Contents

[I. Khái niệm 2](#_Toc471768446)

[II. Black box & White box test, trong những kĩ thuật black box & white box test xem các khái niệm liên quan 2](#_Toc471768447)

[**1.** **Kiểm thử hộp trắng** 2](#_Toc471768448)

[**2.** **Kiểm thử hộp đen** 4](#_Toc471768449)

[**3.** **Sự giống và khác nhau giữa kiểm thử hộp trắng và kiểm thử hộp đen** 6](#_Toc471768450)

[III. Dịch sách: tìm hiểu performance test là gì & những kĩ thuật performance test 7](#_Toc471768451)

[IV. User interface test 7](#_Toc471768452)

[V. Xem các testing trên server side client side Tool: 8](#_Toc471768453)

[**1.** **Selenium** 8](#_Toc471768454)

[**2.** **SoapUI** 8](#_Toc471768455)

[**3.** **Gradle** 9](#_Toc471768456)

[**4.** **Mockito** 9](#_Toc471768457)

[**5.** **So sánh sự khác biệt giữa các tool, tool hỗ trợ những gì** 10](#_Toc471768458)

[VI. Maven, Junit 12](#_Toc471768459)

[**1.** **Các khái niệm cơ bản** 12](#_Toc471768460)

[**2.** **Junit** 14](#_Toc471768461)

[**3.** **Maven:** 14](#_Toc471768462)

[VII. SO SÁNH CÁC LOẠI TOOL 15](#_Toc471768463)

[**1.** **Giống nhau** 15](#_Toc471768464)

[**1.** **Khác nhau** 15](#_Toc471768465)

**ÔN TẬP KIỂM CHỨNG PHẦN MỀM**

1. **Khái niệm**

*KCPM là quá trình thực thi một hệ thống phần mềm để quyết định xem nó có thực hiện đúng như bản đặc tả và trong môi trường mong đợi hay không.*

1. **Black box & White box test, trong những kĩ thuật black box & white box test xem các khái niệm liên quan**
   1. **Kiểm thử hộp trắng**
      1. **Khái niệm**

*Kiểm thử hộp trắng là kiểm thử dựa vào thuật toán, cấu trúc code bên trong của chương trình để đảm bảo rằng tất cả các câu lệnh và điều kiện sẽ được thực hiện ít nhất một lần.*

*Kiểm thử hộp trắng giúp kiểm thử được cấu trúc nội bộ hoặc hoạt động của một chương trình, như tương phản với chức năng được bộc lộ của người dùng cuối.*

* Người kiểm thử truy cập vào mã nguồn chương trình và kiểm tra nó, lấy đó làm cơ sở để hỗ trợ việc kiểm thử.
* Kiểm thử hộp trắng **không** đảm bảo chương trình đã tuân theo đặc tả.
* Kiểm thử hộp trắng không thể biết được một chương trình sai do thiếu đường dẫn.
* Kiểm thử hộp trắng không thể phát hiện được lỗi do dữ liệu. Như vậy, kiểm thử chỉ dùng kỹ thuật hộp trắng là không đủ để phát hiện lỗi.
  + 1. **Các loại kiểm thử hộp trắng**
* Kiểm thử đường dẫn (Path test): Kiểm thử bao quát các dòng source code, nhánh và đường dẫn.
* Kiểm thử luồng điều khiển (Control flow test) Xác nhận truy cứu các lịch sử thực hiện code bằng cách sử dụng trình gỡ lỗi.
* Kiểm thử nội bộ: Xác nhận các tham số, counter, vòng lặp
* Kiểm thử tính năng: Đo thời gian xử lý của module, đường dẫn, dữ liệu cụ thể.
  + 1. **Các kỹ thuật được sử dụng trong kiểm thử hộp trắng bao gồm:**
* Kiểm thử API (giao diện lập trình ứng dụng) - kiểm thử ứng dụng có sử dụng các API công cộng và cá nhân.
* Kiểm thử độ bao phủ mã - tạo ra các bài kiểm thử để đáp ứng một số tiêu chí của bảo hiểm mã (ví dụ, các nhà thiết kế kiểm thử có thể tạo ra các bài kiểm thử để làm tất cả các câu lệnh trong chương trình được thực hiện ít nhất một lần).
* Phương pháp chèn lỗi - cố tình đưa ra lỗi để đánh giá hiệu quả của các chiến lược kiểm thử.
* Phương pháp kiểm thử đột biến.
* Phương pháp thử tĩnh.

