

**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**-----🖎🕮✍-----**



**BÁO CÁO BÀI TẬP LỚN**

**Phân tích công cụ kiểm thử Watir**

|  |  |
| --- | --- |
| ***Giảng viên*** | **:      Thạc sĩ Nguyễn Thu Trang** |
| ***Lớp học phần*** | * **:** [**2021II\_INT3117\_1**](https://courses.uet.vnu.edu.vn/course/view.php?id=6171) |
| ***Môn học*** | **:      Kiểm thử và đảm bảo chất lượng phần mềm** |
| ***Sinh viên thực hiện*** | **Trịnh Thị Kim        –  18020740**  **Vũ Quỳnh Trang**  **Nguyễn Hoài Thu** |

**HÀ NỘI - 2021**

# **Chương 1: Giới thiệu về Watir**

1. Đặt vấn đề

Phát triển phần mềm bao gồm các quá trình: thu thập yêu cầu, thiết kế, lập trình, kiểm thử , triển khai. Kiểm thử cũng nhằm phát hiện lỗi hoặc bất cứ vấn đề gì về sản phẩm. Chúng ta cần kiểm thử vì biết rằng con người luôn có thể mắc sai lầm bởi vậy kiểm thử phần mềm trở thành một quá trình vô cùng quan trọng.

Việc kiểm thử một ứng dụng có thể được thực hiện thủ công hoặc được tự động hóa bằng các công cụ kiểm thử phần mềm. Với kiểm thử thủ công, Tester đóng vai như một người dùng cuối kiểm thử tất cả các tính năng thiết yếu của ứng dụng hoặc phần mềm. Tester thực hiện các test cases và tạo báo cáo kiểm thử mà không cần sự trợ giúp của bất kỳ công cụ kiểm thử phần mềm tự động nào. Do đó, kiểm thử thủ công tốn nhiều thời gian, và không phải lúc nào nó cũng loại bỏ tất cả các lỗi một cách hiệu quả.

Hiện nay xu hướng áp dụng tự động hoá đang được triển khai rộng rãi ở nhiều lĩnh vực, trong đó có kiểm thử phần mềm. Kiểm thử tự động giải quyết được những thách thức kiểm thử thủ công đưa ra. Đặc biệt, khi kiểm thử phần mềm là công đoạn chiếm phần lớn thời gian trong quá trình phát triển dự án phần mềm thì sự ra đời của các công cụ kiểm thử tự động càng có ý nghĩa hơn bao giờ hết, giúp tiết kiệm thời gian, công sức và tiền bạc.

Bài báo cáo này chúng em xin giới thiệu về công cụ kiểm thử **Watir**, một công cụ hỗ trợ kiểm thử tự động dành cho các ứng dụng Web, hoạt động trên hầu hết các trình duyệt phổ biến hiện nay như Firefox, Chrome, Internet Explorer, v.v. cũng như trên nhiều hệ điều hành khác nhau như windows, Mac, Linux

Bài báo cáo còn rất nhiều thiếu sót do hạn chế về kinh nghiệm thực tế, chúng em rất mong nhận được những ý kiến đóng góp từ cô và các bạn.

1. Lịch sử và giới thiệu chung
   1. Lịch sử

**Watir** được phát triển từ ngôn ngữ lập trình ruby do Chris Morris viết vào năm 2001 để điều khiển Internet Explorer thông qua giao diện COM của nó. Thư viện này có tên cliec.

Năm 2003, Bret Pettichord và Brian Marick bắt đầu sử dụng loại code này làm cơ sở cho một khóa học mà họ đang giảng dạy về kiểm thử tự đông hóa. Lấy cảm hứng từ khóa học này, và nâng cấp để có khả năm hoạt động với các trang web nâng cao hơn, Paul Rogers đã tạo thư viện của riêng mình và gọi nó là WTR (Kiểm tra web bằng Ruby). Thư viện này có nhiều chức năng hơn nhưng API kém hơn.

Bret và Paul đã cùng nhau tham dự hội nghị Agile XP vào tháng 8 năm 2004 để lấy những phần tốt nhất của cliec và WTR và tạo ra một API sạch hơn bằng cách sử dụng các nguyên tắc TDD.

