**SWEN-352: Software Testing**

# Activity #4

# Selenium: Browser Automation & Testing and Perfecto: Mobile testing

## Objective:

Learn how to:

* Create automated scripts and tests using the Selenium IDE.
* Develop more intensive and use-case scripts using the Selenium API in Java.
* Use Perfecto for mobile testing on real mobile devices in the cloud.

## Dropbox Instructions:

* One zip file submission for this assignment.  
  1 team students = 1 submission. Label all filenames by Team number, i.e., Team\_1, Team\_2, etc.
* All video submissions should either **.mp4**, **.mov** or .**FLV**.
* **Submission Instructions are at the END of the document.**

## Overview:

This activity is composed of **four parts**:

1. **PART 1** - Makes use of the Selenium IDE plugin in Firefox/Chrome to introduce you to Selenium.
2. **PARTS 2, 3** - Involve programming using the Selenium API in Java.
3. **PART 4** - Uses Perfecto MobileCloud for mobile testing.

**SELENIUM**

## Setup for Selenium:

This activity is going to require the use of Selenium, as well as Firefox/Chrome and the Selenium IDE plug-in.

*You may use your preferred Java IDE for this activity, with Java 8+.*

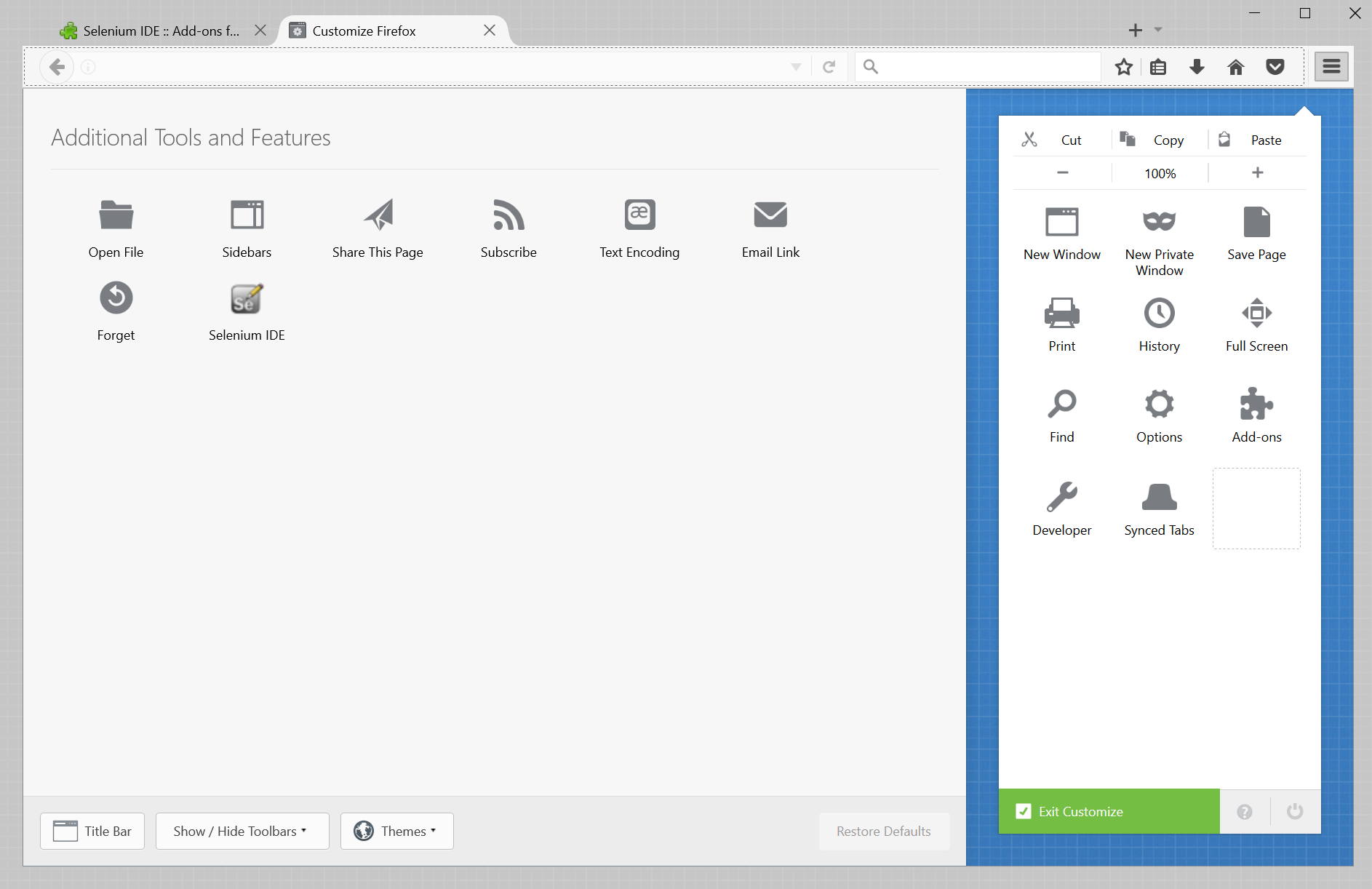
## Setup Selenium IDE for Part 1:

*Note: You may use Google Chrome or Firefox for this part.*

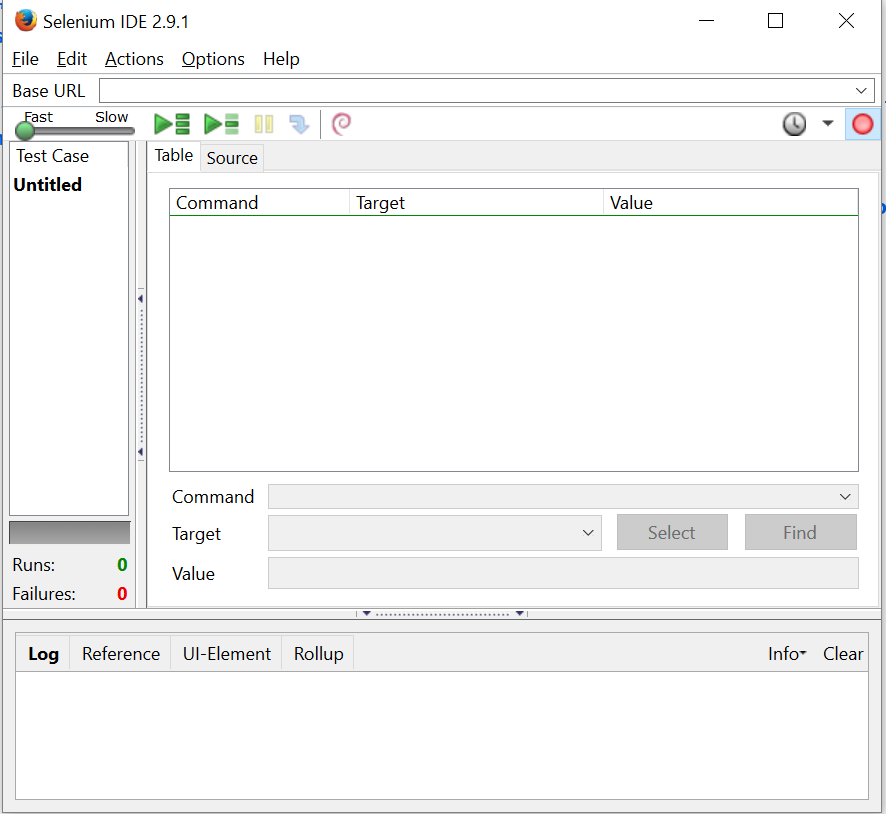
1. If you already have Firefox/Chrome installed on your machine skip to Step 3.
2. Download and install Firefox:
   1. Download Firefox from this webpage - <https://www.mozilla.org/en-US/firefox/new/?f=94> *.*
   2. Install Firefox. (Recommended and default settings are fine for this activity).
3. Inside of Firefox navigate to this [webpage](https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/) and install the Selenium IDE plugin to Firefox (Firefox will require restart to complete the installation, Do this before continuing). Navigate to this [website](https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd?hl=en) for Google Chrome.
4. Firefox will restart and open back on the same page; this is expected.

The rest of the steps are for Firefox only.

1. In the upper right corner, select the “hamburger menu” button to get a dropdown on the right side of the page.
2. Click on the *Customize* button on the bottom of this dropdown.



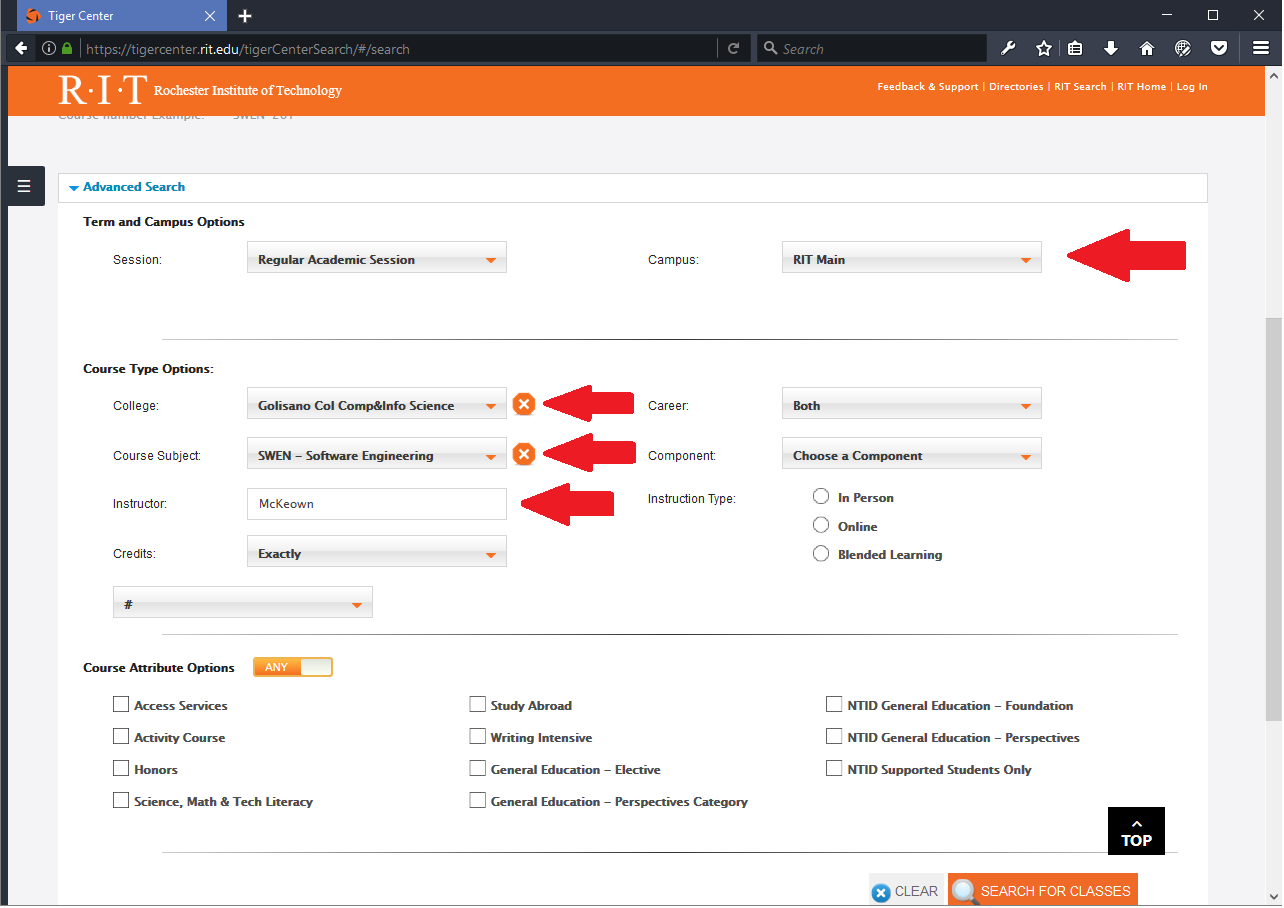
1. Click-and-drag Selenium IDE into the original drop-down bar.
2. Click *Exit Customize* and then re-open the dropdown menu by clicking the hamburger menu in the upper right.
3. Click on Selenium IDE. A small window will open showing the Selenium IDE tool. (The GitHub page for the tool may open, ignore it).