**Kiểm thử luồng dữ liệu:**

* Là kĩ thuật kiểm thử hộp trắng được sử dụng để phát hiện các lỗi giá trị dữ liệu trong chương trình
* Sử dụng đồ thị luồng dữ liệu dể tìm lỗi
* Phương pháp: 4 bước
* Vẽ đồ thị luồng dữ liệu
* Chọn tiêu chuẩn kiểm thử theo luồng
* Xác định đường đi
* Tạo các test case cho các đường đi
  + 1. **Các khái niệm liên quan  
       Phủ kiểm thử (Coverage**): là tỉ lệ các thành phần thực sự được kiểm thử so với tổng thể sau khi đã kiểm thử các test case được chọn. Phủ càng lớn thì độ tin cậy càng cao.
* **Phủ cấp 0:** kiểm thử những gì có thể kiểm thử được, phần còn lại để người dùng phát hiện và báo lại sau.
* **Phủ cấp 1:** kiểm thử sao cho mỗi lệnh được thực thi ít nhất 1 lần
* **Phủ cấp 2:** kiểm thử sao cho mỗi điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các nhánh (Branch coverage). Phủ các nhánh đảm bảo phủ các lệnh.
* **Phủ cấp 3:** kiểm thử sao cho mỗi điều kiện luận lý con (subcondition) của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các điều kiện con (subcondition coverage). Phủ các điều kiện con chưa chắc đảm bảo phủ các nhánh.
* **Phủ cấp 4:** kiểm thử sao cho mỗi điều kiện luận lý con (subcondition) của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE & điểm quyết định cũng được kiểm thử cho cả 2 nhánh. Ta gọi mức kiểm thử này là phủ các nhánh & điều kiện con (branch & subcondition coverage).
  1. **Kiểm thử hộp đen**
     1. **Khái niệm**

*Kiểm thử hộp đen là kiểm thử các chức năng mà không cần bất kỳ kiến thức về cấu trúc và hành vi bên trong phần mềm, chỉ quan tâm đến đầu vào và đầu ra của chương trình để test mà không quan tâm tới code bên trong được viết ra sao.*

* Còn được gọi là kiểm thử hướng dữ liệu hay là kiểm thử hướng in/out
* Người kiểm thử nên xây dựng các nhóm giá trị đầu vào mà sẽ thực thi đầy đủ tất cả các yêu cầu chức năng của chương trình
  + 1. **Các phương pháp kiểm thử hộp đen bao gồm:**
  + Phân vùng tương đương
  + Phân tích giá trị biên
  + Kiểm thử dựa vào đồ thị
  + Đoán lỗi

1. **Phân vùng tương đương (Equivalence partitioning)**
   * *Chia đầu vào thành những nhóm tương đương nhau. Nếu một giá trị trong nhóm hoạt động đúng thì tất cả các giá trị trong nhóm đó cũng hoạt động đúng và ngược lại*
   * Mục đích: Giảm đáng kể số lượng test case cần phải thiết kế vì với mỗi lớp tương đương ta chỉ cần test trên các phần tử đại diện
2. **Phân tích giá trị biên (Boundary value analysis)**

* *Kiểm thử dựa trên giá trị biên của vùng dữ liệu hợp lệ*
* Phương pháp: 3 bước:
* Xác định lớp tương đương
* Xác định biên tương ứng của các lớp tương đương
* Tạo test case cho các mỗi biên bằng cách chọn một điểm trên biên, 1 điểm dưới biên, 1 điểm nằm trong biên
* Ứng dụng: giảm đáng kể số lượng test case mà đáng ra phải thực hiện, nên phù hợp với các hệ thống có nhiều dữ liệu đầu vào nhận giá trị trong các miền
* Một số điểm cần lưu ý khi dùng phương pháp này:
* Luôn test trường hợp “0”.
* Luôn test các chuỗi rỗng.