* 1. Giới thiệu chung

**Watir** (**Web Application Testing in Ruby**) là một thư viện **Ruby** mã nguồn mở dành cho kiểm thử tự động. Các nhà phát triển đã sử dụng nó để kiểm thử hệ thống quy mô lớn, kiểm thử chức năng và để tự động hóa kiểm thử chấp nhận của người dùng. Watir sử dụng ngôn ngữ lập trình thực, là một sản phẩm mã nguồn mở miễn phí và cho phép kiểm soát trực tiếp các đối tượng như HTML và JavaScript trong một trang Web. Watir tương tác với một trình duyệt giống như cách mọi người thường làm: nhấp vào các liên kết, điền vào biểu mẫu và xác thực văn bản.

Người dùng watir có thể xây dựng một web scraper (trích xuất nội dung trang web) để mô phỏng việc vào một trang web, đăng nhập, gửi bình luận hoặc tải dữ liệu về và nhiều thứ khác nữa.

Watir cho phép kỹ sư kiểm thử tạo ra những test case dễ đọc hiểu và maintain. Đây là một công cụ khá dễ sử dụng và linh hoạt.

Được phân loại thành 3 loại chính Watir-classic, Watir-webdriver và Watirspec.

1. Watir-classic

Watir-classic chỉ hỗ trợ Internet Explorer trên Windows

Watir-classic sử dụng Ruby tích hợp khả năng Liên kết và Nhúng đối tượng (OLE). Do đó, nó có thể điều khiển Internet Explorer theo chương trình. Watir-classic hoạt động khác với các công cụ kiểm tra dựa trên HTTP (hoạt động bằng cách mô phỏng một trình duyệt). Thay vào đó, Watir-classic trực tiếp điều khiển trình duyệt thông qua giao thức OLE, được triển khai trên kiến trúc Mô hình Đối tượng Thành phần (COM).

1. Watir-webdriver

Watir-webdriver là phiên bản hiện đại của API Watir dựa trên Selenium . Selenium 2.0 (selenium-webdriver) nhằm mục đích là triển khai tham chiếu của đặc tả WebDriver . Trong Ruby, Jari Bakken đã triển khai Watir API như một trình bao bọc xung quanh API Selenium 2.0. Watir-webdriver không chỉ có nguồn gốc từ Selenium 2.0 mà còn được xây dựng từ đặc tả HTML , vì vậy Watir-webdriver phải luôn tương thích với các thông số kỹ thuật W3C(World Wide Web Consortium) hiện có .

Có thể mô tả Watir-webdriver=Watir-classic +WebDiver + 1 số tính năng bổ sung khác.

Watir-WebDriver hỗ trợ trên các trình duyệt phổ biến như Chrome, Firefox, Internet Explorer, Opera hoặc chạy ở chế độ headless (HTMLUnit).

1. Watirspec

Watirspec là đặc tả thực thi của API Watir, nhằm mục đích viết một đặc tả kỹ thuật thực thi đầy đủ cho API Watir.

1. Cơ chế hoạt động

Hiện nay gem watir-classic đã không còn được sử dụng nữa và tất cả source code của watir web-driver cũng đã được đưa vào gem Watir.

Watir từ phiên bản 6.0 trở đi sử dụng Selenium web-driver. có nghĩa là chúng ta có thể tạo một trang web động với JavaScript. Một gem tương tự là Mechanize - phù hợp cho các trang tĩnh và không sử dụng nhiều JavaScript hoặc AJAX.

Watir-WebDriver hỗ trợ trên các trình duyệt phổ biến như Chrome, Firefox, Internet Explorer, Opera hoặc chạy ở chế độ headless (HTMLUnit).

Chế độ headless cho phép phân tích một trang không có giao diện - trong hầu hết các hệ thống UNIX, Watir yêu cầu cài đặt sẵn Xvfb (trên Ubuntu). Watir sử dụng PhantomJS để mô phỏng trình duyệt web và chạy một trang trong trình mô phỏng. Nếu muốn phân tích trang bằng các trình duyệt cần cài các driver tương ứng.

Cũng giống như những ngôn ngữ lập trình khác thì Ruby cho phép người dùng kết nối tới cơ sở dữ liệu, đọc tập tin dữ liệu, truy xuất tập tin XML và cấu trúc những đoạn code thành những thư viện để phục vụ mục đích tái sử dụng.

