Software Testing, SS 2022

Prof. Dr. Reinhold Plösch, Mag. Rudolf Ramler

Assignment 5: System Testing

Deadline: **May 12, 2022, 23:55**

The goal of this exercise is to implement test cases that exercise a system via its graphical user interface; specifically, we will use Selenium for testing via a Web interface.

Note: This assignment is also the basis for the next assignment #6.

Assignment Part A: Implementing Selenium Web Tests

The framework *Selenium* allows you to test a Web-based application implementing your test in common programming languages like Java. Selenium uses a WebDriver to run the tests with a Web browser (e.g., Chrome or Firefox). Details can be found at

- Selenium: https://www.selenium.dev/documentation/en/

To get started, use the code example provided in *SelenumExample.zip*; have a look at the unit test. The example requires the Chrome browser - adjust the code if necessary. Run the example via *mvn test* to make sure your setup is working correctly (the test should pass successfully).

Write another JUnit test *LisssTest.java* to automate a search on the JKU literature search system. Follow the example in *SeleniumExample.zip*, use Selenium IDE, or start from scratch to create the test:

- Go to the page https://lisss.jku.at/primo-explore/search?vid=ULI&lang=en_US
- Perform a search for the string "software testing"
- Check the **title of the results page** and the **number of found titles**
- Note: Create a new Web driver instance opening the browser window in setup (@BeforeAll) and close the driver in the corresponding teardown. Use a suitable driver for Chrome, Firefox, etc.

Run the test case and make sure it passes successfully.

Assignment Part B: Page Object Pattern for Web Testing

The Page Object Pattern mitigates maintainability issues by a clean separation between test code and Web page specific code. See examples at http://blog.activelylazy.co.uk/2011/07/09/page-objects-in-selenium-2-0/ and related readings such as https://github.com/SeleniumHQ/selenium/wiki/PageObjects.

Make use of Annotations and the PageFactory provided by Selenium to abbreviate the code in the Page Object classes as proposed at https://github.com/SeleniumHQ/selenium/wiki/PageFactory.

Please, write your page objects and test case according to following instructions:

- Create two page objects, *SimpleSearchPage.java* and *ResultsPage.java* according to the example shown in the Selenium documentation. *SimpleSearchPage* should implement a method *searchFor(String)*, *ResultsPage* should implement at least a method *getNumberOfResults().*
- Implement the scenario described in part A as JUnit test case (*LisssPageObjectTest.java*) using the two page objects.

Submission & Presentation

Use the project provided in *SeleniumExample.zip* as starting point for this assignment. Add all tests classes you developed in the existing project structure (under *src/test*). To prepare the submission

- Delete the generated folder *target*
- Create a zip archive of the entire project folder *SeleniumExample*
- Make sure the archive includes the source code of your tests and any related files you created

Submit the zip archive by **uploading it to the Moodle course platform before the deadline**.

Work in teams. Only one upload per team is required, i.e., one team member uploads the files to Moodle.

The work has to be **presented in the lecture on Thr, 23. June 2022**. Each team has to be prepared to present their results. The teams that present will be selected at the beginning of the lecture. No slides (ppt etc.) have to be prepared, just bring a laptop with a development environment including all the tools you used and the source code of your solution.