AUTOMATED SOFTWARE TESTING FRAMEWORK "STASSY"

Denislav Leterov PhD student, Svetoslav Enkov assoc. professor
Faculty of Mathematics and Informatics, Plovdiv University “Paisii Hilendarski”, Bulgaria
denislav.letterov@uni-plovdiv.bg, enkov@uni-plovdiv.bg

Abstract: Over the last two decades, software testing has expanded in terms of both the tools used to perform different types of activities and mind-set of the people using those tools. Automated testing involves use of strategies and instruments that perform testing activities of a software product and where the interference of a person is mainly related to the research of the results of these activities. Automated testing processes are popular because of the benefits that make them an indispensable tool in everyday work for testing activities. The main advantage of automation is the abstraction of rehearsal processes within the tested system. In this way, a complete separation of the human operations from the system is achieved, only the functionality that needs to be programmed. This article presents the development of an automation-based testing framework "Stassy" - System Stable Testing, which supports and helps to implement easily new tests related to Web platforms. The implementation includes the following structure: Object repository, Functional libraries, Global variables and Constants, Data Provider, Test scripts, Configuration files, Recovery scenarios, Loggers, Clean-up scripts and Reporting mechanism. With the advantages of Page Object Pattern and using of the following technologies: Java programming language, JUnit and TestNG extended libraries. The created scripts represent automated acceptance, functional and non-functional tests in which Page Object is used to separate those tests into individual fragments and subsequently call them in a different order according to the automation requirements and business logic. We will track the overall preparation and creation of "Stassy" automated framework and explain the benefits of such approach.

Keywords: AUTOMATION, FRAMEWORK, TESTING, QUALITY, ASSURANCE

1. Introduction

Nowadays software testing technologies provide a lot of testing efforts and solutions, but mainly they are concentrated to present the idea of general answer to the issue, and they are not specified with the current requirements of one particular application. So, of these conclusions we have to say that there is a need for providing an adaptive solution for proposing a set of testing efforts oriented to the current needs for applications of this nature. For example, there are a lot open source specialized tools for automation purpose, but separately they are insufficient to fulfill the whole software development life cycle. In this context, the automated software framework provides flexible solutions for the needs of the software industry.

The initial setup on various projects took a lot of time. At the end of the day, we all go again through similar issues. We resolved this by creating a framework to use it as a template for every new project, offering us the leverage to avoid all of the known problems. When automated tests are created, the first thing we do is to interact with the browser. This can include navigating to a page, clicking a button, or filling in a login form and many different actions. After that, we need to verify and report the actual versus the expected result. While we have many different tests at our disposal, how and when we use them is dependent on the scenario. In some cases, we’ll execute several tests in a specified order. In others, only execute specific tests. In order to achieve all of this, testers usually need to implement different frameworks or libraries along with the Selenium Web driver. [1]

2. Problem area

As we said most projects have common user actions that need to be accomplished in an automated flow. These interactions are developed and implemented in the framework itself and the testers can use them right away without wasting time to write or re-write them again from the start. Another advantage is that we can easily read data from files, for this purpose a Data Provider stub has been created. For example, we can have a list with username and account details credentials which we want to use when we execute tests. With advantages of the framework, there is no need to waste time developing this functionality since it is already done generally. Along with all positives, this development is integrated and extended from Selenium internal frames [2]. This gives us the flexibility to easily choose which tests we want to execute: whole test suite / scenario, or only smoke, sanity tests and etc. also other perspectives or set of tests. It supports data-driven behavior testing and flexible configuration setup.

Main areas of automation needs in an example Web application divided by business logic:

- Main Pages/Re-Direction/Links Testing;
- Categories, Details and Search option Testing;
- Content Detail and Management Testing;
- Payments Testing if is applicable;
- Analytics, Reviews, Ratings and User Profile interaction Testing;
- Promotions/Deals/Ads Testing;
- Social Media Integration and SEO Testing;
- Browser/Device Compatibility Testing;
- Integration Testing;
- Globalization Testing.

