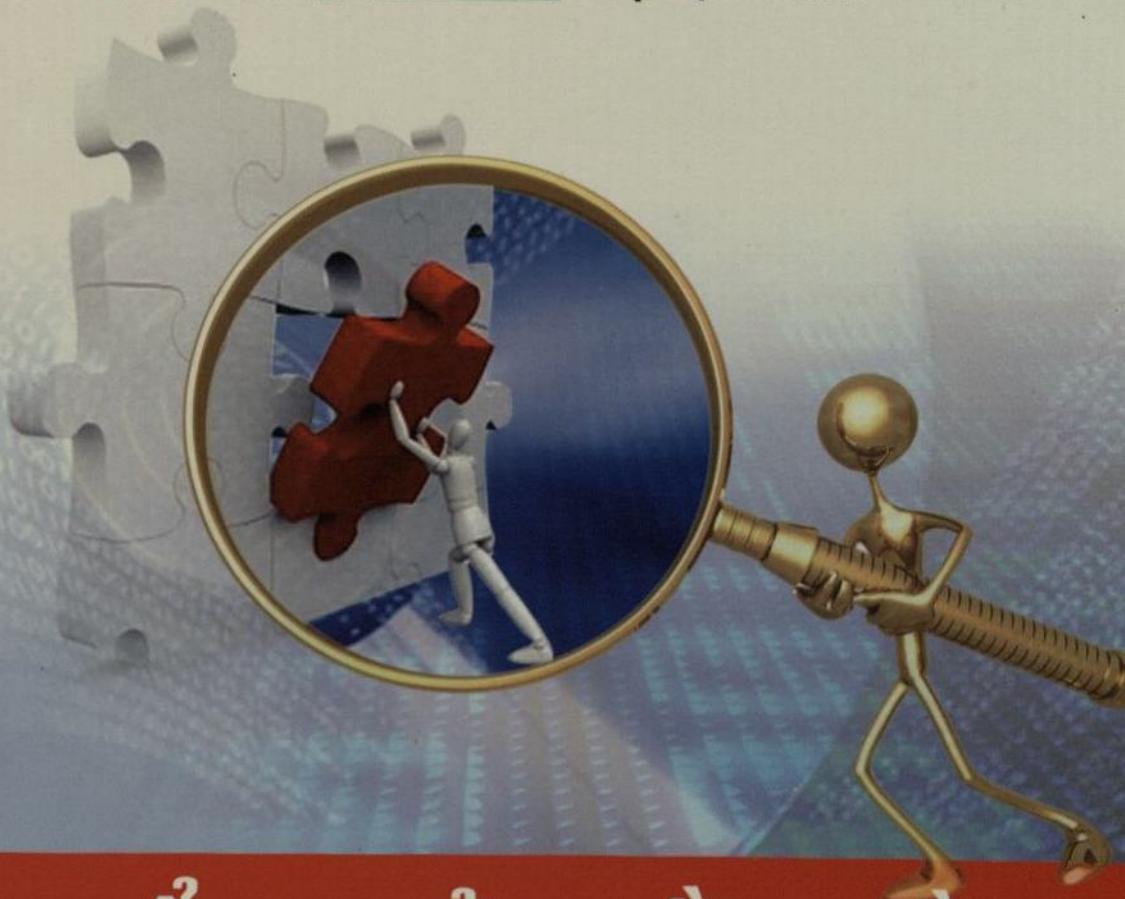




TRẦN TƯỜNG THỰ  
PHẠM QUANG HIỂN



# KIỂM THỬ PHẦN MỀM (TESTING)



QuickTest Pro

Junit

Selenium IDE

Selenium RC



NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

**TRẦN TƯỜNG THỤY - PHẠM QUANG HIỂN**

KIỂM THỬ  
PHẦN MỀM

**NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG**

# TÀI LIỆU THAM KHẢO

Trong sách "Kiểm thử phần mềm" này, các tác giả có tham khảo và trích dẫn một số tài liệu trên trang web tiếng Việt và tiếng Anh lớn nhất về lãnh vực kiểm thử phần mềm:

<http://www.testingvn.com>

<http://www.vietnamesetestingboard.org/>

<http://www.logigear.com/>

2 tài liệu tiếng Việt được đánh giá thích hợp với những ai bắt đầu làm quen với Kiểm thử phần mềm là:

- Luận văn Thạc sỹ "Một số kỹ thuật kiểm thử phần mềm" của tác giả Cao Thị Bích Liên (Đại học Kỹ thuật Thái Nguyên-2009).
- <http://www.testingvn.com/viewtopic.php?f=74&t=491>

<http://webluanvan.com/f32/mot-so-ky-thuat-kiem-thu-phan-mem-19246/>

và tài liệu:

- Testing Applications on the web: Test Planning for Mobile and Internet-Based System

Một số tài liệu liên quan đến kiểm thử phần mềm có thể tải về từ:

- <http://www.mediafire.com/myfiles.php#myfiles>

# GIỚI THIỆU VỀ KIỂM THỬ PHẦN MỀM

Khoa học kỹ thuật ngày càng phát triển với tốc độ cao, hàng loạt các sản phẩm phần mềm được đưa ra phục vụ cho con người. Mỗi ngày chúng ta đều nghe đâu đó tin tức về vấn đề an toàn, bảo mật thông tin, 1 ngân hàng báo cáo số dư tài khoản không chính xác, 1 đoàn tàu bị va chạm, 1 thiết bị hạ cánh trên sao Hỏa bị mất trong không gian, 1 máy quét cửa hàng tạp hoá tính phí quá nhiều cho sản phẩm hoặc một hacker truy cập đến hàng triệu thẻ tín dụng. Tại sao điều này xảy ra? Có thể do lập trình viên máy tính không tìm ra cách để làm cho phần mềm đơn giản?

Thật không may, phần mềm ngày càng trở nên phức tạp hơn, có được nhiều tính năng hơn, được kết nối với nhau nhiều hơn và cũng có nhiều trực trắc hơn từ chương trình. Việc xây dựng và phát triển các sản phẩm phần mềm ngày càng được nâng cao hơn bằng các công cụ hỗ trợ tiên tiến. Nhờ vào đó mà các chuyên gia phát triển thực hiện hiệu quả và đem lại nhiều lợi nhuận hơn trước. Tuy nhiên, với công nghệ ngày càng cao thì đòi hỏi về mức độ ứng dụng lớn và phát sinh ra sự phức tạp cùng với chi phí, thời gian tăng lên. Do đó phương pháp để cải thiện điều này chính là thực hiện kết hợp giữa xây dựng và quá trình kiểm thử. Hầu hết các công ty phần mềm lớn đều cam kết về chất lượng phần mềm do họ tạo ra, họ có một hoặc nhiều xét nghiệm cho mỗi bộ lập trình. Tuy nhiên việc sử dụng, mở rộng các phần mềm từ các trò chơi máy tính, tự động hóa nhà máy cho đến các ứng dụng kinh doanh sẽ luôn có vấn đề về phần mềm...Kiểm thử phần mềm là một trong những nhiệm vụ khó khăn nhất hiện có.

Có nhiều lý do cho việc này:

- Một sản phẩm phần mềm không phải là một đối tượng hữu hình có thể đo đạc, cơ thể cảm thấy, hoặc lấy mẫu. Vì vậy, rất khó khăn để thử nghiệm một sản phẩm phần mềm.
- Kiểm thử phần mềm vẫn không được coi là một trao đổi thương mại được công nhận, do đó, việc tìm kiếm được những người chuyên nghiệp đủ điều kiện cho các công việc thử nghiệm là khó khăn.
- Không giống như quá trình sản xuất đã được xác định và tiêu chuẩn hóa thiết kế sản phẩm trong phát triển sản phẩm, kiểm soát chất lượng, quy trình tương tự như tiêu chuẩn hóa vẫn chưa được xác định để thử nghiệm phần mềm.
- Các công cụ tự động hóa hoạt động kiểm thử phần mềm vẫn còn trong giai đoạn mới bắt đầu, còn phải mất nhiều thời gian để có các công cụ tự động hóa tính vì có sẵn cho các hoạt động kiểm thử phần mềm.

- Nỗ lực tìm kỹ thuật mới cho các hoạt động thử nghiệm phần mềm vẫn đang được phát triển..

Tầm quan trọng của kiểm thử phần mềm là quá bao la! bất cứ thất bại của sản phẩm phần mềm hoặc ứng dụng đều có thể gây thiệt hại hàng tỷ đồng cho công ty. thậm chí nếu các lỗi phần mềm không phải là quá lớn, chi phí hỗ trợ để có thể chạy thử cũng mất cả chục tới cả trăm triệu trong vòng đời của sản phẩm phần mềm. Để hiểu rõ hơn về kiểm thử phần mềm, các bạn cũng như chúng tôi lần đầu tiên tiếp xúc với lãnh vực này cố gắng để hiểu:

Một khuyết khuyết trong sản phẩm là gì, làm thế nào nó ảnh hưởng đến người sử dụng, những gì người dùng cảm thấy khi tìm thấy một khuyết khuyết trong sản phẩm sau khi mua và sử dụng nó, làm thế nào để ngăn ngừa nó, và cuối cùng là làm thế nào để xác định và loại bỏ các khuyết khuyết trong thế giới vật lý. Từ đó chúng ta có thể đi đến công nghệ phần mềm và kiểm thử phần mềm.

Cuốn sách "Kiểm thử phần mềm" gồm 24 chương giới thiệu tới bạn những điều cơ bản của kiểm thử phần mềm trong giáo dục, giảng dạy cả lý thuyết lẫn thực hành không chỉ là kỹ thuật cơ bản mà còn cần những kỹ năng hỗ trợ cần thiết để thử nghiệm phần mềm thành công. Bạn sẽ tìm hiểu làm thế nào để tìm thấy ngay vấn đề trong bất kỳ chương trình máy tính nào, làm thế nào để lập kế hoạch tiếp cận thử nghiệm hiệu quả, làm thế nào để báo cáo rõ ràng phát hiện của bạn, và làm thế nào để biết khi nào phần mềm của bạn đã sẵn sàng cho phát hành? Bạn cần biết, phần mềm của bạn vẫn sẽ được phát hành với lỗi. Tuy nhiên, bằng cách áp dụng những kiến thức trong cuốn sách này bạn sẽ đi một chặng đường dài hướng tới đảm bảo rằng: các lỗi quan trọng nhất không nhiều và nhóm lập trình viên của bạn sẽ tạo ra phần mềm có chất lượng và an toàn nhất có thể.

### **AI NÊN SỬ DỤNG CUỐN SÁCH NÀY?**

Cuốn sách này được viết cho ba nhóm người khác nhau:

- Sinh viên hoặc người có sở thích quan tâm đến máy tính trong kiểm thử phần mềm là một công việc toàn thời gian, thực tập, hoặc làm việc theo nhóm. Đọc cuốn sách này trước cuộc phỏng vấn của bạn hoặc trước ngày đầu tiên của bạn trong công việc sẽ thực sự gây ấn tượng với ông chủ mới của bạn.
- Muốn đổi mới công việc, chuyển từ lĩnh vực chuyên môn của họ vào ngành công nghiệp phần mềm. Có rất nhiều cơ hội cho các chuyên gia phần mềm không áp dụng kiến thức của họ để kiểm thử phần mềm. Ví dụ, một người hướng dẫn bay có thể thử nghiệm một trò chơi mô phỏng chuyến bay, một kế toán có thể kiểm tra phần mềm chuẩn bị thuế, hoặc một giáo viên có thể thử nghiệm một chương trình giáo dục trẻ em mới.

- Các lập trình viên, quản lý dự án phần mềm, và những người khác tạo nên một đội ngũ phát triển phần mềm, những người muốn nâng cao kiến thức và sự hiểu biết của họ về những gì kiểm thử phần mềm mang lại.

Cuốn sách này còn giúp bạn đọc hiểu rõ hơn một khía cạnh khác của kiểm thử phần mềm:

- Làm thế nào kiểm thử phần mềm phù hợp với quá trình phát triển phần mềm.
- Các kỹ thuật kiểm thử phần mềm cơ bản và nâng cao.
- Áp dụng các kỹ năng kiểm tra với nhiệm vụ thử nghiệm phổ biến.
- Nâng cao hiệu quả thử nghiệm với tự động hóa.
- Lập kế hoạch và tài liệu nỗ lực thử nghiệm của bạn.
- Hiệu quả báo cáo những vấn đề bạn tìm thấy.
- Đo nỗ lực thử nghiệm của bạn và sự tiến bộ của sản phẩm của bạn.
- Biết sự khác biệt giữa kiểm tra và đảm bảo chất lượng.
- Tìm một công việc như là một thử nghiệm phần mềm.

Phần mềm cần thiết để sử dụng cuốn sách này:

**Lưu ý:** các phương pháp trình bày trong cuốn sách này là nguyên lý chung và có thể được áp dụng để kiểm tra bất kỳ loại phần mềm máy tính nào. Các ví dụ được sử dụng trong suốt cuốn sách này với các ứng dụng khác nhau, lỗi phần mềm, và các công cụ kiểm tra phần mềm chỉ là một sự chứng thực, chỉ đơn giản được sử dụng để chứng minh các khái niệm về kiểm thử phần mềm. Các bạn có thể vận dụng linh hoạt trong việc chọn các trang web để kiểm thử không nhất thiết theo hướng dẫn trong sách.

Kiểm thử giúp rút ngắn thời gian và giảm chi phí cho sản phẩm phần mềm. Nó giúp cho các chuyên viên kiểm thử tìm ra lỗi trong quá trình tạo phần mềm và đảm bảo hơn về chất lượng. Kiểm thử thực hiện chặt chẽ sẽ hạn chế lỗi, tuy nhiên trong phần mềm vẫn còn tiềm ẩn các lỗi và có thể phát sinh bất cứ lúc nào, dẫn đến khả năng gây thiệt hại cho nhà sản xuất hay xây dựng. Vì vậy cần thực hiện quá trình kiểm thử liên tục, xuyên suốt trong các giai đoạn phát triển của phần mềm. Đó là phương pháp tốt nhất để đảm bảo cho các yêu cầu của người dùng về thiết kế và ứng dụng phần mềm được đáp ứng đầy đủ.

Để kiểm thử không mất nhiều thời gian và chi phí, người ta bắt đầu thực hiện đưa nó vào trong các ứng dụng xây dựng phần mềm để có thể kết hợp chúng lại với nhau. Việc làm này biến quy trình kiểm thử trở thành quy trình bắt buộc thực hiện trong mỗi dự án phần mềm. Nhưng để có được hiệu suất cao nhất người thực hiện kiểm thử cần có phương án, kế hoạch hay chiến lược riêng cho từng ứng dụng phần mềm.

Thực tế để đảm bảo được hai tiêu chí là ứng dụng chạy đúng và đáp ứng yêu cầu đặt ra của khách hàng, đòi hỏi nó phải vận hành tốt trong quá trình kiểm thử phần mềm và phải tổ chức và duy trì sự hoạt động nhịp nhàng của cả một hệ thống các công việc liên quan đến một dự án phần mềm, từ đây xuất hiện một khái niệm có tên là "hệ thống quản lý chất lượng phần mềm" bao gồm các quy trình được thực thi xuyên suốt chu kỳ phát triển của dự án phần mềm song hành cùng việc kiểm thử phần mềm nhằm đảm bảo chất lượng cho phần mềm khi chuyển giao cho khách hàng.

Với thực tế trên, các tác giả mong muốn cung cấp cho sinh viên chuyên ngành tin học - những người sẽ là nguồn nhân lực chủ yếu trong tương lai của các doanh nghiệp phần mềm – những khái niệm, kiến thức và kỹ năng cơ bản ban đầu về kiểm thử phần mềm, các công đoạn kiểm thử, các loại kiểm thử, công cụ kiểm thử, xây dựng tài liệu kiểm thử, dữ liệu kiểm thử ... về qui trình quản lý chất lượng, đảm bảo chất lượng phần mềm thông qua tài liệu (nội bộ) Kiểm thử và đảm bảo chất lượng phần mềm. Đây là tài liệu cơ bản, còn nhiều vấn đề chưa rõ sâu phân tích và thực hiện, còn mang tính lý thuyết nhiều. Tác giả hy vọng bạn đọc đóng góp ý kiến để trong lần biên soạn tập tiếp theo về chủ đề này đáp ứng tốt hơn yêu cầu của nhiều độc giả, của sinh viên và kể cả những người đang công tác tại các phòng phát triển và đảm bảo chất lượng phần mềm.

Sách có thể được dùng làm tài liệu tham khảo cho sinh viên các trường Đại học-Cao đẳng chuyên ngành tin học.

File thực hành trong sách có thể tải về theo đường dẫn sau:

<http://www.mediafire.com/?1r25pzc2nvis3t8>

Các tác giả chân thành cảm ơn sự đóng góp ý kiến của Quý thầy cô, các chuyên viên trong lãnh vực kiểm thử.

Trong quá trình biên soạn, cuốn sách này không thể tránh khỏi những thiếu sót. Kính mong nhận được sự góp ý của quý độc giả.

Xin gửi thư về địa chỉ email:

stkbbook@yahoo.com.vn hay nhasachstkb@yahoo.com.vn

**CHƯƠNG 1**

# TỔNG QUAN VỀ PHẦN MỀM

## 1. ĐỊNH NGHĨA PHẦN MỀM

Trước khi tìm hiểu kiểm thử phần mềm ta cần tìm hiểu phần mềm là gì, đặc điểm, phân loại, cùng nhiều thuật ngữ liên quan. Từ đó mới hiểu rõ kiểm thử phần mềm là gì, tại sao cần tìm hiểu về nó, kiểm thử phần mềm đóng vai trò quan trọng như thế nào trong công nghệ thông tin...

Hiểu theo cách thông thường của một người không được đào tạo trong lãnh vực công nghệ thông tin thì: phần mềm là một chương trình được cài đặt trên máy tính nhằm thực hiện một nhiệm vụ tương đối độc lập, phục vụ cho một ứng dụng cụ thể như quản lý hoạt động của máy tính hoặc áp dụng máy tính trong các hoạt động kinh tế, quốc phòng, văn hóa, giáo dục, giải trí... Việc tạo ra một sản phẩm phần mềm phải trải qua nhiều giai đoạn, người ta gọi đó là quy trình phát triển phần mềm, bắt đầu từ khi có ý tưởng cho đến khi đưa ra sản phẩm phần mềm thực thi. Khối lượng công việc trong từng giai đoạn của quá trình sản xuất phần mềm cũng thay đổi theo thời gian.

Nếu tra trên Bách khoa toàn thư mở (Wikimedia) ta thấy phần mềm được giới thiệu như sau:

Phần mềm máy tính có tên tiếng Anh: Computer Software hay gọi tắt là Phần mềm (Software) là một tập hợp những câu lệnh hoặc chỉ thị (Instruction) được viết bằng một hoặc nhiều ngôn ngữ lập trình theo một trật tự xác định, với các dữ liệu hay tài liệu liên quan nhằm tự động thực hiện một số nhiệm vụ hay chức năng hoặc giải quyết một vấn đề cụ thể nào đó. Phần mềm thực hiện các chức năng của nó bằng cách gửi các chỉ thị trực tiếp đến phần cứng máy tính (Computer Hardware) hoặc bằng cách cung cấp dữ liệu để phục vụ các chương trình hay phần mềm khác. Phần mềm là một khái niệm trừu tượng, nó khác với phần cứng ở chỗ là "phần mềm không thể sờ hay đụng vào", và nó cần phải có phần cứng mới có thể thực thi được.

## 2. ĐẶC ĐIỂM PHẦN MỀM

Trước đây, để tạo ra chương trình máy tính người ta phải làm việc trực tiếp với các con số 0 hoặc 1 (sử dụng hệ số nhị phân), hay còn gọi là ngôn ngữ máy. Công việc này rất khó khăn, chiếm nhiều thời gian, công sức và đặc biệt dễ gây ra lỗi. Để khắc phục nhược điểm này, người ta đã xuất ra hợp ngữ, một ngôn ngữ cho phép thay thế dãy 0 hoặc 1 này bởi các từ gợi nhớ tiếng Anh.

Tuy nhiên, cải tiến này vẫn còn chưa thích hợp với đa số người dùng máy tính, những người luôn mong muốn các lệnh chính là ý nghĩa của các thao tác mà nó mô tả. Vì vậy, ngay từ những năm 1950, người ta đã xây dựng những ngôn ngữ lập trình mà câu lệnh của nó gần với ngôn ngữ tự nhiên. Các ngôn ngữ này được gọi là ngôn ngữ lập trình bậc cao. Chương trình máy tính thường được tạo ra bởi con người, những người này được gọi là lập trình viên, tuy nhiên cũng tồn tại những chương trình được sinh ra bởi các chương trình khác.

#### Đặc tính chung của phần mềm:

- Là hàng hóa vô hình, không nhìn thấy được.
- Chất lượng phần mềm: không mòn đi mà có xu hướng tốt lên sau mỗi lần có lỗi (error/bug) được phát hiện và sửa.
- Phần mềm vốn chứa lỗi tiềm tàng, theo quy mô càng lớn thì khả năng chứa lỗi càng cao.
- Lỗi phần mềm dễ được phát hiện bởi người ngoài.
- Chức năng của phần mềm thường biến hóa, thay đổi theo thời gian (theo nơi sử dụng).
- Hiệu ứng l่าน sóng trong thay đổi phần mềm.
- Phần mềm vốn chứa ý tưởng và sáng tạo của tác giả/nhóm làm ra nó.
- Cần khả năng “tư duy nhị phân” trong xây dựng, phát triển phần mềm.
- Có thể sao chép rất đơn giản.

Một phần mềm được gọi là tốt nếu có các chỉ tiêu cơ bản sau:

- Phản ánh đúng yêu cầu người dùng (tính hiệu quả-effectiveness).
- Chứa ít lỗi tiềm tàng, dễ vận hành, sử dụng.
- Giá thành không vượt quá giá ước lượng ban đầu.
- Có tính an toàn và có độ tin cậy cao.
- Hiệu suất xử lý cao
  - Hiệu suất thời gian tốt (Efficiency Time).
  - Độ phức tạp tính toán thấp (Complexity).
  - Thời gian quay vòng ngắn (Turn Around Time: TAT).
  - Thời gian hồi đáp nhanh (Response time).
  - Sử dụng tài nguyên hữu hiệu: CPU, RAM, HDD, Internet resources...
- Tính dễ hiểu: là chỉ tiêu ngày càng quan trọng của phần mềm
  - Kiến trúc và cấu trúc thiết kế dễ hiểu.
  - Dễ kiểm tra, kiểm thử, kiểm chứng.

- Dễ bảo trì.
- Có tài liệu (mô tả yêu cầu, điều kiện kiểm thử, vận hành, bảo trì, FAQ ...) với chất lượng cao.

### 3. PHÂN LOẠI

Với định nghĩa và các đặc điểm phần mềm như trên, người ta có thể phân loại phần mềm theo những cách sau:

#### Theo phương thức hoạt động

1. **Phần mềm hệ thống:** dùng để vận hành máy tính và các phần cứng máy tính, nó bao gồm các hệ điều hành, phần mềm điều vận thiết bị (device driver), các công cụ phân tích (diagnostic tool), trình phục vụ, hệ thống cửa sổ, các tiện ích... Mục đích của phần mềm hệ thống là để giúp các lập trình viên ứng dụng không phải quan tâm đến các chi tiết của hệ thống máy tính phức tạp được sử dụng, đặc biệt là các tính năng bộ nhớ và các phần cứng khác chẳng hạn như máy in, bàn phím, thiết bị hiển thị. Ví dụ: các hệ điều hành máy tính Windows, Linux, Unix, các thư viện động-Dynamic Linked Library (còn gọi là thư viện liên kết động-DLL) của hệ điều hành, các trình điều khiển (driver), phần sụn (firmware) và BIOS. Đây là các loại phần mềm mà hệ điều hành liên lạc với chúng để điều khiển và quản lý các thiết bị phần cứng.
2. **Phần mềm lập trình:** cung cấp các công cụ hỗ trợ lập trình viên trong khi viết chương trình bằng các ngôn ngữ lập trình khác nhau. Các công cụ này bao gồm các trình soạn thảo, trình biên dịch, trình thông dịch, trình liên kết, trình tìm lỗi v.v... Một môi trường phát triển tích hợp (IDE) kết hợp các công cụ này thành một gói phần mềm, và một lập trình viên có thể không cần gõ nhiều dòng lệnh để dịch, tìm lỗi...
3. **Phần mềm chuyển đổi mã:** bao gồm trình biên dịch và trình thông dịch: các loại chương trình này sẽ đọc các câu lệnh từ mã nguồn được viết bởi các lập trình viên theo một ngôn ngữ lập trình và dịch nó sang dạng ngôn ngữ máy mà máy tính có thể hiểu được, hay dịch nó sang một dạng khác như là tập tin đối tượng (object file) và các tập tin thư viện (library file) mà các phần mềm khác (như hệ điều hành chẳng hạn) có thể hiểu để vận hành máy tính thực thi các lệnh.
4. **Phần mềm ứng dụng:** là một loại chương trình có khả năng làm cho máy tính thực hiện trực tiếp một công việc nào đó người dùng muốn thực hiện. Điều này khác với phần mềm hệ thống tích hợp các chức năng của máy tính, nhưng có thể không trực tiếp thực hiện một tác vụ nào có ích cho người dùng. Thí dụ tiêu biểu cho phần mềm ứng dụng là chương trình xử lý văn bản, bảng tính, chương trình giải trí. Các phần mềm ứng dụng thường được gom lại thành bộ phần mềm.

Microsoft Office và OpenOffice là những bộ phần mềm gồm có chương trình xử lý văn bản, bảng tính, vẽ và các phần mềm khác. Phần mềm doanh nghiệp, phần mềm quản lý nguồn nhân lực, phần mềm giáo dục, cơ sở dữ liệu, phần mềm trò chơi, chương trình tiện ích, hay các loại phần mềm đặc hại cũng là những phần mềm ứng dụng. Các phần mềm riêng biệt trong bộ phần mềm thường có giao diện và tính năng tương tự làm người dùng dễ dàng học và sử dụng. Và các phần mềm thường tương tác được với nhau để đem lại lợi ích cho người dùng. Thí dụ, phần mềm bảng tính có thể nhúng một phần văn bản vào.

Cần lưu ý là: không có sự phân biệt rõ ràng giữa phần mềm hệ thống và phần mềm ứng dụng như Bộ Tư pháp Mỹ và Microsoft tranh cãi Internet Explorer có phải là một phần của Windows hay không. Cũng như trong một số hệ thống nhúng, người dùng không biết được phần mềm ứng dụng trong hệ thống, như các phần mềm điều khiển DVD, VCD, máy giặt hay lò vi sóng.

**5. Phần mềm quản lý:** là phần mềm ứng dụng với nhiệm vụ thực hiện tin học hóa các quá trình quản lý truyền thống, không chỉ đơn thuần là việc lưu trữ hay xử lý thông tin. Việc xây dựng và khai thác phần mềm quản lý đòi hỏi sự am hiểu về chuyên môn quản lý tương ứng, thí dụ quản lý con người, quản lý kho hàng, quản lý lương, v.v... Ngày nay, các phần mềm quản lý có xu hướng trực tuyến nhiều hơn nhờ công nghệ trên nền Internet phát triển mạnh hơn trước đây rất nhiều.

Một số chủng loại phần mềm quản lý tiêu biểu:

- Phần mềm kế toán.
- Phần mềm quản lý nhân sự, tiền lương, quản lý quan hệ khách hàng.
- Phần mềm quản lý thương mại.
- Quản lý thi trắc nghiệm.
- Quản lý phòng game, net, quản lý tài sản.
- Các giải pháp ERP (Quản lý tổng thể doanh nghiệp).

**6. Các nền tảng công nghệ:** là phần mềm xây dựng các chuẩn làm nền tảng trong công nghiệp như truyền thông, kết nối mạng v.v.

Còn có thể phân loại phần mềm theo nhóm: phần mềm thời gian thực (Real-time Software), phần mềm tính toán Khoa học và Kỹ thuật (Eng.& Scienc Software), phần mềm nhúng (Embedded Software), phần mềm máy tính cá nhân (Personal computer Software), phần mềm trên Web (Web-based Software), phần mềm trí tuệ nhân tạo (AI Software) v.v không phải là chủ đề chính nên không trình bày trong phần này.

### Theo khả năng ứng dụng

1. Những phần mềm không phụ thuộc, nó có thể được bán cho bất kỳ khách hàng nào trên thị trường tự do. Ví dụ: phần mềm cơ sở dữ liệu như Oracle, đồ họa như Photoshop, Corel Draw, soạn thảo và xử lý văn bản, bảng tính như Microsoft Office...

**Ưu điểm:** thường đây là những phần mềm có khả năng ứng dụng rộng rãi cho nhiều nhóm người sử dụng.

**Khuyết điểm:** thiếu tính uyển chuyển, tùy biến.

2. Những phần mềm được viết theo đơn đặt hàng hay hợp đồng của một khách hàng cụ thể nào đó (một công ty, bệnh viện, trường học,...). Ví dụ: phần mềm điều khiển, phần mềm hỗ trợ bán hàng...

**Ưu điểm:** có tính uyển chuyển, tùy biến cao để đáp ứng được nhu cầu của một nhóm người sử dụng nào đó.

**Khuyết điểm:** thông thường đây là những phần mềm ứng dụng chuyên ngành hẹp.

## 4. QUÁ TRÌNH TẠO PHẦN MỀM

### Về mặt thiết kế

Tùy theo mức độ phức tạp của phần mềm làm ra, người thiết kế phần mềm sẽ ít nhiều dùng đến các phương tiện để tạo ra mẫu thiết kế theo ý muốn (chẳng hạn như là các sơ đồ khối, các lưu đồ, các thuật toán và các mã giả), sau đó mẫu này được mã hóa bằng các ngôn ngữ lập trình và được các trình dịch chuyển thành các khối lệnh (module) hay/và các files khả thi. Tập hợp các files khả thi và các khối lệnh đó làm thành một phần mềm. Thường khi một phần mềm được tạo thành, để cho hoàn hảo thì phần mềm đó phải được điều chỉnh hay sửa chữa từ khâu thiết kế cho đến khâu tạo thành phiên bản phần mềm một số lần. Một phần mềm thông thường sẽ tương thích với một hay vài hệ điều hành, tùy theo cách thiết kế, cách viết mã nguồn và ngôn ngữ lập trình được dùng.

### Sản xuất và phát triển

Việc phát triển và đưa ra thị trường một phần mềm là đối tượng nghiên cứu của bộ môn kỹ nghệ phần mềm hay còn gọi là công nghệ phần mềm (software engineering). Bộ môn này nghiên cứu các phương pháp tổ chức, cách thức sử dụng nguồn tài nguyên, vòng quy trình sản xuất, cùng với các mối liên hệ với thị trường, cũng như liên hệ giữa các yếu tố này với nhau. Tối ưu hoá quá trình sản xuất phần mềm cũng là đối tượng được cứu xét của bộ môn. Đây là sự áp dụng một cách tiếp cận có hệ thống, và định lượng được cho việc phát triển, sử dụng và bảo trì phần mềm.

Ngành học Kỹ nghệ phần mềm bao trùm kiến thức, các công cụ, và các phương pháp cho việc định nghĩa yêu cầu phần mềm, và thực hiện các tác vụ thiết kế, xây dựng, kiểm thử (software testing), và bảo trì phần mềm. Kỹ nghệ phần mềm còn sử dụng kiến thức của các lĩnh vực như kỹ thuật máy tính, khoa học máy tính, quản lý, toán học, quản lý dự án, quản lý chất lượng... Tác giả Edsger Dijkstra khi giới thiệu về công nghệ phần mềm đã trình bày như sau:

*Khi máy tính chưa xuất hiện, thì việc lập trình chưa có khó khăn gì cả. Khi mới xuất hiện một vài chiếc máy tính chức năng kém thì việc lập trình bắt đầu gặp một vài khó khăn nhỏ nhõ. Giờ đây khi chúng ta có những chiếc máy tính khổng lồ thì những khó khăn ấy trở nên vô cùng lớn. Như vậy ngành công nghiệp điện tử không giải quyết khó khăn nào cả mà họ chỉ tạo thêm ra những khó khăn mới. Khó khăn mà họ tạo nên chính là việc sử dụng sản phẩm của họ.*

Công nghệ phần mềm có một lịch sử khá sớm, các công cụ được dùng cũng như các ứng dụng được viết đã tham gia vào kỹ nghệ phần mềm theo thời gian.

### Dòng thời gian

- Thập niên 1940: các chương trình cho máy tính được viết bằng tay.
  - Thập niên 1950: các công cụ đầu tiên xuất hiện như là phần mềm biên dịch Macro Assembler và phần mềm thông dịch đã được tạo ra và sử dụng rộng rãi để nâng cao năng suất và chất lượng. Các trình dịch được tối ưu hóa lần đầu tiên ra đời.
  - Thập niên 1960: các công cụ của thế hệ thứ hai như các trình dịch tối ưu hóa và công việc kiểm tra mẫu đã được dùng để nâng cao sản phẩm và chất lượng. Khái niệm công nghệ phần mềm đã được bàn thảo rộng rãi.
  - Thập niên 1970: các công cụ phần mềm, chẳng hạn như trong UNIX các vùng chứa mã, lệnh thực hiện v.v... được kết hợp với nhau. Số lượng doanh nghiệp nhỏ về phần mềm và số lượng máy tính cỡ nhỏ tăng nhanh.
  - Thập niên 1980: các PC và máy trạm ra đời. Cùng lúc có sự xuất hiện của mô hình dự toán khả năng, lượng phần mềm tiêu thụ tăng mạnh.
  - Thập niên 1990: phương pháp lập trình hướng đối tượng ra đời. Các quá trình nhanh như là lập trình cực hạn được chấp nhận rộng rãi. Trong thập niên này, các thiết bị máy tính cầm tay phổ biến rộng rãi.
  - Hiện nay: các phần mềm biên dịch và quản lý như là PHP, Java, làm cho việc thiết kế, viết phần mềm ứng dụng trở nên dễ dàng hơn nhiều.
- Bảng trang bên cho thấy tỉ lệ công việc của các giai đoạn phát triển phần mềm.

## TỶ LỆ CÔNG VIỆC CỦA CÁC GIAI ĐOẠN PHÁT TRIỂN PHẦN MỀM

Giai đoạn	Phản tích yêu cầu	Thiết kế sơ bộ	Thiết kế chi tiết	Lập trình và kiểm thử đơn vị	Tích hợp và kiểm thử tích hợp	Kiểm thử hệ thống
Hai thập kỷ 1960-1970			10%	80%	10%	
thập kỷ 1980		20%		60%	20%	
thập kỷ 1990	40%		30%		30%	

**Chi phí liên quan từng giai đoạn của vòng đời phần mềm như sau:**

CÁC GIAI ĐOẠN PHÁT TRIỂN	GIAI ĐOẠN SẢN PHẨM
Phân tích yêu cầu 3%	Vận hành bảo trì 67%
Đặc tả 3%	
Thiết kế 5%	
Lập trình 7%	
Kiểm thử 5%	

Như vậy, một sản phẩm phần mềm không chỉ đơn giản là các đoạn mã chương trình mà còn rất nhiều phần ẩn dật sau nó. Vì vậy, việc mắc lỗi không chỉ xảy ra trong khi lập trình mà còn xảy ra cao hơn trong các công đoạn khác của quá trình phát triển một sản phẩm phần mềm. Để hiểu rõ hơn chỗ đứng của kiểm thử phần mềm ta sẽ tìm hiểu các ngành con của kỹ nghệ phần mềm.

Kỹ nghệ phần mềm có thể được chia thành 10 ngành con, đó là:

1. **Yêu cầu phần mềm:** phân tích, đặc tả và phê chuẩn các yêu cầu đối với phần mềm.
2. **Thiết kế phần mềm:** việc thiết kế phần mềm thường được hoàn thành bằng các công cụ Computer-Aided Software Engineering (CASE) và sử dụng các tiêu chuẩn định dạng, như Unified Modeling Language (UML).
3. **Phát triển phần mềm:** xây dựng phần mềm thông qua việc dùng các ngôn ngữ lập trình.
4. **Kiểm thử phần mềm:** tìm các lỗi hay khuyết điểm phần mềm nhằm đảm bảo hiệu quả hoạt động của phần mềm trong nhiều ngành khác nhau.
5. **Bảo trì phần mềm:** Các hệ thống phần mềm thường có nhiều vấn đề và cần được cải tiến trong một thời gian dài sau khi đã được hoàn tất vào lần đầu tiên. Lĩnh vực con này xem xét các vấn đề đó.
6. **Quản lý cấu hình phần mềm:** vì các hệ thống phần mềm rất phức tạp, cấu hình của chúng (ví dụ như kiểm soát phiên bản và mã nguồn) phải được quản lý bằng các phương pháp chuẩn và có cấu trúc.
7. **Quản lý kỹ nghệ phần mềm:** quản lý hệ thống phần mềm vay mượn rất nhiều khái niệm từ quản lý dự án, nhưng có nhiều khía cạnh riêng biệt nhỏ gặp trong phần mềm mà không gặp trong các ngành quản lý khác.
8. **Quy trình phát triển phần mềm:** quy trình xây dựng phần mềm là điều tranh cãi giữa các nhà thực hành; một số quy trình nổi tiếng là Mô hình Thác nước, Mô hình Xoắn ốc, Phát triển Tăng tiến và Lặp, và Phát triển Linh hoạt.

9. **Các công cụ kỹ thuật phần mềm:** sử dụng các chương trình Computer Aided Software Engineering để hỗ trợ thiết kế.
10. **Chất lượng phần mềm:** đưa ra các giải pháp để nâng cao chất lượng phần mềm.

### **Các ngành liên quan**

Kỹ nghệ phần mềm liên quan đến các ngành khoa học máy tính, khoa học quản lý, và kỹ nghệ hệ thống.

**Khoa học máy tính:** kỹ nghệ phần mềm đã từng được nhiều nhà khoa học coi là một lĩnh vực con của khoa học máy tính. Nhiều nền tảng của kỹ nghệ phần mềm đến từ khoa học máy tính.

**Quản lý dự án:** việc xây dựng một hệ thống phần mềm thường được coi là một dự án và việc quản lý nó vay mượn nhiều nguyên lý từ lĩnh vực quản lý dự án.

**Kỹ nghệ hệ thống:** các kỹ sư hệ thống đã xem xét độ phức tạp của các hệ thống lớn trong nhiều thập kỷ, và những kiến thức của họ được áp dụng cho nhiều vấn đề trong kỹ nghệ phần mềm.

**Các sản phẩm phần mềm:** đối tượng chính của công nghệ phần mềm là sản xuất ra các sản phẩm phần mềm. Sản phẩm phần mềm là các phần mềm được phân phối cho khách hàng cùng với các tài liệu mô tả phương thức cài đặt và cách thức sử dụng chúng.

### **Phân loại**

1. **Sản phẩm tổng quát:** là các phần mềm riêng, được sản xuất bởi một tổ chức phát triển và bán ra thị trường cho bất kỳ khách hàng nào có khả năng tiêu thụ.
2. **Sản phẩm chuyên ngành:** là phần mềm được hỗ trợ tài chính bởi khách hàng trong chuyên ngành. Phần mềm được phát triển một cách đặc biệt cho khách hàng qua các hợp đồng. Cho đến thập niên 1980 hầu hết sản phẩm phần mềm đều làm theo đơn đặt hàng riêng (đặc biệt hoá). Nhưng kể từ khi có máy PC, tình hình hoàn toàn thay đổi. Các phần mềm được phát triển và bán cho hàng trăm ngàn khách hàng là chủ các PC và do đó giá bán các sản phẩm này cũng rẻ hơn nhiều. Microsoft là nhà sản xuất phần mềm lớn nhất hiện nay.

### **Thuộc tính của sản phẩm phần mềm**

Thuộc tính của một sản phẩm phần mềm là các đặc tính xuất hiện từ sản phẩm một khi nó được cài đặt và được đưa ra dùng. Các thuộc tính này không bao gồm các dịch vụ được cung cấp kèm theo sản phẩm đó. Ví dụ: mức hiệu quả, độ bền, khả năng bảo trì, khả năng dùng ở nhiều hệ điều hành là các thuộc tính.

Các thuộc tính biến đổi tùy theo phần mềm. Tuy nhiên những thuộc tính tối quan trọng bao gồm:

- Khả năng bảo trì:** nó có khả năng thỏa mãn yêu cầu của khách hàng (cập nhật, phát triển v.v...).
- Khả năng tin cậy:** khả năng tin cậy của phần mềm bao gồm một loạt các đặc tính như là độ tin cậy, an toàn, và bảo mật. Phần mềm tin cậy không thể tạo ra các thiệt hại vật chất hay kinh tế trong trường hợp hư hỏng.
- Độ hữu hiệu:** phần mềm không thể phạm các nguồn tài nguyên như là bộ nhớ và các chu kỳ vi xử lý.
- Khả năng sử dụng:** phần mềm nên có một giao diện tương đối dễ cho người dùng và có đầy đủ các hồ sơ về phần mềm.

### Thiết kế phần mềm

Thiết kế phần mềm (Software design) là một quá trình giải quyết vấn đề và lập kế hoạch cho một giải pháp phần mềm. Sau khi các mục đích và đặc điểm kỹ thuật của phần mềm được quyết định, lập trình viên sẽ thiết kế hoặc thuê người thiết kế để phát triển một kế hoạch cho giải pháp phần mềm. Nó bao gồm các thành phần cấp thấp, các vấn đề thuật toán cũng như một khung nhìn kiến trúc.

### Phân tích yêu cầu trong quy trình phát triển phần mềm.

Trong các ngành kỹ thuật hệ thống và kỹ nghệ phần mềm, phân tích yêu cầu là công việc bao gồm các tác vụ xác định các yêu cầu cho một hệ thống mới hoặc được thay đổi, dựa trên cơ sở là các yêu cầu (có thể mâu thuẫn) mà những người có vai trò quan trọng đối với hệ thống, chẳng hạn người sử dụng, đưa ra. Việc phân tích yêu cầu có ý nghĩa quan trọng đối với thành công của một dự án.

Việc phân tích yêu cầu một cách có hệ thống còn được gọi là kỹ nghệ yêu cầu (*requirements engineering*). *Đôi khi nó còn được gọi* một cách không thật chính xác bằng những cái tên như thu thập yêu cầu (*requirements gathering, requirements capture*), hoặc đặc tả yêu cầu (*requirements specification*). Thuật ngữ "*phân tích yêu cầu*" còn được áp dụng cụ thể cho công việc thuần túy phân tích (thay vì các việc khác chẳng hạn như làm rõ yêu cầu hay viết tài liệu yêu cầu).

Các yêu cầu phải có tính do được, kiểm thử được, có liên quan đến các nhu cầu hoặc cơ hội doanh nghiệp đã được xác định, và các yêu cầu phải được định nghĩa ở một mức độ chi tiết đủ cho việc thiết kế hệ thống.

### Các kỹ thuật chính

Về khái niệm, việc phân tích yêu cầu bao gồm ba loại hoạt động sau:

- **Làm rõ yêu cầu (Eliciting requirements):** giao tiếp với khách hàng và người sử dụng để xác định các yêu cầu của họ.
- **Xem xét yêu cầu (Analyzing requirements):** xác định xem các yêu cầu được đặt ra có ở tình trạng không rõ ràng, không hoàn chỉnh, đa nghĩa, hoặc mâu thuẫn hay không, và giải quyết các vấn đề đó.
- **Làm tài liệu yêu cầu (Recording requirements):** các yêu cầu có thể được ghi lại theo nhiều hình thức, chẳng hạn các tài liệu ngôn ngữ tự nhiên, các tình huống sử dụng (use case), các diễn tiến với người sử dụng hoặc các đặc tả tiến trình.

Việc phân tích yêu cầu có thể là một quá trình dài và khó khăn, cần đến nhiều kỹ năng tâm lý khéo léo. Các hệ thống mới làm thay đổi môi trường và các mối quan hệ giữa con người, do đó điều quan trọng là phải xác định được tất cả những người có vai trò quan trọng, xem xét tất cả các nhu cầu của họ và đảm bảo rằng: họ hiểu được các hàm ý của hệ thống mới.

Các nhà phân tích có thể sử dụng một số kỹ thuật để làm rõ các yêu cầu của khách hàng. Trong lịch sử, các kỹ thuật này bao gồm các cuộc phỏng vấn, thành lập các nhóm trọng tâm với các cuộc họp bàn về yêu cầu và tạo ra các danh sách yêu cầu. Các kỹ thuật hiện đại hơn gồm có tạo nguyên mẫu (prototyping), và tình huống sử dụng. Khi cần thiết, nhà phân tích sẽ kết hợp các phương pháp này để thiết lập các yêu cầu chính xác từ những người có vai trò quan trọng, nhằm mục đích xây dựng một hệ thống thỏa mãn các yêu cầu doanh nghiệp.

### **Các vấn đề về người dùng và khách hàng**

Trong cuốn Rapid Development, tác giả Steve McConnell đã liệt kê một loạt các khả năng người dùng có thể cản trở quá trình thu thập yêu cầu:

- Người dùng không hiểu họ muốn gì.
- Người dùng không tuân theo một bộ yêu cầu đã được tài liệu hóa.
- Người dùng nhất định đòi hỏi các yêu cầu mới sau khi chi phí và kế hoạch phát triển đã được hoạch định xong.
- Mức độ giao tiếp với người dùng là thấp.
- Người dùng thường không tham gia các đợt thẩm định hoặc không thể tham gia.
- Người dùng không hiểu kỹ thuật.
- Người dùng không hiểu quy trình phát triển.

Những điều này có thể dẫn tới tình huống: yêu cầu người dùng liên tục thay đổi ngay cả khi việc phát triển hệ thống hay sản phẩm đã được bắt đầu.

### Vấn đề về kỹ sư/nhà phát triển

Trong quá trình phân tích yêu cầu, các vấn đề sau có thể này sinh từ phía các kỹ sư và nhà phát triển:

- Nhân viên kỹ thuật và người dùng cuối có thể có ngôn từ khác nhau. Kết quả là họ có thể tin rằng họ hoàn toàn đồng thuận cho đến khi sản phẩm hoàn thiện được đưa ra.
- Các kỹ sư và nhà phát triển có thể cố lái cho các yêu cầu khớp với một hệ thống hay mô hình sẵn có, thay vì phát triển một hệ thống theo sát nhu cầu của khách hàng.
- Việc phân tích có thể do các kỹ sư hoặc lập trình viên thực hiện, thay vì các nhân viên có kỹ năng và kiến thức miễn ứng dụng để có thể hiểu các nhu cầu của khách hàng một cách đúng đắn.

### Giải pháp đã được thực hiện

Một giải pháp đối với các vấn đề về giao tiếp là thuê các chuyên gia về doanh nghiệp hoặc chuyên gia phân tích hệ thống. Các kỹ thuật được đưa ra trong thập kỷ 1990 như tạo nguyên mẫu, UML, tình huống sử dụng và phát triển phần mềm linh hoạt (*Agile software development*) cũng đã được dùng làm giải pháp cho các vấn đề trên.

Như đã trình bày ở trên, ngành con thứ 4 trong kỹ nghệ phần mềm là kiểm thử phần mềm, chúng ta sẽ tìm hiểu sơ bộ ngành con này trước khi tìm hiểu sâu về nó.

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn sản phẩm hay dịch vụ phần mềm trong đúng môi trường chúng dự định sẽ được triển khai nhằm cung cấp cho người có lợi ích liên quan những thông tin về chất lượng của sản phẩm hay dịch vụ phần mềm ấy. Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm trong nhiều ngành khác nhau.

#### Các phương pháp kiểm thử

1. Phương pháp kiểm thử hộp đen.
2. Phương pháp kiểm thử hộp trắng.
3. Phương pháp kiểm thử hộp không trắng.

#### Các cấp độ kiểm thử

1. Kiểm thử đơn vị.
2. Kiểm thử tích hợp.
3. Kiểm thử hệ thống.
4. Kiểm thử tích hợp hệ thống.

**Các phương pháp kiểm thử phi chức năng**

1. Kiểm thử hoạt động và hiệu suất chương trình.
2. Kiểm thử độ ổn định.
3. Kiểm thử tính sử dụng.
4. Kiểm thử khả năng bảo mật.
5. Tính địa phương và toàn cầu.
6. Kiểm thử khả năng chịu lỗi.

**Quy trình phát triển phần mềm tương quan để sản xuất ra một sản phẩm phần mềm.**

Hầu hết các thao tác này được tiến hành bởi các kỹ sư phần mềm. Các công cụ hỗ trợ máy tính về kỹ thuật phần mềm có thể được dùng để giúp trong một số thao tác.

Có 4 thao tác là nền tảng của hầu hết các quy trình phần mềm là:

1. **Đặc tả phần mềm:** các chức năng của phần mềm và điều kiện để nó hoạt động phải được định nghĩa.
2. **Sự phát triển phần mềm:** để phần mềm đạt được đặc tả thì phải có quy trình phát triển này.
3. **Đánh giá phần mềm:** phần mềm phải được đánh giá để chắc chắn rằng nó làm những gì mà khách hàng muốn.
4. **Sự tiến hóa của phần mềm:** phần mềm phải tiến hóa để thỏa mãn sự thay đổi các yêu cầu của khách hàng.

**Các mô hình phát triển sản phẩm phần mềm****Mô hình thác nước**

Mô hình này làm cho ý nghĩa việc sản xuất phần mềm được thấy rõ hơn.

1. **Phân tích các yêu cầu và định nghĩa:** hệ thống dịch vụ, khó khăn và mục tiêu được hình thành bởi sự trợ giúp của hệ thống người tiêu dùng. Sau đó các yếu tố này được định nghĩa sao cho có thể hiểu được bởi cả người phát triển và người tiêu dùng.
2. **Thiết kế phần mềm và hệ thống:** thiết kế hệ thống các quy trình, các bộ phận và các yêu cầu về cả phần mềm lẫn phần cứng. Hoàn tất hầu như tất cả kiến trúc của các hệ thống này. Thiết kế phần mềm tham gia vào việc biểu thị các chức năng hệ thống phần mềm có thể được chuyển dạng thành một hay nhiều chương trình khả thi.
3. **Thực hiện và thử nghiệm các đơn vị:** trong giai đoạn này, thiết kế phần mềm phải được chứng thực như là một tập hợp nhiều chương trình hay nhiều đơn vị nhỏ. Thử nghiệm các đơn vị bao gồm xác minh rằng mỗi đơn vị đáp ứng đặc tả của nó.

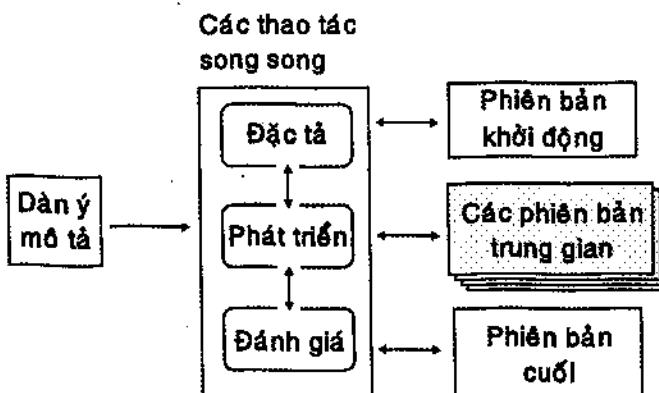
4. Tổng hợp và thử nghiệm toàn bộ: các đơn vị chương trình riêng lẻ hay các chương trình được tích hợp lại và thử nghiệm như là một hệ thống hoàn tất và chứng tỏ được các yêu cầu của phần mềm được thỏa mãn. Sau khi thử nghiệm, phần mềm được cung ứng cho người tiêu dùng.
5. Sản xuất và bảo trì: thông thường (nhưng không bắt buộc) đây là pha lâu nhất của chu kỳ sống (của sản phẩm). Phần mềm được cài đặt và được dùng trong thực tế. Bảo trì bao gồm việc điều chỉnh các lỗi chưa được phát hiện trong các giai đoạn trước của chu kỳ sống; nâng cấp sự thực hiện của hệ thống các đơn vị và nâng cao hệ thống dịch vụ cho các phát hiện về yêu cầu mới.

Chỗ yếu của mô hình này là nó không linh hoạt. Các bộ phận của đề án chia ra thành những phần riêng của các giai đoạn. Hệ thống phân phối đôi khi không dùng được vì không thỏa mãn được yêu cầu của khách hàng. Mặc dù vậy mô hình này phản ánh thực tế công nghệ. Như là một hệ quả, đây vẫn là mô hình cơ sở cho đa số các hệ thống phát triển phần mềm - phần cứng.

### Mô hình phát triển tiến hóa của phần mềm

- Phân loại sự phát triển tiến hóa

- > Lập trình thăm dò: bằng cách làm việc với khách hàng (đối tượng của quy trình) để thăm dò các yêu cầu và phân phối phần mềm dứt điểm. Sự phát triển nên bắt đầu với những phần nào đã được hiểu rõ. Phần mềm sẽ được thêm vào các chức năng mới khi nó được đề nghị cho khách hàng (và nhận về các thông tin).
- > Mẫu thăm dò: đối tượng của phát triển tiến hóa này nhằm hiểu các yêu cầu của khách hàng và phát triển các định nghĩa, yêu cầu tốt hơn cho phần mềm. Các mẫu tập trung trên các thí nghiệm với những phần đòi hỏi nào của khách hàng mà có thể gây sự khó hiểu hay ngộ nhận.



Mô hình phát triển phần mềm theo kiểu tiến hóa

### Mô hình phát triển phần mềm theo kiểu tiến hóa

- Phân tích mô hình: mô hình phát triển tiến hóa này hiệu quả hơn mô hình thác nước. Tuy nhiên, nó vẫn còn các khuyết điểm:
  1. Quy trình không nhìn thấy rõ được: các nhà quản lý cần phân phối thường xuyên để đo lường sự tiến bộ. Nó không kinh tế trong việc làm ra các hồ sơ cho phần mềm.
  2. Phần mềm thường có cấu trúc nghèo nàn: sự thay đổi liên tục dễ làm đổ vỡ cấu trúc của phần mềm, tạo ra sự khó khăn và tốn phí.
  3. Thường đòi hỏi những kỹ năng đặc biệt: hầu hết các hệ thống theo cách này được tiến hành bởi các nhóm nhỏ có kỹ năng cao cung như các cá nhân năng động.
- Mô hình này thích hợp với:
  1. Phát triển các loại phần mềm tương đối nhỏ.
  2. Phát triển các loại phần mềm có đời sống tương đối ngắn.
  3. Tiến hành trong các hệ thống lớn hơn ở những chỗ mà không thể biểu thị được các đặc tả chi tiết trong lúc tiến hành. Thí dụ của trường hợp này là các hệ thống thông minh (trí khôn) nhân tạo (AI) và các giao diện cho người dùng.

### Mô hình xoắn ốc Boehm

Đây là mô hình phát triển từ mô hình thác nước cho thấy mức độ tổng quát hơn của các pha sản xuất của một sản phẩm. Mô hình được đề nghị bởi Boehm vào năm 1988. Mô hình này có thể chỉ ra các rủi ro có thể hình thành trên căn bản của mô hình quy trình (sản xuất) tổng quát.

Mô hình Boehm có dạng xoắn ốc. Mỗi vòng lặp đại diện cho một pha của quy trình phần mềm. Vòng trong cùng tập trung về tính khả thi, vòng kế lo về định nghĩa các yêu cầu, kế đến là thiết kế... Không có một pha nào được xem là cố định trong vòng xoắn. Mỗi vòng có 4 phần tương ứng với một pha.

1. Cài đặt đối tượng: chỉ ra các đối tượng của pha trong đề án. Những khó khăn hay cưỡng bức của quy trình và của sản phẩm được xác định và được lên kế hoạch chi tiết. Xác định các yếu tố rủi ro của đề án. Các phương án thay thế tùy theo các rủi ro này có thể được dự trù.
2. Lượng định và giảm thiểu rủi ro: tiến hành phân tích mỗi yếu tố rủi ro đã xác định, các bước đặt ra để giảm thiểu rủi ro.
3. Phát triển và đánh giá: sau khi đánh giá các yếu tố rủi ro, một mô hình phát triển cho hệ thống được chọn.
4. Lên kế hoạch: đề án được xem xét và quyết định có nên hay không tiếp tục pha mới trong vòng lặp.

### Các quy trình linh hoạt

Là quy trình mà trong đó cấu trúc khởi động sẽ nhỏ nhưng linh động và lớn dần của các đề án phần mềm nhằm tìm ra các khó khăn trước khi nó trở thành vấn đề có thể dẫn tới những hủy hoại. Quy trình này nhấn mạnh sự gọn nhẹ và tập trung hơn là các phương pháp truyền thống. Các quy trình linh hoạt dùng các thông tin phản hồi thay vì dùng các kế hoạch, như là một cơ chế điều khiển chính. Các thông tin phản hồi có được từ các thử nghiệm và các phiên bản phát hành của phần mềm tham gia.

Các quy trình linh hoạt thường có hiệu quả hơn các phương pháp cũ, nó dùng ít thời gian lập trình để sản xuất ra nhiều chức năng hơn, chất lượng cao hơn, nhưng nó không cung cấp một khả năng kế hoạch lâu dài.

Một cách ngắn gọn các phương pháp này cung ứng hiệu quả cao nhất cho vốn đầu tư, nhưng lại không định rõ hiệu quả gì.

Lập trình cực hạn, gọi tắt là XP, là loại quy trình linh hoạt được biết đến nhiều nhất. Trong XP, các pha được xúc tiến trong các bước cực nhỏ (hay liên tục) nếu so với các quy trình kiểu cũ, gọi là các "toán" xử lý. Bước đầu tiên (với chủ định là không hoàn tất) cho đến các bước có thể chỉ tốn một ngày hay một tuần, thay vì phải tốn nhiều tháng như trong phương pháp thác nước.

Đầu tiên, một người viết các thử nghiệm tự động để cung cấp các mục tiêu cụ thể cho sự phát triển. Kế đến là giai đoạn viết mã (bởi một cặp lập trình viên); giai đoạn này hoàn tất khi mà các mã viết qua được tất cả các thử nghiệm và những người lập trình không tìm ra thêm được thử nghiệm cần thiết nào nữa.

Thiết kế và kiến trúc được điều chỉnh và nâng cao ngay sau giai đoạn viết mã này bởi người viết mã đã thực hiện trong giai đoạn trước. Hệ thống chưa hoàn tất nhưng hoạt động được này được khai thác hay được đem ra minh họa cho (một phần) người tiêu dùng mà trong số đó có người trong đội phát triển phần mềm. Thời điểm này những người thực nghiệm lại bắt đầu viết các thử nghiệm cho những phần quan trọng kế tiếp của hệ thống.

### Tầm nhìn của các quy trình

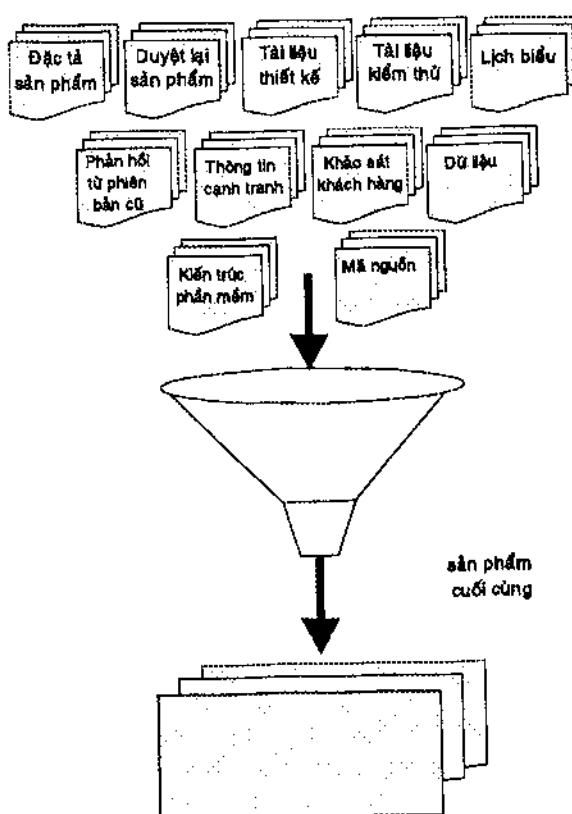
Do bản chất không thể nắm bắt cụ thể của các hệ thống phần mềm, các nhà quản lý quy trình phần mềm cần các báo cáo, các hồ sơ và xem xét để theo dõi tiến độ cũng như những gì xảy ra của công việc. Hầu hết các tổ chức phát triển các phần mềm lớn dùng kiểu quy trình "định hướng chuyển giao được". Mỗi thao tác phải kết thúc bằng một hồ sơ hay báo cáo nhằm làm cho quy trình phần mềm trở nên cụ thể hơn.

Với những khái niệm trình bày ở trên, việc kiểm thử phải được tiến hành trong tất cả các phần tạo nên một sản phẩm phần mềm để cho ra phần mềm hoàn chỉnh.

## Thế nào là lỗi phần mềm?

Có rất nhiều định nghĩa khác nhau về lỗi phần mềm, nhưng nhìn chung, có thể phát biểu một cách tổng quát như sau: "Lỗi phần mềm là sự không khớp giữa chương trình và những đặc tả của nó."

Dựa vào định nghĩa, chúng ta có thể thấy lỗi phần mềm xuất hiện theo ba dạng sau:



- **Sal:** sản phẩm được xây dựng khác với đặc tả.
- **Thiếu:** một yêu cầu đã được đặc tả nhưng lại không có trong sản phẩm được xây dựng.
- **Thừa:** một yêu cầu được đưa vào sản phẩm không có trong đặc tả.

Cũng có trường hợp yêu cầu này có thể là một thuộc tính sẽ được người dùng chấp nhận nhưng khác với đặc tả nên vẫn coi là có lỗi.

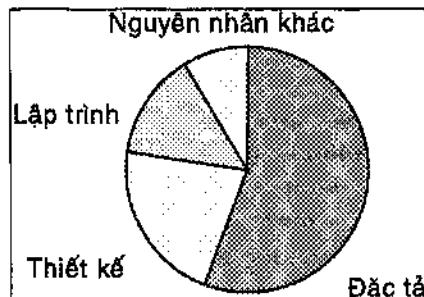
Một hình thức khác nữa cũng được xem là lỗi, đó là phần mềm khó hiểu, khó sử dụng, chậm hoặc dễ gây cảm nhận rằng phần mềm hoạt động không đúng.

## Tại sao lỗi phần mềm xuất hiện?

Khác với sự cảm nhận thông thường, lỗi xuất hiện nhiều nhất không phải do lập trình. Nhiều nghiên cứu đã được thực hiện trong các dự án từ rất nhỏ đến các dự án rất lớn và kết quả luôn giống nhau. Số lỗi do đặc tả gây ra là nhiều nhất, chiếm khoảng 80%. Có một số nguyên nhân làm cho đặc tả tạo ra nhiều lỗi nhất. Trong nhiều trường hợp, đặc tả không được viết ra.

Các nguyên nhân khác có thể do đặc tả không đủ cẩn thận, nó hay thay đổi, hoặc do chưa phối hợp tốt trong toàn nhóm phát triển. Sự thay đổi yêu cầu của khách hàng cũng là nguyên nhân dễ gây ra lỗi phần mềm. Khách hàng thay đổi yêu cầu không cần quan tâm đến những tác động sau khi thay đổi yêu cầu như phải thiết kế lại, lập lại kế hoạch, làm lại những việc đã hoàn thành.

Nếu có nhiều sự thay đổi, rất khó nhận biết hết được phần nào của dự án phụ thuộc và phần nào không phụ thuộc vào sự thay đổi. Nếu không giữ được vết thay đổi rất dễ phát sinh ra lỗi. Hình bên là biểu đồ cho thấy tỷ lệ nguyên nhân gây ra lỗi phần mềm



Nguồn gây ra lỗi lớn thứ hai là thiết kế. Đó là nền tảng mà lập trình viên dựa vào để nỗ lực thực hiện kế hoạch cho phần mềm.

Lỗi do lập trình gây ra cũng khá dễ hiểu. Ai cũng có thể mắc lỗi khi lập trình. Thời kỳ đầu, phát triển phần mềm có nghĩa là lập trình, công việc lập trình thì nặng nhọc, do đó lỗi do lập trình gây ra là chủ yếu.

Ngày nay, việc lập trình chỉ là một phần việc của quá trình phát triển phần mềm, cộng với sự hỗ trợ của nhiều công cụ lập trình cao cấp, việc lập trình trở nên nhẹ nhàng hơn, mặc dù độ phức tạp phần mềm lớn hơn rất nhiều. Do đó, lỗi do lập trình gây ra cũng ít hơn. Tuy nhiên, nguyên nhân để lập trình tạo ra lỗi lại nhiều hơn. Đó là do độ phức tạp của phần mềm, do tài liệu nghèo nàn, do sức ép thời gian hoặc chỉ đơn giản là những lỗi "không nói lên được". Một điều cũng hiển nhiên là nhiều lỗi xuất hiện trên bề mặt lập trình nhưng thực ra lại do lỗi của đặc tả hoặc thiết kế.

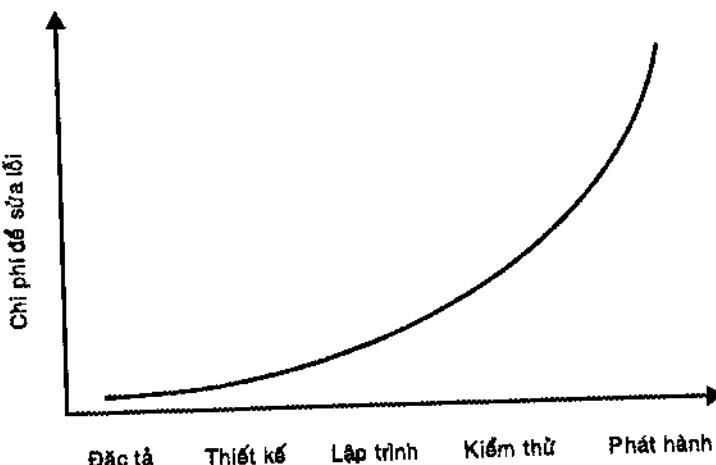
Một nguyên nhân khác tạo ra lỗi là do bản thân các công cụ phát triển phần mềm cũng có lỗi như công cụ trực quan, thư viện lớp, bộ biên dịch...

### **Chi phí cho việc sửa lỗi**

Theo tài liệu trích dẫn của Martin và McCable, bảo trì là phần chi phí chính của phần mềm và kiểm thử là hoạt động chi phí đắt thứ hai, ước tính khoảng 40% chi phí trong quá trình phát triển ban đầu của sản phẩm phần mềm. Kiểm thử cũng là phần chi phí chính của giai đoạn bảo trì do phải tiến hành kiểm thử lại những thay đổi trong quá trình sửa lỗi và đáp ứng yêu cầu người dùng.

Kiểm thử và sửa lỗi có thể được thực hiện tại bất kỳ giai đoạn nào của vòng đời phần mềm. Tuy nhiên chi phí cho việc tìm và sửa lỗi tăng một cách đáng kể trong quá trình phát triển.

Trong tài liệu Boehm, có trích dẫn kết quả nghiên cứu của IBM, GTE và TRW, tổng kết rằng lỗi được phát hiện càng muộn thì chi phí cho việc sửa lỗi càng lớn. Chi phí tăng theo hàm mũ như hình sau.



### Kiểm thử phần mềm

Kiểm thử phần mềm thường đồng nghĩa với việc tìm ra lỗi chưa được phát hiện. Tuy nhiên, có nhiều bối cảnh kiểm thử không bộc lộ ra lỗi. Kiểm thử phần mềm là quá trình thực thi một hệ thống phần mềm để xác định xem phần mềm đó có đúng với đặc tả không và thực hiện trong môi trường như mong đợi hay không. Mục đích của kiểm thử phần mềm là tìm ra lỗi chưa được phát hiện, tìm một cách sớm nhất và đảm bảo rằng lỗi đã được sửa, mà kiểm thử phần mềm không làm công việc chẩn đoán nguyên nhân gây ra lỗi đã được phát hiện và sửa lỗi.

Mục tiêu của kiểm thử phần mềm là thiết kế tài liệu kiểm thử một cách có hệ thống và thực hiện nó sao cho có hiệu quả, nhưng tiết kiệm được thời gian, công sức và chi phí.

### Chất lượng phần mềm

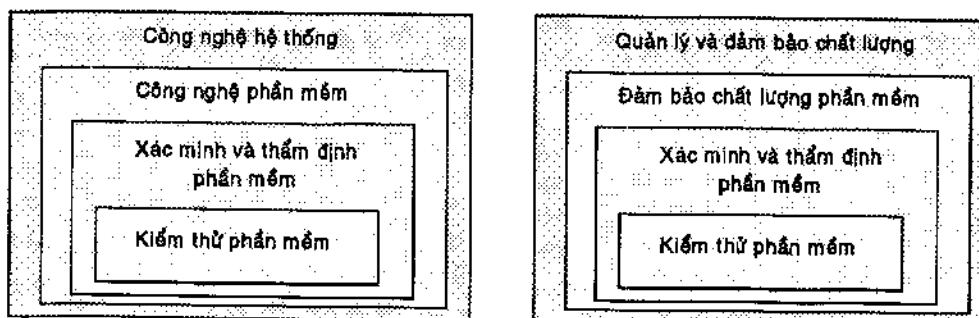
Chất lượng phần mềm là một khái niệm đa chiều, không dễ định nghĩa đơn giản theo cách chung cho các sản phẩm là: "Sản phẩm được phát triển phù hợp với đặc tả của nó." Có một số vấn đề khó trong hệ thống phần mềm, đó là:

Đặc tả phải định hướng theo những đòi hỏi về chất lượng của khách hàng (như tính hiệu quả, độ tin cậy, tính dễ hiểu, tính bảo mật,...) và những yêu cầu của chính tổ chức phát triển phần mềm vốn không có trong đặc tả (như các yêu cầu về khả năng bảo trì, tính sử dụng lại...)

Một số yêu cầu về chất lượng cũng rất khó chỉ ra một cách rõ ràng.

Những đặc tả phần mềm thường không đầy đủ và hay mâu thuẫn.