**Chương 2: Cài đặt và sử dụng (trên Window)**

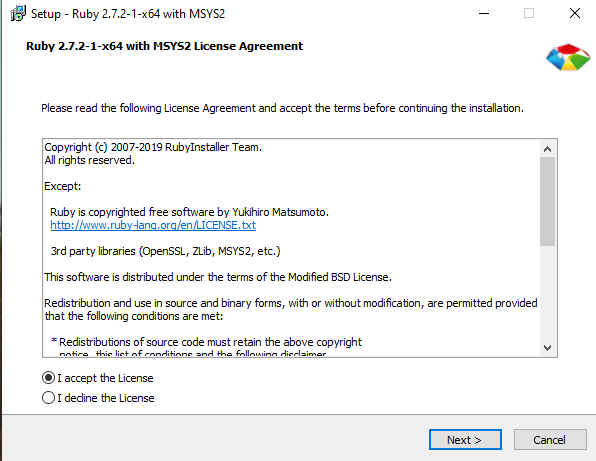
1. Cài đặt Ruby

Trước khi cài Watir, cần tải và cài đặt ruby+devkit lên máy. Truy cập trang: https://rubyinstaller.org/downloads/

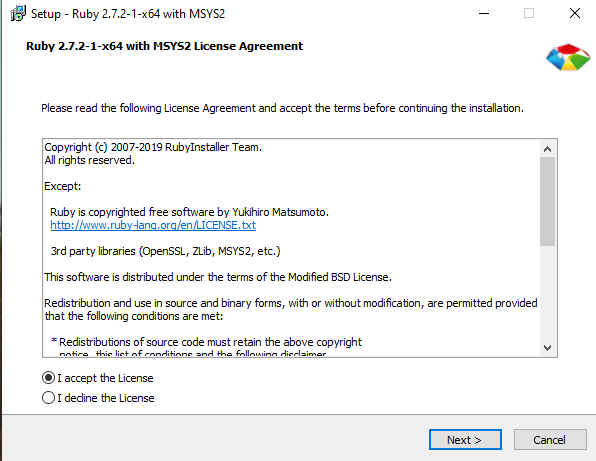
- Chọn phiên bản Ruby phù hợp với hệ điều hành máy tính của bạn.

- Sau khi cài đặt hoàn tất:

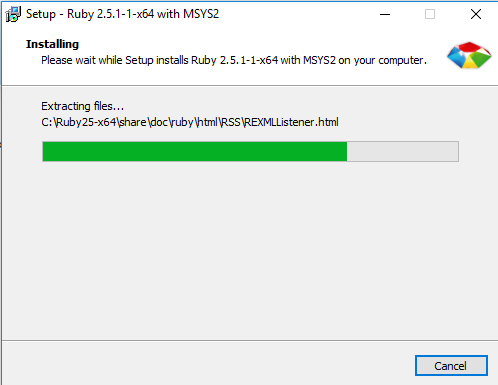
• Chạy trình cài đặt Ruby:



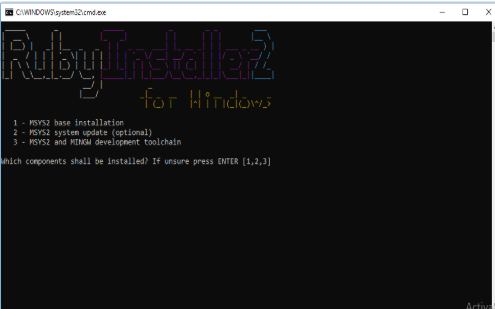
• Chọn chấp nhận điều khoản và ấn Next chuyển qua bước tiếp theo