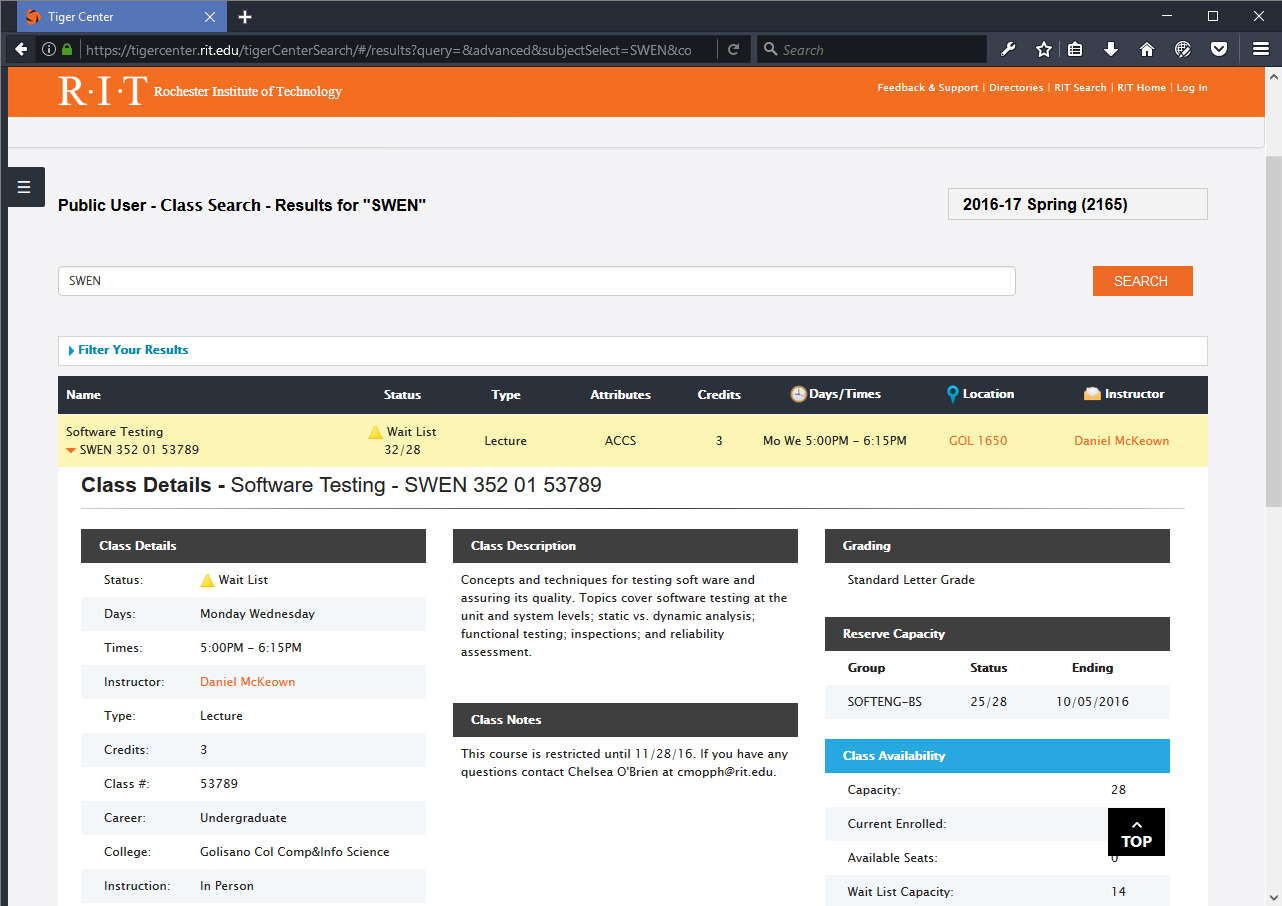
* 1. The tools main content area is going to be taken up by commands generated by navigating around webpages. You may create several different tests with this tool, which will show up in the *Test Case* window on the left side.
  2. The two green triangle buttons at the top allow you to play back either all tests cases in the IDE or the current selected one.
  3. The red circle in the top left of the view is the record button. It currently is set to record all interactions with the Firefox web browser into the current Test Case.

## Activity Instructions for Part 1:

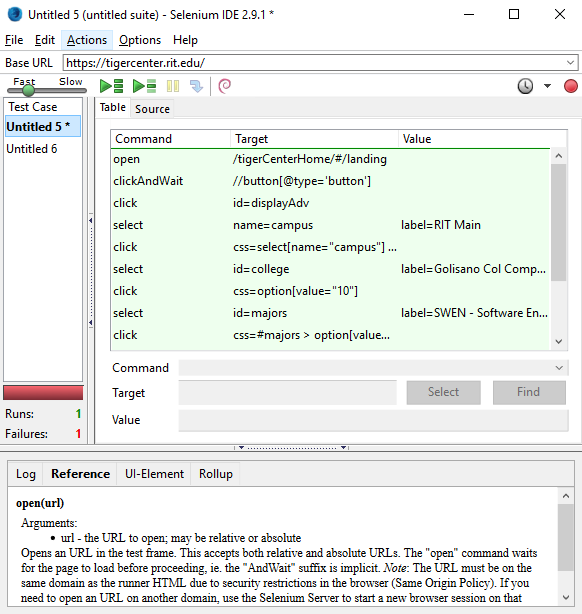
1. In an open Firefox window open the Selenium IDE by following the steps listed in the setup for PART 1 of the instructions. Make sure it is not in record mode.
2. Navigate to <https://tigercenter.rit.edu>
3. In the Selenium IDE, make sure selenium is set to record. The record button should look like , NOT  (coloring of the button, not the surrounding background).
4. Scroll down and select the *Start Searching* button
5. Make sure you have the right semester (this class is not offered in the summer).
6. Click on *Advanced Search* to bring up the Advanced Search drop down and fill in the dropdowns with red arrows matching to this SWEN-352 course, as shown in the following image.



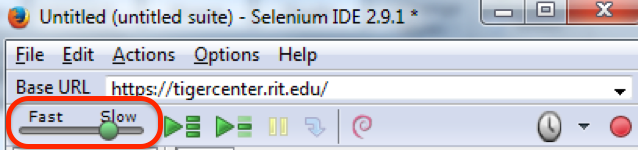
1. Click the *SEARCH FOR CLASSES* button at the bottom of the page.
2. Click on the Software Testing table row to get an image like the one below.



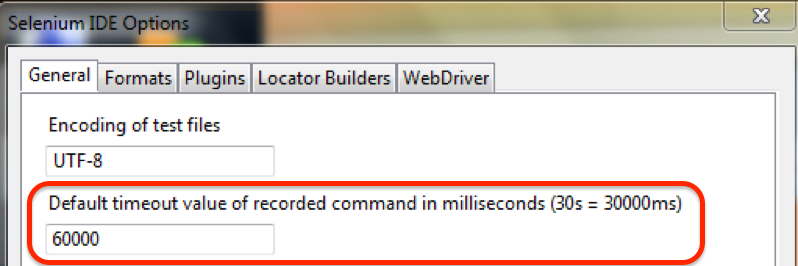
1. Go back to the Selenium IDE tab, and verify that it looks like the following image, if it does not, restart Firefox and try again. This picture is not meant to be exact.



1. Turn off recording in Selenium IDE. Be sure the test works by running the test in the Selenium IDE. Click the second green run button  to run the test we just recorded.  
   The scripts may fail due how fast the script is running, which is usually too fast for the browser to keep up with Selenium.  
   You can control the speed the tests run, suggest running them at least at half speed or slower.

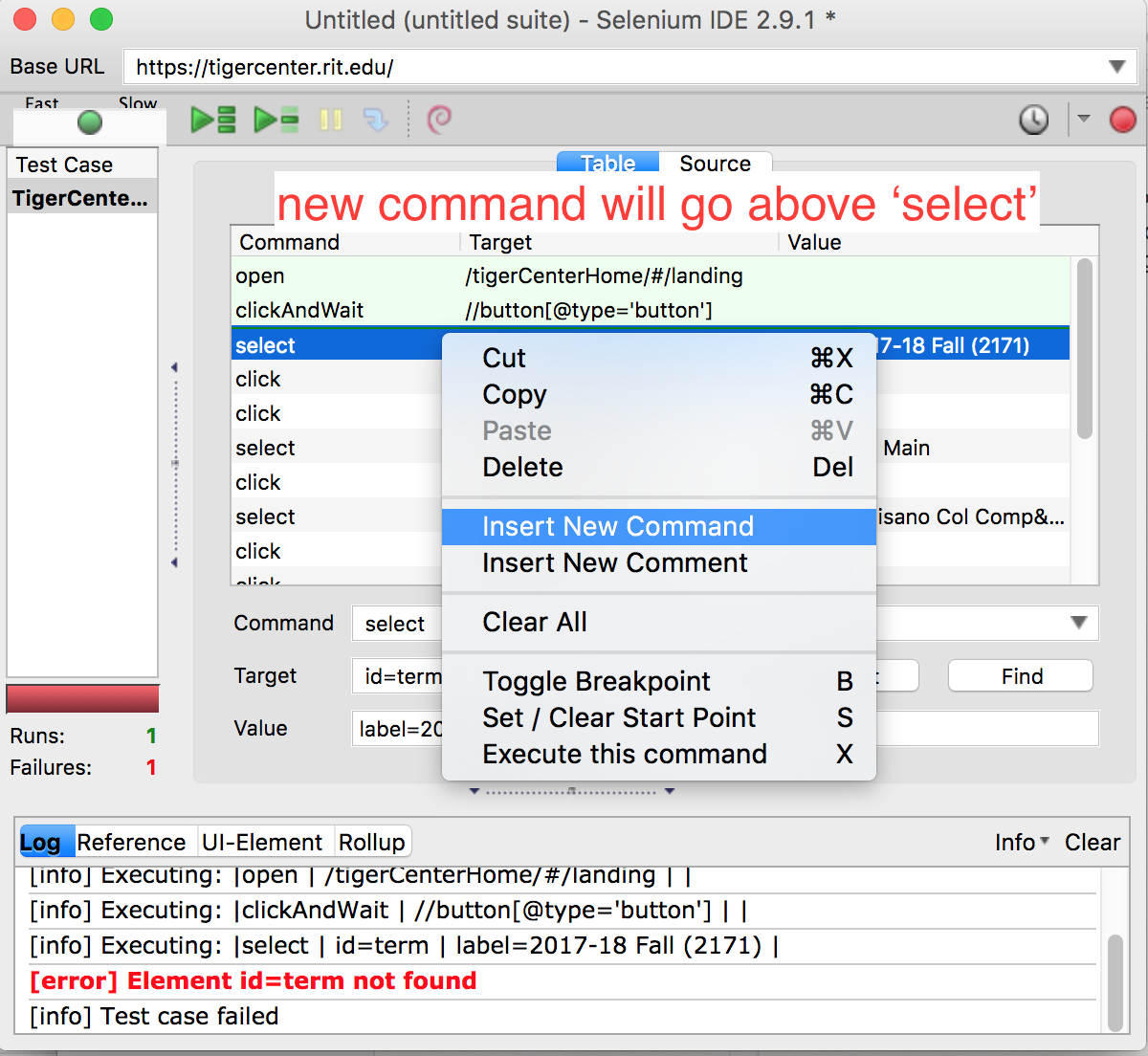


You may also want to change the default timeout from 30s to 60s, Selenium IDE -> Options:

