive testing on the Salesforce Platform (this document).
7. Citizen Development – discusses the processes to enable Citizen Developers within your company.
8. Coding Standards – comprehensive set of Salesforce coding standards (this document).

Scope

This document defines best practices for developing “Classic” Salesforce technology. (“Classic” refers to Salesforce technology that runs on the Force.com platform) for an introduction to the topic please review Development Lifecycle Guide

Introduction

In the development of any software solution, the area that the is under resource is testing, this document outlines Salesforce testing best practices and reviews a number of 3rd party testing tools which can be used to help automate the overall testing process.

Note: Within this document, it assumes that the agile methodology is being used, however, all the principles still apply to other development methodologies.

Testing Best Practices

When building any kind of application, whether using Agile or Waterfall delivery methodology, the biggest make or break step is testing. Many people underestimate this task and do not allocate adequate skills and resources to this task. However, it is the most critical task of the SDLC process.

Why testing is important

The kind of testing described here ensures that the capabilities being developed meet the definition of “Done” as specified in the detailed user stories. The criteria are:

- Drives quality.
- Meets the requirements.
- Works as expected.
- Satisfies the needs of the users and stakeholders.
- Keeps the system usable.

Two significant forms of testing are:

- Static testing: Finds issues quickly and early in the process. It involves code reviews and system walk-throughs with the scrum team. This form of testing is a standard practice in a high-performing scrum team. It is recommended this is done by all scrum teams.
- Dynamic testing: Works by running the software, and is discussed below in Testing Strategy.

When to test

Start testing on day 1 of the project, and test continuously throughout every phase of the project.

Testing Strategy

A test strategy details the testing approach of all Salesforce projects. It informs the scrum teams and others about the key elements of the testing process they should follow. The testing process starts during the sprint planning when the Product Owner and the Scrum team negotiate and prioritize which user stories from the Product Backlog that should be included in the Sprint, the test engineers within the scrum team will be estimating the scope of the testing effort and this should be feed into the negotiation. Also when the business user stories are being re-drafted into technology terms during the sprint planning, the test engineers will start drafting the testing scenarios to cover the definition of “done”. Testing normally falls into a number of phases:

- Component and scenario testing.
- System testing.
- Integrated testing.
- UAT testing (This is not part of the sprint).

Component and Scenario Testing

This test phase starts during the initial user story development

- Each component of the system.
    - Apex - it is the responsibility of the developers to ensure there is a very high level of test coverage, the aim is 100%. Also, it's important the unit test also carries out negative testing to ensure error are trapped correctly, see below for details. As the developer writes the code they should simultaneously write the test classes to verify that the code operates as planned. **Never** add dummy test classes to overcome the 75% code test coverage threshold.
    - Configuration - it is the responsibility of the test engineers that unit test scripts are developed for all configuration especially Validation Rules, Formula fields, workflow, security, if possible. Note: some configuration unit tests can be done via Apex test classes.
    - Visualforce/lightning components - it's jointly the responsibility of the developers & test engineers to complete these unit tests.
    - All code has also had the code security scan carried out.
    - Code Migration: since it's recommended to use a migration tool based on ANT, it's important that the migration scripts are also verified.
- Scenario testing: When a sufficient number of the components of a user story have been developed the test engineers can start the scenario testing within the Sandboxes.
- In parallel with the scenario testing, the test engineers should run their test scripts across all the planned supported web browsers/mobile devices, it's important during the definition of "done" to state the supported Web browsers and mobile devices
- The above 3 steps are very iterative until the near the end of the development of a user story. Remember when developing your unit test to include the data to run your tests.

System testing

When the development team believes they have completed the development of a user story, the next phase of testing can start.

- Integrations: if the user story involves integration to 3rd party systems, testing need to be carried out that the integration works as planned for positive and negative data. This task many require resources from the backend systems, so project alignment is key here.
- Performance: this testing involves 4 major areas:
    - The end-to-end system testing to ensure the roundtrip time is within the parameters defined within the definition of "done".
    - User performance to ensure the user interaction meets the definition of "done".
    - Large Data Volumes, if you application has or plans to have a significant amount of data, you need to ensure the data volumes do not impact the user performance or even times out.
    - If your enhancement have batch processes these need to tested to verify no governor limits are hit.

- Security: the final area of system testing, does the user story meet all the overall Program security requirements and the user story definition of done. As with the other phase, positive and negative testing should carried out.