•Chọn thư mục cài đặt và ấn Install

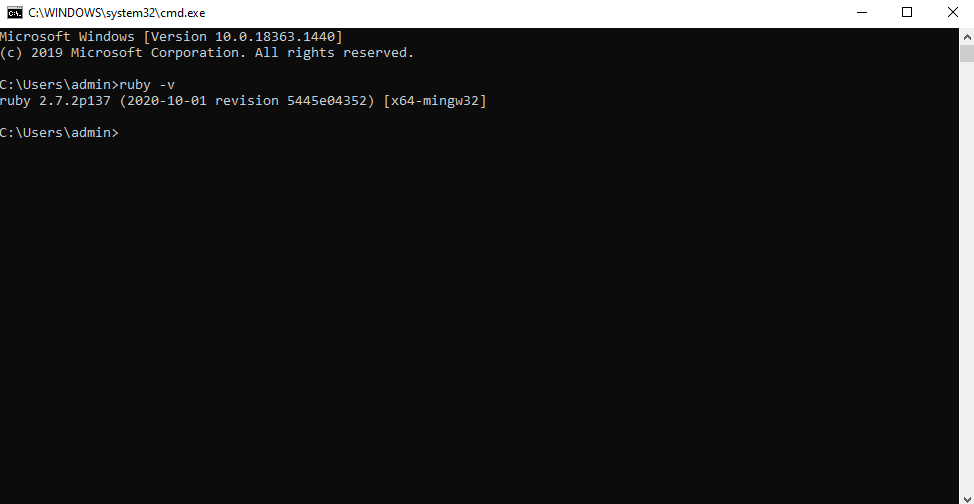


• Đợi quá trình cài đặt kết thúc



• Chọn option 3 - MSYS2 and MINGW development toolchain

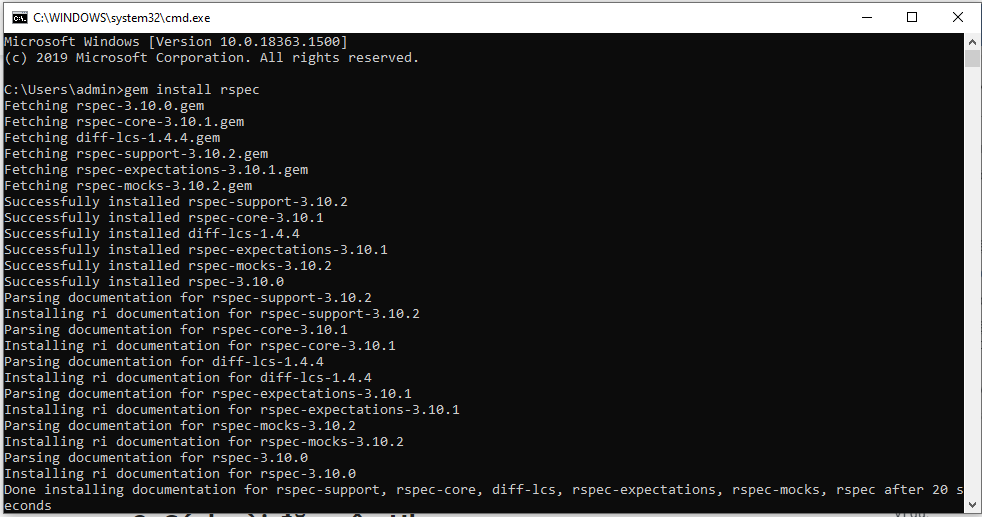
Sau khi cài xong ruby bạn có thể kiểm tra lại xem ruby đã đươc cài thành công hay chưa bằng cách gõ đoạn lệnh ruby -v. Nếu cài đặt thành công sẽ hiện thông báo như dưới đây.



1. Cài đặt Watir

Sau khi cài đặt xong Ruby, việc cài đặt Watir rất đơn giản

Mở command prompt gõ đoạn lệnh: gem install rspec



1. Cài đặt trình điều khiển

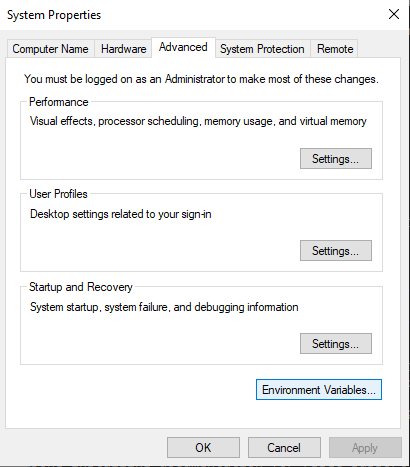
Bạn có thể sử dụng webdrivers gem để đảm bảo rằng trình điều khiển mới nhất được tải xuống và được đặt ở vị trí mà Selenium có thể truy cập nó.

Hoặc có thể thiết lập thủ công bằng cách tải xuống trình điều khiển cho trình duyệt của bạn:

* Google Chrome: chromedriver
* Mozilla Firefox: geckodriver
* Microsoft Edge: Microsoft WebDriver
* Microsoft Internet Explorer: IEDriver
* Apple Safari: safaridriver (safari 10 trở lên được cài đặt sẵn)