### Kiểm thử phần mềm trong một số môi trường:



Trên quan điểm qui trình, kiểm thử phần mềm là một phần của xác minh và thẩm định phần mềm. Xác minh và thẩm định nằm trong công nghệ phần mềm, công nghệ phần mềm lại là một phần của công nghệ hệ thống.

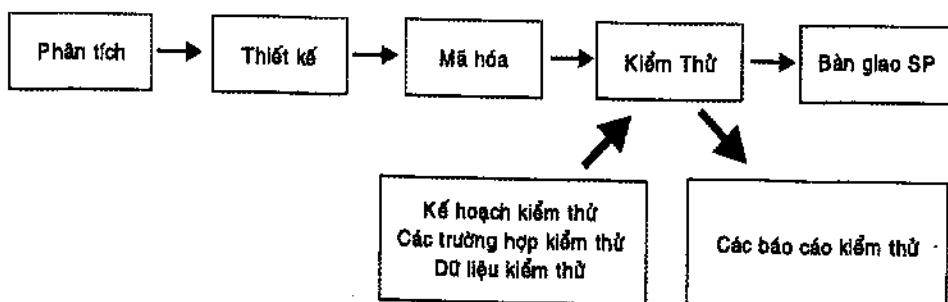
Nhìn từ ngữ cảnh chất lượng, kiểm thử phần mềm cũng là một phần của xác minh và thẩm định phần mềm, nên cũng có thể xem như là một phần của đảm bảo chất lượng phần mềm.

Nếu phần mềm là thành phần của hệ thống lớn hơn thì kiểm thử phần mềm cũng được xem như là một phần của quản lý và đảm bảo chất lượng. Và để đạt phần mềm chất lượng cao, thì kiểm thử có thể coi là một thành phần chủ yếu của hoạt động đảm bảo chất lượng phần mềm.

#### Qui trình kiểm thử phần mềm

Mục đích của kiểm thử là thiết kế một chuỗi các trường hợp kiểm thử có khả năng phát hiện lỗi cao. Để cho việc kiểm thử đạt được kết quả tốt cần có sự chuẩn bị về kế hoạch kiểm thử, thiết kế các trường hợp kiểm thử và các dữ liệu kiểm thử cho các trường hợp.

Đây chính là đầu vào cho giai đoạn kiểm thử. Và sản phẩm công việc của giai đoạn kiểm thử chính là “báo cáo kiểm thử” mà tài liệu hóa tất cả các trường hợp kiểm thử đã chạy, dữ liệu đầu vào, đầu ra mong đợi, đầu ra thực tế và mục đích của kiểm thử.



## GIAI ĐOẠN KIỂM THỬ TRONG XỬ LÝ PHẦN MỀM

**Qui trình kiểm thử bao gồm một số giai đoạn:**

Lập kế hoạch kiểm thử. Bước đầu tiên là lập kế hoạch cho tất cả các hoạt động sẽ được thực hiện và các phương pháp được sử dụng.

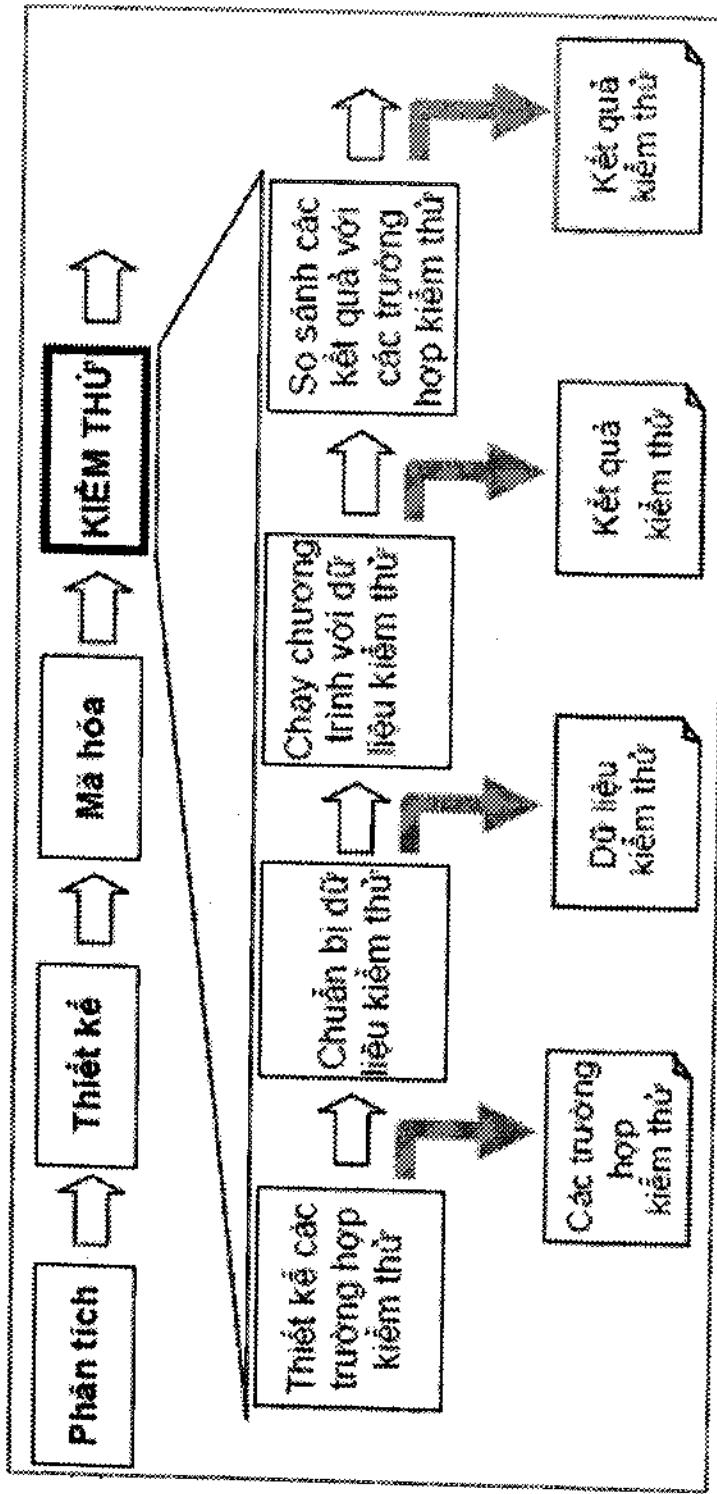
Các chuẩn IEEE bao gồm các thông tin về tác giả chuẩn bị kế hoạch, danh sách liệt kê của kế hoạch kiểm thử.

Một số vấn đề sau là quan trọng đối với kế hoạch kiểm thử.

- **Mục đích:** qui định về phạm vi, phương pháp, tài nguyên và lịch biểu của các hoạt động kiểm thử.
- **Các tài liệu tham khảo.**
- **Các định nghĩa.**
- **Khái quát về xác minh và thẩm định:** tổ chức, tài nguyên, trách nhiệm, các công cụ, kỹ thuật và các phương pháp luận.
- **Vòng đời:** các nhiệm vụ, các dữ liệu vào và các kết quả ra trên một giai đoạn vòng đời.
- **Báo cáo xác minh và thẩm định phần mềm:** mô tả nội dung, định dạng và thời gian cho tất cả các báo cáo.
- **Các thủ tục quản lý:** bao gồm các chính sách, thủ tục, các chuẩn, thực nghiệm và các quy ước.
- **Giai đoạn bố trí nhân viên kiểm thử:** việc kiểm thử thường phải tiến hành một cách độc lập và các nhóm độc lập có trách nhiệm tiến hành các hoạt động kiểm thử, gọi là các nhóm kiểm thử.
- **Thiết kế các trường hợp kiểm thử:** các trường hợp kiểm thử là các đặc tả đầu vào cho kiểm thử và đầu ra mong đợi của hệ thống cùng với các câu lệnh được kiểm thử.
- **Các phương pháp hộp đen để kiểm thử dựa trên chức năng.**
- **Các phương pháp hộp trắng để kiểm thử dựa vào cấu trúc bên trong.**
- **Xử lý do lưỡng kiểm thử bằng cách thu thập dữ liệu.**
- **Đánh giá sản phẩm phần mềm để xác nhận sản phẩm có thể sẵn sàng phát hành được chưa?**

**Mô hình chung của quy trình kiểm thử phần mềm được thể hiện trong hình trang bên.**

# MÔ HÌNH CHUNG CỦA QUI TRÌNH KIỂM THỬ PHẦN MỀM

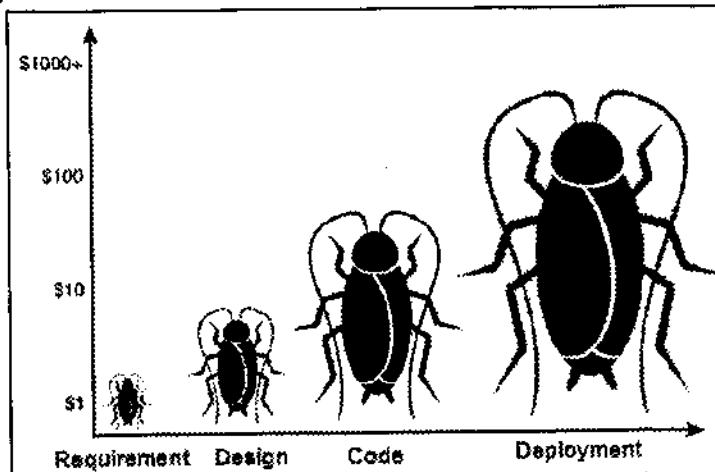


## CHƯƠNG 2

# TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

### 1. KIỂM THỬ PHẦN MỀM

Trong quá trình phát triển và phục vụ các tiện ích cho cuộc sống, nhiều sản phẩm phần mềm ra đời phục vụ nhu cầu của người dùng. Các sản phẩm này trải qua một quy trình xây dựng và thử nghiệm bởi các nhà sản xuất và các chuyên gia lập trình. Quy trình này qua các giai đoạn như: **Requirement** (Yêu cầu) > **Design** (Thiết kế) > **Coding** (mã hóa) > **Testing** (Kiểm thử) > **Deployment** (Triển khai). Trong các giai đoạn, chi phí kiểm thử sẽ tăng theo thời gian kiểm thử.



Phản quan trọng để đánh dấu sự ra đời của một sản phẩm công nghệ là giai đoạn Testing. Bất kỳ một sản phẩm nào, đến tay người dùng đều trải qua một quá trình kiểm nghiệm gắt gao bởi các chuyên gia kiểm thử (người ta thường gọi là Tester). Các Tester thực hiện quá trình kiểm thử và thử nghiệm các ứng dụng phần mềm, tìm ra các lỗi sai trong quá trình thiết kế hay xây dựng. Sản phẩm vượt qua đánh giá thẩm định này sẽ được trao đến tay khách hàng của họ. Công cuộc đánh giá đòi hỏi phải theo sát quá trình phát triển của phần mềm và thực hiện các chuỗi thực nghiệm để đánh giá và đưa ra các lỗi sai sót. Như vậy quá trình phát triển của một phần mềm phải kết hợp với quá trình kiểm thử và thử nghiệm này.

Nền tảng này cung cấp các khái niệm và các mục tiêu cho việc kiểm định phần mềm. Trong giai đoạn đầu của quá trình xây dựng phần mềm, các chuyên gia sẽ đưa những yêu cầu thành một ứng dụng cụ thể.

Những ứng dụng này sẽ giúp việc kinh doanh phần mềm thành công khi vượt qua giai đoạn kiểm định. Việc kiểm định sẽ tìm ra tất cả các thành phần lỗi trong phần mềm, nhằm mục đích cung cấp các lợi ích chất lượng cho sản phẩm và tối ưu hóa trong nhiều lĩnh vực khác nhau.

Ngày nay việc kiểm định được cải thiện, tạo hiệu quả hơn thông qua những ứng dụng chuyên dùng. Dựa trên các mục tiêu của phần mềm, mà quy trình kiểm định xác định được các lỗi sai của một phần mềm trong môi trường ứng dụng thực tiễn. Nhưng lỗi và khuyết điểm sẽ được ghi lại và đưa vào đánh giá sau đó loại bỏ hay sửa chữa.

## 2. QUÁ TRÌNH KIỂM THỬ

Quá trình kiểm thử diễn ra trên một hệ thống phần mềm bắt đầu từ việc nắm rõ các yêu cầu ban đầu của phần mềm, theo sau là quy trình thiết kế và mã hóa chương trình. Những thông tin này gọi chung là cấu hình của một sản phẩm phần mềm.

Bước tiếp theo, thực hiện kế hoạch cho việc tiến hành kiểm thử và thực hiện các thủ tục có liên quan. Vận hành chương trình phần mềm và bắt lỗi ngay trên môi trường kiểm thử.

- **Thông tin cấu hình của phần mềm** (Software configuration): các thông tin này bao gồm mô tả về yêu cầu của phần mềm (Software Requirement Specification), Mô tả về thiết kế của chương trình (Design Specification) và các mã của chương trình.
- **Thông tin cấu hình về kiểm thử bao gồm:** kế hoạch kiểm thử, thủ tục kiểm thử và các chương trình chạy kiểm thử như: chương trình giả lập môi trường, chương trình tạo các trường hợp kiểm thử... Các trường hợp kiểm thử phải đi cùng với kết quả mong muốn. Trong thực tế những thông tin này cũng là một phần của cấu hình phần mềm (software configuration) ở trên.

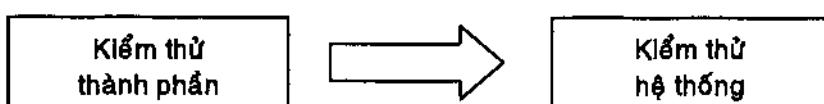
Bắt đầu từ các thông tin đầu vào của phần mềm được kiểm thử và đưa ra kết quả để so sánh, đánh giá kết quả với các kết quả mong muốn. Khi việc so sánh thất bại thì mỗi lỗi được phát hiện và quá trình sửa lỗi sẽ bắt đầu trở lại. Quá trình sửa lỗi không thể dự tính trước, do lỗi được gây ra bởi sự khác nhau của kết quả kiểm thử. Sẽ tốn nhiều thời gian và công sức để tìm ra nguyên nhân và chỉnh sửa và cũng chính sự không chắc chắn này mà làm cho quá trình kiểm định rất khó đưa ra một tiến trình chắc chắn.

**Quá trình kiểm thử phần mềm có hai mục tiêu:**

1. Chứng minh cho người xây dựng phần mềm và khách hàng thấy các yêu cầu của phần mềm. Với phần mềm truyền thống, điều này có nghĩa là phải có ít nhất một thử nghiệm cho mỗi yêu cầu của người dùng và tài liệu hệ thống yêu cầu.

Với các sản phẩm phần mềm chung, điều đó có nghĩa là: nên thử nghiệm tất cả các đặc tính của hệ thống sẽ được kết hợp trong sản phẩm phát hành.

2. Phát hiện các lỗi và khiếm khuyết trong phần mềm: phần mềm thực hiện không đúng, không như mong đợi hoặc không làm theo như đặc tả. Kiểm tra khiếm khuyết tập trung vào việc tìm ra tất cả các kiểu thực hiện không như mong đợi của hệ thống, như sự đổ vỡ hệ thống, sự tương tác không mong muốn với hệ thống khác, tính toán sai và sai lạc dữ liệu.



## Người phát triển phần mềm

#### **Nhóm kiểm thử độc lập**

Lúc kết quả kiểm định được thống kê và đánh giá, chất lượng và độ tin cậy của một phần mềm được ước lượng.

Chất lượng của chương trình không tốt nếu có những lỗi nghiêm trọng xảy ra thường xuyên và những lỗi này dẫn đến cần phải thay đổi thiết kế của chương trình. Nhưng nếu ngược lại các module/hàm đều hoạt động đúng đắn như thiết kế ban đầu và những lỗi được tìm thấy có thể chỉnh sửa dễ dàng, thì 2 kết luận có thể được đưa ra :

- Chất lượng của phần mềm là chấp nhận được
  - Những kiểm định có thể không thỏa đáng/thích hợp để phát hiện ra những lỗi nghiêm trọng đã đề cập trên.

Nếu quá trình kiểm định phát hiện không có lỗi sẽ có một chút nghi ngờ rằng: những thông tin cấu hình về kiểm thử không đủ và lỗi vẫn tồn tại trong phần mềm.

Những lỗi này sẽ được phát hiện sau này bởi người sử dụng và được chỉnh sửa bởi lập trình viên nhưng ở giai đoạn bảo trì và chi phí của những công việc này sẽ tăng lên 60 đến 100 lần so với chi phí cho mỗi lần chỉnh sửa trong giai đoạn phát triển.

Ta thấy rằng chi phí tiêu tốn quá nhiều cho quá trình bảo trì để chỉnh sửa một lỗi do đó cần phải có những kỹ thuật hiệu quả để tạo được các trường hợp kiểm thử tốt.

### 3. NGUYỄN TẮC KIỂM THỬ PHẦN MỀM TRÊN THỰC TẾ

Để thực hiện tốt quá trình kiểm định phần mềm, cần nắm rõ các nguyên tắc kiểm thử dựa trên thực tế sau:

1. **Testing show presence of defects** (kiểm thử và chỉ ra lỗi): kiểm thử thử nghiệm để cho thấy phần mềm đang có lỗi nhưng không xác định được chính xác lỗi đó. Kiểm thử giúp làm giảm xác suất lỗi chưa nhận ra trong phần mềm (hoặc không có lỗi).
2. **Exhaustive testing is impossible**: kiểm thử một số điểm cần thiết dựa trên mức độ ưu tiên trong phần mềm. Việc kiểm thử toàn bộ khó có thể thực hiện được. Vì vậy dựa trên sự phân tích rủi ro chỉ cần tiến hành kiểm thử điểm nghi vấn hay lỗi chính trong tổ hợp điều kiện.
3. **Early testing** (kiểm thử sớm): thực hiện kiểm thử càng sớm thì càng tốt cho quá trình phát triển của phần mềm. Dễ dàng cho việc tìm ra lỗi và chỉnh sửa. Nhờ vậy, có thể tập trung vào các hoạt động trên kế hoạch vạch trước.
4. **Defect clustering** (phân loại lỗi): việc kiểm thử thực hiện cân bằng và tập trung vào mật độ của các lỗi trong dự kiến và phát hiện các lỗi trong các mô đun. Trong các mô đun thường tồn tại nhiều lỗi ẩn, không dễ phát hiện ra trong lúc kiểm thử phần mềm.

Để hiểu rõ hơn nguyên tắc này, hãy xem xét 3 điều sau:

- **Nguyên tắc tổ gián**: chỗ nào có 1 vài con gián thì ở đâu đó xung quanh nó sẽ có cả tổ gián > có rất nhiều gián > chỗ nào có 1 vài lỗi thì xung quanh đó sẽ có nhiều lỗi.
- **Nguyên tắc 80/20**: thông thường 20% chức năng quan trọng trong một chương trình có thể gây ra đến 80% tổng số lỗi phát hiện được trong chương trình đó.
- **Exhaustive testing is impossible (nguyên tắc thứ 2)**: do đó cần phải analysis (phân tích) + priorities (tính toán mức độ ưu tiên) để quyết định tập trung vào kiểm thử chỗ nào.

Test kỹ chức năng quan trọng > tìm lỗi > kiểm thử những gì liên quan những chức năng gần nó để tìm ra lỗi nhiều hơn.

5. **Pesticide paradox (nghịch lý thuốc trừ sâu)**: việc kiểm thử lặp lại nhiều lần dẫn đến trường hợp một trong số chúng không tìm được lỗi nào. Để khắc phục điều này, cần sử dụng nguyên tắc Pesticide paradox, áp dụng thay đổi thường xuyên cách kiểm tra. Thực hiện nhiều test case khác nhau sẽ giúp tìm ra được nhiều lỗi ẩn bên trong hơn.
6. **Testing is context dependent** (kiểm thử theo bối cảnh độc lập): việc kiểm thử phụ thuộc vào bối cảnh, và thực hiện kiểm thử trong ngữ cảnh khác nhau. Cùng là một phần mềm, có thể đặt vào các bối cảnh ứng dụng khác nhau để tìm ra lỗi.

**7. Absence-of-errors fallacy** (sal lầm về việc không có lỗi): việc tìm kiếm và sửa chữa lỗi sẽ không còn giá trị nếu hệ thống được phát triển xong, nhưng không thể dùng được hay đáp ứng được yêu cầu và sự mong đợi của người dùng.

### 7. CÁC NGUYÊN TẮC KIỂM THỬ PHẦN MỀM KHÁC:

1. Một phần quan trọng của 1 cuộc kiểm thử là định nghĩa của đầu ra hay kết quả mong muốn.
2. Lập trình viên nên tránh tự kiểm thử phần mềm do bản thân xây dựng.
3. Nhóm lập trình không nên kiểm thử phần mềm của chính họ.
4. Kiểm thử kỹ lưỡng mọi kết quả của mỗi kiểm thử, các cuộc kiểm thử phải được viết cho các trạng thái đầu vào không hợp lệ và không mong muốn, tương tự áp dụng như cho các đầu vào hợp lệ và mong muốn.
5. Khảo sát 1 phần mềm để xem liệu nó có thực hiện chức năng cần thực hiện chỉ là một phần, phần còn lại là xem liệu phần mềm có thực hiện cái mà nó không cần phải thực hiện hay không.
6. Tránh các cuộc kiểm thử sơ sài, trừ khi phần mềm thực sự là 1 phần mềm đơn giản, không dự kiến kết quả của kiểm thử theo giả thiết mặc định sẽ dẫn đến việc không tìm thấy lỗi.
7. Xác suất tồn tại lỗi trong 1 đoạn phần mềm là tương ứng với số lỗi đã tìm thấy trong đoạn đó.
8. Kiểm thử là 1 nhiệm vụ cực kỳ sáng tạo và có tính thử thách trí tuệ.

### 8. TRÌNH TỰ KIỂM THỬ

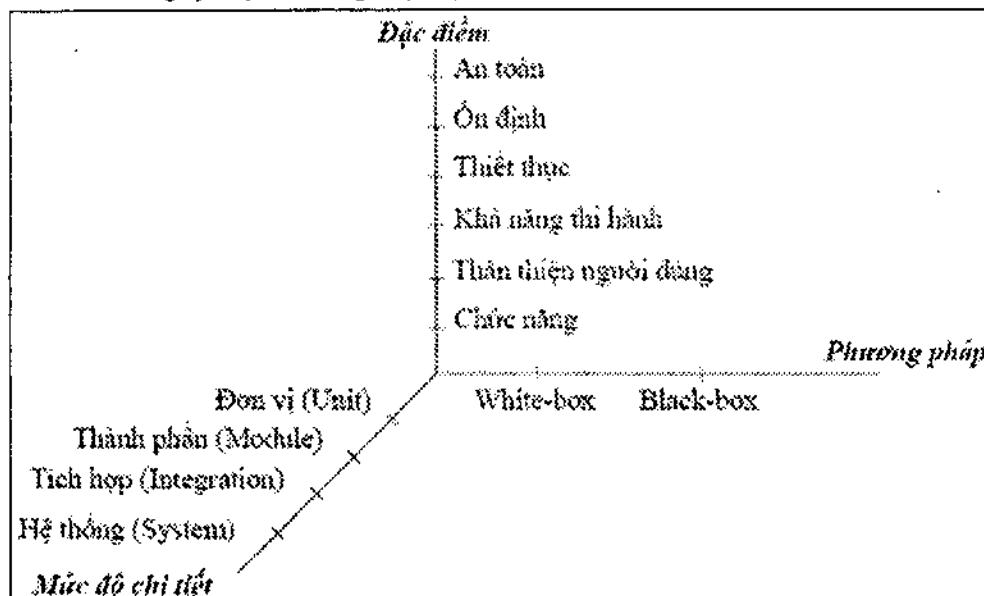
Là trình tự thực hiện kiểm thử hệ thống phần mềm từ khi có yêu cầu kiểm thử cho đến khi sản phẩm được thông báo, đưa ra sử dụng. Một trình tự kiểm thử bao gồm các bước.

- Lập kế hoạch kiểm thử (Test Plan).
- Chuẩn bị môi trường kiểm thử.
- Thiết kế kiểm thử (Test case).
- Thực hiện kiểm thử, theo dõi và xử lý lỗi.
- Thống kê báo cáo kết quả kiểm thử, thông báo phát hành sản phẩm.

### 9. VÒNG ĐỜI CỦA VIỆC KIỂM THỬ

Mỗi quá trình kiểm thử diễn ra đều có một thời gian tiến hành tất cả các công đoạn kiểm thử, có thể gọi đó là vòng đời kiểm thử cho một phần mềm. Hình trang bên mô tả các công đoạn phát triển một phần mềm và cách khắc phục lỗi. Lỗi có thể xảy ra trong tất cả các công đoạn từ: Mô tả yêu cầu > Thiết kế > Lập trình. Như vậy từ công đoạn này chuyển sang công đoạn khác thường滋生 các sai sót (do dư thừa hoặc thiếu theo mô tả yêu cầu).

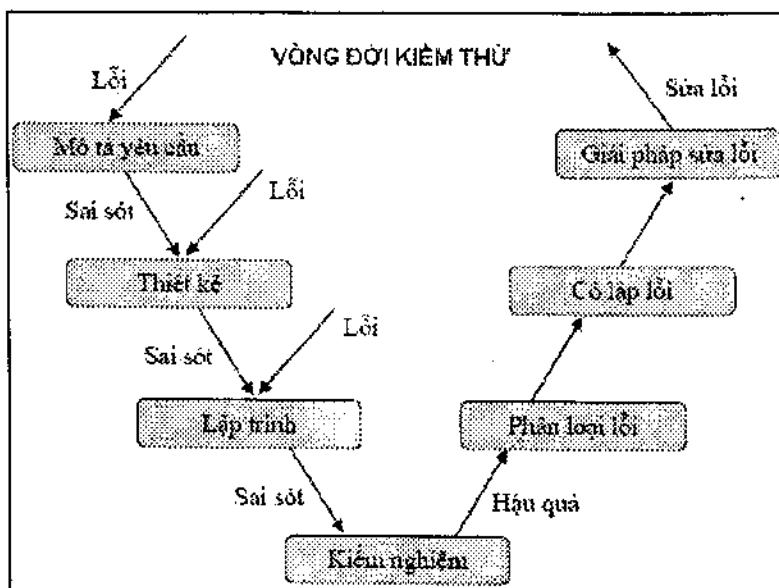
Đến công đoạn kiểm nghiệm sẽ phát hiện ra các kết quả không mong muốn. Quá trình sửa lỗi bao gồm "phân loại lỗi", "cô lập lỗi" (tìm ra nguyên nhân và nơi gây lỗi), để ra "giải pháp sửa lỗi" và cuối cùng là khắc phục lỗi.



### Tương quan giữa các công đoạn xây dựng phần mềm và loại kiểm thử

Mô hình V nhằm giải thích sự tương quan giữa các công đoạn xây dựng phần mềm và các loại kiểm thử.

Ở mỗi công đoạn xây dựng phần mềm sẽ tương ứng với một loại kiểm thử và cần có một tài liệu tương ứng được thành lập để phục vụ cho việc kiểm thử.



**CHƯƠNG 3****CHIẾN LƯỢC KIỂM THỬ**

Một chiến lược kiểm thử phần mềm bao gồm các phương pháp thiết kế, đưa kiểm thử phần mềm thành một chuỗi các bước được lập kế hoạch rõ ràng về cấu trúc để khi thực hiện kiểm thử đạt hiệu quả.

Điều quan trọng nhất trong chiến lược kiểm thử phần mềm là cung cấp một phương pháp cho người xây dựng phần mềm, tổ chức đảm bảo chất lượng và nguồn khách hàng.

Chiến lược này có thể cùng được thực hiện cùng với quá trình xây dựng phần mềm hoặc trễ hơn tùy vào quy mô của phần mềm đó. Nó được thành lập bởi các nhóm có chuyên môn kiểm thử hoặc áp dụng bởi những chuyên gia thiết kế - xây dựng, để hạn chế lỗi và rút ngắn thời gian.

### **1. NGUYÊN LÝ THIẾT KẾ VÀ KIỂM THỬ PHẦN MỀM**

Trước khi áp dụng các phương pháp để thiết kế các trường hợp kiểm thử hiệu quả, người thiết kế phần mềm cần hiểu các nguyên lý cơ bản hướng dẫn việc kiểm thử phần mềm.

- Tất cả các kiểm thử phải có thể mô tả theo các yêu cầu của khách hàng.
- Các kiểm thử phải được lập kế hoạch từ lâu trước khi kiểm thử bắt đầu.
- Kiểm thử cần bắt đầu trong "phạm vi nhỏ" và quá trình hướng đến các kiểm thử trong "phạm vi rộng".
- Kiểm thử toàn diện là không thể xảy ra.
- Để đạt hiệu quả nhất, kiểm thử cần thực hiện bởi một nhóm làm việc độc lập thứ ba.
- Một lập trình viên nên tránh việc cố gắng kiểm thử chương trình của chính mình; đồng thời một tổ chức lập trình cũng không nên tự kiểm thử phần mềm của chính họ.

Các trường hợp kiểm thử phải được viết cho các điều kiện đầu vào không hợp lệ và không mong đợi, cũng như các điều kiện hợp lệ và được mong đợi. Và một phần cần thiết nữa là phải xác định đầu ra hay kết quả mong đợi. Vì vậy, một trường hợp kiểm thử phải gồm hai phần:

- Mô tả chi tiết đầu vào hợp lệ được mong đợi hoặc không hợp lệ, không được mong đợi.

- Mô tả chi tiết dấu ra đúng cho một tập đầu vào tương ứng.
- Kiểm tra cẩn thận kết quả của mỗi trường hợp:**
- Kiểm tra một chương trình xem nó có thực hiện đúng những gì nó phải thực hiện và những gì dự kiến không thực hiện.
- Tránh bỏ qua những trường hợp kiểm thử trừ khi chương trình thực sự là một sản phẩm bỏ đi.
- Không nên đặt kết quả dưới một giả định rằng sẽ không phát hiện một lỗi nào.
- Xác suất tồn tại lỗi càng cao ở những phần có nhiều lỗi được phát hiện.
- Kiểm thử phần mềm là một nhiệm vụ mang tư duy sáng tạo và tính trách nhiệm cao.

#### **Phương pháp tiếp cận kiểm thử phần mềm:**

Kiểm thử là một tập các hoạt động có thể được lập kế hoạch trước và được thực hiện một cách có hệ thống. Vì lý do đó, một khuôn mẫu để kiểm thử phần mềm (tập hợp các bước xác định các phương pháp thiết kế trường hợp kiểm thử) sẽ được định nghĩa cho quá trình phát triển phần mềm.

Chiến lược kiểm thử phần mềm cung cấp cho người xây dựng một khung biểu mẫu kiểm thử mang các đặc điểm sau:

- Kiểm thử bắt đầu tại mức module và các công việc “phát triển” hướng tới việc tích hợp toàn bộ hệ thống.
- Các kỹ thuật kiểm thử khác nhau thích hợp tại những thời điểm cũng khác nhau.
- Kiểm thử được thực hiện bởi người phát triển phần mềm và nhóm kiểm thử độc lập (cho các dự án lớn).
- Kiểm thử và gỡ rối là các hoạt động khác nhau, nhưng gỡ rối phải có trong mọi chiến lược kiểm thử.

#### **Xác minh và thẩm định:**

Kiểm thử phần mềm là một phần của để tài rộng hơn mà thường được đề cập tới như là sự xác minh và thẩm định (V&V). Thẩm định và xác minh là từ chung để chỉ quá trình kiểm tra để đảm bảo phần mềm thỏa mãn các yêu cầu của chúng và các yêu cầu đó đáp ứng yêu cầu của khách hàng.

Xác minh là một tập các hoạt động đảm bảo rằng, phần mềm cài đặt chức năng cụ thể một cách chính xác. Thẩm định là tập hợp hoạt động khác đảm bảo phần mềm đã được xây dựng theo đúng các yêu cầu của khách hàng đặt ra.

#### **Có thể phát biểu theo cách khác:**

- Xác minh (Verification):** trả lời cho câu hỏi “Sản phẩm có đúng với thiết kế không?”.

- **Thẩm định (Validation):** trả lời cho câu hỏi “Sản phẩm có đúng với yêu cầu thực tiễn không?”.

Xác minh và thẩm định là một phần của hoạt động đảm bảo chất lượng phần mềm, bao gồm việc duyệt lại kỹ thuật, kiểm định chất lượng và cấu hình, theo dõi hiệu suất, mô phỏng, nghiên cứu tính khả thi, duyệt lại tài liệu, xem lại cơ sở dữ liệu, phân tích thuật toán, kiểm thử phát triển, kiểm thử chất lượng và kiểm thử cài đặt. Kiểm thử đóng vai trò rất quan trọng trong việc xác minh và thẩm định phần mềm và nhiều hoạt động khác trong phát triển phần mềm.

## 2. TỔ CHỨC VIỆC KIỂM THỬ

Với mọi dự án phần mềm, có một xung đột cố định về quyền lợi xuất hiện khi kiểm thử bắt đầu. Những người xây dựng phần mềm được yêu cầu kiểm thử phần mềm. Điều này tưởng như vô hại:

Sau tất cả các thành phần xây dựng dự án, không có ai hiểu rõ chương trình hơn người phát triển.

Từ quan điểm tâm lý, phân tích và thiết kế phần mềm (cùng với mã hóa) là những công việc xây dựng. Người kỹ sư phần mềm tạo ra các chương trình máy tính, các tài liệu của nó và các cấu trúc dữ liệu liên quan.

Thường có một số quan niệm sai có thể dẫn đến kết luận sai từ sự tranh luận trên:

1. Người phát triển phần mềm sẽ không thực hiện một kiểm thử nào.
2. Phần mềm sẽ được đưa ra để một người lạ sẽ kiểm thử nó một cách khắt khe, chi tiết.
3. Những người kiểm thử tham gia dự án chỉ khi các bước kiểm thử dự án sắp bắt đầu.

**Mỗi phát biểu trên là không đúng:**

Người phát triển phần mềm luôn có trách nhiệm kiểm thử các đơn vị (module) riêng biệt của chương trình, đảm bảo rằng mỗi đơn vị thực hiện chức năng mà nó đã được thiết kế.

Val trò của nhóm kiểm thử độc lập (ITG) là loại bỏ các vấn đề cố hữu liên quan đến việc người phát triển tự kiểm thử những gì đã được xây dựng. Kiểm thử độc lập cũng loại bỏ các xung đột khác có thể xảy ra. Cuối cùng, nhân viên nhóm độc lập được trả lương để tìm các lỗi.

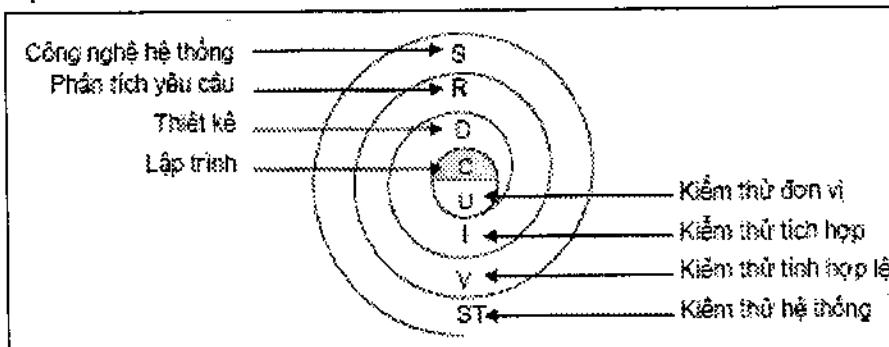
### Chiến lược kiểm thử phần mềm

Tiến trình công nghệ phần mềm có thể được xem như một xoắn ốc, việc phát triển phần mềm, đi vào đọc theo đường xoắn ốc, giảm dần trên mỗi vòng, gồm các bước:

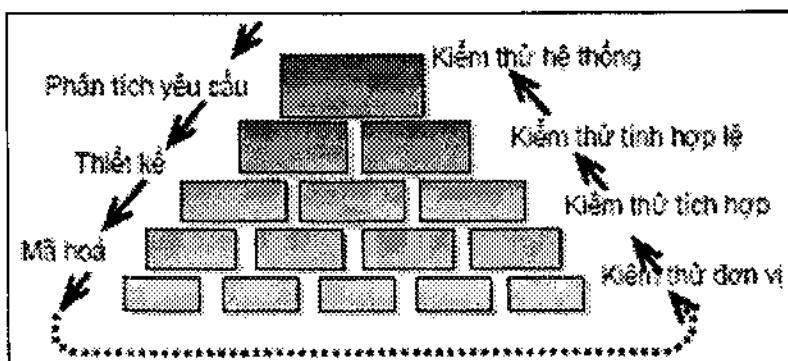
**Công nghệ hệ thống > Phân tích yêu cầu > Thiết kế > Mã hóa.**

Chiến lược kiểm thử phần mềm cũng có thể di chuyển dọc theo đường xoắn ốc và di ra theo đường xoắn ốc theo luồng mở rộng phạm vi kiểm thử trên mỗi vòng, tức theo thứ tự ngược lại, tương ứng như sau:

Kiểm thử hệ thống - Kiểm thử tính hợp lệ - Kiểm thử tích hợp - Kiểm thử đơn vị.



Theo quan điểm thủ tục, kiểm thử nằm trong ngữ cảnh công nghệ phần mềm trên thực tế là dãy bốn bước được cài đặt tuần tự. Các bước được mô tả như hình:



- Kiểm thử đơn vị:** tập trung trên mỗi module riêng biệt, đảm bảo rằng các chức năng của nó tương ứng với một đơn vị.
- Kiểm thử tích hợp:** tập trung vào việc thiết kế và xây dựng kiến trúc phần mềm.
- Kiểm thử tính hợp lệ:** trong đó các yêu cầu đã được thiết lập như một phần của việc phân tích yêu cầu phần mềm được thẩm định, dựa vào phần mềm đã xây dựng. Tiêu chuẩn hợp lệ cần được kiểm thử.
- Kiểm thử hệ thống:** là một phần của công nghệ hệ thống máy tính, trong đó phần mềm và các thành phần khác của hệ thống được kiểm thử. Kiểm thử hệ thống nhằm xác minh rằng tất cả các thành phần hệ thống khớp nhau một cách hợp lý, và tất cả các chức năng hệ thống và hiệu suất đạt được theo yêu cầu.

### Điều kiện hoàn thành kiểm thử

Một câu hỏi khó trong kiểm thử phần mềm, đó là: "Khi nào chúng ta hoàn thành việc kiểm thử - và làm thế nào để biết chúng ta đã kiểm thử đủ?".

Không có câu trả lời dứt khoát cho câu hỏi này.

Thật ra, "không bao giờ hoàn thành việc kiểm thử, trách nhiệm này thường chuyển từ người phát triển cho các khách hàng". Mỗi lần, khách hàng (người sử dụng) thực hiện chương trình máy tính, chương trình sẽ được kiểm thử với tập dữ liệu mới. Có rất nhiều tranh cãi về câu trả lời cho câu hỏi trên, tuy nhiên, các kỹ sư phần mềm cần phải có các tiêu chuẩn chặt chẽ để xác định khi nào kiểm thử đạt hiệu quả.

Helsel (1997) đưa ra bốn tiêu chuẩn có thể cho việc kết thúc kiểm thử:

- Khi dự án hết tiền hoặc thời gian.
- Khi đội ngũ kiểm thử không tiến hành được thêm một trường hợp kiểm thử nào.
- Khi kiểm thử được tiếp tục mà không phát hiện được bất kỳ lỗi mới nào.
- Khi đạt đến một mức của độ phủ thích hợp.

Chiến lược phổ biến nhất hiện nay là kiểm thử cho đến khi dự án hết tiền hoặc thời gian. Tuy nhiên, chiến lược này sẽ bao gồm một vài rủi ro: nếu việc phát triển đã vượt quá ngân sách thì việc kiểm thử sẽ mất chất lượng.

Một chiến lược khác là sử dụng mô hình thống kê và lý thuyết độ tin cậy phần mềm, các mô hình thất bại phần mềm (được phát hiện trong quá trình kiểm thử) theo hàm thời gian thực hiện có thể được phát triển. Mô hình thất bại được gọi là mô hình thời gian thực hiện Log Poisson, có dạng:

$$f(t) = \left(\frac{1}{p}\right)x \ln[ l_0(pt + 1) ]$$

Trong đó:

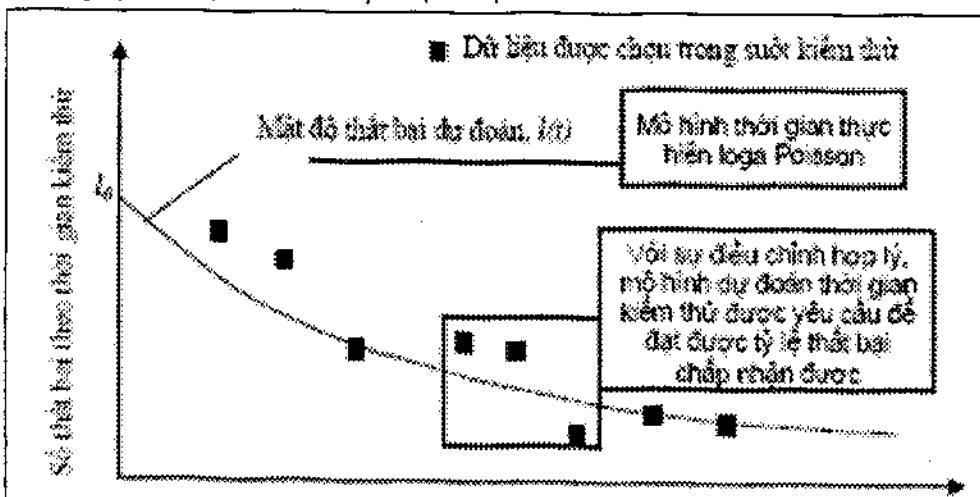
- $f(t)$  là số thất bại lũy tích được dự đoán xuất hiện mỗi lần phần mềm được kiểm thử trong khoảng thời gian thực hiện  $t$  nào đó.
- $l_0$  là cường độ thất bại phần mềm ban đầu (số thất bại trên một đơn vị thời gian) khi bắt đầu kiểm thử.
- $p$  là biến đổi số mũ trong cường độ thất bại do các lỗi được phát hiện và các khắc phục được thực hiện.

Cường độ thất bại tức thời,  $I(t)$  có thể được suy ra bằng cách tính đạo hàm  $f(t)$ :

$$l(t) = \frac{l_0}{l_0 + pt + 1}$$

Sử dụng quan hệ được ghi trong phương trình trên, người kiểm thử có thể dự đoán việc giảm lỗi trong quá trình kiểm thử.

Cường độ lỗi thực tế có thể được vẽ dọc theo đường cong dự đoán (hình dưới). Nếu dữ liệu thực tế được tập hợp lại trong quá trình kiểm thử và mô hình thời gian thực hiện Log Poisson là phù hợp với nhau trên một số điểm dữ liệu, mô hình có thể được sử dụng để dự đoán tổng thời gian thực hiện cần để đạt được tỷ lệ thất bại có thể chấp nhận được.



Bằng các phép tập hợp trong việc kiểm thử và sử dụng các mô hình độ tin cậy phần mềm đang tồn tại, có thể phát triển chỉ dẫn để trả lời cho câu hỏi: "Khi nào chúng ta hoàn thành kiểm thử?"

Hình trên chỉ ra mối quan hệ giữa số lượng kiểm thử được thực hiện và số lỗi được tìm thấy. Nếu chúng ta kiểm thử quá nhiều thì chi phí sẽ tăng một cách khó chấp nhận được, ngược lại nếu kiểm thử ít thì chi phí thấp, nhưng sẽ còn nhiều lỗi. Số lượng kiểm thử tối ưu chỉ ra rằng chúng ta không kiểm thử quá nhiều nhưng cũng không ít quá.

## CHƯƠNG 4

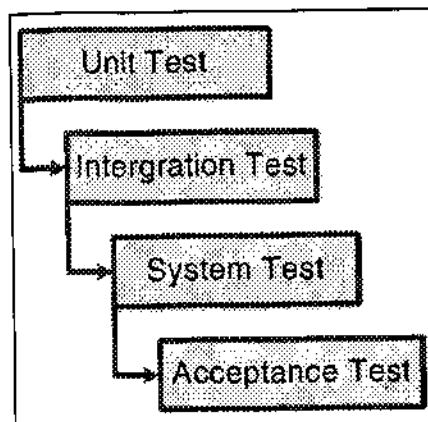
# CÁC GIAI ĐOẠN KIỂM THỬ (TESTING)

## 1. CÁC CẤP ĐỘ KIỂM THỬ

Như đã trình bày ở chương 3, sau các bước Phân tích yêu cầu > Thiết kế > Mã hóa cho ra sản phẩm phần mềm. Bước tiếp theo, cần kiểm thử phần mềm trước khi đưa sản phẩm tới người dùng.

Có 4 cấp độ kiểm thử như sau:

- **Unit Test:** kiểm thử đơn vị.
- **Integration Test:** kiểm thử tích hợp.
- **System Test:** kiểm thử hệ thống.
- **Acceptance Test:** kiểm thử tính hợp lệ.



Qua tham khảo nhiều sách – Ebook (Trình bày trong tài liệu tham khảo), các sách cũng đều trình bày kiểm thử theo 4 cấp độ trên do các sách đều căn cứ vào việc lập kế hoạch cụ thể cho từng cấp độ (Detail Test Planning) dựa vào những yêu cầu có được ở pha khảo sát. Phần trình bày sau giúp bạn đọc hiểu rõ hơn Tiến trình thực hiện phát triển phần mềm (Software development process) cũng như nội dung của 4 cấp độ kiểm thử.

### UNIT TEST – KIỂM THỬ ĐƠN VỊ

**Đơn vị Unit:** một Unit được coi là một thành phần nhỏ trong phần mềm có thể kiểm thử. Các thành phần được coi là unit như: Function (hàm), Procedure (thủ tục), Class (lớp), Method (phương thức)..

Unit được dùng để kiểm thử các chức năng đơn giản có kích thước nhỏ, vì vậy không gây nhiều khó khăn cho việc thực hiện kiểm định và phân tích kết quả. Phương pháp thông thường là: khoanh vùng 1 đơn vị unit, kiểm thử nguyên nhân và khắc phục ngay khi tìm được lỗi.

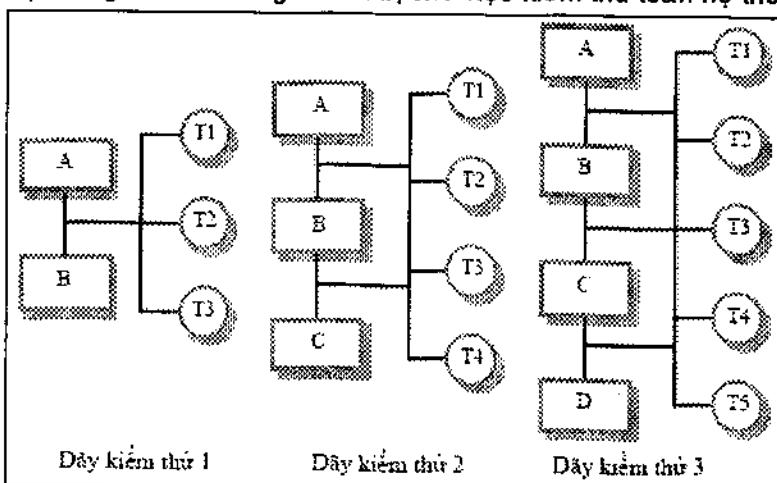
Kỹ thuật kiểm định sẽ được thực hiện theo một quy trình riêng biệt so với quy trình phát triển của phần mềm. Do đó những lỗi trong chi tiết mã code và các vấn đề tiềm ẩn khác sẽ được phát hiện kịp thời.

Unit được coi là kỹ thuật kiểm nghiệm tự động, nó sẽ kiểm thử từng bộ phận của hệ thống bằng mã cấu trúc và được thực hiện thường xuyên định kỳ.

Giai đoạn này được thực hiện bởi các lập trình viên trong suốt quá trình xây dựng phần mềm. Vì vậy đòi hỏi tester cần có kiến thức thiết kế code của chương trình.

### INTEGRATION TEST – KIỂM THỬ TÍCH HỢP

Hình thức kiểm thử này là sự kết hợp các thành phần của ứng dụng, sau đó kiểm thử như nó đã hoàn thành. Unit Test thực hiện kiểm thử các thành phần và Unit riêng lẻ, nhưng Integration Test sẽ kết hợp chúng lại và kiểm thử sự giao tiếp giữa chúng. Phát hiện ra các lỗi giao tiếp và tích hợp unit đơn thành các hệ thống nhỏ, còn gọi là Subsystem. Tăng dần mức độ lên đến nguyên hệ thống và cuối cùng chuẩn bị cho việc kiểm thử toàn hệ thống.



#### Hai mục tiêu chính của kiểm thử tích hợp:

1. Phát hiện các lỗi giao tiếp xảy ra giữa các Unit.
2. Tích hợp các Unit đơn thành các hệ thống nhỏ (*Subsystem*) và sau cùng là nguyên hệ thống hoàn chỉnh (*System*) chuẩn bị cho kiểm thử ở mức hệ thống (*System Test*).

Các lập trình viên phải phát hiện các lỗi chức năng và bên trong các unit bằng Unit Test. Còn có một vài phương pháp kiểm thử đơn giản trên giao tiếp giữa các unit và các thành phần có liên quan khác. Nhưng riêng những giao tiếp liên quan đến unit chỉ được kiểm thử đầy đủ khi các unit kết hợp lại với nhau trong các bước kiểm thử tích hợp (Integration Test).

Kiểm thử tích hợp thường được thực hiện trên những Unit đã qua kiểm thử bằng phương pháp Unit Test, trừ một số trường hợp ngoại lệ thì các lỗi đã được sửa chữa cẩn thận ở mức Unit.

Không được hiểu nhầm ở giai đoạn Unit Test, các unit đã kiểm thử thì không cần thực hiện kiểm thử tích hợp. Thực tế việc tích hợp giữa các Unit dẫn đến những tình huống hoàn toàn khác.

Khi một nhóm unit được tích hợp vào một nhóm khác, tại một thời điểm và đã hoàn tất các đợt kiểm thử tích hợp. Đến lúc này chúng ta chỉ cần kiểm thử các giao tiếp của unit mới thêm vào. Việc này sẽ giúp cho lượng kiểm thử giảm đi và các sai sót cũng giảm được nhiều hơn.

#### Có 4 loại kiểm thử tích hợp như sau:

- **Kiểm thử cấu trúc (Structure Test):** hình thức kiểm thử cấu trúc này đảm bảo cho các thành phần bên trong của một chương trình chạy đúng. Đặc biệt quan trọng đối với hoạt động của các thành phần cấu trúc bên trong như câu lệnh và các nhánh.
- **Kiểm thử chức năng (Functional Test):** hình thức kiểm thử chức năng chỉ cần quan tâm đến chức năng của chương trình mà không quan tâm đến cấu trúc bên trong. Thực hiện khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.
- **Kiểm thử hiệu năng (Performance Test):** hình thức kiểm thử quá trình vận hành của hệ thống.
- **Kiểm thử khả năng chịu tải (Stress Test):** hình thức kiểm thử các giới hạn của hệ thống.

### SYSTEM TEST - KIỂM THỬ MỨC HỆ THỐNG

Nhiệm vụ của System Test là kiểm thử thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không. Kiểm thử mức độ hệ thống bắt đầu khi tất cả các bộ phận của phần mềm đã được tích hợp hoàn chỉnh. Loại kiểm thử này tốn rất nhiều công sức và thời gian.

Trong một số trường hợp, việc kiểm thử cần phải có một số thiết bị hỗ trợ gồm phần mềm hoặc phần cứng. Đặc biệt chú trọng các ứng dụng thời gian, hệ thống phân bố, hệ thống nhúng.

Ở mức độ hệ thống, chuyên viên kiểm thử thực hiện tìm kiếm lỗi và chú trọng đến việc đánh giá hoạt động, thao tác, sự tin cậy và các nhu cầu khác liên quan đến chất lượng của toàn hệ thống.

Sau khi hoàn thành kiểm thử tích hợp, hệ thống phần mềm đã được hình thành các thành phần đã được kiểm thử hoàn chỉnh. Thời điểm hiện tại chuyên viên kiểm thử bắt đầu kiểm thử hệ thống hoàn toàn. Nên thực hiện phương pháp kiểm thử này ở giai đoạn lập kế hoạch và phân tích yêu cầu.

Điểm khác nhau của 2 hình thức kiểm thử IntegrationTest và System Test:

- **System Test:** tập trung đến các hoạt động và lỗi trên toàn hệ thống.

- **Integration Test:** tập trung các giao tiếp giữa các unit hoặc các đối tượng làm việc cùng nhau. (Phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự liên hệ giữa chúng hoạt động đúng, trước khi thực hiện System Test).

Kiểm thử mức hệ thống thực hiện bước kiểm thử tất cả các hoạt động chức năng và yêu cầu chất lượng như: độ tin cậy, tính tiện ích, hiệu suất, khả năng bảo mật... Mức kiểm thử này thích hợp cho việc kiểm thử và phát hiện lỗi giao tiếp giữa phần mềm và phần cứng.

Kết thúc giai đoạn kiểm thử này, sản phẩm phần mềm có thể được công bố với khách hàng dùng thử và nhận đánh giá cuối cùng từ họ.

Mức kiểm thử hệ thống đòi hỏi mất khá nhiều thời gian, công sức và độ chính xác. Thường nhóm chuyên viên thực hiện quy trình kiểm thử này làm việc độc lập với nhóm xây dựng dự án phần mềm.

System Test bao gồm nhiều loại kiểm thử khác nhau, dưới đây là một số kiểm thử thường xuyên sử dụng:

- **Kiểm thử chức năng (Functional Test):** bảo đảm các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế.
- **Kiểm thử khả năng vận hành (Performance Test):** bảo đảm tối ưu việc phân bổ tài nguyên hệ thống nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng câu truy vấn...
- **Kiểm thử khả năng chịu tải (Stress Test hay Load Test):** bảo đảm hệ thống vận hành đúng dưới áp lực cao (nhieu người truy xuất cùng lúc). Stress Test tập trung vào các trạng thái tới hạn, các "diểm chết", các tình huống bất thường như đang giao dịch thì ngắt kết nối (xuất hiện nhiều trong test thiết bị như POS, ATM...).
- **Kiểm thử cấu hình (Configuration Test):** bảo đảm tính tương thích cấu hình.
- **Kiểm thử khả năng bảo mật (Security Test):** bảo đảm tính toàn vẹn, bảo mật của dữ liệu và của hệ thống.
- **Kiểm thử khả năng phục hồi (Recovery Test):** bảo đảm hệ thống có khả năng khôi phục trạng thái ổn định trước đó trong tình huống mất tài nguyên hoặc dữ liệu; đặc biệt quan trọng đối với các hệ thống giao dịch như ngân hàng trực tuyến...

Nhìn từ quan điểm người dùng, các cấp độ kiểm thử hệ thống rất quan trọng vì nó giúp đưa sản phẩm ra hoạt động trên môi trường thực tế.

**Chú ý:** Không nhất thiết phải thực hiện tất cả các loại kiểm thử nêu trên. Tùy yêu cầu và đặc trưng của từng hệ thống, tùy khả năng và thời gian cho phép của dự án, khi lập kế hoạch, người quản lý dự án sẽ quyết định áp dụng những loại kiểm thử nào.

## ACCEPTANCE TEST - KIỂM THỬ CHẤP NHẬN SẢN PHẨM

Giai đoạn này thực hiện sau khi kiểm thử hệ thống hoàn thành, quy trình kiểm thử chấp nhận sản phẩm thường là do người dùng đánh giá hoặc giao riêng cho một nhóm dùng thử đánh giá độc lập. Mục đích của việc kiểm thử này là đưa ra các chứng minh phần mềm thỏa mãn yêu cầu của người dùng và được chấp nhận bởi họ. Kiểm tra chấp nhận sản phẩm có ý nghĩa hết sức quan trọng, mặc dù trong hầu hết mọi trường hợp, các phép kiểm thử của System Test và Acceptance Test gần như giống nhau, nhưng về bản chất và cách thực hiện hoàn toàn khác. Đối với những sản phẩm dành để bán rộng rãi trên thị trường cho nhiều người sử dụng, thông thường sẽ thông qua hai loại kiểm thử gọi là Alpha Test và Beta Test. Với Alpha Test, người dùng (user) kiểm thử phần mềm ngay tại nơi xây dựng và phát triển phần mềm, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa chữa. Với Beta Test, phần mềm sẽ được gửi cho người dùng (user) để kiểm thử ngay trong môi trường thực tiễn, và nhận lại các phản hồi, để các lập trình viên tổ chức sửa chữa.

Trên thực tế cho thấy, nếu người dùng tham gia đánh giá sản phẩm thì quá trình phát triển chúng sẽ gặp nhiều sai lệch và không đánh giá chung về sự đáp ứng của phần mềm. Sự sai lệch này ảnh hưởng rất lớn đến việc hiểu các yêu cầu và nhu cầu của người dùng.

## 2. MỘT SỐ CẤP ĐỘ KIỂM THỬ KHÁC

Ngoài các cấp độ trên, còn một số cấp độ kiểm thử khác như:

### KIỂM THỬ HỒI QUY – REGRESSION TESTING:

Kiểm thử hồi quy là sự kiểm thử có lựa chọn của hệ thống hay thành phần hệ thống để xác định sự thay đổi không gây ra các lỗi không mong muốn (Theo chuẩn IEEE610.12-90). Quá trình kiểm thử này được thực hiện cho các phần mềm đã qua các cuộc kiểm thử trước đó.

Sự lặp lại kiểm thử nhằm mục đích xác định sự hoạt động của phần mềm không bị thay đổi. Hiển nhiên là sự thỏa hiệp phải được thực hiện giữa sự đảm bảo được đưa ra bởi kiểm thử hồi quy mỗi lần thực hiện một sự thay đổi và những tài nguyên được yêu cầu thực hiện điều đó.

### KIỂM THỬ TÍNH ĐÚNG – CORRECTNESS TESTING:

Kiểm thử tính đúng đáp ứng yêu cầu tối thiểu về tính đúng của phần mềm. Kiểm thử viên có thể biết hoặc không biết đến các chi tiết bên trong của module phần mềm. Nó sẽ tìm đến kiểm thử đáng tin cậy và chỉ ra các hoạt động đúng từ các hoạt động sai.

### CÁC PHƯƠNG PHÁP KIỂM THỬ CON NGƯỜI

Tồn tại hai phương pháp kiểm thử con người chủ yếu là: **CODE INSPECTIONS, WALKTHROUGHS**.

Áp dụng cho một nhóm người đọc và kiểm thử theo mã lệnh của chương trình, thực hiện mục tiêu là tìm ra lỗi mà không gỡ lỗi. Chúng là sự cải tiến của phương pháp kiểm thử mà lập trình viên đọc chương trình của họ trước khi kiểm thử nó. Inspections và Walkthroughs hiệu quả hơn là bởi vì những người khác sẽ kiểm thử chương trình tốt hơn chính tác giả của chương trình đó.

Inspections/Walkthroughs và kiểm thử bằng máy tính thực hiện bổ sung cho nhau. Hiệu quả tìm lỗi sẽ kém đi nếu thiếu đi 1 trong 2 phương pháp. Và đối với việc sửa đổi chương trình cũng nên sử dụng các phương pháp kiểm thử này cũng như các kỹ thuật kiểm thử hối quy.

### **TỔNG DUYỆT (WALKTHROUGH)**

Walkthrough là một thuật ngữ mô tả sự xem xét kỹ lưỡng của một quá trình ở mức trừu tượng trong đó nhà thiết kế hay lập trình viên lãnh đạo các thành viên trong nhóm và những người có quan tâm khác thông qua một sản phẩm phần mềm. Ngoài ra, còn có những ai tham gia đặt câu hỏi, và ghi chú những lỗi có thể có, sự vi phạm các chuẩn phát triển và các vấn đề khác.

Walkthrough mã lệnh là 1 tập các thủ tục và các công nghệ dò lỗi cho việc đọc nhóm mã lệnh. Trong một Walkthrough, nhóm các nhà phát triển có khoảng 3 hoặc 4 thành viên là tốt nhất, thực hiện xét duyệt lại. Chỉ 1 trong các thành viên là tác giả của chương trình.

Một ưu điểm khác của walkthroughs là giảm chi phí gỡ lỗi, đây là một thực tế mà khi một lỗi được tìm thấy, nó thường được định vị chính xác trong mã lệnh. Thêm vào đó, phương pháp này thường tìm ra 1 tập các lỗi, cho phép sau đó các lỗi đó được sửa tất cả với nhau. Mặt khác, kiểm thử dựa trên máy tính, chỉ tìm ra triệu chứng của lỗi (chương trình không kết thúc hoặc đưa ra kết quả vô nghĩa), và các lỗi thường được tìm ra và sửa lần lượt từng lỗi một.

### **THANH TRA MÃ NGUỒN (CODE INSPECTION)**

Thanh tra mã nguồn là 1 tập hợp các thủ tục và các kỹ thuật dò lỗi cho việc đọc các nhóm mã lệnh. Một nhóm kiểm duyệt thường gồm 4 người.

Một trong số đó đóng vai trò là người điều tiết, một lập trình viên lão luyện và không được là tác giả của chương trình và phải không quen với các chi tiết của chương trình. Người điều tiết có nhiệm vụ: phân phối nguyên liệu và lập lịch cho các buổi kiểm duyệt, chỉ đạo phiên làm việc, ghi lại tất cả các lỗi được tìm thấy và đảm bảo là các lỗi sau đó được sửa. Thành viên thứ hai là một lập trình viên. Các thành viên còn lại trong nhóm thường là nhà thiết kế của chương trình (nếu nhà thiết kế khác lập trình viên) và một chuyên viên kiểm thử.

## CHƯƠNG 5

# CÁC KỸ THUẬT KIỂM THỬ

### 1. KỸ THUẬT KIỂM THỬ TÍNH (STATIC TESTING)

Kỹ thuật kiểm thử tĩnh được thực hiện bằng tay và sử dụng phần kiểm thử logic các chi tiết mà không vận hành phần mềm. Kỹ thuật được thực hiện bởi chuyên viên thiết kế viết mã đơn (thực hiện độc lập) và cần phải lọc các yêu cầu của phần mềm.

Kỹ thuật này có thể thực hiện tự động hóa, tự kiểm thử toàn bộ phần mềm thông qua trình biên dịch để xác định tính hợp lệ của cú pháp bên trong mã phần mềm.

### 2. KỸ THUẬT KIỂM THỬ ĐỘNG (DYNAMIC TESTING)

Kỹ thuật kiểm thử động được thực hiện bằng cách vận hành phần mềm trên máy tính và thực hiện kiểm thử tác động của nó. Đó là quá trình kiểm thử dựa trên các lần kiểm thử xác định bằng sự thực thi của đối tượng kiểm thử hay chạy phần mềm.

Kỹ thuật này kiểm thử các hoạt động của mã lệnh khi vận hành. Tương quan như kiểm thử vật lý hệ thống tới các biến, biến đổi theo thời gian. Trong tiến trình kiểm thử động, phần mềm phải được biên dịch và vận hành trên máy. Kiểm thử động thực sự bao gồm làm việc với phần mềm, nhập các giá trị đầu vào và kiểm thử xem liệu đầu ra có đạt yêu cầu hay không. (Các giai đoạn kiểm thử động gồm có kiểm thử Unit – Unit Tests, Kiểm thử tích hợp – Intergration Tests, Kiểm thử hệ thống – System Tests, và Kiểm thử chấp nhận sản phẩm – Acceptance Tests).

### 3. KỸ THUẬT KIỂM THỬ HỘP TRẮNG (WHITE BOX TESTING)

#### Mục đích

Quá trình kiểm thử hộp trắng dùng để thực hiện bước kiểm thử tính logic của cấu trúc bên trong phần mềm. Các chuyên viên kiểm thử sẽ truy cập vào cấu trúc dữ liệu và giải thuật bên trong của phần mềm để thực hiện khảo sát kiểm thử. Mục đích của bất cứ phương pháp kiểm thử an ninh nào cũng là để đảm bảo tình trạng tốt của một hệ thống trên bề mặt của những sự tấn công độc hại hoặc là những sự thất bại bởi phần mềm thông thường. Kiểm thử hộp trắng thực hiện dựa vào sự hiểu biết về những phần tử bên trong của một hệ thống. Nó bao gồm phân tích dòng dữ liệu, điều khiển dòng, dòng thông tin, mã thực hành, các ngoại lệ và những lỗi trình bày trong hệ thống để kiểm thử những hoạt động của phần mềm không được định trước.

Kiểm thử hộp trắng có thể thực hiện để xác định tính hợp lệ cho dù việc triển khai mã nguồn thực hiện sau đó nhằm xác nhận tính hợp lệ để triển khai thực hiện chức năng bảo mật và giảm rủi ro từ việc bị tấn công ở những chỗ có thể khai thác được.

Nó yêu cầu phải truy nhập vào mã nguồn mặc dù kiểm thử hộp trắng có thể thực hiện tại bất cứ thời điểm nào trong vòng đời của phần mềm sau khi mã được phát triển. Đó là một quá trình tốt để thực hiện kiểm thử hộp trắng trong suốt giai đoạn kiểm thử đơn vị.

### Các phương pháp kiểm thử hộp trắng

- Kiểm thử giao diện lập trình ứng dụng - API testing (Application Programming Interface): là phương pháp kiểm thử những ứng dụng sử dụng các API công khai và riêng tư.
- Bao phủ mã lệnh - Code coverage: tạo các kiểm thử để đáp ứng một số tiêu chuẩn về bao phủ mã lệnh.
- Các phương pháp gán lỗi - Fault Injection.
- Các phương pháp kiểm thử hoán chuyển - Mutation testing methods.
- Kiểm thử tĩnh – Static testing: kiểm thử hộp trắng bao gồm mọi kiểm thử tĩnh.

Kiểm thử hộp trắng dùng để đánh giá sự hoàn thành của một quá trình kiểm thử. Điều này cho phép nhóm phần mềm khảo sát các chức năng quan trọng của 1 phần mềm thường dễ bị bỏ sót trong quá trình kiểm thử.

Những thông tin liên quan đến kiểm thử hộp trắng bao gồm: Mã nguồn, bản phân tích các rủi ro, đặc tả an toàn, tài liệu thiết kế và các tài liệu liên quan để đảm bảo chất lượng phần mềm.

#### Chi tiết các thành phần:

- Mã nguồn: là thành phần quan trọng nhất để thực hiện kiểm thử hộp trắng. Không truy cập vào mã nguồn thì không thể thực hiện được quá trình kiểm thử.
- Thiết kế kiến trúc và phân tích rủi ro: được dùng làm chỉ dẫn cho tất cả những hoạt động có liên quan đến kiểm thử hộp trắng, bao gồm kế hoạch kiểm thử, tạo các ca kiểm thử, lựa chọn dữ liệu kiểm thử, lựa chọn kỹ thuật kiểm thử và lựa chọn tiêu chuẩn dùng kiểm thử.
- Đặc tả an toàn hay yêu cầu là phải có để hiểu và kích hoạt các tính năng bảo mật của việc thử nghiệm bên dưới phần mềm.
- Tài liệu thiết kế: là thành phần chính yếu để cải thiện việc hiểu phần mềm và để phát triển một cách có hiệu quả các lần kiểm thử và thông qua những quyết định thiết kế và giả định.

- Các tài liệu đảm bảo chất lượng: để tìm hiểu về chất lượng phần mềm một cách chi tiết và dựa vào đó dự định các chức năng của nó. Tài liệu đảm bảo chất lượng nên bao gồm một chiến lược kiểm thử, một bản kế hoạch kiểm thử, và báo cáo những khuyết điểm. Kiểm thử sự thực hiện và vận hành là việc quan trọng để hiểu những ràng buộc của hệ thống và hoạt động của hệ thống.

### Rủi ro và chiến lược kiểm thử hộp trắng

Việc kiểm thử đòi hỏi chuyên viên thực hiện phải có sự phân tích hợp lý các quá trình xây dựng phần mềm. Phải có khả năng nhận diện lỗi và đảm bảo sản phẩm đầu ra đạt yêu cầu và chất lượng. Các kế hoạch kiểm thử thực hiện đầy đủ để loại bỏ các rủi ro bất ngờ trong khi thực hiện kiểm thử.

Bước đầu tiên của một quy trình kiểm thử hộp trắng là xác định và phát triển một kế hoạch chiến lược dựa trên bảng phân tích rủi ro. Mục đích của việc này là khoanh vùng và sàng lọc các hoạt động chính phức tạp, các quyết định và các thử thách trong kiểm thử. Cần phải nhận biết phạm vi kiểm thử, kỹ thuật áp dụng kiểm thử, môi trường, yêu cầu kỹ năng và ngôn ngữ thông tin và hoạt động kiểm thử theo nhóm. Các chiến lược cần có tính xác thực về thời gian và kinh phí, các ràng buộc cũng như những điểm cấm của từng thành phần của hệ thống phần mềm. Vì thế, phải cân bằng hiệu lực kiểm thử với hiệu quả kiểm thử dựa trên phân tích rủi ro hệ thống. Mức độ của hiệu lực tất nhiên phụ thuộc vào mục đích của phần mềm và kết quả những thất bại của nó, chi phí cho sự thất bại của phần mềm càng cao thì sự phức tạp và khắt khe của phương pháp kiểm thử phải đảm bảo hiệu lực. Chiến lược kiểm thử về cơ bản là quản lý hoạt động và người quản lý kiểm thử chịu trách nhiệm cho phát triển và quản lý một chiến lược kiểm thử.

### Lập kế hoạch kiểm thử

Trước tiên phải liệt kê các chiến lược kiểm thử và tổ chức kế hoạch cho tiến trình kiểm thử sau đó. Nó bao gồm những vùng kiểm thử khép kín và lựa chọn, xác nhận dữ liệu mang tính hợp lệ cho kết quả. Kế hoạch kiểm thử rất có ích cho việc quản trị, lập kế hoạch và báo cáo. Càng mô tả chi tiết thì lần kiểm thử sau diễn ra càng nhanh chóng hơn.