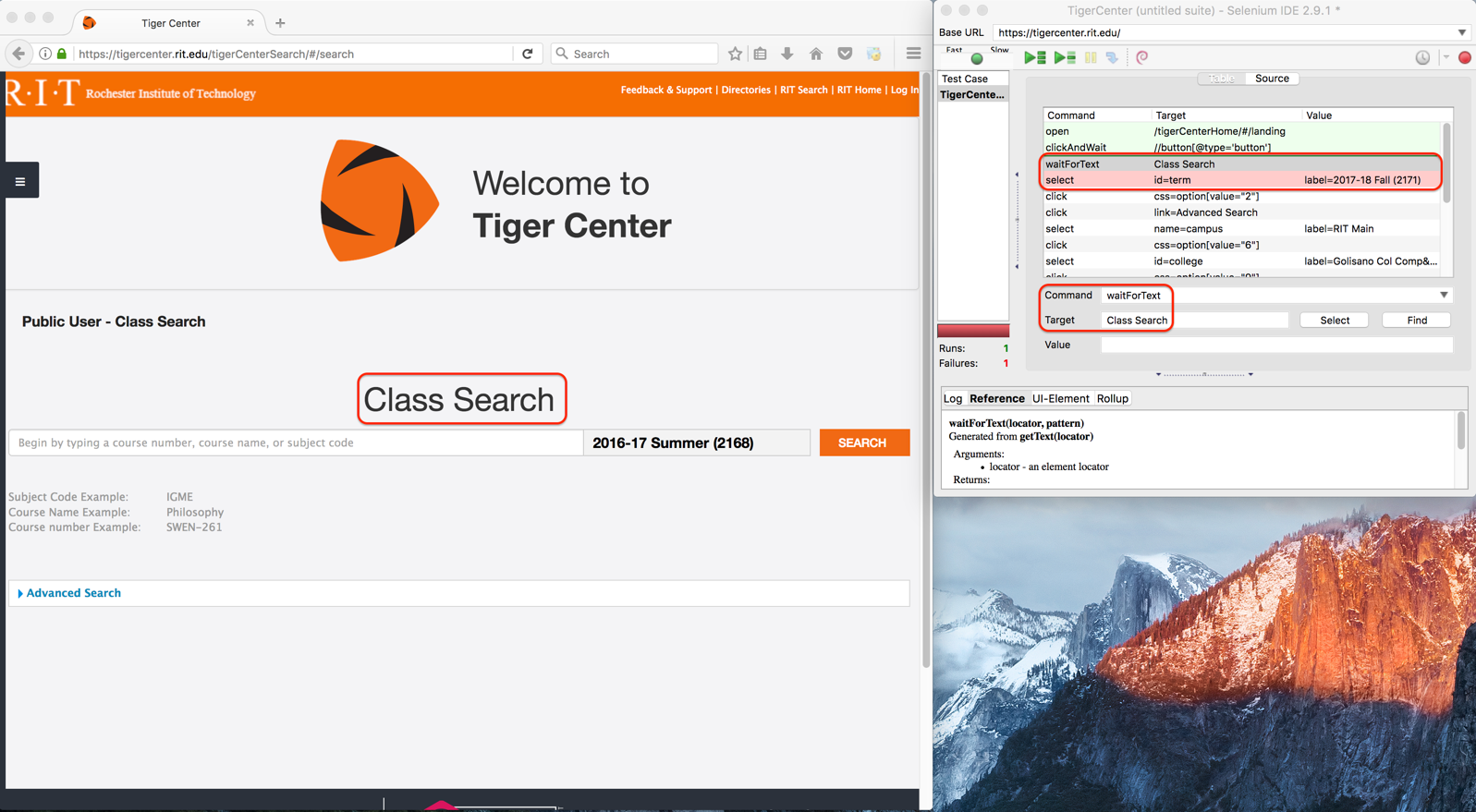


If you get an error stating that an element cannot be found, consider adding a command such as **waitForText**.

* To add a custom command in the script, you right click on the command below where you want to insert the new command and select adding a command.

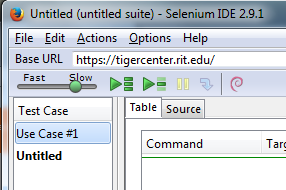


* Use the **Select** button to find text on the screen. Notice how the **waitForText** command is *before* the **select** command. *Note: sometimes the* ***waitForText*** *command fails, but does not cause the whole script to fail, usually the script will just keep going.*

\

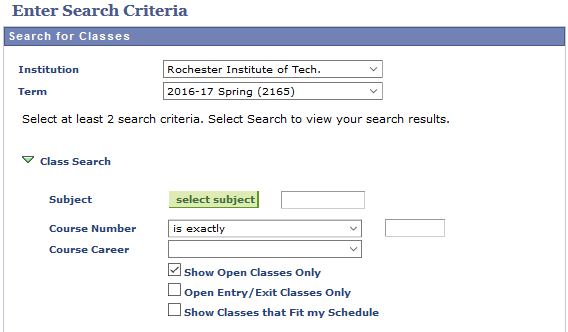
* One issue you might encounter is getting an error “cross process js call failed”. Or if you encounter any issues with running the Selenium IDE, please try the following fixes:
  + [Downgrade your Firefox](https://ftp.mozilla.org/pub/firefox/releases/) below version 55.
  + Install Selenium IDE: Flow Control extension to Firefox.

Right click on the test case name; rename it to**Use Case #1**.



Open a new test case with File -> New Test Case (or CTRL/Command + N). Select the new test case so the second test case is written in bold.

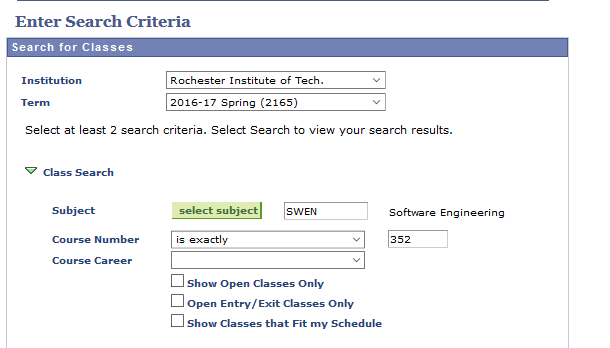
1. Go to this [web address](https://mycampus.rit.edu/psc/sasrch/EMPLOYEE/HRMS/c/COMMUNITY_ACCESS.CLASS_SEARCH.GBL) to access the SIS class search page.
2. Start recording again on Selenium IDE.
3. Select the courses for *next* semester (if possible), this example shows using Spring 2017.



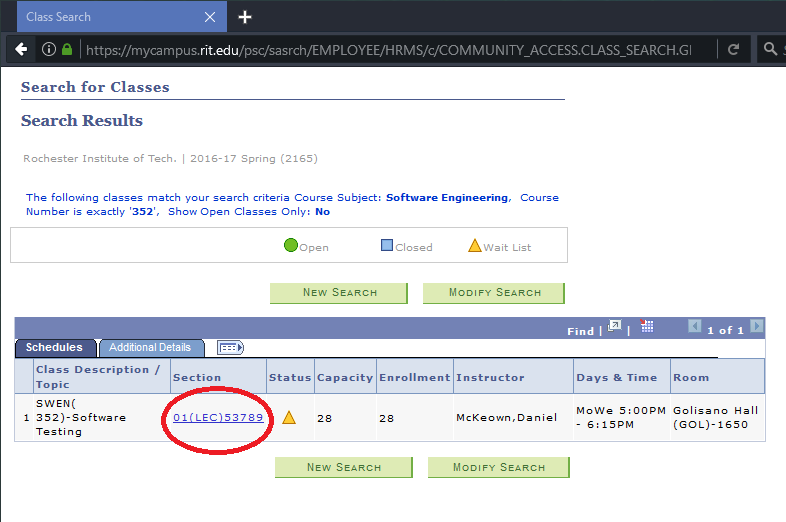
1. Click the *select subject* button on the page.
2. Click on ‘S’ at the top of the page to show majors starting with ‘S’, and then click Select next to Software Engineering.



1. Enter the course number for this course (352), and uncheck *Show Open Classes Only* if it appears on the page directly below the *Course Career* dropdown. The result should look like this:



1. Click *Search* at the bottom of the page.
2. Click on the blue highlighted Section text for your Software Testing class.



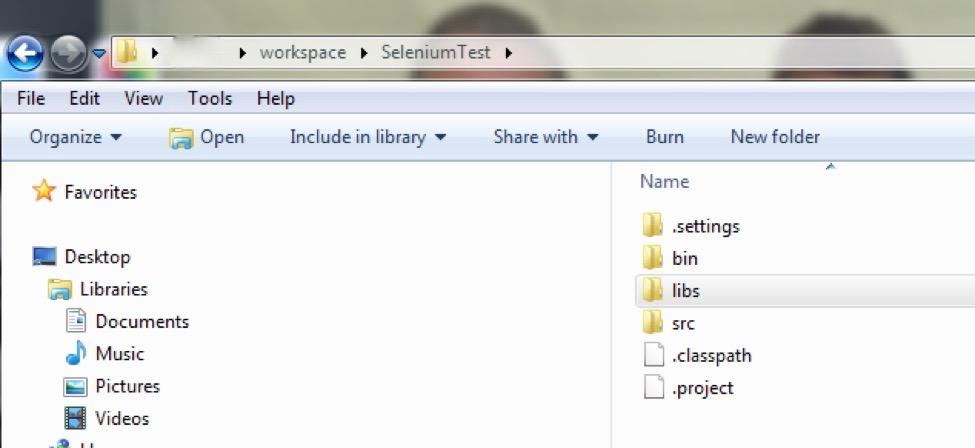
1. End Selenium recording. Be sure the test works by running the test in the Selenium IDE.

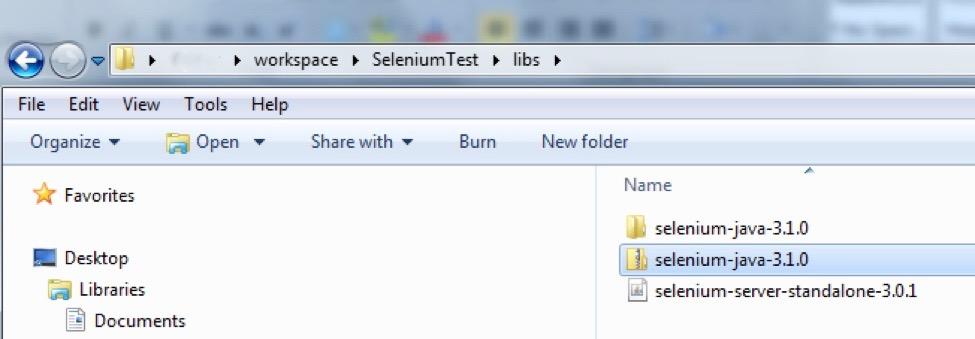
The same script execution issues may occur, apply what you learned for Use Case #1.

1. Rename this **to Use Case #2**.
2. Record both of your scripts running. Each script in a different recording. Clear the log before every recording. Selenium IDE must be shown while it's running on the browser in the recordings. Include both videos in your final submission. The video format must be either **.mp4** or **.mov**. Name the video files **Use** **Case #1** and **Use** **Case #2**.