1. **Đồ thị nguyên nhân - kết quả (Cause & Effect Graphing)**

Đồ thị nguyên nhân – kết quả hỗ trợ trong việc lựa chọn một cách có hệ thống tập các ca kiểm thử có hiệu quả cao.

Kỹ thuật gồm có 4 bước:

* Xác định điều kiện vào và hành động cho mỗi module cần kiểm định.
* Xác định đồ thị nguyên nhân – kết quả.
* Đồ thị được chuyển thành bảng quyết định.
* Những phần trong bảng quyết định được chuyển thành test case.

1. **Đoán lỗi (Error Guessing)**

* Tester được đưa cho 1 chương trình đặc biệt, họ phỏng đoán, cả bằng trực giác và kinh nghiệm, các loại lỗi có thể và sau đó viết các ca kiểm thử để đưa ra các lỗi đó.
  1. **Sự giống và khác nhau giữa kiểm thử hộp trắng và kiểm thử hộp đen**
     1. **Giống nhau**

Hai loại hình kiểm thử đều nhằm mục đích là kiểm định lại phần mềm nhằm loại bỏ những lỗi và những gì thiếu trong quá trình làm phần mềm nhằm mục đích là đem lại một sản phẩm tốt đến khách hàng.

* + 1. **Khác nhau**

|  |  |  |
| --- | --- | --- |
| **STT** | **Kiểm thử hộp đen** | **Kiểm thử hộp trắng** |
| 1 | Kiểm tra hộp đen là phương pháp thử nghiệm phần mềm được sử dụng để kiểm tra các phần mềm mà không biết cấu trúc nội bộ của mã hoặc chương trình | Kiểm tra hộp trắng là phương pháp kiểm thử phần mềm, trong đó tetster có thể biết cơ cấu nội bộ phần mềm. |
| 2 | Thực hiện bởi testers | Thông thường được thực hiện bởi lập trình viên. |
| 3 | Không yêu cầu kiến thức implement, kiến thưc lập trình | Yêu cầu phải có kiến thức implement, kiến thức lập trình |
| 4 | Thường được áp dụng vào: System Testing, Acceptance testing(higher levels of testing) | Thường được áp dụng vào Unit Testing, Integration testing (lower levels of testing) |
| 5 | functional test or external testing | structural test or interior testing. |
| 6 | Chủ yếu tập trung vào các chức năng của hệ thống đang kiểm tra | Chủ yếu tập trung vào việc kiểm tra của mã chương trình của hệ thống được thử nghiệm như cấu trúc mã, ngành, điều kiện, vòng |
| 7 | Mục đích chính là để kiểm tra về những gì chức năng được thực hiện bởi hệ thống đang kiểm thử. | Mục đích chính là kiểm tra hệ thống đang thực thi như thế nào |
| 8 | Có thể bắt đầu bằng tài liệu yêu cầu chi tiết | Có thể bắt đầu bằng tài liệu thiết kế cơ bản |

1. **Dịch sách: tìm hiểu performance test là gì & những kĩ thuật performance test**

**Performance testing** là việc thực hiện test để xác định một hệ thống có thể đáp ứng và ổn định với yêu cầu độ tải cao. Nó có thể phục vụ để điều tra, đo đạc, xác nhận hoặc xác minh chất lượng các thuộc tính của hệ thống như: khả năng thay đổi, tính tin cậy, và tài nguyên sử dụng.