### Lập các quy trình kiểm thử

Việc lập ra các quy trình kiểm thử bao gồm những điều kiện: điều kiện mang tính quyết định, đặc điểm chung và riêng của đầu vào kiểm thử, từng bước thực hiện và kết quả.. Có rất nhiều định nghĩa và các định dạng trong việc mô tả các quy trình kiểm thử. Mục đích của các quy trình kiểm thử là để nắm bắt những gì cụ thể mà quy trình muốn đạt tới. Phân tích rủi ro, chiến lược kiểm thử, và kế hoạch kiểm thử nên sử dụng làm hướng dẫn cho sự phát triển các ca kiểm thử.

## Môi trường kiểm thử

Kiểm thử yêu cầu sự tồn tại của một môi trường kiểm thử. Phần cứng và phần mềm (gồm cả công cụ, ngôn ngữ, phần mềm lõi giữa, ...) đáp ứng yêu cầu người dùng được lựa chọn.

## Thực hiện kiểm thử

Thực hiện kiểm thử liên quan đến việc chạy các quy trình kiểm thử đã được phát triển cho hệ thống và báo cáo các kết quả kiểm thử.

## 4. KỸ THUẬT KIỂM THỬ HỘP ĐEN (BLACK BOX)

### Mục đích

Một trong những chiến lược kiểm thử quan trọng là kiểm thử hộp đen, kiểm thử hộp đen xem phần mềm như là một "hộp đen" với các hướng dữ liệu vào hoặc ra.

Mục đích của chuyên viên là hoàn toàn không quan tâm về cách cư xử và cấu trúc bên trong của chương trình. Thay vào đó, tập trung vào tìm các trường hợp mà phần mềm không thực hiện theo các đặc tả của nó. Theo hướng tiếp cận này, dữ liệu kiểm thử được lấy chỉ từ các đặc tả.

### Các phương pháp kiểm thử hộp đen

- Phân lớp tương đương – Equivalence partitioning.
- Phân tích giá trị biên – Boundary value analysis.
- Kiểm thử mọi cặp – All-pairs testing.
- Kiểm thử fuzz – Fuzz testing.
- Kiểm thử dựa trên mô hình – Model-based testing.
- Ma trận dấu vết – Traceability matrix.
- Kiểm thử thăm dò – Exploratory testing.
- Kiểm thử dựa trên đặc tả – Specification-base testing.

Kiểm thử dựa trên đặc tả tập trung vào kiểm thử tính thiết thực của phần mềm theo những yêu cầu thích hợp. Do đó, kiểm thử viên nhập dữ liệu vào, và chỉ thấy dữ liệu ra từ đối tượng kiểm thử.

Mức kiểm thử này thường yêu cầu các ca kiểm thử cung cấp thông tin cho kiểm thử viên để có thể xác minh là đối với dữ liệu đầu vào đã cho, giá trị đầu ra (hay cách thức hoạt động) có giống với giá trị mong muốn đã được xác định trong ca kiểm thử đó hay không.

Kiểm thử dựa trên đặc tả là cần thiết, nhưng không đủ để ngăn chặn những rủi ro chắc chắn.

### **Ưu và nhược điểm**

Kiểm thử hộp đen không có mối liên quan nào tới mã lệnh, công việc kiểm thử viên rất đơn giản với quan niệm là một mã lệnh phải có lỗi.

Công việc của kiểm thử viên là tìm ra những lỗi mà những lập trình viên đã không thể tìm ra, và họ không cần biết đến quá trình xây dựng của phần mềm. Đó là lý do nhiều cuộc kiểm thử được lặp lại trong quá trình kiểm thử hộp đen, mà không thực hiện bằng một cuộc kiểm thử duy nhất và một số phần của phần mềm không được kiểm thử gì cả.

**Ưu điểm** của kiểm thử hộp đen là đánh giá khách quan mọi lỗi trong phần mềm. **Nhược điểm** của nó là thăm dò mù, không có định hướng trước về phần mềm.

Kiểm thử tập trung vào các yêu cầu chức năng của phần mềm, đồng thời tạo ra các điều kiện đầu vào, để kiểm thử tất cả các chức năng phần mềm. Về bản chất kiểm thử hộp đen có sự trái ngược hoàn toàn so với kiểm thử hộp trắng. Nó có khả năng bổ sung cho phương pháp kiểm thử hộp trắng, và có thể phát hiện các lỗi khác nhau nhiều hơn.

#### **Kiểm thử hộp đen phát hiện các loại lỗi như sau:**

- Không đúng hay mất một số hàm/module
- Giao diện không phù hợp/ lỗi về giao diện (Interface).
- Lỗi về cấu trúc dữ liệu hay thao tác lên dữ liệu bên ngoài.
- Lỗi thực thi.
- Lỗi về khởi động và hủy dữ liệu, biến.

Kiểm thử hộp đen thực hiện vào giai đoạn sau của quá trình kiểm thử phần mềm. Mục đích của nó là tập trung trên phần thông tin, không tập trung trên vùng mã chương trình. Các trường hợp kiểm thử để trả lời các câu hỏi như sau:

- Như thế nào là hàm/chức năng hợp lệ?
- Lớp gì của thông tin đầu vào sẽ tạo ra những trường hợp kiểm thử tốt?
- Hệ thống có khả năng bị tổn thương với một giá trị nhập vào nào đó không?
- Ranh giới của các vùng dữ liệu có độc lập với nhau hay không?
- Tỷ lệ và kích thước dữ liệu mà hệ thống có thể chịu là bao nhiêu?

## 5. KIỂM THỬ HỘP XÁM (GRAY BOX TESTING)

Kiểm thử hộp xám đòi hỏi phải truy cập vào cấu trúc dữ liệu và giải thuật bên trong của phần mềm.

Từ đó xác định các mục đích xây dựng kiểm thử. Việc thao tác với dữ liệu đầu vào và định dạng dữ liệu đầu ra là không rõ ràng, giống như một chiếc "hộp xám", bởi vì đầu vào và đầu ra rõ ràng là ở bên ngoài "hộp đen" mà chúng ta vẫn gọi về hệ thống được kiểm thử.

Sự khác biệt này đặc biệt quan trọng khi quản lý kiểm thử tích hợp – Intergration testing giữa 2 module mã lệnh được viết bởi các chuyên viên thiết kế khác nhau.

Trong đó, chỉ có giao diện là được đưa ra để kiểm thử. Kiểm thử hộp xám có thể cũng bao gồm cả thiết kế đối chiếu để quyết định, giá trị biên hay thông báo lỗi.

## CHƯƠNG 8

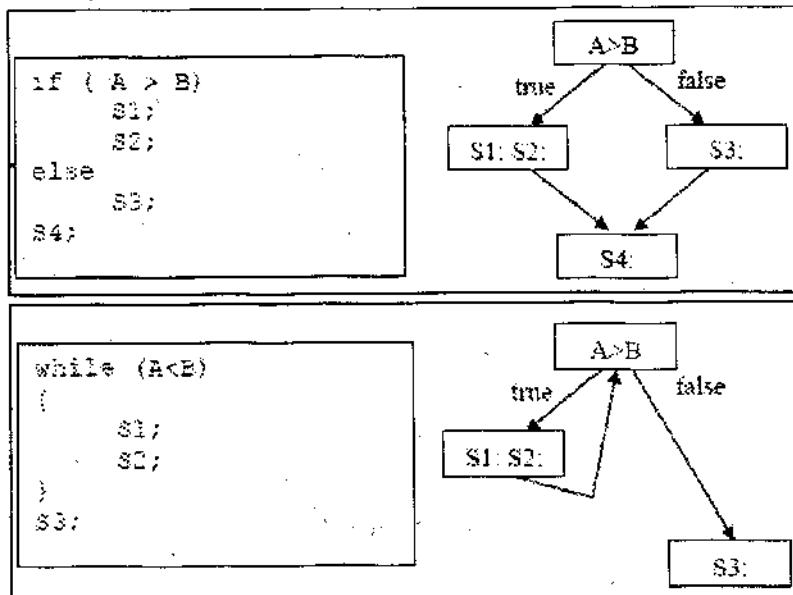
# KỸ THUẬT KIỂM THỬ HỘP TRẮNG

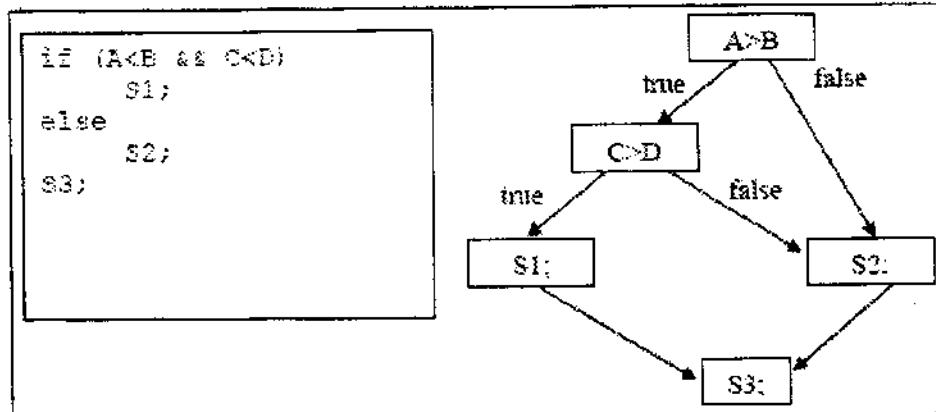
## CÁC KỸ THUẬT KIỂM TRA

### KIỂM THỬ ĐƯỜNG CƠ BẢN (BASIC PATH TESTING)

Kiểm thử hộp trắng là một kỹ thuật được Tom McCabe đưa ra, mức kiểm thử này là cơ sở cho việc xác định độ phức tạp và logic trong thiết kế quy trình kiểm thử. Những trường hợp kiểm thử được suy diễn để thực hiện tập cơ sở, các trường hợp kiểm thử đó được đảm bảo để thực hiện mỗi lệnh trong chương trình ít nhất một lần trong quá trình kiểm thử. Đồ thị lưu trình khá giống đồ thị luồng điều khiển của chương trình. Nó nhận được từ đồ thị luồng bằng cách gộp các lệnh thứ tự và thay thế lệnh rẽ nhánh, cũng như các điểm kết thúc của các đường điều khiển. Kỹ thuật luồng điều khiển là một chiến lược kiểm thử có cấu trúc sử dụng luồng điều khiển chương trình như một mô hình. Nó dựa trên sự lựa chọn cẩn thận một bộ những đường kiểm thử thông qua chương trình. Kỹ thuật đường cơ bản, đồ thị lưu trình có thể giúp những người thiết kế kiểm thử nhận biết độ phức tạp logic của một thủ tục và sử dụng độ phức tạp này như một hướng dẫn hạn chế thực hiện một bộ các đường cơ bản.

Phương pháp kiểm tra bao trùm mọi đường dẫn của chương trình và cần kết hợp với lược đồ tiến trình.



**LƯU ĐỒ**

Trên thực tế, phương pháp đường dẫn cơ sở có thể được dùng mà không cần sử dụng lưu đồ. Tuy nhiên, lưu đồ thì là một công cụ hữu ích để hiểu quá trình điều khiển và minh họa phương pháp tiếp cận. Phần này sẽ trình bày một số ký hiệu đơn giản của quá trình điều khiển, được gọi là lưu đồ hay đồ thị lưu trình.

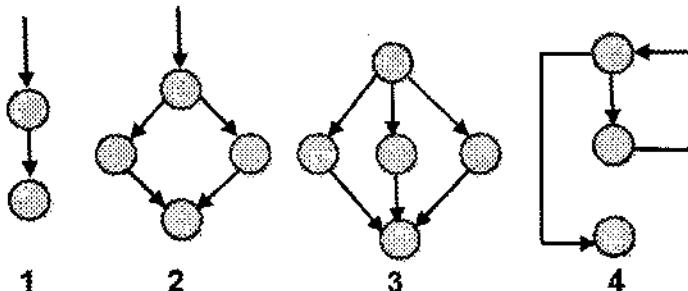
Mỗi cấu trúc điều khiển có một ký hiệu đồ thị tương ứng. Cấu trúc của lưu đồ bao gồm các nút hình tròn biểu thị một hay một số lệnh tuần tự, hoặc thay cho điểm hội tụ các đường điều khiển.

Mỗi cạnh nối hai nút biểu diễn dòng điều khiển. Đồ thị dòng chia mặt phẳng thành nhiều miền có nút biểu thị sự phân nhánh.

**Các kiểu cấu trúc thành phần đồ thị dòng**

- > Mỗi hình tròn gọi là đỉnh, biểu diễn một hoặc nhiều lệnh thủ tục.
- > Con trỏ được gọi là cung hoặc liên kết biểu diễn quá trình điều khiển. Một cung cần phải kết thúc tại một đỉnh.
- > Mỗi đỉnh có chứa điều kiện gọi là đỉnh điều kiện.
- > Phần được bao bởi các cung và các đỉnh gọi là vùng.

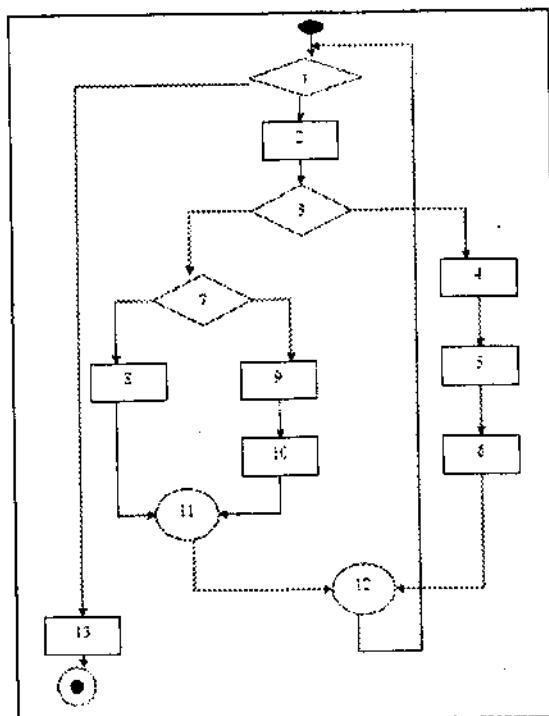
Lưu đồ trình bày quá trình điều khiển logic sử dụng một số ký hiệu được minh họa như hình dưới.



1. Tuần tự
  2. Rẽ nhánh
  3. Lựa chọn
  4. Vòng lặp while

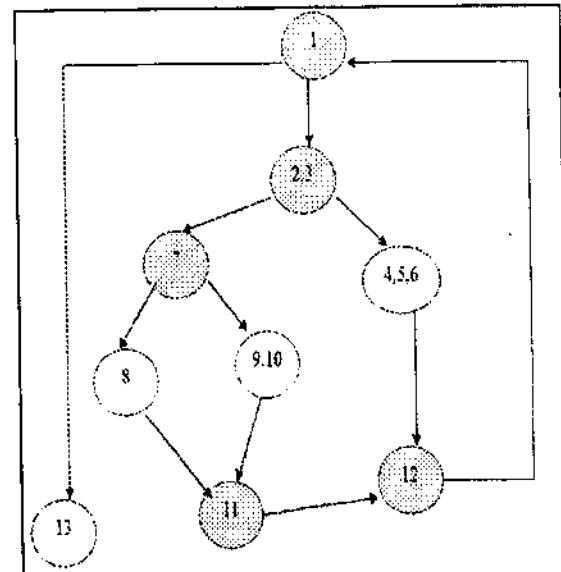
### Ví dụ về lưu đồ

Ta có 1 lưu đồ điều khiển của chương trình như hình bên.



Theo quy tắc chuyển đổi ta có:

- Gom các lệnh tuần tự ta có các lệnh số: (2,3); (4,5,6) và (9,10).
  - Các điểm rẽ nhánh và kết thúc của các đường điều khiển: 1, 3, 7, 11 và điểm 12.



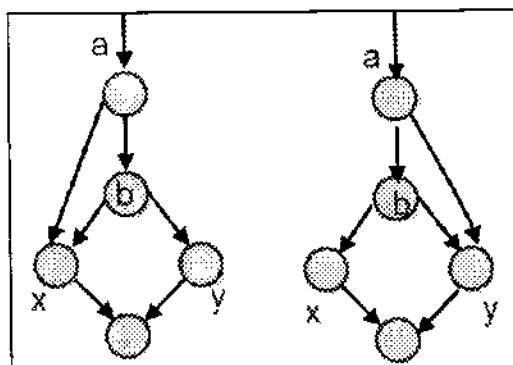
Đồ thị gồm có: 9 nút, trong đó các nút là: 1, (2,3), 7, 11 và 12, 11 cung chia mặt phẳng thành 4 miền.

### Biểu diễn các điều kiện phức trong đồ thị lưu trình

Khi gặp các điều kiện phức xuất hiện trong câu lệnh điều kiện được biểu diễn gồm một hoặc nhiều phép toán logic (AND, OR, NOT), cần phải được chia thành các điều kiện đơn trong thực hiện kiểm thử đường dẫn cơ sở. Mỗi đỉnh chứa điều kiện được gọi là đỉnh điều kiện và được đặc trưng bởi hai hoặc nhiều cạnh bắt nguồn từ nó.

Ví dụ: if (a OR b) then {Procedure x } else {Procedure y}

If (c AND d) then {Procedure x} else {Procedure y}



### Độ phức tạp Cyclomatic

Độ phức tạp cyclomatic được dùng để đo lường độ phức tạp của phần mềm. Khi được sử dụng trong môi trường của phương pháp đường cơ bản thì các giá trị được xác định có độ phức tạp cyclomatic. Các câu lệnh được kiểm thử bắt buộc ít nhất một lần.

Đồng thời, một đường dẫn độc lập là một đường dẫn bất kỳ trong chương trình đưa ra ít nhất một tập lệnh xử lý hoặc điều kiện mới. Để đảm bảo rằng mọi câu lệnh được thực hiện ít nhất một lần, cần tìm được tất cả các đường điều khiển độc lập trong chương trình (khác với các đường khác ít nhất một lệnh).

Số các đường độc lập trong một chương trình là giới hạn trên số các kiểm thử cần được tiến hành. Nó được gọi là độ phức tạp chu trình của chương trình. Các đường độc lập của một chương trình trùng với các đường độc lập của đồ thị dòng.

Một đường dẫn độc lập là một chuỗi các chỉ lệnh mà bắt đầu từ cổng vào của một thủ tục (phương thức) và kết thúc ở cổng ra của nó.

Một đường dẫn độc lập là một đường đưa vào ít nhất một lệnh hoặc một điều kiện mới hoặc đưa vào ít nhất một cạnh mới trong đồ thị dòng.

Một đường dẫn độc lập phải di chuyển tiến lên ít nhất một cạnh mà chưa được di ngang qua trước khi đường dẫn được định nghĩa. Việc tính toán độ phức tạp Cyclomat sẽ cho biết có bao nhiêu đường dẫn cần tìm.

Cho lưu đồ G, độ phức tạp Cyclomat  $V(G)$  được tính bằng ba cách như sau:

1.  $V(G) = R$ , trong đó  $R$  là số vùng của lưu đồ.
2.  $V(G) = P + 1$ , trong đó  $P$  là số đỉnh điều kiện (nút vị từ) có trong lưu đồ G.
3.  $V(G) = E - N + 2$ , trong đó  $E$  là số cạnh của đồ thị lưu trình,  $N$  là số đỉnh (nút vị từ) của lưu đồ G.

**Ví dụ:** với lưu đồ trên ta có độ phức tạp là 4 vậy sẽ có 4 con đường độc lập gồm:

- Đường dẫn: 1, 13
- Đường dẫn: 1, 2, 3, 4, 5, 6, 12, 1, 13
- Đường dẫn: 1, 1, 3, 7, 8, 11, 12, 1, 13
- Đường dẫn: 1, 2, 3, 7, 9, 10, 11, 12, 1, 13

#### Xác định các ca thử nghiệm:

- **B1:** từ một thiết kế hoặc mã vẽ một đồ thị dòng G tương ứng.
- **B2:** xác định độ phức tạp của chu trình  $V(G)$  tương ứng của nó.
- **B3:** xác định tập cơ bản các con đường độc lập.
- **B4:** chuẩn bị các ca kiểm thử cho mỗi con đường trong tập các con đường đó.
- **B5:** chạy các ca kiểm thử và kiểm tra kết quả của chúng.

#### Ma trận kiểm thử

Ma trận kiểm thử là một ma trận vuông có kích thước bằng số các nút trong đồ thị dòng. Mỗi dòng/cột tương ứng với tên một nút. Mỗi ô là tên một cung nối nút dòng với nút cột. Nhân liên tiếp k ma trận này được ma trận chỉ số con đường k cũng từ nút dòng tới nút cột. Ma trận kiểm thử được sử dụng như một dữ liệu có cấu trúc để kiểm tra các con đường cơ bản. Khi kiểm thử ta nên thêm trọng số cho các cung của ma trận kiểm thử như hình.

	1	23	456	7	8	910	11	12	13
1	1								1
23		1	1						
456									1
7					1	1			
8							1		
910							1		
11								1	
12	1								
13									

- ❖ Xác suất cung đó được thực thi.
- ❖ Thời gian xử lý của tiến trình đi qua cung đó.
- ❖ Bộ nhớ dài hồi của tiến trình đi qua cung đó.
- ❖ Nguồn lực dài hồi của tiến trình đi qua cung đó.

Xét ví dụ với đồ thị dòng như trên ta có ma trận sau:

Kiểm thử điều kiện với mục đích sử dụng tất cả những điều kiện logic trong mô-dun chương trình.

#### Có thể xác định:

- ❖ Biểu thức quan hệ (E1, E2): khi E1 và E2 là các biểu thức số học.
- ❖ Điều kiện đơn: biến Boolean hoặc biểu thức quan hệ có thể được xếp thứ tự bởi một toán tử điều hành NOT.
- ❖ Điều kiện phức hợp: gồm có hai hoặc nhiều hơn những điều kiện đơn toán tử Boolean và những dấu ngoặc đơn.
- ❖ Biểu thức Boolean: điều kiện không có biểu thức quan hệ.

#### Những kiểu lỗi trong điều kiện logic kiểm thử:

- ❖ Sai biến Bool.
- ❖ Sai toán tử Bool.
- ❖ Sai số hạng trong biểu thức toán tử Bool.
- ❖ Sai toán tử quan hệ.
- ❖ Sai biểu thức số học.

Phương pháp kiểm thử điều kiện tập trung vào kiểm thử mỗi điều kiện đơn trong chương trình. Một vài chiến lược kiểm thử điều kiện đó là kiểm thử phân nhánh và kiểm thử miền.

**Chiến lược kiểm thử phân nhánh:** là chiến lược kiểm thử từng điều kiện trong chương trình. Với mỗi điều kiện phức hợp C thì với mỗi nhánh true và false của C, mỗi điều kiện đơn trong C phải được thực hiện ít nhất một lần.

**Chiến lược kiểm thử miền:** đòi hỏi cần 3 hoặc 4 kiểm thử cho một biểu thức quan hệ. (<; >; =; <>). Nếu biểu thức Bool có n biến mà n nhỏ thì dễ thực hiện, nếu n lớn sẽ khó thực hiện.

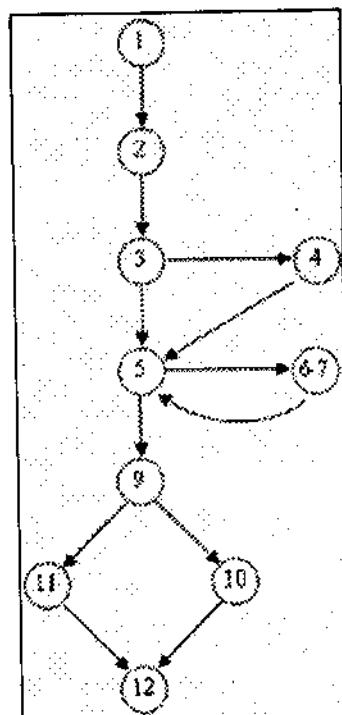
### KIỂM THỬ CẤU TRÚC ĐIỀU KHIỂN (CONTROL - FLOW TESTING)

#### 1. KIỂM THỬ THEO CÂU LỆNH

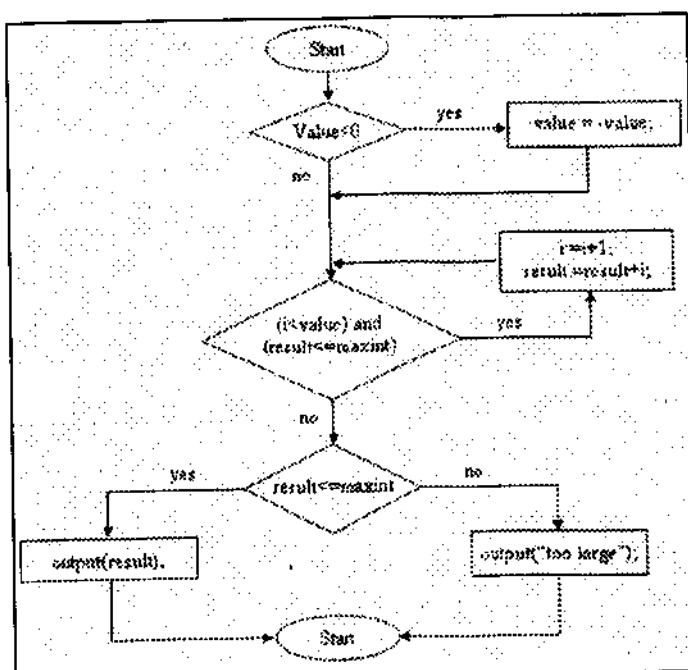
Thực hiện quá trình kiểm tra bao đảm cho mỗi câu lệnh của chương trình được thực hiện ít nhất một lần. Phương pháp kiểm tra này xuất phát từ ý tưởng: một câu lệnh cần được thực hiện nếu không thể biết được có lỗi xảy ra trong câu lệnh đó hay không. Nhưng việc kiểm tra với một giá trị đầu vào không đảm bảo là sẽ đúng cho mọi trường hợp.

Ví dụ: đoạn chương trình thực hiện tính như sau:

1. result = 0+1+...+lvalue;
2. nếu result <= maxint, báo lỗi trong trường hợp ngược lại.
3. PROGRAM maxsum (maxint, value : INT)
4. INT result := 0 ; l := 0 ;
5. IF value < 0
6. THEN value := - value ;
7. WHILE (l < value) AND (result <= maxint)
8. DO l := l + 1 ;
9. result := result + l ;
10. OD;
11. IF result <= maxint
12. THEN OUTPUT (result)
13. ELSE OUTPUT ("too large")
14. END.

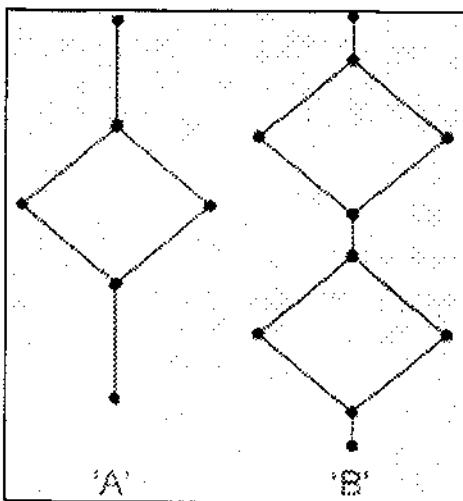


Ví dụ: với các bộ giá trị input: Maxint = 10, Value = -1 hay Maxint = 0, Value = -1

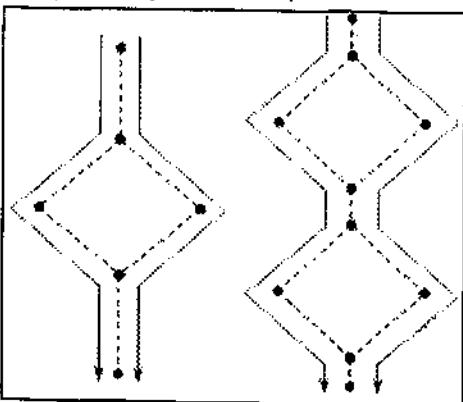


### Đánh giá độ phức tạp của của phương pháp kiểm tra câu lệnh

Tham khảo hình ví dụ sau: có 2 trường hợp A và B, quan sát và nhận xét chúng.



Tuy nhiên, ta thấy số lần kiểm tra tối thiểu để có thể kiểm tra toàn bộ các câu lệnh như trên cho cả 2 hàm đều là 2. Vì vậy, phương pháp này không tương ứng với sự phức tạp của mã lệnh.



## 2. KIỂM THỬ BIỂU THỨC ĐIỀU KIỆN

Là phương pháp kiểm thử trên những điều kiện logic của hàm hay module. Một điều kiện đơn giản là một biến Boolean hoặc là một biểu thức quan hệ:

**Điều kiện đơn:** là một biến logic hoặc một biểu thức quan hệ, có thể có toán tử NOT (!) đứng trước.

- X hay Not X một điều kiện logic đơn giản.

- Biểu thức quan hệ thường có dạng: E1 <phép toán quan hệ> E2

E1, E2 là các biểu thức số học và phép toán quan hệ là một trong các phép toán sau : <, <=, ==, !=, > hay >=.

Điều kiện phức: gồm hai hay nhiều điều kiện đơn, toán tử logic AND (&&) hoặc OR (||) hoặc NOT (!) và các dấu ngoặc đơn '(' và ')', ví dụ, (a > b + 1) AND (a <= max).

Vì vậy, các thành phần trong một điều kiện có thể gồm phép toán logic, biến logic, cặp dấu ngoặc logic (bao một điều kiện đơn hoặc phức), phép toán quan hệ, hoặc biểu thức toán học.

Một điều kiện kết hợp của 2 hay nhiều điều kiện đơn giản, các phép toán Boolean: OR (||), AND (&) and NOT (!)

#### Các loại lỗi của điều kiện bao gồm:

- Lỗi trong các thao tác luận lý (lỗi tồn tại một biểu thức không đúng, thiếu hoặc thừa các thao tác luận lý).
- Lỗi do giá trị của biến luận lý.
- Lỗi do dấu ngoặc.
- Lỗi do phép toán quan hệ.
- Lỗi trong biểu thức toán học.

Mục đích của kiểm thử cấu trúc điều kiện là phát hiện không chỉ lỗi trong điều kiện mà còn những lỗi khác trong phần mềm. Nếu một tập kiểm thử cho một chương trình P là hiệu quả cho việc phát hiện lỗi trong điều kiện của P, thì bộ kiểm thử đó cũng có thể phát hiện các lỗi khác trong P.

#### E1 <phép toán quan hệ> E2

Ba trường hợp kiểm thử được yêu cầu để kiểm tra là giá trị E1 lớn hơn, nhỏ hơn và bằng giá trị của E2.

Nếu <phép toán quan hệ> là không đúng và E1, E2 là đúng thì 3 loại kiểm thử trên đảm bảo có thể xác định được lỗi trong phép toán quan hệ. Để phát hiện lỗi trong E1 và E2 thì các trường hợp kiểm thử E1 lớn hơn, nhỏ hơn E2 có thể phát hiện ra được lỗi. Một biểu thức có n biến, thì có  $2^n$  khả năng kiểm thử xảy ra khi ( $n > 0$ ).

#### Mục đích của kiểm thử điều kiện

Để xác định không chỉ các lỗi điều kiện mà cả các lỗi khác trong chương trình. Có một số phương pháp kiểm thử điều kiện được đề xuất:

**Kiểm thử nhánh (Branch Testing):** là phương pháp kiểm thử điều kiện đơn giản nhất.

**Kiểm thử miền (Domain Testing):** cần 3 hoặc 4 kiểm thử cho biểu thức quan hệ. Với một biểu thức quan hệ có dạng E1 < phép toán quan hệ > E2, cần có 3 kiểm thử được thiết kế cho E1 = E2, E1 > E2, E1 < E2.

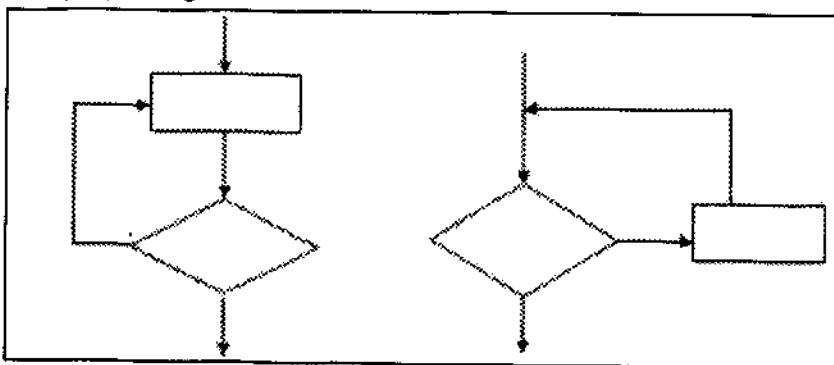
## Kiểm thử nhánh và toán tử quan hệ (Branch and Relational Operator - BRO).

### KIỂM THỬ VÒNG LẶP

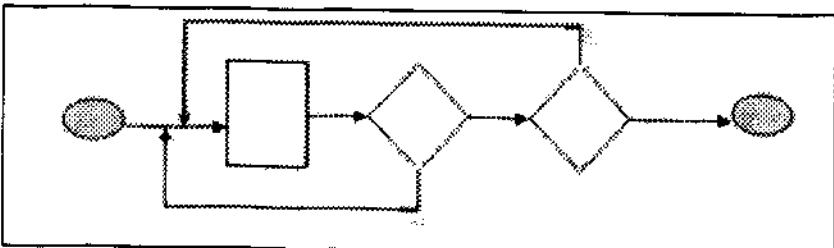
Vòng lặp là nền tảng cho hầu hết các thuật toán được cài đặt trong phần mềm. Tuy nhiên, chúng ta thường ít quan tâm đến nó khi thực hiện việc kiểm thử phần mềm. Kiểm thử vòng lặp là một kỹ thuật kiểm thử hộp trắng tập trung trên tính hợp lệ của các cấu trúc lặp. Việc xây dựng các trường hợp kiểm thử cho mỗi loại cần thực hiện như sau:

Là phương pháp tập trung vào tính hợp lệ của các cấu trúc vòng lặp. Có 4 loại vòng lặp, mỗi loại dùng một tập các phép thử khác nhau.

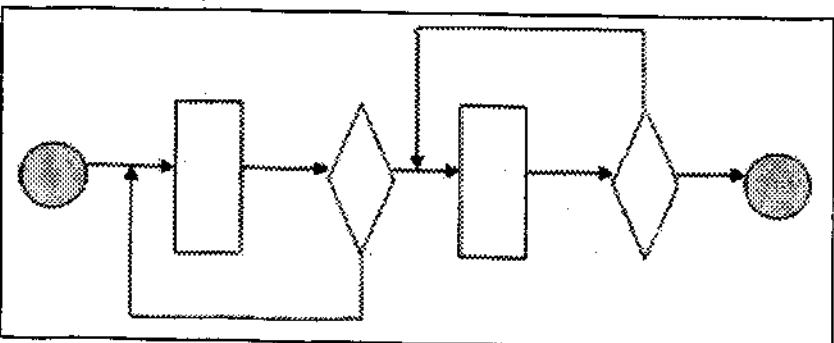
- Vòng lặp đơn giản.



- Vòng lặp lồng nhau.

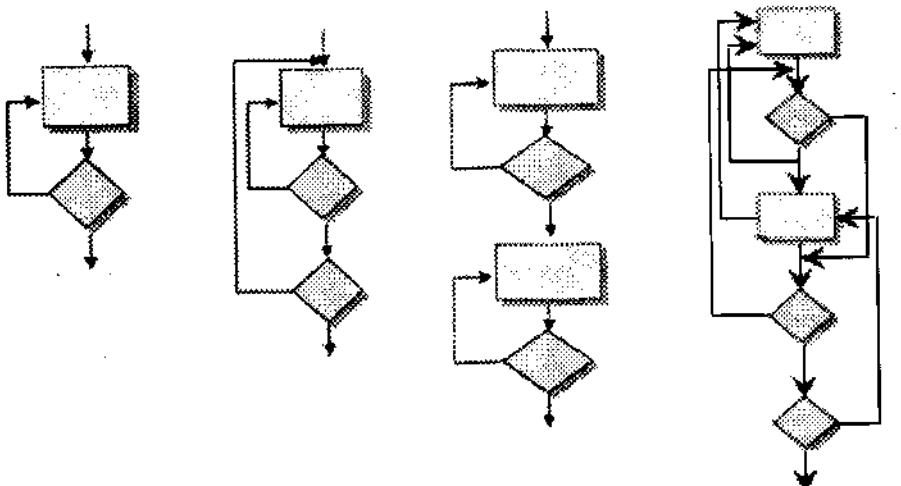


- Vòng lặp nối tiếp.



- Vòng lặp không có cấu trúc.

### Tổng hợp các vòng lặp biểu diễn đơn giản



Vòng lặp  
đơn giản

Vòng lặp  
lồng nhau

Vòng lặp  
nối tiếp

Vòng lặp  
không cấu trúc

#### Các bước cần kiểm tra cho vòng lặp đơn:

- Bỏ qua vòng lặp.
- Lặp một lần.
- Lặp hai lần.
- Lặp  $m$  lần với  $m < n$ .
- Lặp  $n-1$ ;  $n$ ;  $n+1$  lần.
- Trong đó  $n$  là số lần lặp tối đa của vòng lặp.

#### Các bước cần kiểm tra cho vòng lặp dạng lồng nhau

Nếu mở rộng phương pháp kiểm thử vòng lặp đơn cho vòng lặp lồng nhau thì các kiểm thử có thể sẽ tăng theo mức phát triển vòng lặp. Điều này có thể tạo ra các trường hợp kiểm thử không thực tế.

Vì vậy, một cách tiếp cận để qui như sau sẽ giảm bớt số trường hợp kiểm thử.

- Khởi đầu với vòng lặp nằm bên trong nhất. Thiết lập các tham số lặp cho các vòng lặp bên ngoài về giá trị nhỏ nhất.
- Kiểm tra với tham số min +1, 1 giá trị tiêu biểu, max-1 và max cho vòng lặp bên trong nhất trong khi các tham số lặp của các vòng lặp bên ngoài là nhỏ nhất.

- Tiếp tục tương tự với các vòng lặp liền ngoài tiếp theo cho đến khi tất cả vòng lặp bên ngoài được kiểm tra.

**Các bước cần kiểm tra cho vòng lặp nối tiếp:**

- Nếu các vòng lặp là độc lập với nhau thì kiểm tra như trường các vòng lặp dạng đơn, nếu không thì kiểm tra như trường hợp các vòng lặp lồng vào nhau.

**Ví dụ:**

```
// LOOP TESTING EXAMPLE PROGRAM
import java.io.*;

class LoopTestExampleApp {
    // ----- FIELDS -----
    public static
    BufferedReader keyboardInput =
    new BufferedReader (new InputStreamReader(System.in));
    private static final int MINIMUM = 1;
    private static final int MAXIMUM = 10;
    // ----- METHODS -----
    /* Main method */
    public static void main(String[] args) throws IOException
    {
        System.out.println("Input an integer value:");
        int input = new Integer(keyboardInput.readLine()).intValue();
        int numberOfliterations=0;
        for(int index=input;index >= MINIMUM && index <=
        MAXIMUM;index++) { numberOfliterations++;
        }
        // Output and end
        System.out.println("Number of iterations = " + numberOfliterations);
    }
}
```

Giá trị đầu vào	Kết quả (số vòng lặp)
11	0 (bỏ qua vòng lặp)
10	1 (lặp 1 lần)
9	2 (lặp 2 lần)
5	6 (lặp m lần khi m < n)
2	9 (lặp N - 1 lần)
1	10 (lặp N lần)
0	0 (bỏ qua vòng lặp)

### Kiểm thử dựa trên luồng dữ liệu

Phương pháp kiểm thử luồng dữ liệu chọn lựa một số đường diễn tiến của chương trình dựa vào việc cấp phát, định nghĩa, và sử dụng những biến trong chương trình.

Để hình dung ra cách tiếp cận này ta giả sử rằng: mỗi câu lệnh của chương trình được gán một số duy nhất và mỗi hàm không được thay đổi thông số của nó và biến toàn cục.

$$\text{DEF}(S) = \{ X \mid \text{lệnh } S \text{ chứa định nghĩa } X \}$$

$$\text{USE}(S) = \{ X \mid \text{lệnh } S \text{ chứa một lệnh/biểu thức sử dụng } X \}$$

Nếu S là câu lệnh if hay loop, thì tập DEF của S là rỗng và USE là tập dựa trên điều kiện của câu lệnh S.

Định nghĩa 1: biến X tại câu lệnh S được cho là vẫn còn tại câu lệnh S' nếu như tồn tại một đường từ câu lệnh S đến câu lệnh S' không chứa bất kỳ định nghĩa nào của X.

Định nghĩa 2: một chuỗi dùng của biến X (gọi là DU của X) ký hiệu [X, S, S'] là định nghĩa của X trong câu lệnh S vẫn tồn tại trong câu lệnh S'.

Phương pháp kiểm thử luồng dữ liệu yêu cầu: tất cả các chuỗi DU đều được kiểm thử ít nhất một lần. Có thể thấy rằng, bộ kiểm thử cho luồng dữ liệu có thể không bao trùm tất cả các nhánh của chương trình.

Tuy nhiên nếu một nhánh đảm bảo được sẽ được phát hiện bởi phương pháp kiểm thử này. Trong một số ít trường hợp như là cấu trúc lệnh if-then, trong phần then không có định nghĩa thêm một biến nào và phần else không tồn tại. Trong tình huống này, nhánh else của câu lệnh trên không cần thiết phải bảo hộ bởi phương pháp này.

Kiểm thử luồng dữ liệu có lợi cho việc kiểm thử những chương trình có nhiều lệnh if và lệnh lặp lồng nhau nhiều cấp.

Ví dụ: một thủ tục với lệnh điều kiện và lệnh lặp phức tạp:

```

PROC X
    A1;
    DO WHILE B1
        IF B2
            THEN
                IF B4
                    THEN A4;
                ELSE A5
                ENDIF;
            ELSE
                IF B3
                    THEN A2
                ELSE A3
                ENDIF;
            ENDIF
        END DO
    A6
END PROC

```

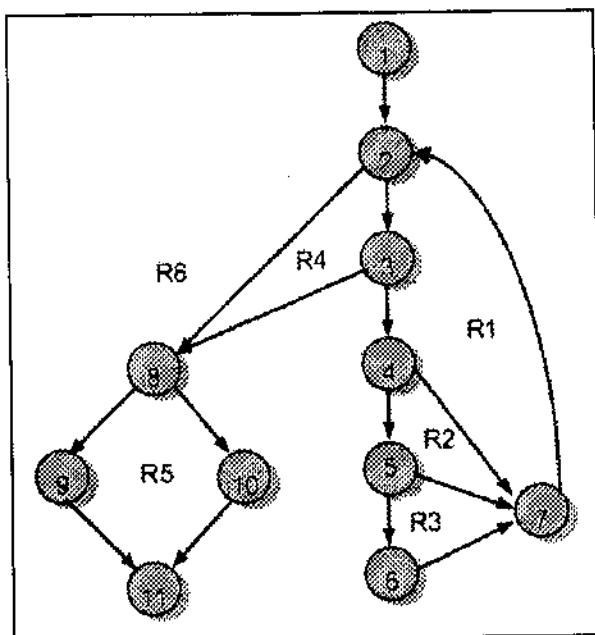
Để xây dựng các trường hợp kiểm thử luồng dữ liệu cho thủ tục trên, cần phải biết định nghĩa và sử dụng biến ở mỗi điều kiện hoặc một khối trong thủ tục ở trên. Giả sử biến X được định nghĩa trong câu lệnh cuối của khối lệnh A2, A3, A4 và A5. Và biến X được sử dụng ở đầu của các khối A2, A3, A4, A5 và A6. Kiểm thử DU yêu cầu đường thực thi ngắn nhất từ A1,  $0 < i \leq 5$  đến B1  $1 < j \leq 6$ .

### **3. KIỂM THỬ ĐỘT BIẾN (MUTATION TESTING)**

Kiểm thử đột biến được sử dụng để kiểm tra chất lượng của bộ kiểm tra, điều này được thực hiện bởi việc thay đổi các lệnh trong mã nguồn và kiểm tra nếu mã kiểm tra của bạn có thể tìm thấy lỗi. Tuy nhiên, việc thực hiện kiểm thử đột biến rất tốn kém, đặc biệt là trên các ứng dụng lớn. Jester là một công cụ kiểm tra đột biến có thể được sử dụng để chạy thử nghiệm đột biến trên mã Java, Jester có thể tìm và "nhìn" vào các khu vực cụ thể của mã nguồn.

Ví dụ: buộc một đường dẫn thông qua một lệnh nếu việc thay đổi các giá trị không đổi, và thay đổi các giá trị Boolean.

### BÀI TẬP KIỂM TRÌ HỘP TRẮNG:



**Bước 1: vẽ lưu đồ**

**Bước 2: xác định độ phức tạp cyclomatic**

- $V(G) = R$  (số vùng) = 6
- $V(G) = P$  (số đỉnh điều kiện) + 1 = 5 + 1 = 6
- $V(G) = E$  (số cạnh) -  $N$  (số đỉnh) + 2 = 17 - 13 + 2 = 6

**Bước 3: tìm tập cơ sở các đường dẫn độc lập, các đỉnh (nút vị từ) 2, 3, 4, 5, 8 là các đỉnh điều kiện.**

- Đường dẫn 1:  $1 \Rightarrow 2 \Rightarrow 8 \Rightarrow 9 \Rightarrow 11$
- Đường dẫn 2:  $1 \rightarrow 2 \rightarrow 8 \Rightarrow 10 \Rightarrow 11$
- Đường dẫn 3:  $1 \rightarrow 2 \Rightarrow 3 \Rightarrow 8 \rightarrow 9 \rightarrow 11$
- Đường dẫn 4:  $1 \rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 7 \Rightarrow 2 \rightarrow \dots$
- Đường dẫn 5:  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \Rightarrow 5 \Rightarrow 7 \rightarrow 2 \rightarrow \dots$
- Đường dẫn 6:  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \Rightarrow 6 \Rightarrow 7 \rightarrow 2 \rightarrow \dots$

**Bước 4: thiết kế các trường hợp kiểm thử cho mỗi đường dẫn độc lập trong tập cơ sở đã chọn.**

**Trường hợp kiểm thử đường dẫn 1**

- Đầu vào: values = {3, 5, 11, -999}, min = 0, max = 100
- Đầu ra mong muốn: average =  $(3 + 5 + 11)/3$

**Mục đích:** để kiểm thử việc tính trung bình chính xác.

**Chú ý:** đường dẫn 1 không thể kiểm thử một mình mà phải được kiểm thử như là một phần của các kiểm thử đường dẫn 4, 5, và 6.

**Trường hợp kiểm thử đường dẫn 2**

- Đầu vào: values = {-999}, min = 0, max = 0
- Đầu ra mong muốn: averag = -999
- Mục đích: để tạo ra average = -999

**Trường hợp kiểm thử đường dẫn 3**

- Đầu vào: values = {3, 5, 30, ..., 76} (101 số), min = 0, max = 100
- Đầu ra mong muốn: trung bình của 100 số đầu tiên.

**Mục đích:** chỉ tính trung bình cho 100 số hợp lệ đầu tiên .

**Trường hợp kiểm thử đường dẫn 4**

- Đầu vào: values = {67, -2, 12, 23, -999}, min = 0, max = 100
- Đầu ra mong muốn:  $(67 + 12 + 23)/3$

**Mục đích:** kiểm thử biên dưới (values[i]<min, i<100).

**Trường hợp kiểm thử đường dẫn 5**

- Đầu vào: values = {7, 32, 102, 23, 86, 2, -999}, min = 0, max = 100
- Đầu ra mong muốn:  $(7 + 32 + 23 + 86 + 2)/5$

**Mục đích:** kiểm thử biên trên (values[i]>max, i<100).

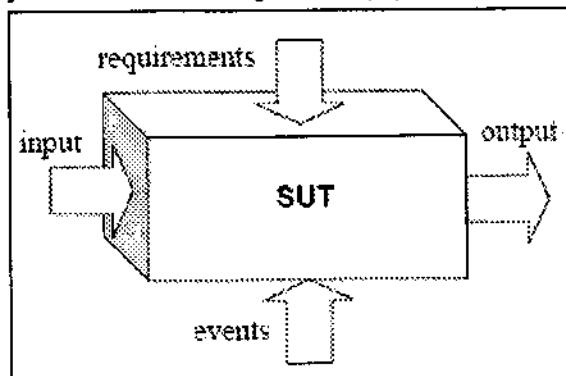
**Trường hợp kiểm thử đường dẫn 6**

- Đầu vào: values = {7, 32, 99, 23, 86, 2, -999}, min = 0, max = 100
- Đầu ra mong muốn:  $(7 + 32 + 99 + 23 + 86 + 2)/6$

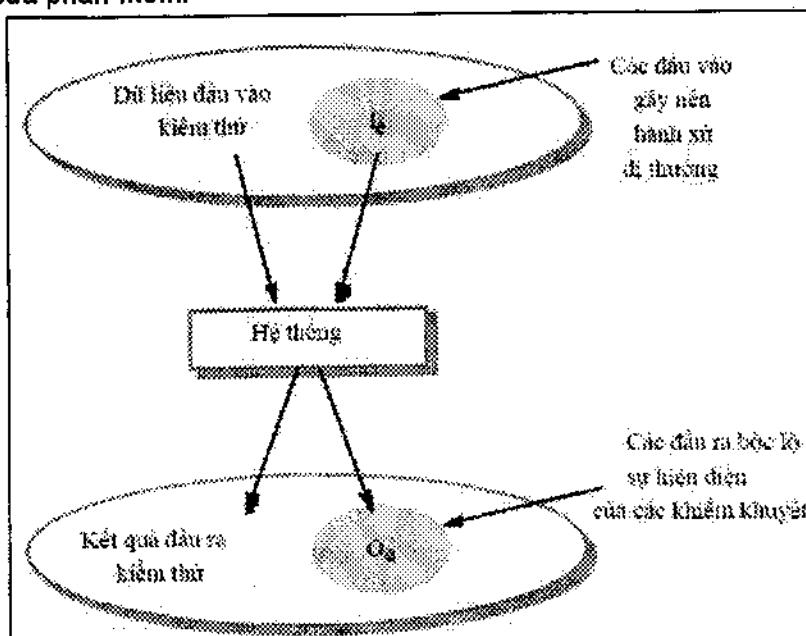
**Mục đích:** Việc tính trung bình là đúng.

**CHƯƠNG 7****KỸ THUẬT KIỂM THỬ HỘP ĐEN****1. KỸ THUẬT KIỂM THỬ HỘP ĐEN (BLACK BOX TESTING)**

Kỹ thuật kiểm thử hộp đen còn được gọi là kiểm thử hướng dữ liệu (data-driven) hay là kiểm thử hướng vào/ra (input/output driven).



Trong kỹ thuật này, quá trình kiểm thử xem phần mềm như là một hộp đen. Người kiểm thử hoàn toàn không quan tâm cấu trúc và hành vi bên trong của phần mềm.



Họ chỉ cần quan tâm đến việc tìm các hiện tượng mà phần mềm không hành xử theo đúng đặc tả (mô tả lỗi) của nó. Vì thế, dữ liệu kiểm thử sẽ xuất phát từ đặc tả.

Như vậy, cách tiếp cận kiểm thử hộp đen tập trung vào các yêu cầu chức năng của phần mềm. Kiểm thử hộp đen cho phép các chuyên viên kiểm thử xây dựng các nhóm giá trị đầu vào sẽ thực thi đầy đủ tất cả các yêu cầu chức năng của phần mềm. Kiểm thử hộp đen không thay thế kỹ thuật hộp trắng, nhưng bổ sung khả năng phát hiện các lỗ lỗi khác với các phương pháp hộp trắng.

Việc kiểm thử này được thực hiện mà không cần quan tâm đến các thiết kế và viết mã của chương trình. Kiểm nghiệm theo cách này chỉ quan tâm đến chức năng đã đề ra của chương trình. Vì vậy, kiểm nghiệm loại này chỉ dựa vào bảng mô tả chức năng của chương trình, xem chương trình có thực sự cung cấp đúng chức năng đã mô tả trong bảng chức năng hay không mà thôi. Kiểm nghiệm hộp đen dựa vào các định nghĩa về chức năng của chương trình. Các trường hợp thử nghiệm (test case) sẽ được tạo ra dựa nhiều vào bảng mô tả chức năng chứ không phải dựa vào cấu trúc của chương trình.

#### Kiểm thử hộp đen cố gắng tìm các loại lỗi sau:

- Các chức năng thiếu hoặc không đúng.
- Các lỗi giao diện.
- Các lỗi cấu trúc dữ liệu trong khi truy cập cơ sở dữ liệu bên ngoài.
- Các lỗi thi hành.
- Các lỗi khởi tạo hoặc kết thúc.
- Và các lỗi khác...

Không giống kiểm thử hộp trắng được thực hiện sớm trong quá trình kiểm thử, kiểm thử hộp đen nhắm đến các giai đoạn sau của kiểm thử. Vì kiểm thử hộp đen không để ý có chủ đích cấu trúc điều khiển, sự quan tâm tập trung trên miền thông tin.

Nếu muốn sử dụng phương pháp này để tìm tất cả các lỗi trong chương trình thì điều kiện bắt buộc là phải kiểm thử tất cả các đầu vào, tức là mỗi một điều kiện đầu vào có thể có là một trường hợp kiểm thử.

Lý do: vì nếu chỉ kiểm thử một số điều kiện đầu vào thì không đảm bảo được chương trình đã hết lỗi. Tuy nhiên, điều này thực tế không thể thực hiện được.

#### Các phương pháp kiểm thử hộp đen

- Phân lớp tương đương – Equivalence partitioning.
- Phân tích giá trị biên – Boundary value analysis.

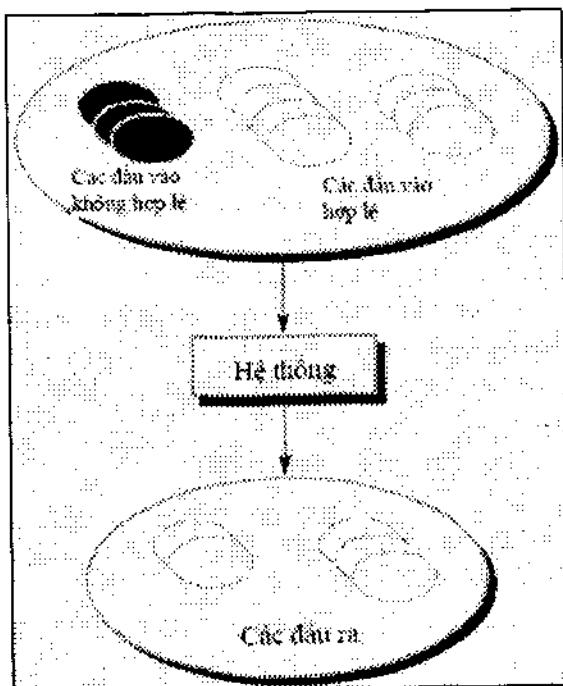
- Kiểm thử mọi cặp – All-pairs testing.
- Kiểm thử fuzz – Fuzz testing.
- Kiểm thử dựa trên mô hình – Model-based testing.
- Ma trận dấu vết – Traceability matrix.
- Kiểm thử thăm dò – Exploratory testing.
- Kiểm thử dựa trên đặc tả – Specification-base testing.

## 2. PHÂN HOẠCH TƯƠNG ĐƯỜNG

Việc thực hiện kiểm thử tất cả các dấu vào của phần mềm là không thể. Cho nên, khi kiểm thử chương trình nên giới hạn một tập con tất cả các trường hợp dấu vào có thể có.

**Một tập con như vậy cần có hai tính chất:**

Mỗi trường hợp kiểm thử nên gồm nhiều điều kiện dấu vào khác nhau có thể để giảm thiểu tổng số các trường hợp cần thiết.



Nên cố gắng phân hoạch các miền dấu vào của một chương trình thành một số xác định các lớp tương đương, sao cho có thể già định hợp lý rằng việc kiểm thử một giá trị đại diện của mỗi lớp là tương đương với việc kiểm thử một giá trị bất kỳ trong cùng lớp. Hai vấn đề xem xét ở trên tạo thành một phương pháp của kỹ thuật hộp đen và gọi là phân hoạch tương đương. Vấn đề thứ hai được sử dụng để phát triển một tập các điều kiện cần quan tâm phải được kiểm thử.

Vấn đề thứ nhất được sử dụng để phát triển một tập cực tiểu các trường hợp kiểm thử phù các điều kiện trên.

Thiết kế trường hợp kiểm thử bằng phân hoạch tương đương được xử lý theo hai bước: phân hoạch các miền dấu vào/ra thành các lớp tương đương, và thiết kế các trường hợp kiểm thử đại diện cho mỗi lớp.

### Xác định các lớp tương đương

Vạch định tương đương được định nghĩa theo lý thuyết tập hợp.

- Quan hệ  $r$  trên hai tập  $A$  và  $B$  là một tập con của tích  $\mathcal{D}$  các  $A \times B$ , nghĩa là  $a \mathrel{r} b$  trong đó  $a \in A$  và  $b \in B$ .
- Quan hệ  $r$  có thể được định nghĩa trên chính tập  $A$ , tức là khi  $B = A$ .
- Quan hệ  $r$  trên tập  $A$  gọi là phản xạ nếu  $a \mathrel{r} a$  với  $\forall a \in A$
- Quan hệ  $r$  trên tập  $A$  gọi là đối xứng nếu  $a \mathrel{r} b$  và  $b \mathrel{r} a$  với  $\forall a, b \in A$
- Quan hệ  $r$  trên tập  $A$  gọi là bắc cầu nếu  $a \mathrel{r} b$  và  $b \mathrel{r} c$  và  $a \mathrel{r} c$  với  $\forall a, b, c \in A$
- Một quan hệ có tính phản xạ, đối xứng và bắc cầu gọi là quan hệ tương đương.
- Một quan hệ tương đương phân hoạch tập hợp thành các lớp tương đương rời rạc.

Như vậy, các lớp tương đương được nhận dạng bằng cách lấy từng điều kiện đầu vào (thông thường là một câu lệnh hoặc một cụm từ trong đặc tả) và phân hoạch thành hai hoặc nhiều nhóm. Các lớp tương đương biểu diễn một tập các trạng thái hợp lệ hoặc không hợp lệ cho điều kiện đầu vào. Điều kiện đầu vào là giá trị số xác định, hoặc miền giá trị, tập giá trị có liên quan, hoặc điều kiện logic. Để làm điều này, chúng ta sử dụng bảng liệt kê các lớp tương đương.

#### Bao gồm các thành phần:

- Điều kiện vào/ra
- Lớp tương đương hợp lệ
- Lớp tương đương không hợp lệ

#### Các lớp tương đương được định nghĩa theo các nguyên tắc sau:

Nếu điều kiện đầu vào xác định một khoảng giá trị  $[a, b]$ , thì phân hoạch thành một lớp tương đương hợp lệ và một lớp tương đương không hợp lệ.

Chẳng hạn, nếu đầu vào  $x$  nằm trong khoảng  $[0, 100]$  thì: lớp hợp lệ là  $0 \leq x \leq 100$ ; các lớp không hợp lệ là  $x < 0$  và  $x > 100$ .

- Nếu điều kiện đầu vào yêu cầu một giá trị xác định, phân hoạch thành một lớp tương đương hợp lệ và hai lớp tương đương không hợp lệ. Chẳng hạn, nếu đầu vào  $x = 5$  thì lớp hợp lệ là  $x = 5$ , các lớp không hợp lệ là  $x < 5$  và  $x > 5$ .
- Nếu điều kiện đầu vào xác định một phần tử của tập hợp, thì phân hoạch thành một lớp tương đương hợp lệ và thêm một lớp tương đương không hợp lệ.
- Nếu điều kiện đầu vào là Boolean, thì phân hoạch thành một lớp tương đương hợp lệ và một lớp tương đương không hợp lệ tương ứng với hai

trạng thái true và false.

Ngoài ra, một nguyên tắc thứ năm được bổ sung là sử dụng khả năng phán đoán, kinh nghiệm và trực giác của người kiểm thử.

#### Xác định các trường hợp kiểm thử

Bước thứ hai trong phương pháp phân hoạch tương đương là thiết kế các trường hợp kiểm thử dựa trên sự ước lượng của các lớp tương đương cho miền đầu vào.

Tiến trình này được thực hiện như sau:

- Gán một giá trị duy nhất cho mỗi lớp tương đương.
- Khi tất cả các lớp tương đương hợp lệ được phủ bởi các trường hợp kiểm thử thì viết một trường hợp kiểm thử mới phủ nhiều nhất có thể các lớp tương đương hợp lệ chưa được phủ.
- Đến khi tất cả các lớp tương đương không hợp lệ được phủ bởi các trường hợp kiểm thử thì hãy viết các trường hợp kiểm thử mới sao cho mỗi trường hợp kiểm thử mới chỉ phủ duy nhất một lớp tương đương không hợp lệ chưa được phủ.

Xem bảng ví dụ về lớp tương đương sau:

Điều kiện đầu vào	Các lớp tương đương hợp lệ	Các lớp tương đương không hợp lệ
Số ID của sinh viên	Các ký số	Không phải ký số
Tên sinh viên	Ký tự chữ cái Không rỗng	Không phải chữ cái Rỗng
Giới tính sinh viên	Ký tự chữ cái. "M" hoặc "F"	Không phải chữ cái Không phải "M" hoặc "F"
Điểm của sinh viên	Số Từ 0 đến 100	Không phải số Số nhỏ hơn 0, số lớn hơn 100

### 3. PHÂN TÍCH GIÁ TRỊ BIÊN (BVA - BOUNDARY VALUE ANALYSIS)

Khi thực hiện việc kiểm thử phần mềm theo dữ liệu, chúng ta kiểm tra xem đầu vào của người dùng, kết quả nhận được và kết quả tạm thời bên trong có được xử lý chính xác hay không. Các điều kiện biên là tình trạng trực tiếp ở phía trên và dưới của các lớp tương đương đầu vào và lớp tương đương đầu ra.

Việc phân tích các giá trị biên khác với phân hoạch tương đương theo hai điểm:

Từ mỗi lớp tương đương, phân hoạch tương đương sẽ chọn phần tử bắt

kỳ làm phần tử đại diện, trong khi việc phân tích giá trị biên sử dụng một hoặc một số phần tử.

Như vậy, mỗi biên của lớp tương đương chính là đích kiểm thử. Không chỉ chú ý tập trung vào những điều kiện đầu vào, các trường hợp kiểm thử cũng được suy ra từ việc xem xét các kết quả ra (tức các lớp tương đương đầu ra). Rất khó có thể liệt kê hết các hướng dẫn cụ thể cho các trường hợp. Tuy nhiên, cũng có một số nguyên tắc phân tích giá trị biên như trình bày sau:

1. Nếu điều kiện đầu vào xác định một khoảng giá trị giữa a và b, các trường hợp kiểm thử sẽ được thiết kế với giá trị a và b, và các giá trị sát trên và sát dưới a và b.
2. Nếu một điều kiện đầu vào xác định một số các giá trị, các trường hợp kiểm thử sẽ được phát triển để thực hiện tại các giá trị cực đại, cực tiểu. Các giá trị sát trên và dưới giá trị cực đại, cực tiểu cũng được kiểm thử.
3. Nguyên tắc 1 và 2 được áp dụng cho các điều kiện đầu ra.
4. Nếu cấu trúc dữ liệu chương trình bên trong được qui định các biên (chẳng hạn, mảng được định nghĩa giới hạn 100 mục), tập trung thiết kế trường hợp kiểm thử để thực thi cấu trúc dữ liệu tại biên của nó.

Ngoài ra, người kiểm thử có thể sử dụng sự xét đoán và sáng tạo của mình để tìm các điều kiện biên. Tóm lại, cần phải kiểm thử mỗi biên của một lớp tương đương về tất cả các phía. Một chương trình nếu vượt qua những trường hợp kiểm thử đó có thể vượt qua các kiểm thử khác từ lớp đó.

**Ví dụ:** phân tích giá trị biên

Nhập vào số **integer maxint** và **value** tính toán giá trị **result** như sau:

$$\text{Result} = \sum_{k=0}^{\text{Value}} k$$

CONDITION	LỚP TƯƠNG ĐƯƠNG 'VALID'
Abs(value)	Value < 0, value ≥ 0
Maxint	$\sum k \leq \text{Maxint}; \sum k > \text{Maxint}$

Ta tìm cần giá trị giữa  $\text{maxint} < 0$  và  $\text{maxint} \geq 0$ :

$\text{Maxint} \text{ maxint} < 0, 0 \leq \text{maxint} < \sum k, \text{maxint} \geq \sum k,$

Các lớp tương đương valid:

Abs(value)      Value < 0, value ≥ 0

Maxint            Maxint < 0, 0 ≤ maxint <  $\sum k$ , maxint  $\geq \sum k$

Các trường hợp kiểm thử.

Xem bảng trang bên.

Maxint	Value	Result	Maxint	Value	Result
55	10	55	100	0	0
54	10	error	100	-1	1
56	10	55	100	1	1
0	0	0	-	-	-

#### 4. KỸ THUẬT ĐỒ THỊ NHÂN-QUẢ (CAUSE - EFFECT GRAPH)

Trong nhiều trường hợp, việc cố gắng chuyển một chính sách hoặc một thủ tục trong ngôn ngữ tự nhiên vào phần mềm dẫn đến sự thất bại và các vấn đề khó hiểu.

- Đồ thị nhân - quả là một phương pháp thiết kế trường hợp kiểm thử trên cơ sở đưa ra một sự mô tả súc tích các điều kiện logic và các hành vi đi kèm theo.
- Đồ thị nhân - quả sử dụng mô hình các quan hệ logic giữa nguyên nhân và kết quả cho thành phần phần mềm. Mỗi nguyên nhân được biểu diễn như một điều kiện (đúng hoặc sai) của một đầu vào, hoặc kết hợp các đầu vào. Mỗi kết quả được biểu diễn như là một biểu thức Bool biểu diễn một kết quả tương ứng cho những thành phần vừa thực hiện.

#### ĐẶC TÍNH:

Cause → Inputs (trạng thái hiện tại)

Effect → Effect (trạng thái mới)

- Tạo một đồ thị kết nối Causes và Effects
- Chú thích nếu không thể kết hợp Causes và Effects.
- Phát triển bảng quyết định từ đồ thị ứng với mỗi cột, một sự kết hợp đặc biệt của đầu vào và đầu ra.
- Mỗi trường hợp test phải thay đổi cột.

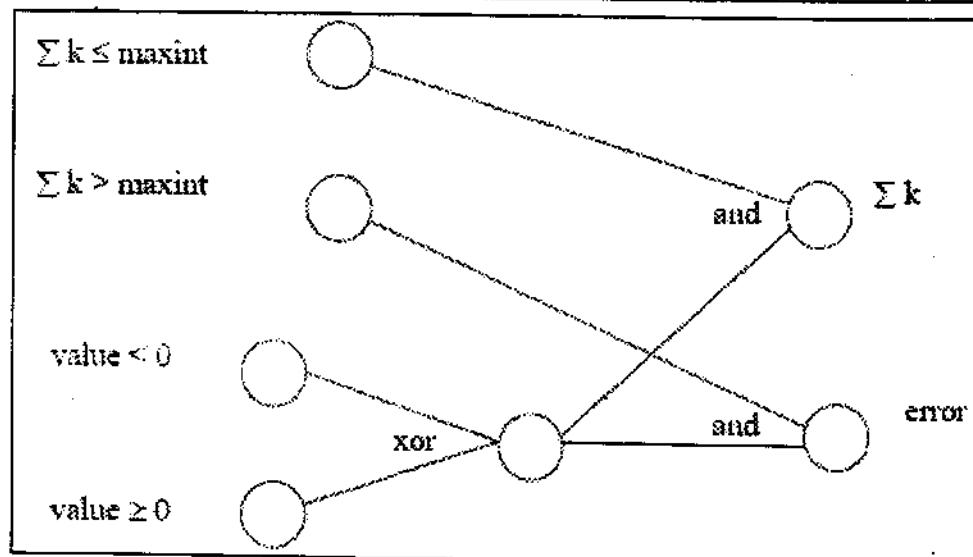
Như ví dụ trên, ta có các trường hợp như sau:

	MAXINT	VALUE	RESULT
Valid	100	10	55
	100	-10	55
	10	10	Error
Invalid	10	-	Error
	10	30	Error

	XYZ	10	Error
	100	9.1E4	Error

**BẢNG TRƯỜNG HỢP:**

Causes	$\Sigma k \leq \text{Maxint}$	1	1	0	0
Inputs	$\Sigma k > \text{Maxint}$	0	0	1	1
	Value < 0	1	0	1	0
	Value $\geq 0$	0	1	0	1
Effects	$\Sigma k$	1	1	0	0
Outputs	Error	0	0	1	1

**Đồ thị nhân - quả được tạo như sau:**

Tất cả các nguyên nhân (các đầu vào) và kết quả (các đầu ra) được liệt kê dựa trên đặc tả và được định danh cho mỗi nhân - quả. Các quan hệ giữa các nguyên nhân (các đầu vào) và các kết quả (các đầu ra) được biểu diễn trong đồ thị làm rõ ràng các quan hệ logic.

Từ đồ thị tạo ra bảng quyết định biểu diễn các quan hệ giữa nguyên nhân và kết quả.

Dữ liệu kiểm thử được sinh ra dựa trên các qui tắc trong những bảng

này. Các ký hiệu được đơn giản hóa sử dụng trong đồ thị nhân quả, gồm các phần tử mô tả trong hình trang bên.