Integrated Testing

The final area of testing for the scrum team is when all the user stories have been completed the full release can be tested. If all testing has been via the use of scripts, this phase is to run all the already developed tests. However, this should be done with larger data volumes. It is also best practice to include an additional test to model complete end to end business processes which include the Backend systems.

UAT

User Acceptance Testing is the final testing phase of the project in which users validate that a solution works for business and will work for end users' normal, day-to-day use.

During the UAT phase, testing should be conducted by the Power Users. UAT is one of the final phases of software development in which the business accepts the solution delivered prior to releasing it to production. UAT testers are encouraged to use test cases specially created by themselves too closely mimic the normal business use of the application, not to run test scripts developed by QA. The advantage of getting the Power users to carry out UAT is that they are the first users with visibility of the new capabilities and this will help with the on-boarding of the wider user community.

UAT acts as a final verification of the business functionality and proper functioning of the system, emulating real-world scenarios as defined in the User Stories. If the software works as designed and without issues during normal use, one can reasonably extrapolate the same level of stability in production.

During the UAT phase, testing by Power Users normally focuses on business functions rather than on cosmetic issues or software crashes, as those are normally tested, discovered and resolved in the Unit testing and QA phases. If any of these issues are discovered, they are added to the Bug Backlog and it's the Product Owner working with the Release Manager to decide is the release pushed to production.

Note: There is an additional phase of testing that can be carried out by end users, this is “End User Friendless” testing to verify the new release will improve the productivity of the average user.

Data Migration

Many Salesforce projects involve the on-boarding of new users, which may involve the migration of data from systems that will be retired. Too many customers underestimate the effort of data migrating and just as important is the cleansing of the data before the users go live. If your capabilities require data migration it is recommended the data be cleansed during the migration process, otherwise, it will not happen and the adoption of the system will be impacted. This should be considered as a complete sprint in its own right and carried out early within the release. There is a 3rd party tool Salesforce Data Compare which is useful for post-data migrations to ensure that the data is migrated successfully.

Regression Testing

If the capabilities being developed are going to be deployed into an existing Org it is paramount that the test scripts for all the existing capabilities and application are run on a regularly base to ensure zero defects.

Roles and Responsibilities

Role	Responsibility
Project Manager	• Set schedule for testing phases • Provide sign-off on all test plan deliverables • Work with Scrum Master to plan the application code promotions to test environments • Review all submitted defects during defect review meeting and confirm initial priority of the defect • Coordinate UAT testing with Power Users
Developers	• Develop the capabilities • Develop Apex unit tests • Develop Configuration unit tests
Test Engineer	• Create test scenarios and scripts • Execute test scenarios • Defect reporting and verification • Regression testing • Communicate testing status
Automation Test Engineer	• Plan Automation Tests • Build and Maintain Automation Test Scripts • Execute Automation Test Scripts
Product Owner	• Provide business direction • Ensure the definition of Done is achieved • Approve the release testing strategy • Prioritize defects and add to Sprint Backlog or Product Backlog
Power Users	• User Acceptance testing
Release Manager	• Managing the application to production

Apex - Best Practices

Apex Unit tests should be developed at the same time as the main code is being written. The unit tests can be written either before, during or after the main code is being written but should always be considered part of the main code writing task.

Any time code is updated to include new functionality or changed functionality, the unit tests must also be updated to include tests for the new/changed functionality, and not forgetting the data sets required to run the test. If this is the case, then times needs to be allocated during the sprint planning to complete.

Test code should be reviewed in the same way as functionality code and go through the code review process.

The following are best practice all team should follow:

- Unit test coverage should be 75% (recommendation, strive for 100%) for all new code.
- If a Class / Trigger / Controller is updated and the test coverage is low, this technical debt should be removed and the test coverage should be increased.
- All Unit Tests should be run regularly.
- Test both valid and invalid input to methods to also verify the Try/Catch block logic.
- Include both positive and negative tests to ensure it fails correctly.
- Test for bulk processing with collections of at least 20 records where applicable.
- Do not use live data; create all test data for unit tests. Using manufactured data helps with unit test portability across different types of sandbox environments since you can't assume the test data is available.
- Remember, test classes are code; as such, they should adhere to coding best practices and standards and be properly commented.
- Use a standard naming convention for test classes, such as classname_Test for test classes.
- Do not create one large test method; break the tests into multiple methods with narrow goals. This approach makes it easier to see the purpose and specific functionality of the test and to quickly find problems when they occur.
- All unit test methods must contain "ASSERT" method calls since it is the only true way of testing the outcome of your code execution.
- Use the RunAs() method to test user data security where appropriate see runAs method.
- Remember to test interfaces and integrations.
- Remember to use large data sets to test for any impact on the governor limits.
- Consider using TestVisible annotation which allows test methods to access private or protected members of another class outside the test class.

