CS490 Project 4: Web Testing

What you will learn in this project

- Set up and use Selenium, the Web Browser Automation Solution.
  - Control the browser by programming via Selenium WebDriver.
- Analyze the functionality requirements and design the test cases.
- Write the Selenium WebDriver test script to automatically run your test cases.
- Use an existing combinatorial testing tool fire-eye to generate pairwise test cases.
- Integrate the test cases generated with your Selenium WebDriver script and run the pairwise testing.

Create your workspace

In this project, you are going to create automated tests for a web application called **SchoolMate**.

To save your effort, a virtual machine image with the server components and the subject program is provided. You are supposed to install the necessary tools, write your test scripts and run the automated tests in the VM.

1. The open sourced virtual machine VirtualBox will be used. Please download and install the suitable version for your platform.
2. Download the VM image here. The guest OS is Ubuntu 10.10. We have pre-installed apache, mySql, PHP and SchoolMate. Following is the login information in case you need them:

   ```
   [Guest OS]   Username: hi     Password: hi
   [MySQL]      Username: root   Password: 123456
   [phpMyadmin] Username: root   Password: 123456
   [SchoolMate] Username: admin Password: admin (the default administrator)
   ```

   In the VM, SchoolMate can be accessed at

   ```
   http://localhost/schoolmate/
   ```

   phpMyadmin can be accessed at

   ```
   http://localhost/phpmyadmin/
   ```

3. Import the VM image downloaded into your VirtualBox. (How to import?)
4. Boot up the guest OS in the VM.
5. After logged in, open a terminal and install the **Maven** and **Eclipse**

   ```
   $ sudo apt-get install maven2
   $ sudo apt-get install eclipse
   (Use the pwd "hi" when prompted to "[sudo] password for hi" )
   ```

6. Install Selenium WebDriver
   a. Please read and follow the steps in Setting Up a Selenium-WebDriver Project - Java.
   b. Please import the maven project into Eclipse.
   c. Please read and try the test script in Introducing the Selenium-WebDriver API by Example written in Java. It will give you the first impression how Selenium WebDriver Script works.
   In case you got errors in Eclipse...
      - Please make sure you're set to compile against Java 6 or above. (How? go to "Window -> Preferences -> Java -> Compiler", set the "Compiler compliance level" to >= 1.6)

- If you got errors like "Unbound classpath variable: 'M2_REPO/cglib/cglib-nodep/2.1_3/cglib-nodep-2.1_3.jar' in project ..." in Ecplise, go to "Window -> Preferences -> Java -> Build Path -> Classpath Variables -> new", set Name to "M2_REPO" and Path to "/home/hi/.m2/repository". Click "OK"

Reference:

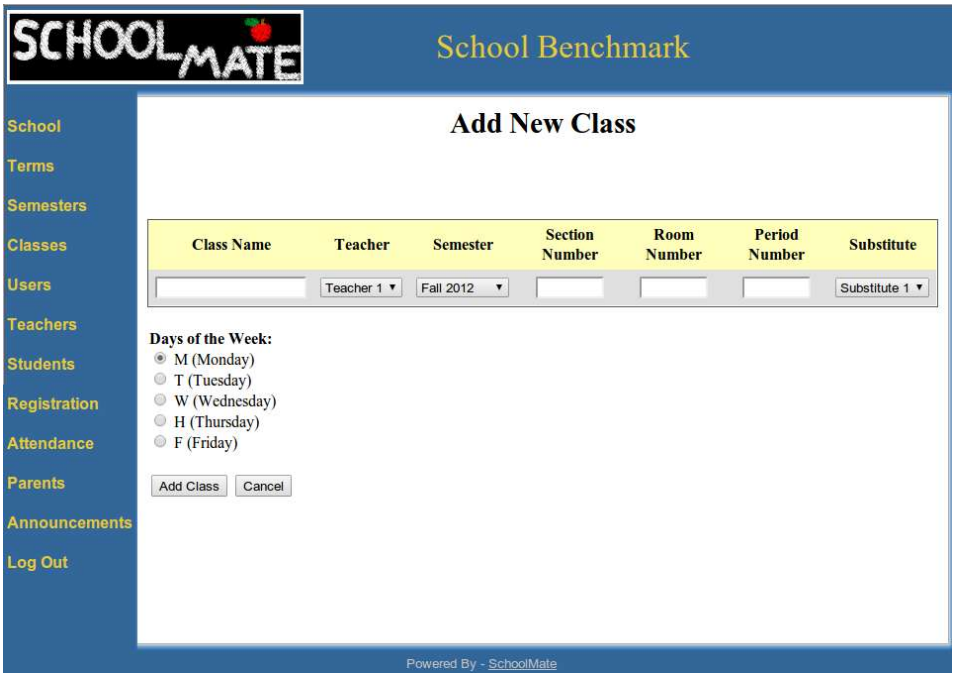
Get started with Selenium 2, End-to-end functional testing of your web applications in multiple browsers

---

Part 1: Analyze the functionality requirement and Design the test cases (9 pts)

You are going to do the **black-box testing** for the functionality of **"Add New Class"** in the SchoolMate.