STT	KÝ HIỆU	Ý NGHĨA	GIẢI NGHĨA
1	INDENTY 	Tương đương	Nếu a đúng thì b đúng
	NOT 	NOT (phù dịnh)	Nếu a sai thì b đúng
2	AND 	AND (và)	Nếu a và b đúng thì c đúng
3	OR 	OR (hoặc)	Nếu a hoặc b hoặc c đúng thì d đúng
4	E 	LOẠI TRÙ	Nếu a đúng thì b sai; hoặc a sai thì b đúng
5	O 	BAO HÀM	a bao hàm b

<b>6</b>		<b>YÊU CẦU</b>	a yêu cầu b
----------	--	----------------	-------------

Các qui tắc trong bảng quyết định được mô tả như sau:

Tên bảng	Qui tắc				Tên bảng: cho biết tên logic
	1	2	..	n	
Điều kiện 1	Y	Y		Y	Qui tắc: đánh số để phân biệt các qui tắc quyết định logic.
Điều kiện 2	Y	--		Y	
Điều kiện 3	Y	--		N	Các dòng điều kiện: Mỗi dòng bao gồm các điều kiện để tạo quyết định cho chương trình.
...	...	...		...	
Điều kiện n	--	--		Y	Y: "true"
Hành động 1	X	X		X	N: "false"
Hành động 2	--	X		X	-- : Không có quyết định được tạo ra.
Hành động 3	X	--		X	Các hành động: Mỗi dòng chỉ định có các xử lý được thực hiện hoặc không.
...	...	...		...	
Hành động n	--	--		X	X: Xử lý được thực hiện.
					-- : Không có xử lý được thực hiện.

**Ví dụ:** Để tính thuế thu nhập, người ta có mô tả sau:

Người vô gia cư nộp 4% thuế thu nhập

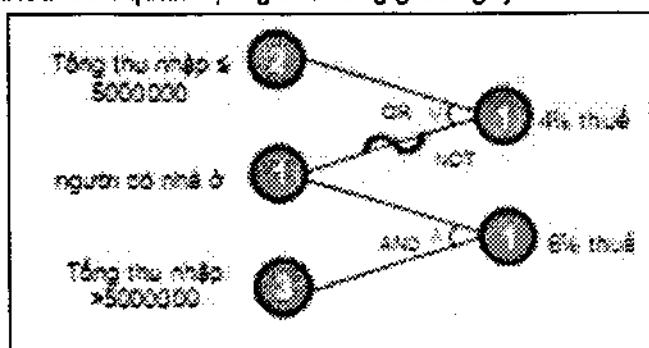
Người có nhà ở nộp thuế theo bảng sau:

<= 5.000.000 đồng	4%
> 5.000.000 đồng	6%

Quan hệ giữa nguyên nhân (đầu vào) và kết quả (đầu ra) thể hiện trong bảng dưới thể hiện như hình ở trang sau:

Nguyên nhân	Kết quả
1. Người có nhà ở	1. Nộp 4 % thuế
2. Tổng thu nhập <= 5.000.000 đồng	2. Nộp 6% thuế
3. Tổng thu nhập > 5.000.000 đồng	

### Đồ thị biểu diễn quan hệ logic rõ ràng giữa nguyên nhân - kết quả



Xây dựng bảng quyết định dựa trên đồ thị. Từ đây xây dựng được bốn trường hợp kiểm thử (một trường hợp cho việc nộp thuế 6% và ba trường hợp kiểm thử cần cho việc nộp thuế 4%).

		Trường hợp kiểm thử			
Nguyên nhân và kết quả		1	2	3	4
Nguyên nhân	1. Người có nhà ở	Y	Y	N	-
	2. Có tổng thu nhập <= 5.000.000	N	Y	--	Y
	3. Có tổng thu nhập > 5.000.000	Y	N	--	--
Kết quả	11. Nộp thuế 4%	--	X	X	X
	12. Nộp thuế 6%	X	--	--	--

Để đảm bảo phủ nhận quả 100%, các trường hợp kiểm thử phải được phát sinh tương ứng với các qui tắc trong bảng trên.

#### Kiểm thử so sánh

Có một số trường hợp (như thiết bị điện tử trên máy bay, điều khiển thiết bị năng lượng hạt nhân) trong đó độ tin cậy của phần mềm là tuyệt đối quan trọng, người ta thường gọi là phần mềm tuyệt đối đúng. Trong các ứng dụng như vậy, phần cứng và phần mềm không cần thiết được sử dụng thường xuyên để giảm tối thiểu khả năng lỗi. Khi phần mềm không cần thiết được phát triển, các nhóm công nghệ phần mềm riêng biệt phát triển các phiên bản độc lập của ứng dụng sử dụng cùng một đặc tả.

Trong các trường hợp như vậy, mỗi phiên bản có thể được kiểm thử với cùng dữ liệu kiểm thử để đảm bảo tất cả cung cấp đầu ra giống nhau. Sau đó tất cả các phiên bản được thực thi song song với so sánh thời gian thực các kết quả để đảm bảo tính chắc chắn.

Các phiên bản độc lập là cơ sở của kỹ thuật kiểm thử hộp đen được gọi là kiểm thử so sánh hay kiểm thử back-to-back.

Kiểm thử so sánh là không rõ ràng. Nếu đặc tả mà tất cả các phiên bản được phát triển trên đó là có lỗi, thì tất cả các phiên bản sẽ có khả năng dẫn đến lỗi. Hơn nữa, nếu mỗi phiên bản độc lập tạo ra giống nhau, nhưng không đúng, các kết quả, kiểm thử điều kiện sẽ thất bại trong việc phát hiện lỗi.

### Đoán lỗi

Không cần một phương pháp đặc biệt nào, một số chuyên gia có thể kiểm tra các điều kiện lỗi bằng cách đoán lỗi dễ xảy ra. Trên cơ sở trực giác và kinh nghiệm, với các chương trình cụ thể, các chuyên gia đoán trước các loại lỗi có thể, rồi viết các trường hợp kiểm thử để phơi ra các lỗi này.

Một ý tưởng khác là chỉ ra các trường hợp kiểm thử liên quan đến giả định rằng lập trình viên đã mắc phải khi đọc đặc tả.

**CHƯƠNG 8**

# XÂY DỰNG TEST CASE

## 1. XÂY DỰNG TEST CASE VÀ VAI TRÒ CỦA NÓ

Xây dựng Test case là quá trình cấu thành các phương pháp kiểm tra để phát hiện các sai sót, lỗi, khuyết điểm của phần mềm. Kết quả thu được là một phần mềm đạt tiêu chuẩn và đảm bảo các yêu cầu liên quan đến người dùng thực tế. Vai trò của việc xây dựng Test Case là tạo các cuộc kiểm tra tốt nhất, có khả năng phát hiện các lỗi của phần mềm cao nhất. Sẽ không bỏ sót lỗi, đồng thời tiết kiệm nhiều thời gian, chi phí và công sức của người tham gia kiểm thử.

## 2. QUY TRÌNH XÂY DỰNG 1 TEST CASE

Với những yêu cầu về chi phí và ràng buộc thời gian khi thực hiện việc kiểm thử phần mềm dẫn đến điều quan trọng nhất trong kiểm thử là: thiết kế nhiều cuộc kiểm thử. Phương pháp kém hiệu quả nhất là kiểm thử tất cả đều vào ngẫu nhiên, các giá trị cũng được chọn một cách ngẫu nhiên.

Để kiểm thử hộp đen và kiểm thử hộp trắng một cách hoàn toàn là không thể. Do đó, một chiến lược kiểm thử hợp lý là chiến lược có thể kết hợp sức mạnh của cả hai phương pháp trên: phát triển 1 cuộc kiểm thử nghiêm ngặt và khép kín vừa bằng việc sử dụng các phương pháp thiết kế các cuộc kiểm thử hướng hộp đen nào đó và bổ sung thêm những cuộc kiểm thử này bằng việc khảo sát tính logic của chương trình, sử dụng phương pháp hộp trắng. Những chiến lược kết hợp đó bao gồm:

<b>KIỂM THỬ HỘP TRẮNG</b>	<ul style="list-style-type: none"> <li>&gt; Bao phủ câu lệnh.</li> <li>&gt; Bao phủ quyết định.</li> <li>&gt; Bao phủ điều kiện.</li> <li>&gt; Bao phủ điều kiện – quyết định.</li> <li>&gt; Bao phủ đa điều kiện.</li> </ul>
<b>KIỂM THỬ HỘP ĐEN</b>	<ul style="list-style-type: none"> <li>&gt; Phân lớp tương đương.</li> <li>&gt; Phân tích giá trị biên.</li> <li>&gt; Đồ thị nguyên nhân – kết quả.</li> <li>&gt; Đoán lỗi.</li> </ul>

Mỗi phương pháp kiểm thử đều có những ưu điểm cũng như khuyết điểm riêng. Do đó, để có được tập hợp các cuộc kiểm thử tối ưu chúng ta cần kết hợp hầu hết các phương pháp lại với nhau.

Quy trình xây dựng các cuộc kiểm thử sẽ bắt đầu bằng việc phát triển các cuộc kiểm thử sử dụng phương pháp hộp trắng và sau đó phát triển bổ sung các phương pháp hộp đen.

### BƯỚC 1: KIỂM THỬ HỘP TRẮNG

Kiểm thử hộp trắng có liên quan tới mức độ mà các ca kiểm thử thực hiện hay bao phủ tính logic (mã nguồn) của chương trình. Kiểm thử hộp trắng cơ bản là việc thực hiện mọi đường đi trong chương trình, nhưng việc kiểm thử đầy đủ đường đi là một mục đích không thực tế cho một chương trình với các vòng lặp. Các tiêu chuẩn trong kiểm thử bao phủ logic gồm có:

#### Phương pháp bao phủ câu lệnh (Statement Coverage)

Thực hiện mọi câu lệnh trong chương trình ít nhất 1 lần.

Xét đoạn mã lệnh JAVA và chương trình kiểm thử lỗi như sau:

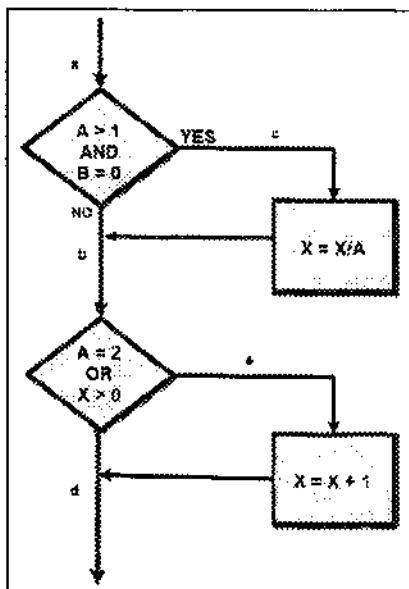
```
Public void foo (int a, int b, int x)
```

```
{
    if (a>1 && b==0)
    {
        x=x/a;
    }
    if (a==2||x>1)
    {
        x=x+1;
    }
}
```

Thực hiện mọi câu lệnh bằng việc viết 1 kiểm thử đơn đi qua đường ace. Tức là, bằng việc đặt A = 3, B = 0 và X = 4 tại điểm a, mỗi câu lệnh sẽ được thực hiện 1 lần (trên thực tế, X có thể được gán bất kỳ giá trị nào).

Thường tiêu chuẩn này khá kém, nếu quyết định đầu tiên là phép or mà không phải phép and thì lỗi này sẽ không được phát hiện. Nếu quyết định thứ hai mà bắt đầu với  $x > 0$ , lỗi này cũng sẽ không được tìm ra.

Quan sát chương trình kiểm của đoạn mã lệnh Java hình bên.



Ngoài ra, còn có 1 đường đi qua chương trình mà ở đó x không thay đổi như đường đi abd. Nếu đây là 1 lỗi, thì lỗi này có thể không tìm ra. Hay nói cách khác, tiêu chuẩn bao phủ câu lệnh quá yếu dẫn đến: nó thường không có giá trị.

### **Phương pháp bao phủ quyết định (decision coverage)**

Viết dù các cuộc kiểm thử mà mỗi quyết định có kết luận đúng hay sai ít nhất 1 lần. Nói cách khác, mỗi hướng phân nhánh phải được xem xét kỹ lưỡng ít nhất 1 lần. Trong số đó có thể xét đến các câu lệnh rẽ nhánh, quyết định như câu lệnh Switch, Do-While, If-Else. Ngoài ra, còn có các câu lệnh đa đường GOTO thường sử dụng trong một số ngôn ngữ lập trình như Fortran.

Bao phủ quyết định thường thỏa mãn bao phủ câu lệnh. Vì mỗi câu lệnh dựa trên sự bắt nguồn một đường đi phụ nào đó hoặc là từ 1 câu lệnh rẽ nhánh hoặc là từ điểm vào của phần mềm. Từng câu lệnh phải được thực hiện nếu từng quyết định rẽ nhánh được thực hiện.

Tuy nhiên, có ít nhất 3 ngoại lệ:

- Những chương trình không có quyết định.
- Những chương trình hay chương trình con với phương thức nhiều điểm vào. Một câu lệnh đã cho có thể được thực hiện nếu và chỉ nếu chương trình được nhập vào tại 1 điểm đầu vào riêng.
- Các câu lệnh bên trong các ON-Unit. Việc đi qua mỗi hướng rẽ nhánh sẽ là không nhất thiết làm cho tất cả các ON-Unit được thực thi.

Việc bao phủ câu lệnh là điều kiện cần thiết, cho nên trong mỗi chiến lược kiểm thử tốt nhất là bao phủ quyết định bao gồm bao phủ câu lệnh. Do đó, bao phủ quyết định yêu cầu mỗi quyết định phải có kết luận đúng hoặc sai, và mỗi câu lệnh đó phải được thực hiện ít nhất 1 lần.

Phương pháp này chỉ xem xét những quyết định hay những sự phân nhánh 2 đường và phải được sửa đổi cho những chương trình có chứa những quyết định đa đường.

Ví dụ:

- Các chương trình JAVA có chứa các lệnh Select (case).
- Các chương trình FORTTRAN chứa các lệnh số học If hoặc các lệnh tính toán hay số học GOTO.
- Các chương trình COBOL chứa các lệnh GOTO biến đổi hay các lệnh GO-TO-DEPENDING-ON (các lệnh GOTO phụ thuộc).

Với những chương trình như vậy, tiêu chuẩn này đang sử dụng mỗi kết luận có thể của tất cả các quyết định ít nhất 1 lần và gọi mỗi điểm vào tới chương trình hay chương trình con ít nhất 1 lần.

Trong đoạn mã lệnh trên, bao phủ quyết định có thể đạt được bởi ít nhất 2 cuộc kiểm thử bao phủ các đường ace và abd hoặc acd và abe.

Nếu chúng ta chọn khả năng thứ hal, thì 2 đầu vào test case là A=3, B=0, X=3 và A=2, B=1, X=1. Bao phủ quyết định là một tiêu chuẩn mạnh hơn bao phủ câu lệnh, nhưng vẫn còn khá yếu.

Ví dụ: chỉ có 50% cơ hội sẽ tìm ra con đường, trong đó x không bị thay đổi (chỉ khi bạn chọn khả năng thứ nhất). Nếu quyết định thứ hai bị lỗi (nếu như đáng lẽ phải nói là  $x < 1$  thay vì  $x > 1$ ), lỗi này sẽ không được phát hiện bằng 2 cuộc kiểm thử trong lần kiểm thử trước.

### Phương pháp bao phủ điều kiện (Condition coverage)

Viết đủ các cuộc kiểm thử để đảm bảo mỗi điều kiện trong một quyết định đảm nhận tất cả các kết quả có thể ít nhất một lần.

Vì vậy, như với bao phủ quyết định, thì bao phủ điều kiện không phải luôn luôn dẫn tới việc thực thi mỗi câu lệnh. Thêm vào đó, trong tiêu chuẩn bao phủ điều kiện, mỗi điểm vào chương trình hay chương trình con, cũng như các ON-Unit, được gọi ít nhất 1 lần.

Ví dụ: Câu lệnh rẽ nhánh Do k=0 to 50 while (j+k<quest) có chứa 2 điều kiện là:  $k \leq 50$  và  $j+k < quest$ .

Do đó, các cuộc kiểm thử sẽ được yêu cầu cho những tình huống:

$k \leq 50$ ,  $k > 50$  (để đến lần lặp cuối cùng của vòng lặp)

$j+k < quest$ ,  $j+k \geq quest$ .

Trên lưu đồ trên có 4 điều kiện:  $A > 1$ ,  $B = 0$ ,  $A = 2$ ,  $X > 1$ .

Do đó các ca kiểm thử đầy đủ là cần thiết để thúc đẩy những trạng thái mà  $A > 1$ ,  $A \leq 1$ ,  $B = 0$  và  $B \neq 0$  có mặt tại điểm a và  $A = 2$ ,  $A \neq 2$ ,  $X > 1$ ,  $X \leq 1$  có mặt tại điểm b.

Số lượng đầy đủ các ca kiểm thử thỏa mãn tiêu chuẩn và những đường đi mà được đi qua bởi mỗi cuộc kiểm thử là:

1.  $A=2, B=0, X=4$  ace

2.  $A=1, B=1, X=1$  abd

Chú ý: mặc dù cùng số lượng các ca kiểm thử được tạo ra cho ví dụ này, nhưng bao phủ điều kiện thường tốt hơn bao phủ quyết định vì nó có thể (nhưng không luôn luôn) gây ra mọi điều kiện riêng trong một quyết định để thực hiện với cả hai kết quả, trong khi bao phủ quyết định lại không.

Ví dụ:

Trong cùng câu lệnh rẽ nhánh: DO K=0 TO 50 WHILE (J+K<QUEST) là 1 nhánh 2 đường (thực hiện ở thân vòng lặp hay bỏ qua nó).

Nếu đang sử dụng kiểm thử quyết định, tiêu chuẩn này có thể được thỏa mãn bằng cách cho vòng lặp chạy từ K=0 tới 51, mà chưa từng kiểm thử trường hợp trong đó mệnh đề WHILE bị sai.

Tuy nhiên, với tiêu chuẩn bao phủ điều kiện, một ca kiểm thử sẽ cần phải cho ra 1 kết quả sai cho những điều kiện  $J+K < QUEST$ .

Mặc dù nếu mới nhìn thoáng qua, tiêu chuẩn bao phủ điều kiện xem ra thỏa mãn tiêu chuẩn bao phủ quyết định, nhưng không phải lúc nào cũng vậy. Nếu quyết định IF (A&B) được kiểm thử, thì tiêu chuẩn bao phủ điều kiện sẽ cho phép bạn viết 2 ca kiểm thử (A đúng, B sai), và (A sai, B đúng) nhưng điều này sẽ không làm cho mệnh đề THEN của câu lệnh IF được thực hiện.

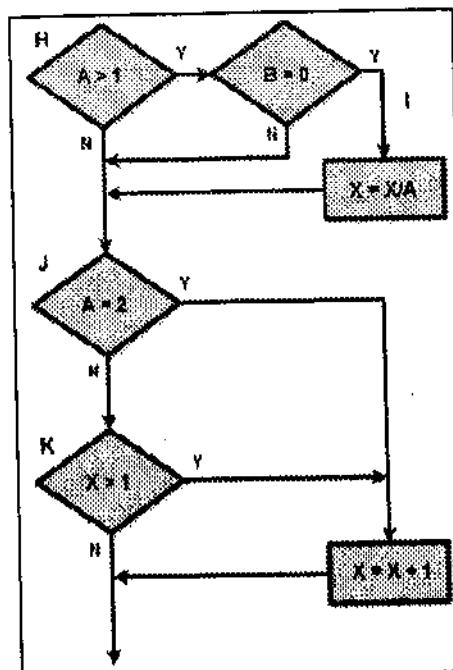
Ví dụ: 2 ca kiểm thử khác:

1. A=1, B=0, X=3
2. A=2, B=1, X=1

Bao phủ tất cả các kết quả điều kiện, nhưng chúng chỉ bao phủ 2 trong 4 kết quả quyết định (cả 2 đều bao phủ đường đi abd và do đó, không sử dụng kết quả True của quyết định đầu tiên và kết quả False của quyết định thứ hai).

#### Bao phủ quyết định/điều kiện – Decision/condition coverage

Thực hiện đủ các cuộc kiểm thử mà mỗi điều kiện trong một quyết định thực hiện trên tất cả các kết quả có thể ít nhất 1 lần, và mỗi điểm vào được gọi ít nhất 1 lần. Điểm yếu của bao phủ quyết định/điều kiện là mặc dù nó có thể sử dụng tất cả các kết quả của tất cả các điều kiện, nhưng thường không phải vậy vì những điều kiện chắc chắn đã cần các điều kiện khác. Máy tính cho chương trình trong lưu đồ tiến trình từ hình bên là cách một trình biên dịch tạo ra mã máy trong chương trình kiểm thử. Các quyết định đa điều kiện trong chương trình nguồn đã bị chia thành các quyết định và các nhánh riêng vì hầu hết các máy không được chế tạo để có thể thực hiện các quyết định đa điều kiện.



Khi đó một bao phủ kiểm thử chi tiết hơn xuất hiện là việc sử dụng tất cả các kết quả có thể của mỗi quyết định gốc.

hai cuộc kiểm thử bao phủ quyết định trước không làm được điều này. Và không thể sử dụng kết quả false của quyết định H và kết quả true của quyết định K.

Lý do, như đã được chỉ ra trong hình là: những kết quả của các điều kiện trong các biểu thức and và or có thể cản trở hay ngăn chặn việc ước lượng các quyết định khác.

Ví dụ: nếu 1 điều kiện and là sai, không cần kiểm thử các điều kiện tiếp theo trong biểu thức. Tương tự như vậy, nếu 1 điều kiện or là đúng thì cũng không cần kiểm thử các điều kiện còn lại. Do đó, các lỗi trong biểu thức logic không phải lúc nào cũng được phát hiện bằng các tiêu chuẩn bao phủ điều kiện và bao phủ quyết định/điều kiện.

#### Bao phủ đa điều kiện – Multiple condition coverage

Viết đủ các cuộc kiểm thử mà tất cả những sự kết hợp của các kết quả điều kiện có thể trong mỗi quyết định, và tất cả các điểm vào phải được gọi ít nhất 1 lần.

Ví dụ: xét chuỗi mã lệnh sau

**NOTFOUND = TRUE;**

**DO I=1 TO TABSIZE WHILE (NOTFOUND); /\*SEARCH TABLE\*/  
...searching logic...;**

**END;**

Các tình huống để kiểm thử như sau:

1.  $I \leq TABSIZE$  và NOTFOUND có giá trị đúng (dang duyệt).
2.  $I \leq TABSIZE$  và NOTFOUND có giá trị sai (tìm thấy mục vào trước khi gặp cuối bảng).
3.  $I > TABSIZE$  và NOTFOUND có giá trị đúng (gặp cuối bảng mà không tìm thấy mục vào).
4.  $I > TABSIZE$  và NOTFOUND có giá trị sai (mục vào là cái cuối cùng trong bảng).

Dễ nhận thấy là: tập hợp các ca kiểm thử thỏa mãn tiêu chuẩn đa điều kiện cũng thỏa mãn các tiêu chuẩn bao phủ quyết định, bao phủ điều kiện và bao phủ quyết định/điều kiện.

Các cuộc kiểm thử phải bao phủ 8 sự kết hợp: (Lưu đồ tiến trình).

- |                      |                      |
|----------------------|----------------------|
| 1. $A > 1, B = 0$    | 5. $A = 2, X > 1$    |
| 2. $A > 1, B < 0$    | 6. $A = 2, X \leq 1$ |
| 3. $A \leq 1, B = 0$ | 7. $A < 2, X \geq 1$ |
| 4. $A \leq 1, B < 0$ | 8. $A < 2, X \leq 1$ |

Vì là cuộc kiểm thử sớm hơn, nên cần chú ý là các trường hợp từ 5 đến 8 biểu diễn các giá trị tại vị trí câu lệnh IF thứ hai. Vì X có thể thay đổi ở trên câu lệnh IF này, nên giá trị cần tại câu lệnh IF này phải được sao dự phòng thông qua tính logic để tìm ra các giá trị đầu vào tương ứng. Những sự kết hợp để được kiểm thử này không nhất thiết phải thực hiện cả 8 cuộc kiểm thử.

Trên thực tế, chúng có thể được bao phủ bởi 4 cuộc kiểm thử. Các giá trị đầu vào kiểm thử, và sự kết hợp mà chúng bao phủ, là như sau:

- **A=2, B=0, X=4 (Bao phủ trường hợp 1, 5)**
- **A=2, B=1, X=1 (Bao phủ trường hợp 2, 6)**
- **A=1, B=0, X=2 (Bao phủ trường hợp 3, 7)**
- **A=1, B=1, X=1 (Bao phủ trường hợp 4, 8)**

Thực tế là việc có 4 cuộc kiểm thử và 4 đường đi riêng biệt chỉ là sự trùng hợp ngẫu nhiên, 4 cuộc kiểm thử này không bao phủ mọi đường đi, chúng bỏ qua đường đi acd.

**Ví dụ:** chúng ta sẽ cần 8 cuộc kiểm thử cho quyết định sau mặc dù nó chỉ chứa 2 đường đi:

```
If (x==y&&length(z)==0&&FLAG)
{
    J=1;
}
Else
{
    I=1;
}
```

Trong trường hợp các vòng lặp, số lượng các cuộc kiểm thử được yêu cầu bởi tiêu chuẩn đa điều kiện thường ít hơn nhiều với số lượng đường đi.

Tóm lại, đối với những chương trình chỉ chứa 1 điều kiện trên 1 quyết định, thì 1 tiêu chuẩn kiểm thử nhỏ nhất là một số lượng đủ các ca kiểm thử để gọi tất cả các kết quả của mỗi quyết định ít nhất 1 lần.

Ngoài ra, gọi mỗi điểm của mục vào (như là điểm vào hay ON-Unit) ít nhất 1 lần, để đảm bảo tất cả các câu lệnh được thực hiện ít nhất 1 lần.

Đối với những chương trình chứa các quyết định có đa điều kiện thì tiêu chuẩn tối thiểu là số lượng đủ các cuộc kiểm thử để gọi tất cả những sự kết hợp có thể của các kết quả điều kiện trong mỗi quyết định, và tất cả các điểm vào của chương trình ít nhất 1 lần.

## BƯỚC 2: KIỂM THỬ HỘP ĐEN

### Phân lớp tương đương – Equivalence Partitioning

Phân lớp tương đương là một phương pháp kiểm thử hộp đen chia miền đầu vào của một chương trình thành các lớp dữ liệu, từ đó suy dẫn ra các cuộc kiểm thử. Phương pháp này giúp xác định 1 cuộc kiểm thử và đưa ra các lớp lỗi, đồng thời giúp làm giảm tổng số các trường hợp kiểm thử sẽ thiết kế.

Việc thiết kế cuộc kiểm thử cho phân lớp tương đương dựa trên sự đánh giá về một điều kiện đầu vào với các lớp tương đương. Lớp tương đương biểu thị cho tập hợp các trạng thái hợp lệ hay không hợp lệ đối với các điều kiện đầu vào.

Cách xác định tập con này để nhận ra: 1 cuộc kiểm thử được lựa chọn tốt cũng nên có 2 đặc tính khác:

- Giảm thiểu số lượng các cuộc kiểm thử khác mà phải được phát triển để hoàn thành mục tiêu đã định của kiểm thử "hợp lý".
- Bao phủ một tập rất lớn các cuộc kiểm thử có thể khác. Nó nói cho chúng ta một thứ gì đó về sự có mặt hay vắng mặt của những lỗi qua tập giá trị đầu vào cụ thể.

Thiết kế Test-case bằng phân lớp tương đương tiến hành theo 2 bước:

- Xác định các lớp tương đương.
- Xác định các cuộc kiểm thử.

#### Xác định các lớp tương đương

Các lớp tương đương được xác định bằng cách lấy mỗi trạng thái đầu vào và phân chia nó thành 2 hay nhiều nhóm (thường là 1 câu hay 1 cụm từ trong đặc tả).

#### Một mẫu cho việc liệt kê các lớp tương đương

ĐIỀU KIỆN BÊN NGOÀI	CÁC LỚP TƯƠNG ĐƯƠNG HỢP LỆ	CÁC LỚP TƯƠNG ĐƯƠNG KHÔNG HỢP LỆ
------------------------	-------------------------------	-------------------------------------

Nên chú ý là hai kiểu lớp tương đương được xác định:

- Lớp tương đương hợp lệ mô tả các đầu vào hợp lệ của chương trình.
- Lớp tương đương không hợp lệ mô tả tất cả các trạng thái có thể khác của điều kiện. Với một đầu vào hay điều kiện bên ngoài đã cho, việc xác định các lớp tương đương hầu như là một quy trình mang tính kinh nghiệm.

Để xác định các lớp tương đương, có thể áp dụng tập hợp các nguyên tắc sau đây:

1. Nếu 1 trạng thái đầu vào định rõ giới hạn của các giá trị, đồng thời xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương không hợp lệ.

2. Nếu 1 trạng thái đầu vào xác định số giá trị, xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương không hợp lệ.
3. Nếu 1 trạng thái đầu vào chỉ định tập các giá trị đầu vào và chương trình sử dụng mỗi giá trị là khác nhau. Xác định 1 lớp tương đương hợp lệ cho mỗi loại và 1 lớp tương đương không hợp lệ.
4. Nếu 1 trạng thái đầu vào chỉ định một tình huống “chắc chắn – must be”, xác định được 1 lớp tương đương hợp lệ và 1 lớp tương đương không hợp lệ.

Nếu có bất kỳ lý do nào để xác định chương trình không xử lý các phần tử trong cùng 1 lớp là như nhau, thì hãy chia lớp tương đương đó thành các lớp tương đương nhỏ hơn.

#### Xác định các cuộc kiểm thử

Với các lớp tương đương xác định được ở bước trên, bước thứ hai là sử dụng các lớp tương đương đó để xác định các cuộc kiểm thử. Quá trình này thực hiện như sau:

1. Gán 1 số duy nhất cho mỗi lớp tương đương.
2. Cho đến khi tất cả các lớp tương đương hợp lệ được bao phủ bởi các cuộc kiểm thử. Hãy viết 1 cuộc kiểm thử mới bao phủ càng nhiều các lớp tương đương, và chúng chưa được bao phủ lần nào thì càng tốt.
3. Cho đến khi các cuộc kiểm thử đã bao phủ tất cả các lớp tương đương không hợp lệ. Hãy viết 1 cuộc kiểm thử mà bao phủ một và chỉ một trong các lớp tương đương không hợp lệ chưa được bao phủ.
4. Lý do mà mỗi cuộc kiểm thử riêng bao phủ các trường hợp không hợp lệ là vì các kiểm thử đầu vào không đúng nào đó che giấu hoặc thay thế các kiểm thử đầu vào không đúng khác.

Mặc dù việc phân lớp tương đương là rất tốt khi lựa chọn ngẫu nhiên các cuộc kiểm thử, nhưng nó vẫn có những thiếu sót như: bỏ qua các kiểu Test – case có lợi.

hai phương pháp tiếp theo là phân tích giá trị biên và đồ thị nguyên nhân – kết quả, bao phủ được những thiếu sót này.

#### Phân tích giá trị biên – Boundary Value Analysis

Theo nhiều đánh giá cho thấy, các cuộc kiểm thử có khảo sát tỉ mỷ các điều kiện biên có tỷ lệ phản tröm thành công cao hơn các kiểm thử khác.

Các điều kiện biên là những điều kiện mà các tình huống tại chỗ, ở trên và dưới các cạnh của các lớp tương đương đầu vào và các lớp đầu ra.

Phân tích các giá trị biên là phương pháp thiết kế kiểm thử bổ sung thêm cho phân lớp tương đương, nhưng khác với phân lớp tương đương ở 2 khía cạnh nêu ra trong trang bên.

1. Phân tích giá trị biên không lựa chọn phần tử bất kỳ nào trong 1 lớp tương đương là điển hình. Tuy nhiên, phân tích có yêu cầu mỗi cạnh của lớp tương đương đó chính là đối tượng kiểm thử.
2. Ngoài việc chỉ tập trung chú ý vào các trạng thái đầu vào, các cuộc kiểm thử cũng nhận được bằng việc xem xét không gian kết quả (các lớp tương đương đầu ra).

Phân tích giá trị biên yêu cầu có óc sáng tạo và mức chuyên môn hóa nhất định đòi hỏi người kiểm thử có một quá trình và kinh nghiệm rất cao. Tuy nhiên, có một số quy tắc chung như sau:

1. Nếu 1 trạng thái đầu vào định rõ giới hạn của các giá trị, hãy viết các kiểm thử cho các giá trị cuối của giới hạn. Đồng thời, các cuộc kiểm thử đầu vào không hợp lệ cho các trường hợp vượt ra ngoài phạm vi cho phép.
2. Nếu 1 trạng thái đầu vào định rõ số lượng giá trị, hãy viết các cuộc kiểm thử cho con số lớn nhất và nhỏ nhất của các giá trị và 1 giá trị trên, 1 giá trị dưới của chúng.
3. Sử dụng quy tắc 1 cho mỗi trạng thái đầu vào. (việc xem xét giới hạn của kết quả là quan trọng vì không phải lúc nào các biên của miền đầu vào cũng mô tả cùng một tập sự kiện như biên của giới hạn đầu ra. Ngoài ra, không phải lúc nào cũng có thể tạo ra 1 kết quả bên ngoài giới hạn đầu ra, tuy nhiên phải xem xét các vấn đề tiềm ẩn đó).
4. Sử dụng nguyên tắc 2 cho mỗi trạng thái đầu ra.
5. Nếu đầu vào hay đầu ra của một chương trình là tập được sắp thứ tự, tập trung chú ý vào các phần tử đầu tiên và cuối cùng của tập hợp.
6. Sử dụng sự khéo léo, cẩn thận để tìm các điều kiện biên.

### **Đồ thị nguyên nhân và kết quả - Cause & Effect Graphing**

Một yếu điểm của phân tích giá trị biên và phân tích lớp tương đương là không khảo sát sự kết hợp của các trường hợp đầu vào. Việc kiểm thử sự kết hợp đầu vào không phải là một nhiệm vụ đơn giản bởi vì nếu phân lớp tương đương các trạng thái đầu vào, thì số lượng sự kết hợp thường rất lớn. Tuy nhiên, nếu không có cách lựa chọn có hệ thống một tập con các trạng thái đầu vào, sẽ phải chọn ra một tập ngẫu nhiên từ các điều kiện, điều này có thể dẫn tới việc kiểm thử không có hiệu quả.

Đồ thị nguyên nhân và kết quả hỗ trợ trong việc lựa chọn một cách có hệ thống tập các cuộc kiểm thử có hiệu quả cao. Nó tác động có lợi tới việc chỉ ra tình trạng chưa đầy đủ và nhập nhằng trong đặc tả. Đồng thời, cung cấp cả cách biểu diễn chính xác cho các điều kiện logic và hành động tương ứng.

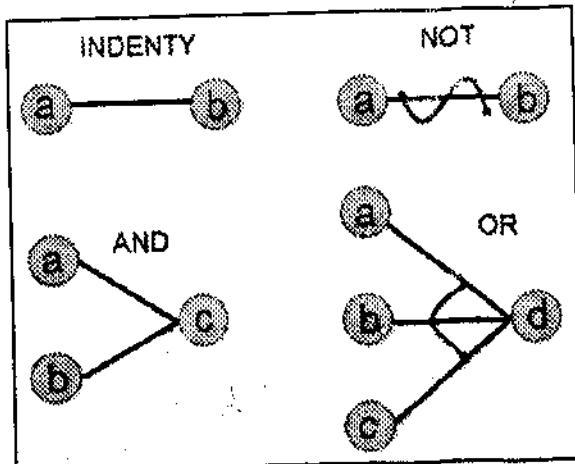
Quá trình dưới đây được sử dụng để xây dựng các Test Case:

- Đặc tả (mô tả lõi) được chia thành các phần có thể thực hiện được. Điều này là cần thiết bởi vì đồ thị nguyên nhân và kết quả trở nên khó sử dụng khi được áp dụng với những đặc tả lớn.
- Nguyên nhân và kết quả nằm trong các đặc tả được nhận biết. Một nguyên nhân là một trạng thái đầu vào nhất định hay một lớp tương đương của các trạng thái đầu vào.

Một kết quả là một trạng thái đầu ra hay 1 sự biến đổi hệ thống (kết quả còn lại mà 1 đầu vào có trạng thái của 1 chương trình hay hệ thống). Có thể nhận biết nguyên nhân và kết quả bằng việc đọc từng nội dung của đặc tả và gạch chân các từ hoặc cụm từ mô tả nguyên nhân và kết quả. Khi đã được nhận biết, mỗi nguyên nhân và kết quả được gán cho 1 số duy nhất.

- Xây dựng đồ thị nguyên nhân và kết quả bằng cách phát triển và biến đổi nội dung ngữ nghĩa của đặc tả thành đồ thị Boolean, nối giữa nguyên nhân và kết quả.
- Đồ thị được dò diễn giải với các ràng buộc mô tả những sự kết hợp của nguyên nhân và/hoặc kết quả là không thể vì các ràng buộc ngữ nghĩa và môi trường.
- Bằng việc dò theo các điều kiện trạng thái trong đồ thị một cách cẩn thận, bạn chuyển đổi đồ thị thành một bảng quyết định mục vào giới hạn. Mỗi cột trong bảng mô tả một ca kiểm thử.
- Các cột trong bảng quyết định được chuyển thành các ca kiểm thử.

Ký hiệu cơ bản cho đồ thị được chỉ ra trong hình sau. Giả sử mỗi nút có giá trị là 0 hoặc 1; 0 mô tả trạng thái vắng mặt và 1 mô tả trạng thái có mặt. Hàm đồng nhất phát biểu như sau: nếu a là 1 thì b là 1; ngược lại b là 0.

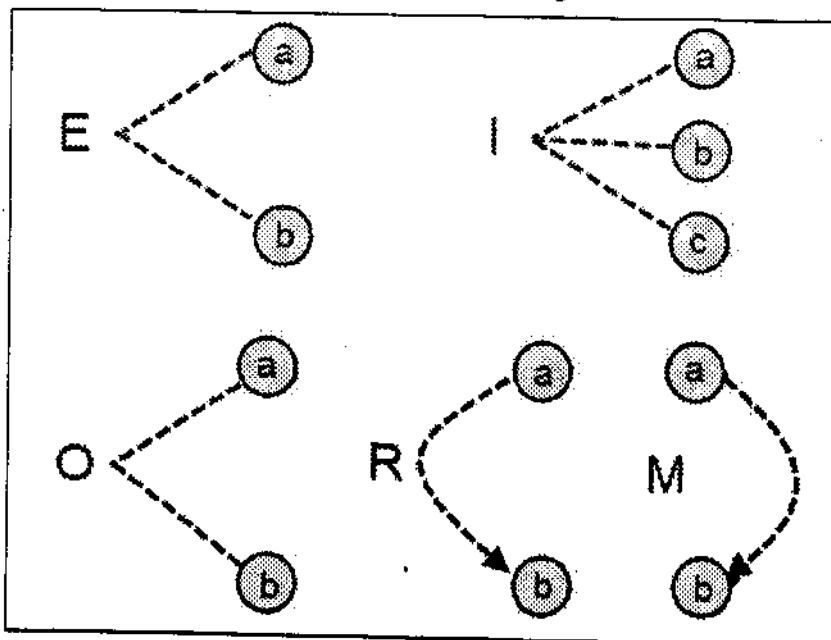


- Hàm NOT khẳng định: nếu a là 1 thì b là 0; ngược lại b là 1.
- Hàm OR khẳng định: nếu a hoặc b hoặc c là 1, thì d là 1; ngược lại d là 0.
- Hàm AND khẳng định: nếu cả a và b là 1 thì c là 1; ngược lại c là 0.
- Hai hàm OR và AND được phép có số lượng đầu vào bất kỳ.

#### Các ký hiệu đồ thị nguyên nhân – kết quả cơ bản

Trong hầu hết các phần mềm, sự kết hợp nào đó của một số nguyên nhân là không thể bởi lý do: ý nghĩa ngôn ngữ và môi trường.

Khi đó, hãy sử dụng ký hiệu ràng buộc trong hình dưới.



- **Ràng buộc E (Exclude – loại trừ)** khẳng định: tối đa, chỉ có hoặc a hoặc b có thể là 1 (a và b không thể đồng thời là 1).
- **Ràng buộc I (Include – bao hàm)** khẳng định: ít nhất một trong a, b hoặc c phải luôn luôn là 1 (a, b hoặc c không thể đồng thời là 0).
- **Ràng buộc O (Only – chỉ một)** khẳng định: một và chỉ một hoặc a hoặc b phải là 1.
- **Ràng buộc R (Request – yêu cầu)** khẳng định: khi a là 1, thì b phải là 1 (không thể có trường hợp a là 1, còn b là 0).
- **Ràng buộc M (Mask – mặt nạ)** khẳng định: nếu kết quả a là 1, kết quả b sẽ bắt buộc phải là 0.

Bước tiếp theo là tạo bảng quyết định mục vào giới hạn – Limited-entry decision table. Tương tự với các bảng quyết định, thì nguyên nhân chính là các điều kiện và kết quả chính là các hành động.

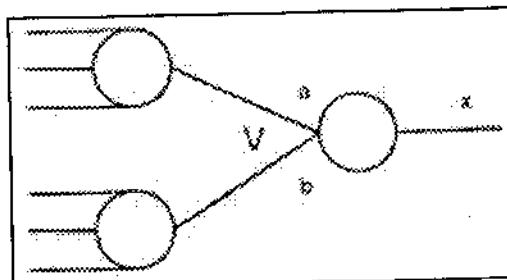
**Quy trình được sử dụng như sau:**

- Chọn một kết quả là trạng thái có mặt.
- Quay ngược trở lại đồ thị, tìm tất cả những sự kết hợp của các nguyên nhân (đối tượng cho các ràng buộc) mà sẽ nhóm kết quả này thành 1.
- Tạo một cột trong bảng quyết định cho mỗi sự kết hợp nguyên nhân.
- Với mỗi sự kết hợp, hãy quy định trạng thái của tất cả các kết quả khác và đặt chúng vào mỗi cột.

Trong khi biểu diễn bước 2, cần quan tâm các vấn đề sau:

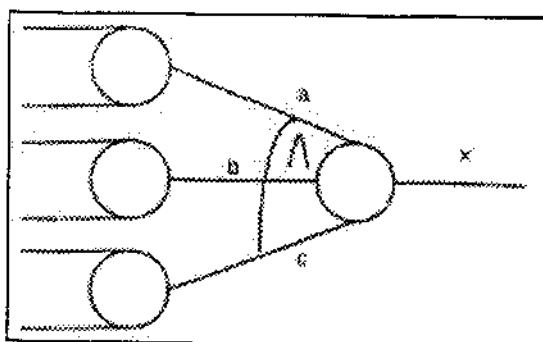
- Khi lần ngược trở lại qua một nút OR mà đầu ra của nó là 1, không bao giờ thiết lập nhiều hơn 1 đầu vào cho nút OR là 1 một cách đồng thời. Điều này được gọi là Path Sensitizing – làm nhạy đường đi. Mục tiêu của nó là để ngăn chặn dò lỗi thất bại vì một nguyên nhân che đi một nguyên nhân khác.
- Khi lần ngược trở lại qua một nút AND mà đầu ra của nó là 0, để nhiên, phải liệt kê tất cả các sự kết hợp đầu vào dẫn tới đầu ra 0. Tuy nhiên, nếu bạn đang khảo sát trạng thái mà một đầu ra là 0 và một hay nhiều đầu ra khác là 1, thì không nhất thiết phải liệt kê tất cả các điều kiện mà dưới điều kiện đó các đầu vào khác có thể là 1.
- Khi lần ngược trở lại qua một nút AND mà đầu ra của nó là 0, chỉ cần liệt kê 1 điều kiện trong đó tất cả đầu vào bằng 0. (Nếu nút AND ở chính giữa của đồ thị thì tất cả các đầu vào của nó xuất phát từ các nút trung gian khác, có thể có quá nhiều trạng thái mà trong trạng thái đó tất cả các đầu vào của nó bằng 0.)

**Những xem xét được sử dụng khi dò theo đồ thị**



- Nếu  $x=1$ , không quan tâm về trường hợp  $a=b=1$ .
- Nếu  $x=0$ , liệt kê tất cả các trường hợp trong đó  $a=b=0$ .
- Nếu  $x=1$ , liệt kê tất cả các trường hợp trong đó  $a=b=c=1$ .

- Nếu  $x=0$ , bao gồm chỉ 1 trường hợp mà  $a=b=c=0$ . Đối với các trạng thái mà abc là 001, 010, 100, 011, 101 và 110 bao gồm chỉ 1 trường hợp cho mỗi trạng thái.



Những sự xem xét này có thể xuất hiện thất thường, nhưng chúng có một mục đích rất quan trọng:

- Để giảm bớt các kết quả được kết hợp của đồ thị, liệt kê các trường hợp hướng về các cuộc kiểm thử ít có lợi.
- Nếu các ca kiểm thử ít có lợi không được liệt kê, một đồ thị nguyên nhân – kết quả lớn sẽ tạo ra một số lượng ca kiểm thử cực kỳ lớn. Nếu số lượng các cuộc kiểm thử trên thực tế là quá lớn, sẽ chọn ra 1 tập con nào đó, nhưng không đảm bảo là các cuộc kiểm thử ít có lợi sẽ là những cuộc kiểm thử được liệt kê. Do đó, tốt hơn hết là liệt kê chúng trong suốt quá trình phân tích của đồ thị.

## KẾT LUẬN

- Vẽ đồ thị nguyên nhân – kết quả là phương pháp tạo các ca kiểm thử có hệ thống mô tả sự kết hợp của các điều kiện. Sự thay đổi sẽ là một sự lựa chọn kết hợp không thể dự tính trước, nhưng khi thực hiện như vậy, có vẻ như sẽ bỏ sót nhiều cuộc kiểm thử “thú vị” được xác định bằng đồ thị nguyên nhân – kết quả.
- Vẽ đồ thị nguyên nhân – kết quả yêu cầu chuyển một đặc tả thành một mạng logic Boolean, nó cung cấp một triển vọng khác và sự hiểu biết sâu sắc hơn nữa về đặc tả. Trên thực tế, sự phát triển của một đồ thị nguyên nhân – kết quả là cách hay để khám phá sự mơ hồ và chưa đầy đủ trong các đặc tả.
- Mặc dù việc vẽ đồ thị nguyên nhân – kết quả tạo ra tập các ca kiểm thử hữu dụng, nhưng thông thường nó không tạo ra tất cả các ca kiểm thử hữu dụng mà có thể được nhận biết.

- **Đồ thị nguyên nhân – kết quả** không khảo sát thỏa đáng các điều kiện giới hạn. Dĩ nhiên, bạn có thể cố gắng bao phủ các điều kiện giới hạn trong suốt quá trình. Tuy nhiên, vấn đề trong việc thực hiện điều này là nó làm cho đồ thị rất phức tạp và dẫn tới số lượng rất lớn các ca kiểm thử. Vì thế, tốt nhất là xét một sự phân tích giá trị giới hạn tách rời nhau.
- **Đồ thị nguyên nhân – kết quả** làm chúng ta mất thời gian trong việc chọn các giá trị cụ thể cho các toán hạng, nên các điều kiện giới hạn có thể bị pha trộn thành các cuộc kiểm thử xuất phát từ đồ thị nguyên nhân – kết quả. Vì vậy, chúng ta đạt được một tập các ca kiểm thử nhỏ nhưng hiệu quả mà thỏa mãn cả 2 mục tiêu.
- Nên chú ý là việc vẽ đồ thị nguyên nhân – kết quả phù hợp với một số quy tắc trong giới thiệu phần đầu trong sách. Việc xác định đầu ra mong đợi cho mỗi ca kiểm thử là một phần cố hữu của kỹ thuật (mỗi cột trong bảng quyết định biểu thị các kết quả được mong đợi). Cũng chú ý là nó khuyến khích việc tìm kiếm các kết quả có tác dụng không mong muốn
- Khía cạnh khó nhất của kỹ thuật này là quá trình chuyển đổi đồ thị thành bảng quyết định. Quá trình này có tính thuật toán, tức là có thể tự động hóa nó bằng việc viết một chương trình. Trên thị trường đã có một vài chương trình thương mại giúp cho quá trình chuyển đổi này.

### 3. ĐOÁN LỖI – ERROR GUESSING

Một kỹ thuật thiết kế Test-case khác là Error Guessing – Đoán lỗi. Kiểm thử viên được giao cho một chương trình đặc biệt, họ thực hiện phỏng đoán bằng trực giác và kinh nghiệm. Tìm các loại lỗi có thể tìm được và sau đó thực hiện viết các cuộc kiểm thử để đưa ra các lỗi đó.

Rất khó khăn để đưa ra một quy trình cho kỹ thuật đoán lỗi vì nó là một quy trình có tính trực giác cao và không thể dự đoán trước. Ý tưởng cơ bản là liệt kê một danh sách các lỗi có thể hay các trường hợp dễ xảy ra lỗi và sau đó viết các cuộc kiểm thử dựa trên danh sách đó.

Một cách khác để xác định các cuộc kiểm thử có liên đới với các giả định mà lập trình viên có thể đã thực hiện khi đọc đặc tả (tức là, những thứ bị bỏ sót khỏi đặc tả, hoặc là do tình cờ, hoặc là vì người viết có cảm giác những đặc tả đó là rõ ràng). Nói cách khác, bạn liệt kê những trường hợp đặc biệt đó có thể đã bị bỏ sót khi chương trình được thiết kế.

### 4. CHIẾN LƯỢC

Các phương pháp thiết kế Test-case đã được thảo luận có thể được kết hợp thành một chiến lược toàn diện.

Vì mỗi phương pháp có thể đóng góp 1 tập riêng các cuộc kiểm thử hữu dụng, nhưng không cái nào trong số chúng tự đóng góp một tập hợp trọn vẹn các cuộc kiểm thử.

**Chiến lược hợp lý** như sau:

1. Nếu đặc tả có chứa sự kết hợp của các điều kiện đầu vào, hãy bắt đầu với việc vẽ đồ thị nguyên nhân – kết quả.
2. Trong trường hợp bất kỳ, sử dụng phương pháp phân tích giá trị biên. Hãy nhớ rằng đây là một sự phân tích của các biến đầu vào và đầu ra. Phương pháp phân tích giá trị biên mang lại một tập các điều kiện kiểm thử bổ sung, và rất nhiều hay toàn bộ các điều kiện này có thể được hợp nhất thành các kiểm thử nguyên nhân – kết quả.
3. Xác định các lớp tương đương hợp lệ và không hợp lệ cho đầu vào và đầu ra, bổ sung các ca kiểm thử được xác định trên nếu cần thiết.
4. Sử dụng kỹ thuật đoán lỗi để thêm các ca kiểm thử thêm vào.

Khảo sát tính logic của chương trình liên quan đến tập các ca kiểm thử. Sử dụng tiêu chuẩn bao phủ quyết định, bao phủ điều kiện, bao phủ quyết định/điều kiện, hay bao phủ đa điều kiện (trong đó bao phủ đa điều kiện là được sử dụng nhiều nhất).

Nếu tiêu chuẩn bao phủ không đạt được bởi các cuộc kiểm thử được xác định trong bốn bước trước, và nếu bản chất của chương trình làm những sự kết hợp chắc chắn của các điều kiện không thể đạt được tiêu chuẩn, hãy thêm vào các ca kiểm thử có khả năng làm cho thỏa mãn tiêu chuẩn.

Tuy việc sử dụng những chiến lược này không đảm bảo rằng tất cả các lỗi sẽ được tìm thấy, nhưng nó là một sự thỏa thuận hợp lý.

**CHƯƠNG 9****ỨNG DỤNG KIỂM THỬ****CHƯƠNG TRÌNH MINH HỌA****Xây dựng Test Case một chương trình trong Java**

Chương trình đọc vào 3 giá trị nguyên từ hộp thoại vào. Ba giá trị này tương ứng với chiều dài 3 cạnh của 1 tam giác. Chương trình hiển thị 1 thông điệp cho biết tam giác đó là tam giác thường, cân, hay đều. Ba giá trị nhập vào thỏa mãn là 3 cạnh của một tam giác khi và chỉ khi cả 3 số đều là số nguyên dương, và tổng của 2 số bất kỳ trong 3 số phải lớn hơn số thứ 3. Khi đó, một tam giác đều là tam giác có 3 cạnh bằng nhau, tam giác cân là tam giác có 2 trong 3 cạnh bằng nhau, và tam giác thường thì có 3 cạnh khác nhau.

**MÃ LỆNH CỦA CHƯƠNG TRÌNH:**

```

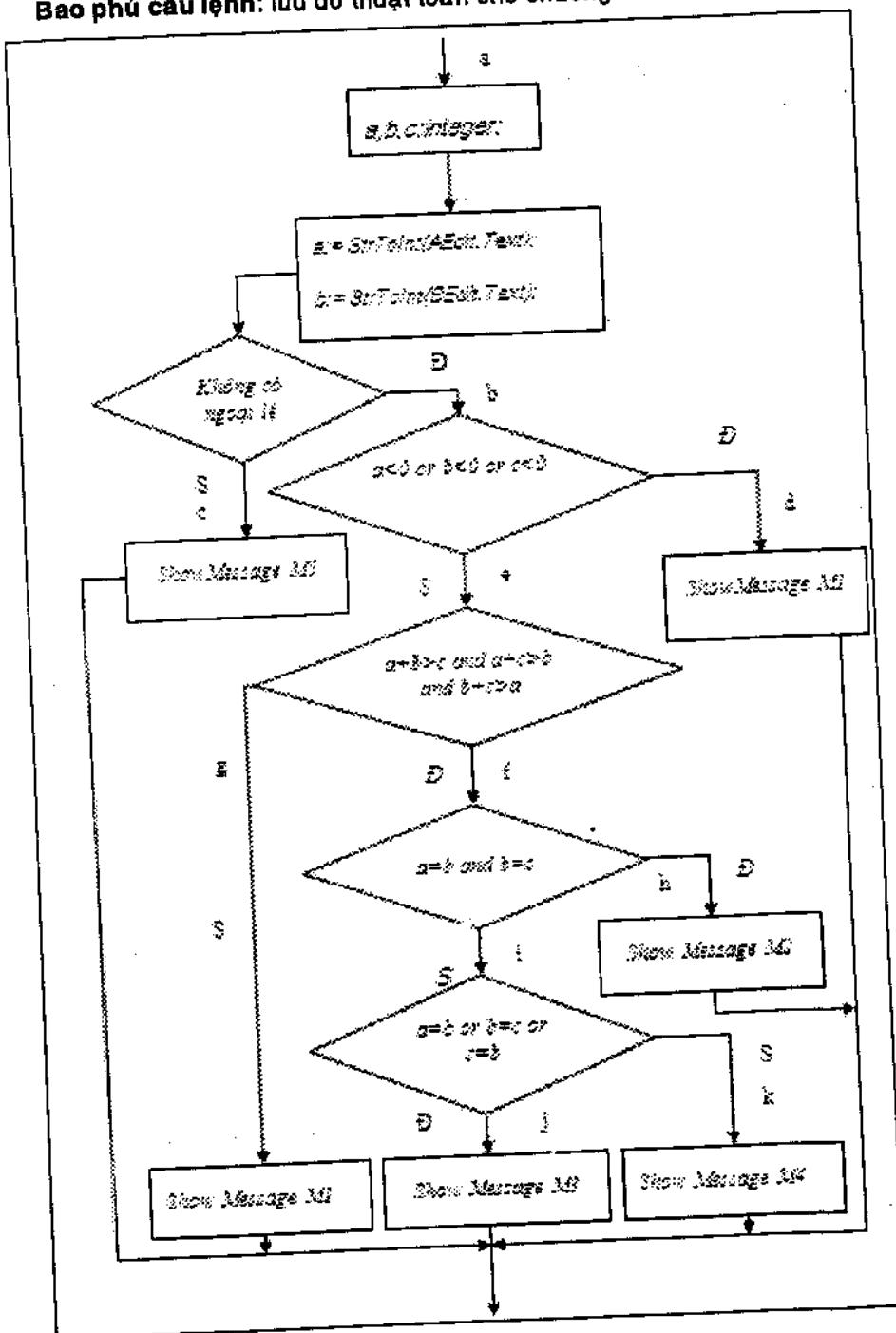
unit main;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, ExtCtrls;
type
  TMainForm = class(TForm)
    AEdit: TLabaledEdit;
    BEdit: TLabaledEdit;
    CEdit: TLabaledEdit;
    btnTest: TButton;
    procedure btnTestClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  MainForm: TMainForm;
implementation

```

```
{$R *.dfm}
Procedure TMainForm.btnTestClick(Sender: TObject);
var
  a, b, c: Integer;
begin
  try
    a := StrToInt(AEdit.Text);
    b := StrToInt(BEdit.Text);
    c := StrToInt(CEdit.Text);
    if (a < 0) Or (b < 0) Or (c < 0) then
      ShowMessage('3 canh A, B, C khong theo man
la 3 canh cua mot tam giac.')
    else
      if (a + b > c) And (a + c > b) And (b + c > a) then
        begin
          if (a = b) And (b = c) then
            ShowMessage('3 canh A, B, C lap thanh
mot tam giac deu.')
          else
            if (a=b) Or (b=c) Or (c=b) then
              ShowMessage('3 canh A, B, C lap
thanh mot tam giac can.')
            else
              ShowMessage('3 canh A, B, C lap
thanh mot tam giac thuong.');
        end
      else
        ShowMessage('3 canh A, B, C khong theo man
la 3 canh cua mot tam giac.');
    except
      ShowMessage('Loi dinh dang du lieu. De nghi ban
xem va nhap lai.');
    end;
  end;
end.
```

## CÁC PHƯƠNG PHÁP HỘP TRẮNG

Bao phủ câu lệnh: lưu đồ thuật toán cho chương trình tam giác:



Trong đó:

- M1: Ba cạnh A, B, C không thỏa mãn là 3 cạnh của 1 tam giác.
- M2: Ba cạnh A, B, C lập thành 1 tam giác đều.
- M3: Ba cạnh A, B, C lập thành 1 tam giác cân.
- M4: Ba cạnh A, B, C lập thành 1 tam giác thường.
- M5: Lỗi định dạng dữ liệu. Đề nghị xem và nhập lại.

Các ca kiểm thử thu được:

- |    |           |                |           |
|----|-----------|----------------|-----------|
| 1. | -1, 1, 1  | và các hoán vị | (abdl)    |
| 2. | 1, 1, 1   | và các hoán vị | (abefhl)  |
| 3. | 2, 2, 1   | và các hoán vị | (abelijl) |
| 4. | 2, 3, 4   | và các hoán vị | (abefikl) |
| 5. | 1, 2, 4   | và các hoán vị | (abegl)   |
| 6. | A, 1, 1   | và các hoán vị | (aci)     |
| 7. | 1.1, 1, 1 | và các hoán vị | (a,c,l)   |

### Bao phủ quyết định

Các ca kiểm thử thu được:

- |    |                        |                |           |
|----|------------------------|----------------|-----------|
| 1. | -32768, -32768, -32768 | và các hoán vị | (abdl)    |
| 2. | 32767, 32767, 32767    | và các hoán vị | (abefhl)  |
| 3. | 32767, 32767, 327676   | và các hoán vị | (abelijl) |
| 4. | 32767, 32766, 32765    | và các hoán vị | (abefikl) |
| 5. | 32767, 1, 2            | và các hoán vị | (abegl)   |
| 6. | A, 1, 1                | và các hoán vị | (aci)     |
| 7. | 1.1, 1, 1              | và các hoán vị | (a,c,l)   |

### Bao phủ điều kiện

Các ca kiểm thử thu được là:

- |    |           |                 |
|----|-----------|-----------------|
| 1. | -1, 1, 1  | và các hoán vị. |
| 2. | 2, 3, 4   | và các hoán vị. |
| 3. | 1, 2, 4   | và các hoán vị. |
| 4. | 2, 2, 1   | và các hoán vị. |
| 5. | 1, 1, 1   | và các hoán vị. |
| 6. | A, 1, 1   | và các hoán vị. |
| 7. | 1.1, 1, 1 | và các hoán vị. |

**Bao phủ quyết định – điều kiện**

Các ca kiểm thử thu được là:

1. -1, 1, 1 và các hoán vị.
2. 2, 3, 4 và các hoán vị.
3. 1, 2, 4 và các hoán vị.
4. 2, 2, 1 và các hoán vị.
5. 1, 1, 1 và các hoán vị.
6. A, 1, 1 và các hoán vị.
7. 1.1, 1, 1 và các hoán vị.

**Bao phủ đa điều kiện**

Các ca kiểm thử thu được là:

1. -1, 1, 1 và các hoán vị.
2. 2, 3, 4 và các hoán vị.
3. 1, 2, 4 và các hoán vị.
4. 2, 2, 1 và các hoán vị.
5. 1, 1, 1 và các hoán vị.
6. A, 1, 1 và các hoán vị.
7. 1.1, 1, 1 và các hoán vị.

**Vẽ đồ thị nguyên nhân – kết quả**

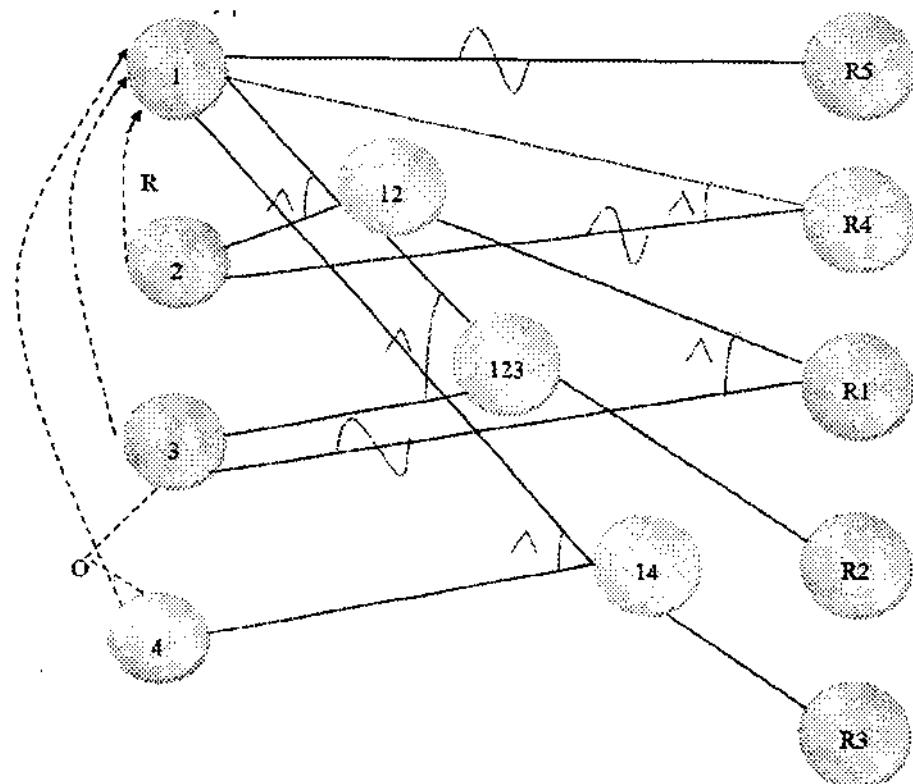
Do đặc tả có sự kết hợp đầu vào nên trước tiên, áp dụng phương pháp vẽ đồ thị nguyên nhân – kết quả.

**Nguyên nhân:**

1. Cả 3 giá trị nhập vào đều là số nguyên dương.
2. Tổng 2 số bất kỳ trong 3 số lớn hơn số còn lại.
3. Hai trong 3 số có giá trị bằng nhau.
4. Ba số có giá trị bằng nhau.

**Kết quả:**

1. R1. Thông báo ba giá trị nhập vào lập thành tam giác thường.
2. R2. Thông báo ba giá trị nhập vào lập thành tam giác cân.
3. R3. Thông báo ba giá trị nhập vào lập thành tam giác đều.
4. R4. Thông báo ba giá trị nhập vào không lập thành một tam giác.
5. R5. Thông báo lỗi nhập dữ liệu.

**ĐỒ THỊ NGUYÊN NHÂN - KẾT QUẢ:**

Bước tiếp theo là tạo bảng quyết định mục vào giới hạn.

Chọn kết quả R1 là đầu tiên. R1 có mặt nếu nút các nút 12 và 3 = 1,0.  
Nút 12 = 1 khi 1 và 2 = 1,1.

Áp dụng lần lượt cho sự có mặt của từng kết quả đầu vào, ta được bảng quyết định như sau:

	1	2	3	4	5
1	1	1	1	1	0
2	1	1		0	
3	0	1			
4			1		

R1	1	0	0	0	0
R2	0	1	0	0	0
R3	0	0	1	0	0
R4	0	0	0	1	0
R5	0	0	0	0	1

Bước cuối cùng là chuyển đổi bảng quyết định thành các cuộc kiểm thử. Các ca kiểm thử thu được như sau:

STT	Các điều kiện	Ca kiểm thử	Hành động
1	Cả 3 giá trị nhập vào đều là số nguyên dương, và tổng của 2 số bất kỳ trong 3 số luôn lớn hơn số thứ 3, và không có cặp 2 số bất kỳ nào trong 3 số đó là = nhau.	2,3,4 2,4,3 3,2,4 3,4,2 4,2,3 4,3,2	R1
2	Cả 3 giá trị nhập vào đều là số nguyên dương, và tổng của 2 số bất kỳ trong 3 số luôn lớn hơn số thứ 3, và tồn tại một cặp 2 số trong 3 số đó là = nhau.	3,3,4 3,4,3 4,3,3	R2
3	Cả 3 giá trị nhập vào đều là số nguyên dương, và cả 3 số có giá trị bằng nhau.	3,3,3	R3
4	Cả 3 giá trị nhập vào đều là số nguyên dương, và tồn tại 2 số trong 3 số có tổng nhỏ hơn hoặc bằng số còn lại.	1,2,4 Và 5 hoán vị của nó	R4
5	Tồn tại một giá trị nhập vào không phải là số nguyên dương.	A,2,2 -1,1,1 1,1,1,1 Và 2 hoán vị của mỗi trường hợp	R5

**PHÂN LỚP TƯƠNG ĐƯƠNG****Xác định các lớp tương đương**

Các giá trị nhập vào là số.	Cả 3 giá trị đều là số (1).	Tồn tại 1 giá trị không phải là số (2).
Các giá trị là nguyên.	Cả 3 giá trị đều nguyên (3).	Tồn tại 1 giá trị không nguyên (4).
Các giá trị là dương.	Cả 3 giá trị đều dương (5).	Tồn tại 1 giá trị $\leq 0$ (6).
Hằng số.	-32768 : 32767 (7).	<-32768 (8), >32767 (9).
Tổng 2 số bất kỳ so với số thứ 3.	Lớn hơn (10).	Nhỏ hơn hoặc bằng (11).

## CHƯƠNG 10

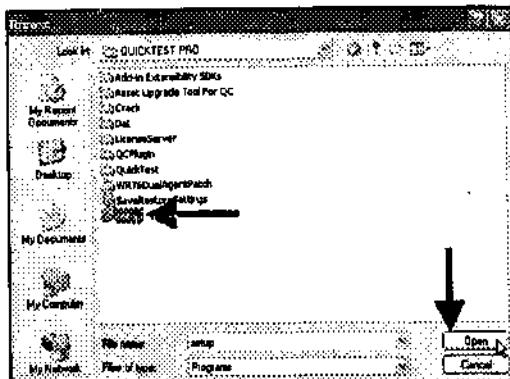
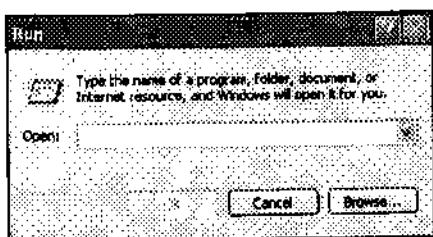
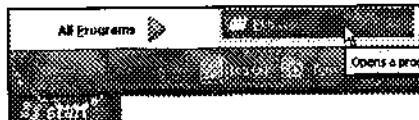
# KIỂM THỬ VỚI QUICKTEST PROFESSIONAL 10.0

Hiện nay, quá trình kiểm thử được chia ra nhiều công đoạn và các kiểm thử viên được hỗ trợ bằng những phần mềm chuyên dùng cho việc kiểm thử như: LOADRUNNER, JUNIT, ROBOT... một phần mềm kiểm thử trong số đó hay được áp dụng cho các ứng dụng Web, Vba là QUICKTEST PRO. Phần trình bày sau giới thiệu cách sử dụng phần mềm này.

## CÁC BƯỚC HƯỚNG DẪN CÀI ĐẶT CHƯƠNG TRÌNH:

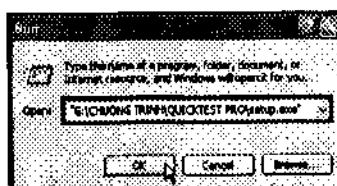
Trước khi thực hiện cài đặt chương trình, cần kiểm tra cấu hình máy có thỏa một số yêu cầu của chương trình hay không (tham khảo trên [www.stkbook.com](http://www.stkbook.com) hay trang web của chính hãng). Các bạn cần chuẩn bị gói cài đặt chương trình trong máy tính (có thể dùng chương trình đĩa kèm trong đĩa DVD theo sách hoặc tải về trên mạng).

Trên màn hình Windows nhấp chọn Start > Run. Hộp thoại Run xuất hiện, nhấn chọn nút Browse... để mở file cài đặt chương trình.



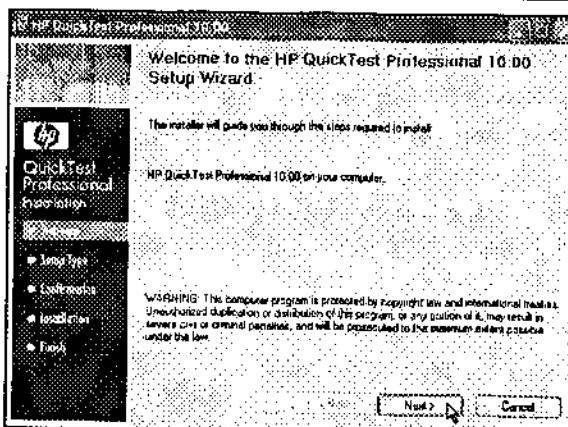
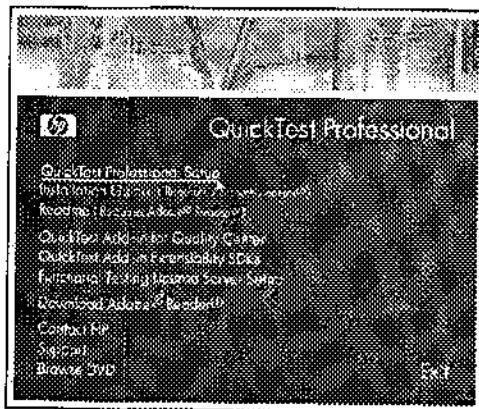
Hộp thoại Browse xuất hiện, chỉ đường dẫn đến file cài đặt Setup.exe sau đó nhấp nút Open như hình bên.