Notes:

- Review System and Test Classes within the Apex Code Developer Guide.
- At specified intervals, typically the end of each sprint, a security scan should be run. Analyze the results of the scan for any false positives. Discuss results of the scan with the Product Owner, and prioritize resulting actions into the order backlog.

Configuration – Best Practices

The majority of the configuration will be tested using automated web testing tools, described below. An automated tool creates testing scripts. The outcomes of running these scripts are recorded. The scripts can then be replayed to verify that the capability has not altered due to other enhancements. Also, editing the scripts easily allows testing of different record types and page layouts associated with different profiles. In addition, some key configurations can be tested via Apex test classes, including:

- **Workflow**: Apex classes can be developed to simulate the creation of tasks and field updates workflow rules. You can write a test class that inputs a record and then by using ASSERT you can verify the new task has been added. Be aware there may be timing issues so it is recommended using test.starttest() and test.stoptest() around the data creations.
- **Verification**: In the case, you would create a test class to input data and then use ASSERT to verify the record has to be created. In the case of negative data, you need to have a try block and then test from the DML exception.
- **Formulas**: When creating Apex test classes for formulas fields, you need to remember that formula fields are only calculated one the record has been inserted or updated, but the approach is similar to Workflow rules.

Visualforce – Best Practices

Customers creating Visualforce pages usually have Controllers and/or Controller Extensions, which can be tested using Apex test classes. The UI components should be tested using a web scripting tool.

Test to ensure HTML and Javascript meet the security requirements by carrying out these tests:

- Cross-Site Scripting (XSS).
- Cross-Site Request Forgery (CSRF).
- SOQL Injection.
- Data Access Control

Note It's important to have test scripts that not only test the Visualforce Page but also test all the code paths through the Controller or Controller Extension.

Lightning Components – Best Practices

Since Lightning Components are independent web components, these components require their own testing strategy and Salesforce Lightning Inspector can help:

- Easily inspect and navigate the structure of your components.
- Identify performance bottlenecks by looking at a graph of component creation time.
- Debug server interactions faster by monitoring and modifying responses.
- Navigate the component tree, inspect components and their associated DOM elements.
- Track event firing and handling sequences.

If you use a component within your applications, you should test its use using the normal techniques. Also, there is a new Lightning Linting Tool which flags issues it finds with your Java Script.

Mobile - Best Practices

Mobile testing involves a number of extra dimensions you need to consider, such as:

- User input is different; e.g. finger, not mouse, and can include gestures
- Orientation (portrait/landscape)
- Proliferation of mobile devices
    - OS version fragmentation.
    - Multiple platforms (iOS/Android/Windows).
    - Device size & Resolution differences.
    - Widely differing performance capability and memory capacity.
- Network availability and performance
- With Summer 16 Salesforce will offer native Offline support, this adds another dimension to the testing matrix
- Testing VisualForce Mobile Pages
- Salesforce1 Mobile App Development Guide

Benefits of Automated Testing

- **Reliable**: Tests perform precisely the same operations each time they are run, thereby eliminating human error.
- **Repeatable**: You can test how the software reacts under repeated execution of the same operations.
- **Programmable**: You can program sophisticated tests that bring out hidden information from the application.
- **Comprehensive**: You can build a suite of tests that covers every capability in your application.
- **Reusable**: You can reuse tests on different versions of an application, even if the user interfaces changes.
- **Better Quality Software**: You can run more tests in less time with fewer resources.
- **Fast**: Run tests significantly faster than human users, hence regression testing can be carried out regularly, so defects are found quicker.
- **Cost Reduction**: Fewer resources are required for testing.
- **Agility**: Code and configuration can be refactored with confidence, thereby removing technical debt.
- **Quality**: Complete test cycles can be carried out quickly, so break-fix releases can be delivered more often if required.

Automated testing can be used for:

- **Functional** – testing that operations perform as expected.
- **Regression** – testing that the behavior of the system has not changed.
- **Exception or Negative** – forcing error conditions in the system.
- **Performance** – providing assurance that the performance of the system will be met the requirements of the Business.

Test Data

When testing your application it's important to ensure your test scripts have the appropriate test data. This can be achieved via

- The test data is inserted during the running of the test script
- After the Sandbox has been created, use a tool to load the test data. There are a number of useful tools (This technique is useful for Developer and Developer Pro Sandboxes)
    - Data Loader from Salesforce.
    - sfApex which is more powerful and can obfuscated.
    - Your corporate ETL tool can also be used.