Các kĩ thuật của Performance testing

* **Load testing** là một quá trình thêm nhu cầu vào một hệ thống hoặc thiết bị và đo lường phản ứng của nó. Load testing được thực hiện để xác định ứng xử của hệ thống trong các điều kiện tải bình thường và cao hơn điều kiện tải dự kiến. Nó giúp xác định công suất vận hành tối đa của một ứng dụng như các điểm “thắt cổ chai” (bottleneck) và xác định phần tử nào là nguyên nhân gây ra điều đó. Khi mức tải trên hệ thống vượt ra ngoài những cách thức sử dụng bình thường – để kiểm tra phản ứng của hệ thống ở những điểm cao bất thường hoặc cao điểm tải – thì được gọi là stress testing.
* **Stress testing** là một hình thức kiểm thử được sử dụng để xác định tính ổn định của một hệ thống hoặc một thực thể được đưa ra. Nó liên quan đến những kiểm thử vượt quá khả năng bình thường của hệ thống, thường để xác định điểm phá vỡ, để quan sát các kết quả.
* **Volume testing** thuộc về nhóm kiểm thử phi chức năng, nó thường bị hiểu nhầm hoặc dùng thay thế cho loại kiểm thử khác. Volume testing đề cập tới việc kiểm thử phần mềm ứng dụng với một lượng dữ liệu nhất định. Số lượng này có thể là kích thước cơ sở dữ liệu hoặc nó cũng có thể là kích thước của 1 tập tin giao tiếp là đối tượng của volume testing.

1. **User interface test**

User Testing là thử nghiệm người dùng, kiểm tra mức độ hài lòng của người dùng đối với một website/ứng dụng nào đó.

Các UX Designer sẽ theo dõi hành vi người dùng trên website/ứng dụng và phân tích. Kết quả cuối cùng là tìm ra những ưu điểm và nhược điểm trên từng giao diện để khắc phục và sửa chữa.

1. **Xem các testing trên server side client side Tool:** 
   1. **Selenium**

* **Khái niệm**: là một công cụ kiểm thử phần mềm tự động, gồm có 4 phần:
* IDE: công cụ cho phép chúng ta record/ play back 1 test script. Là một add-on hỗ trợ trên firefox, record trên firefox mà play back đa trình duyệt
* Grid: hệ thống hỗ trợ người dùng thực thi test script trên nhiều trình duyệt một cách song song mà không cần chỉnh sửa test script
* RC, Web Driver: là thư viện cho phép chúng ta lập trình test script trên nhiều ngôn ngữ lập trình khác nhau: python, java, c#...

Cấu trúc Selenium có 4 phần:

* Selenium IDE
* Selenium RC (Selenium 1 – Selenium Remote Control)
* Selenium Gird
* Selenium WebDriver (Selenium 2)
* **Đặc điểm:**
* Mã nguồn mở
* Cộng đồng hỗ trợ
* Hỗ trợ nhiều ngôn ngữ
* Chạy trên nhiều hệ điều hành khác nhau
* Chạy test script ở background
* Không hỗ trợ Win app
  1. **SoapUI**
* **Khái niệm:** là công cụ kiểm thử mã nguồn mở API.
* **Đặc điểm:**
* Giao diện đơn giản, thân thiện, dễ sử dụng
* Kiểm thử chức năng: Hỗ trợ tính năng kéo thả, gỡ lỗi
* Kiểm thử bảo mật: Ngăn chặn SQL Injection để bảo đảm dữ liệu, thực hiện fuzzing scan và boundary scan để tránh các hành vi bất thường
* Kiểm thử tải: Kiểm thử khả năng chịu tải của ứng dụng web sử dụng loadUI, cho phép báo cáo chi tiết để nắm hiệu suất
* Hỗ trợ các giao thức và công nghê: SOAP, REST, HTML, JMS,…
  1. **Gradle**

**Định nghĩa:** Là một công cụ tự động build project dựa trên các khái niệm của Apache Ant và Apache Maven, đồng thời giới thiệu một “domain specific language” dựa trên Groovy thay cho hình thức khai báo cấu hình truyền thống (XML) của project. Được tích hợp sẵn tự động trong  Android Studio để giúp những nhà phát triển tổ chức, quản lý và từng bước xây dựng ứng dụng của họ.