Trở về hộp thoại Run, trong khung Open lúc này xuất hiện đường dẫn của file cài đặt chương trình. Nhập OK để bắt đầu cài đặt chương trình.

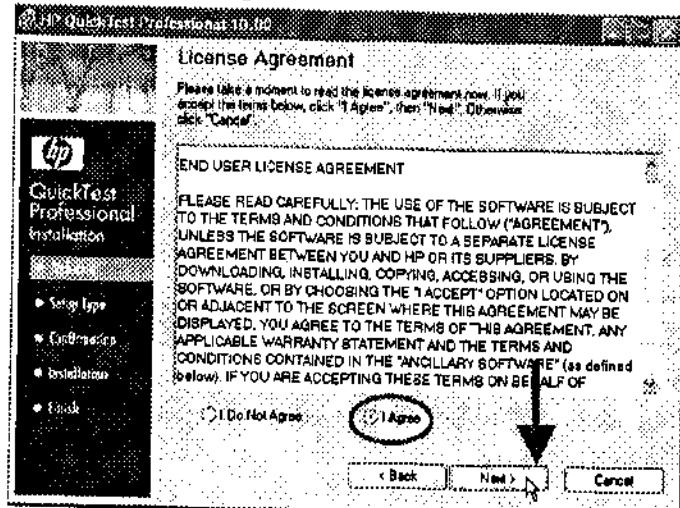


Hộp thoại QuickTest Professional xuất hiện với các tùy chọn, nhấp chọn dòng đầu tiên QuickTest Professional Setup để bắt đầu tiến trình cài đặt.

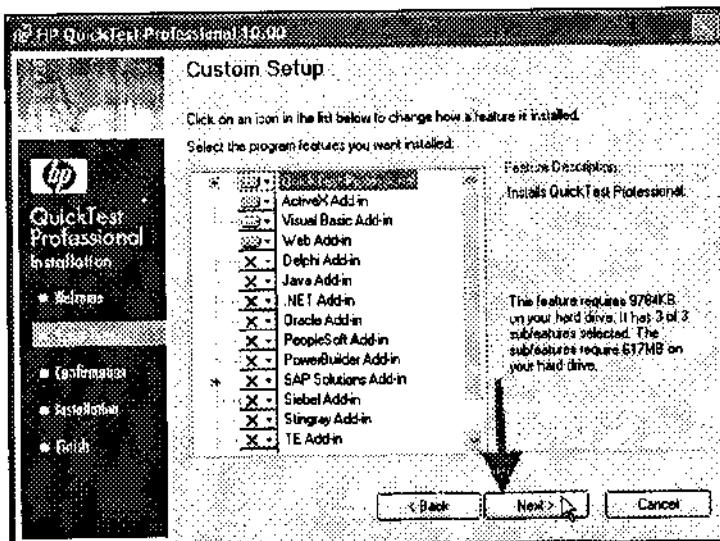
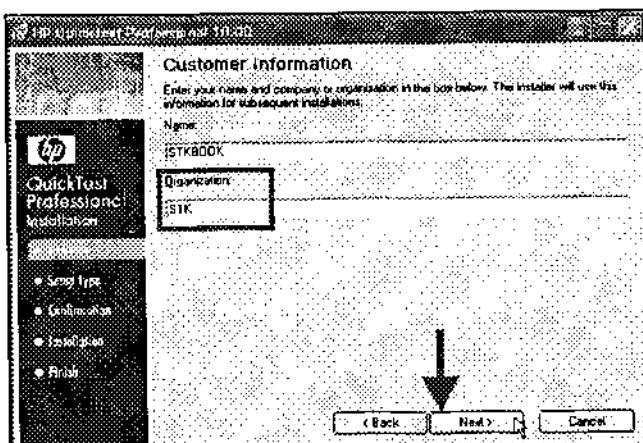
**Hộp thoại Welcome to the HP QuickTest Professional 10.00 Setup Wizard** xuất hiện, nhấp chọn Next để cài đặt.



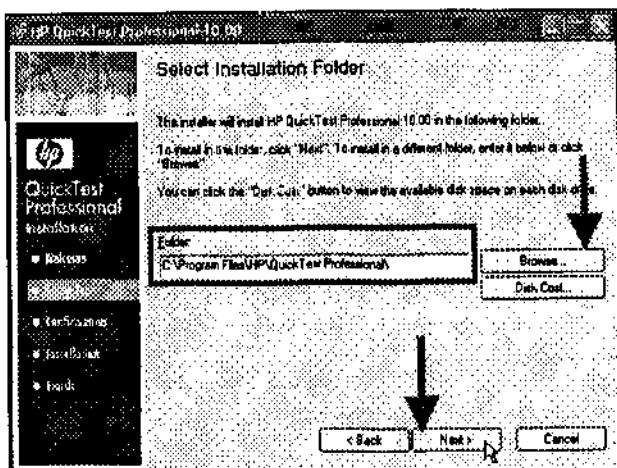
Hộp thoại License Agreement xuất hiện như hình dưới, đánh dấu kiểm vào trước mục "I Agree" chấp nhận các điều khoản cài đặt sau đó nhấp nút Next để tiếp tục cài đặt chương trình.



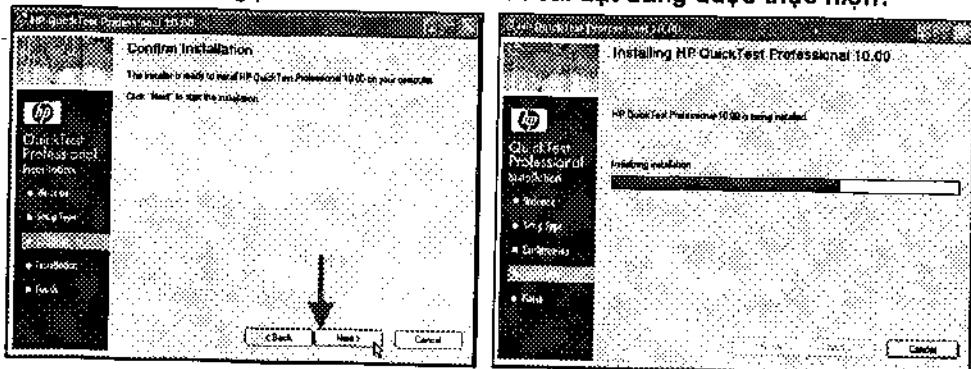
Hộp thoại **Customer Information** xuất hiện, nhập tên và tổ chức trong mục **Name** và **Organization**. Nhập xong nhấp nút **Next** tiếp tục cài đặt. Hộp thoại **Custom Setup** xuất hiện, nhấp chọn các thành phần muốn cài đặt sau đó nhấp nút **Next**.



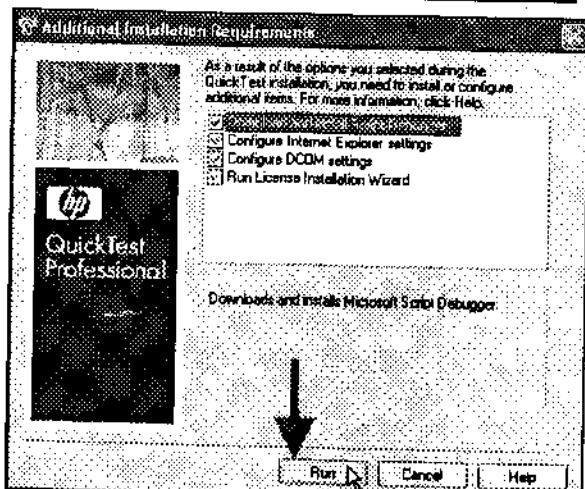
Hộp thoại **Select Installation Folder** xuất hiện, nhấp nút **Browse** chọn đường dẫn lưu file thực hiện của chương trình. Ở đây ta chọn đường dẫn mặc định, sau đó nhấp nút **Next** để tiếp tục cài đặt.



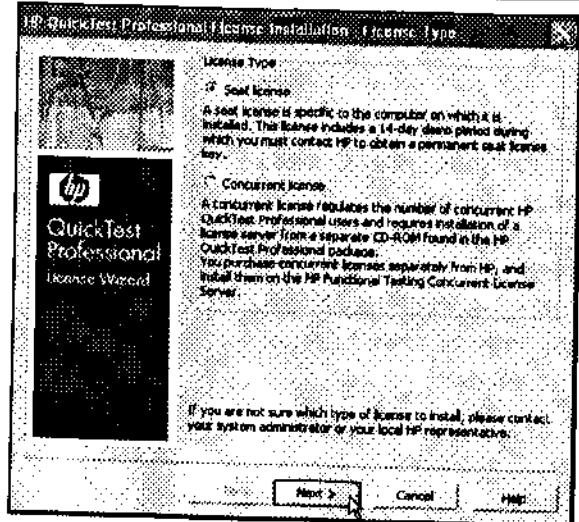
**Hộp thoại Confirm Installation** xuất hiện, nhấp nút Next. Một vệt sáng lan dần từ trái sang phải cho biết tiến trình cài đặt đang được thực hiện.



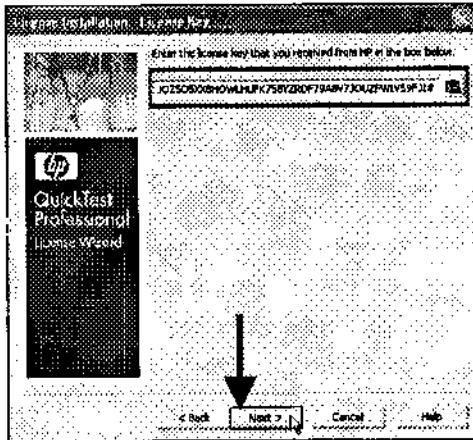
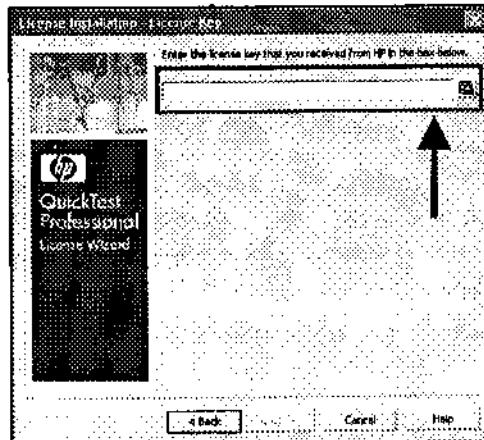
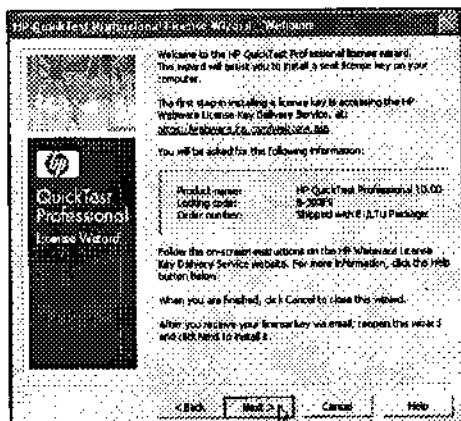
**Hộp thoại Additional Installation Requirements** xuất hiện, chọn các chế độ cài đặt (ở đây ta chọn hết) sau đó nhấp nút Run.



**Hộp thoại License Type** xuất hiện, nhấp chọn chế độ **Seat license** để thiết lập quyền sử dụng phần mềm. Nhấp nút Next để tiếp tục thực hiện cài đặt.



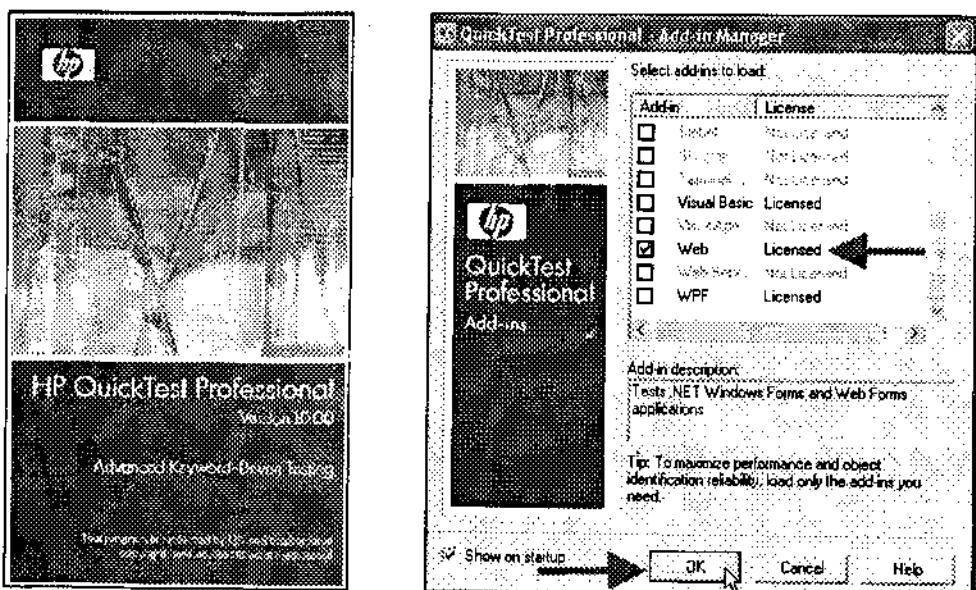
Hộp thoại **Welcome** xuất hiện, cho biết thông tin về chương trình. Nhấp nút **Next** để thực hiện bước kế tiếp. Hộp thoại **License Key** xuất hiện, nhập License Key trong gói cài đặt của chương trình. Hãy copy và dán License Key vào khung **License Key** như hình dưới sau đó nhấp nút **Next** hoàn thành cài đặt phần mềm.



Lúc này bạn có thể khởi động chương trình QuickTest Professional 10.0 (phiên bản 2010) để dùng bằng cách chọn: **Start > All Programs > QuickTest Professional > QuickTest Professional**.

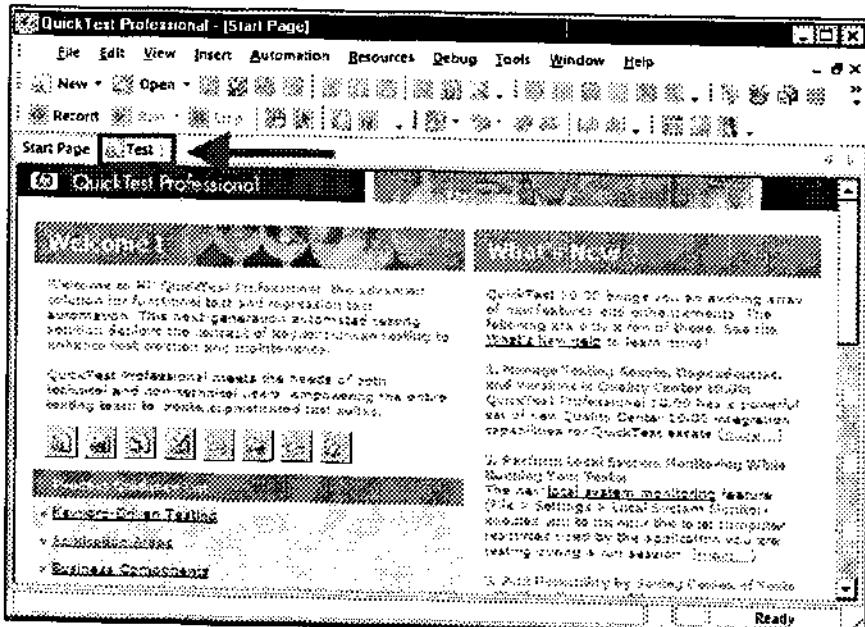


Một cửa sổ thông báo của của chương trình xuất hiện sau đó xuất hiện hộp thoại quản lý Add – in Manager, nhấp chọn Add In muốn dùng, sau đó nhấp chọn nút OK (ở đây, ta nhấp chọn Add-in Web).



Khi đã chọn một Add-in để làm việc, chương trình sẽ ghi nhớ và trong phiên làm việc sau tùy chọn này sẽ được đặt mặc định. Các Add-in được chương trình hỗ trợ sẽ nằm trong danh sách. Nếu muốn kích hoạt chúng, hãy thực hiện theo các hướng dẫn HP QuickTest của chương trình.

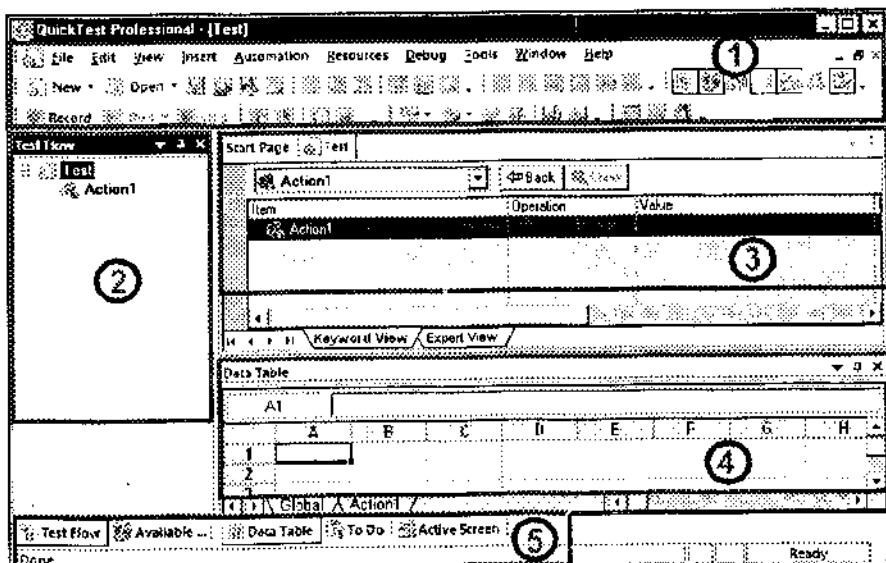
Giao diện làm việc chương trình xuất hiện, phần hiển thị là phần giới thiệu các thông tin mới của chương trình. Nhấp chọn tab **Test** để chuyển sang thực hiện một quá trình kiểm thử mới.



Trong tab **Start Page** có thể thực hiện các thao tác như sau:

- Nhấp chọn vào các link liên kết để có thể đọc các hướng dẫn sử dụng chương trình. Nhấp chọn danh sách các quy trình hướng dẫn trong mục **Process Guidance List**.
- Nhấp vào các phím tắt để mở một quá trình kiểm thử mới hoặc dùng các phần tùy chọn ở vùng chức năng.
- Nhấp chọn vào phần **What's New** để tìm hiểu các điểm mới của chương trình.

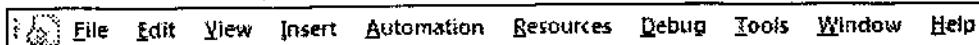
Khi nhấp chọn tab **Test**, giao diện làm việc của chương trình xuất hiện bao gồm 5 khu vực như sau:



- Khu vực 1:** thanh tiêu đề, menu và các công cụ tùy chỉnh.
- Khu vực 2:** bảng Test Flow kiểm tra quá trình kiểm thử.
- Khu vực 3:** bảng hiển thị thông tin kiểm thử.
- Khu vực 4:** màn hình hiển thị bảng dữ liệu hoặc các chế độ khác tùy vào các tùy chọn hiển thị ở khu vực 5.
- Khu vực 5:** hiển thị các tab tùy chọn làm việc.

#### KHU VỰC 1:

Gồm hệ thống các menu chính chứa các lệnh của chương trình với 9 menu như sau: **File, Edit, View, Insert, Automation, Resources, Debug, Tools, Window, Help**.



Phần kế tiếp, là thanh các công cụ mở gồm các button tùy chọn khi thực hiện việc kiểm thử phần mềm.

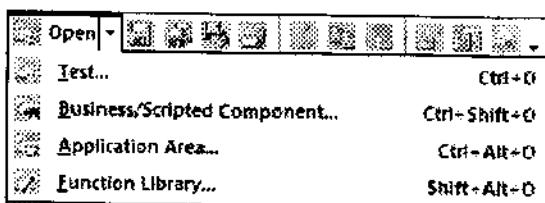
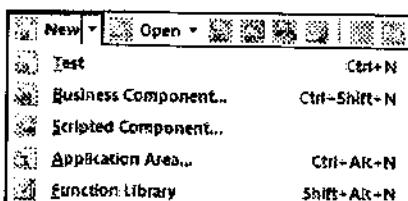


Các hệ Button hỗ trợ kiểm thử nhanh được trích từ thanh Menu lệnh có công dụng như sau:

Thanh công cụ **Standard**: có các nút hỗ trợ việc quản lý tài liệu trích từ menu File.



Nhấp thả danh sách bên trong thanh công cụ sẽ có thể kích hoạt các lệnh khác và phím tắt chức năng của nó như hình sau:



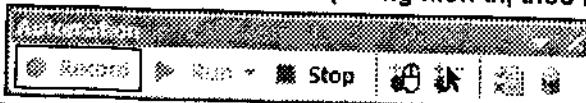
Bảng thống kê lệnh của thanh Standard:

	Lệnh	Phím tắt	Chức năng
	New > Test	CTRL+N	Tạo một kiểm thử mới.
	New > Business Component	CTRL+SHIFT+N	Tạo một thành phần kinh doanh mới.
	New > Scripted Component		Tạo một kịch bản mới.
	New > Application Area	CTRL+ALT+N	Tạo khu vực ứng dụng.
	New > Function Library	SHIFT+ALT+N	Tạo thư viện chức năng.
	Open > Test	CTRL+O	Mở một ứng dụng hiện có.
	Open > Business/Scripted Component	CTRL+SHIFT+O	Mở một thành phần kinh doanh và kịch bản mới.

	<b>Open &gt; Application Area</b>	<b>CTRL+ALT+O</b>	Mở một khu vực ứng dụng hiện có.
	<b>Open &gt; Function Library</b>	<b>SHIFT+ALT+O</b>	Mở một thư viện chức năng hiện có.
	<b>Close</b>		Đóng thư viện hiện tại.
	<b>Close All Function Libraries</b>		Đóng tất cả các thư viện chức năng.
	<b>Quality Center Connection</b>		Mở một kết nối với trung tâm chất lượng cho phép kết nối với trung tâm.
	<b>Quality Center Version Control</b>		Phiên bản kết nối từ trung tâm chất lượng, cung cấp một trình đơn phụ tùy chọn cho việc quản lý phiên bản của QuickTest các tài sản tại trung tâm chất lượng. Trình đơn phụ có sẵn khi kết nối được với trung tâm chất lượng.
	<b>Save</b>	<b>CTRL+S</b>	Lưu trữ các tài liệu đang mở, đang thực thi.
	<b>Save As</b>		Mở hộp thoại Save có liên quan để lưu các tài liệu mở.
	<b>Save Test with Resources</b>		Lưu một bản sao độc lập của các kiểm tra hiện tại cùng với các tập tin tài nguyên của nó.
	<b>Save All</b>		Lưu tất cả tập tin.
	<b>Enable Editing</b>		Kích hoạt chức năng chỉnh sửa cho thư viện chỉ cho phép đọc.
	<b>Export Test to Zip File</b>	<b>CTRL+ALT+S</b>	Tạo 1 file zip (nén) cho tài liệu đang hoạt động.

	<b>Import Test from Zip File</b>	<b>CTRL+ALT+I</b>	Nhập một tài liệu từ file zip.
	<b>Convert to Scripted Component</b>	<b>CTRL+ALT+C</b>	Chuyển đổi thành phần kinh doanh vào thành phần kịch bản.
	<b>Print</b>	<b>CTRL+P</b>	In ấn.
	<b>Print Preview</b>		Xem và chỉnh sửa tài liệu trước khi in ấn.
	<b>Settings</b>		Thiết đặt thông tin.
	<b>Process Guidance Management</b>		Mở hộp thoại hướng dẫn quá trình quản lý, cho phép người dùng quản lý danh sách các quá trình có sẵn trong QuickTest.
	<b>Associate Library '&lt;Function Library Name&gt;' with '&lt;Document Name&gt;'</b>		Tập hợp các thư viện chức năng hoạt động với các tài liệu mở. (chỉ có sẵn từ các thư viện chức năng).
	<b>Recent Files</b>		Danh sách các tập tin được xem gần đây.
	<b>Exit</b>		Thoát khỏi chương trình kiểm thử.

**Thanh công cụ Automation:** có các button hỗ trợ vận hành một quá trình kiểm thử, trích từ menu Automation (không hiển thị theo mặc định).



	Lệnh	Phím tắt	Chức năng
	<b>Record</b>	<b>F3</b>	Bắt đầu ghi.
	<b>Run</b>	<b>F5</b>	Bắt đầu một phiên chạy từ đầu hoặc từ dòng đã được tạm dừng trước đó.

	<b>Stop</b>	F4 (Có thể xác định một phím tắt hoặc tổ hợp phím trong nhóm Run trong bảng thuộc tính Testing Options)	Dừng ghi phiên hoạt động.
	<b>Run Current Action</b>		Chỉ chạy hoạt động hiện tại.
	<b>Run from Step</b>	CTRL+F5	Khởi động chạy phiên làm việc đã được chọn trước.
	<b>Maintenance Run Mode</b>		Bắt đầu một phiên chạy trong chế độ bảo trì Run Wizard sẽ mở ra các bước không thành công bởi một đối tượng không được tìm thấy trong ứng dụng (nếu có).
	<b>Update Run Mode</b>		Bắt đầu một phiên chạy để cập nhật mô tả đối tượng kiểm tra và các tùy chọn khác (nếu có).
	<b>Analog Recording</b>	SHIFT+ALT+F3	Bắt đầu ghi trong chế độ Analog.
	<b>Low Level Recording</b>	CTRL+SHIFT+F3	Bắt đầu ghi trong chế độ cấp thấp.
	<b>Record and Run Settings</b>		Mở chế độ ghi và nhóm cài đặt Run, cho phép thiết lập thông tin của trình duyệt để ghi và chạy thử nghiệm.
	<b>Process Guidance List</b>		Danh sách các quá trình có sẵn dành cho các loại tài liệu hiện tại và cho cuộc kiểm thử nhanh đang được tải Add-in, cho phép mở chúng ra.
	<b>Results</b>		Cho phép xem kết quả một buổi chạy thử nghiệm

Thanh công cụ Edit: có các button để hỗ trợ trong việc chỉnh sửa kiểm thử hoặc thư viện chức năng, trích từ menu Edit.



	Lệnh	Phím tắt	Chức năng
	<b>Undo</b>	<b>CTRL+Z</b>	Đảo ngược lệnh cuối cùng hoặc xóa thông tin cuối đã nhập.
	<b>Redo</b>	<b>CTRL+Y</b>	Đảo ngược hoạt động gần đây nhất của lệnh Undo.
	<b>Cut</b>	<b>CTRL+X</b>	Xóa các lựa chọn từ tài liệu.
	<b>Copy</b>	<b>CTRL+C</b>	Tạo bản sao các lựa chọn từ tài liệu.
	<b>Paste</b>	<b>CTRL+V</b>	Dán lựa chọn từ tài liệu.
	<b>Delete</b>	<b>DELETE</b>	Xóa lựa chọn từ tài liệu.
	<b>Copy Document - action to Clipboard</b>		Sao chép các nội dung của cột tài liệu thông qua từ khóa, cho phép dán nó vào một ứng dụng bên ngoài.
	<b>Action &gt; Split Action</b>		Tách các hành động được lồng vào nhau.
	<b>Action &gt; Rename Action</b>	<b>SHIFT+F2</b>	Thay đổi tên một hành động.
	<b>Action &gt; Delete Action</b>		Cho phép loại bỏ các hành động, hoặc xóa các hành động và cuộc gọi từ các kiểm tra hoạt động.
	<b>Action &gt; Action Properties</b>		Cho phép chỉ định các tùy chọn, các thông số, và các kho chứa đối tượng liên quan cho một hành động lưu trữ.
	<b>Action &gt; Action Call Properties</b>		Cho phép chỉ định số lượng lặp lại chạy theo số lượng hàng trong bảng dữ liệu, và để xác định các giá trị của tham số đầu vào và vị trí lưu trữ các thông số đầu ra.

	<b>Step Properties &gt; Comment Properties</b>	<b>Ctrl+Enter ; Alt+Enter</b>	Mở hộp thoại Properties và nhập thông tin phản hồi. Chỉ có khi bước được lựa chọn là một phản hồi.
	<b>Step Properties &gt; Object Properties</b>	<b>Ctrl+Enter ; Alt+Enter</b>	Mở hộp thoại Properties để tùy chỉnh cho một đối tượng. Có sẵn khi bước lựa chọn có một đối tượng kiểm tra.
	<b>Step Properties &gt; Checkpoint Properties</b>		Mở hộp thoại Properties Checkpoint có liên quan một đối tượng được lựa chọn. Chỉ thực hiện khi bước được chọn là bước được kiểm soát.
	<b>Step Properties &gt; Output Value Properties</b>		Mở hộp thoại Properties và định giá trị giá tăng cho một đối tượng được lựa chọn. Chỉ thực hiện khi bước được lựa chọn là một bước giá trị đầu ra.
	<b>Step Properties &gt; Report Properties</b>	<b>Ctrl+Enter ; Alt+Enter</b>	Hiển thị hộp thoại Properties cung cấp thuộc tính cho một bước báo cáo. Chỉ thực hiện khi bước được chọn là một bước Reporter, ReportEvent.
	<b>Find</b>	<b>CTRL+F</b>	Tìm kiếm một chuỗi được xác định trước từ khóa.
	<b>Replace</b>	<b>CTRL+H</b>	Tìm kiếm và thay thế một chuỗi quy định.
	<b>Go To</b>	<b>CTRL+G</b>	Di chuyển chuột đến một dòng cụ thể trong các thử nghiệm.
	<b>Bookmarks</b>	<b>CTRL+B</b>	Đánh dấu trang trong kịch bản dễ dàng điều chỉnh.
	<b>Advanced &gt; Comment Block</b>	<b>CTRL+M</b>	Khóa các thông báo phản hồi.
	<b>Advanced &gt; Uncomment Block</b>	<b>Ctrl+Shift +M</b>	Mở khóa các thông báo phản hồi đã khóa trước đó.

	<b>Advanced &gt; Indent</b>	<b>Tab</b>	Xác định khoảng cách giữa các tab trong hộp thoại thuộc tính.
	<b>Advanced &gt; Outdent</b>	<b>Backspace</b>	Xác định khoảng cách lùi vào đầu dòng Backspace.
	<b>Advanced &gt; Go to Function Definition</b>	<b>Alt+G</b>	Điều hướng tới định nghĩa của chức năng được lựa chọn.
	<b>Advanced &gt; Complete Word</b>	<b>Ctrl+Space</b>	Hoàn thành từ khi nhập trong VBScript hoặc đối tượng.
	<b>Advanced &gt; Argument Info</b>	<b>Ctrl+Shift +Space</b>	Hiển thị cú pháp của một phương pháp.
	<b>Advanced &gt; Apply "With" to Script</b>	<b>Ctrl+W</b>	Tạo ra báo cáo cho các hành động hiển thị xem trong chế độ chuyên gia.
	<b>Advanced &gt; Remove "With" Statements</b>	<b>Ctrl+Shift +W</b>	Gỡ bỏ các báo cáo cho các hành động hiển thị xem trong chế độ chuyên gia để báo cáo.
	<b>Optional Step</b>		Chèn một bước tùy chọn

Thanh công cụ View: có các button để hỗ trợ trong việc quan sát trong quá trình kiểm thử, trích từ menu View.



	<b>Lệnh</b>	<b>Chức năng</b>
	<b>Start Page</b>	Xem trang bắt đầu
	<b>Active Screen</b>	Hiển thị màn hình hoạt động.
	<b>Available Keywords</b>	Ẩn từ khóa có sẵn.
	<b>Data Table</b>	Hiển thị bảng dữ liệu.
	<b>Debug Viewer</b>	Ẩn và hiện bảng Debug View.

	<b>Information</b>	Ẩn và hiện bảng Information.
	<b>Missing Resources</b>	Ẩn và hiện bảng Missing Resources.
	<b>Process Guidance</b>	Ẩn và hiện bảng Process Guidance.
	<b>Resources</b>	Ẩn và hiện bảng Resources.
	<b>Test Flow</b>	Ẩn và hiện bảng TestFlow.
	<b>To Do</b>	Ẩn và hiện bảng To Do.
	<b>Expand All</b>	Mở rộng tất cả các khóa.
	<b>Collapse All</b>	Đóng tất cả các khóa đã mở.
	<b>Keyword View</b>	Hiển thị từ khóa.
	<b>Expert View</b>	Hiển thị trong chế độ chuyên gia.
	<b>Toolbars</b>	Hiển thị thanh công cụ.
	<b>Window Theme</b>	Hiển thị nền màn hình.

Thanh công cụ **Insert**: có các button để chèn các bước, các hoạt động, Checkpoint, những giá trị Output trong kiểm thử, chúng được trích từ menu **Insert**.



	Lệnh	Phím tắt	Chức năng
	<b>Checkpoint &gt; Existing Checkpoint</b>	<b>Alt +F12</b>	Mở hộp thoại Existing Checkpoint, cho phép đánh dấu kiểm soát hiện tại cho một đối tượng hoặc một bảng.
	<b>Checkpoint &gt; Standard Checkpoint</b>	<b>F12</b>	Mở hộp thoại Properties Checkpoint, cho phép tạo ra một trạm kiểm soát cho một đối tượng hoặc một bảng.
	<b>Output Value &gt; Existing Output Value</b>	<b>Shift+Ctrl + F12</b>	Mở hộp thoại Existing Output Value, cho phép tạo ra một giá trị đầu ra tiêu chuẩn cho một đối tượng hoặc một bảng.

	<b>Output Value &gt; Standard Output Value</b>	<b>Ctrl+F12</b>	Mở hộp thoại <b>Standard Output Value</b> , cho phép tạo ra một giá trị đầu ra giả tăng cho một đối tượng hoặc một bảng.
	<b>Step Generator</b>	<b>F7</b>	Mở hộp thoại <b>Step Generator</b> .
	<b>Function Definition Generator</b>		Mở hộp thoại <b>Function Definition Generator</b>
	<b>Synchronization Point</b>		Chèn một điểm đồng bộ hóa trong các thử nghiệm, hướng dẫn kiểm thử nhanh tạm dừng các kiểm tra cho đến khi đạt được giá trị đối tượng sở hữu (hoặc lần ra).
	<b>New Step</b>	<b>F8; Insert</b>	Chèn một bước tiến mới vào khung Keyword view.
	<b>New Step After Block</b>	<b>Shift+F8</b>	Chèn một bước tiến mới sau khi một khối có điều kiện, vòng lặp trong khung Keyword view.
	<b>Operation</b>		Chèn một hoạt động (chức năng) vào một thành phần.
	<b>Comment</b>		Chèn một bước thảo luận trong Keyword view.
	<b>Report</b>		Chèn một Report vào khung Keyword view, hướng dẫn kiểm thử nhanh và báo cáo một sự kiện từ kết quả thử nghiệm.
	<b>Conditional Statement</b>		Chèn một hàm If...Then , Elseif ... Then hoặc Else tùy theo lựa chọn của bạn.
	<b>Loop Statement</b>		Chèn một hàm While...Wend, For...Next, Do...While, hoặc Do...Until tùy theo lựa chọn của bạn.

	<b>Call to New Action</b>		Tạo một hành động mới và chèn nó vào vị trí quy định.
	<b>Call to Copy of Action</b>		Gọi bước sao chép hành động trước đó, để có thể chỉnh sửa.
	<b>Call to Existing Action</b>		Gọi bước sao chép một hành động có thể tái sử dụng hiện tại.
	<b>Call to WinRunner</b>		Gọi bước sao chép một thử nghiệm WinRunner hoặc chức năng người dùng định nghĩa. (WinRunner đã được cài đặt trên máy tính).
	<b>Start Transaction</b>		Chèn một bước <b>StartTransaction</b> trong các thử nghiệm, đánh dấu sự bắt đầu.
	<b>End Transaction</b>		Chèn một bước <b>EndTransaction</b> trong các thử nghiệm, đánh dấu sự kết thúc.

Thanh công cụ Tool: có các button để thiết lập những tùy chọn, kiểm tra cú pháp và làm việc Object Spy, chúng được trích từ menu Tool.



	Lệnh	Phím tắt	Chức năng
	<b>Options</b>		Mở hộp thoại Options, cho phép chỉnh sửa các tùy chọn thử nghiệm.
	<b>View Options</b>		Mở hộp thoại <b>Keyword Options view</b> , cho phép tùy chỉnh kiểm tra và các thư viện chức năng được hiển thị trong <b>Expert View</b> và cửa sổ <b>Function Library</b> .
	<b>Check Syntax</b>	<b>CTRL+F7</b>	Kiểm tra cú pháp của tài liệu đang hoạt động.
	<b>Object Identification</b>		Mở hộp thoại xác định đối tượng, cho phép chỉ định như thế nào kiểm thử nhanh một đối tượng cụ thể.

	<b>Object Spy</b>		Mở hộp thoại Object Spy, cho phép xem các thuộc tính bản địa và hoạt động của đối tượng bất kỳ trong một ứng dụng mở, cũng như hệ thống phân cấp đối tượng kiểm tra, xác định tính chất và hoạt động của đối tượng kiểm tra QuickTest sử dụng để đại diện cho đối tượng.
	<b>Web Event Recording Configuration</b>		Mở hộp thoại Web cấu hình ghi sự kiện, cho phép chỉ định một cấp độ cấu hình ghi.
	<b>Data Driver</b>		Mở hộp thoại dữ liệu điều khiển, hiển thị các hằng số, danh sách mặc định cho hành động.
	<b>Change Active Screen</b>		Thay thế màn hình đăng nhập trước đó được ghi lại với màn hình đăng nhập chọn.
	<b>Virtual Objects &gt; New Virtual Object</b>		Mở Wizard đối tượng ảo, cho phép tiến hành kiểm thử nhanh để nhận ra một khu vực của ứng dụng của bạn như là một đối tượng kiểm tra tiêu chuẩn.
	<b>Virtual Objects &gt; Virtual Object Manager</b>		Mở trình quản lý đối tượng ảo, cho phép quản lý tất cả các bộ sưu tập đối tượng ảo được xác định trên máy tính.
	<b>Customize</b>		Mở hộp thoại Customize, cho phép tùy biến thanh công cụ và menu, và tạo ra thực đơn mới.

**Thanh công cụ Debug:** có các button để giúp gỡ lỗi trong quá trình kiểm thử, chúng được trích từ menu Debug.

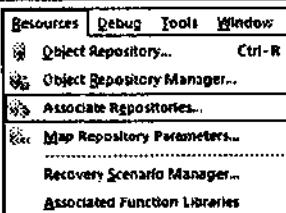


	Lệnh	Phím tắt	Chức năng
	<b>Pause</b>		Tạm dừng phiên gỡ lỗi.

	<b>Step Into</b>	F11	Chỉ chạy dòng hiện tại của kịch bản. Nếu dòng hiện tại là một phương thức, thì nó không được hiển thị trong khung nhìn.
	<b>Step Over</b>	F10	Chỉ chạy dòng hiện tại của kịch bản. Khi dòng hiện tại là một phương thức, nó được thực hiện toàn bộ, nhưng không được hiển thị trong khung nhìn.
	<b>Step Out</b>	Shift+F11	Chạy kết thúc của phương thức, sau đó tạm dừng phiên chạy.
	<b>Run to Step</b>	Ctrl+F10	Chạy cho đến bước hiện tại.
	<b>Debug from Step</b>		Chạy từ bước lựa chọn thay vì bắt đầu thử nghiệm.
	<b>Add to Watch</b>	Ctrl+T	Thêm một mục được chọn vào tab Watch.
	<b>Insert/Remove Breakpoint</b>	F9	Thiết lập hoặc xóa một điểm dừng trong các thử nghiệm.
	<b>Enable/Disable Breakpoint</b>	Ctrl+F9	Cho phép hoặc vô hiệu hóa một điểm dừng trong các thử nghiệm.
	<b>Clear All Breakpoints</b>	Ctrl+Shift+F9	Xóa tất cả các breakpoint trong các kiểm thử.
	<b>Enable/Disable All Breakpoints</b>		Cho phép hoặc vô hiệu hóa tất cả các breakpoint trong các kiểm thử.

Ngoài các thanh các công cụ được trích ra từ các Menu lệnh còn có một số menu lệnh hỗ trợ làm việc khác như sau:

**Menu Resource:** có các lệnh để quản lý đối tượng và nguồn lực của quá trình kiểm thử.



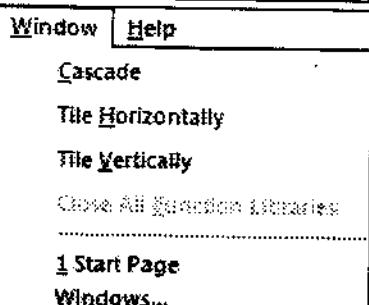
	Lệnh	Phím tắt	Chức năng
	<b>Object Repository</b>	CTRL+R	Mở cửa sổ Repository Object, hiển thị một cây có chứa tất cả các đối tượng trong kiểm thử hiện tại hoặc một thành phần.

	<b>Object Repository Manager</b>		Mở hộp thoại Object Repository Manager, cho phép mở và chỉnh sửa nhiều đối tượng.
	<b>Associate Repositories</b>		Mở hộp thoại Repositories, cho phép quản lý các nhóm lưu trữ các đối tượng cho kiểm tra.
	<b>Map Repository Parameters</b>		Mở hộp thoại Map Repository Parameters, cho phép đọc các thông số lưu trữ cần thiết.
	<b>Recovery Scenario Manager</b>		Mở kịch bản phục hồi hộp thoại quản lý.
	<b>Associated Function Libraries</b>		Danh sách các thư viện chức năng có liên quan với các tài liệu hoạt động, cho phép mở chúng.

**Menu Window:** có các lệnh để sắp xếp, định vị trí các khung làm việc của của quá trình kiểm.

Ngoài ra, còn hiển thị của các cuộc kiểm thử trước đó và tùy chỉnh các cửa sổ hiển thị tài liệu.

Hãy tìm hiểu tính năng qua bảng lệnh sau:



Lệnh	Chức năng
<b>Cascade</b>	Hiển thị hết các tài liệu hay cửa sổ đang mở.
<b>Tile Horizontally</b>	Hiển thị các cửa sổ làm việc theo phương ngang.
<b>Tile Vertically</b>	Hiển thị các cửa sổ làm việc theo phương dọc
<b>Close All Function Libraries</b>	Đóng thư viện tất cả các chức năng đang mở.
<b>Open files section</b>	Mở danh sách các tài liệu hiện đang dùng trong phiên kiểm thử
<b>Windows</b>	Mở hộp thoại Windows, cho phép quản lý các cửa sổ tài liệu mở.

## KHU VỰC 2

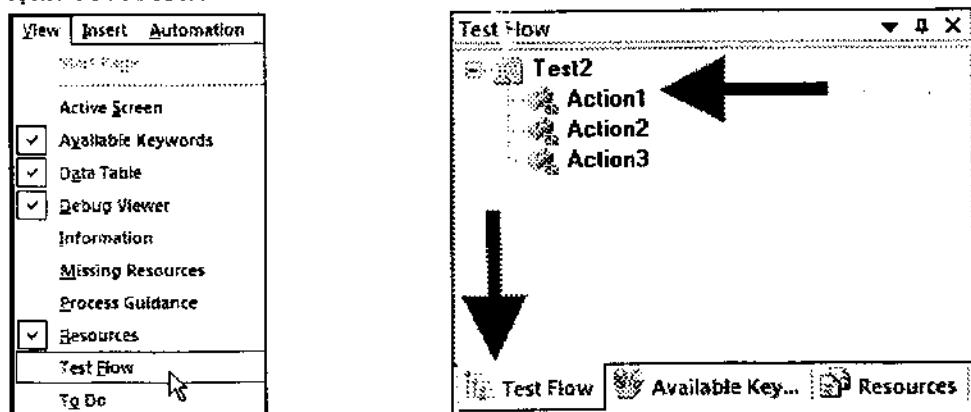
Khu vực này cho phép hiển thị bảng nguồn của quá trình kiểm thử theo dạng cấu trúc cây. Trên bảng nguồn cho phép hiển thị 3 thành phần nguồn (Sources) là: Test Flow, Available keyword, Resources.

Khi nhấp chọn 1 trong 3 thành phần này nằm ở cuối bảng thì nội dung của nó xuất hiện theo dạng cấu trúc cây ngay trên bảng.

### 1. Bảng Test Flow

Bảng Test Flow bao gồm một hệ thống các action hiện tại và các action được gọi ra, thêm vào trong quy trình kiểm thử hiện tại. Bảng thể hiện thứ tự của các action theo thứ tự hoạt động của chúng. Mỗi Action được biểu diễn tại một nút của cây và bao gồm tất cả các truy cập action của kiểm thử. Nếu muốn quan sát mở rộng hãy nhấp đúp vào các nút action và đồng thời các dòng kiểm thử sẽ xuất hiện trên bảng Keyword view và Expert View.

Bảng sẽ được hiển thị mặc định khi khởi động chương trình QuickTest Professional. Nếu bảng không được hiển thị, hãy nhấp vào biểu tượng tab **Test Flow** cuối khu vực 2, hoặc nhấp chọn menu View sau đó đánh dấu chọn lệnh **Test Flow**.



### Cách sử dụng bảng Test Flow

Cửa sổ cho phép xem tất cả các action trong cuộc kiểm thử hiện tại và trật tự hoạt động (chạy) của chúng. Trong bảng, người dùng có thể thực hiện kiểm tra, lập các action và thuộc tính của action, làm việc với kho lưu trữ các đối tượng và chạy các action cụ thể.

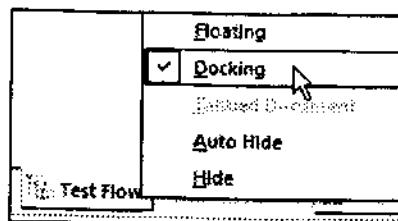
Trong bảng Test Flow có các biểu tượng và chức năng của chúng như bảng sau:

Biểu tượng	Chức năng
	Một cuộc kiểm thử.

	Gọi một action không thể tái sử dụng.
	Gọi một action từ bên ngoài.
	Gọi một action từ bên ngoài có điều kiện.
	Gọi một action có thể tái sử dụng.
	Gọi một action tái sử dụng có điều kiện.
	Gọi một action bị lỗi (không được lưu vào cuộc kiểm thử).
	Gọi một action có điều kiện bị lỗi.
	Gọi một action vòng có thể tái sử dụng.
	Gọi một action vòng có điều kiện tái sử dụng.
	Gọi một action vòng từ bên ngoài.
	Gọi một action vòng có điều kiện bên ngoài .

### Chú ý

Có thể nhấp phải vào tab Test Flow ở vị trí cuối bảng để tùy chọn các chế độ hiển thị khác nhau, bao gồm Floating (nổi), Docking (chìm), Auto Hide (tự động ẩn), Hide (ẩn) ...

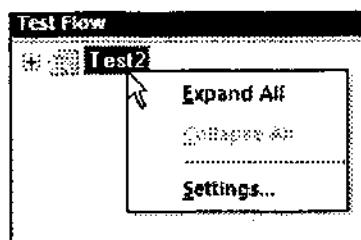


### Làm việc với các Action trong bảng

Khi nhấp đúp vào 1 action trong quá trình kiểm thử, thì trong cửa sổ Keyword view và Expert View cũng sẽ hiển thị các hoạt động đã lựa chọn.

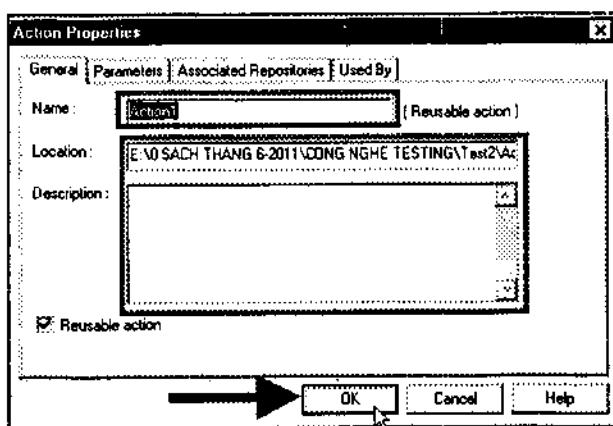
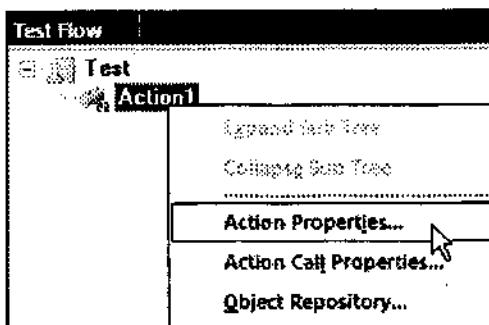
The screenshot illustrates the interaction between the Test Flow table and the Keyword view/Expert View windows. On the left, the Test Flow table shows a test case named 'Test2' containing three actions: Action1, Action2, and Action3. A double-headed horizontal arrow connects the 'Action1' row in the Test Flow table to the 'Action1' window on the right. The 'Action1' window displays the selected keyword 'Tú sách STK' and its associated steps: 'Điền SÁCH TÚ SÁCH STK', 'Đăng Nhập', 'ĐIỀN DÀN TỜ SÁCH STK', and 'AutoComplete Passwords'. The 'Operation' column for these steps is also visible.

Cho phép ẩn và hiện các action con trong cây kiểm thử. Hãy nhấp phải vào nút cộng của cây Test Flow và chọn các lệnh: Expand All (mở rộng tất cả các action), Collapse All (thu gọn tất cả các action). Ngoài ra có thể thực hiện cài đặt thông tin thuộc tính bằng cách nhấp chọn lệnh Settings.

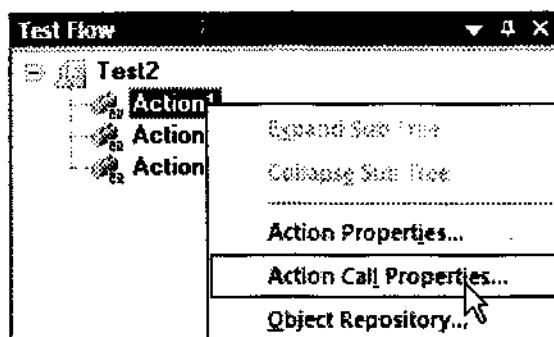


**Chú ý:** có thể dùng phím "+" hoặc "\*" và phím "-" để mở rộng hoặc đóng các action.

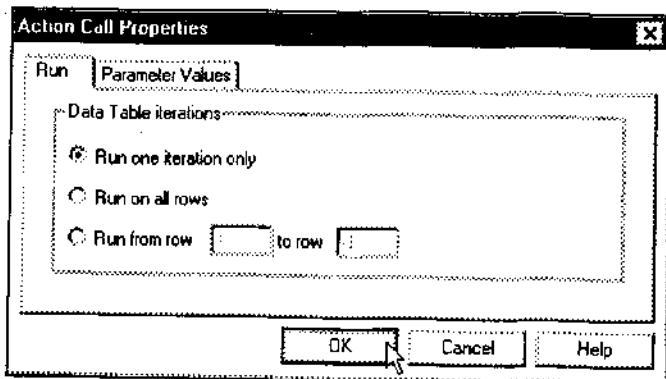
Để hiển thị các thuộc tính của action, trên cây Test Flow hãy nhấp chuột phải vào action đó và chọn lệnh **Action Properties**. Hộp thoại Action Properties xuất hiện, hãy nhập tên của action vào khung Name và thông tin mô tả action vào khung Description, sau đó nhấp nút OK đóng ý.



Để hiển thị các thuộc tính gọi của action, trên cây Test Flow hãy nhấp chuột phải vào action đó và chọn lệnh **Action Call Properties**. Hộp thoại Action Call Properties xuất hiện, lựa chọn số lần chạy của action, và nhấp nút OK đóng ý.



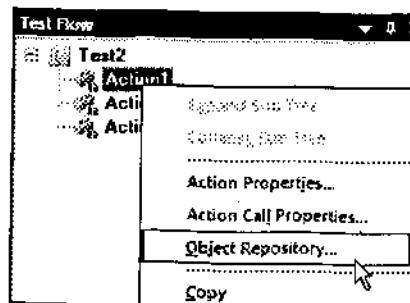
Thông tin của hộp thoại này giúp xác định số lần chạy của action và chỉ định cho chúng bao nhiêu lần chạy. Có thể tùy chọn chạy action dựa vào số dòng, số cột trong bảng dữ liệu.



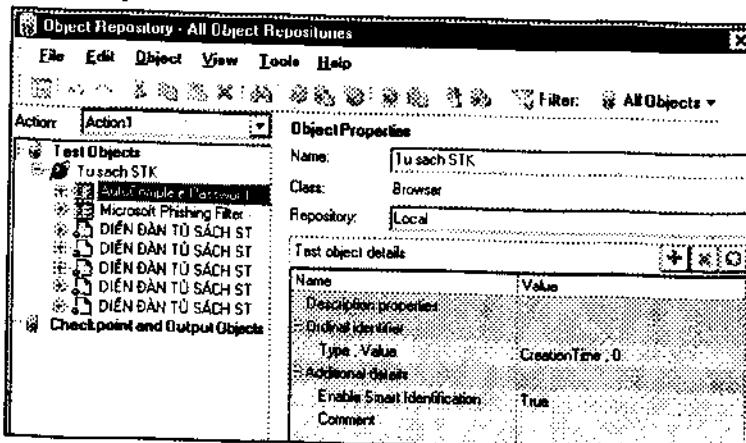
Ngoài ra còn có thể xác định giá trị ban đầu đối với bất kỳ thông số đầu vào và vị trí mà bạn muốn để lưu trữ các giá trị thông số hành động đầu ra.

Để làm việc với cơ sở dữ liệu đối tượng. Trên cây Test Flow hãy nhấp chuột phải vào action đó và chọn lệnh **Object Repository** để mở hộp thoại **Repository Object**.

Trong hộp thoại, hiển thị một cây có chứa tất cả các đối tượng trong thử nghiệm hiện tại.



Repository Object Manager cho phép quản lý tất cả các đối tượng trong kho được sử dụng từ dự án của bạn tại một địa điểm duy nhất là trung tâm. Có thể bổ sung, xác định, thay đổi và mô tả các đối tượng bên trong một Action. Trong đó, có thể thực hiện sao chép, tái sử dụng các thành phần nằm trong kiểm thử và vận dụng chia sẻ chúng cho các đối tượng action khác nhau. Ngoài ra còn có thể tùy chọn và chỉnh sửa thông tin riêng của từng đối tượng kiểm thử này.



**Chú ý:** thay vì, sửa đổi hoặc bổ sung, kho đổi tượng chia sẻ, có thể chọn để lưu trữ tất cả hoặc một số đổi tượng trong một kho lưu trữ đổi tượng riêng cho mỗi hành động, để có thêm thông tin về các kho chứa đổi tượng.

Nếu các giá trị của các đổi tượng ứng dụng khác với giá trị của quá trình kiểm thử sử dụng để xác định các đổi tượng thì cuộc kiểm thử có thể thất bại. Để khắc phục vấn đề này, các bạn phải thực hiện thay đổi các giá trị xác định đổi tượng trong kho dữ liệu sao cho tương ứng rồi mới tiếp tục kiểm thử.

Nếu một đổi tượng có cùng tên và mô tả nằm ở cả hai kho lưu trữ các đổi tượng lưu trữ trong ổ đĩa và trong một kho lưu trữ đổi tượng chia sẻ được liên kết với cùng một hành động, hành động sử dụng sẽ định nghĩa đổi tượng lưu trữ trong ổ đĩa.

Nếu một đổi tượng có cùng tên và mô tả nằm trong kho lưu trữ nhiều hơn một đổi tượng chia sẻ, và những kho chứa đổi tượng chia sẻ tất cả các liên kết với cùng một hành động, kiểm thử nhanh sẽ sử dụng định nghĩa đổi tượng từ sự xuất hiện đầu tiên của đổi tượng, theo thứ tự kho đổi tượng chia sẻ được kết hợp với hành động. Bạn có thể sử dụng kho lưu trữ đổi tượng chia sẻ cùng với nhiều hành động.

**Quản lý action:** nhấp chuột phải vào một hành động trong cây và sau đó chọn **Copy** hoặc **Delete**, thực hiện sao chép hay xóa bỏ một action.

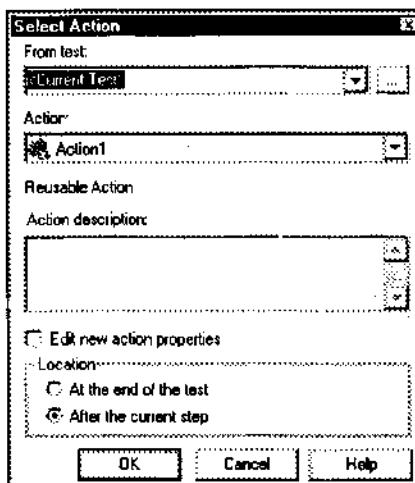
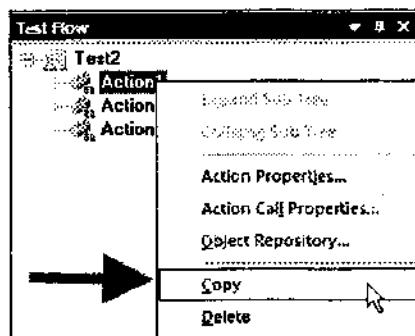
Chọn **Copy** để mở hộp thoại **Select Action** và tạo ra một bản sao của các hành động trong cuộc kiểm thử. Trong khung **From test**, nhấp chọn mã tên và chọn action lưu trữ trước đó.

**Lưu ý:** bạn có thể nhập vào một thư mục kiểm thử hoặc một đường dẫn tương đối trong hộp kiểm tra.

Trong khung **Action**, hãy nhấp chọn các Action muốn sao chép.

Khung này hiển thị tất cả các action từ cuộc kiểm thử lưu trữ và hiện tại tùy vào lựa chọn.

Thực hiện nhập thông tin mô tả cho các action sẽ sao chép vào khung **Action description**.



Nếu muốn chỉnh sửa các thuộc tính của action vừa sao chép, hãy đánh dấu chọn vào dòng **Edit new action properties**. Nếu chọn tùy chọn này, hộp thoại Properties action được hiển thị khi nhấp OK. Sau đó, có thể sửa đổi các thuộc tính action như mô tả trong việc thiết lập các thuộc tính cuộc gọi hành động. Thành phần location, giúp thực hiện quyết định điểm chèn của action vừa mới sao chép. Có 2 tùy chọn đó là **At the end of the test** (cuối cuộc kiểm thử) và **After the current step** (sau bước kiểm thử sao chép). Sau cùng nhấp nút OK đồng ý.

Action được đưa vào kiểm thử như một action độc lập, không tái sử dụng. Bạn có thể chuyển action tới vị trí khác trong kiểm thử bằng cách kéo nó đến vị trí mong muốn.

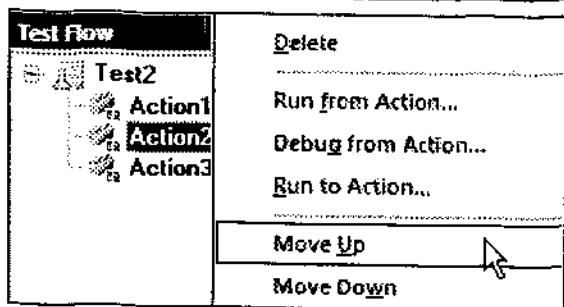
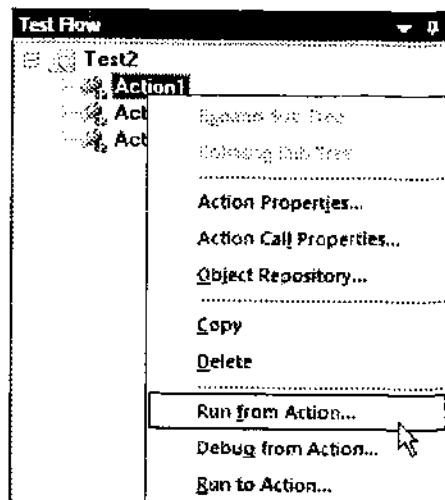
Trên cây Test Flow nhấp phải chọn tùy chọn **Delete** để loại bỏ các action từ cuộc kiểm tra.

**Khởi động kiểm thử:** nhấp chuột phải vào một hành động trong cây và sau đó chọn **Run from Action** hoặc **Run to action**, để bắt đầu một phiên chạy từ đầu của action được lựa chọn, hoặc chạy thử nghiệm cho đến điểm bắt đầu của action được lựa chọn và sau đó tạm dừng phiên chạy.

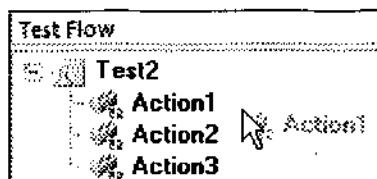
**Gỡ lỗi (Debug) kiểm thử:** nhấp chuột phải vào một hành động trong cây và sau đó chọn **Debug From Action** để bắt đầu (và tạm dừng) một phiên gỡ lỗi vào lúc bắt đầu của action lựa chọn.

**Thay đổi thứ tự chạy các action:** nhấp chuột phải vào một hành động cấp cao nhất trong cây và sau đó chọn **Move Up** hoặc **Move Down**. Cũng có thể nhấp **Ctrl + Mũi tên lên** hoặc **Ctrl + mũi tên xuống** để di chuyển một hành động và hành động.

Ngoài ra, có thể dùng cách kéo một action cấp cao nhất trong các cây lên hoặc xuống đến vị trí yêu cầu. Khi kéo một action được lựa chọn, một dòng được hiển thị, cho phép xem các vị trí trong cây mà action sẽ được di chuyển.



Chỉ có thể kéo các hành động cấp cao. Lựa chọn action lớn tự động bao gồm tất cả các action con, không thể kéo một action con, cũng không thể kéo một action lớn với chỉ có một số action con của nó.



### KHU VỰC 3

Khu vực này cho phép hiển thị bảng thông tin các hành động kiểm thử. Trên bảng cho phép hiển thị 2 thành phần là: **Keyword View**, **Expert View**.

#### Keyword View

Khung này cho phép tạo và xem các bước của các thành phần trong một từ khóa điều khiển, định dạng mô-dun. Keyword View tạo các cột phân chia thông tin khác nhau của từng thành phần trong action, mỗi cột đại diện cho các bộ phận khác nhau của các bước. Bạn có thể sửa đổi các cột hiển thị cho phù hợp với yêu cầu của kiểm thử. Có thể tạo và chỉnh sửa các thành phần bằng cách chọn các mục và action trong Keyword View, sau đó nhập thông tin theo yêu cầu.

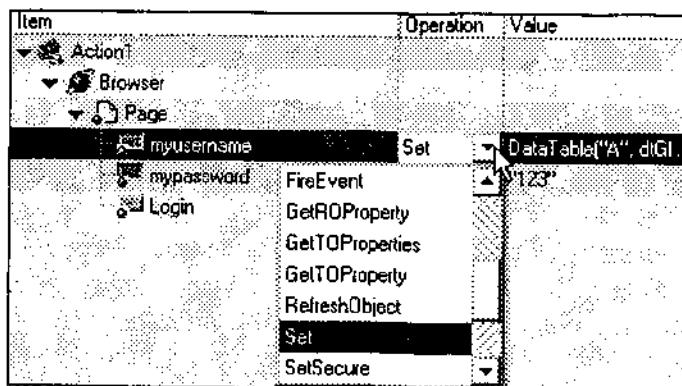
Start Page   Test2			
Action1		Back   Close	
Item	Operation	Value	Documentation
Actions			
Tuach STK	Click		Click the "Đăng Nhập" link.
DIEN DAN TU SACH STK_1	Click		
DIEN DAN TU SACH STK_2			
AutoComplete Password			
DIEN DAN TU SACH STK_3			
DIEN DAN TU SACH STK_4			
AutoComplete Password			
DIEN DAN TU SACH STK_5			
DIEN DAN TU SACH STK_6			
DIEN DAN TU SACH STK_7			

Action trong trong bảng Keyword View là mức cao nhất trong hệ thống phân cấp kiểm thử. Nó chứa tất cả các bước của action khi thực hiện trên tài liệu. Có thể nhấp chọn vào từng nút cộng để mở các bước nhỏ bên trong. Mỗi hành động bao gồm các bước, mỗi bước được chèn vào như là một hàng trong bảng Keyword.

Ví dụ: Keyword View có thể chứa các hàng sau đây:

Item	Operation	Value	Documentation
Action1			
Browser			
Page			
myusername	Set	DataTable	Enter <the value of the 'A' Data Table column
mypassword	Set	"123"	Enter "123" in the "mypassword" edit box.
Login	Click		Click the "Login" button.

Trên các bước này, có thể thực hiện các tùy chỉnh thông tin hay thêm các bước mới vào action. Nhấp chọn vào dòng cần thiết đặt lại thuộc tính và chọn thông tin thuộc các cột thả các tùy chọn xuống, sau đó lựa chọn thiết đặt phù hợp.

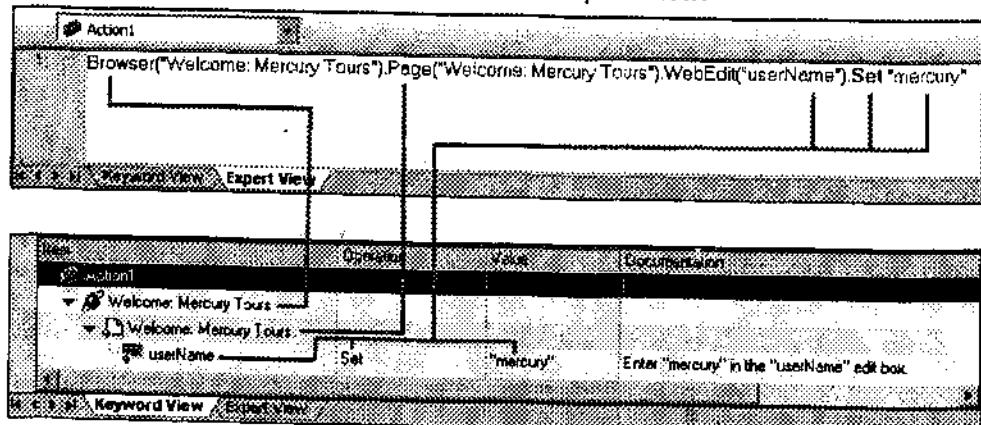


### Expert View

Expert View cho phép hiển thị các bước tương tự như Keyword View, nhưng trong 1 định dạng khác:

Trong Keyword View, thông tin kiểm thử thể hiện nhanh mỗi bước và cho thấy hệ thống phân cấp đối tượng dựa trên các biểu tượng.

Hiển thị từng bước như là một dòng VB Script hoặc Statement. Trong từng bước của đối tượng cơ bản, báo cáo VB Script xác định hệ thống phân cấp đối tượng. Sơ đồ dưới đây cho thấy hệ thống phân cấp cùng một đối tượng được hiển thị trong Keyword View và Expert View:



Mỗi dòng của VBScript trong Expert View đại diện cho một bước trong các kiểm thử.

Bảng dưới đây giải thích cách các phần khác nhau của cùng một bước đại diện Keyword View và Expert View.

Keyword View	Expert View	Giải thích
	<b>Browser</b> ("Welcome: Mercury Tours")	Tên của đối tượng kiểm tra trình duyệt: <b>Mercury Tours</b>
	<b>Page</b> ("Welcome: Mercury Tours")	Tên của trang hiện tại là: <b>Mercury Tours</b>
	<b>WebEdit</b> ("userName")	Các loại đối tượng là WebEdit, tên của hộp sửa đổi hoạt động được thực hiện là <b>userName</b> .
	<b>Set</b>	Phương pháp thực hiện trên hộp soạn thảo là <b>Set</b> .
	"mercury"	Giá trị chèn vào hộp chỉnh sửa tên người dùng là <b>mercury</b> .

Trong Expert view, mô tả các đối tượng hiển thị trong dấu ngoặc đơn. Đối với các đối tượng được lưu trữ trong các kho, tên đối tượng là một mô tả đầy đủ. Các đối tượng trong hệ thống phân cấp đối tượng được ngăn cách bởi một dấu chấm.

Trong ví dụ sau đây, trình duyệt và trang là hai đối tượng riêng biệt trong cùng một hệ thống phân cấp:

**Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours")**

Các hoạt động (Method) thực hiện trên đối tượng luôn luôn hiển thị ở cuối của bảng statement, theo sau bởi bất kỳ giá trị liên quan đến hoạt động.

Trong ví dụ sau đây, từ Mercury được nhập vào hộp **userName** chỉnh sửa bằng cách sử dụng các phương pháp Set:

**Browser ("Welcome: Mercury Tours"). Page("Welcome: Mercury Tours"). WebEdit("userName"). Set "mercury".**

#### KHU VỰC 4

Cho phép hiển thị bảng Data Table (bảng dữ liệu), Active Screen (màn hình hoạt động), Debug Viewer (khung sửa lỗi), Missing Resources (bảng nguồn lỗi)...

## DATA TABLE

Bảng dữ liệu chứa một tab Global với một bổ sung cho mỗi action trong kiểm thử. Để xem bảng dữ liệu, nhấn vào tab Data Table hoặc chọn View > Data Table. Bảng dữ liệu là một bảng tính với các cột và các hàng đại diện cho các dữ liệu trong kiểm thử.

Bảng dữ liệu có những đặc điểm của một bảng tính Microsoft Excel, chúng ta có thể lưu trữ và sử dụng dữ liệu trong ô. Có thể sử dụng bảng dữ liệu được cung cấp bởi chương trình, hoặc có thể sử dụng bất kỳ tập tin Microsoft Excel (xls).