Useful URL's

- 10 Principles of Apex Testing
- Testing Best Practices

Testing Tools

It is recommended all scrum teams that use the same testing tools and you may wish to select multiple tools.

Considerations for selections of a Testing Tool

- Corporate standards.
- Cost.
- Capabilities.
- Ability to automate Test Scripts.
- The skillset of current scrum teams.
- Hosted on premise or cloud.
- Support from Vendor.
- Etc.

However, before the process of selecting tools, the CoE Core Management team need to define their own specific criteria for selection. (The information presented is summarized from the vendor's web sites.)

The follow tools will be discussed:

- Selenium.
- Cucumber.
- UFT (Unified Functional Testing).
- Project Cinnamon.
- Silk Test.
- Jmeter.
- LoadRunner.
- WebLoad.
- Sauce Labs.
- BrowserStack.

- Appium.
- Provar.

Selenium

Selenium is a set of different software tools each with a different approach to supporting test automation. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium's key features is the support for executing one's tests on multiple browser platforms. Selenium is open source technology.

Selenium 2 (WebDriver)

Selenium 2 is the future direction of the Selenium and the newest addition to the Selenium toolkit and is based on the original WebDriver technology. Selenium 2.0 supports the WebDriver API and underlying technology, along with the Selenium 1 technology underneath the WebDriver API for maximum flexibility in porting your tests. In addition, Selenium 2 still runs Selenium 1's Selenium RC interface for backward compatibility.

Selenium 1 (Remote Control)

Selenium RC was the main Selenium project for a long time before the WebDriver/Selenium merge brought up Selenium 2. Now Selenium 1 is deprecated and is not actively supported (mostly in maintenance mode).

Selenium IDE

Selenium IDE (Integrated Development Environment) is a prototyping tool for building test scripts. It is a Firefox plugin and provides an easy-to-use interface for developing automated tests. Selenium IDE has a recording feature, which records user actions as they are performed and then exports them as a reusable script that can be later executed.

Selenium-Grid

Selenium-Grid allows the Selenium RC solution to scale for large test suites and for test suites that must be run in multiple environments. Selenium Grid allows you to run your tests in parallel.

Choosing Your Selenium Tool

Many people get started with Selenium IDE. If you are not already experienced with a programming or scripting language you can use Selenium IDE to get familiar with Selenium commands. Using the IDE you can create simple tests quickly.
It is not recommended do all your test automation using Selenium IDE. To effectively use Selenium you will need to build and run your tests using either Selenium 2 (recommended) or Selenium 1.

Cucumber

Cucumber is a testing framework based on Behavior-driven development, where you start by writing the test cases in plain English before you start coding. It merges user stories and

tests documentation into one cohesive document, which encourages closer collaboration between the scrum team and the business. There is comprehensive reporting available, however, you do need to use a 3rd party web scripting tool.

UFT (Unified Functional Testing)

HP Unified Functional Testing (UFT) software is HP's main automated functional testing tool and incorporates the features of various important legacy products such as QuickTest Professional, WinRunner, and HP Service Test. UFT automates functional tests by recording the actions of a user on the system under test and replaying the actions on demand to execute test scripts. Supports:

- Automatic conversion of manual test to automate regression tests
- Continuous testing
- API and Web Services testing
- Cross-browser testing
- Mobile testing
- Test scripts can be stored within a source control system and be versioned

Project Cinnamon

Cinnamon is a Salesforce application that enables you to build and run Selenium tests to validate your custom UI pages with Visualforce or Javascript in your Salesforce Org.
With Cinnamon, you can

- Create and execute Selenium Tests from within your Salesforce Org.
- Get out-of-box integration with Sauce Labs, which provides comprehensive OS and browser coverage.

Note: This project was originally developed by a Salesforce Engineer and has been open sourced, however, there has been no update for 12 months.

Silk Test

Silk Test originally developed by Borland portability enables users to test applications more effectively with lower complexity and cost in comparison to other functional testing tools on the market. Silk Test's role based testing enables business stakeholders, QA engineers, and developers to contribute to the whole automation testing process, which drives collaboration and increases the effectiveness of software testing.

- Cross Browser Testing.
- Complete test automation.
- Keyword-driven testing.
- Mobile Support.
- Role Based Testing.
- Very mature product.

Jmeter

The Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. JMeter may be used to test performance both on static and dynamic resources (Web Services (SOAP/REST), Web

dynamic languages). It can be used to simulate a heavy load on a system, network or object to test its strength or to analyze overall performance under different load types. You can use it to make a graphical analysis of performance or to test your server/script/object behavior under heavy concurrent load.