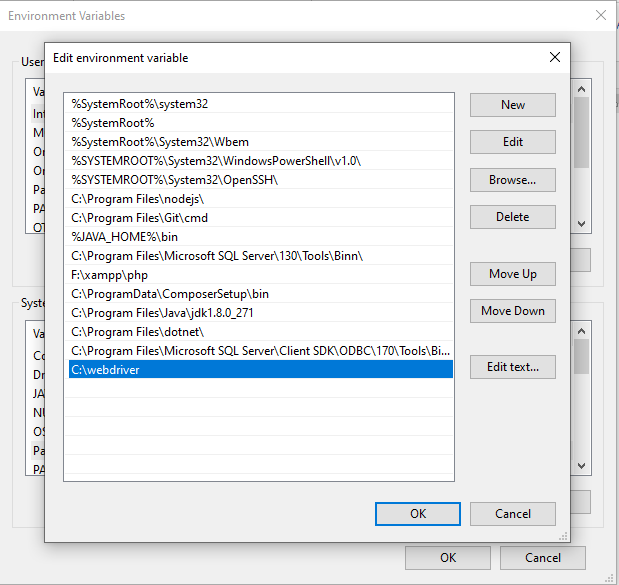
Sau đó bạn cần thêm đường dẫn đến trình điều khiển của mình vào biến môi trường PATH (trừ Safari)

Bước 1: Mở hộp thoại Systerm Properties



Bước 2: Chọn Enviroment Variables, sau đó chọn edit Path

Bước 3: Thêm đường dẫn của thư mục chứa trình điều khiển bạn vừa cài đặt rồi nhấn OK



Hoàn tất quá trình cài đặt.

**Chương 3: DEMO một vài ví dụ**

1. Hướng dẫn sử dụng
   1. Tải thư viện Watir

*require ‘watir’*

* 1. Mở trình duyệt

*browser=Watir::Browser.new:chrome # firefox or ie*

* 1. Mở trình duyệt với URL cụ thể: (mở với trình duyệt mặc định: chrome)

*browser=Watir::Browser.start(“*[*https://www.google.com*](https://www.google.com)*/”)*

* 1. Di chuyển đến URL cụ thể:

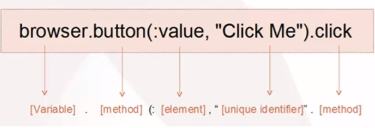
*browser.goto(“*[*https://www.facebook.com/*](https://www.facebook.com/)*”)*

* 1. Đóng trình duyệt

*Browser.close*

* 1. Truy cập tới 1 Element

Cấu trúc:



* + 1. Text box hoặc text area

*t = browser.text\_field({name: "username"})*

hoặc *t = browser.text\_field name: "username"*

* + 1. Button

*b = browser.button({value: "Click Here"})*

* + 1. Drop down list

*d = browser.select\_list({name: "month"})*

* + 1. Check box

*c = browser.checkbox({name: "enabled"})*

* + 1. Radio Button

*r = browser.radio({name: "payment type"})*

* + 1. Form

*f = browser.form({name: "address"})*

*f = browser.form({action: "submit"})*

* + 1. Link

*l = browser.link({url: "http://google.com"})*

*l = browser.link({href: "http://google.com"})*

* + 1. Table: trỏ tới một ô trong bảng (hàng 2 cột 1)

*td = browser.table({name: 'recent\_records'})[2][1]*

* 1. Kiểm tra nội dung
* Trả về html của trang hoặc element:

*browser.html*

*e.html*

* Trả về text của trang hoặc element:

*browser.text*

*e.text*

* Trả về title của tài liệu

*browser.title*

* Lấy text từ status bar

*browser.status*

* Trả về true nếu text mô tả xuất hiện trên trang

*browser.text.include? 'llama'*

* Trả về nội dung của bảng dạng array

*browser.table({id, 'recent\_records'}).to\_a*

1. Demo ví dụ
   1. Ví dụ 1:

Công việc: mở trang “Google.co.in” và nhấp vào liên kết Gmail.



Giải thích mã:

1*) request 'watir'* - yêu cầu thư viện Watir để chạy chương trình

2) *test\_site = “*[*https://www.google.co.in/*](https://www.google.co.in/)*” -* Khai báo biến.

3) browser = Watir :: Browser.new: firefox: Mở trình duyệt Firefox.Ngoài ra, chúng ta có thể mở trình duyệt khác như sau:

*Watir :: Browser.new: internet\_explorer -* Dành cho Internet Explorer

*Watir :: Browser.new: safari -* Dành cho Safari

Nếu chúng ta không đề cập đến tên trình duyệt, thì chúng ta sẽ chỉ cần viết dòng lệnh *[Watir :: Browser.new],* Watir sẽ mở trình duyệt Chrome vì Chrome là trình duyệt mặc định cho Watir.

