SANAE
GO LIVE with us

SANAE, s. r. o.
Kopanice 3570/6
821 04 Bratislava

Nalgoo Automation Framework

Summary

For a fast growing project, we decided that the best solution would be a simple framework that would meet all the customer needs and at the same time allow easy maintenance. And that is what we have created.

Keywords: selenium, automation, framework, java, UI tests

Solution

How we handled initial system state and preconditions

Every test scenario has some preconditions, some initial state or some data that are required before the test can be executed. This can be achieved by having some initial state of database that stores data for all test scenarios. But this approach is time consuming, because restoring bigger databases takes more time, and also difficult to maintain. Updating one record in your database dump can prove challenging. So we decided that every test will create its own data. This will not only save time if you want to execute only few test scenarios but also make the tests independent on initial state.

How we achieved easy maintenance

To make maintenance easy, we split the implementation into 3 layers.

Mapping layer

This is the lowest layer there is, the backbone of our solution. In this layer is stored all mapping to elements of the UI. Every textbox, every combobox, every button, every table… everything. Each of these elements will let you perform some basic operations on them, like finding element on page, writing some value into element, clicking on element, etc. If mapping of existing element changes, this is the only layer that needs to be updated. If a new element in UI was added, it must be added to this layer as well.

Operations layer

This is the second layer that is above mapping, the muscles of our solution. Here are stored some basic operations. Examples are: open new form, fill form, submit form, search for record, open record detail etc. If the logic of operation changes, this is the layer you need to update. If a new element is required or if the order of actions changed or if some action is now obsolete.

SANAE, s. r. o.
Kopanice 3570/6
821 04 Bratislava

Test scenario layer

This is the layer where actual test scenarios are implemented. This is the brain of our solution. On top of splitting scenario implementation from mapping and UI logic, it made writing new scenarios much easier, as the code looks really simple and does not require knowledge of selenium. This layer requires little to none maintenance, as the tests mostly remain intact regardless of changes to underlying layers. The only time you need to make changes in this layer is when some functionality undergoes major changes and the test scenarios are no longer valid as a sequence of operations. Or some functionality becomes obsolete altogether.

How to do a useful reporting

Every time you find a bug you need to report it so that someone else is able to reproduce it. For this you need to gather enough information about what happened. Not just that the test failed. To achieve this, we added into the mapping layer for every access, for every click, for every value that is stored into element few lines of code that store this action into log. After test scenario is executed we write results with all stored actions into database. From here it is just a matter of displaying them in useful way to anyone looking at them. And how better to do this than by filtering. There are multiple types of messages stored in log. Some are errors, some are operations, some accessing elements on UI. You decided what information you need to identify the problem. On top of that, framework lets you store screenshots of UI on demand or when element is not found. This helps you identify error messages that mostly popup and block UI.

How to increase performance

Every set of regression tests always does one thing. Grow. And that is a good thing. More tests will give you better results. But more tests also means more time executing them. To tackle this problem, we added parallel execution of test scenarios. This way the time required to execute tests can be drastically reduced. The only downside is, that you need more hardware to do it. So to make it also possible to run tests when you are short on time, you have the possibility to select only those test that are really essential. And since all tests scenarios are independent you just pick the ones you need.

Benefits

Different test suites let you save execution time and finish tests on schedule. Easy implementation of new test scenarios does not require developers skilled with selenium. Easy maintenance when elements are moved around or removed and added to UI. And since execution is automatic, no supervision is required while tests are executed.