Cũng có thể sử dụng DataTable, DTSheet và các đối tượng tiện ích DTPParameter để thao tác dữ liệu trong bất kỳ ô trong bảng dữ liệu. Để có thêm thông tin về các đối tượng này, xem phần tiện ích mô hình tham chiếu đối tượng của HP QuickTest Professional.

**Lưu ý:** việc sử dụng các công thức phức tạp And/Or lồng nhau trong bảng dữ liệu không được hỗ trợ.

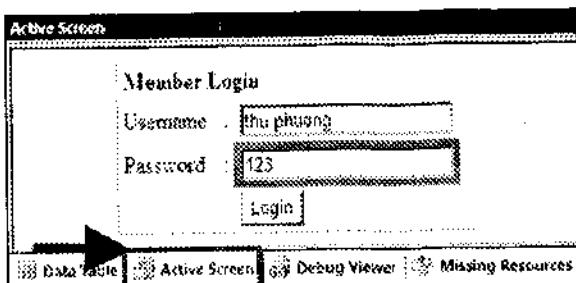
Có thể chèn các thông số Data Table và giá trị sản lượng vào cuộc kiểm thử. Sử dụng các thông số Data Table And/Or các giá trị dẫu ra trong một bài kiểm tra cho phép tạo ra một hành động kiểm tra dữ liệu theo định hướng hoặc chạy nhiều lần bảng cách sử dụng các dữ liệu cung cấp. Trong mỗi lần lặp lại, hoặc lặp đi lặp lại, QuickTest sử dụng một giá trị khác từ các Data Table.

### Chỉnh sửa các bảng dữ liệu (Data Table):

Có thể chỉnh sửa thông tin trong bảng dữ liệu bằng cách gõ trực tiếp vào các ô của bảng. Sử dụng bảng dữ liệu giống như một bảng tính Microsoft Excel, bao gồm cả chèn công thức vào trong cell. Ngoài ra, cũng có thể nhập dữ liệu lưu trong Microsoft Excel, tập tin văn bản (txt), hoặc định dạng ASCII. Đối với thông tin về phiên bản hỗ trợ của Microsoft Excel có trong HP QuickTest Professional Readme. Mỗi hàng trong bảng đại diện cho tập hợp các giá trị mà cuộc kiểm thử nạp cho các đối số tham số trong một phiên duy nhất của các kiểm tra hoặc action. Cũng có thể nhập dữ liệu và công thức trong các ô trong các cột không có ý định sử dụng với các thông số bảng dữ liệu (các cột mà không có một tên tham số trong tiêu đề cột).

## ACTIVE SCREEN

Màn hình active này cung cấp phần hiển thị giao diện của các ứng dụng, và màn hình chỉ xuất hiện trong từng bước kiểm thử nhất định trong lần ghi mà bạn thực hiện.



Ngoài ra, tùy thuộc vào các tùy chọn chụp màn hình đang sử dụng trong khi ghi, trang hiển thị trong màn hình active có thể chứa thông tin sở hữu chi tiết về từng đối tượng hiển thị trên trang.

### Làm việc với màn hình active

Màn hình active cho phép nhập tham số giá trị đối tượng và chèn các điểm kiểm tra (checkpoint), phương thức (methods), và các giá trị đầu ra (output) cho hầu hết các đối tượng bất kỳ trong trang, sau khi hoàn thành buổi ghi.

Kiểm thử viên có thể xác định thông tin của cuộc kiểm thử và thông tin của từng giao diện của đối tượng trong khi ghi. Người ghi có thể tự điều khiển chương trình QulckTest để ghi lại các đối tượng mà mình muốn.

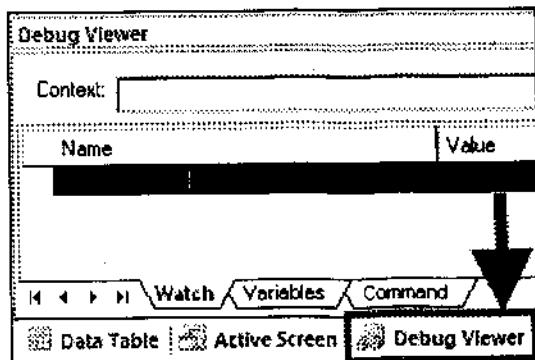
Chương trình tạo ra một màn hình Active cho một ứng dụng dựa trên web và nó lưu đường dẫn đến các hình ảnh và nguồn tài nguyên của website chứ không tải về hay lưu trữ chúng. Do đó, cần phải cung cấp thông tin đăng nhập để xem các nguồn tài liệu được bảo vệ bằng mật khẩu.

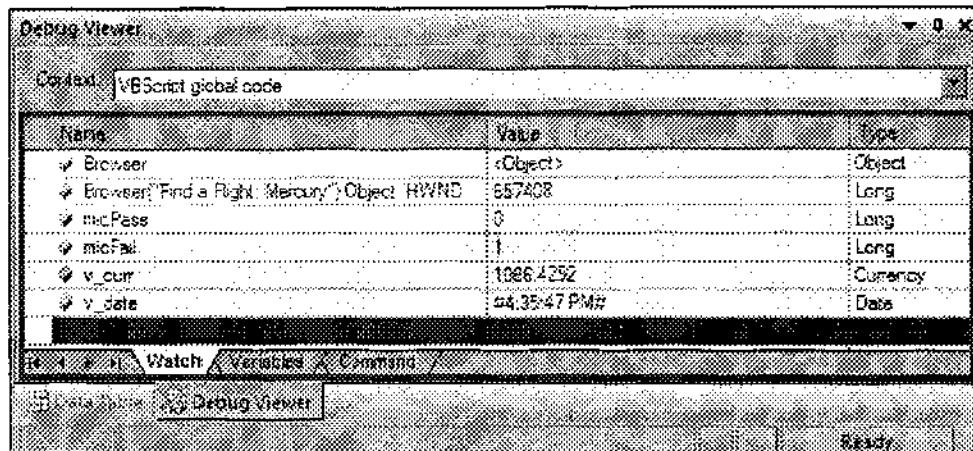
Khi làm việc với các ứng dụng dựa trên web, cần phải xác định các tiêu chuẩn màn hình cho trang web được ghi. Người dùng có thể tùy chọn lưu hoặc không lưu nội dung của màn hình active với cuộc kiểm thử. Tuy nhiên việc lưu trữ này rất tốt cho việc chỉnh sửa các kiểm tra lưu trực tiếp từ màn hình active. Khi muốn kiểm tra lại chỉ cần nhấp chọn action và chọn màn hình active để xem lại.

## DEBUG VIEWER

Cửa sổ Debug Viewer giúp gỡ rối các kiểm thử hoặc thư viện Function. Trên khung Debug có 3 thành phần chính hiển thị thành dạng tab là: Watch, Variables, Command.

Các chức năng của các tab như sau:





### Tab watch

Cho phép hiển thị giá trị hiện tại và bất kỳ loại biến hoặc VBScript muốn thêm vào tab. Nó cũng cho phép thiết lập và sửa đổi các giá trị của biến và thuộc tính được hiển thị.

### Tab Variables

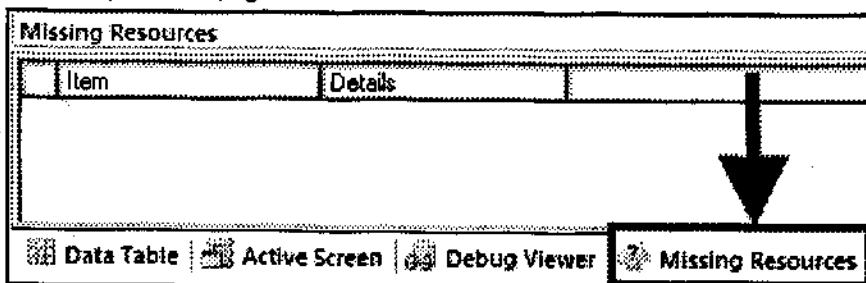
Tab này hiển thị giá trị hiện tại và tất cả các biến đã được chấp nhận đến bước cuối cùng và chạy trong phiên kiểm thử cuối mà người dùng muốn sửa lỗi. Ngoài ra, cũng có thể thực hiện thiết lập hay sửa đổi của các biến được hiển thị.

### Tab Comand

Tab này cho phép chạy dòng kịch bản để thiết lập hoặc sửa đổi các giá trị hiện tại của một biến hay VBScript của đối tượng kiểm thử.

## MISSING RESOURCES

Khung biểu diễn các nguồn bị lỗi và cung cấp một danh sách các nguồn tài nguyên được quy định cụ thể trong thử nghiệm của bạn nhưng không thể tìm thấy. Nguồn lỗi có thể bao gồm các thư viện Function bị mất, thiếu kịch bản phục hồi, unmapped kho đối tượng chia sẻ, và các thông số được kết nối với kho dữ liệu đối tượng chia sẻ.



Mỗi lần mở các thành phần hoặc khu vực ứng dụng, chương trình QuickTest tự động kiểm tra tất cả các nguồn lực xác định có thể truy cập.

Nếu chương trình tìm thấy nguồn không thể truy cập, danh sách các nguồn này sẽ xuất hiện trong cửa sổ dữ liệu lỗi.

**Missing resources** chứa các cột sau đây:

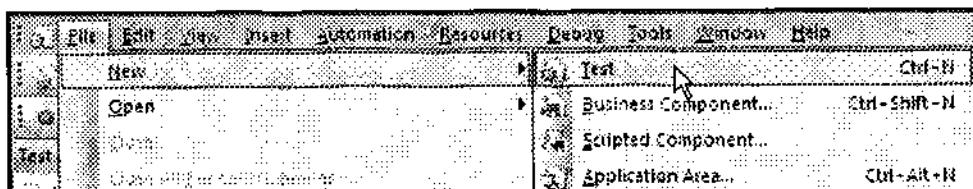
- **Cột Item:** liệt kê các nguồn dữ liệu bị mất.
- **Cột Details:** thông tin chi tiết về mỗi nguồn dữ liệu bị mất, chẳng hạn như vị trí, trong đó kiểm thử hy vọng sẽ tìm thấy những nguồn tài nguyên.

Missing Resources	
File	Missing Object Repository: Repository_1.xls
Repository Parameters	Unmapped repository parameters
Missing Recovery Scenario: RS_2	Un-Quick Test\tests\Missing Resources\Recovery_Repos\RS2.xls
Missing Function Library: Common.xls	Common.xls

### VÍ DỤ: THỰC HIỆN GHI LẠI CÁC BƯỚC KIỂM THỬ 1 WEBSITE.

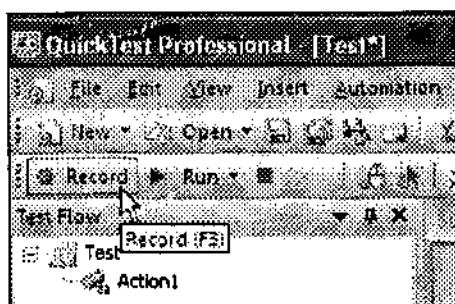
Để áp dụng các lệnh tùy chỉnh đã được trình bày ở trên để thực hiện một kiểm thử cho 1 Form trên Website.

Thực hiện khởi động chương trình Quicktest pro, sau đó trên thanh trình đơn nhấp chọn **Menu File > chọn New > Test**, tạo một kịch bản kiểm thử mới.



Bước tiếp theo, trên thanh công cụ > nhấp chọn nút **Record (F3)**, để ghi lại các bước kiểm thử.

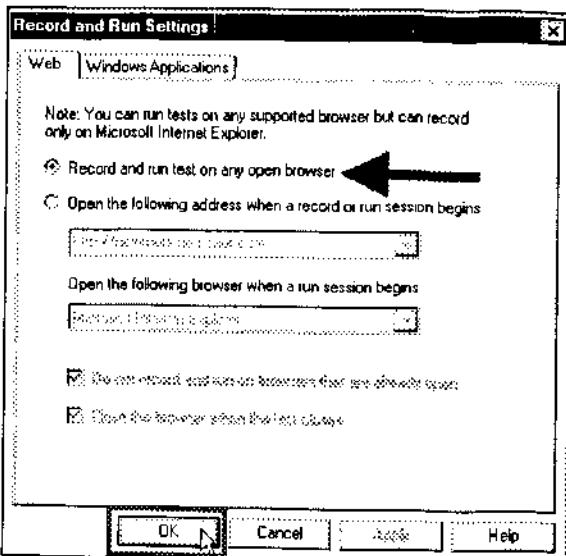
Hộp thoại **Record and Settings** xuất hiện (hình trang bên).



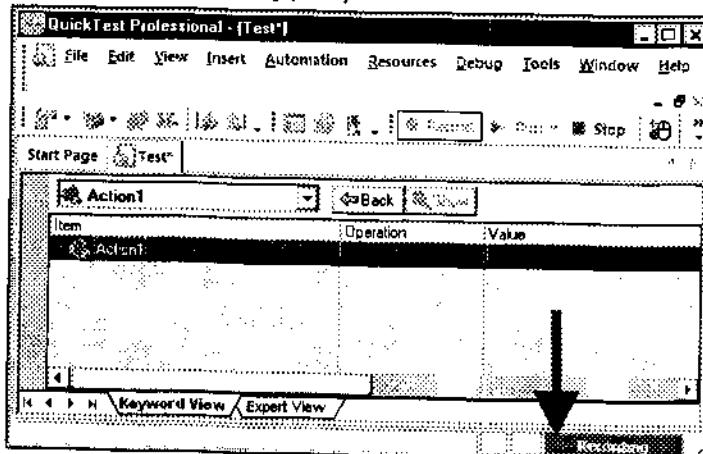
Hãy nhấp chọn lệnh **Record anh run test on any open browser**, sau đó nhấp nút **OK**, đồng ý thực hiện ghi các bước kiểm thử.

### Nội dung của các tùy chọn trong hộp thoại:

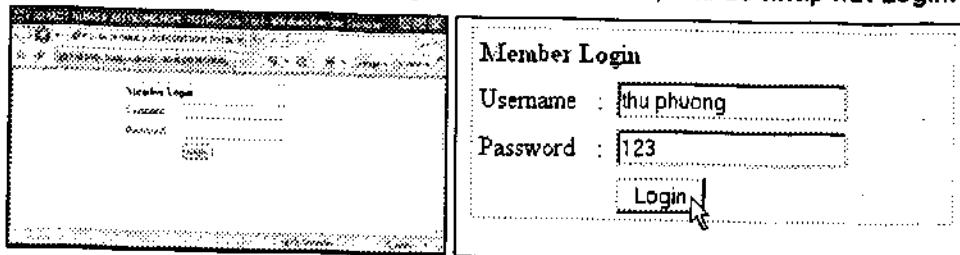
- Record anh run test on any open browser.**  
Cho phép kiểm thử trên trình duyệt hiện tại đang sử dụng trên máy.
- Open the following address when a record or run session begins:** ghi lại các bước kiểm thử trên địa chỉ được nhập vào.



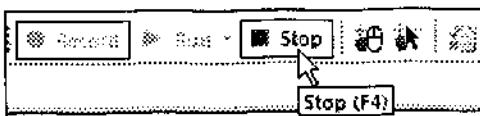
**Khung Test Action 1** xuất hiện, thông báo quá trình ghi lại đã bắt đầu, thông báo này nằm ở góc phải của giao diện chương trình. (Form dùng kiểm thử phải thực hiện bằng trình duyệt IE).



Khởi động 1 Form Login của thành viên trên trình duyệt IE, thực hiện nhập một thông tin vào các khung User và Passwork, sau đó nhấn nút Login.



Hoàn thành xong, nhấp nút Stop dừng ghi các bước kiểm thử trên Form Login.



Các bước thực hiện được ghi lại trên tab Keyword View một cách chi tiết và phân chia trong 4 cột chính: Item, Operation, Value, Documentation.

Item	Operation	Value	Documentation
myusername	Set	"thu phuong"	Enter "thu phuong" in the "myusername" edit box
mypassword	Set	"123"	Enter "123" in the "mypassword" edit box
Login	Click		Click the "Login" button.

- Item:** hiển thị mỗi bước kiểm tra theo thứ bậc và cấu trúc cây.
- Operation:** hiển thị các thao tác đã thực hiện trên các Item như: nhấp chọn hay xác nhận.
- Value:** giá trị nhập vào cho thao tác vừa chọn.
- Documentation:** mô tả ngắn gọn các thao tác.

#### Tạo tập tin cơ sở dữ liệu cho kiểm thử

Hãy gọi bảng dữ liệu bằng cách: trên thanh trình đơn nhấp chọn Menu View > đánh dấu chọn Data Table. Hộp dữ liệu sẽ xuất hiện bên dưới bảng Keyword View.

A1	B	C
1		
2		
3		
4		
5		

Mỗi một trường dữ liệu đầu vào trên form sẽ tương ứng với một cột dữ liệu trên bảng dữ liệu (dạng bảng tính Excel). Mỗi trường hợp còn lại kiểm thử sẽ thể hiện trên cột của bảng.

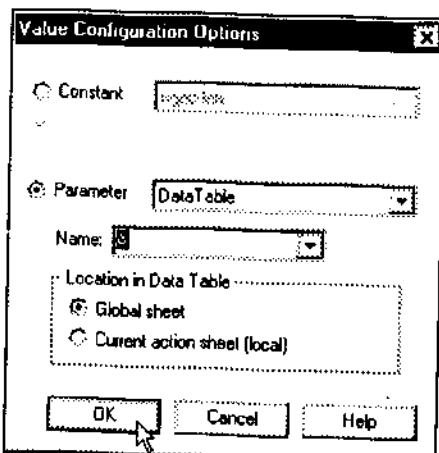
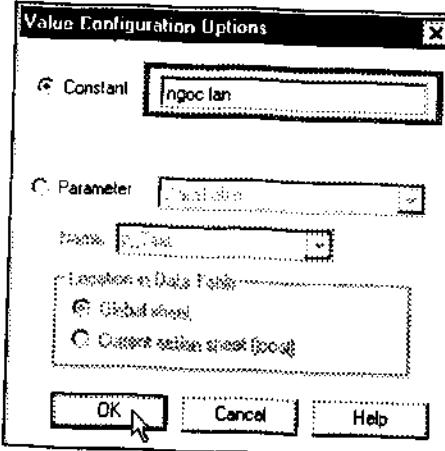
Thực hiện tùy chỉnh thông tin đã kiểm thử và thay thế chúng trong bảng dữ liệu hãy nhấp vào các cột trên khung Keyword View và nhập thông tin muốn thay thế vào.

Item	Operation	Value	Documentation
Action1			
Browser			
Page			
Set username	Set	"thu phuong"	Enter "thu phuong"
Set password	Set	Text	Configure the value (Ctrl+F11)
Login	Click		

Hộp thoại **Value Configuration Options** xuất hiện, cho phép thực hiện thay đổi các dữ liệu đã thực hiện khi ghi các bước kiểm thử.

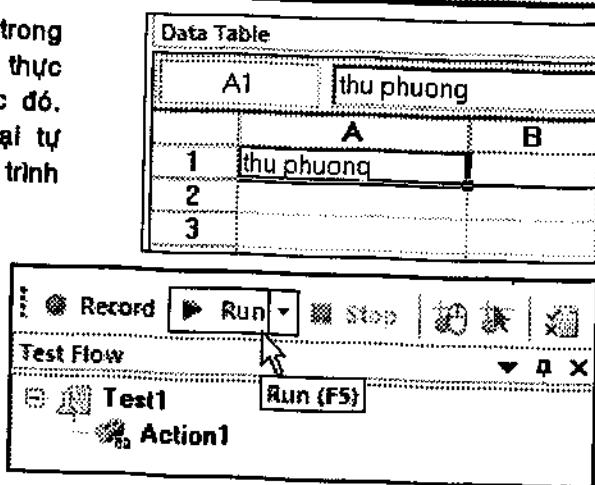
Ví dụ: nhập một **username** mới vào khung Constant, sau đó nhấp nút **OK**. Thông tin sẽ được cập nhật vào bảng dữ liệu.

Ngoài ra, có thể thực hiện chọn vị trí chèn dữ liệu vào bảng dữ liệu thông qua tùy chọn lệnh **Parameter** và chọn tên cột mà bạn muốn nhập vào. Sau đó nhấp nút **OK** hoàn thành.



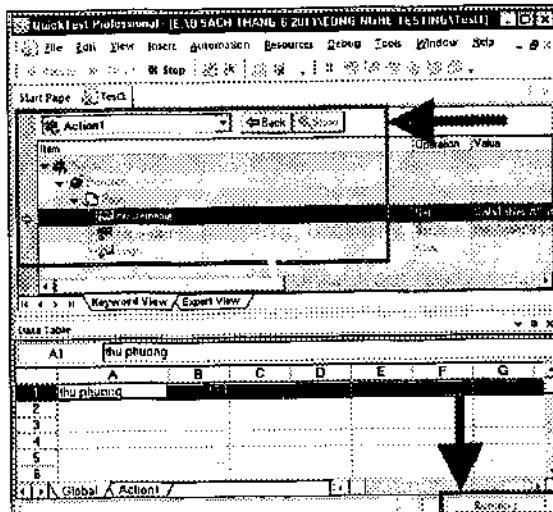
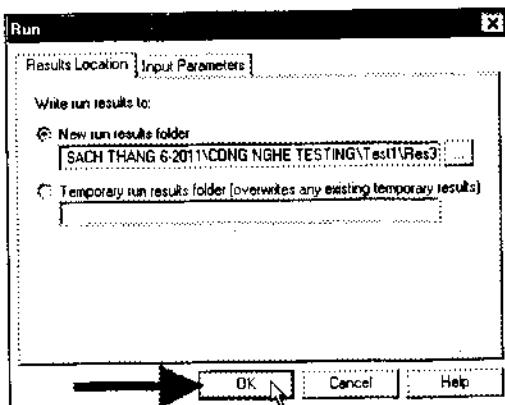
Các thông tin hiển thị trong các ô trong bảng theo thứ tự thực hiện và tùy chọn vị trí trước đó. Bảng dữ liệu sẽ được lưu lại tự động khi hoàn thành quá trình kiểm thử.

Sau khi thực hiện xong các bước tạo bảng dữ liệu, hãy nhấp chọn nút **Run** trên thanh công cụ, hoặc nhấn phím **F5**.

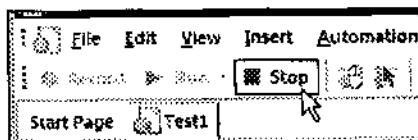


Hộp thoại Run xuất hiện, yêu cầu chọn đường dẫn đến thư mục chứa cuộc kiểm thử. Sau đó nhấp nút OK để thực hiện chạy các kiểm thử. Nếu muốn chạy 1 kiểm thử khác hãy nhấp nút ở cuối dòng New run results folder.

Một cửa sổ chạy chương trình hoạt động, cho thấy các action được vận hành kiểm tra.



Sau khi chạy xong, nhấp nút Stop để dừng quá trình kiểm thử. Bảng Test Results xuất hiện, chứa các kết quả kiểm tra về form.



Trong bảng thông báo cho biết quá trình kiểm tra và các lỗi, các cảnh báo liên quan đến tất cả các dữ liệu bên trong action. Từng phần trong bước kiểm tra sẽ có một báo cáo kết quả chi tiết đi kèm.

Thông qua báo cáo này người kiểm thử có thể thực hiện một bảng dữ liệu đầy đủ về các lỗi chức năng của ứng dụng. Khung bên trái trong cửa sổ kết quả kiểm thử có chứa các kết quả chạy dưới dạng cây. Cửa sổ bên phải có chứa các chi tiết cho một bước lựa chọn trong cây kết quả chạy. Các chi tiết cho một bước lựa chọn có thể bao gồm: một bảng tóm tắt kiểm tra, chi tiết các bước, hình ảnh ứng dụng, phím của ứng dụng, hoặc kết quả các hệ thống kiểm tra.

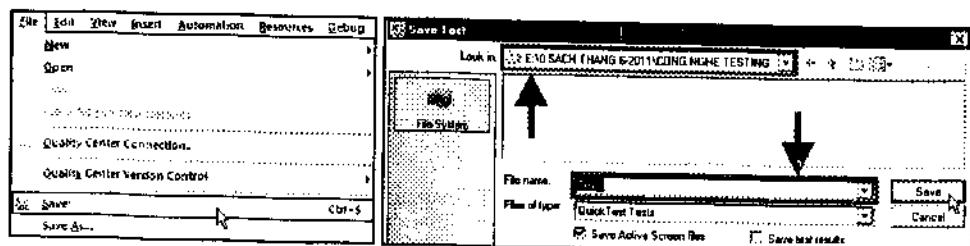
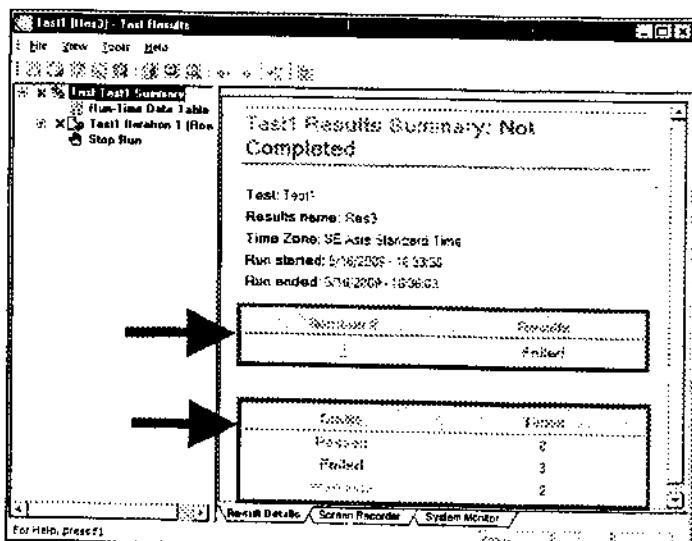
Ngoài ra, có thể mở cửa sổ kết quả thử nghiệm như một ứng dụng độc lập từ trình đơn Start. Để mở cửa sổ kết quả thử nghiệm, chọn Start > Programs > QuickTest Professional > Test Results Viewer. Các thử nghiệm được thể hiện trong ví dụ trên bao gồm ba lần lặp lại, như thể hiện trong cây kết quả chạy.

Cửa sổ kết quả kiểm thử có chứa các yếu tố chính sau đây:

- **Test result title bar:** hiển thị tiêu đề của bài kiểm thử.
- **Menu bar:** hiển thị các menu lệnh có sẵn
- **Run results toolbar:** có các nút để xem kết quả kiểm thử.
- **Run results tree:** chứa các bước kiểm thử đã chạy dưới dạng cây.
- **Result Details tab:** hiển thị chi tiết các lựa chọn bước chạy trong cây, thông tin chi tiết sẽ hiển thị ở khung bên phải.
- **Screen Recorder Tab:** hiển thị phim được ghi lại từ cuộc kiểm thử.
- **System Monitor tab:** hiển thị đồ thị kết quả cho các phần hệ thống được kích hoạt dùng trong kiểm thử.
- **Status bar:** hiển thị 5 trạng thái lệnh đang được chọn.

Sau khi thực hiện xong bước trên, nhấp chọn File > Save để lưu kết quả quá trình kiểm thử.

Hộp thoại Save Test xuất hiện, nhấp chọn đường dẫn lưu file và đặt tên cho cuộc kiểm thử, sau đó nhấp nút Save lưu kết quả.



## CHƯƠNG 11

**BÀI TẬP KIỂM THỬ WEBSITE**

Chương 11 là bài tập nhỏ thực hành kiểm thử một website giúp chúng ta hiểu rõ hơn các bước thực hiện quy trình kiểm thử đã được trình bày lý thuyết ở chương 10 bằng chương trình QUICKTEST PRO.

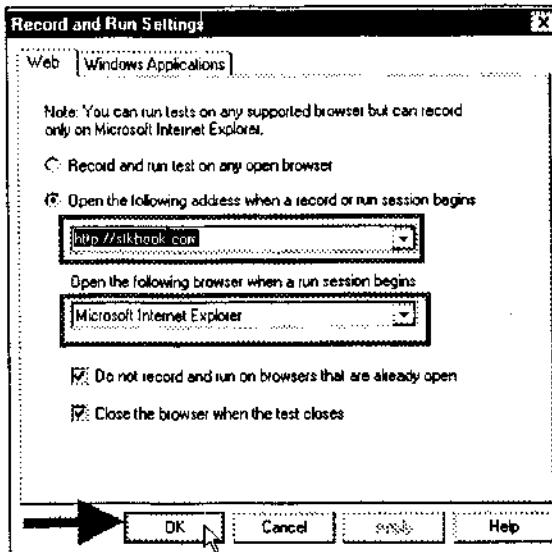
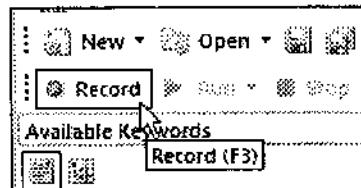
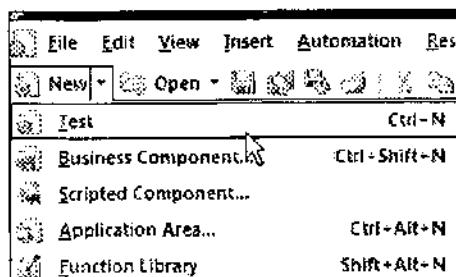
Các bước thực hiện: khởi động chương trình QUICKTEST PRO và chuẩn bị một trang web chạy trên trình duyệt IE.

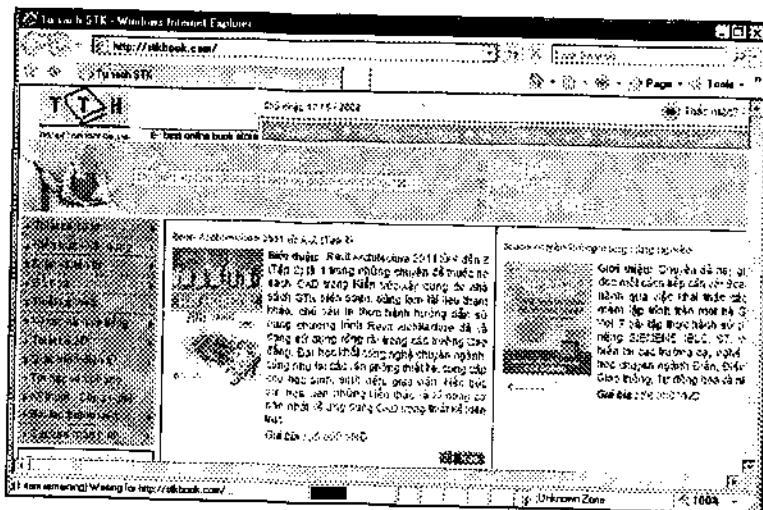
Trên thanh công cụ, nhấp chọn **New > Test**, tạo một kiểm thử mới. Trước khi khởi tạo hãy tùy chọn kiểm thử là Web, mở trang web cần kiểm thử, trong ví dụ này ta mở trang web sau: <http://www.stkbook.com> (các bạn có thể chọn một trang web khác để kiểm thử).

Kế tiếp nhấp chọn nút **Record** để bắt đầu các thao tác kiểm thử. Hộp thoại **Record and Run Settings** xuất hiện, nhập địa chỉ trang web cần kiểm thử vào khung **Open the following address...**.

Kế tiếp, nhấp chọn trình duyệt thực hiện kiểm thử là **Microsoft Internet Explorer**. Sau đó nhấp chọn nút **OK** để ghi kết quả.

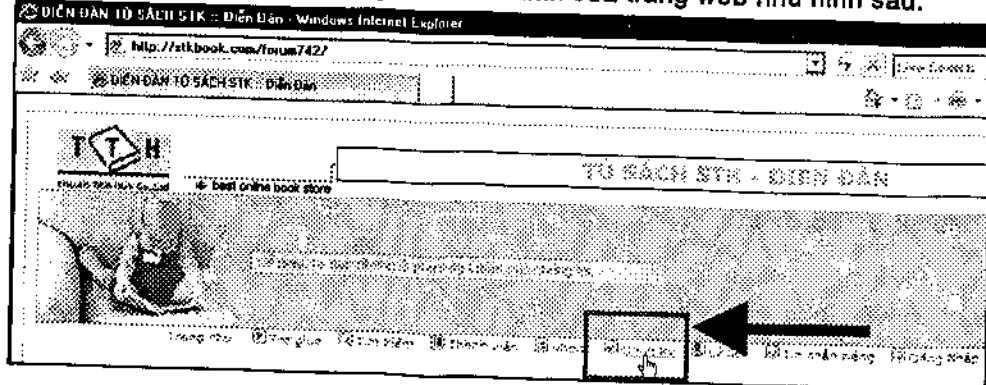
Chương trình bắt đầu hiển thị quá trình ghi, và gọi trang web đã nhập vào khung yêu cầu của hộp thoại như hình bên. Trên trang web, thực hiện thao tác tại khu vực muốn ghi lại kết quả kiểm thử. Tất cả các thao tác sẽ được ghi lại từng bước.



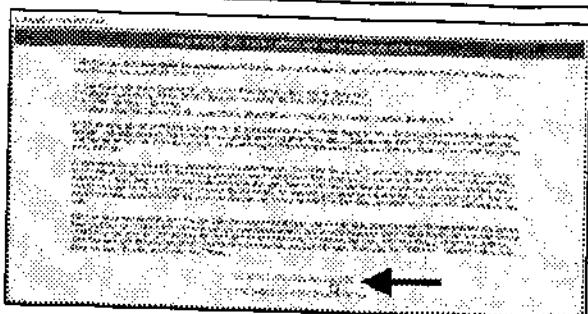


Ta thực hiện kiểm thử chức năng đăng ký vào diễn đàn của trang web.

Nhấp chọn nút Đăng ký trên diễn đàn của trang web như hình sau:

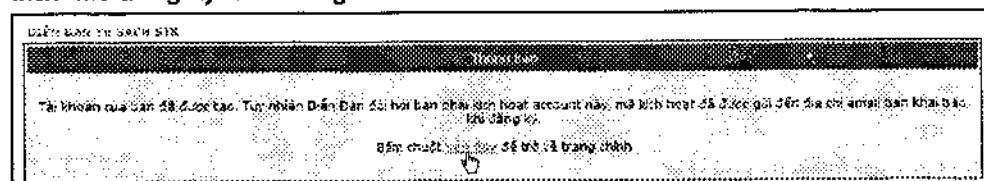


Kế tiếp, nhấp chọn Tôi đồng ý với những quy định trên để đăng ký vào diễn đàn. Thực hiện theo các hướng dẫn của form đăng ký. Trong từng khung hãy nhập các giá trị kiểm tra cần thiết để tìm ra lỗi.

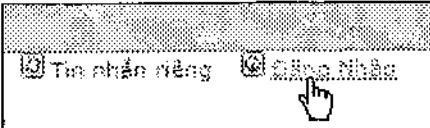


Nhập các tên truy cập, địa chỉ email, mật khẩu, code kiểm tra... các bước thao tác này sẽ được chương trình ghi lại một cách chi tiết để có thể phát hiện lỗi xảy ra trong form đăng ký này.

Sau khi thực hiện xong, nhấp nút đồng ý đăng ký và thực hiện kết thúc thao tác đăng ký trên trang web [stlbook.com](http://stlbook.com).



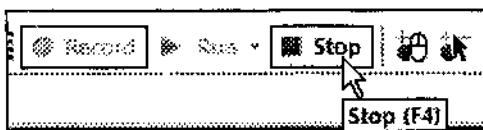
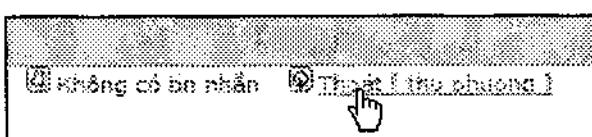
Với tài khoản đăng ký kiểm tra, chúng ta thực hiện đăng nhập vào diễn đàn và tiến hành kiểm tra chức năng của khung đăng nhập.



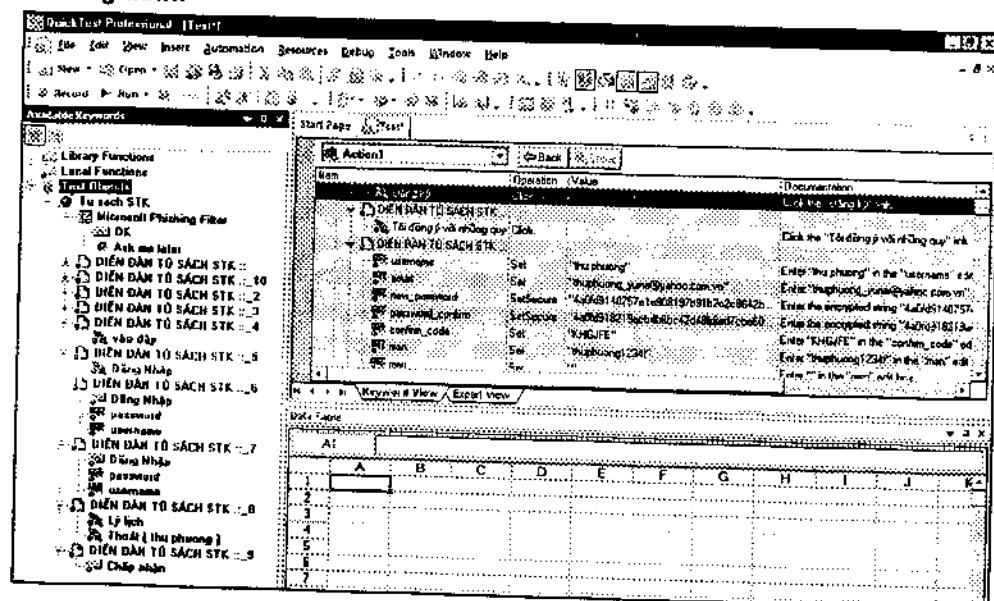
Trong khung đăng nhập, thực hiện điền thông tin người truy nhập và mật khẩu đã khai báo trước đó như hình dưới.

Nếu kiểm tra đăng nhập thành công, thì tiếp tục thực hiện bước thoát khỏi trang web.

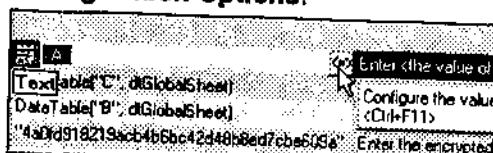
Sau khi hoàn tất các kiểm thử, hãy nhấp nút Stop trên thanh công cụ để dừng các hoạt động.



Tất cả các hoạt động trên trang web sẽ được ghi lại trên giao diện chương trình.



Để cập nhật dữ liệu của các bước kiểm thử trong action, nhấp chọn vào dòng Value và mở hộp thoại Value Configuration Options.



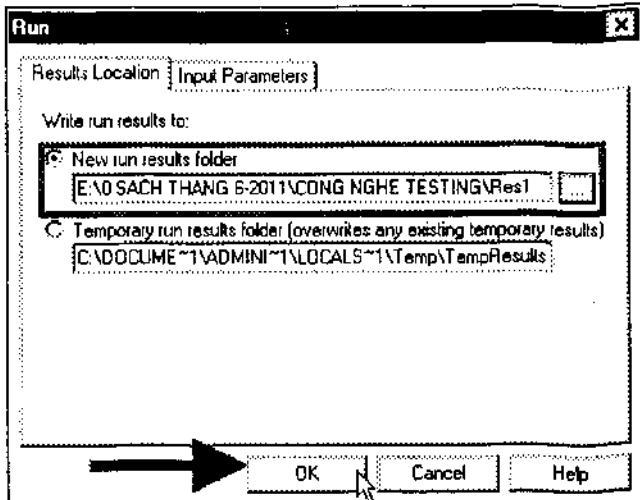
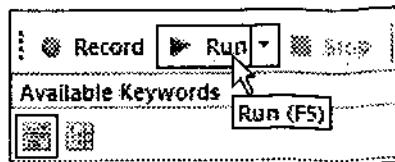
Trong hộp thoại, nhập chọn cột trong bảng cơ sở dữ liệu mà ta muốn gán dữ liệu trong action vào, sau đó nhấp nút OK.

Thực hiện các bước tương tự, để tạo ra một bảng cơ sở dữ liệu cho cuộc kiểm thử. Phân chia các cột trong bảng dữ liệu và cập nhật tất cả các thông tin giá trị trong cuộc kiểm thử vào bảng. Bảng này sẽ giúp người kiểm thử dễ dàng thống kê lỗi và cho ra báo cáo kiểm thử.

Data Table			
	USER	PASS	EMAIL
1	thu phuong	4a0fd9140757e1e908197b91b2c2c0642b41	thuphuong_yuna@yahoo.com.vn
2			
3			

Bước tiếp theo, thực hiện chạy kiểm thử các action đã ghi lại để thu kết quả báo lỗi của ứng dụng web. Hãy nhấp chọn nút Run để thực hiện chạy (hoặc nhấp phím F5).

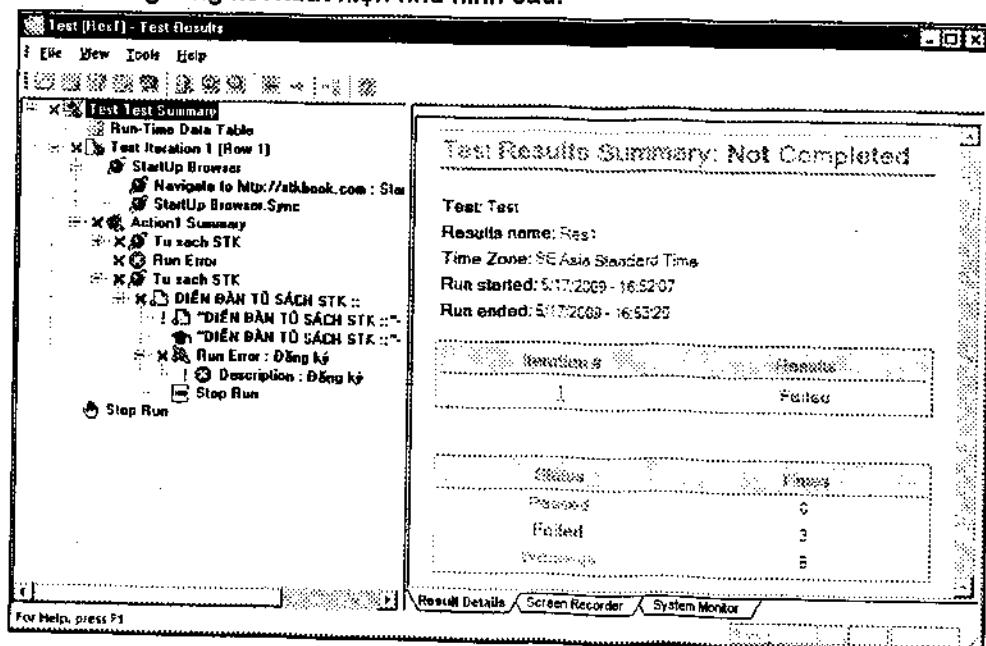
Hộp thoại Run xuất hiện, nhập đường dẫn đến nơi lưu file Test trong mục New run results folder sau đó nhấp nút OK đồng ý. Sau khi chạy xong, nhấp nút Stop để dừng. Lúc này chương trình sẽ cung cấp một bảng thông báo các lỗi xảy ra trong ứng dụng đã kiểm thử.



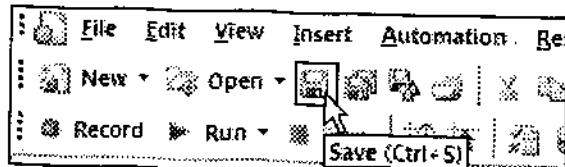
Chương trình còn ghi lại phím của quá trình kiểm lỗi. Từng bước thực hiện kiểm tra lỗi sẽ được thể hiện trong bảng theo mô hình hình cây và bảng tổng kết.

Item	Operation	Value
1. DIEN DAN TU SACH STK... 4		
2. DIEN DAN TU SACH STK... 3		
3. DIEN DAN TU SACH STK... 2		
4. DIEN DAN TU SACH STK... 1		

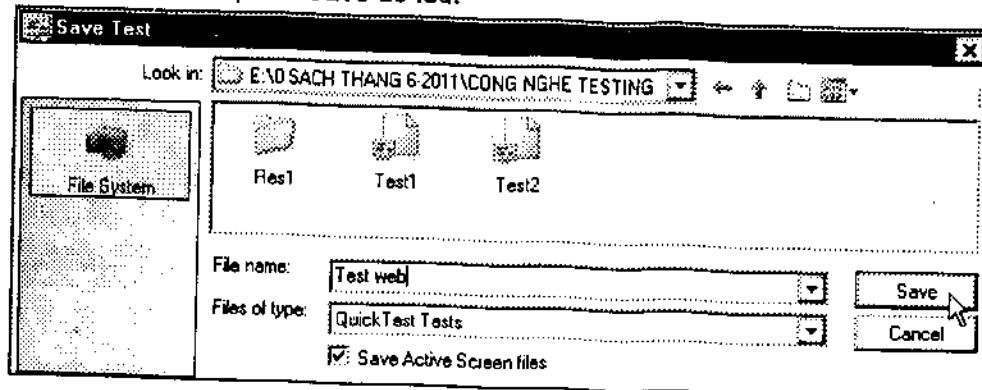
Bảng tổng kết xuất hiện như hình sau:



Sau khi thực hiện xong, hãy nhấp chọn biểu tượng Save trên thanh công cụ.



Hộp thoại Save Test xuất hiện, chương trình lưu kết quả kiểm tra tại thư mục đã khai báo trước đó (trong mục Look in). Nhập tên file trong khung File name sau đó nhấp nút Save để lưu.



**CHƯƠNG 12****LÀM VIỆC VỚI ỨNG DỤNG****TẠO KHO ĐỔI TƯỢNG:**

Chương 12 giới thiệu các đối tượng kiểm tra, thời gian chạy đối tượng và mô tả cách để tạo và chia sẻ kho đối tượng.

Nội dung chương gồm 3 phần chính sau:

- Giới thiệu đối tượng kiểm tra và đối tượng kho lưu trữ.
- Tìm hiểu các đối tượng trong ứng dụng.
- Sử dụng nhiều kho lưu trữ đối tượng.

**Đối tượng kiểm tra và đối tượng kho lưu trữ**

Trước khi tạo ra một kiểm thử, việc đầu tiên là cần phải thiết lập các nguồn dữ liệu sẽ được sử dụng kiểm thử. Một trong những dữ liệu quan trọng nhất để kiểm tra là kho lưu trữ đối tượng. Một kho lưu trữ đối tượng là một kho chứa các đối tượng kiểm thử được sử dụng.

Đối tượng kiểm tra được lưu trữ các đại diện của các đối tượng thực tế (hoặc điều khiển) ứng dụng. Chương trình tạo ra các đối tượng kiểm tra bằng việc ghi nhận một tập hợp lựa chọn các thuộc tính và giá trị của các đối tượng trong ứng dụng.

Chương trình sử dụng thông tin để xác định các đối tượng run-time trong ứng dụng. Mỗi đối tượng kiểm tra là một phần của một hệ thống phân cấp đối tượng kiểm tra. Ví dụ: một đối tượng liên kết có thể là một phần của một hệ thống phân cấp trình duyệt (Browser)/Trang (Page)/Link.

Đối tượng cấp cao, chẳng hạn như Browsers, được biết đến như các đối tượng chứa đối tượng thấp hơn, chẳng hạn như các đối tượng frame.

**Lưu ý:** kho đối tượng cũng có thể bao gồm các đối tượng trạm kiểm soát. Đối tượng loại này được bao phủ trong tạo điểm kiểm tra và sử dụng chức năng.

Đối tượng Run-time được tạo ra và duy trì trong một phiên hoạt động của chương trình. Trong một phiên chạy, chương trình thực hiện quy định đối tượng thử nghiệm phương pháp trên đối tượng thời gian chạy. Thời gian chạy các đối tượng được lưu trữ trong một kho lưu trữ đối tượng, nó có sẵn trong thời gian chạy kiểm thử. Sau đó, sử dụng hộp thoại Spy Object để xem tài nguyên và hoạt động của các đối tượng trong ứng dụng của bạn.

### Đối tượng Repositories

Chương trình có thể lưu trữ các đối tượng kiểm thử trong hai loại đối tượng kho lưu trữ tập tin: chia sẻ (shares) và địa phương (local).

Một kho lưu trữ đối tượng chia sẻ chứa các đối tượng kiểm tra có thể được sử dụng trong nhiều action. Tính linh hoạt này làm cho các loại kho lưu trữ được ưa thích để lưu trữ và duy trì các đối tượng kiểm tra.

Bằng cách kết hợp một kho lưu trữ đối tượng chia sẻ với một action, làm cho các đối tượng kiểm tra trong kho lưu trữ có sẵn để sử dụng trong action đó. Bất kỳ tùy chỉnh sửa chữa thực hiện trên một đối tượng trong một đối tượng chia sẻ kho được phản ánh trong bất kỳ bước nào sử dụng đối tượng.

Một đối tượng kho lưu trữ địa phương với các cửa hàng kiểm tra đối tượng có thể được sử dụng trong một action cụ thể. Các đối tượng trong kho lưu trữ loại địa phương không được sử dụng trong bất kỳ action nào khác.

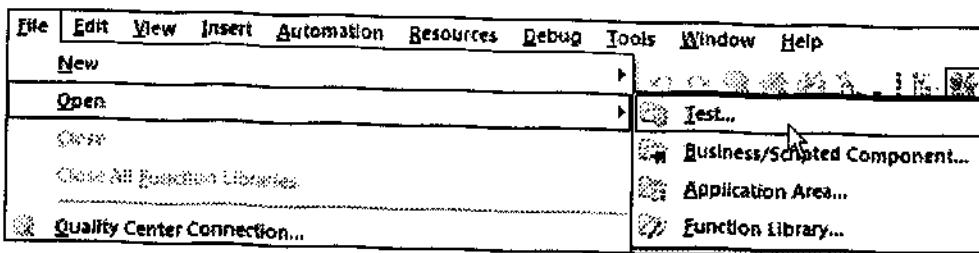
Các kho đối tượng Local rất hữu ích cho việc sao lưu các đối tượng kiểm thử. Khi tạo ra một kho lưu trữ đối tượng thì phải chọn các đối tượng cần thiết cho cuộc kiểm thử. Như vậy kho lưu trữ sẽ nhỏ và giúp việc bảo trì nhanh chóng hoặc lựa chọn các đối tượng dễ dàng hơn. Ngoài ra, cần cung cấp tên hợp lý để người khác dễ dàng lựa chọn chính xác các đối tượng tạo ra và sửa đổi các kiểm thử.

Trong một phiên làm việc, chương trình sẽ tham khảo các đối tượng được lưu trữ trong kho và thực hiện các hoạt động tương ứng trong ứng dụng. Phần tiếp theo, hướng dẫn sử dụng chương trình để tìm hiểu thêm về các tùy chọn đối tượng từ trang web STKBOOK.COM. Sẽ thực hiện 1 action và lưu trữ kho các đối tượng trong kiểm thử.

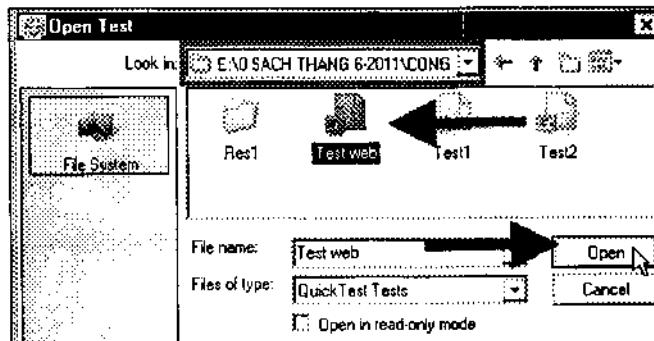
**Bắt đầu kiểm thử:** thực hiện khởi động chương trình, sau đó chọn ngôn ngữ kiểm thử là web và khởi động trang web có liên quan trên trình duyệt IE.

**Mở 1 action trước đó bằng cách:**

Trên thanh trình đơn, nhấp chọn **Menu File > Open > Test** (sử dụng bài tập test web ở chương trước).



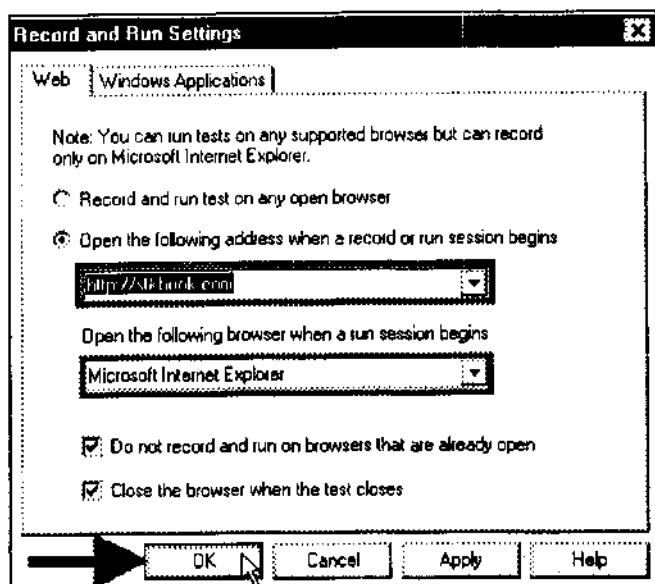
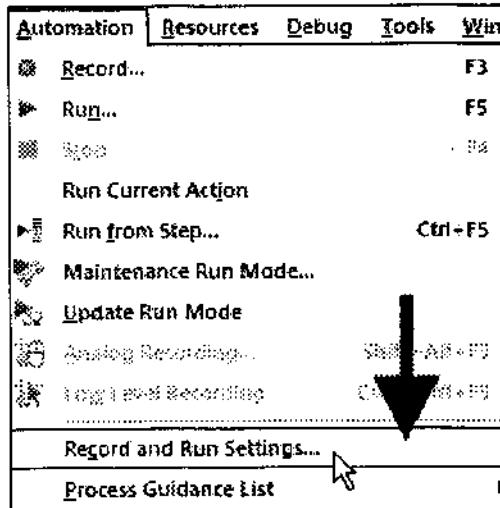
Hộp thoại **Open Test** xuất hiện, chỉ đường dẫn đến file Test web đã lưu ở bài tập trước, nhấp chọn gói **Test case** và nhấn nút **Open** để mở.



**Thiết lập lại quá trình ghi kiểm thử:**

Trên thanh trình đơn Automation > nhấp chọn Record and Run Settings...

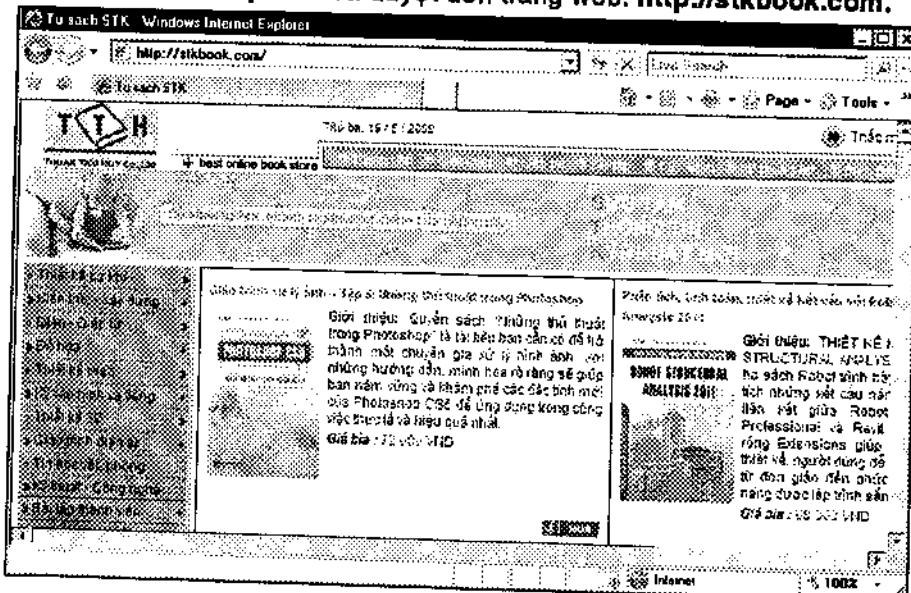
**Hộp thoại Record and Run Settings** xuất hiện, nhập lại và xác nhận lại đường dẫn của website là: <http://stkbook.com>. Trong khung tiếp theo, thực hiện tùy chọn trình duyệt sẽ hoạt động kiểm thử. **Microsoft Internet Explorer** là trình duyệt được sử dụng trong hướng dẫn này.



### Đánh dấu xác nhận vào 2 chỉ mục bên dưới:

- Xác nhận không ghi lại và chạy các trình duyệt đã mở.
  - Đóng trình duyệt khi thực hiện đóng cuộc kiểm thử được lựa chọn.
- Sau cùng nhấp nút **OK** để thực hiện kiểm thử.

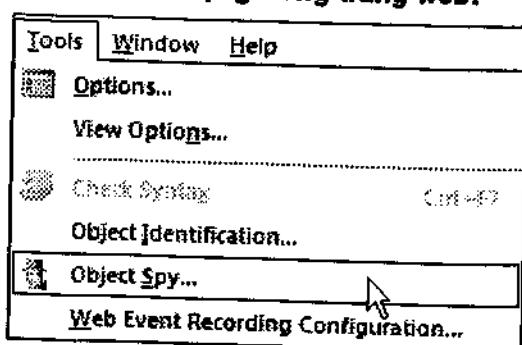
Mở Internet Explorer và duyệt đến trang web: <http://stkbook.com>.



### Xem thuộc tính và hoạt động của các đối tượng trong trang web:

Trên thanh trình đơn, nhấp chọn **Menu Tools** > chọn lệnh **Object Spy** mở hộp thoại Spy.

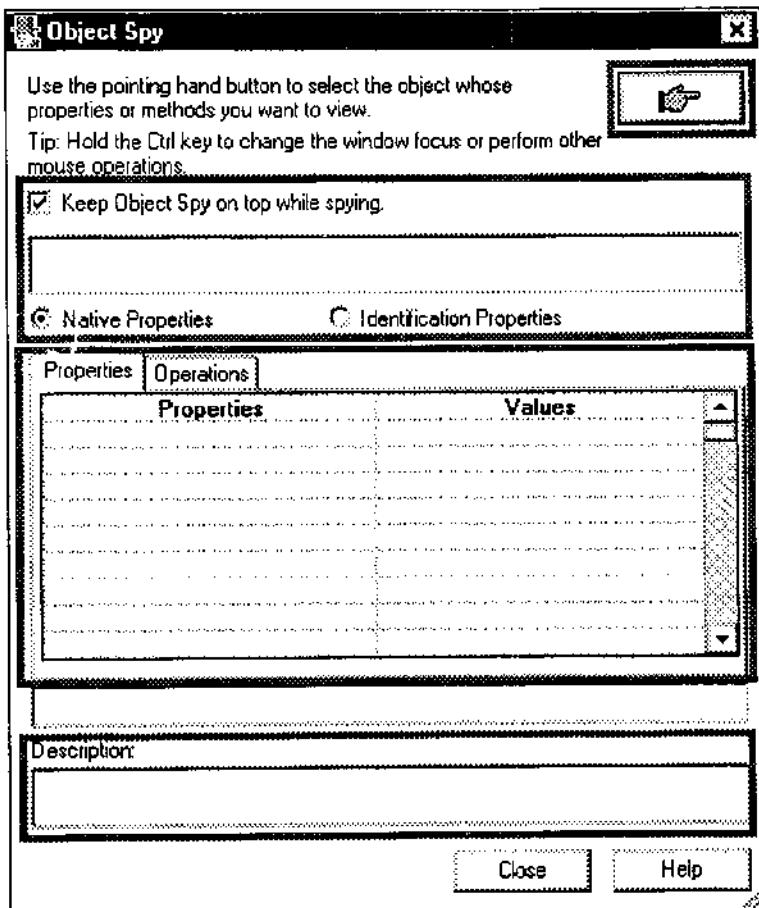
Hộp thoại Object Spy xuất hiện, cho phép xem các thuộc tính của từng hoạt động trên trang web. Khi khởi động hộp thoại để xem hoạt động, giao diện chương trình sẽ tự động ẩn đi.



### Các chức năng của Object Spy

Trong hộp thoại có 9 thành phần và chia làm 4 cụm lệnh. Khi muốn xem hoạt động của thành phần nào thì nhấp chọn vào đối tượng đó trên trang web và hoạt động của nó sẽ hiển thị dưới dạng cây quản lý và thuộc tính trong bảng Properties.

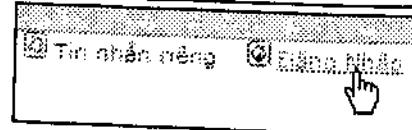
**Chú ý:** chỉ thành phần được nhấp chọn trên trang web mới có thể xem thuộc tính của nó trong bảng khi kết thúc bước chọn.



Tùy chọn và biểu tượng	Công dụng
	Khi nhấp chuột chọn vào biểu tượng này, con trỏ chuột sẽ làm nổi các vùng chọn đối tượng trên Website có thuộc tính muốn xem. Khi di chuyển con trỏ dưới dạng bàn tay đến vị trí các đối tượng trong ứng dụng, những chi tiết có liên quan sẽ hiển thị trong hộp thoại Object Spy.
<b>Tùy chọn: Keep Object Spy on top while spying.</b>	Nhấp chọn hộp kiểm này để giữ lại hộp thoại Object Spy khi đã thực hiện nhấp chọn đối tượng trên web. <b>Chú ý:</b> nếu gỡ bỏ đánh dấu này thì hộp thoại sẽ bị ẩn đi khi chọn đối tượng.

Khung hiển thị cây phân cấp (Test Object hierarchy tree).	<p>Cho phép hiển thị cây phân cấp các thứ bậc hoạt động của đối tượng kiểm tra đã nhấp chọn trước đó.</p> <p>Khi nhấp chọn 1 đối tượng trong cây, nó sẽ được tô đậm và hiển thị các thuộc tính của nó vào khung Properties bên dưới. Để xem các thuộc tính, giá trị, hoặc các hoạt động cho một đối tượng trong cây hiển thị, hãy nhấp chọn đối tượng kiểm tra trong cây.</p>
Tùy chọn: <b>NativeProperties/Operations</b>	Tùy chọn này để hiển thị thuộc tính khu vực ứng dụng hoặc các hoạt động của đối tượng được lựa chọn trong cây hệ thống phân cấp.
<b>Tùy chọn: Identification Properties/Operations</b>	Tùy chọn này để hiển thị thuộc tính xác định và các hoạt động của đối tượng kiểm tra nằm trong cây hệ thống phân cấp.
<b>Tab Properties</b>	<p>Hiển thị các thuộc tính bản địa hoặc các thuộc tính xác định cho các đối tượng được chọn trong cây. Bên trong Tab có chứa 2 cột là Properties và Values:</p> <ul style="list-style-type: none"> <li>➤ <b>Properties:</b> hiển thị tên thuộc tính xác định của đối tượng được lựa chọn.</li> <li>➤ <b>Values:</b> hiển thị giá trị của thuộc tính.</li> </ul>
<b>Tab Operations</b>	Hiển thị các hoạt động địa phương và các hoạt động của đối tượng kiểm tra, kết hợp với cú pháp tương ứng của nó.
Khung hiển thị thuộc tính, giá trị, hoạt động	Tùy vào tùy chọn tab bên trên mà khung cho biết tên thuộc tính và cú pháp, giá trị của từng dòng đối tượng được chọn gần nhất.
<b>Khung Description</b>	Cho phép cập nhật thông tin liên quan đến các thuộc tính hay hoạt động có sẵn.

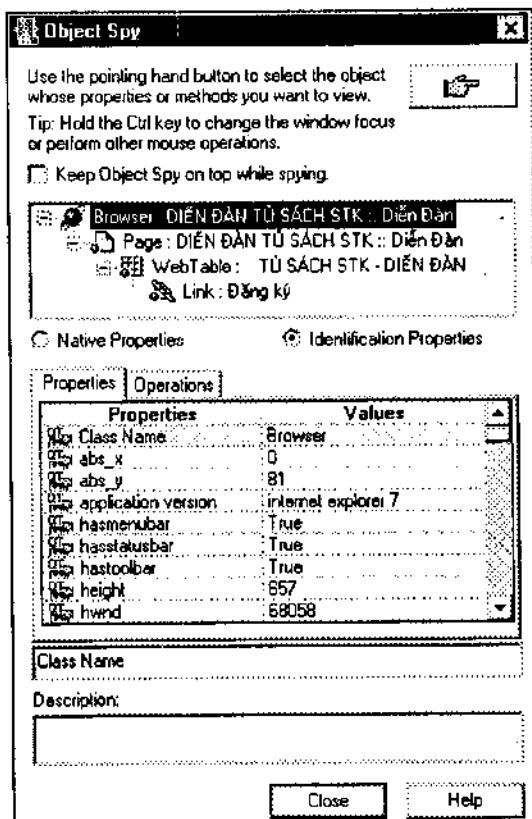
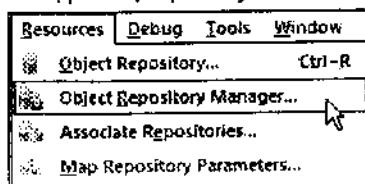
Hãy nhấp chọn vào biểu tượng bàn tay và nhấp chọn vào chức năng Đăng nhập forum của trang web như hình bên:



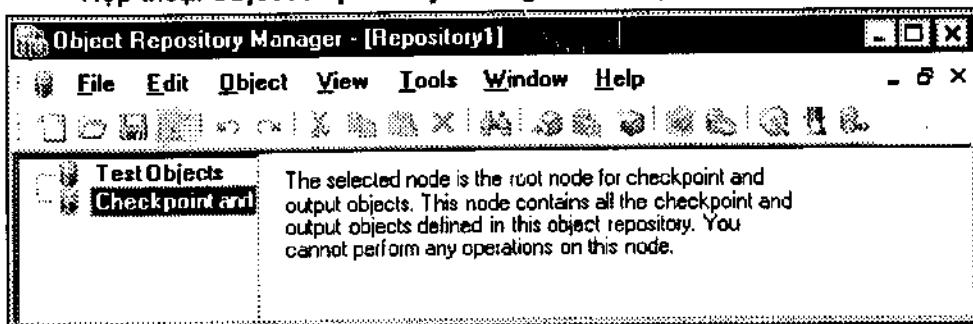
Các thuộc tính của đối tượng trong các bước chọn sẽ hiển thị trong tab Properties và Operations. Các đối tượng trong hệ thống sẽ phân chia theo cấp bậc hình cây như hình bên. Tùy chọn các thành phần muốn xem thông tin thuộc tính hệ thống và hoạt động. Sau đó, nhấp rút **Close** để đóng hộp thoại Object Spy.

Quản lý đối tượng trong cuộc kiểm thử:

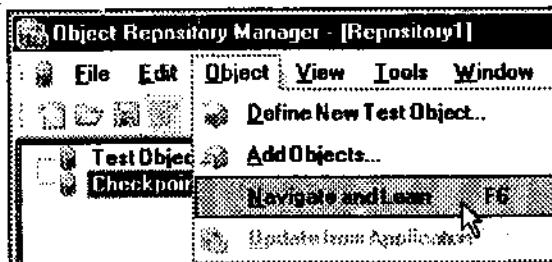
Trên thanh trình đơn Resources, nhấp chọn lệnh **Object Repository Manager**, mở hộp thoại quản lý kho.



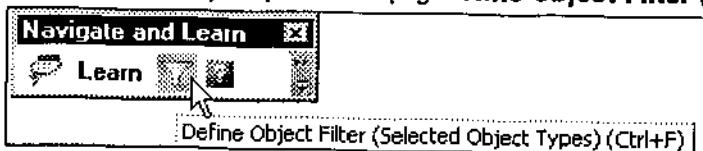
Hộp thoại Object Repository Manager xuất hiện như hình dưới.



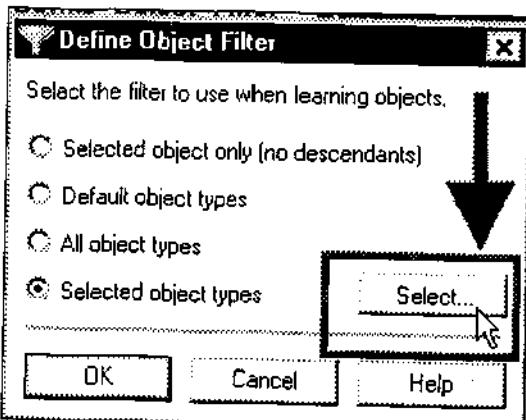
Bước kế tiếp, trên thanh trình đơn của hộp thoại chọn Menu **Object** > chọn lệnh **Navigate and Learn** (F6). Ngay sau đó cửa sổ chương trình và Object Repository Manager đều bị ẩn đi.



Trong hộp thoại, nhấp chọn biểu tượng Define Object Filter (Ctrl + L).



**Hộp thoại Define Object Filter** xuất hiện, hộp thoại giúp các bạn thêm một số đối tượng vào kho lưu trữ đối tượng, chúng thường là một đối tượng bao hàm như một trình duyệt hoặc trang trong một môi trường Web, hộp thoại ứng dụng Windows... Bên trong hộp thoại có các tùy chọn giúp lựa chọn các đối tượng lọc ra từ danh sách các đối tượng trong ứng dụng.