4) *browser.goto test\_site* hoặc *browser.goto “https://www.google.co.in/” -*  được sử dụng để điều hướng đến trang web được chỉ định.

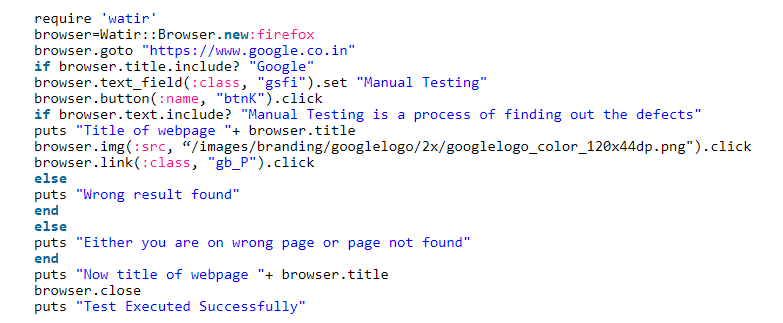
5) *browser.link (: href, “https://mail.google.com/mail/?tab=wmosystem).click* - Tại đây chúng ta nhấp vào liên kết Gmail. Siêu kết nối được coi là một liên kết trong mã và chúng ta đang sử dụng thuộc tính “href” của nó để định vị siêu kết nối này.

6*) puts “Test Executed”* - Dòng này sẽ viết chuỗi “Test Executed” trên dấu nhắc lệnh.

7) *browser.close* - Dòng này sẽ đóng trình duyệt.

* 1. Ví dụ 2:

Công viêc: mở trang web “www.google.co.in”.  Tìm kiếm văn bản " Manual Testing". Cuối cùng quay lại trang chủ và nhấp vào liên kết “Gmail”.



Giải thích mã:

1) *browser.goto https://www.google.co.in* - Điều hướng đến trang web được chỉ định.

2) *if browser.title.include? “Google”* - Ở đây chúng ta đang kiểm tra tiêu đề của trang web xem nó có như mong đợi hay không. Trong phần khác của phần "*if*" này, chúng tôi đưa ra tuyên bố này " *Either you are on the wrong page or page not found*”. Câu lệnh này sẽ được hiển thị trên dấu nhắc lệnh.

3*) browser.text\_field(:class, “gsfi”).set “Manual Testing”* - Ở đây chúng tôi đang nhập văn bản “Kiểm tra thủ công” vào hộp văn bản tìm kiếm.

4*) browser.button(:name, “btnK”).click* - Ở đây chúng tôi đang nhấp vào nút ““Google Search””.

5) *if browser.text.include? “Manual Testing is a process of finding out the defects”* - Bây giờ chúng tôi đang kiểm tra văn bản trên trang web để xác minh xem chúng tôi có nhận được kết quả tìm kiếm chính xác hay không. Trong phần '*else*' của câu lệnh "*if*" này, chúng tôi đặt câu lệnh "Đã tìm thấy kết quả sai".

6) đặt “Tiêu đề trang web“ + browser.title - Ở đây chúng tôi đang hiển thị tiêu đề của trang web trên dấu nhắc lệnh.

7)*browser.img(:src, “/images/branding/googlelogo/2x/googlelogo\_color\_120x44dp.png”).click* - Tại đây chúng tôi nhấp vào hình ảnh để quay lại trang chủ.

8) *browser.link(:class, “gb\_P”).click* - Tại đây chúng tôi đang nhấp vào liên kết Gmail.

9) *puts “Now the title of webpage “+ browser.title* - Bây giờ một lần nữa chúng ta đặt tiêu đề của trang web trên dấu nhắc lệnh. Nó phải là tiêu đề của trang Gmail.

10) *browser.close* - Ở đây chúng ta đang đóng trình duyệt.

11) *puts “Test Executed Successfully*” – Để đảm bảo rằng tất cả các bước được thực hiện chính xác, chúng ta đặt câu lệnh này trên dấu nhắc lệnh.