Problems that can occur are different, more from less, here are the most common cases that can happen while we are developing our software product:

1. Frequently Changing Requirements

Considering the latest technologies emerging these days, our web applications need to be upgraded. This calls for a change. This change can be related to any new version, integration with the third party tools or maybe sometimes, there are new functionalities that we want to implement in our web application.

2. Increasing Complexity of Testing Web Elements

Latest web functionalities that we implement in our web applications can include various web elements. Those elements can be embedded frames and other products as well. Sometimes, large enterprise websites contain complex flowcharts, diagrams, maps, etc. These make the website’s UI test automation complex.

3. Handling Multiple Errors

Error handling has been an issue with user interface (UI) automation testing. Whenever there are complex UI test scenarios with tight deadlines, most time is utilized in creating test scripts. Thus, testers choose manual testing over automation for UI testing. Error handling becomes extremely difficult when you manually revoke the error messages and automate the same.

4. Maintaining Automated Scripts

Web developers often make changes to UI rather to the logic of the features and functionalities. With this, the UI test scripts fail each time there are new changes to the UI. Hence, maintenance of the UI scripts has been a challenge for long. [3]
3. Framework Architecture

The Framework can support and help the flow of making data-managed scripts more compact and less susceptible to failure. Utilities can also facilitate gradual and manageable conversion of existing scripts. We have functions for each common step under a common library and we call it in the test scripts when it is necessary, it also contains data file that is provided from external storage, such as SQL, CSV or other external resource files or any data carrier from which we can retrieve the necessary records to test the software.

Custom exception package

This package contains user-defined exception class that expands the primary class that is used for processing when an exception occurs during the execution. It is used to provide a simple message for quality assurance (QA) specialists about the logging of errors at the time of interruption. Generally, this helps to understand the failure and the debugging can be more effectively. [4]

Utilities package

In this folder, we captured different Web elements and web element identifiers from the Data Object Model (DOM) in a form of specific variables stub content that can be accessed everywhere in the Framework. [6]

Reusable functions package

The folder is made up of classes that contain functions and methods that can be shared between several classes. It is very often expected to perform pre and post-test operations, including tools, information creation, methods, procedures, and operations. Thus it is always recommended to create a separate class for such activities instead of coding them repeatedly inside each of the test script. This may lead to a delay in automatic tests.

Test cases specs package

The "specs" folder is the main set of the tests and all folders which represents the modules of tested software. Each of these folders is composed of test scripts specific to the associated module (application area). The test spec is a logical mix of more than one test, so the end user can note acceptance criteria of each test script in the package which he wants to perform in every run. He can arrange the scenario in the way he wants to test the specific behavior, feature or implementation.

Object repository package

This folder contains classes that will call the basic XML file for evaluation of the test scripts. They are a TestNG XML file. This can be used to set the flow priority to run test cases. This will be the first class which will be fulfilled. [4] [5]
Reporting mechanism functionality

Reporting mechanisms are the most important part of any testing purpose. This action provides valuable information to the user, who can calculate the current status of the Web application and also the accuracy and the reasons for failures after test execution. On the other perspective, there is a need to identify any measure to eradicate errors in case of loss. TestNG implies generating a different report type from test execution. This includes HTML and XML reports. TestNG may also allow its users to write their own Reporter class with individual custom methods. In this situation we developed our own way for handling report’s stage and we have the freedom to perform generation of it as the way we want. [2] [4]

Library compressed logic

In the Library package folder, we have stored all the jar files that are required by the project needs. These jar files are used by Java classes. They are presented in ZIP archive format and are used to sum up multiple files in one basic file. JAR’s are used as common backup tools, and the main motivation for development is that Java applets and their components (.class files, images, sounds and packages) can be downloaded via browser with an HTTP protocol copy instead of opening a new connection each time. In our project we have used Selenium and TestNG custom compact JAR’s to access the functionality of these tools. [2] [4]