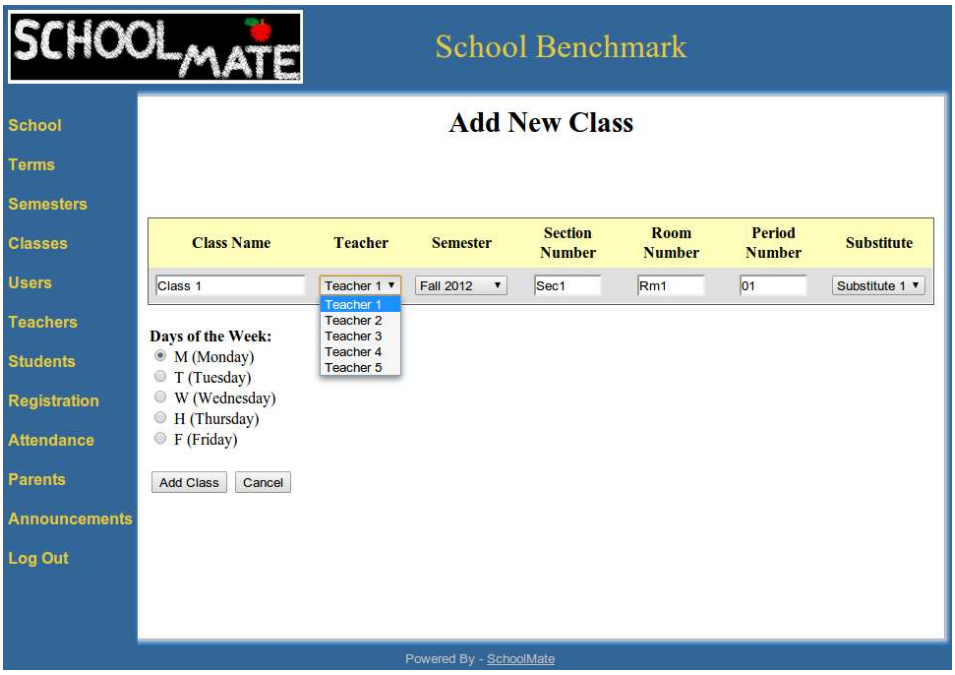
The page **"Add New Class"** will show up after you log in as the **Admin** and click the button "**Add**". In case this page is not there, it can be reached by clicking "**Classes**" in the left menu and then clicking "**Add**". The target page is shown in the following figure.



1.1 Functionality Description of "Add New Class"

In this page, users are supposed to:

- **Fill the textboxs** : Class Name, Section Number, Room Number, Period Number
- **Pick a value for options** : Teacher, Semester, Substitute, Days of the Week



The requirements for the values filled in the textboxs are listed as following.

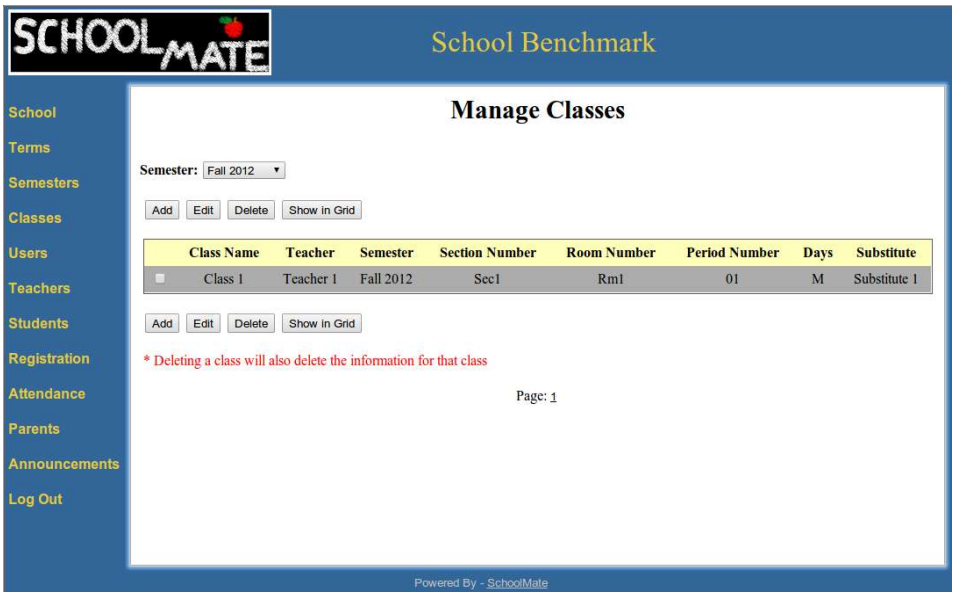
- **[R-1] Class Name** : alphabets and numbers are allowed.
- **[R-2] Section Number** : only numbers are allowed.
- **[R-3] Room Number** : only numbers are allowed.
- **[R-4] Period Number** : only numbers are allowed.
- **[R-5] All textbox fields** : no Cross-Site Scripting (XSS) injection vulnerabilities.