**Đặc điểm:**

* + - Rất linh hoạt
    - Mạnh trong việc xây dựng các đa dự án
    - Quản lí các phụ thuộc (dependency) rất tốt (dựa trên Apache Ivy)
    - Sử dụng DST,Groovy là build script
    - Là một mô hình phong phú để miêu tả dự án
  1. **Mockito**

**Khái niệm:**

* + 1. *Khái niệm/mục đích*
       - Mockito là một testing framework mã nguồn mở cho Java phát hành theo Giấy phép MIT. Framework cho phép tạo ra các đối tượng đôi kiểm tra (đối tượng giả) trong unit tests cho mục đích phát triển Test-driven (TDD) hay Behavior Driven Development (BDD).
       - Mocking là một cách để kiểm tra các chức năng của một lớp độc lập.Mocking không đòi hỏi một kết nối cơ sở dữ liệu hay tập tin thuộc tính đọc hoặc máy chủ tập tin đọc để thử nghiệm một chức năng. các đối tượng giả làm mocking của dịch vụ thực sự. Một đối tượng giả trả về một dữ liệu giả tương ứng với một số đầu vào giả truyền cho nó.
       - Mockito là một mock framework phổ biến mà có thể được sử dụng kết hợp với JUnit. Mockito cho phép bạn tạo và cấu hình các đối tượng giả. Sử dụng Mockito đơn giản hoá các bài test cho các lớp học với phụ thuộc bên ngoài một cách đáng kể.
    2. *Hỗ trợ*
       - Không có chữ viết tay - Không cần phải viết các đối tượng giả trên của riêng bạn.
       - Refactoring An toàn - Đổi tên tên phương pháp giao diện hoặc sắp xếp lại các thông số sẽ không phá vỡ các mã kiểm tra như Mocks được tạo ra tại thời gian chạy.
       - Hỗ trợ quay trở lại giá trị
       - Hỗ trợ ngoại lệ - Hỗ trợ trường hợp ngoại lệ.
       - Hỗ trợ kiểm tra - Hỗ trợ kiểm tra theo lệnh của phương thức gọi.
       - Hỗ trợ chú thích - Hỗ trợ tạo mocks sử dụng chú thích
  1. **So sánh sự khác biệt giữa các tool, tool hỗ trợ những gì**

Automation Test là việc sử dụng một phần mềm để quản lý một quá trình kiểm thử.

Công việc chính của chúng ta là tạo ra các điều kiện kiểm thử, điều khiển quá trình kiểm thử và báo cáo kết quả kiểm thử

Một cách khái chung là test tự động bao gồm tự động quá trình chạy bằng tay quá trình mà chúng ta kiểm thử.