#### Hộp thoại có các tùy chọn sau đây:

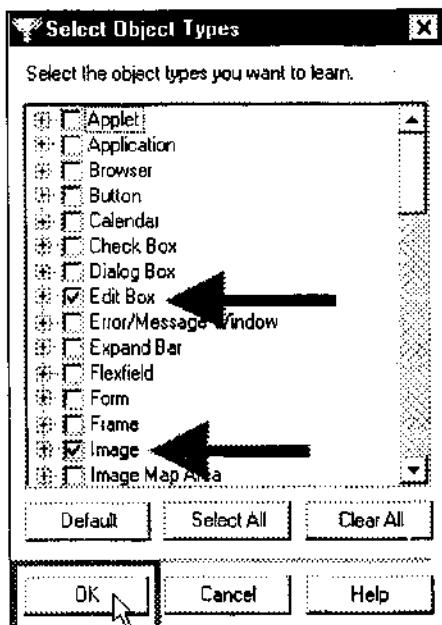
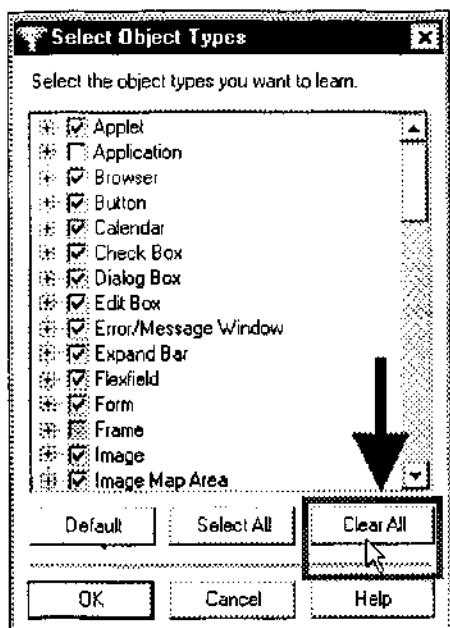
**Selected object only (no descendants):** thêm các đối tượng đã chọn trước cùng với thuộc tính và các giá trị của nó vào kho đối tượng mà không có các thuộc tính hay giá trị của đối tượng con.

**Default object types:** thêm vào kho đối tượng các thuộc tính và giá trị của đối tượng cho trước đó. Các thuộc tính và giá trị của đối tượng con được thêm vào tuân theo quy định của bộ lọc mặc định. Để có thể xác định bộ lọc mặc định, các bạn có thể lựa chọn nút Default trong hộp thoại chứa bộ lọc.

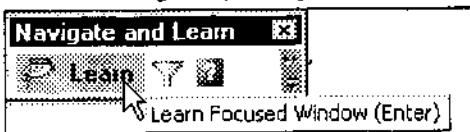
**All object types:** thêm tất cả các thuộc tính và giá trị của đối tượng vào kho đối tượng mà không phân biệt đối tượng con.

**Selected object types:** cho phép tùy chọn các thuộc tính và giá trị của đối tượng sẽ thêm vào kho. Các loại thuộc tính và giá trị được định sẵn trong bộ lọc và phân ra theo nhánh, có thể dùng nút Select để đánh dấu chọn các thành phần thuộc tính hay giá trị cần thiết và bỏ dấu Check để gỡ bỏ các thành phần không cần thiết khác.

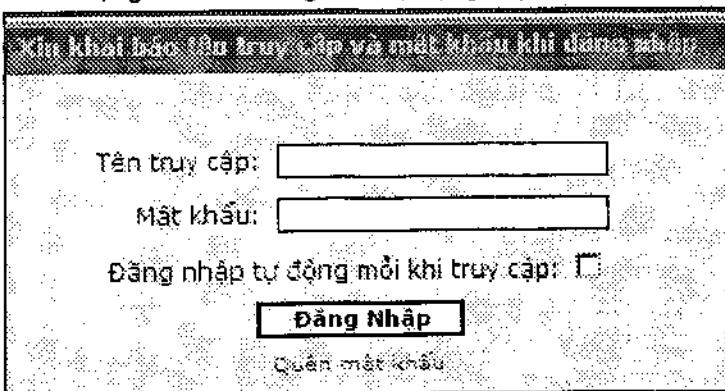
Trong hộp thoại, nhấp chọn vào chế độ Selected object types, sau đó nhấp nút Select để mở bảng tùy chọn đối tượng ra. Hộp thoại Select Object Types xuất hiện hiển thị bộ lọc các đối tượng có thể tùy chọn. Kế tiếp, nhấp chọn nút Clear All, để xóa hết các đối tượng đã đánh dấu theo mặc định. Hãy thực hiện đánh dấu vào 2 đối tượng là Edit Box và Image như hình minh họa trang bên, sau đó nhấp nút OK để hoàn thành bước lựa chọn đối tượng kiểm thử).



Quay trở lại giao diện làm việc của chương trình, nhấp chọn vào trang web sau đó nhấp nút **Learn** trong hộp thoại **Navigate and Learn** để ghi nhận tất cả thành phần của Form đăng nhập trang web.



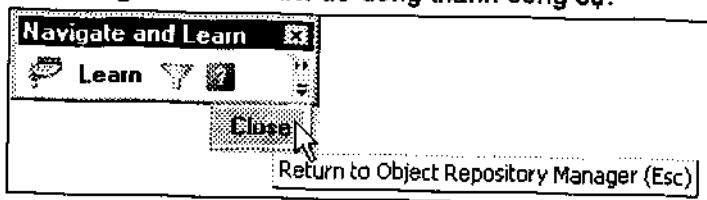
Trên trang web hãy nhấp chọn Form đăng nhập như hình, quá trình xử lý ghi nhận đối tượng sẽ do chương trình tự động cập nhật.



Tìm hiểu các loại đối tượng được lựa chọn từ trang web [stlbook.com](http://stlbook.com):

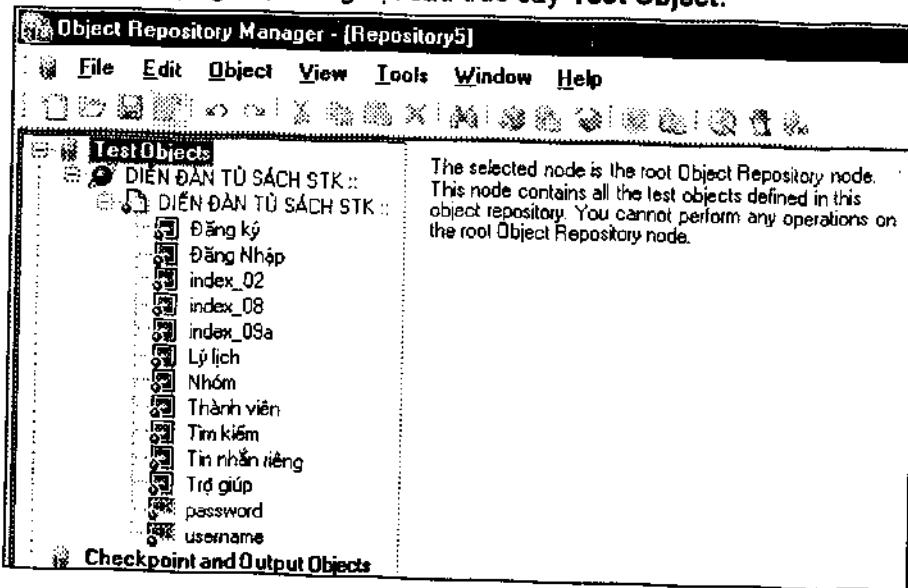
Thực hiện kiểm tra các thành phần của đối tượng có phù hợp với các bộ lọc đã chọn và thực hiện thêm chúng vào kho lưu trữ.

Đã hoàn tất việc thêm các đối tượng vào kho đối tượng, nhấp nút **Close** trên hộp thoại **Navigate and Learn** để đóng thanh công cụ.

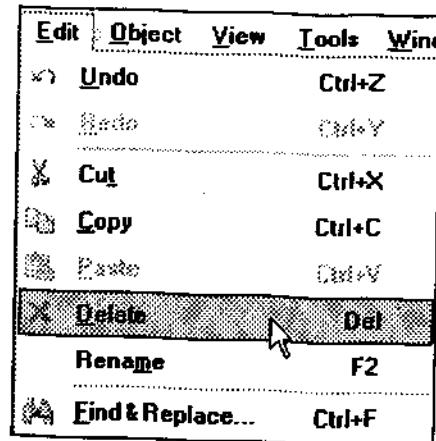


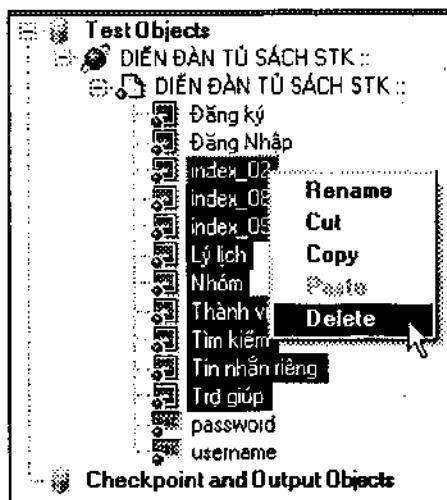
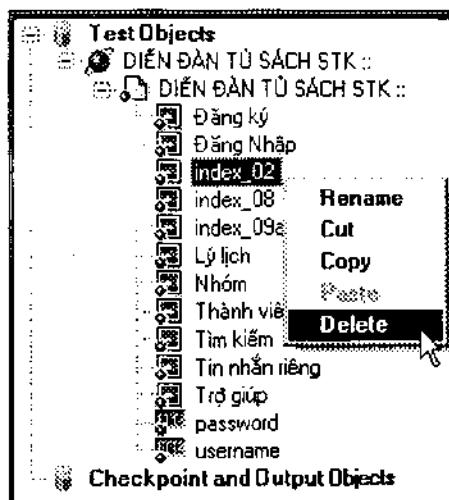
Giao diện làm việc chương trình và hộp thoại **Object Repository Manager** lúc này xuất hiện trở lại.

Trong hộp thoại **Object Repository Manager**, tất cả các thành phần của đối tượng sẽ được ghi lại trong cột cấu trúc cây **Test Object**.



Kiểm tra lại danh sách các đối tượng, thực hiện loại bỏ các thành phần không cần thiết và không liên quan. Hãy nhấp chọn các thuộc tính này trong cây thuộc tính. Xóa tất cả các đối tượng kiểm tra thành phần: đăng ký, password, và username. Nhấp lựa chọn các đối tượng và chọn **Edit > Delete** hoặc nhấp nút **Delete**. Bạn có thể nhấp chọn nhiều đối tượng bằng cách sử dụng phím **SHIFT** và **CTRL** sau đó nhấp chuột phải và chọn lệnh **Delete** trong cửa sổ mới mở.

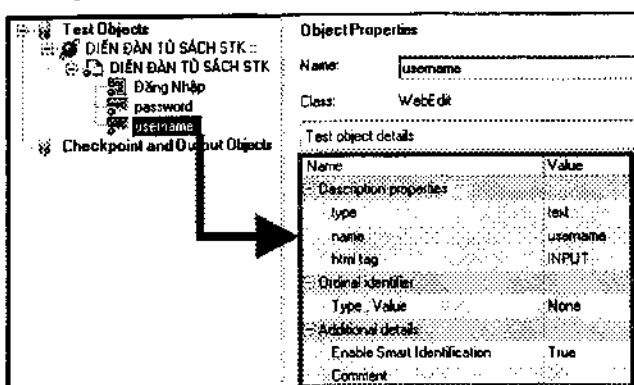
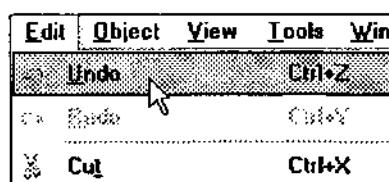
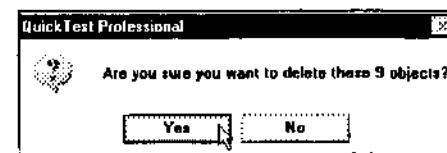




Hộp thoại thông báo xuất hiện, hỏi lại một lần nữa bạn có chắc chắn xóa các đối tượng đã chọn hay không. Nhấp nút Yes để xác nhận.

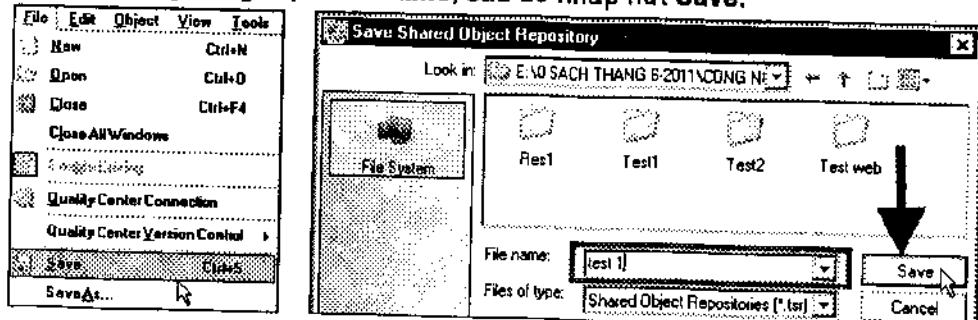
Trên cây còn lại các đối tượng như hình bên. Nếu xóa nhầm đối tượng “Đăng nhập”, password hoặc username do nhầm lẫn, hãy nhấp nút lệnh Undo khôi phục lại các đối tượng đã bị xóa.

Trong cây đối tượng kiểm thử, hãy nhấp chọn username, khung bên phải xuất hiện thông báo các thuộc tính đối tượng. Hình dưới là một số mô tả các thuộc tính và giá trị của đối tượng.



### Lưu trữ kho đối tượng

Trong cửa sổ Object Repository Manager, nhấp chọn File > Save, thực hiện lưu. Hộp thoại Save Shared Object Repository xuất hiện, hãy nhập tên của đối tượng trong mục File name, sau đó nhấp nút Save.

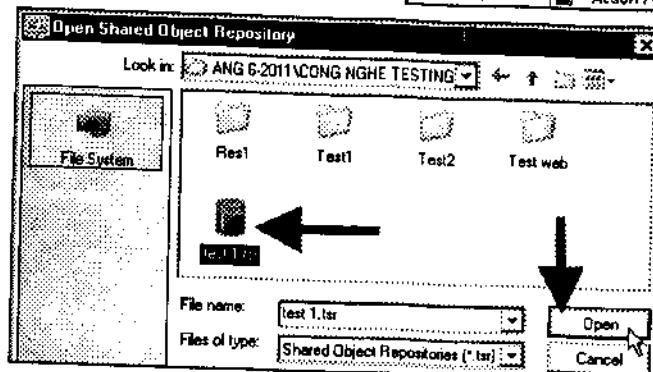
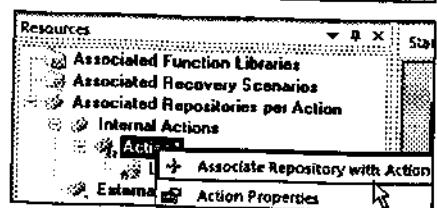
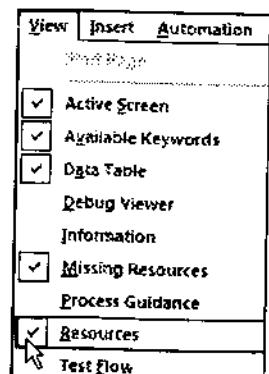


### Gọi đối tượng trong kho lưu trữ đối tượng

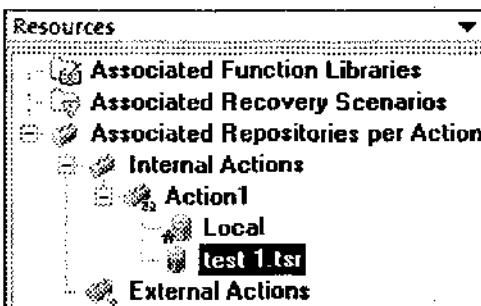
Trên thanh trình đơn chính của giao diện chương trình, nhấp chọn Menu View, tiếp theo đánh dấu chọn lệnh Resources.

Bảng Resources xuất hiện dưới dạng cây như hình dưới, nhấp vào nút cộng trên cây và nhấp phải vào thành phần Action1 sau đó chọn lệnh Associate Repository with Action. Hộp thoại Open Shared Object Repository xuất hiện, nhấp chọn đường dẫn đến thư mục lưu trữ thuộc tính đối tượng.

Trong thư mục, nhấp chọn đối tượng test1.tsr, sau đó nhấp nút Open để mở. Đối tượng sẽ được nhập vào chương trình, trở thành một thành phần dữ liệu của Action.



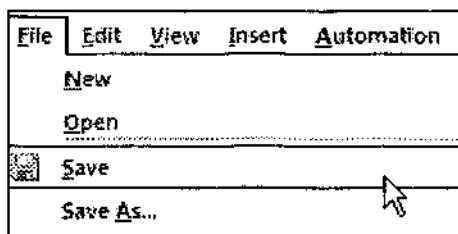
Những gì hiển thị bên dưới action, được col là thành phần con. Liên kết một đối tượng trong kho lưu trữ với một hành động cho phép bạn sử dụng bất kỳ đối tượng từ kho lưu trữ và trong bất cứ bước nào trong hành động liên quan.



### Lưu cuộc kiểm thử

Chọn **File > Save** hoặc nhấp vào nút **Save**.

**Lưu ý:** nếu một bài kiểm tra đã thay đổi và không được lưu, tên kiểm tra trong thanh tiêu đề có chứa một dấu hoa thị.



### Sử dụng nhiều đối tượng trong kho

Trong phần trước, chúng ta đã tạo một kho lưu trữ đối tượng. Tiếp theo, bạn sẽ tạo ra một chia sẻ với đối tượng lưu trữ trong kho cho các trang còn lại trong trang web. Người kiểm thử nên luôn luôn tạo ra một kho lưu trữ đối tượng chia sẻ riêng biệt cho mỗi trang trong trang web hoặc từng khu vực của ứng dụng. Điều này làm cho nó dễ tìm ra đối tượng thích hợp khi thêm hoặc sửa đổi các bước kiểm tra hoặc khi thực hiện nhiệm vụ bảo trì khi các đối tượng được thêm vào, gỡ bỏ, hoặc sửa đổi trong ứng dụng.

Theo mục đích của hướng dẫn này, chúng ta dùng chương trình để tìm hiểu tất cả các đối tượng trên mỗi trang.

Tại thời điểm này, không cần phải kết hợp kho đối tượng với một hành động cụ thể.

Nếu đối tượng vẫn còn mở, hãy đăng nhập vào diễn đàn của trang web:

- Truy cập bằng **username** và **password**.
- Nhấp vào nút đăng nhập, để mở trang diễn đàn của trang web ra.

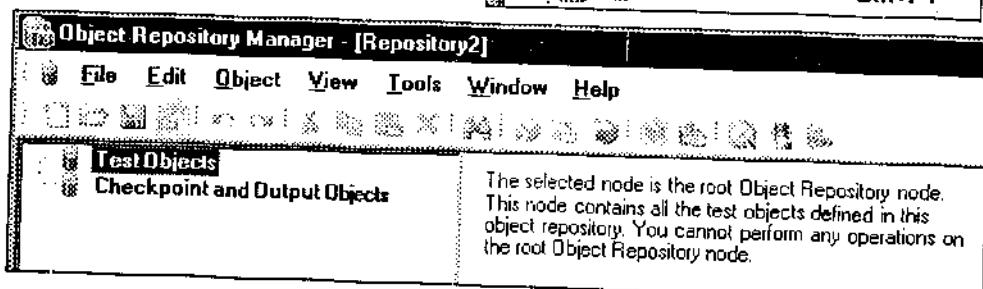
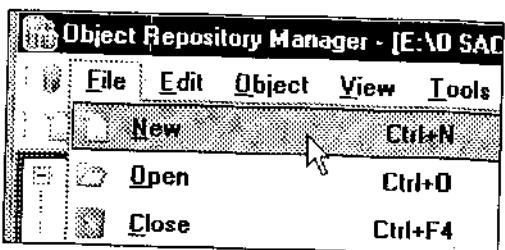
### Tạo một kho lưu trữ đối tượng chia sẻ

Mở lại cửa sổ quản lý đối tượng **Object Repository Manager** bằng cách sau: trên thanh trình đơn **Resources**, nhấp chọn **Object Repository Manager**, cửa sổ kho quản lý sẽ mở ra.

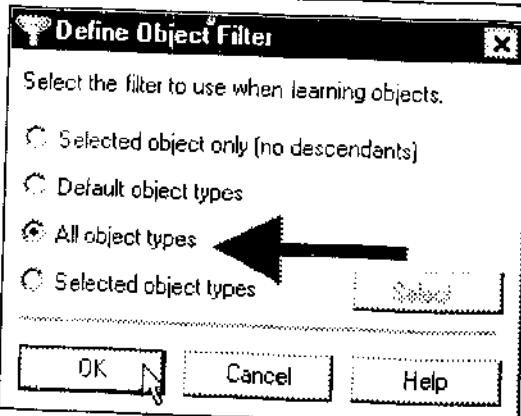
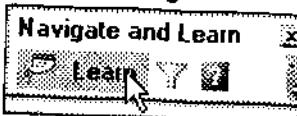
Kế tiếp, trên thanh trình đơn của cửa sổ nhấp chọn **File > New (Ctrl + N)** tạo một kho đối tượng mới.

Một hộp thoại kho đối tượng mới xuất hiện, bên trong không chứa bất cứ đối tượng nào.

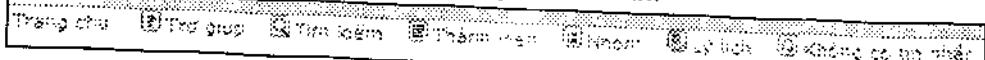
Thực hiện tiếp bước thiết lập bộ lọc các đối tượng như hướng dẫn ở trên.



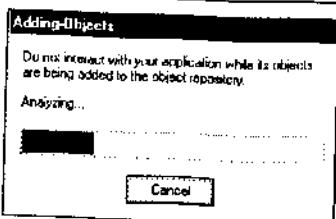
Hãy mở hộp thoại **Define Object Filter**, trong hộp thoại xác định đối tượng lọc, hãy chọn tất cả các loại đối tượng và nhấn nút **OK** xác nhận. Nhấp chọn trang diễn đàn và nhấp nút **Learn** để ghi nhận tất cả thành phần trong diễn đàn của trang web.



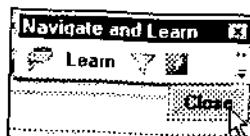
Các thành phần trong trang diễn đàn bao gồm: Trang chủ, Trợ giúp, Tìm kiếm, Thành viên, Nhóm, Lý lịch, Không có tin nhắn.



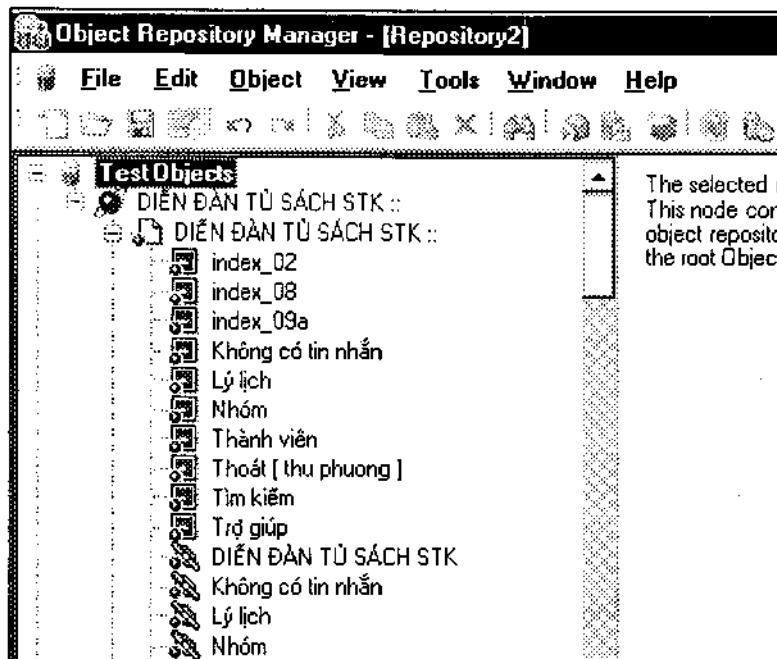
Khi nhấp chọn các khung thành phần trong diễn đàn, chương trình sẽ xuất hiện hộp thoại thông báo đang xử lý các dữ liệu đối tượng liên quan. Hộp thoại **Adding Objects** xuất hiện, cho thấy tiến trình diễn ra, nếu không muốn cập nhật đối tượng nữa, hãy nhấp nút **Cancel**.



Sau khi thực hiện cập nhật thông tin của tất cả các đối tượng, nhấp chọn nút **Close** để đóng hộp thoại **Navigate and Learn** và dừng việc cập nhật.



Các thành phần đối tượng sẽ được đưa vào kho dữ liệu một cách đầy đủ như hình sau:

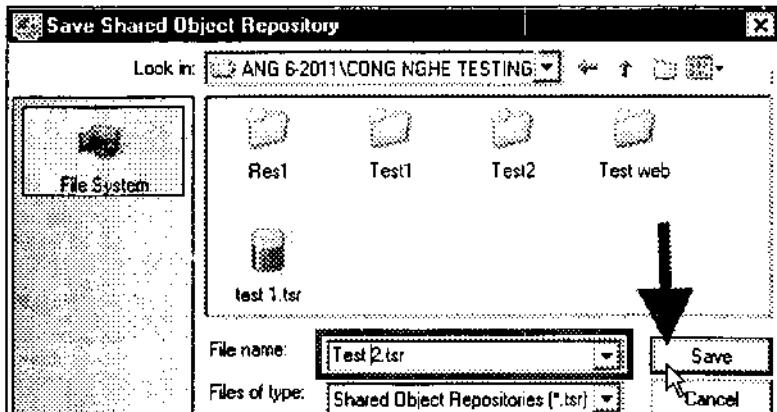


Trong bước này, dùng chương trình để tìm hiểu tất cả các đối tượng trong trang web phù hợp với bộ lọc và để thêm chúng vào một kho lưu trữ đối tượng chia sẻ.

#### **Lưu trữ đối tượng vào kho**

Trong cửa sổ Object Repository Manager, chọn File > Save (hoặc nhấn vào biểu tượng Save).

Hộp thoại Save Shared Object Repository xuất hiện, nhập tên của kho đối tượng và chỉ đường dẫn đến thư mục lưu trữ, sau cùng nhấn nút Save.



## TẠO CÁC HÀM VÀ THƯ VIỆN HÀM

Chương trình QUICKTEST cho phép xây dựng các hàm và phương thức để đáp ứng nhu cầu kiểm thử của người dùng. Tuy nhiên, người dùng cần tương tác thêm để tạo ra các thành phần hàm hay phương thức phù hợp với cuộc kiểm thử. Người dùng có thể tạo ra tập tin văn bản và lưu nó vào tập tin hệ thống, hoặc có thể tạo thêm một bước truy cập dữ liệu từ bảng tính Excel.

### Tìm hiểu về cách tạo hàm và thư viện hàm:

Chúng ta sẽ sử dụng các chức năng thư viện của chương trình tạo ra hàm và liên kết nó với cuộc kiểm thử. Kết hợp một thư viện hàm với một cuộc kiểm thử cho phép chương trình gọi bất kỳ hàm nào trong thư viện từ cuộc kiểm thử đó.

Hàm là một tập hợp các bước được mã hóa để thực hiện một nhiệm vụ cụ thể. Người dùng có thể dùng những thuật ngữ riêng để định nghĩa cho hàm thực hiện các hoạt động không có sẵn để sử dụng cùng với 1 lớp đối tượng cụ thể trong kiểm thử. Trong phần thực hành dưới đây, chúng ta sẽ thực hiện tạo một hàm để kiểm tra định dạng ngày tháng năm trong chức năng tạo bởi trang diễn đàn từ trang web.

### Các bước tạo hàm

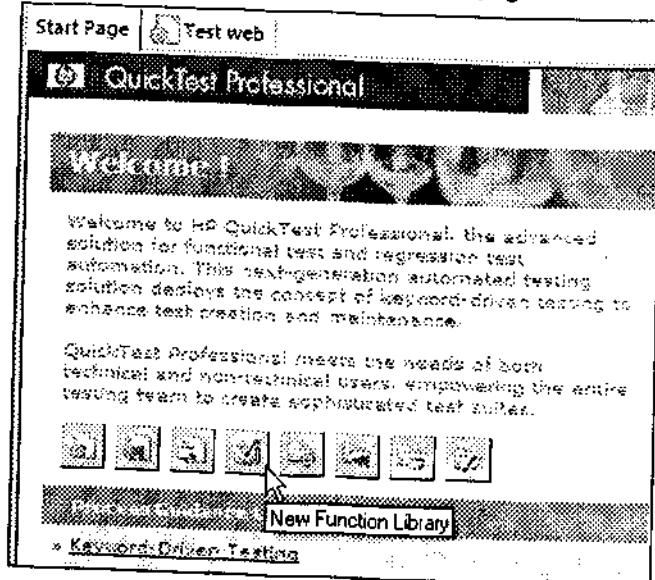
Thực hiện tạo hàm và gọi lên trong kiểm thử, nó có chức năng kiểm tra định dạng MM/DD/YYYY. Hàm nhằm xác định tính hợp lệ của các ngày nhập vào từ form đăng ký thành viên của trang.

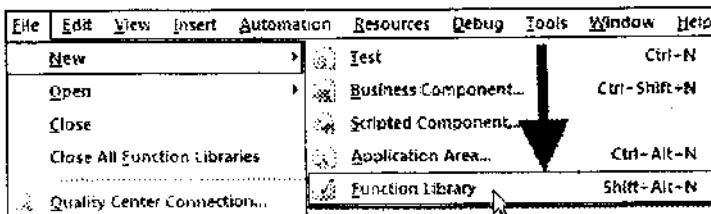
Bước đầu tiên, khởi động chương trình và tạo 1 hàm mới. Nếu đã mở chương trình thì hãy đảm bảo chỉ có trang diễn đàn được hoạt động.

Trên tab Start Page, nhấp chọn lệnh New Function Library để tạo mới một thư viện hàm như hình bên.

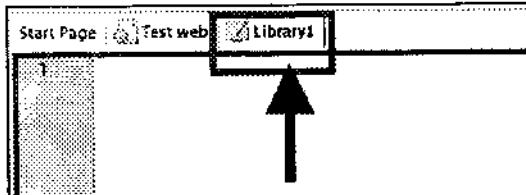
### Chú ý

Nếu chương trình mở mà không hiển thị cửa sổ Start, hãy nhấp chọn File > New > Function Library hoặc nhấp vào mũi tên hướng xuống tại nút New và chọn Function Library.





Một cửa sổ thư viện xuất hiện, cho phép nhập hàm. Hãy nhập đoạn hàm kiểm tra định dạng MM/DD/YYYY dưới đây vào khung Library.



```

Function check_data_validity( dateStr )
    Dim firstSlashPos, secondSlashPos
    Dim mmPart, ddPart, yyyyPart
    firstSlashPos = InStr( dateStr, "/" )
    secondSlashPos = InStrRev( dateStr, "/" )
    If ( firstSlashPos <> 3 or secondSlashPos <> 6 ) Then
        reporter.ReportEvent micFail, "Format check", "Date string is
missing at least one slash ( / )."
        check_data_validity = False
    End If
    Exit function
    mmPart = Mid( dateStr, 1, 2 )
    ddPart = Mid( dateStr, firstSlashPos + 1, 2 )
    yyyyPart = Mid( dateStr, secondSlashPos + 1, 4 )
    If mmPart > 12 Then
        reporter.ReportEvent micFail, "Format Check" , "The month value is
invalid. It exceeds 12."
        check_data_validity = False
    End If
    If ddPart > 31 Then
        reporter.ReportEvent micFail, "Format Check" , "The date value is
invalid. It exceeds 31."
        check_data_validity = False
    End If

```

```

End If
If yyyyPart < 2000 Then
    reporter.ReportEvent micFail, "Format Check", "The year value is
invalid. (Prior to 2000)"
check_data_validity = False
Exit function
End If
check_data_validity = True
End Function

```

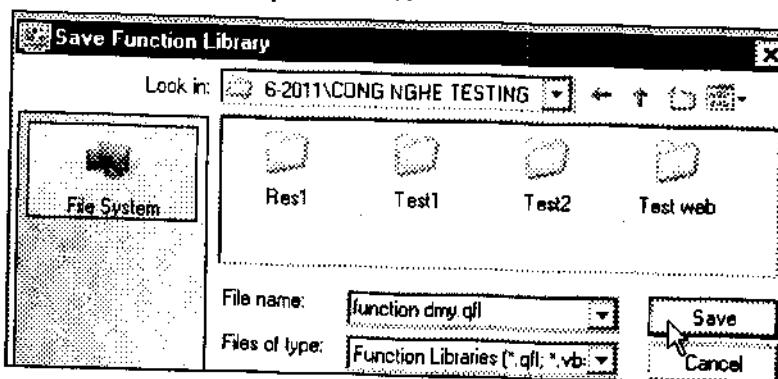
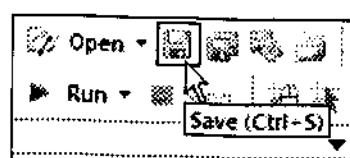
Trong bảng Library, nội dung của hàm nhập vào như hình sau:

```

Start Page | Test web | Library1 |
Function check_data_validity(dateStr)
1 Dim firstSlashPos, secondSlashPos
2 Dim mmPart, ddPart, yyyyPart
3
4
5 firstSlashPos = InStr(dateStr, "/")
6 secondSlashPos = InStrRev(dateStr, "/")
7 If (firstSlashPos <> 3 Or secondSlashPos <> 6) Then
8     reporter.ReportEvent micFail, "Format check", "Date string is missing at least one slash (/)"
9     check_data_validity = False
10    Exit function
11 End If
12 mmPart = Mid(dateStr, 1, 2)
13 ddPart = Mid(dateStr, firstSlashPos + 1, 2)
14 yyyyPart = Mid(dateStr, secondSlashPos + 1, 4)
15
16 If mmPart > 12 Then
17     reporter.ReportEvent micFail, "Format Check", "The month value is invalid. It exceeds 12."
18     check_data_validity = False
19     Exit function
20 End If

```

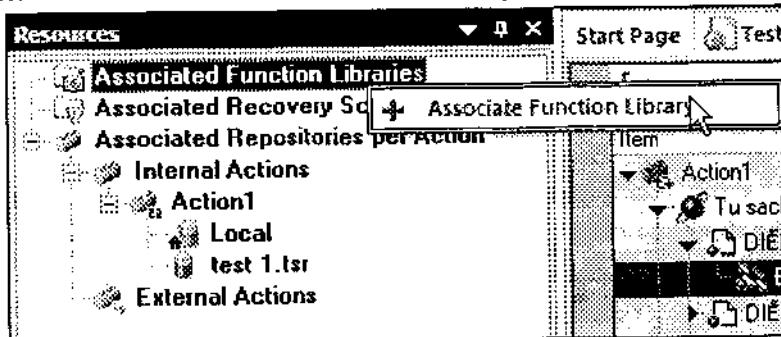
Bước sau cùng, thực hiện lưu hàm này vào thư viện bằng cách: nhấp chọn biểu tượng **Save (Ctrl + S)**. Hộp thoại **Save Function Library** xuất hiện, nhập tên của file hàm vào khung **File name** sau đó nhấp nút **Save**.



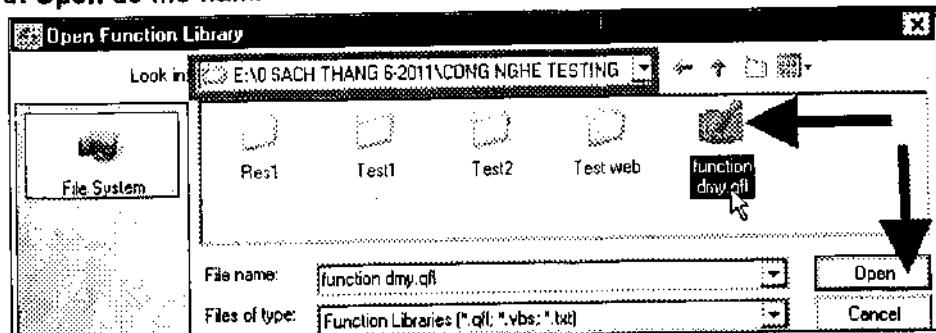
### Liên kết thư viện hàm với cuộc kiểm thử

Chúng ta đã tạo ra một hàm và lưu trữ trong thư viện hàm, bước kế tiếp cần thực hiện kết hợp các thư viện hàm và chức năng kiểm thử. Thực hiện mở cuộc kiểm thử của trang diễn đàn trước, sau đó thực hiện mở hàm trong thư viện ra bằng cách:

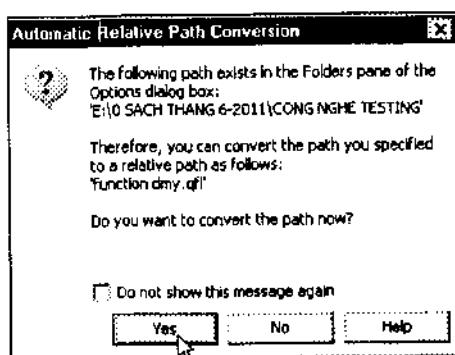
Trên thanh trình đơn View, chọn lệnh **Resources** để mở bảng **Resources**. Trong bảng này, hãy nhấp phải chuột vào thành phần **Associated Function Libraries > Associate Function Library**.



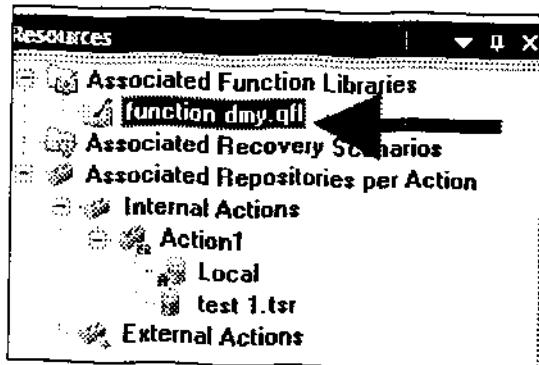
Hộp thoại **Open Function Library** xuất hiện, nhấp chọn đường dẫn đến thư mục lưu trữ hàm và nhấp chọn tên file lưu trữ có đuôi "qfl", sau đó nhấp nút **Open** để mở hàm.



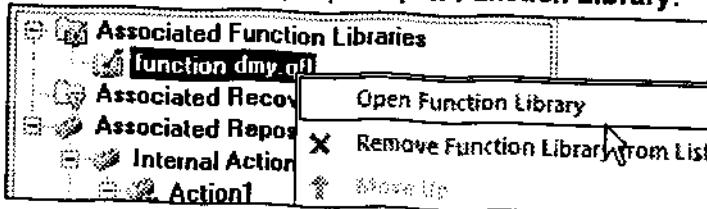
Hộp thoại **Automatic Relative Path Conversion** xuất hiện, xác định vị trí và chọn thư viện hàm. Nhấp **Yes** trong hộp thoại, đường dẫn tương đối tự động chuyển đổi. (Sử dụng một đường dẫn tương đối giữ đường dẫn hợp lệ khi bạn di chuyển các thư mục chứa các bài kiểm tra và các file khác từ một địa điểm khác, miễn là hệ thống phân cấp thư mục vẫn giữ nguyên).



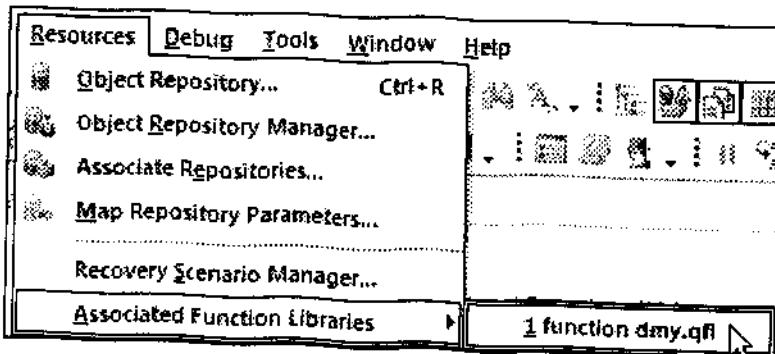
Thư viện hàm có liên quan với cuộc kiểm thử trang web được hiển thị trong cửa sổ **Resources** dưới dạng nút con của thư viện hàm **Associated Function Libraries** như hình sau:



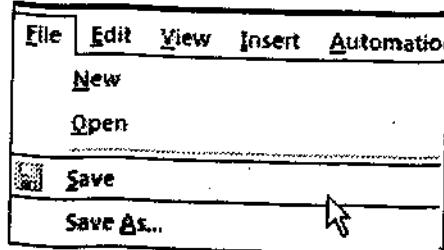
Xác minh các tập tin hàm có liên quan đến kiểm thử bằng cách mở nó ra: nhấp phải vào tập hàm và chọn lệnh **Open Function Library**.



Bước tiếp theo, kiểm tra tình trạng liên kết: nhấp chọn **Resources > Associated Function Libraries > function dmy.qfl** được liệt kê trong menu như hình sau:

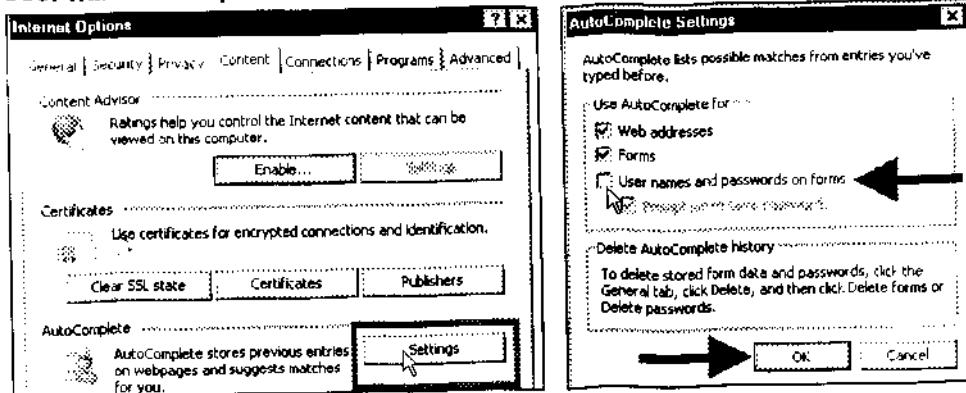


Bước cuối cùng, hãy thử kiểm tra lại action 1 kết hợp với hàm để tìm ra lỗi. Nếu hoàn thành xong, hãy nhấp chọn **File > Save** để lưu kiểm thử và liên kết hàm lại.

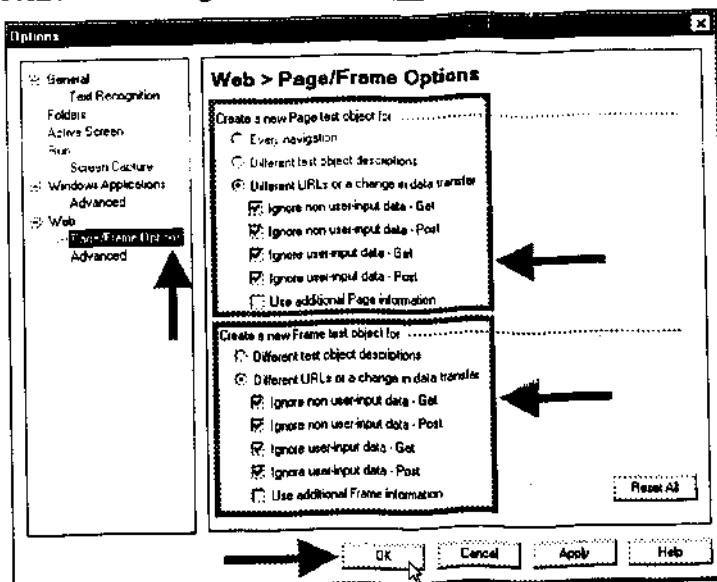
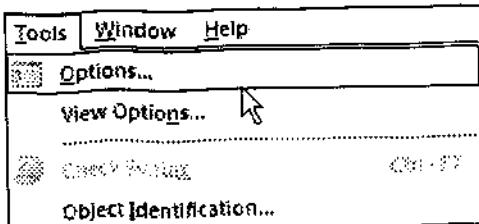


## TẠO MỘT THỬ NGHIỆM

Chuẩn bị để tạo ra một thử nghiệm, đảm bảo các ứng dụng được chạy trên chương trình. Bước đầu tiên, trong Internet Explorer, xóa lựa chọn cho các tên người dùng và mật khẩu trong hộp thoại AutoComplete bằng cách: nhấp chọn Tools > Internet Options > Tab Content sau đó chọn Settings trong mục AutoComplete. Hộp thoại AutoComplete Settings xuất hiện, bỏ chọn tại dòng User names and passwords on forms. Nhấp nút OK, xác nhận.



Quay lại giao diện chương trình QUICKTEST, nhấp chọn Tools > Options... mở hộp thoại Options. Nhấp chọn tùy chọn Web, và đánh dấu vào các thành phần trong phần Different URLs or a change...



Sau cùng nhấp nút **OK**, xác nhận thiết lập các tùy chọn.

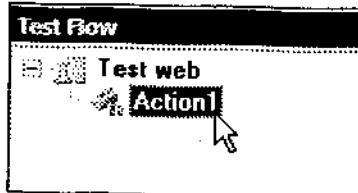
### THÊM HÀNH ĐỘNG ĐĂNG NHẬP

Thực hiện thêm 1 hành động tại action đăng nhập của trang web stkbbook.com, có nhiều cách để thêm một hành động. Khi thêm các hành động này sẽ giúp cho người kiểm thử phân tích tốt hơn các từ khóa.

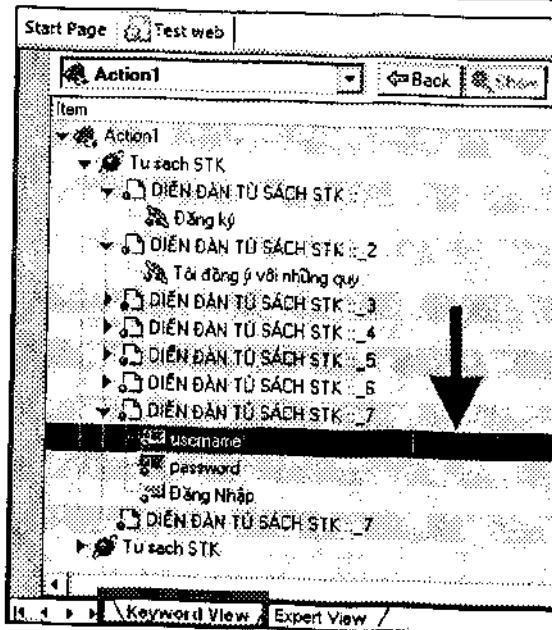
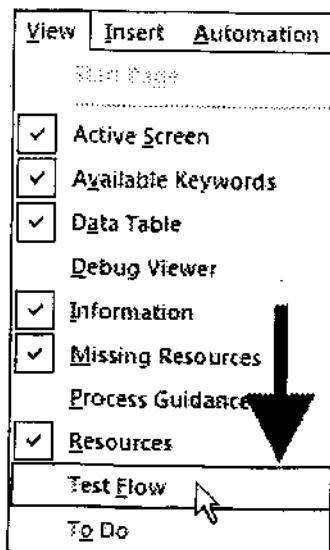
Hãy mở lại cuộc kiểm thử trước và tùy chọn phần action “đăng nhập” trong khung Keyword View.

Thực hiện nhấp chọn dòng kiểm tra tại vị trí kiểm tra của form đăng nhập trong action. Nếu không nhìn thấy các tab dòng kiểm thử, chọn **View > Test Flow**, hoặc nhấp vào nút hiển thị **Test Flow**.

Trong bảng **Test Flow** chọn action có dòng làm việc hiện hành.



Nhấp vào tab **Keyword View** để chọn thành phần được hiển thị.



Trong cửa sổ dòng kiểm thử, nhấp đúp vào các hành động đăng nhập. Kế tiếp, nhấp chọn Menu **Insert > New Step**, tạo ra một bước mới.

Một đối tượng mới xuất hiện trong action đăng nhập.

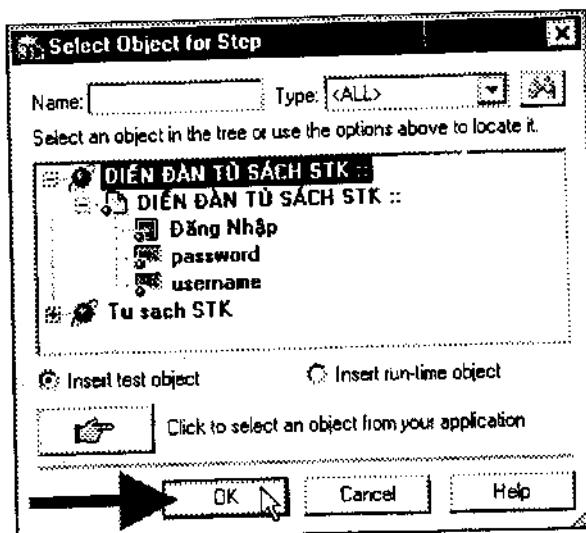
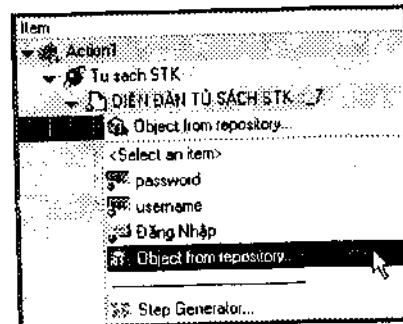
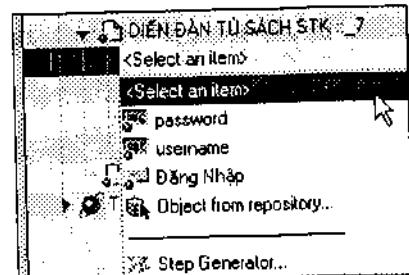
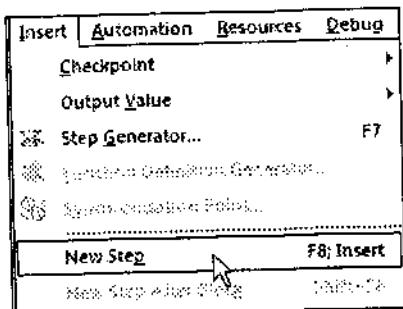
Thể hiện bằng một dòng hành động **<Select an item>**.

Nhấp mũi tên cuối dòng “**Select an Item**” để thực hiện tùy chọn thêm một đối tượng con vào action.

Danh sách các đối tượng được liệt kê, nhấp chọn lệnh **Object from repository...** để mở hộp thoại tùy chọn các đối tượng.

Hộp thoại **Select Object for Step** xuất hiện, nhấp thả các thành phần của diễn đàn và nhấp chọn **username** sau đó nhấp nút **OK** xác nhận.

Các đối tượng được bổ sung vào action, và hiển thị trên **Keyword View**.



Trong một phiên chạy, chương trình sử dụng các đối tượng để giúp xác định đối tượng thực tế mà nó cần phải thực hiện một action.

Trong bước này, chọn đối tượng kiểm tra là username để chỉnh sửa, và bổ sung vào các item, method mặc định, thiết lập, được bổ sung vào các thành phần hoạt động. Nhưng tài liệu không được thêm vào các thành phần tài liệu vì bước này vẫn còn thiếu một giá trị cần thiết.

Item	Operation	Value	Documentation
Action1			
DIỄN ĐÀN TỦ SÁCH STK::			
DIỄN ĐÀN TỦ SÁCH STK::	Set		<No documentation summary is available for the current step. >

Trong các thành phần giá trị của bước này, hãy chèn vào cột "Value" một giá trị là "quynhnga" như hình dưới:

Item	Operation	Value	Documentation
Action1			
DIỄN ĐÀN TỦ SÁCH STK::			
DIỄN ĐÀN TỦ SÁCH STK::	Set	"quynhnga"	Enter "tutorial" in the "username" edit box.

Hãy nhấp vào tab Expert View để xem cú pháp của các bước trong VBScript.

Step	Code
1	Browser("DIỄN ĐÀN TỦ SÁCH STK").Page("DIỄN ĐÀN TỦ SÁCH STK").WebEdit("username").Set "quynhnga"
2	Browser("Tu sach STK").Page("DIỄN ĐÀN TỦ SÁCH STK").WebEdit("username").Set "thuphuong"
3	Browser("Tu sach STK").Page("DIỄN ĐÀN TỦ SÁCH STK").WebEdit("password").SetSecure "4e0idbb79ae0d0"
4	Browser("Tu sach STK").Page("DIỄN ĐÀN TỦ SÁCH STK").WebButton("Đăng Nhập").Click
5	

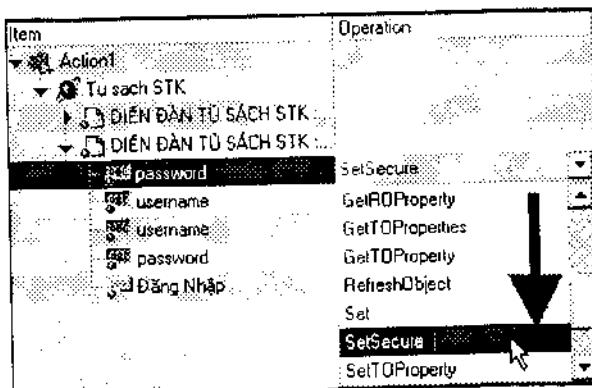
**Chú ý: bước này được thực hiện trên đối tượng kiểm tra Edit Web (box).**

#### Thêm bước tiếp theo

Trên thanh trình đơn, nhấp chọn **Insert > New Step**, tạo ra một bước mới. Nhấp chọn tiếp đối tượng Password trong danh sách.

Item	Operation	Value
Action1		
Tu sach STK		
DIỄN ĐÀN TỦ SÁCH STK		
DIỄN ĐÀN TỦ SÁCH STK	Set	
password	Set	
username	Set	"quynhnga"

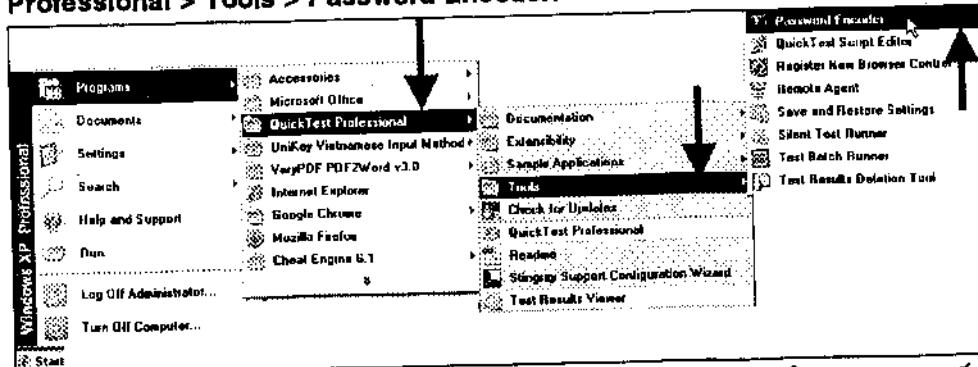
Chọn mật khẩu từ danh sách các đối tượng. Khi bước hoạt động Password được thêm vào, nhấp chọn vào cột Operation tại vị trí của dòng đó và chọn thuộc tính **SetSecure** (tham khảo hình trang bên).



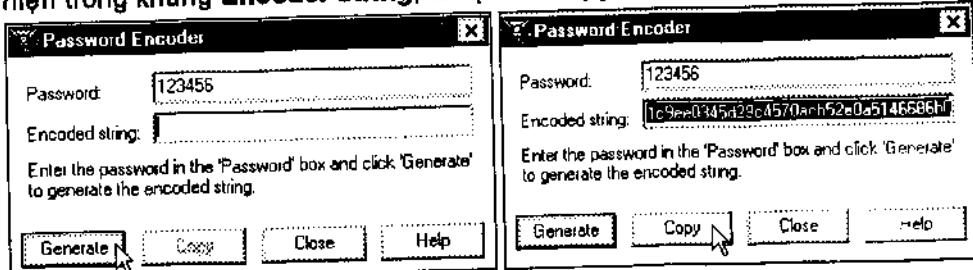
Để có thể mã hóa phần giá trị của hành động password, các bạn cần thực hiện phương pháp mã hóa mật khẩu trước.

### Tạo ra một mật khẩu mã hóa

Từ giao diện Windows chọn Start > Programs > QuickTest Professional > Tools > Password Encoder.



Hộp thoại Password Encoder xuất hiện, nhập mật khẩu vào và nhấn nút Generate để mã hóa password. Sau khi mã hóa, thành phần Code xuất hiện trong khung Encoder string, nhấp nút Copy để sao chép Code.



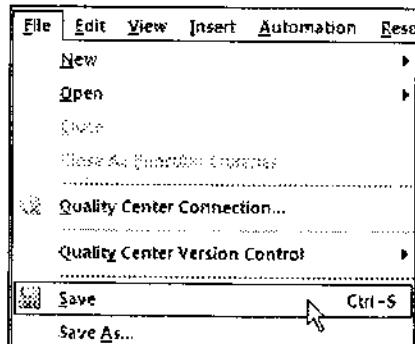
Quay lại bảng Keyword View, dán vào cột Value tại dòng hiện hành của bước thiết lập mật khẩu.

Sau khi thực hiện copy trong bảng giải mã xong, nhấp nút Close để đóng hộp thoại.

Action1	Operation	Value
Action1		
TuSach.STK		
DIỄN ĐÀN TỦ SÁCH STK		
password	SetSecure	4e1c9ee0345d29c4570acb52e0a5146686b0
username	Set	EncryptedText

Sau khi thực hiện xong, lưu trường hợp thiết lập trên lại bằng cách: nhấp chọn biểu tượng **Save** hoặc chọn Menu **File > Save (Ctrl + S)**.

Khi đã lưu các bước hoạt động mới tao, nhấp nút **Run** để kiểm tra, khi chạy thì user và password mới sẽ được nhập vào khung đăng nhập của trang web và cho ra thông báo không thể đăng nhập.



Tuy nhiên sẽ không phát sinh bất cứ lỗi nào khi thêm một hành động đăng nhập.

**Chú ý:** có thể tham khảo thêm về các thành phần khác trong cửa sổ Keyword View.

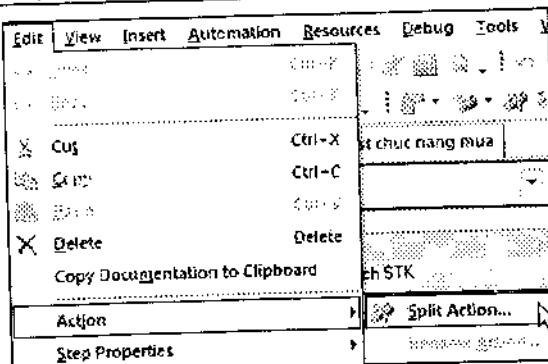
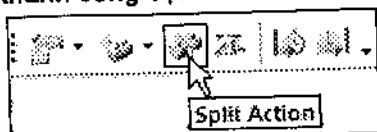
### ĐUA MỘT HÀNH ĐỘNG VÀO 2 ACTION

Chọn trang mà bạn muốn tạo hành động thứ hai để bắt đầu. Nếu cần, hãy thực hiện kiểm thử chức năng đặt mua trên trang web. Sau đó nhấp vào bảng **Test Flow** để chọn action muốn chia thành 2 action. Nhấp 2 lần vào action, để mở các hành động bên trong của action ra, các hành động này được xuất hiện dưới dạng cây trong bảng Keyword View.

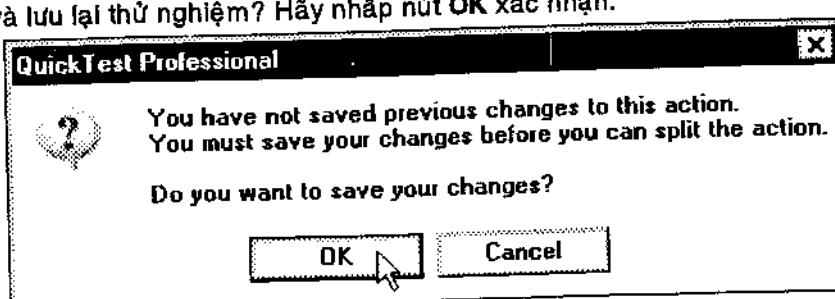
Trong bảng **Keyword View**, nhấp chọn thành phần đăng ký các thông tin mua trên trang web và sẽ tiến hành tách nó ra thành một action. Tham khảo hình minh họa trang bên.

Item	Operat.	Value	Documentation
DANG NHAP			
Tu sach STK			
Tu sach STK	Click		Click the "Chọn để mua" image.
Chon de mua			
Tu sach STK_2			
Windows Internet Explorer			
Tu sach STK_2			
Tu sach STK_3			
name	Set	"thu phuong"	Enter "thu phuong" in the "name" edit box.
address	Set	"q7"	Enter "q7" in the "address" edit box.
city	Set	"ho chi minh"	Enter "ho chi minh" in the "city" edit box.
country	Set	"viet nam"	Enter "viet nam" in the "country" edit box.
telephone	Set	"12345"	Enter "12345" in the "telephone" edit box.
Gửi email	Click		Click the "Gửi email" button.

Sau khi nhấp chọn nhóm action cần tách, nhấp chọn lệnh **Action > Split Action** trong menu **Edit** hoặc biểu tượng **Split Action** trên thanh công cụ.



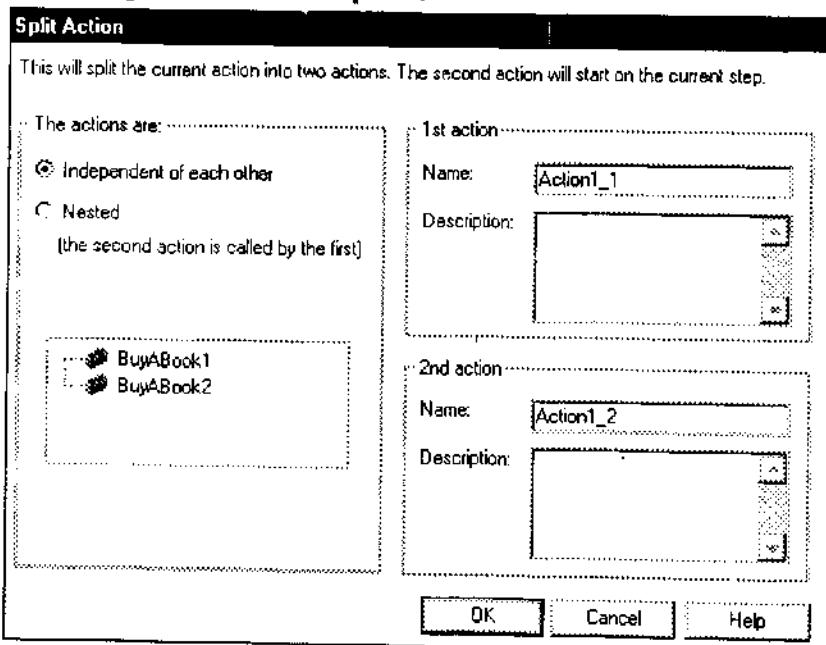
Một bảng thông báo xuất hiện, thông báo bạn có muốn tách hành động này và lưu lại thử nghiệm? Hãy nhấp nút OK xác nhận.



Hộp thoại **Split Action** xuất hiện, trong đó có các tùy chọn sau:

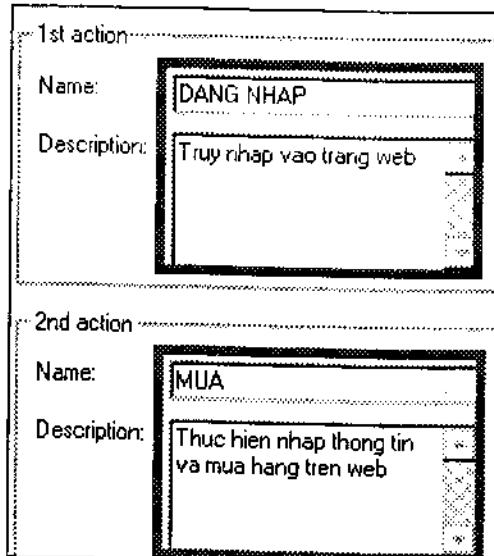
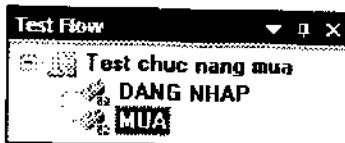
- **Independent of each other:** cho phép tách 1 hành động thành 2 hành động độc lập.
- **Nested (the second action is called by the first):** cho phép tách 1 hành động con từ một hành động cha, nó nằm trong cùng một bước gọi. Hai hành động cha/con được lồng vào nhau.

- Nếu muốn sửa đổi tên và mô tả của hai hành động hãy nhập nội dung vào khung Name và Description.



Trong khung action 1 hãy nhập vào tên là "ĐĂNG NHẬP" và mô tả action này. Tương tự, nhập tên "MUA" vào khung action 2 và mô tả như hình bên.

Sau khi tùy chọn và nhập thông tin xong, hãy nhấp nút OK ở cuối hộp thoại để xác nhận bước tách này. Trong cây ở bảng Test Flow xuất hiện 2 hành động như hình sau.



**Chú ý:** nếu một hành động được gọi lại nhiều hơn 2 lần trong một cuộc kiểm tra và phân chia các hành động thành 2 hành động độc lập, thì mỗi lần khởi động, hành động trong kiểm tra sẽ kéo theo hành động mới tách ra. Có thể tái sử dụng các hành động tách ra vào các cuộc kiểm thử khác, tuy nhiên có thể gây ra trường hợp các hành động không được dùng tới.

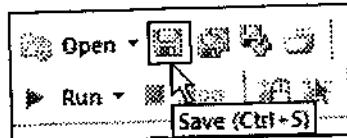
Các hành động bên trong các action được tách ra cũng hiển thị dưới dạng cây trong Keyword View.

DANG NHAP		Operat...	Value	Documentation
Item				
• DANG NHAP				
• Tu sach STK				
► Tu sach STK				
► Tu sach STK_2				
► Windows Internet Explorer				
• Tu sach STK_2				
• QTYSP_0	Set	"1"		Enter "1" in the "QTYSP_0" edit box.
• Cập nhật	Click			Click the "Cập nhật" button.
• Tiếp tục	Click			Click the "Tiếp tục" button.

Tương tự, trong action “MUA” cũng chứa các hành động trong nhóm đã chọn tách ra trước đó. Nó có thể hoạt động độc lập như các action có trong cuộc kiểm thử.

MUA		Operat...	Value	Documentation
Item				
• MUA				
• Tu sach STK				
► Tu sach STK_3				
• name	Set	"thu phuong"		Enter "thu phuong" in the "name" edit box.
• address	Set	"97"		Enter "97" in the "address" edit box.
• city	Set	"ho chi minh"		Enter "ho chi minh" in the "city" edit box.
• country	Set	"viet nam"		Enter "viet nam" in the "country" edit box.
• telephone	Set	"12345"		Enter "12345" in the "telephone" edit box.
• Gửi email	Click			Click the "Gửi email" button.

Sau cùng, nhấp nút Save trên thanh công cụ để lưu các bước thực hiện trên vào cuộc kiểm thử.



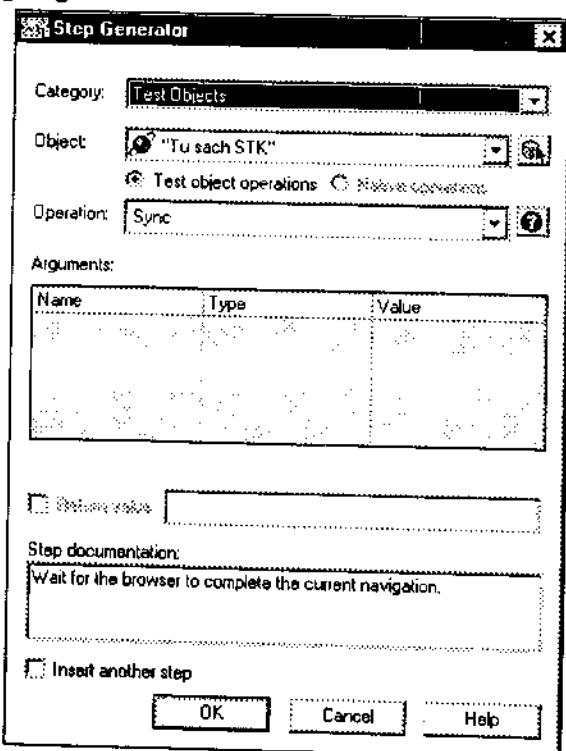
### THÊM MỘT BƯỚC HOẠT ĐỘNG TRONG ACTION

Đây là bước giúp thực hiện thêm thao tác, phương thức sử dụng đối tượng kiểm tra, phương thức tiện ích đối tượng hoặc gọi hàm Function.

Lưu ý: trong hộp thoại Step Generator các đối tượng trong khung Keyword View được hiển thị bên trong nó và các thông tin khác.

Nhấp chọn khung Expert View trước khi mở hộp thoại Step Generator, các thông tin cú pháp của đối tượng được chọn sẽ hiển thị khi chọn đối tượng đó trong danh sách.

Trong thư viện hàm, **Step Generator** có các tiêu đề khác nhau và các đối tượng tiện ích, hàm được xây dựng. Các thành phần hàm sẽ được biểu diễn trong khung **Arguments**.



Khi hộp thoại **Step Generator** mở ra, các đối tượng từ bước lựa chọn được hiển thị trong hộp đối tượng và phương pháp mặc định cho các đối tượng được thể hiện trong hộp hoạt động.

#### Xác định một bước mới

Khi xác định một bước tiến mới, trước tiên bạn chọn kiểu bước muốn thêm vào kiểm thử. Sau đó, có thể chọn đối tượng cụ thể và hoạt động cho các bước, hoặc chức năng mà ta muốn bước sử dụng.

Sau khi chọn các hoạt động cho bước, có thể xác định giá trị đối số có liên quan và vị trí cho giá trị trả về nếu áp dụng. Những giá trị này có tham số nếu cần thiết.

Cuối cùng, có thể xem các tài liệu hướng dẫn bước hoặc cú pháp. Tuyên bố và thêm bước tiến mới của bạn hoặc báo cáo thử nghiệm, thư viện chức năng của bạn.

**Lưu ý:** mặc dù hộp thoại **Step Generator** hiển thị thông tin về các bước đang được chọn, lựa chọn được thực hiện trong **Step Generator** sẽ thêm một bước mới vào thử nghiệm, không sửa đổi các bước hiện có.

### **Chọn bước thêm**

Trong khung danh sách **Category**, có thể chọn một trong các tùy chọn như sau:

**Test Objects:** cho phép chọn một đối tượng kiểm tra và hoạt động cho bước (xét nghiệm). Để biết thêm thông tin, xem chỉ định một đối tượng thử nghiệm và hoạt động cho bước.