XSS reference :

1. Cross-site Scripting part in the slides here.(Page 9-15)
2. The threats of cross-site scripting, Launching an attack and Sample attack scenarios
3. Alternate XSS Syntax Section in this article

The possible values for the options are listed as following (**DO NOT** remove or modify option values listed).

- **Teacher**: Teacher 1, Teacher 2, Teacher 3, Teacher 4, Teacher 5
- **Semester**: Fall 2012, Spring 2013
- **Substitute**: Substitute 1, Substitute 2, Substitute 3
- **Days of the Week**: M, T, W, H, F



After clicking the button **Add Class**, in the next page, the requirements are:

- **[R-6]** The class record added is successfully shown in the table.
- **[R-7]** The values are exactly the same as those were entered or selected.

1.2 Design the test cases against the functionality requirements (3 pts)

You are supposed to design test cases for the requirements numbered as [R-x] above.

In this step, you **DO NOT** need to consider the combination of values, i.e., you only need to consider one field in one test case.

Write down your test cases in a text file named "input1". In this file, each test case should follow the following format:

```
//Comment. Explain:
//  (1) This test case is designed for which requirement(s).
//  (2) How to ensure the result meets the requirement(s).
["$CName$", "$TCH$", "$SEM$", "$SecNum$", "$RmNum$", "$PeriodNum$", "$SUB$", "$Days$"]
```

The comment is the explanation for this test case. The tuple is the concrete values for each field/option.

For example, "input1" should be like:

```
// (1)For R-j. (2)Will use an assert to check the result whether it ....
["cn1", "t1", "s1", "11", "1234", "1", "name1", "M"]
// (1)For R-i. (2)Will use an assert to check the result. If ... happens, ...
["cn2", "t2", "s2", "12", "1234", "1", "name2", "T"]
```

1.3 Implement the Selenium WebDriver test script template (6 pts)

You are supposed to implement a test script template in **Java**. You only need to test the program in FireFox.

More specifically, your script should be able to accomplish:

1. Accept and parse the input file in the format defined in section 1.2. The comments should be ignored.
2. Enter or select the values in the input to the corresponding fields. For example, in a test case from the input file, "TCH" is "Teacher 2", the script should select "Teacher 2" in the drop down list of Teacher.

3. Click the "Add Class" button to submit.
4. Check the requirements when the result page arrives.

The goal is that your Selenium test script can automatically take and run the test cases in "input1" one by one.

The script should output the results to a file named "output1".

For each test case, if no violation is found, the script should output

```
P ["$CName$", "$TCH$", "$SEM$", "$SecNum$", "$RmNum$", "$PeriodNum$", "$SUB$", "$Days$"]
```

The tuple is the concrete values for the current test case. Otherwise, it should output

```
F ["$CName$", "$TCH$", "$SEM$", "$SecNum$", "$RmNum$", "$PeriodNum$", "$SUB$", "$Days$"] : Problem description
```

to indicate a bug.

Note: Please do not hard code the buggy cases. Your template script will be tested in a different setup. Otherwise, the maximum point you may get for the script is 1.

SUBMISSION (DUE: 11/27 MIDNIGHT)

- The test cases file: input1
- The selenium WebDriver test script implementing the template written in **Java**
- The test result file: output1

PART 2: COMBINATORIAL TESTING AND AUTOMATED SOLUTION DESIGN (6 PTS)

2.1 AUTOMATED COMBINATORIAL TESTING (4 PTS)

In this part, you will need to consider the combinations of all fields (the textboxs, the drop-down lists and the radio button) and run the pairwise testing to cover them.

1. Use an existing combinatorial testing tool fire-eye to generate pairwise test cases.
    - Run the fire-eye.

      ```
      $ java -jar acts_gui_beta_v1_r9.1.jar
      ```

    - Create a system with the field names and the corresponding values.
    - Build the system and export the test cases generated in "Nist format", which is a text file.
2. Generate a input file for your template script: "input2". For fields that are not in the generated test case (e.g. fire-eye may output "don't care" for a field in a test case), simply pick a valid value to complete the test case.
3. Run your test script and output the results to a file named "output2" following the same format defined in previous step.

2.2 SOLUTION DESIGN (2 PTS)

Sketch an automated solution (on paper) that can score a black-friday item (i.e. the item that will get sold in a few seconds) from Amazon.

In your design, you should clearly explain the challenge and how your automated solution works to add a super hot item in your shopping cart asap.

Submit your solution as a file named "solution.pdf"

SUBMISSION (DUE: 12/2 MIDNIGHT)

1. Automated combinatorial testing
    - The test cases generated by fire-eye in "Nist format"
    - The selenium WebDriver test script.
    - The test case input file: input2
    - The test result output file: output2
2. Solution design "solution.pdf"

WHERE TO GET HELP

Yunhui Zheng (zheng16 at cs.purdue.edu) is managing this project. His office hours are

- 11/20, Tuesday, 2:30pm-4:30pm
- 11/29, Thursday, 2:30pm-4:30pm

at LWSN B116B.

We recommend you to use the Piazza for sharing your questions and get answers faster.