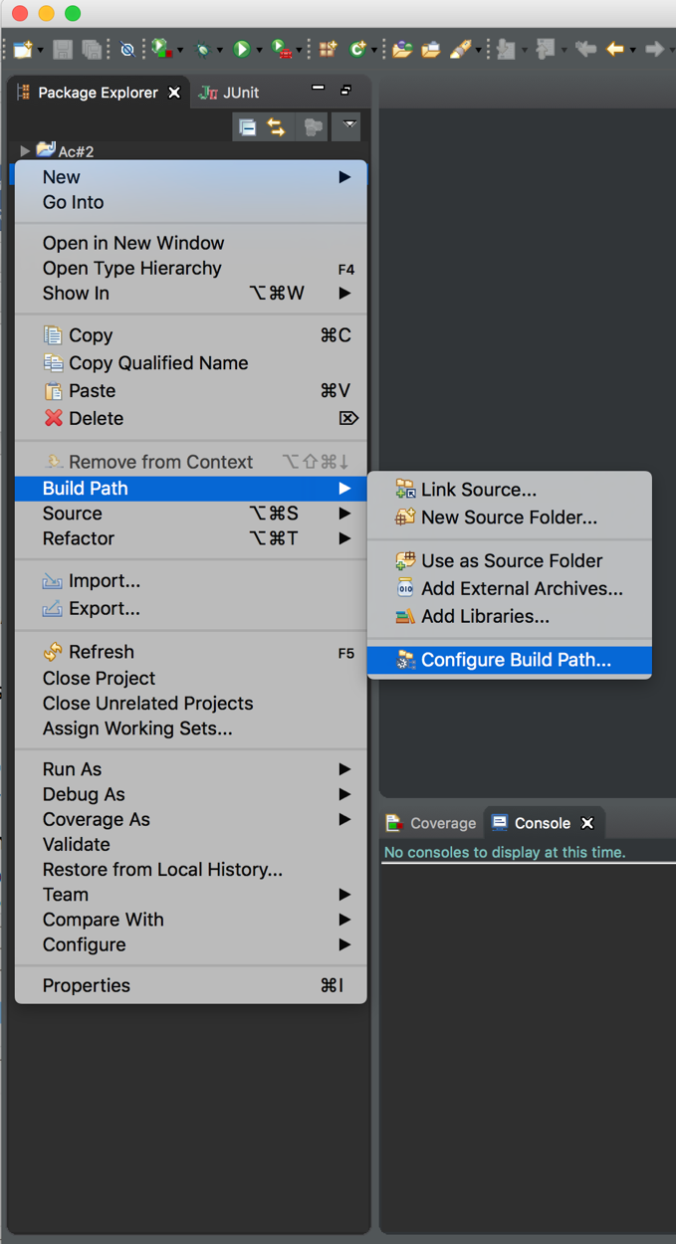
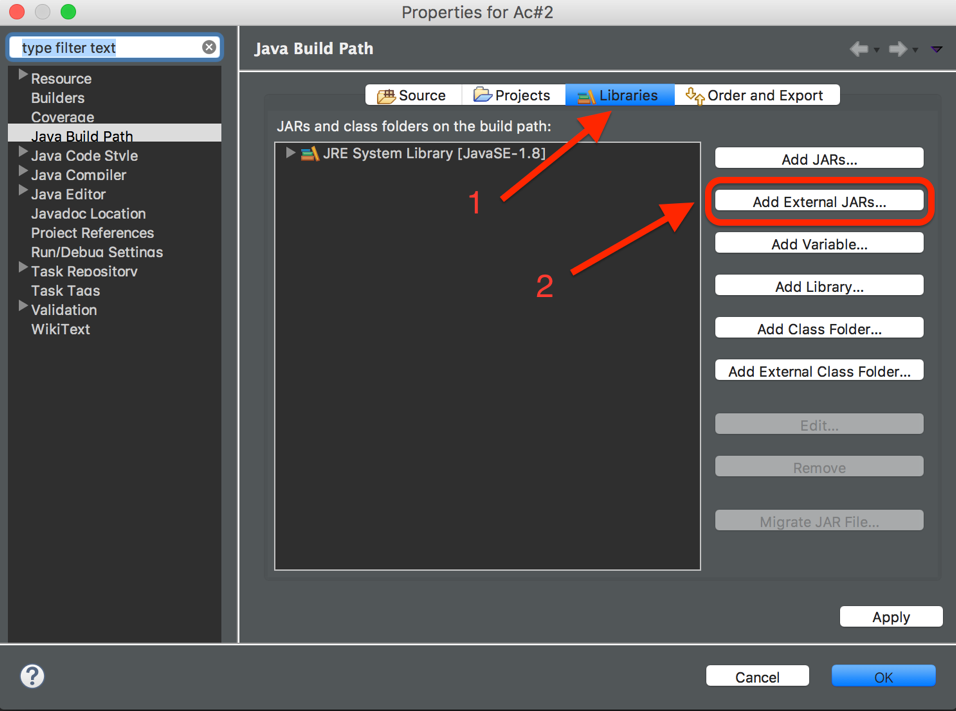
## Setup Selenium & Java for Part 2:

*Note: the instructions below are for Google Chrome, you are allowed to run this on Firefox if you choose but you will be responsible for the setup.*

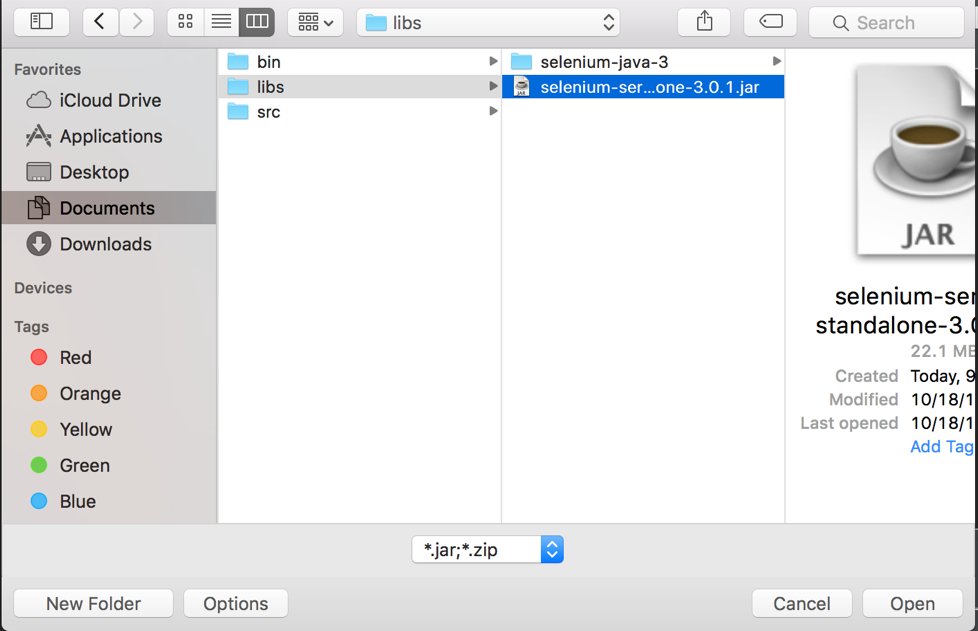
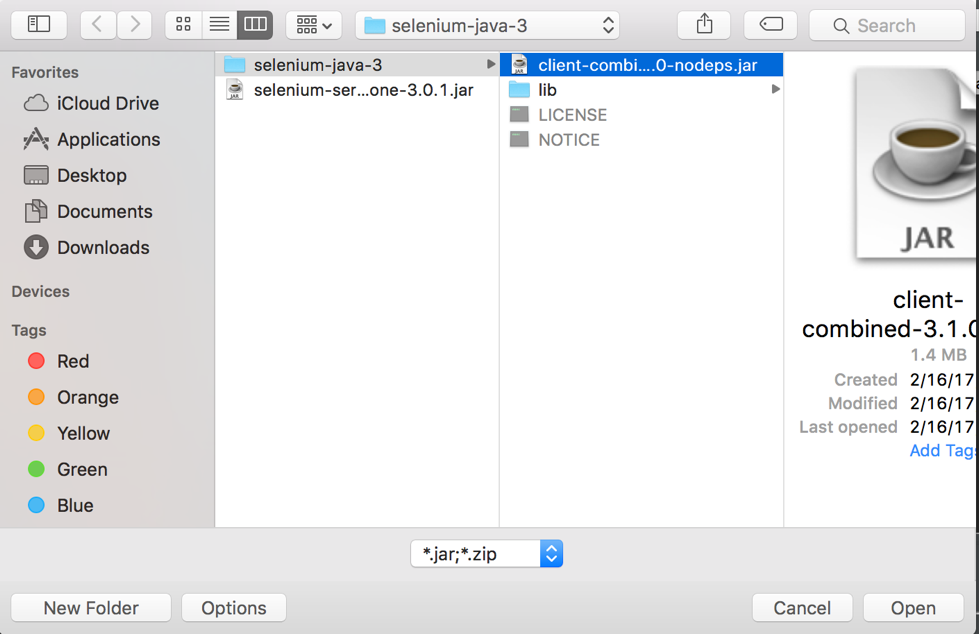
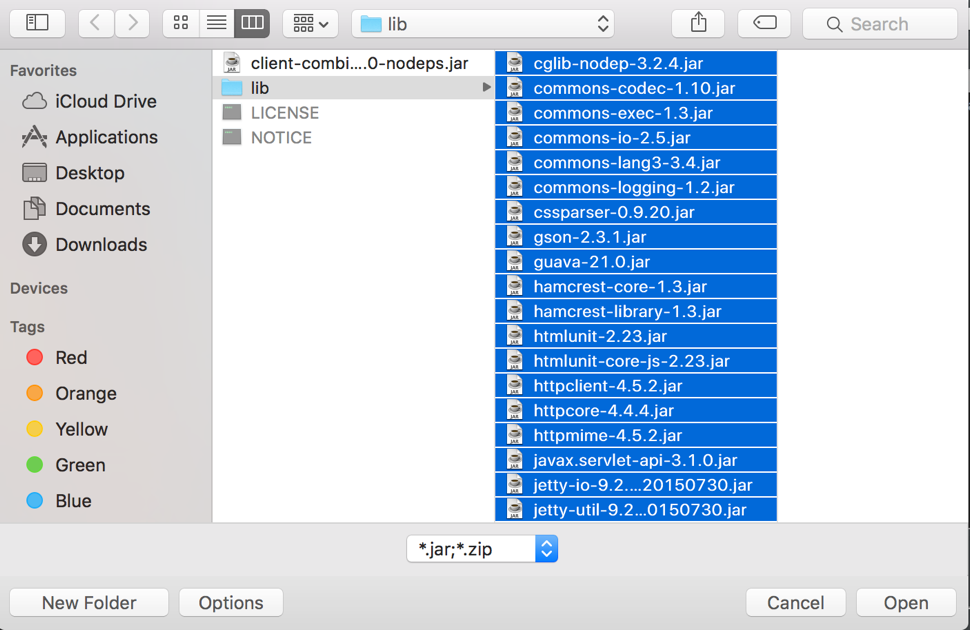
1. Go to this webpage - <http://www.seleniumhq.org/download/> and download the Selenium Client & WebDriver Java language bindings.
2. Go to the selenium [standalone server](https://selenium-release.storage.googleapis.com/index.html?path=3.0/) (3.0.1) and download selenium-server-standalone-3.-1.jar.
3. Place the two above downloads in a place they can be found later.
4. Open Eclipse and make a new Java project: File -> New -> Java Project.
5. Name the Project **SeleniumTest** and make sure execution environment JRE is Java 8.
6. Create a local folder named *libs* in your project area on the same level as *src* and *bin.*
7. Unzip Selenium-java (if it is not already extracted)



1. Add the JAR files as libraries to the Java project in Eclipse by right clicking on the SeleniumTest project (picture below has a project with a different name), then performing steps #1 and #2 in red in the picture. (click **Ok** at end).