* 1. Ví dụ 3

Trong ví dụ này, chúng tôi sẽ cố gắng sử dụng các lệnh Watir tối đa. Chúng ta sẽ xem mô tả cùng với mã ở dưới đây.

|  |
| --- |
| require 'watir'  browser=Watir::Browser.new:chrome  browser.goto “http://watir.com/examples/forms\_with\_input\_elements.html”    # Ở đây chúng ta sẽ xem cách chèn dữ liệu vào trường văn bản.  t1=browser.text\_field id: 'new\_user\_first\_name'  t1.set "Testing"  t2=browser.text\_field id: 'new\_user\_last\_name'  t2.set “testdata"    # Ở đây chúng ta sẽ thấy cách chọn dữ liệu từ danh sách chọn.  sl=browser.select\_list id: 'new\_user\_languages'  sl.select 'English'  sl.selected\_options    # Ở đây chúng ta sẽ thấy cách chọn nút radio.  rd=browser.radio value: 'no'  rd.set  chk=browser.checkbox value: 'cars'  chk.set    # Ở đây chúng ta sẽ thấy cách bấm vào nút.  btn=browser.button name: 'new\_user\_button\_2'  btn.click    # Đây là một yêu cầu chờ đợi. Yêu cầu chương trình phải chờ đợi  # cho đến khi có sự kiện sảy ra  Watir::Wait.until {browser.text.include? 'submit'}    # Ở đây tôi đã sử dụng câu lệnh if này vì đây là trang web thử nghiệm  # và không có gì xảy ra sau khi nhấp vào nút. Vì vậy, chỉ để hiển thị  # rằng nút được nhấp, tôi đã sử dụng câu lệnh If  if browser.text.include? "submit"  puts "Data Submitted"  else  puts "Data not yet submitted"  end    # Đây là một tuyên bố chờ đợi ngầm.  # Nó sẽ khiến chương trình của bạn bắt buộc phải đợi trong 30 giây.  browser.driver.manage.timeouts.implicit\_wait = 30    # Bạn có thể gửi các khóa đặc biệt cho trình duyệt.  # Có một danh sách lớn các phím này.  browser.send\_keys :page\_up    # Ở đây chúng tôi đang chụp ảnh màn hình và lưu nó trong một tệp png.  browser.screenshot.save ‘screenshot1.png'    browser.send\_keys :page\_down  browser.screenshot.save ‘screenshot2.png'    puts "TestCase Executed"  browser.close |

**Chương 4: So sánh với một số công cụ khác**

1. So sánh Watir với một vài công cụ kiểm thử

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tính năng | Watir | Selenium | UFT | Katalon Studio |
| Nền tảng phát triển test | Đa nền tảng | Đa nền tảng | Windows | Đa nền tảng |
| Ứng dụng có thể sử dụng | Web apps | Web apps | Windows desktop, Web, mobile apps | Web and mobile apps |
| Ngôn ngữ viết script | Ruby | Java, C#, Perl, Python, JavaScript, Ruby, PHP, Groovy | VBScript | Java/Groovy |
| Hệ điều hành được hỗ trợ | Windows, Mac, Linux | Windows, Mac, Linux | Windows | Windows, Mac, Linux |
| Yêu cầu kĩ năng lập trình | Không nhiều như Selenium Webdriver / UFT.Nhưng cần kiến thức tốt về Ruby | Kỹ năng nâng cao cần thiết để tích hợp các công cụ khác nhau | Không yêu cầu. Cần kỹ năng tạo test scripts | Không yêu cầu. Cần kỹ năng tạo test scripts |
| Mức độ tăng hiểu biết khi tìm hiểu về công cụ | Trung Bình | Cao | Trung Bình | Trung Bình |
| Độ khó khi cài đặt và sử dụng | Dễ. Cấu hình WATIR web driver sử dụng ruby command, yêu cầu phần lớn chỉ là cài đặt Ruby | Khó. Yêu cầu cài đặt và tích hợp nhiều công cụ và thư viện | Dễ dàng cài đặt và chạy | Dễ dàng cài đặt và chạy |
| Thời gian tạo script | Mất nhiều thời gian tương đương Selenium Webdriver | Mất nhiều thời gian | Mất ít thời gian | Mất ít thời gian |
| Lưu trữ đối tượng và bảo trì | Ruby đã xây dựng các khả năng Liên kết và nhúng (Object) và nhúng (OLE) | XPath, UI Maps | Xây dựng dựa vào kho đối tượng để lấy ra đối tượng chính xác | xây dựng dựa vào kho đối tượng, Xpath, nhận dạng lại đối tượng |
| Hỗ trợ kết nối tới Cơ sở dữ liệu (CSDL) | Có thể kết nối đến CDSL | Selenium không thể kết nối tới CSDL của chính nó.Mà nó phụ thuộc vào sự hỗ trợ của một số ngôn ngữ khác ví dụ như Java có JDBC API | Sử dụng Visual Basic script và ODBC, để kết nối với CSDL | Có thể kết nối tới CSDL nhờ sử dụng java |
| Tạo và lưu kết quả test | Sử dụng RSpec framework WATIR có thể tạo HTML report để giữ kết quả test | Selenium Webdriver có thể tạo kết quả test bằng việc tương tác với frameworks khác (như TestNG) | Sau mỗi lần thực hiện test QTP/UFThiển thị kết quả test , mỗi dòng test script sẽ hiển thị pass/fail | Tạo report của test suites với những định dạng như HTML, CSV, PDF và Junit sử dụng context menu trong test explorer |
| Framework được hỗ trợ | Cucumber, RSpec, Test/Unit | C# : Nunit Java: JUnit/TestNG Python: pyunit, py.test, robot Javascript:WebdriverJS, WebdriverIO, Nightwatch JS | Data Driven automation framework Keyword Driven automation framework |  |
| Giấy phép (License type) | Mã nguồn mở | Mã nguồn mở (Apache 2.0) | Độc quyền | Miễn phí |
| Chi phí sử dụng | Miễn phí | Miễn phí | Phí bản quyền và maintain | Miễn phí |