**Ưu điểm:**

+ Kỹ thuật viên ít can thiệp

+ Giảm thời gian và công sức cho kỹ thuật viên

**Khuyết điểm:**

+ Tốn chi phí tạo script

+ Tốn chi phí bảo trì các test script

+ Đòi hỏi các kỹ thuật viên có kỹ năng tạo script cao

|  |  |
| --- | --- |
| **Test Tool** | **Tính năng** |
| **Selenium** | * + Là framework đơn giản   + Cho phép record/Playback đoạn test scrip dưới nhiều ngôn ngũ khác nhau như Java, C#, Python, Ruby   + Add-on hỗ trợ Firefox phiên bản 2.0 trở lên |
| **SOAP UI** | * + Kiểm thử chức năng:   + Một công cụ mạnh mẽ cho phép tester viết Functional API Tests trong SoapUI   + Hỗ trợ tính năng kéo-thả mà làm tăng tốc độ phát triển scriptHỗ trợ gỡ lỗi và cho phép tester phát triển data driven tests.   + Kiểm thức bảo mật: Ngăn chặn SQL Injection để bảo đảm cơ sở dữ liệu   + Kiểm thức tải : thử khả năng chịu tải của một ứng dụng web sử dụng loadUI. Tích hợp với các công cụ tự động khác: Maven, HUDSON, Junit, Apache - Ant |
| **Mockito** | Kiểm tra các chức năng của một lớp độc lập  cho phép tạo ra các đối tượng đôi kiểm tra (đối tượng giả) trong unit tests cho mục đích phát triển Test-driven (TDD) hay Behavior Driven Development |
| **Gradle** | Quản lí các phụ thuộc (dependency) rất tốt (dựa trên Apache Ivy)  Sử dụng DST,Groovy là build script  Là một mô hình phong phú để miêu tả dự án |

1. **Maven, Junit**
   1. **Các khái niệm cơ bản**
      1. **Unit Test**
   * Unit Testing là một kỹ thuật kiểm thử mọi đơn vị hoạt động của mã nguồn với một quy trình tách biệt so với quy trình phát triển của phần mềm, giúp phát hiện sai sót kịp thời.
   * Unit Test là một phần mã nguồn dùng để kiểm tra một phần mã nguồn khác. Các đoạn mã Unit Test hoạt động liên tục hoặc định kỳ để thăm dò và phát hiện các tiềm ẩn, lỗi kỹ thuật trong suốt quá trình phát triển.
   * Unit Testing là một kỹ thuật quan trọng trong Test driven development.
   * Unit Testing là phương pháp bổ sung cho các phương pháp kiểm thử khác, giúp phát hiện lỗi từ sớm, ngay từ ý tưởng thiết kế (reviews code, walkthroughs…).

* Unit Test được viết bởi người phát triển. Test “White Box”, “Black-Box” trong quá trình phát triển phần mềm.
* Unit of Code: mỗi đơn vị mã nguồn có thể là chương trình riêng lẻ, chức năng, thủ tục, class, phương thức,…
* Unit Test có 3 trạng thái: Fail, Ignore, Pass.
* Mỗi UT đều được tiết kế theo trình tự sau:

+ Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng.

+ Gọi các phương thức cần kiểm tra.

+ Kiểm tra sự hoạt động của phương thức

+ Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

- Lợi ích:

+ Đảm bảo chất lượng từng Unit trong phần mềm.

+ Phát hiện lỗi sớm và chỉnh sửa kịp thời.

+ Giảm chi phí bảo trì và kiểm thử.

+ Dễ tích hợp.

+ Tài liệu hóa.

+ Giúp Thiết Kế.

* + 1. **Mock object**

Là một đối tượng ảo, mô phỏng các tính chất và hành vi giống hệt như đối tượng thực được truyền vào bên trong khối mã đang vận hành nhằm kiểm tra tính đúng đắn của các hoạt động bên trong.

Đặc điểm:

+ Đơn giản hơn đối tượng thực nhưng vẫn giữ được sự tương tác với các đối tượng khác.

+ Không lặp lại nội dung đối tượng thực.

+ Cho phép thiết lập các trạng thái riêng trợ giúp kiểm tra.

- Lợi ích:

+ Đảm bảo công việc kiểm nghiệm không bị gián đoạn bởi các yếu tố bên ngoài, giúp các chuyên viên tập trung vào một chức năng nghiệp vụ cụ thể  UnitTest vận hành nhanh hơn.

+ Giúp tiếp cận hướng đối tượng tốt hơn. Nhờ MO có thể phát hiện interface cần tách ở một số lớp.

+ Dễ dàng cho việc kiểm nghiệm.

- Cách thực hiện:

+ Đưa ra interface mô tả đối tượng.