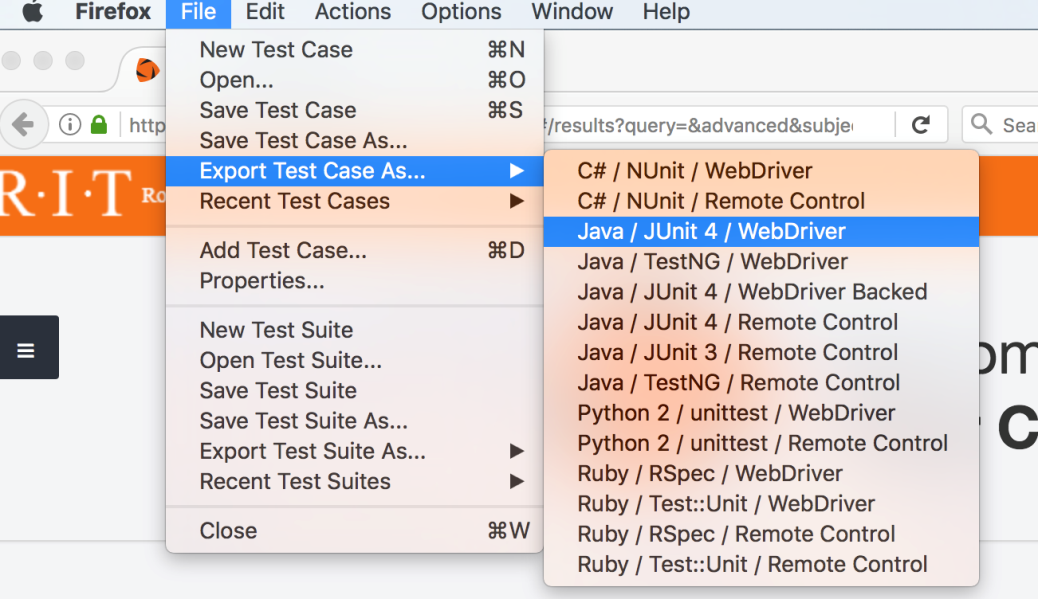


*Note: there are jar files in 3 different locations*

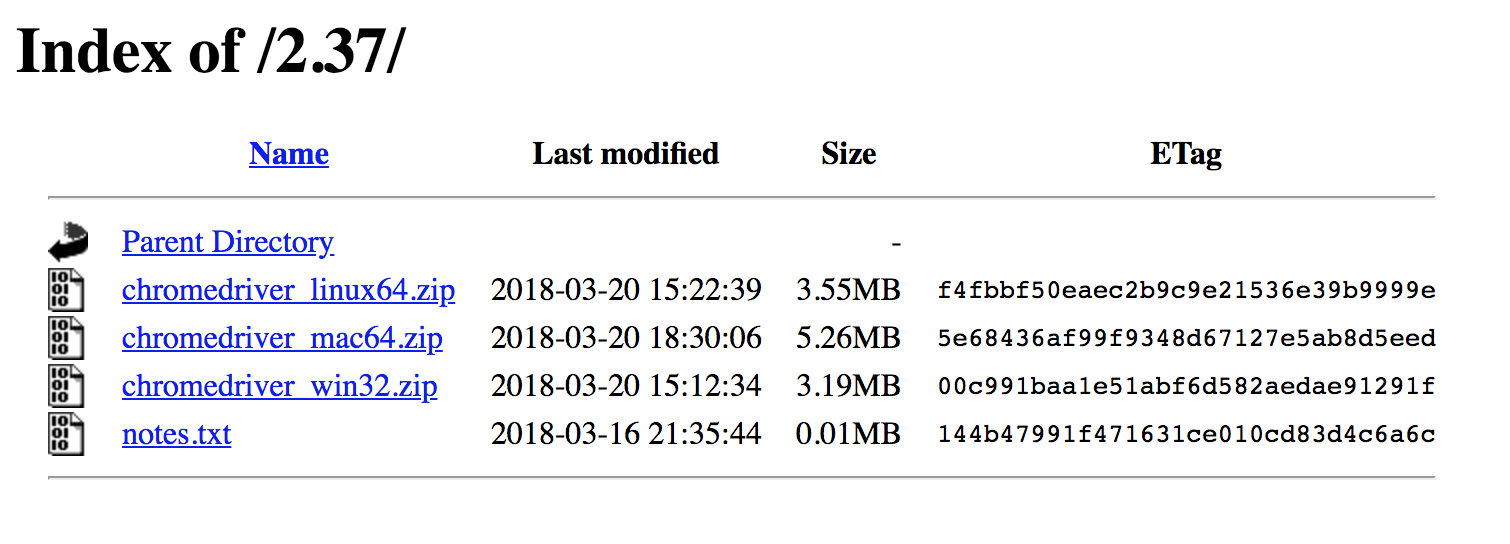
* 1. Add selenium jar file 
  2. Add client jar file 
  3. Add lib jar files 

Your project should look similar to this:

1. In the Selenium IDE, export TigerCenter test case from the first part of this activity to Java **Junit4** with WebDriver. Save the file directly in the /src/ of the java project and name it “**TigerSearch.java**”.



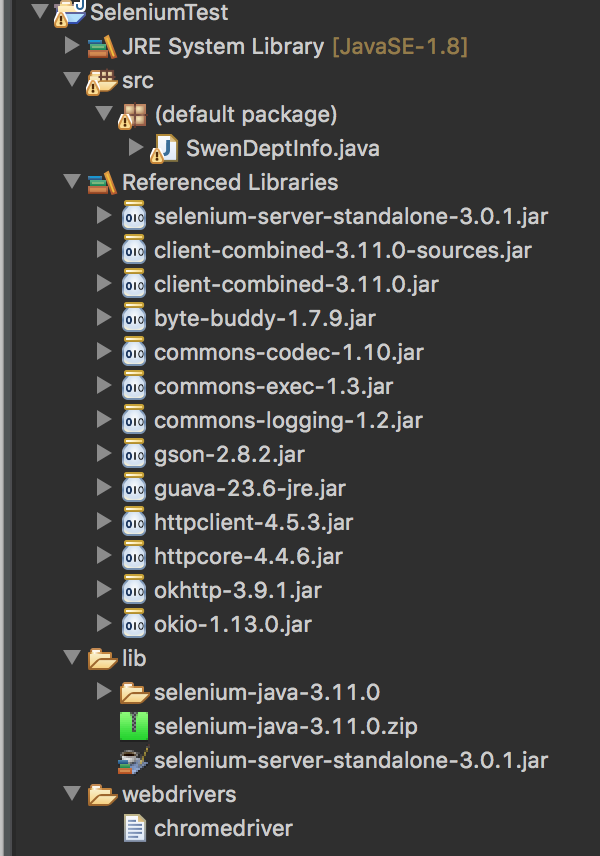
1. Navigate to [Google Chrome Driver download page](https://chromedriver.storage.googleapis.com/index.html?path=2.37/) and download the driver version that is compatible with your current OS.



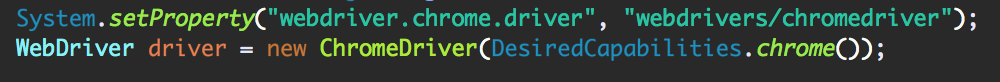
*Please Note – The WebDrivers requires the base web browser installed to function. (For example, the ChromeDriver won’t work without the Google Chrome browser installed).*

1. Place the downloaded WebDriver in a webdrivers/ folder after you extract it from the zipfile in your Selenium Java project. This will allow easy access to the WebDrivers without dealing with System settings and the PATH variable. Your Java project should look similar to the image below (the **SwenDeptInfo.java** file is from part 3).

*Note: for OS systems other than Windows the webdriver file might not have the* ***.exe****, you should follow the instructions as they are applicable to the OS you are currently using.*



1. In the setup method of **ALL** the Selenium java files, find the line that initializes the WebDriver driver, and replace it with the following:



*Note: one issue you might have is getting NoClassFoundException. That happens because Eclipse is unable to find the .class files in the specified path, this could be due to issues with the libraries as well. A suggested fix for this problem is to completely delete the project and starting over with a new project. You would also have to redo the setup steps again.*

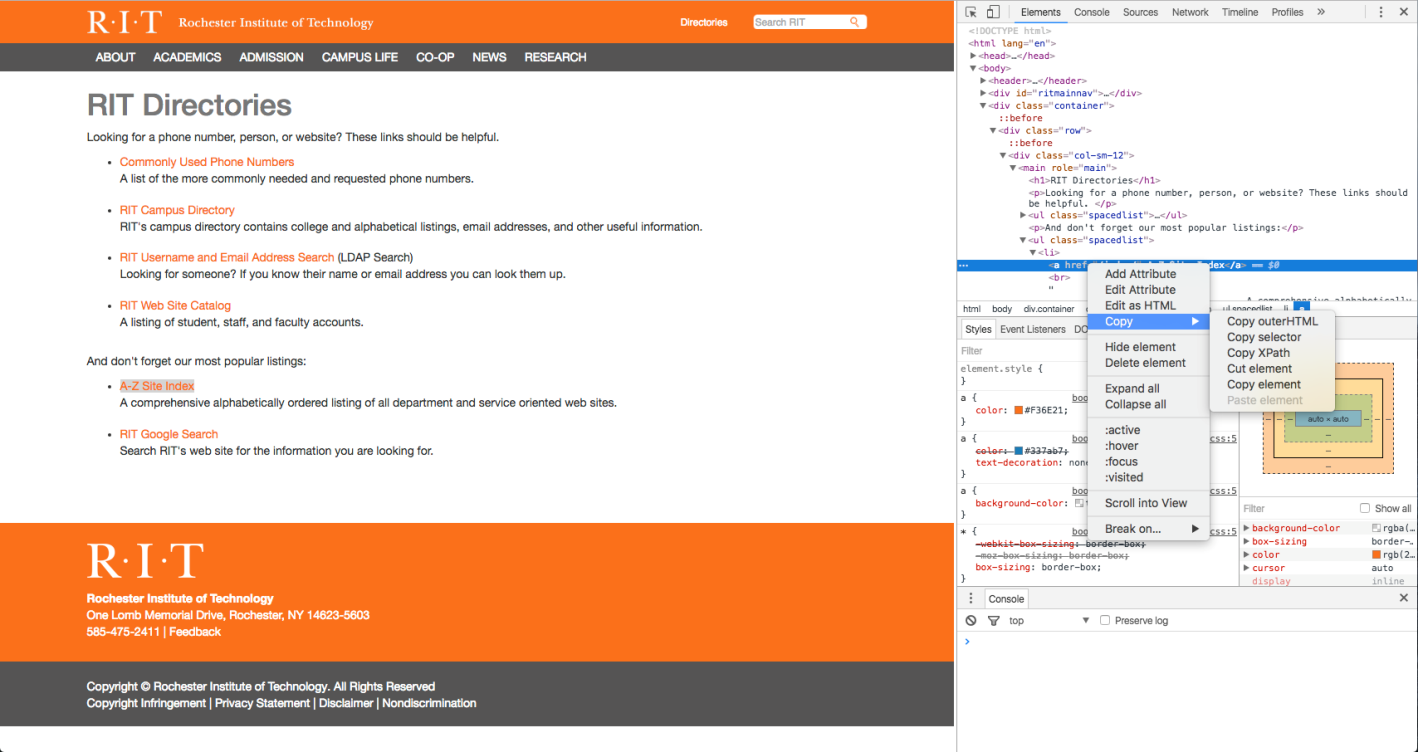
**Activity Instructions for Part 2:**

While it is neat to record clicks inside of web pages, we can do a lot more with programming languages and selenium bindings than is possible with the Selenium IDE. Go through the Setup process for Part 2 of this activity if you haven’t already.

The tests written in the first portion of this activity are very simple tests and get us through the web applications to find the details of this course. Let’s add some data validation/verification to the tests.

*Note: Some of those steps are for debugging purposes and will not be included in the actual script, however, they will make it easier for you to finish this part. Doing all of the steps for this part is highly encouraged.*

Debugging hints:

* Use either Google Chrome to get the attributes you want (you can use any of those to find elements in the webdriver).  
  
* Comment out the line responsible for closing the driver when the script has finished or failed. When the browser window stays open, it will be easier for you to see what caused it to fail or which step is failing in your script.

1. Output to the terminal the name of the course, taken from the header of the Class Details section (Don’t include **Class Details** in your output).
2. At the end of the Tiger Center test case, grab a List of WebElements of the table rows in the Class Details table. To do this follow the steps below:
   1. On the Search Results page, with the class details open (the page at the end of the first half of Part 1) inspect the *Class Details* element and find the ID tag associated with the table.
   2. In the Tiger Center test, write the code below to store the entire table in a WebElement, using the ID tag found.
   3. Store a List of WebElements by finding all the Elements in your variable from Step 3B that have the “*tr*” tag.
3. Go through this list of elements and output to the terminal the Days, Time, Location & instructor of the class. Make sure it is properly formatted so it can be easily readable.
4. Repeat this test for an additional 3 classes, so that your Tiger Center test case has four separate tests for 4 different classes (each should be annotated *@Test*).
5. Run all four test cases to ensure they work properly and produce the desired output. Once you are satisfied with the output.
6. You may notice most of the contents of these tests are similar, simplify the Tiger Center test case to include a single *test* method, that that takes different variables representing the different courses you are searching for.
7. When your script runs successfully with the simplified method, record the execution and include the video recording in your final submission. Ensure the recording shows successful execution for all four cases and shows the console output at the end of the execution.

**Part 3:**

For this part, we are going to start a script from scratch only using Java, without using the click and record IDE in the browser. The script should be a new class in the project you have been working, the class should be named “**SwenDeptInfo.java**”.

**Your script should get to the Software Engineering Department website, contact us page, print the following information to the console:**

* Office location
* Phone number
* Fax number
* Email address
* Facebook page
* Mailing address

**In addition, your script should do the following:**

* Get to the Software Engineering Website by going to <https://www.rit.edu/>. *Points will be taken off* if your script goes to the Software Engineering Department website right away using the URL
* Include the proper waits/delays and verifications, at the same time, the delays should not be extensive *(no wait/delay beyond 40 seconds).*
* Verify that the information you want to print is there, if it is not for some reason, print the following: “Contact information not found”.
* When printing the contact information, make sure they are properly formatted and resemble the information on the website as closely as possible.
* When your script runs successfully, record the successful execution and include the video recording in your final submission. Ensure the recording shows successful execution for all four cases and shows the console output at the end of the execution.