4. Benefits

Using automation framework provide several of benefits such like code re-usage, higher portability, easy maintainability, reduced script maintenance, low cost and many more features. In this developed Framework, mainly the BaseClass will be executed on first place which calls customized XML suite file, the main purpose of it is to arrange and prioritize the order of test case/specs execution. When the first test case is executed it will load the predefined functions which are retrieved from a pool of reusable methods package. They are developed to be used in many places, which include: clicking elements, buttons, entering text to box values, select values from drop down boxes, focusing and retrieving elements and content with JavaScript and many more actions from Web perspective. These functions are called in the customized package class, which as the reusable functionalities like login page, register page, home page and etc. can be used for reusable purpose. They take the values from ObjectRepository.class and Properties file. ObjectPatternRepository has the variables of all web elements and Properties file has details of values which are needed to populate different kind of forms, application URL addresses, Credentials data and etc. If an error occurs in the execution flow we track and capture this issue in CustomException class. We extended Exception.class and used it to handle the exception which occurs from this execution and provides a meaningful message on every error.

In test suite QA specialist needs to login to Web platform, test internal functionality of the software product and run smoke tests related to the main activities of the application. To avoid repetition, there are main functionalities developed in the utility tool file. Each action is indicated and implemented in a step of actions.

In the very end, one spec of tests looks in the following way. Those tests represent empty login functionality with negative behavior.

```java
[Test]
public void Empty_Username_Validation()
{
    Pages.Login.LoginAs(string.Empty).WithPassword
        (TestData.UserPasswordHosted).Login();
    Assert.AreEqual("Username is required.",
        Pages.Login.GetEmptyFieldError());
}
[Test]
public void EmptyPassword()
{
    Pages.Login.LoginAs(TestData.User).WithPassword(string.Empty)
        .Login();
    Assert.AreEqual(TestData.PasswordEmptyMessage,
        Pages.Login.GetEmptyFieldError());
}
```

The annotation for “Test” is taken from the configuration in which group of methods will be executed.

Each method in the class represents an assembly of all inherited and abstract classes. This presents a more convenient way for framing and integrates elements with operation in the spec class for arranging and structuring tests.

![Figure 5 Architecture of the Framework](image)

![Figure 6 Activity Diagram flow of a Login Functionality](image)
In Figure 7 is listed comparison from the traditional basic Selenium Web Driver approach versus our custom created framework called “Stassy”. We can see overall time duration on tests execution against Google Web application (the whole package apps) environment. With our framework we managed to save approximately one hour and a half, which is very important in one testing argument. This is because of the custom developed packages, classes and methods.

5. Conclusion

In this paper we have proposed the “Stassy” - System Stable Testing framework approach to test Web applications configured and extended on Selenium Web Driver basis. The proposed automation framework reduces the required time to write and run test cases and increases pass percentage rate of them by covering all the main steps in applications of this kind. It also reduces vulnerable workload of testers. With the usage of this framework, specialists can implement it over any application oriented to Web platforms by configuring the classes and main methods by their needs. With the custom developed packages such as: Controller, Asserter, Logger and Data provider stubs, we easily can generate customized actions, test reports, errors handling and also analyze the failures using screenshots of failed test cases and manage to trace the issue precisely. QA specialists can maintain all the data from centralized stored place. This framework is useful for dynamically changing Web applications, which implement new features very often. Developments of such kind are the future of software quality assurance automation process, recent years trends have shown that solutions of this nature are needed for complete application security and comprehensiveness. Automation test scripts are easy to handle and are understandable. In this way automation framework helps the companies to test Web applications more accurately and efficiently.

6. References


For contacts:
UNIVERSITY OF PLOVDIV PAISII HILENDARSKI, 24 TZAR ASEN, 4000 PLOVDIV
Denislav Lefterov, PhD student, Faculty of Mathematics and Informatics
E-mail: denislav.lefterov@uni-plovdiv.bg