Project: A DSL for browser automation

Browser Automation

In your successful job of Web application architect, you have often to perform repetitive tasks with your browser: operations like logging into systems, filling forms, navigating to specific locations, etc... Your colleagues (especially in the testing department) have the same problem, and some of them are not experienced programmers. You are asked to improve your company productivity by implementing an external DSL for browser automation.

By this DSL you want to be able to express procedures like:

- open a browser window (e.g., Firefox)
- go on "http://campus.mines-nantes.fr"
- click on the link "connexion"
- click on the button "Cliquez ici ou sur le logo C'zam pour vous identifier"
- fill the text field "username" with "mtisi08"
- fill the text field "password" with "12345"
- select the checkbox "warn"
- click on the button "Connexion"
- go to the url « https://campusneo.mines-nantes.fr/campus/course/view.php?id=1571»
- verify that the link "Consulter le profil" contains the string "Massimo Tisi"

Besides basic navigations like the previous example, the DSL should allow users to:

- define loops, e.g. do/while;
- apply operations on collections of page elements, e.g. "select all the checkboxes of the page";
- define conditional flow, e.g. if/then/else;
- read information from a point of the Web page, and use it later in an operation;
- define parametrical subprocedures and call them from other procedures.

Selenium WebDriver

You were starting developing your Java program to do all that, when you discovered that a Java library that performs exactly the operations you need is already freely available. It's Selenium http://docs.seleniumhq.org/.

Selenium 2.0 provides a very simple API to automate browser execution, WebDriver. A short but useful documentation is provided here: http://docs.seleniumhq.org/docs/03_webdriver.jsp

Tasks

Design and implement the browser automation DSL as an external process language in Java, representing a flow of browser operations. Selenium greatly simplifies the task of implementing the semantics of your language, but there is still a lot of work to do:

- design the abstract syntax of your language,
- design a textual syntax for your DSL and generate a textual editor using tools like TCS or Xtext,

    - with syntax highlighting, code assistance, autocompletion, outline, ...
- implement one of these two options:
    - an interpreter for executing your language from the development environment,
    - a compiler to Java, to generate an executable program to deploy in the machines of your colleagues,
- design a graphical syntax for your DSL and create a graphical editor in Eclipse.

The final product of your work is going to be a set of Eclipse plug-ins that, when added to an Eclipse installation, will allow users to create programs in your DSL by using a textual syntax or a graphical syntax, and to execute them.