**Helpful hints:**

* Look around in **Directories** to find the SE website that *does* not open a new tab.
* Whenever you get stuck, consider xpath and cssSelector.
* Sometimes it appears that a webdriver function is not working, but most of the time issues like that happen because it is going too fast for the browser to keep up with it. A good idea is to insert a wait/delay right before the line of code that is causing it to fail. You can use Thread.sleep(millis).
* The easiest way to find elements is by linkText(), but remember, that only works on html link elements.

**PERFECTO**

**Part 4:**

**Overview:**

For this part, we are going to use Perfecto MobileCloud for mobile testing. While, it’s good to use Selenium to do automated testing cross-browsers, we want to run automated testing for mobile devices. Perfecto is a testing tool built using a web-based user interface, which allows the creation of tests scripts that are easy to generate and easy to use with no coding involved at all. The commands are readily available on the website in the form of widgets. Test developers can either record the actions through point and click or use the provided commands to add to the scripts and define its properties in the user interface. Refer to the Perfecto Instructions & Hints at the bottom of the document if you need help.

**Setup instructions for Perfecto**

* Login into your Perfecto mobile account: <https://mobilecloud.perfectomobile.com/>
* You should see three tabs at the top Interactive, Automation, and Reporting.
* Click on “Launch a mobile device” and select any device from any region.
* Familiarize yourself with the layout of Perfecto’s user interface.

**Part 4.1 Using Perfecto’s Interactive record tool to create test scripts**

* For this part, we will use Perfecto’s interactive record tool, to make a couple basic test scripts.
* Create a script using the record button from “Automation” tab.
  + Record a stopwatch running for several laps using an Apple device.
  + Save the recording as script1.
  + **Include these files in your submission: the script as part1-script1.mcr, the legacy report PDF file, and the video.**
* Create another script using the record button from the “Automation” tab.
  + Record a stopwatch running for several laps using a different Apple device.
  + Save the recording as script2.
  + **Include these files in your submission: the script as part1-script2.mcr, the legacy report PDF file, and the video. Include these files in your submission.**

**Part 4.2 Using Perfecto’s Automation functions to create test scripts for SIS**

* While, using Perfecto’s interactive record tool is convenient and easy to do, it doesn’t scale well and isn’t portable across every mobile device. For this part, we want to create a test script that can work on any mobile platform we would like to test on.
* Create a script that navigates to <https://www.rit.edu/infocenter/>, Clicks on “SIS Class Search” and searches for our section of SWEN 352.
* Use the following functions found under the “Automation” tab as necessary:
  + Browser go to
  + Find text
  + Button.Click(text)
  + Webpage Elements
  + Object Spy
* *Note: You may use more functions that are available to you than the ones listed above.*
* You must use image checkpoints throughout to verify that the elements on the page are correct.
* **Include these files in your submission: the script as part2-script.mcr, the legacy report PDF file, and the video. Include these files in your submission.**

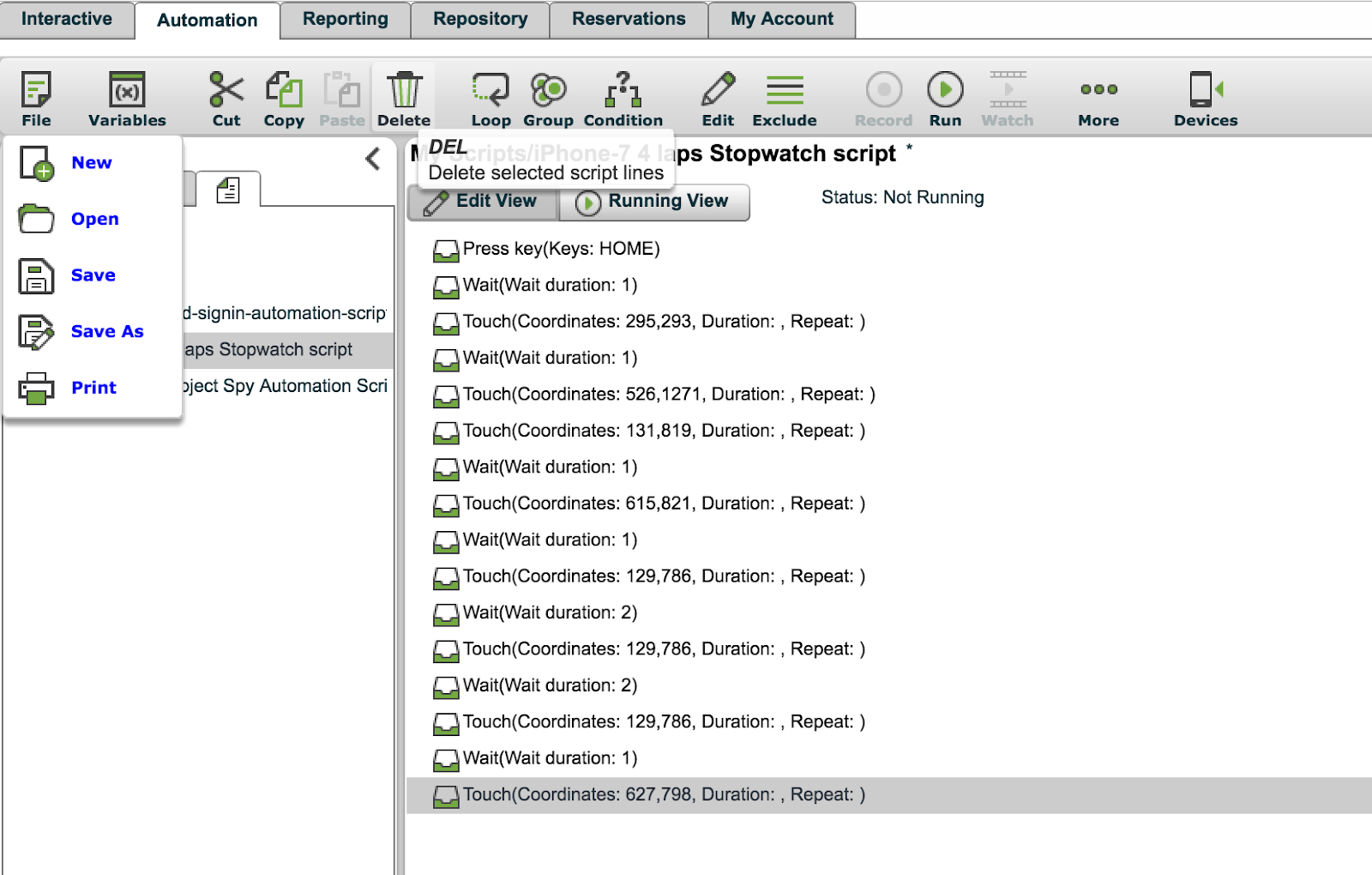
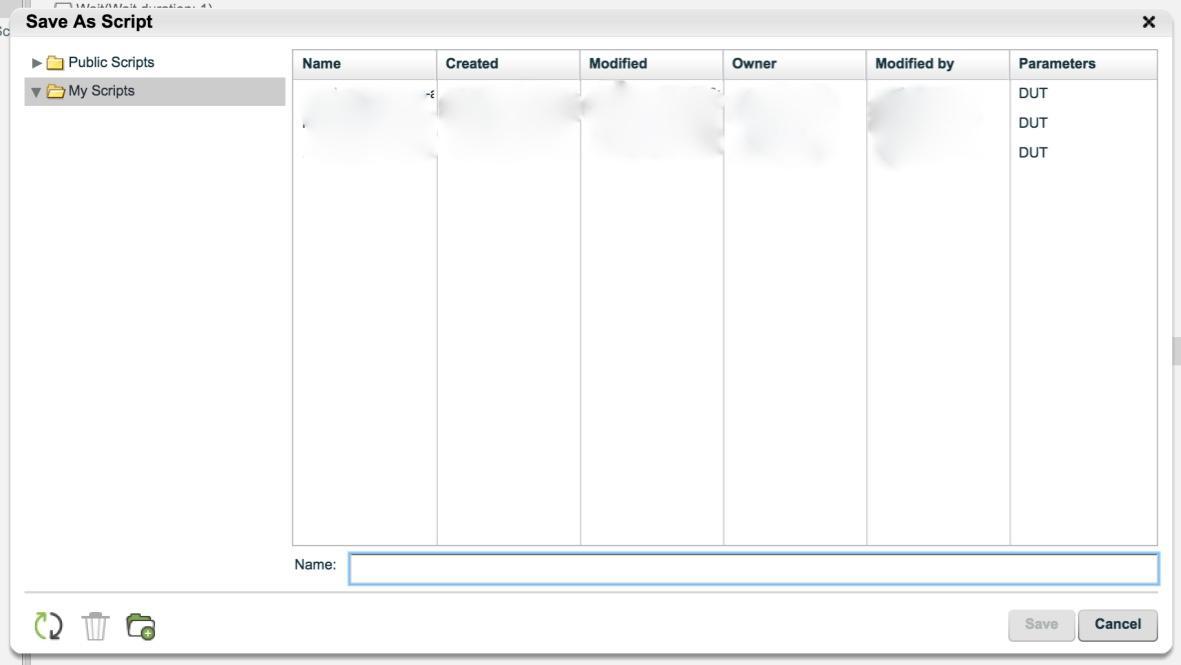
*Note: all PDF reports generated by Perfecto should have the following naming convention: part#-pdf.pdf. For instance, the PDF report for part 1 should be labeled: part1-pdf.pdf.*

**Part 4.3 Using Perfecto’s Automation functions to create test scripts for TigerCenter**

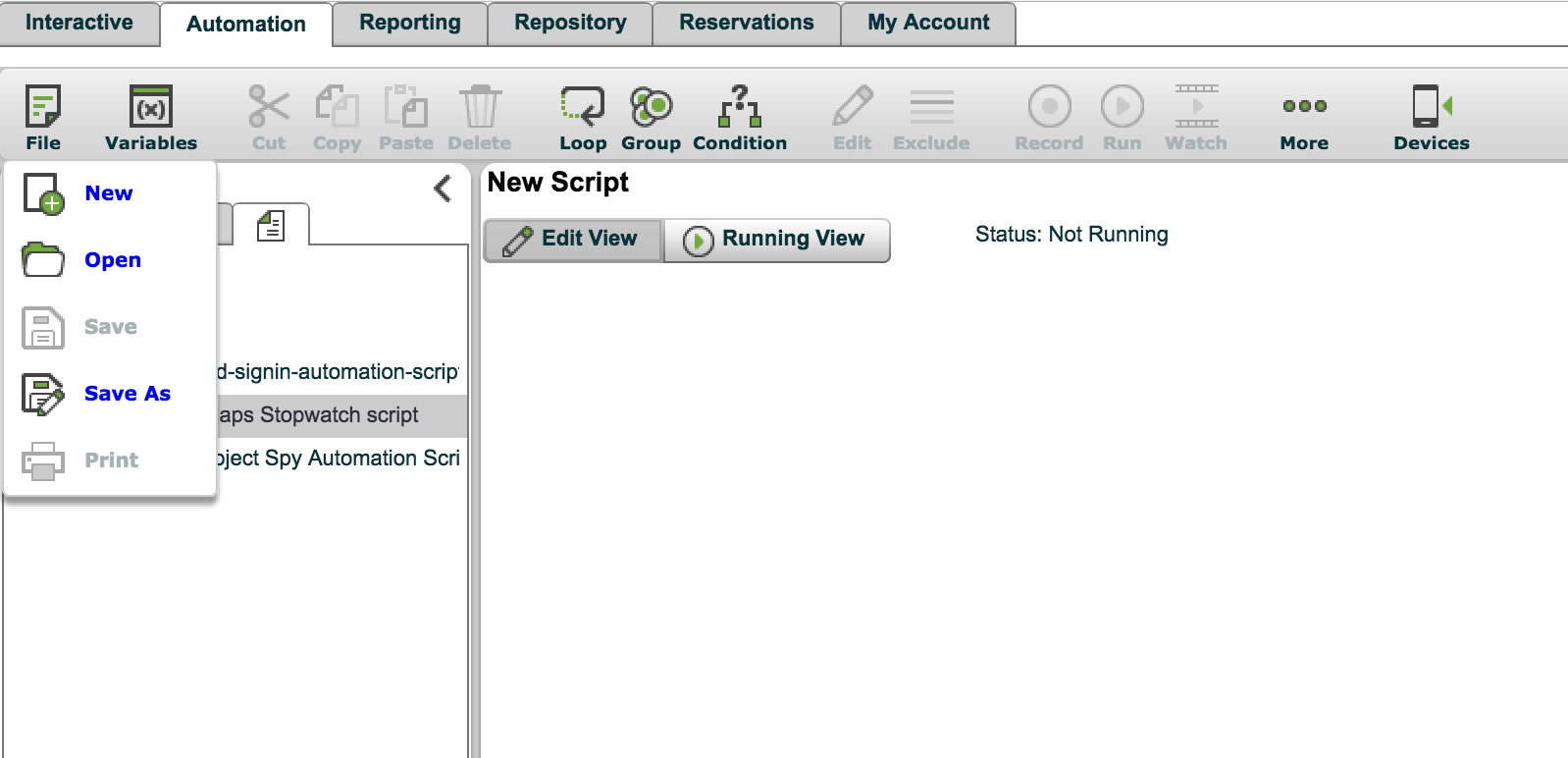
* Similar to part 4.2, we want to use the automation functions again, but this time create a script for TigerCenter.
* Create a script that navigates to <https://tigercenter.rit.edu/tigerCenterHome/#/landing>, Clicks on “Start searching”, and searches for our section of SWEN 352.
* Use all the following functions from part 4.2 as necessary and must use image checkpoints.
* **Include these files in your submission: the script as part3-script.mcr, the legacy report PDF file, and the video. Include these files in your submission.**