1. Ưu nhược điểm của các công cụ kiểm thử

|  |  |  |
| --- | --- | --- |
| **Công cụ** | **Điểm mạnh** | **Hạn chế** |
| Watir | Mã nguồn mở, miễn phí. Không cần máy chủ bên ngoài để chạy thử nghiệm mà chỉ cần chạy "ruby test.rb". Tốc độ thực thi bài test nhanh hơn selenium. Dễ dàng cài đặt và thực thi | Việc triển khai ở mỗi trình duyệt khác nhau có một chút khác biệt và trình điều khiển cho mỗi trình duyệt được viết độc lập. Do đó, kết quả đưa ra ở mỗi trình duyệt có thể khác nhau. Ngôn ngữ kịch bản chỉ sử dụng ruby. Không được sử dụng rộng rãi như selenium. |
| Selenium | Mã nguồn mở, không có phí cấp phép và bảo trì. Cộng đồng người dùng và phát triển lớn và tích cực để kịp với các công nghệ phần mềm. Có thể tích hợp nhiều công cụ và các framework khác nhau. Sử dụng nhiều ngôn ngữ kịch bản khác nhau | Các nhóm kiểm thử cần phải có kĩ năng và kinh nghiệm lập trình tốt để thiết lập và tích hợp với Selenium với các công cụ và framework khác. Hỗ trợ chậm từ cộng đồng. |
| QTP/UFT | Các tính năng kiểm thử tự động toàn diện được tích hợp vào một hệ thống duy nhất. Hỗ trợ người dùng chuyên dụng với một cộng đồng người dùng lớn được thành lập. Chỉ yêu cầu các kĩ năng lập trình cơ bản để bắt đầu với việc tạo và thực thi kiểm thử. | Tốn kém: giấy phép và phí bảo trì cao đáng kể. Chi phí cao để nâng cấp và các module bổ sung. Chỉ hỗ trợ VBScript |
| Katalon Studio | Không có phí cấp phép và bảo trì cần thiết (dịch vụ hỗ trợ chuyên dụng có trả phí có sẵn nếu cần). Tích hợp các framework và tính năng cần thiết để tạo và thực hiện các trường hợp kiểm thử nhanh. Được xây dựng dựa trên Selenium nhưng loại bỏ nhu cầu về các kĩ năng lập trình nâng cao cần thiết cho Selenium. | Giải pháp mới nổi với một cộng đồng phát triển nhanh chóng. Bộ tính năng vẫn đang phát triển. Thiếu các lựa chọn cho các ngôn ngữ kịch bản: chỉ hỗ trợ Java/ Groovy. |
| Katalon Studio | Không có phí cấp phép và bảo trì cần thiết (dịch vụ hỗ trợ chuyên dụng có trả phí có sẵn nếu cần). Tích hợp các framework và tính năng cần thiết để tạo và thực hiện các trường hợp kiểm thử nhanh. Được xây dựng dựa trên Selenium nhưng loại bỏ nhu cầu về các kĩ năng lập trình nâng cao cần thiết cho Selenium. | Giải pháp mới nổi với một cộng đồng phát triển nhanh chóng. Bộ tính năng vẫn đang phát triển. Thiếu các lựa chọn cho các ngôn ngữ kịch bản: chỉ hỗ trợ Java/ Groovy. |