+ Viết nội dung cho đối tượng thực dựa trên interface như thông thường.

+ Trích interface từ đối tượng thực và triển khai MO dựa trên interface đó.

* Stub là phần cài đặt một interface và trả về giá trị giả cho phương thức trong interface.
* Fakes là phần cài đặt một interface là bản làm giả bản thật nằm trong bộ nhớ (cài đặt bán hoàn chỉnh).

Test-driven development

TDD là một chiến lược phát triển sử dụng kỹ thuật UT theo nguyên tắc tạo ra các công đoạn kiểm nghiệm trước khi xây dựng mã.

* 1. **Junit**
     1. **Định nghĩa:** Framework đơn giản dùng cho việc tạo các unit testing tự động. Cung cấp cho chúng ta các gói lớp có sẵn cho phép chúng ta viết các phương thức test một cách dễ dàng. Chạy các test có thể lặp đi lặp lại
     2. **Đặc điểm:**

Xác nhận việc kiểm tra kết quả được mong đợi

Các Test Suite cho phép chúng ta dễ dàng tổ chức và chạy các test

Hỗ trợ giao diện đồ họa và giao diện dòng lệnh

* 1. **Maven:**
     1. **Khái niệm:** là chương trình quản lý dự án được sử dụng nhiều bởi Java developer, nó thực hiện các công việc như khởi tạo project, biên dịch, đóng gói, chạy test… một cách tự động và nhanh chóng. Nó đặc biệt hữu dụng với các dự án tầm trung và lớn.
     2. **Đặc điểm:**

Giúp các nhà phát triển Java dễ dàng hơn trong việc quản lý các dự án Java.

Cung cấp các tính năng như Unit Test, Integration Test

Cung cấp các cách thiết lập dự án nhanh chóng chỉ trong vài giây

Tạo một chuẩn thống nhất cho tất cả các dự án

Chạy các dự án song song (Maven 3)

1. **SO SÁNH CÁC LOẠI TOOL**
2. **Giống nhau**

Đều là những công cụ kiểm thử ứng dụng web tự động, là những công cụ mã nguồn mở, có thể triển khai trên đa nền tảng như: Windows, Ubuntu,…

1. **Khác nhau**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tool/Tiêu chí** | **SOAP UI** | **Sahi** | **Watir** | **Selenium** |
| Nền tảng | - có trình thuật sĩ (Wizard Setup) riêng để cài chương trình, sau khi cài sử dụng như các chương trình windows bình thường (sử dụng tệp .exe) | - Chạy trên nhiều trình duyệt khác nhau (IE,FF,Chrome,Opera,Safari) - Sử dụng các thuật toán khác nhau đê xác định các phần tử mà con người dễ dàng nhận thấy đc | Chạy trên các trình duyệt Firefox, Chrome, IE, Opera | -Chạy trên nhiều trình duyệt ( trừ IDE có record trên firefox) |
| Ngôn ngữ | - Groovy, JavaScript | Sahi Script, Java, Ruby Sahi Script, | ASP.Net, JSP,PHP, Rails | Java, Ruby, Perl, Python, C# |
| Tính dễ sử dụng | - Giao diện trực quan, dễ nắm bắt - cần có kiến thức chuyên sâu về API để có thể sử dụng thành thạo | - Cần cài đặt, thời gian từ 10-30 phút, phụ thuộc vào lúc cài đặt java - Hỗ trợ APIs cho các task phức tạp | Hỗ trợ test case đơn giản, linh hoạt, dễ đọc | - Thời gian cài đặt dưới 5 phút vì sử dụng plugin của firefox - Cần kiến thức ngôn ngữ lập trình |
| Tính ổn định của script và dễ dàng bảo trì | ổn định | - Không phụ thuộc XPath - Sử dụng \_near, and \_in | Dễ bảo trì | - Chạy test script trên background - Khó khăn cho tester trong việc hiểu và sửa lỗi |