**Perfecto Instructions & Hints**

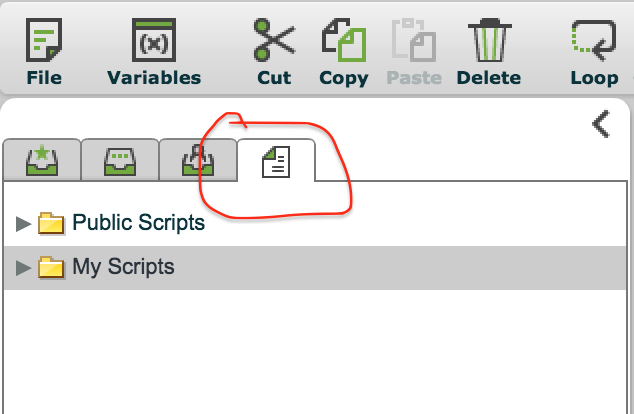
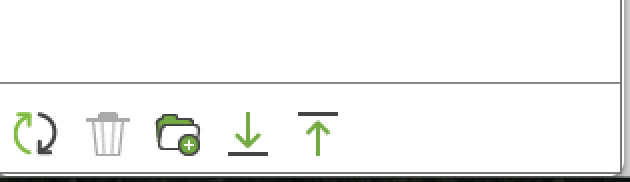
Saving your scripts on Perfecto’s cloud:

1. Click on the File button on the left top of the automation view
2. Click on “Save As” option in the menu
3. A pop menu will appear allowing you to name the file and selecting where in the repository file structure you want it to be saved
4. Save the script under the “My Scripts” folder
5. Click save

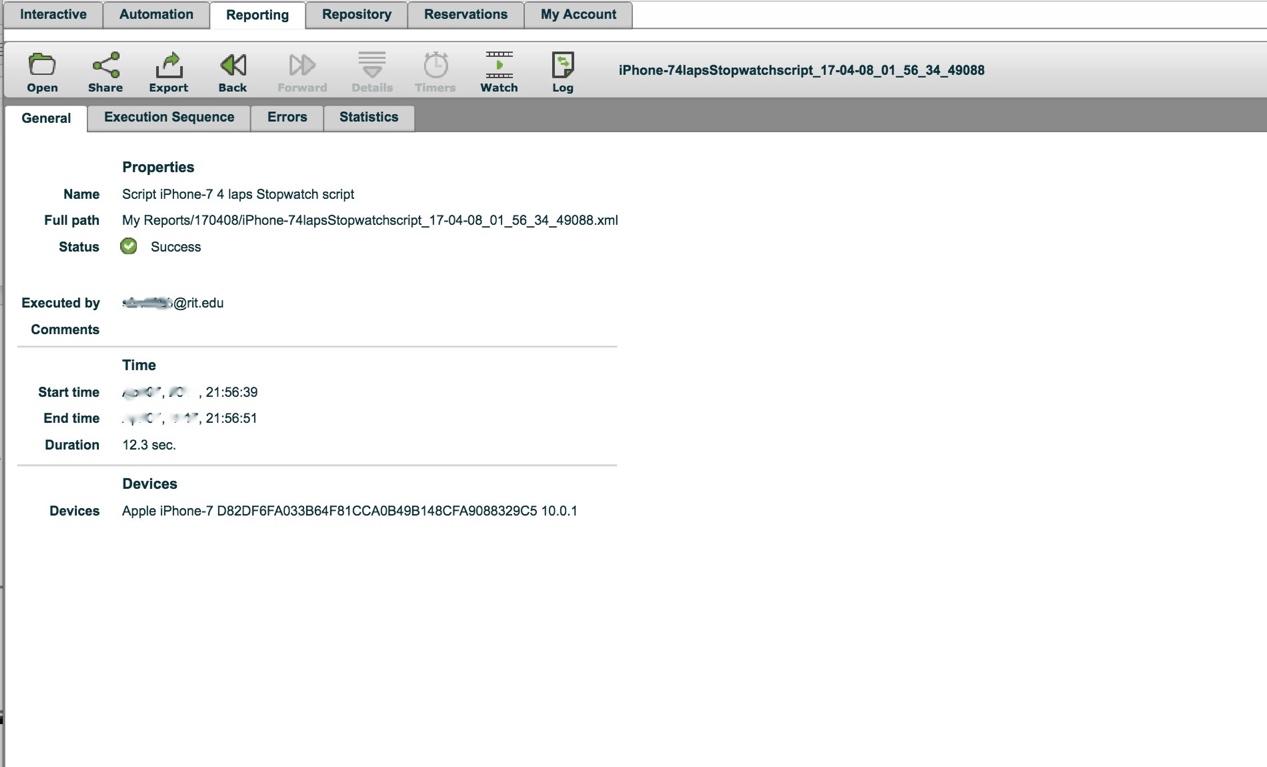
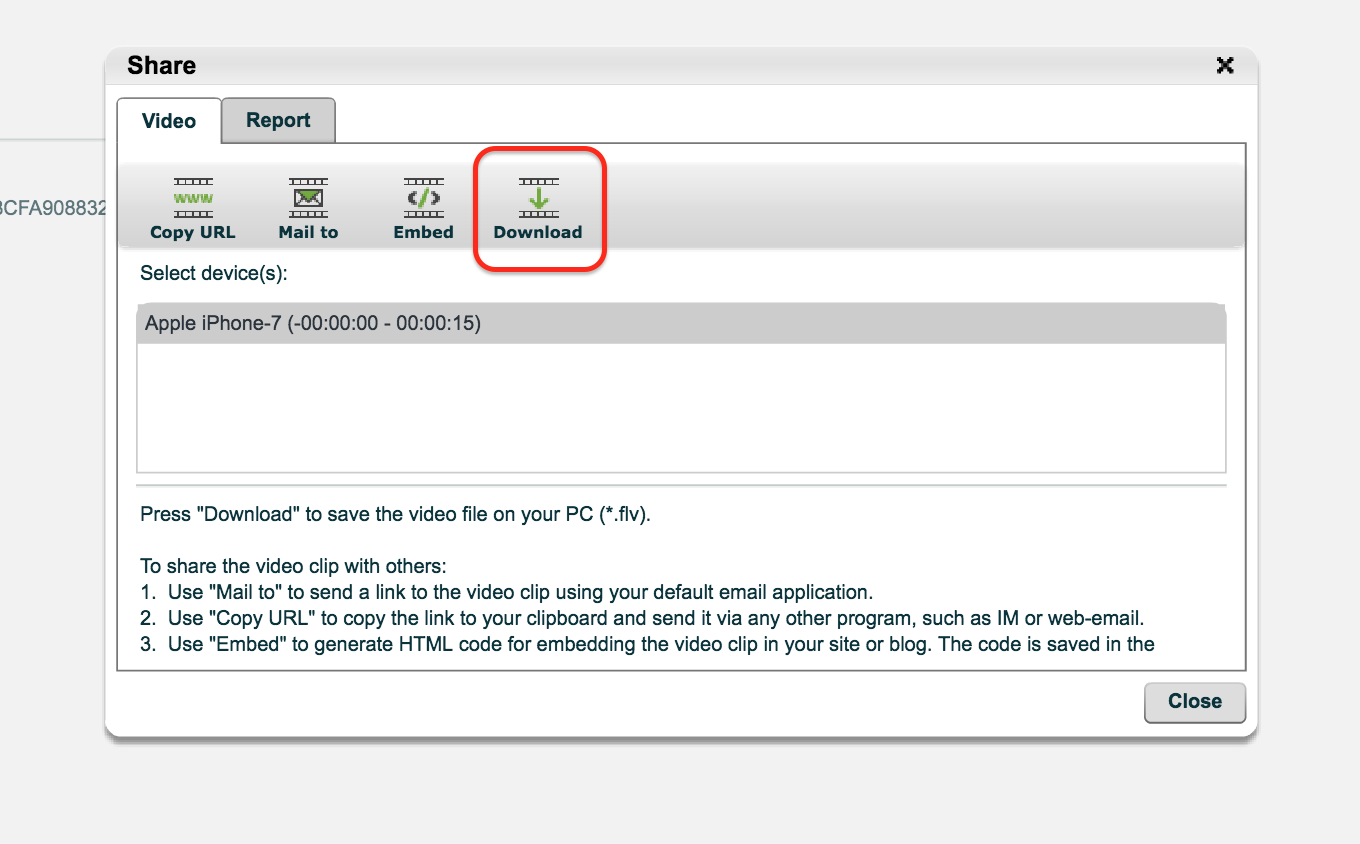
Retrieving your scripts from Perfecto’s cloud:

1. Open the automation tab
2. Click on the “File” icon on the top left of the screen
3. Click “Open”
4. Navigate through the repository, select your file and click “Open” and your script will load

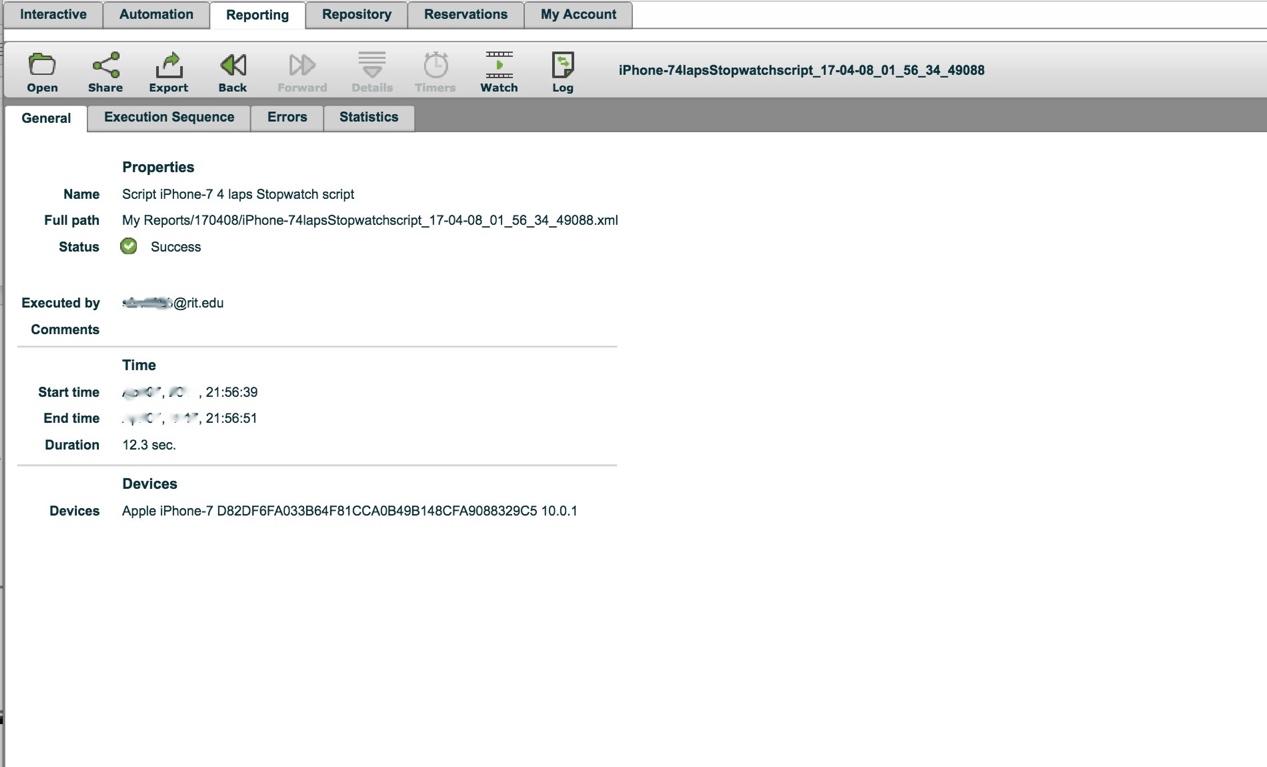
Downloading and uploading your scripts:

