CS 1632 - DELIVERABLE 3: Web Testing with BDD

Hoodpopper Ruby Tokenizer/Parser/Compiler

Michael Bowen

Summary

The first hurdled that needed to be cleared was getting Selenium set up to work in Eclipse. This turned out to be straightforward once the correct version of the .jar files to use was discovered. Adding the .jar files to the build path was the same was with using Mockito, so that was simple enough.

Writing the user stories for this particular assignment seemed to be very easy and very difficult all at once. The user stories that I was able to write up seem somewhat generic, but that might be expected out of user stories as they are less formal. But, because the user stories simply focus on the functionality of the three buttons provided by Hoodpopper, they stories ended up being pretty much the same only tailored toward that particular button.

Coming up with the scenarios that I focused on was relatively simple as well, as the assignment markdown file gave some good examples of what to look for. The test cases that are included in the code are in no way indicative of a complete test suite. I actually worry that, due the nature of the user stories and scenarios, the test coverage is somewhat lacking. All of the tests focus on whether or not the buttons perform their intended function and not on how the site itself functions. For instance, there is no scenario that tests whether or not code written into the code entry screen will remain if the “Back” link is clicked after one of the buttons is used. This is the kind of functionality that it would be important to test, but does not fall under one of the three user stories. I would also have liked to more exhaustively test the different operations/special characters using many different sections of starting code, but fear that would have required too much time and a better-than-rudimentary knowledge of Ruby.

Determining how to enter code into the system via the JUnit tests took some thinking but, once I was familiar with retrieving specific web elements and the functionality included in Selenium, it ended up being a straightforward ordeal. Likewise, it ended up being somewhat tricky to select the correct tags/buttons on the Hoodpopper pages due to the tags not being labeled with ids. It took a bit of tinkering and looking up how to use the By.xpath functionality for selecting WebElements, but I got there pretty quickly in the end.

All in all, I found this testing process to be enlightening. It certainly seems useful to be able to run tests like these on a website without actually needing to use the record/mirror action functionality. It would have been nice to know how to do this last semester for my web development class. While these tests are not sufficient to give me the utmost confidence in the application, I feel they serve well as a decent test base and could be easily expanded upon.

Code for the JUnit tests: github.com/mjb236/CS1632/tree/master/Deliverable%203

Screenshot