LoadRunner

Hewlett Packard LoadRunner software is the industry standard software for performance testing. Enterprise applications are becoming increasingly complex. With modern applications, there are many moving parts that can easily become points of failure if not tested prior to deployment including Salesforce. Platforms such as mobile, Cloud, and hybrid environments offer their own share of challenges. LoadRunner software is a comprehensive solution for testing system behavior and performance. It provides an efficient and robust means to verify that your application's architecture is built for more efficient performance and reliability. LoadRunner helps you:

- Test a broad range of applications, including the latest Web and Mobile technologies.
- Run high-scale tests.
- Identify end-to end performance bottlenecks using advanced monitoring and analysis tools, and ensure that new or upgraded applications meet the performance requirements of your business.
- Comprehensive Analysis and Reporting.
- Continuous Testing Support via Jenkins, Eclipse & Selenium.

WebLoad

WebLOAD is an enterprise-scale performance and load testing tool. It simulates heavy load in a broad range of web, mobile, and cloud environments is designed as an open framework platform to integrate with any tool and technology and provides actionable intelligence to rapidly identify and resolve bottlenecks.

- The Integrated Development Environment (IDE) enables generating load test scenarios via recording and scripting.
- The WebLOAD Console creates test definitions and executes tests while charting results in real time, querying server performance data, and saving measurements for analysis and reporting.
- WebLOAD Analytics charts results in a coherent report that communicates what happened during a test. The charts support many forms of analysis including root cause.
- The Web Dashboard is an embedded web server that allows multiple stakeholders to share and examine test results.

Sauce Labs

Sauce Labs is for manual and automated functional testing of your web and mobile applications across multiple browsers.

Automated testing with Sauce is based on Selenium WebDriver, at the most basic level, Sauce Labs provides a Selenium Grid that enables you to run multiple tests for multiple browser configurations from the same test scripts. For each test that you run on Sauce, they

spin up a pristine virtual machine (VM) to match the desired capabilities of your test (such as the operating system, browser, and browser version). You can write your test scripts in any one of several popular languages, and then use them to run tests on your local machine on the browsers in the Sauce Labs testing cloud. All you need to do is provide your Sauce Labs authentication credentials and the desired capabilities for your test, along with the path to the site or application you want to test, and Sauce Labs does the rest.

Mobile Application Testing on Emulators and Simulators Made Easy with Appium
Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS and Android platforms.

BrowserStack

BrowserStack allows the testing of your web application across 700+ desktops on an exhaustive range of browsers and operating systems with different screen sizes and resolutions. They provide real browsers installed on real devices.
They also offer the ability to test your UI design and layouts on 100's of mobile devices.

Appium

Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS and Android platforms. Native applications are those written using the iOS or Android SDKs. Mobile web applications are web applications accessed using a mobile browser (Appium supports Safari on iOS and Chrome or the built-in 'Browser' app on Android). Hybrid apps have a wrapper around a "webview" – a native control that enables interaction with web content. Projects like Phonegap, make it easy to build applications using web technologies that are then bundled into a native wrapper, creating a hybrid application.

Provar

Provar is developed by a Salesforce Consulting Partner that have been implementing Salesforce projects for over six years and experienced firsthand how difficult it is to test Salesforce projects. None of the existing vendor tools can reliably test the UI especially complex Visualforce pages without developing scripts. It offers the following capabilities:

- Innovative Test Builder - 'Records' test steps interactively and produces well-structured maintainable test cases.
- Combines API and UI Testing - Provar allows API and UI Test Steps to be combined in a single Test Case. Use API Test Steps to create and assert data quickly
- Service Cloud Testing - Provar manages the complexity of the menus and tabs in Salesforce Console applications
- Visualforce Testing - Generated element Ids for the fields change when superficial changes are made to a page.
- Multi-Browser Testing - Provar uses the industry standard Selenium drivers to interact with Chrome, Firefox and Internet Explorer. Different screen resolutions can be tested using Provar's in-built browser configuration management.
- Data Driven Testing - Provar integrates with Excel and Databases to read and write data.

- Reporting - Generate PDF reports with embedded Screenshots and links to test data.
- User Extendable - You can extend Provar for bespoke testing needs by creating custom test APIs.

About the Author

James Burns is a Senior Director – Solution Architect at Salesforce and is the Global SME (Subject Matter Expert) on Salesforce Governance and works with customers on building their governance frameworks globally.