1. Navigate to the “Scripts” tab next to “Functions” and “Common Functions”
2. Once you are there and have a script selected, the buttons on the bottom of that tab will allow you to download the selected script to your machine or upload a new one to the cloud. The extension of scripts that download is (\*.mcr)

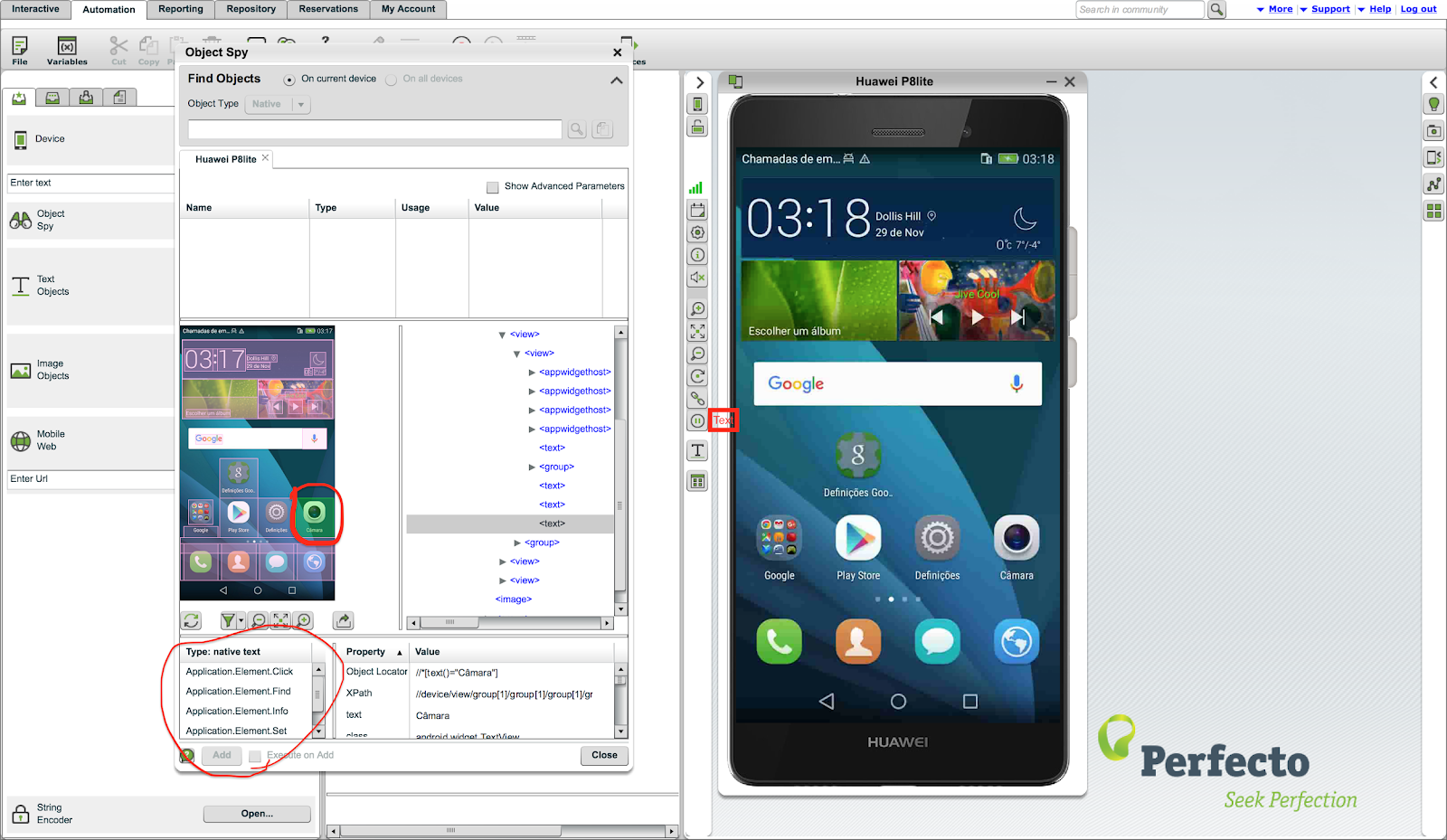
Downloading a video of your script’s execution:

* + - 1. Once your script finishes execution, you will be prompted to view the report which Perfecto generates for you at the end of every execution, click Yes
      2. You should see a screen like the picture below, which contains your report
      3. Click on Share
      4. Then click on Download

Exporting a PDF report of your script:

* + - 1. Once your script finishes execution, you will be prompted to view the report which Perfecto generates for you at the end of every execution, click Yes
      2. You should see a screen like the picture below, which contains your report
      3. Click on Export
      4. Select to PDF

Object Spy

Object spy is one of the most helpful tools that Perfecto offers to make automation easier and more accurate. Once you click the Object Spy button, a popup will appear with a picture of what is currently being displayed on the interactive device. Once you click on of the objects on the screen, you will be able to select the action you would like performed on that object

Notice that in the picture above, once the Camera object was select, a list of applicable operations appeared underneath. Once you click on one of those operations it will automatically be added to your script so you do not have to keep closing and opening Object Spy.

Text & Image Objects

Those are more specific than the Object Spy, as the naming suggests, one interacts with text objects while the other handles the interactions of image objects. There is several operations that can be applied to both of those object types. They both help you make your script safer and more generic especially when testing web pages on mobile devices. You can, for example, look for and click a button using the Text that it has using the Text Objects operations, or if the button happens to be an image, you can do the exact same operation using Image Objects.

Setting and/or getting the image or text objects can be found under the “Edit” button of that object type.

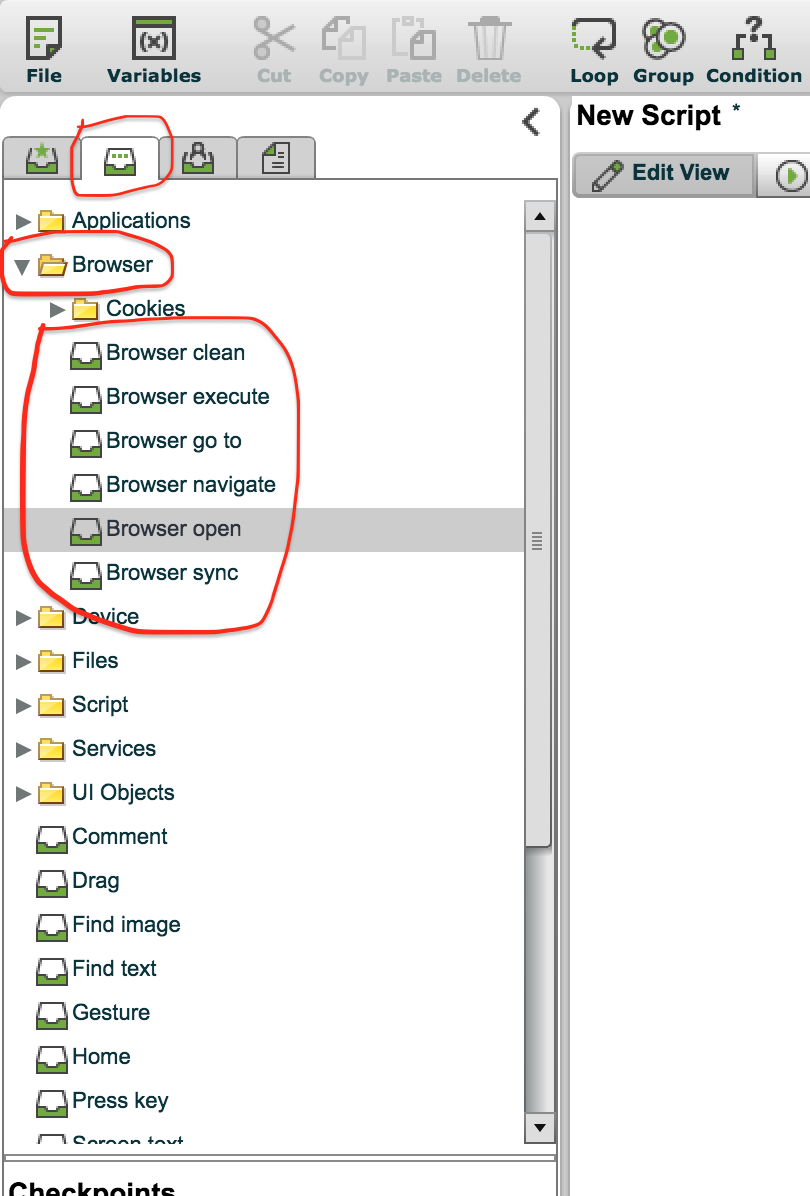
Perhaps the most important function in automation that you can do using those two object types is the Sync operation which serves as a checkpoint. The Sync operation helps you verify that there is a specific text or image object before you carry on with the rest of the script to ensure you are the right place. The operation also uses a few parameters, one of those is the timeout parameters, which basically sets value for the cut off time

In the picture above, the string “Phone” was used as the value of the expected text to be looked for. The timeout value is defaulted to 60 seconds, you can change that to make it shorter so your scripts do not wait for long.



A good way to verify that you have the correct text object is to press the “Try” button to see if the text object can be found where you expect it to be. As can be seen from the picture above, the text “Phone” was set as the expected text and it was successfully found using the “Try” button.

Perfecto’s Generic Commands:

In addition to all that's been discussed above, Perfecto provides a number of generic commands that run on all platforms and devices such as open browser. Those commands can be found when switching to the “Functions” tab

You can add those functions to your script by doubling clicking on them.

Conditions

You can add a condition for your script to perform actions in case of success or failure by clicking on the “Conditions” button right above your script's name. You can use the Text & Image Objects Sync operation as the condition by placing them on the line right before the condition statement, and then just drag operations on the success or failure lines.

Scripts hints:

* You do not have to worry about devices that take too long to load the page
* There are many different conditions you can have to ensure your script runs correctly on any device, keeping in mind that web pages look different in phones and tablets
* Make sure your scripts have “waits” or wait for a specific condition to be true to account for pages loading
* You don’t have to worry about ‘one time situations’ such as the device asking for the default browser and/or policy agreements

**SUBMISSION INSTRUCTIONS**

* **Place all the documents/folders you are going to submit in a single folder and zip it. The name of the zip file should be after your team’s number (Ex. Team\_#.zip)**
  + **Neatness and organization – 5%**
* **Part 1 (Selenium IDE) Videos: Should be placed in the submission folder along with the document and the project. Videos should not be too big in size and follows the correct format.**
  + **20% (10% each)**
* **Parts 2 & 3 (Selenium WebDriver) Videos: Should be placed in the submission folder along with the document and the project. Videos should not be too big in size and follows the correct format.**
  + **20% (10% each)**
* **Parts 2 & 3 (Selenium WebDriver): Submit a zip file of your Java source code files so it can be executed in Java (make sure it runs on all the team’s machines before submitting to ensure robustness on TA’s machine).**
  + **Successful execution – 5%**
  + **Comprehensive comments (for both classes) – 10% (5% each)**
* **Part 4 (Perfecto): Provide a total of 12 files for this part in your final submission zip file:**
* **part1-script1.mcr, the legacy report PDF file, and the video – 10%**
* **part1-script2.mcr, the legacy report PDF file, and the video – 10%**
* **part2-script.mcr, the legacy report PDF file, and the video – 10%**
* **part3-script.mcr, the legacy report PDF file, and the video – 10%**

*Perfecto script breakdown (per script):*

* + *MCR script execution: 2%,*
  + *PDF report: 4%*
  + *Video recording: 6%*