Liferay Inc.

TDD - Good Practices Guides

1. TDD - Test Driven Development

The idea of TDD is that you start writing a unit test that fails, then you code until it succeed and repeat that cycle until the work is done.

2. F.I.R.S.T Principles for Writing Good Unit Tests

The idea of F.I.R.S.T is that can help make your tests shine and ensure that they pay off more than they cost.

Acronym FIRST stand for below test features:

- [F]ast
- [I]solated
- [R]epeatable
- [S]elf-validating
- [T]imely

Please, check this LINK more information.

3. LFRGS Selenium Commons Setup (how to prepare your environment)

Gradle's tasks should be executed to perform your setups. All informations and steps you can find on README.md from your project (.../modules/test), such as:

- which tasks you need to run (to create a functional tests properties, create a environment configurations, download and automatically setting webDrivers: Firefox Driver and Chrome Driver Headless, etc);

4. Code Structure

4.1. Page Object Pattern Structure of Functional Tests

This approach is one easy technique to keep your code more understandable and easier to make the maintenance. You should create two initial structures.
The **src/functionalTest/java/** must have four folders/packages:

- **com.liferay.[project].pages:** This path contains all classes mapped between functions per pages, for example, the Login Page, should have all elements and methods that the test will need to perform the actions on the respective page, the same logic is applicable on Welcome Page.
- **com.liferay.[project].utils:** This path have all the things that can be util to the code, for example the class “CommonMethods”, “FunctionalTest”, “RetryTestExecution".
- **com.liferay.[project].tests:** This path has all the tests isolated by their own context, for example, the LoginTest that should have all tests related to the login feature, and each test should be able to be performed isolated.
- **com.liferay.[project].testsSuite:** This path has the only one file, that is the suite class that is able to run all tests.

The **src/functionalTest/resources/** must have all the resources files.

In this kind of organization will be easier to developer or tester, developing the code and looking for the future, realize the maintenance.

5. Common Methods

FRW Selenium Commons

About the shared classes in "Util Class" section, you can know this information, on this link: [Existing Classes on frw-selenium-commons]

6. Why would you use ID attributes

6.1. Clean automation code. Your automation code will be more readable if you get the attribute's locator by ID instead of getting it by XPath or CSS selectors.

6.2. In some cases, it's inevitable to find an element by ID, but if you create a XPath with the ID element, this XPath will be more robust.

6.3. Fastest way to locate elements on page because selenium gets it down tóexecuting document.getElementById().

6.4. Fragility in UI tests is hard to manage, so other issues might also need to be resolved before you see the benefits of adding IDs. This point can achieve everybody's confidence in the team about the automated tests.

6.5. Best discussions and articles about this subject can be found here:

6.5.1. Good practices for thinking in ID and Class name by Google.

6.5.1.1. Google's HTML code has ID's with some_id and also with some-id. I believe that the two one is correct. It must be chosen one way to follow and make the code readable.

6.5.2. Is adding ids to everything standard practice when using selenium

6.5.3. Which is the best and fastest way to find the element using webdriver by xpath

7. Auxiliary tool during test automation process

7.1. Firebug with FirePath - a great help when you need to find XPath expressions, he creates one for you by inspecting the element, work only on Firefox version 50.