**Utility Objects:** cho phép chọn một đối tượng hữu ích và hoạt động cho bước. Để biết thêm thông tin, xem chỉ định một đối tượng hữu ích và hoạt động cho bước.

**Functions:** cho phép chọn một chức năng cho bước từ các chức năng thư viện có sẵn (kiểm tra), VBScript chức năng, và các chức năng kịch bản nội bộ. Để biết thêm thông tin, xem chỉ định một chức năng cho bước.

### **Xác định giá trị đối số**

Sau khi chọn đối tượng và hoạt động (phương thức, thuộc tính, hoặc hàm) cho bước, có thể chỉ định các giá trị đối số có liên quan. Những giá trị này có thể được gán tham số nếu cần thiết. Nếu hoạt động được chọn tham số, khu vực lập luận hiển thị tên và kiểu của mỗi đối số.

Trong cột giá trị, có thể định nghĩa các giá trị cho các đối số như sau:

**Mandatory argument:** nếu tên của các đối số được theo sau bởi một dấu hoa thị màu đỏ (\*), phải chỉ định một giá trị cho đối số. Bạn không thể chèn các bước hoặc xem các tài liệu hướng dẫn bước nếu các giá trị không được định nghĩa cho tất cả các đối số bắt buộc.

**Optional argument:** nếu tên của các đối số không theo sau bởi một dấu hoa thị màu đỏ (\*), có thể chỉ định một giá trị cho đối số hoặc để trống cell. Nếu không chỉ định một giá trị, QuickTest sử dụng giá trị mặc định cho đối số (có thể xem các giá trị mặc định bằng cách di chuyển con trỏ qua cell).

**Required argument:** chỉ định một giá trị cho một đối số tùy chọn, sau đó phải xác định các giá trị cho bất kỳ đối số tùy chọn được liệt kê trước khi lập luận. Nếu không chỉ định các giá trị này, chương trình QuickTest sử dụng các giá trị mặc định cho tất cả các đối số cần thiết. Có thể nhìn thấy giá trị mặc định cho mỗi đối số trong một tooltip, bằng cách di chuyển con trỏ qua cột Value.

**Parameterized argument:** có thể sử dụng một tham số cho bất kỳ giá trị đối số bằng cách nhấn vào nút tham số. Để biết thêm thông tin, xem cấu hình một giá trị lựa chọn.

**Predefined constants:** nếu có một danh sách được xác định trước các giá trị, QuickTest cung cấp một danh sách thả xuống của giá trị có thể.

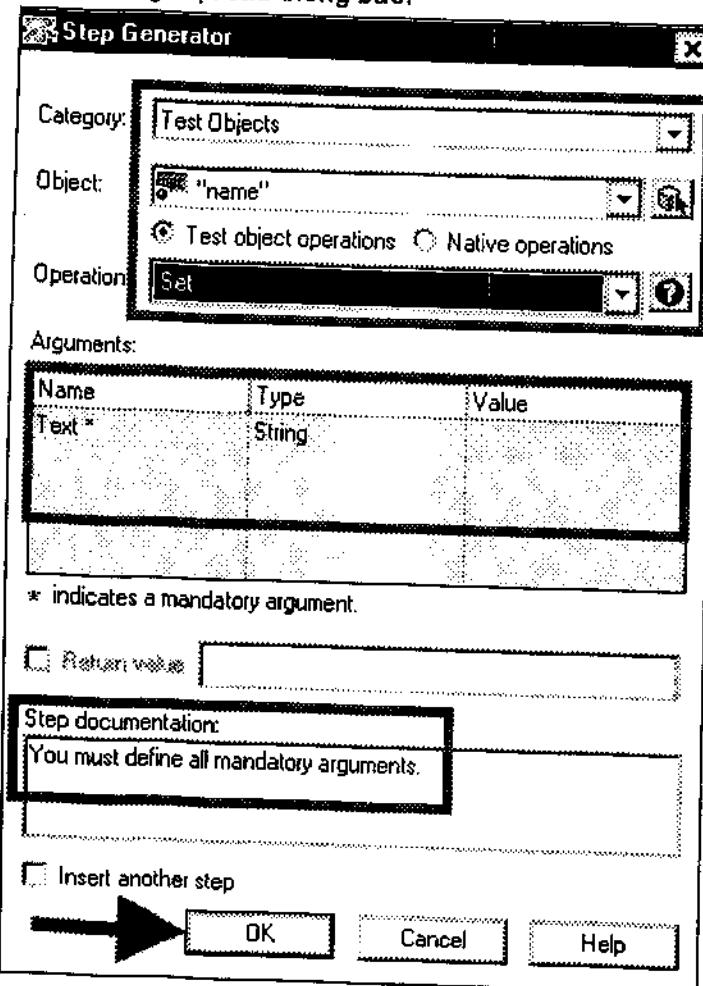
Nếu một danh sách các giá trị được cung cấp, có thể loại một giá trị trong ô này.

### **Chỉ định giá trị trả về**

Nếu hoạt động được lựa chọn trả về một giá trị, bạn có thể chỉ định rằng muốn để lưu trữ các giá trị phải chọn hộp kiểm tra giá trị trả về. Khi hộp kiểm này được chọn, một biến mặc định hiển thị như là vị trí giá trị trả về. Có thể cung cấp một định nghĩa khác nhau thay đổi bằng cách chỉnh sửa giá trị và có thể chọn một vị trí lưu trữ khác nhau cho các giá trị trả về bằng cách nhấp vào các giá trị hiển thị và sau đó nhấp nút lưu trữ đầu ra.

### **Xem tài liệu Keyword View**

Nếu mở Step Generator từ Keyword View, hộp Step Documentation ở dưới cùng của hộp thoại Step Generator có thể hiển thị thông tin tóm tắt về các bước hiện tại trong một câu thông báo.



Nếu chọn một trong hai đối tượng kiểm tra, chủng loại đối tượng tiện ích và xác định tất cả các giá trị bắt buộc và cần thiết cho hoạt động hiện tại. Hộp Step Documentation mô tả các hoạt động thực hiện từng bước. Khi các bước được đưa vào thử nghiệm, mô tả này được hiển thị trong cột Documentation trong Keyword View.

Nếu tất cả các giá trị đối số bắt buộc và yêu cầu không được xác định cho hoạt động, hộp Step Documentation hiển thị một thông điệp cảnh báo.

Lưu ý: nếu chọn loại chức năng, Step Documentation có sẵn cho người sử dụng xác định chức năng, thông tin này cung cấp khi định nghĩa chúng. Để biết thêm thông tin, hãy xem phần hướng dẫn Function.

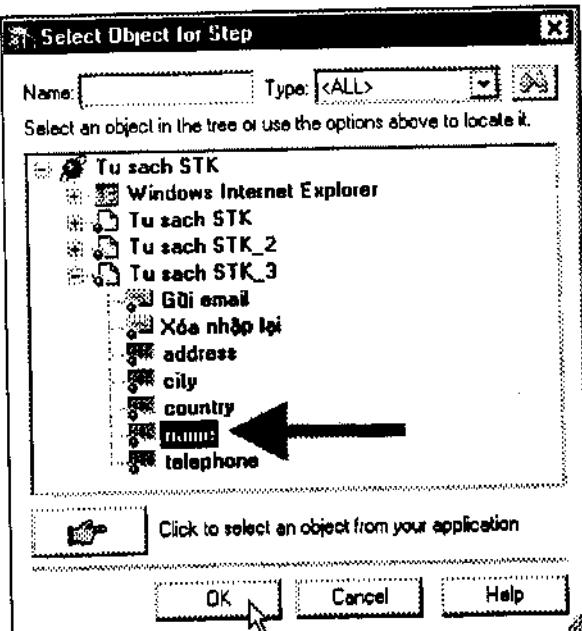
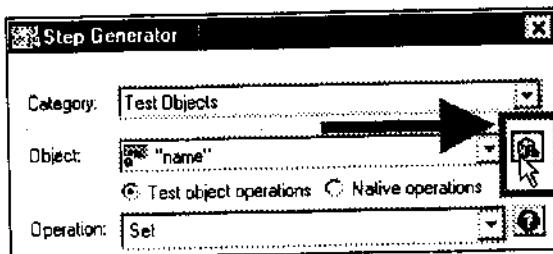
### Mở Step Generator trong khung Expert View

Nếu mở hộp thoại Step Generator từ Expert View, các hộp thoại Step Generator hiển thị các báo cáo được xác định cho bước. Nhấp chọn nút lệnh cuối dòng Object để chọn đối tượng cần thêm.

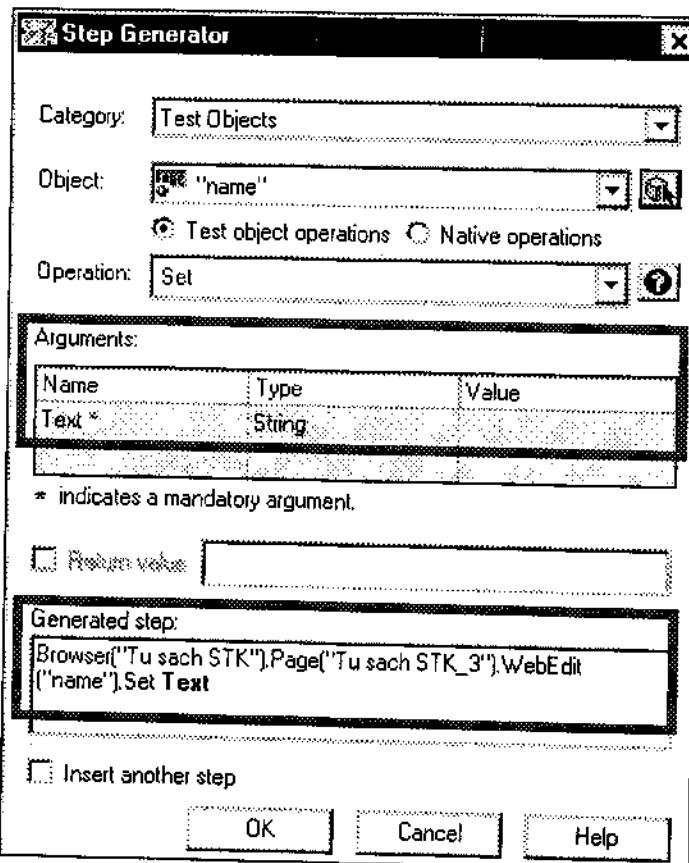
Trong hộp thoại Select Object for Step, nhấp chọn bước hoạt động cần chèn trong cây, sau đó nhấp nút OK xác nhận.

Quay lại hộp thoại Step Generator, các thông tin cấu trúc của chức năng hoạt động vừa chọn sẽ xuất hiện trong khung Generated Step.

Khi đã thiết lập tất cả các thông tin đối số cho hành động sẽ chèn thêm, hãy nhấp nút OK để xác nhận.



Nếu tất cả các giá trị đối số bắt buộc và yêu cầu không được xác định cho hoạt động, tên của các đối số không xác định được nhấn mạnh trong văn bản in đậm. Nếu cố gắng để chèn bước, một thông báo lỗi sẽ hiển thị.



### Xem bước tạo ra một thư viện chức năng.

Nếu bạn mở Step Generator một thư viện chức năng sẽ hiển thị các báo cáo được xác định cho bước.

Nếu tất cả các giá trị đối số bắt buộc và yêu cầu không được xác định cho bảng tuyên bố, tên của các đối số không xác định được nhấn mạnh trong văn bản in đậm. Nếu cố gắng để chèn bước, một thông báo lỗi sẽ hiển thị.

### Chèn bước

Sau khi xác định tất cả các giá trị đối số bắt buộc đối với các hoạt động hiện tại ta có các tùy chọn sau:

- Để chèn các bước hiện tại và đóng Step Generator, đảm bảo chèn một bước kiểm tra được xóa. Khi nhấp nút OK, bước hoạt động được thêm vào cuộc kiểm tra và đóng hộp thoại Step Generator.
- Để chèn bước hiện tại và tiếp tục bổ sung thêm các bước tại cùng một vị trí, chọn Insert một bước kiểm tra. Khi nhấp vào Insert, bước hiện tại được thêm vào để kiểm tra và hộp Step Generator vẫn mở, cho phép xác định một bước.

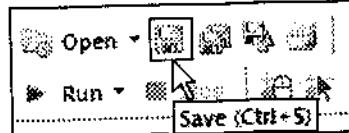
Khi chèn một bước tiến mới trong hộp thoại Step Generator, bước hiện tại được thêm vào để kiểm thử.

Item	Operation	Value	Documentation
✓ HUA			
✓ Tu sach STK			
✓ Tu sach STK_3			
name	Set	"thu phuong"	Enter "thu phuong" in the "name" edit box.
Object	SetActive		Access the native setActive method/property of the "name" edit box!
Object	accept		Access the native accept method/property of the "name" edit box. Store the result in the variable.
address	Set	"q7"	Enter "q7" in the "address" edit box.
city	Set	"ho chi minh"	Enter "ho chi minh" in the "city" edit box.
country	Set	"viet nam"	Enter "viet nam" in the "country" edit box.
telephone	Set	"12345"	Enter "12345" in the "telephone" edit box.
Gửi email	Click		Click the "Gửi email" button.

1	Browser("Tu sach STK").Page("Tu sach STK_3").WebEdit("name").Set "thu phuong"
2	Browser("Tu sach STK").Page("Tu sach STK_3").WebEdit("name").Object.setActive
3	var accept = Browser("Tu sach STK").Page("Tu sach STK_3").WebEdit("name").Object.accept
4	Browser("Tu sach STK").Page("Tu sach STK_3").WebEdit("address").Set "q7"
5	Browser("Tu sach STK").Page("Tu sach STK_3").WebEdit("city").Set "ho chi minh"
6	Browser("Tu sach STK").Page("Tu sach STK_3").WebEdit("country").Set "vietnam"
7	Browser("Tu sach STK").Page("Tu sach STK_3").WebEdit("telephone").Set "12345"
8	Browser("Tu sach STK").Page("Tu sach STK_3").WebButton("Gửi email").Click

Sau cùng, nhấp nút Save trên thanh công cụ để lưu các bước thực hiện trên vào cuộc kiểm thử.



## TẠO CÁC ĐẤU KIỂM SOÁT VÀ SỬ DỤNG HÀM

Chúng ta đã tiến hành kiểm tra một số chức năng trên trang web và chúng chạy tốt. Trạm kiểm soát checkpoint đã xác nhận các thông tin dự kiến sẽ được hiển thị trong ứng dụng kiểm thử.

Phần trình bày sau là hướng dẫn thêm các điểm kiểm soát và sử dụng chức năng hàm để việc kiểm thử thêm hiệu lực đối với một số đối tượng trong các trang web.

Ta có các thành phần kiểm soát như sau:

- Kiểm tra các đối tượng (Object).
- Kiểm tra trang (Page).
- Kiểm tra bảng (Table).
- Kiểm tra văn bản (Text).
- Quản lý các điểm kiểm tra trong Repository đối tượng.
- Chạy và phân tích một thử nghiệm với các điểm kiểm tra.
- Thực hiện một kiểm tra sử dụng một hàm (Function).

Thêm một trạm kiểm soát tiêu chuẩn trong một trang của web:

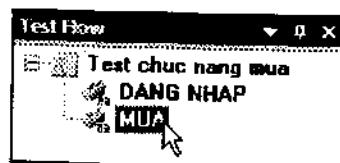
Trạm kiểm soát này đã thẩm định giá trị trong hộp có chứa tên của khách hàng đăng ký mua.

Để chèn các trạm kiểm soát, hãy khởi động 1 trang web ứng dụng trước. Bắt đầu khởi động chương trình QuickTest và mở kiểm tra test 1 trong thư mục lưu trữ.

Lưu thêm một bảng kiểm thử trước khi thực hiện đánh dấu kiểm soát. Trên thanh trình đơn, nhấp chọn File > Save As.

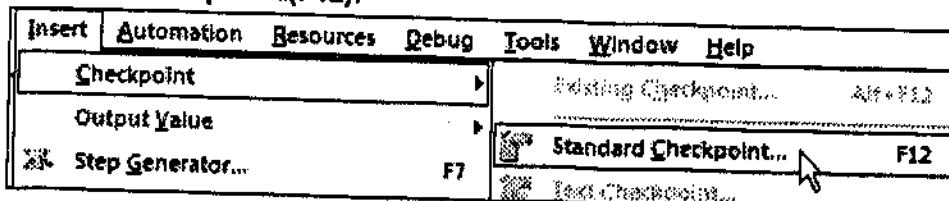
Hiển thị các bước hành động muốn thêm vào một trạm kiểm soát tiêu chuẩn: mở bảng Test Flow để kiểm tra các thành phần action trong kiểm thử. Nếu không nhìn thấy bảng Test Flow, nhấp chọn Menu View > Test Flow, để mở lại bảng này.

Nhấp đúp vào action **MUA** trong bảng Test Flow để mở action trong bảng Keyword View. Nhấp chọn một action bên trong hành động mua này, ví dụ như hành động name.



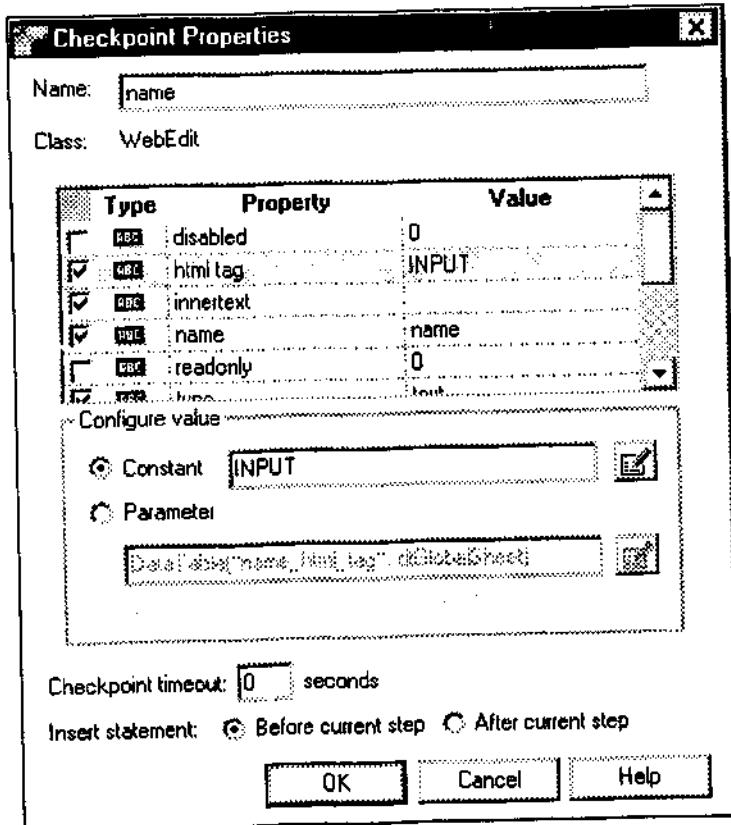
Item	Operation	Value	Documentation
• MUA			
• Tu sach STK			
• Tu sach STK_3			
• name	Set	"thu phuong"	Enter "thu phuong" in the "name" edit box.
• address	Set	"q7"	Enter "q7" in the "address" edit box.
• city	Set	"ho chi minh"	Enter "ho chi minh" in the "city" edit box.
• country	Set	"viet nam"	Enter "viet nam" in the "country" edit box.
• telephone	Set	"12345"	Enter "12345" in the "telephone" edit box.
• Gửi email	Click		Click the "Gửi email" button.

Kế tiếp, trên thanh trình đơn nhấp chọn menu Insert > Checkpoint > Standard Checkpoint... (F12).



Hộp thoại **Checkpoint Properties** xuất hiện, trong đó có thể chỉ định các thuộc tính của đối tượng để kiểm tra và chỉnh sửa các giá trị của nó.

Các yếu tố thuộc tính phụ thuộc vào loại đối tượng đang kiểm tra. Chúng ta có thể tìm hiểu các tính năng như hướng dẫn trong trang bên.



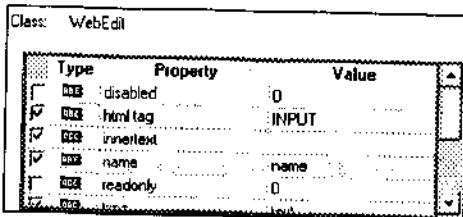
Trong hộp thoại sẽ hiển thị thuộc tính của đối tượng:

THÔNG TIN	MÔ TẢ
NAME	<p>Tên của trạm kiểm soát. Theo mặc định, tên trạm kiểm soát cũng giống như tên của đối tượng kiểm tra trạm kiểm soát đã được tạo ra.</p> <p>Bạn có thể chỉ định một tên khác nhau cho các trạm kiểm soát hoặc chấp nhận tên mặc định.</p> <p>Nếu đổi tên các trạm kiểm soát, hãy chắc chắn tên này:</p> <ul style="list-style-type: none"> <li>• Là duy nhất.</li> <li>• Không bắt đầu hoặc kết thúc với một không gian.</li> <li>• Không chứa "(dấu ngoặc kép tăng gấp đôi).</li> <li>• Không chứa kết hợp ký tự sau đây: : = @ #</li> </ul>

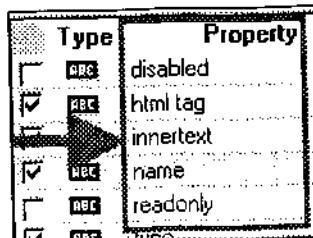
CLASS	Các loại đối tượng.
Nút FIND IN REPOSITORY	<p>Hiển thị các trạm kiểm soát trong kho đối tượng của nó, tùy chọn này có sẵn khi chỉnh sửa một điểm kiểm soát hiện có. Nó không phải là có sẵn khi tạo một trạm kiểm soát mới. Khi có sẵn, nó nằm ở bên phải của hộp Name.</p> <p>Khi làm việc với các thành phần, tùy chọn này chỉ có trong chế độ nâng cao.</p>

### Lựa chọn đối tượng để kiểm tra

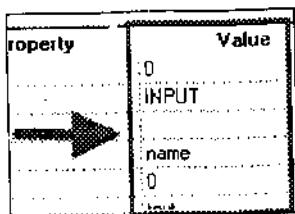
Các thuộc tính cho các đối tượng được liệt kê trong cửa sổ thuộc tính của hộp thoại. Bảng này bao gồm tên các loại thuộc tính, giá trị và khung đánh dấu như hình sau:



PHẦN TỬ TRONG BẢNG	MÔ TẢ														
<b>CHECK BOX</b> <table border="1"> <thead> <tr> <th>Type</th> <th>Property</th> </tr> </thead> <tbody> <tr> <td>checkbox</td> <td>disabled</td> </tr> <tr> <td>checkbox</td> <td>html tag</td> </tr> <tr> <td>checkbox</td> <td>innerText</td> </tr> <tr> <td>checkbox</td> <td>name</td> </tr> <tr> <td>checkbox</td> <td>readonly</td> </tr> <tr> <td>checkbox</td> <td>type</td> </tr> </tbody> </table>	Type	Property	checkbox	disabled	checkbox	html tag	checkbox	innerText	checkbox	name	checkbox	readonly	checkbox	type	<p>Đối với mỗi lớp đối tượng, chương trình sẽ đánh dấu kiểm tra thuộc tính đối tượng mặc định. Người dùng có thể chấp nhận kiểm tra mặc định hoặc thay đổi chúng cho phù hợp.</p> <ul style="list-style-type: none"> <li>Để kiểm tra một đối tượng chọn hộp kiểm tra tương ứng.</li> <li>Để loại trừ một kiểm tra thuộc tính, nhấp chuột bỏ chọn hộp kiểm tra tương ứng.</li> </ul>
Type	Property														
checkbox	disabled														
checkbox	html tag														
checkbox	innerText														
checkbox	name														
checkbox	readonly														
checkbox	type														
<b>TYPE</b> <table border="1"> <thead> <tr> <th>Type</th> <th>Property</th> </tr> </thead> <tbody> <tr> <td>checkbox</td> <td>disabled</td> </tr> <tr> <td>checkbox</td> <td>html tag</td> </tr> <tr> <td>checkbox</td> <td>innerText</td> </tr> <tr> <td>checkbox</td> <td>name</td> </tr> <tr> <td>checkbox</td> <td>readonly</td> </tr> <tr> <td>checkbox</td> <td>type</td> </tr> </tbody> </table>	Type	Property	checkbox	disabled	checkbox	html tag	checkbox	innerText	checkbox	name	checkbox	readonly	checkbox	type	<ul style="list-style-type: none"> <li>Biểu tượng này chỉ ra rằng giá trị của thuộc tính là một hằng số.</li> <li>Biểu tượng này chỉ ra rằng giá trị của thuộc tính hiện đang là một thử nghiệm hoặc tham số hành động.</li> <li>Biểu tượng này chỉ ra rằng giá trị của thuộc tính là một bảng dữ liệu tham số.</li> <li>Biểu tượng này chỉ ra giá trị của thuộc tính hiện đang là một tham số biến môi trường.</li> <li>Biểu tượng này chỉ ra giá trị của thuộc tính là một tham số số ngẫu nhiên.</li> </ul>
Type	Property														
checkbox	disabled														
checkbox	html tag														
checkbox	innerText														
checkbox	name														
checkbox	readonly														
checkbox	type														

**PROPERTY**

Tên của thuộc tính.

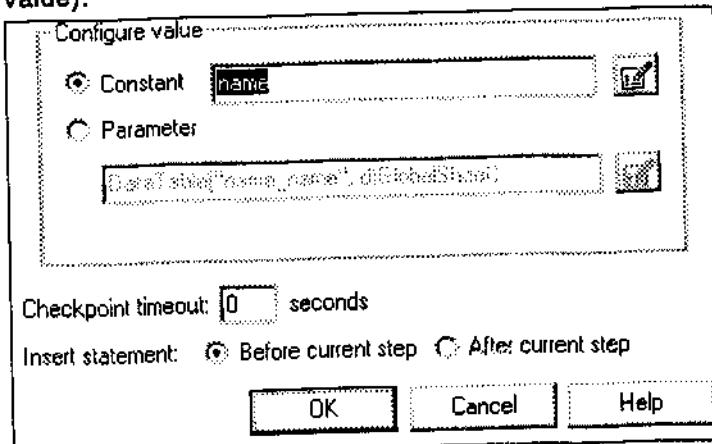
Hiển thị tất cả các tên của thuộc tính nếu có.  
Cho phép chọn và tùy chỉnh giá trị của nó.**VALUE**

Giá trị của thuộc tính.

Thực hiện thay đổi giá trị này khi cần thiết.

**Chỉnh sửa giá trị dự kiến của một thuộc tính đối tượng:**

Trong giá trị cấu hình, chúng ta có thể xác định giá trị mong muốn của thuộc tính để kiểm tra như là một hằng số hoặc tham số. Đối với thông tin về thay đổi giá trị thuộc tính, hãy tham khảo thêm phần thiết lập thuộc tính (configure value).

**Thiết lập tùy chọn Checkpoint Timeout**

Khi làm việc với các thành phần, tùy chọn này chỉ có trong chế độ nâng cao. Trạm kiểm soát thời gian chờ, chỉ định khoảng thời gian (tính bằng giây) trong thời gian đó chương trình sẽ thực hiện các trạm kiểm soát thành công. Chương trình tiếp tục thực hiện các trạm kiểm soát cho đến xong hoặc cho đến khi thời gian chờ xảy ra. Nếu các trạm kiểm soát không hoàn thành trước khi thời gian chờ xảy ra, trạm kiểm soát sẽ thất bại.

Nếu chỉ định một thời gian chờ khác lớn hơn 0, và trạm kiểm soát thất bại, cửa sổ kết quả kiểm thử sẽ hiển thị thông tin về thời gian chờ.

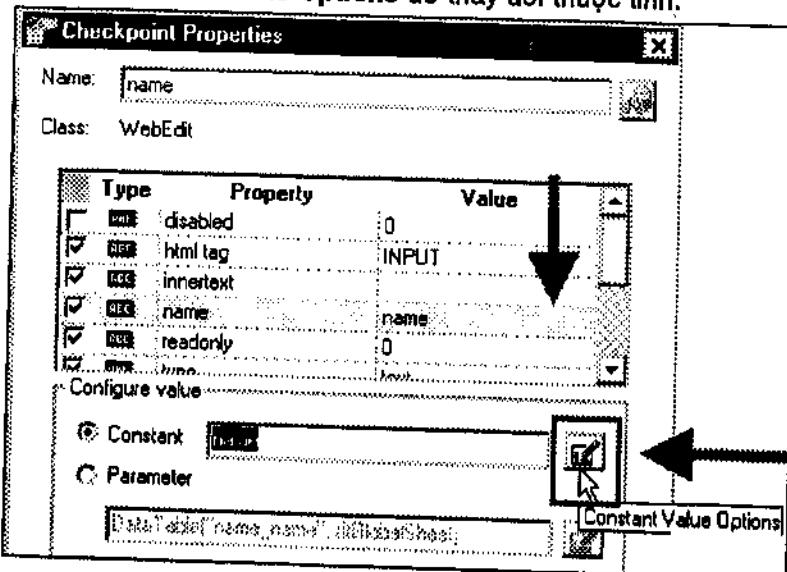
### **Chèn dấu check point vào kiểm thử:**

Tùy chọn **Insert statement** là các quy định cụ thể khi thực hiện các trạm kiểm soát trong các kiểm thử. Ta có 3 tùy chọn như sau:

- Chọn **Before Current Step**: nếu muốn kiểm tra giá trị của thuộc tính đối tượng trước khi bước được đánh dấu được thực hiện.
- Chọn **After Current Step**: nếu muốn kiểm tra giá trị của thuộc tính sau khi bước được đánh dấu được thực hiện.

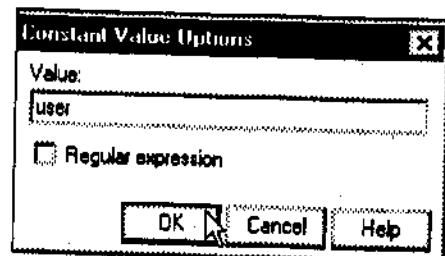
**Chú ý:** các tùy chọn **Insert statement** là không có sẵn khi thêm một trạm kiểm soát trong quá trình ghi hoặc khi sửa đổi một điểm kiểm soát đối tượng hiện có. Nó có sẵn khi thêm một trạm kiểm soát mới, đến một kiểm thử hiện có khi chỉnh sửa. Khi nắm rõ các khái niệm cơ bản trên, các bạn hãy thực hiện thêm một kiểm tra vào đối tượng name trong kiểm thử.

Trong bảng **Checkpoint Properties**, nhấp chọn đối tượng name và nhấp vào nút **Constant Value Options** để thay đổi thuộc tính.



Hộp thoại **Constant Value Options** xuất hiện, trong khung **Value** nhập giá trị mong muốn sau đó nhấp nút **OK** xác nhận.

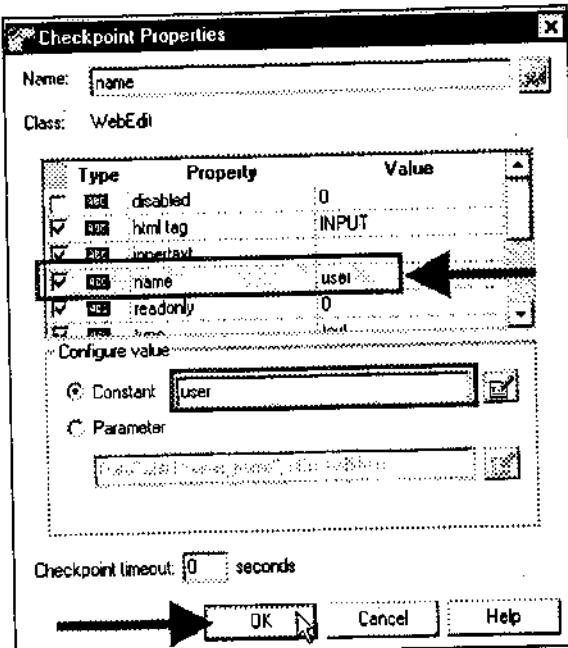
Tùy chọn **Regular expression**: thiết lập giá trị được định nghĩa biểu hiện thường xuyên.



Quay lại hộp thoại **Checkpoint Properties**, giá trị sẽ được cập nhật vào cột **Value**, sau đó nhấp nút **OK** để xác nhận.

Trong khung **Keyword View**, bước mới thêm ở vị trí bên trên của hành động đã chọn.

Sau khi thực hiện xong, hãy thực hiện lưu kiểm thử lại. Nhấp nút **Save** trên thanh công cụ để lưu các bước thực hiện kiểm thử.



Item	Operation	Value	Documentation
MUA			
Tu sach STK			
Tu sach STK_3			
<input checked="" type="checkbox"/> name	Check	CheckPoint("name")	Check whether the "name" edit box has the
<input type="checkbox"/> name	Set	"thu phuong"	Enter "thu phuong" in the "name" edit box.

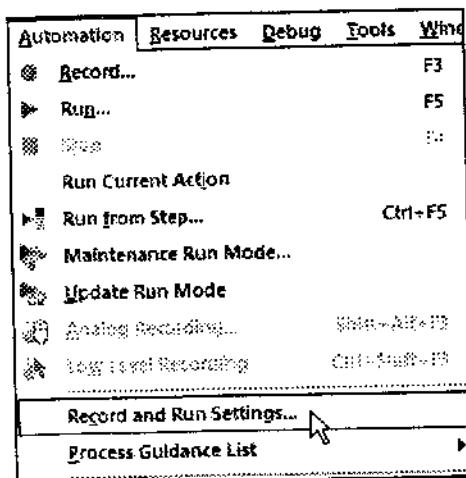
Bạn có thể chèn các trạm kiểm soát theo cách mô tả ở trên. Một số các loại đặc biệt của các trạm kiểm soát được giải thích trong các phần sau.

### Kiểm tra trang (Page)

Để kiểm tra theo dạng này, đầu tiên hãy nhấp chọn lệnh **Record and run Settings**, để cấu hình lại đường dẫn đến trang web cần kiểm tra.

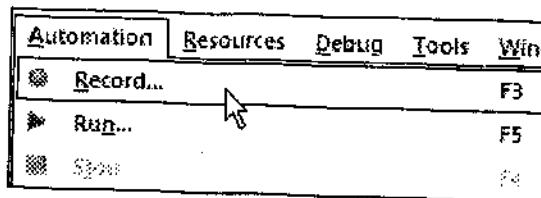
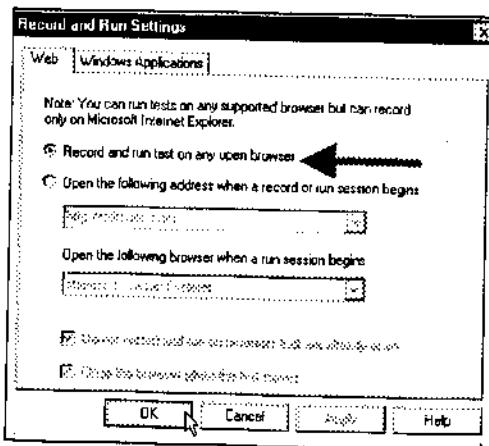
Hộp thoại **Record and run Settings** xuất hiện, nhấp chọn chế độ **Record and run test on any open browser**, sau đó nhấp nút **OK** xác nhận.

Không chọn lại đường dẫn nữa, mà có thể mở với IE.

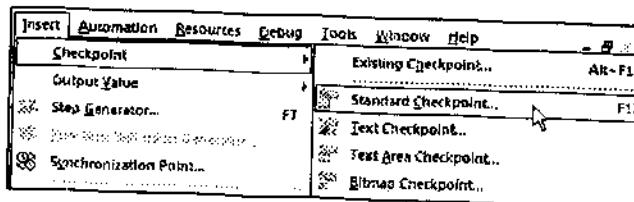


Chương trình có thể tự động ghi lại trên trình duyệt Microsoft Internet Explorer hay chạy trên bất kỳ trình duyệt Web mở được hỗ trợ. Như vậy, hãy khởi động trang web [stlbook.com](http://stlbook.com) trước khi chọn chế độ như trên.

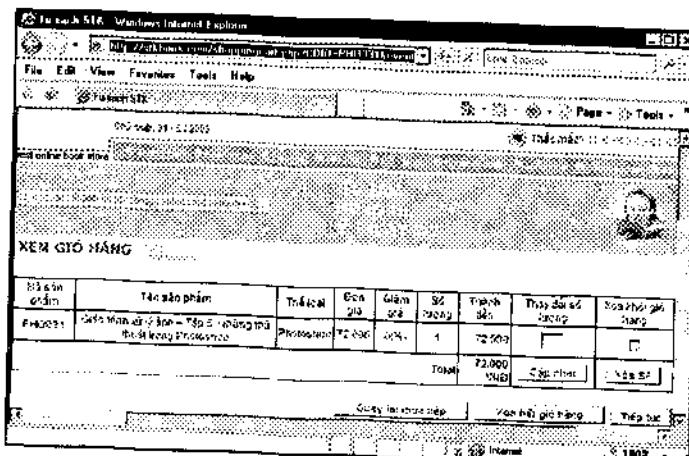
Kế tiếp, trên thanh trình đơn nhấp chọn Menu **Automation** > chọn **Record (F3)**, bắt đầu ghi lại các tiến trình và các bước thao tác trên trang web.



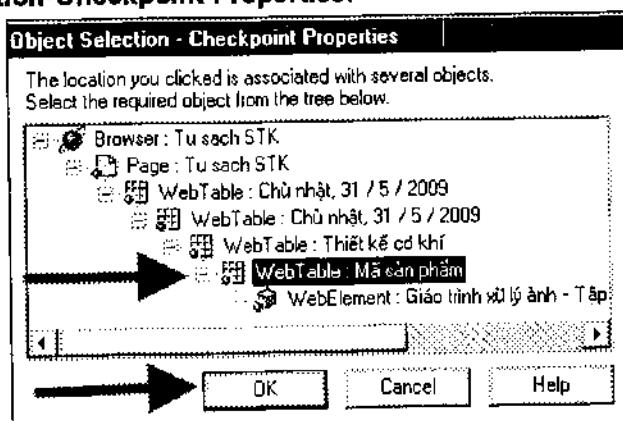
Trên thanh trình đơn, nhấp chọn menu **Insert > Checkpoint > Standard Checkpoint**. Chương trình sẽ tự động dẹp bỏ các cửa sổ chỉ để lại trình duyệt web đã lựa chọn.



Trên trang web, nhấp chọn khu vực giờ hàng để thực hiện ghi lại thao tác đặt hàng.



Các đối tượng được chọn trong web sẽ tự động cập nhật vào hộp thoại Object Selection-Checkpoint Properties.



**Hộp thoại Table Checkpoint Properties** xuất hiện, các thành phần của bước kiểm thử đã chọn sẽ thể hiện đầy đủ trong bảng.

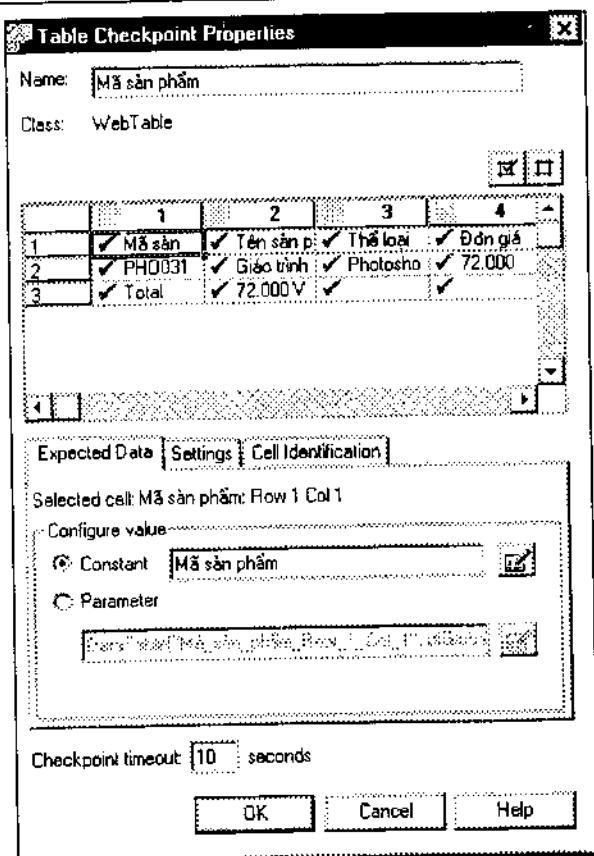
Phần này mô tả các cài đặt chung và các tùy chọn được hiển thị trong hộp thoại Checkpoint Properties.

Hầu hết các tùy chọn được mô tả trong phần này là có sẵn và được đánh dấu check theo từng phần.

#### Mô tả Thông tin:

Các phần trên của hộp thoại Table Properties Checkpoint chứa các tùy chọn sau đây:

Thành phần thứ nhất là khung name chứa tên của đối tượng và lớp (Class) của nó là bảng biểu.



Name:	<input type="text" value="Mã sản phẩm_4"/>
Class:	<input type="text" value="WebTable"/>

THÔNG TIN	MÔ TẢ
NAME	<p>Tên của bước kiểm thử, theo mặc định, tên bước kiểm thử cũng giống như tên của đối tượng kiểm tra từ trạm kiểm soát đã được tạo ra.</p> <p>Bạn có thể chỉ định một tên khác nhau cho các trạm kiểm soát hoặc chấp nhận tên mặc định.</p> <p>Nếu đổi tên các trạm kiểm soát, hãy chắc chắn rằng tên này:</p> <ul style="list-style-type: none"> <li>Là duy nhất.</li> <li>Không bắt đầu hoặc kết thúc với một không gian.</li> <li>Không chứa "(dấu ngoặc kép tăng gấp đôi).</li> <li>Không chứa kết hợp ký tự sau đây: : = @ #</li> </ul>
CLASS	Các loại đối tượng.

#### Xác định các đối tượng kiểm tra:

Trong khu vực lưới khung của hộp thoại Table Properties Checkpoint, có chứa các thành phần trong bảng. Nó sẽ chia các đối tượng của hành động thành các cột như hình.

	1	2	3	4
1	✓ Mã sản phẩm	✓ Tên sản phẩm	✓ Thể loại	✓ Đơn giá
2	✓ PH0031	✓ Giáo trình	✓ Photosho	✓ 72.000
3	✓ Total	✓ 72.000 V	✓	✓

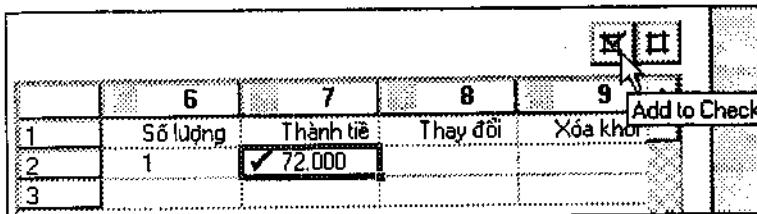
Chúng ta có thể thay đổi độ rộng cột và chiều cao dòng của bảng bằng cách kéo các cột và ngăn tiêu đề hàng.

	1	2	3	4
1	✓ Mã sản phẩm	✓ Tên sản phẩm	✓ Thể loại	✓ Đơn giá
2	✓ PH0031	✓ Giáo trình	✓ Photosho	✓ 72.000
3	✓ Total	✓ 72.000 V	✓	✓

**Chú ý:** một số môi trường và các đối tượng hỗ trợ lựa chọn một phạm vi hàng. Điều này cho phép bạn chỉ định các hàng được hiển thị trong khu vực lướt.

Nhấp vào nút **Thay đổi** cho phép bạn chỉnh sửa phạm vi hàng. Khi tạo ra một trạm kiểm soát bảng mới, tất cả các cell có chứa một dấu kiểm màu xanh cho thấy đó đều là những lựa chọn để xác minh.

Bạn có thể hướng dẫn chương trình để kiểm tra toàn bộ bảng, hàng, cột hoặc các cell cụ thể. Chương trình chỉ kiểm tra cell có chứa một dấu chọn.



Sau khi thực hiện xong, nhấp nút **Stop** để dừng bước kiểm thử lại. Thành phần được kiểm thử sẽ được chèn vào vị trí của action đã chọn.



Item	Operation	Value	Documentation
MUA			
Tu_sach_STK			
Tu_sach_STK_2			
Mã sản phẩm	Check	CheckPoint! 'Mã sả... Check whether the content	
Tu_sach_STK_3			

### Kiểm tra chữ (Text)

Hình thức kiểm tra cho phép chỉ định các văn bản được kiểm tra, cũng như xác định văn bản được hiển thị trước và sau khi văn bản được kiểm tra.

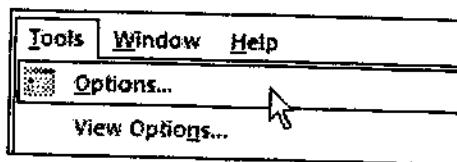
Những tùy chọn cấu hình đặc biệt hữu ích khi chuỗi văn bản mong muốn kiểm tra xuất hiện nhiều lần hoặc khi nó có thể thay đổi một cách dự đoán được trong các buổi chạy.

### Tạo một kiểm tra văn bản

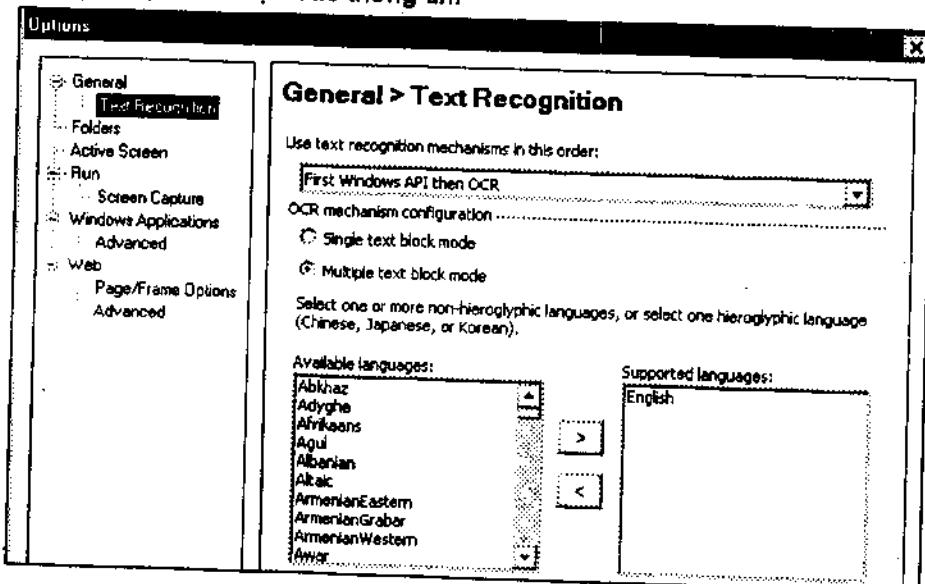
Thực hiện thêm một trạm kiểm soát trong văn bản khi thực hiện ghi chỉnh sửa các bước trong Windows hoặc ứng dụng dựa trên web.

Lưu ý: trước khi tạo ra một trạm kiểm soát văn bản, hãy chắc chắn đã thiết lập cấu hình các văn bản trong chương trình.

Trên thanh công cụ Tools, nhấp chọn lệnh Options, hộp thoại Options xuất hiện. Trong hộp thoại, tùy chỉnh các thông tin cần thiết trong tab General > Text Recognition.

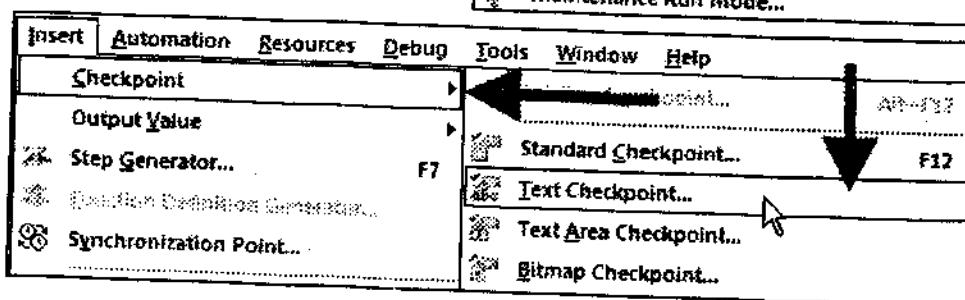
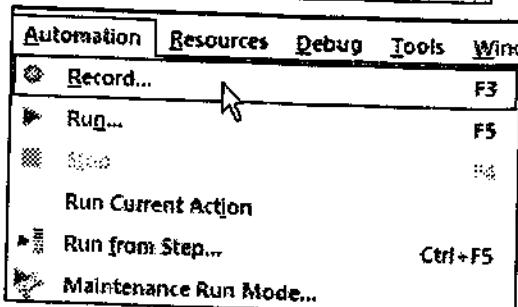


Trong hộp thoại có thể tùy chỉnh các thông tin cần thiết như ngôn ngữ chuyên dùng, tùy chỉnh nhận dạng text. Khi hoàn tất tùy chỉnh, nhấp nút OK ở cuối hộp thoại xác nhận các thông tin.

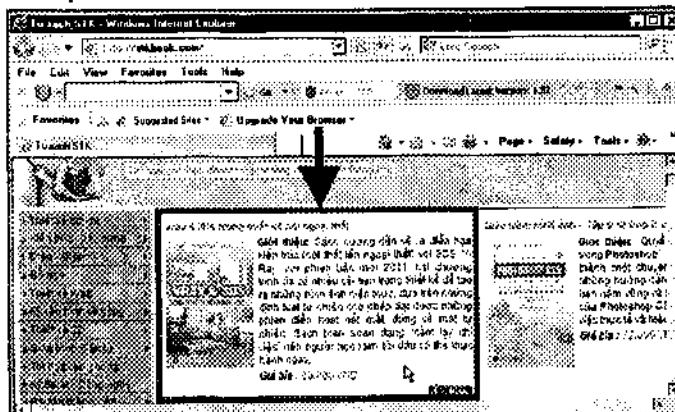


Trên thanh trình đơn nhấp chọn Menu Automation > Record (F3), bắt đầu ghi lại các tiến trình và bước thao tác trên trang web.

Bước tiếp theo, nhấp chọn Menu Insert > Checkpoint > Text Checkpoint.

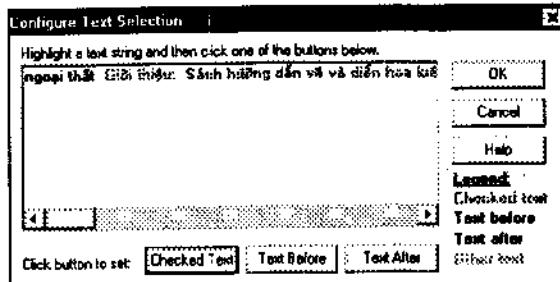
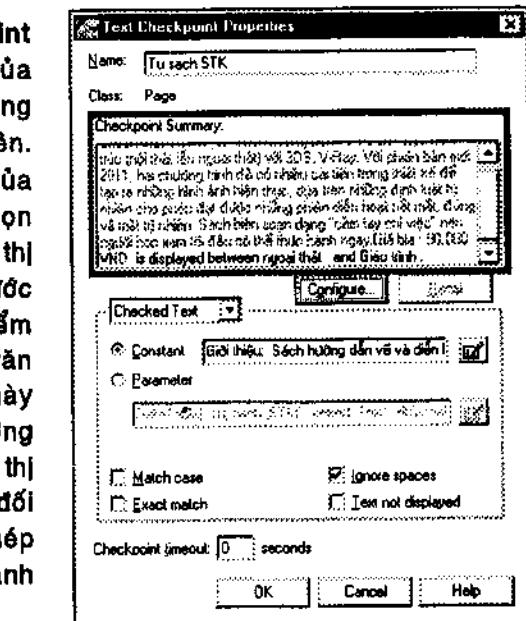


Cửa sổ chương trình sẽ ẩn, và con chuột thay đổi thành một bàn tay. Trên giao diện trang web nhấp chọn chuỗi văn bản muốn tạo ra các trạm kiểm soát Checkpoint như hình sau:



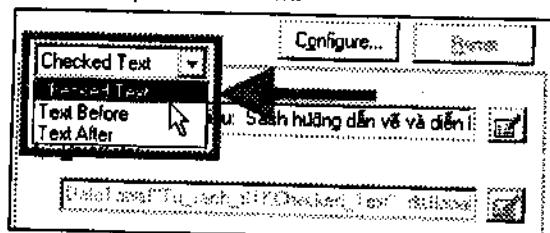
**Hộp thoại Text Checkpoint Properties** xuất hiện, nội dung của văn bản sẽ được cập nhật vào khung **Checkpoint Summary** như hình bên. Trong khung **Name** xuất hiện tên của đối tượng kiểm thử. Đoạn được chọn để thiết lập trạm kiểm soát sẽ hiển thị trong bảng cùng với đoạn nằm trước và sau nó. Văn bản được chọn kiểm tra hiển thị màu đỏ, còn các đoạn văn bản trước và sau đoạn kiểm tra này có màu xanh lam. Đối với môi trường web bình thường sẽ cho phép hiển thị cả 3 thành phần trên. Tuy nhiên đối với môi trường Window chỉ cho phép hiển thị đoạn văn bản đã chọn để đánh dấu kiểm soát.

Để xem các thao tác trong văn bản được lựa chọn, hãy nhấp nút **Configure**. Hộp thoại **Configure Text Selection** xuất hiện, nhấp chọn các tùy chọn thiết lập trạm kiểm soát **Checked Text**, **Text Before**, **Text After**.



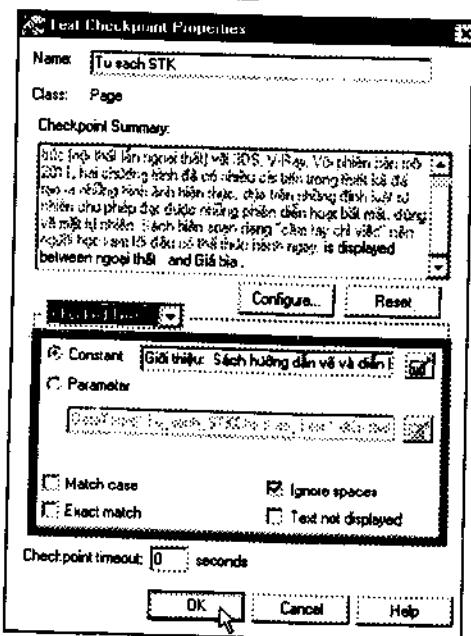
Nhấp vào nút OK để đóng hộp thoại, một tuyên bố cho biết trạm kiểm soát được thêm vào đối tượng được lựa chọn.

Quay lại hộp thoại chính thực hiện lựa chọn các thuộc tính và thiết lập các thông số còn lại cho trạm kiểm soát.

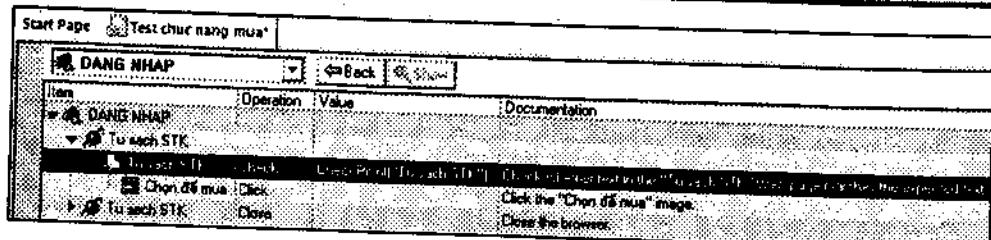
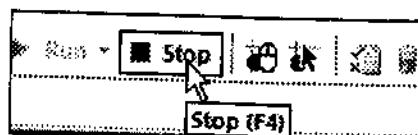


Thực hiện tương tự như vậy để thiết lập các thông số hàm cho trạm kiểm soát.

Cuối cùng nhấp nút OK thêm trạm kiểm soát vào trong action.



Sau khi thực hiện xong, nhấp nút Stop, dừng bước kiểm thử. Thành phần được kiểm thử sẽ chèn vào vị trí của action đã chọn.



**CHƯƠNG 13****KIỂM THỬ VỚI JUNIT 4.9**

Trong 3 chương 10, 11, 12 các bạn đã tìm hiểu và sử dụng QuickTest Professional trong kiểm thử phần mềm, chương 13 hướng dẫn các bạn sử dụng một chương trình khác cũng hỗ trợ cho việc kiểm thử đó là JUnit. JUnit được xây dựng bởi Kent Beck và Erich Gamma là công cụ giúp thử nghiệm, gỡ rối chương trình Java, là một framework đơn giản dùng cho việc tạo các Unit testing tự động. JUnit có thể chạy các kiểm tra lặp đi lặp lại, nó chỉ là một phần của họ khung trúc XUnit cho việc tạo các Unit Testing.

Với JUnit, kiểm thử viên dễ dàng theo dõi diễn biến của chương trình, nhanh chóng dàn dựng hàng loạt phép thử (Test Case) để kiểm tra xem mọi việc có xảy ra đúng như dự định hay không. JUnit là chuẩn cho Unit Testing trong Java, sau này các lập trình viên Java biến JUnit thành một đề án nguồn mở. Hiện nay, JUnit trở thành một "công cụ chuẩn" mà mọi lập trình viên Java đều nên biết cách dùng.

JUnit sử dụng mã nguồn mở giúp các lập trình viên phát triển Java viết những Unit Test kiểm tra từng module của dự án khi phát triển hệ thống. Framework giúp thiết lập một mối quan hệ giữa người kiểm thử và người phát triển phần mềm. Đầu tiên, cần viết ra các đoạn code sẽ làm việc. Sau đó viết code và dùng JUnit Test Runner kiểm tra xem nó có bị chênh hướng so với dự định ban đầu như thế nào.

Integration Testing xác nhận rằng: những hệ thống con khác nhau sẽ làm việc tốt khi kết hợp chúng lại với nhau. Trong khi đó, Acceptance Testing xác nhận chính xác một ứng dụng có làm việc đúng như yêu cầu khách hàng mong đợi không. Unit Test được gọi như vậy vì sẽ kiểm tra từng đoạn code đơn lẻ, nó có thể chỉ là một class đơn trong Java.

Khác với các Unit Test đặc thù, ở đó người lập trình có khuynh hướng sẽ viết đoạn kiểm tra (test) sau khi hoàn thành module. JUnit giúp kết hợp coding và testing trong suốt quá trình phát triển. Lúc này, mục đích chính là kiểm tra module ở mức nhỏ (kiểm tra các chức năng) hơn là kiểm tra các khối cơ bản của hệ thống tại một thời điểm nào đó.

#### **Một số đặc điểm quan trọng của JUnit:**

- Xác nhận (assert) 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.

Các Test Case của JUnit là các lớp của Java, các lớp này bao gồm một hay nhiều các phương thức Unit Testing, và những kiểm tra này lại được nhóm thành các Test Suite.

Mỗi phương thức kiểm tra trong JUnit phải được thực thi nhanh chóng. Tốc độ là thuộc tính rất quan trọng vì càng nhiều kiểm tra được viết và tích hợp vào bên trong quá trình xây dựng phần mềm, cần phải tốn nhiều thời gian hơn cho việc chạy toàn bộ Test Suite. Các lập trình viên không muốn bị ngắt quãng trong một khoảng thời gian dài trong khi các kiểm tra đang chạy. Do đó, các kiểm tra chạy càng lâu sẽ có nhiều khả năng là: các lập trình viên sẽ bỏ qua bước không kém phần quan trọng này.

Các kiểm tra (test) trong JUnit có thể là các kiểm tra được chấp nhận hay thất bại, các test này được thiết kế để chạy mà không cần có sự can thiệp của con người. Từ những thiết kế như thế, bạn có thể thêm các bộ test vào quá trình tích hợp và xây dựng phần mềm một cách liên tục và cho các test chạy một cách tự động.

## MỘT SỐ PHƯƠNG THỨC TRONG JUNIT

### Các phương thức assertXXX()

Các phương thức assertXXX() được dùng để kiểm tra các điều kiện khác nhau. Junit.framework.TestCase, lớp cha cho tất cả các Test Case, thừa kế từ lớp junit.framework.Assert. Lớp này định nghĩa khá nhiều các phương thức assertXXX(). Các phương thức test hoạt động bằng cách gọi những phương thức này.

Phần trình bày sau mô tả các phương thức assertXXX() khác nhau có trong lớp junit.framework.

- **Boolean assertEquals():** so sánh 2 giá trị để kiểm tra bằng nhau. Test sẽ được chấp nhận nếu các giá trị bằng nhau
- **Boolean assertFalse():** Đánh giá một biểu thức luận lý. Test sẽ được chấp nhận nếu biểu thức sal.
- **Boolean assertNotSame():** so sánh địa chỉ vùng nhớ của 2 tham chiếu đối tượng bằng cách sử dụng toán tử ==. Test sẽ được chấp nhận nếu cả 2 đều tham chiếu đến các đối tượng khác nhau
- **Boolean assertNull():** so sánh tham chiếu của một đối tượng với giá trị null. Test sẽ được chấp nhận nếu tham chiếu là null
- **Boolean assertSame():** so sánh địa chỉ vùng nhớ của 2 tham chiếu đối tượng bằng cách sử dụng toán tử ==. Test sẽ được chấp nhận nếu cả 2 đều tham chiếu đến cùng một đối tượng
- **Boolean assertTrue():** đánh giá một biểu thức luận lý. Test sẽ được chấp nhận nếu biểu thức đúng

- **Void fail():** phương thức này làm cho test hiện hành thất bại, phương thức này thường được sử dụng khi xử lý các trường hợp đặc biệt.

Bạn có thể chỉ cần sử dụng phương thức `assertTrue()` cho gần như hầu hết các test, tuy nhiên việc sử dụng một trong các phương thức `assertXXX()` cụ thể sẽ làm cho các test của bạn dễ hiểu hơn và cung cấp các thông điệp thất bại rõ ràng hơn.

Tất cả các phương thức trên đều nhận vào một String không bắt buộc làm tham số đầu tiên. Khi được xác định, tham số này cung cấp một thông điệp mô tả test thất bại.

**Ví dụ:**

`assertEquals(employeeA, employeeB);`

`assertEquals("Employees should be equal after the clone() operation.", employeeA, employeeB)..Set Up và Tear Down`

Hai phương thức `setUp()` và `tearDown()` là một phần của lớp JUnit - Framework - TestCase. Khi sử dụng 2 phương thức `setUp()` và `tearDown()` sẽ giúp ta tránh được việc trùng mã khi nhiều test cùng chia sẻ nhau ở phần khởi tạo và dọn dẹp các biến.

JUnit tuân thủ theo một dây có thứ tự các sự kiện khi chạy các test. Đầu tiên, nó tạo ra một thể hiện mới của Test Case ứng với mỗi phương thức test. Từ đó, nếu bạn có 5 phương thức test thì JUnit sẽ tạo ra 5 thể hiện của Test Case. Vì lý do đó, các biến thể hiện không thể được sử dụng để chia sẻ trạng thái giữa các phương thức test. Sau khi tạo xong tất cả các đối tượng test case, JUnit tuân theo các bước sau cho mỗi phương thức test:

- Gọi phương thức `setUp()` của Test Case.
- Gọi phương thức test.
- Gọi phương thức `tearDown()` của Test Case.

Quá trình này được lặp lại đối với mỗi phương thức test trong Test Case. Sau đây chúng ta sẽ xem xét ví dụ: tính cộng, trừ và nhân hai số nguyên:

### CODE

```
public class CongTruNhan
```

```
{
```

```
    int add (int x, int y){  
        return (x + y );  
    }  
    int multiply (int x, int y){  
        return (x*y);  
    }
```

```

    }

    Int subtract (int x, int y){
        return (x-y);
    }

}

```

Mã Code Test Case của lớp trên có sử dụng phương thức `setUp()` và `tearDown()`:

#### CODE

```

import junit.framework.*;
public class Test CongTruNhan extends TestCase {
    public Test CongTruNhan (String name) {
        super(name);
    }
    // Initialize common test data
    int x;
    int y;
    protected void setUp()
    {
        System.out.println("setUp - Initialize common test data");
        x = 7;
        y = 5;
    }
    protected void tearDown(){
        System.out.println("tearDown - Clean up");
    }
    /**
     * Test of add method, of class CongTruNhan.
     */
    public void testAdd() {
        System.out.println("add");
        CongTruNhan instance = new CongTruNhan ();
        Int expResult = 12;
        Int result = instance.add(x, y);
    }
}

```

```
assertEquals(expResult, result);
}

/*
 * Test of multiply method, of class CongTruNhan.
 */
public void testMultiply() {
    System.out.println("multiply");
    CongTruNhan instance = new CongTruNhan();
    int expResult = 35;
    int result = instance.multiply(x, y);
    assertEquals(expResult, result);
}
/*
 * Test of subtract method, of class CongTruNhan.
 */
public void testSubtract() {
    System.out.println("subtract");
    CongTruNhan instance = new CongTruNhan();
    int expResult = 2;
    int result = instance.subtract(x, y);
    assertEquals(expResult, result);
}
public static void main(String[] args) {
    System.out.println("Running the test using junit.textui.TestRunner.");
    run() method...");
    junit.textui.TestRunner.run(TestCongTruNhan.class);
}
}
```

Thông thường chúng có thể bỏ qua phương thức tearDown() vì mỗi Unit Test riêng không phải là những tiến trình chạy tốn nhiều thời gian, và các đối tượng được thu dọn khi JVM thoát. Teardown() có thể được sử dụng khi test thực hiện những thao tác như mở kết nối đến cơ sở dữ liệu hay sử dụng các loại tài nguyên khác của hệ thống và bạn cần phải dọn dẹp ngay lập tức.

Nói chung, chạy một số lượng lớn các Unit Test thì khi bạn trả tham chiếu của các đối tượng đến null bên trong thân phương thức tearDown() sẽ giúp cho bộ dọn rác lấy lại bộ nhớ khi các test khác chạy.

Đôi khi bạn muốn chạy vài đoạn mã khởi tạo chỉ một lần, sau đó chạy các phương thức test, và bạn chỉ muốn chạy các đoạn mã dọn dẹp chỉ sau khi tất cả test kết thúc. Ở phần trên, JUnit gọi phương thức setUp() trước mỗi test và gọi tearDown() sau khi mỗi test kết thúc. Vì thế để làm được điều như trên, chúng ta sẽ sử dụng lớp junit.extensions.TestSetup để đạt được yêu cầu trên.

Ví dụ:

CODE

```
import junit.extensions.TestSetup;
import junit.framework.*;
public class TestPerson extends TestCase {
    public TestPerson(String name)
        { super(name); }
    public void testGetFullName()
    {
        Person p = new Person("Aidan", "Burke");
        assertEquals("Aidan Burke", p.getFullName());
    }
    public void testNullsInName()
    {
        Person p = new Person(null, "Burke");
        assertEquals("? Burke", p.getFullName()); p = new Person("Tanner",
        null); assertEquals("Tanner ?", p.getFullName());
    }
}

public static Test suite()
{
    TestSetup setup = new TestSetup(new TestSuite(TestPerson.class))
    {
        protected void setUp() throws Exception
        {
            //Thực hiện các đoạn mã khởi tạo một lần ở đây
        }
        protected void tearDown() throws Exception
        {
            //Thực hiện các đoạn mã dọn dẹp ở đây
        }
    };
}
```

```

    }
};

return setup;
}
}
}

```

TestSetup là một lớp thừa kế từ lớp junit.extension.TestDecorator. Lớp TestDecorator là lớp cơ sở cho việc định nghĩa các test biến thể. Lý do chính để mở rộng TestDecorator là thực thi đoạn mã trước và sau khi một test chạy. Các phương thức `setUp()` và `tearDown()` của lớp TestSetup được gọi trước và sau khi bất kỳ Test nào được truyền vào constructor.

Trong ví dụ trên, chúng ta đã truyền một tham số có kiểu TestSuite vào constructor của lớp TestSetup.

```
TestSetup setup = new TestSetup(new TestSuite(TestPerson.class))
```

Điều này có nghĩa: 2 phương thức `setUp()` được gọi chỉ một lần trước toàn bộ bộ test và `tearDown()` được gọi chỉ một lần sau khi các test trong bộ test kết thúc.

**Chú ý:** các phương thức `setUp()` và `tearDown()` bên trong lớp TestPerson vẫn được thực thi trước và sau mỗi phương thức test bên trong lớp TestPerson.

### Tạo test class và tạo bộ test

Test các chức năng của một lớp: nếu muốn viết các Unit Test với Junit, việc đầu tiên phải tạo một lớp con thừa kế từ lớp junit.framework.TestCase. Mỗi Unit Test được đại diện bởi một phương thức `testXXX()` bên trong lớp con của lớp TestCase

Ta có một lớp Person như sau:

#### CODE

```
public class Person { private String firstName; private String lastName;
public Person(String firstName, String lastName) {
if (firstName == null && lastName == null) {
throw new IllegalArgumentException("Both names cannot be null");
}
this.firstName = firstName;
this.lastName = lastName;
}
public String getFullName() {
```

```

String first = (this.firstName != null) ? this.firstName : "?"; String last =
(this.lastName != null) ? this.lastName : "?"; return first + last;
}

public String getFirstName() {
return this.firstName;
}

public String getLastName() {
return this.lastName;
}
}

```

Sau đó thực hiện viết một test case đơn giản để test một số phương thức của lớp trên:

#### CODE

```

import junit.framework.TestCase;
public class TestPerson extends TestCase
{
    public TestPerson(String name) {
        super(name);
    }
    /**
     * Xac nhan rang name duoc the hien dung dinh dang
     */
    public void testGetFullName() {
        Person p = new Person("Aidan", "Burke");
        assertEquals("Aidan Burke", p.getFullName());
    }

    /**
     * Xac nhan rang nulls da duoc xu ly chinh xac
     */
    public void testNullsInName()
    {

```

```

Person p = new Person(null, "Burke"); assertEquals("? Burke",
p.getFullName());
p = new Person("Tanner", null); assertEquals("Tanner ?",
p.getFullName());
}
)

```

Lưu ý: mỗi Unit Test là một phương thức public và không có tham số được bắt đầu bằng tiếp đầu ngữ test.

Nếu không tuân theo quy tắc đặt tên này thì JUnit sẽ không xác định được các phương thức test một cách tự động. Để biên dịch TestPerson, cần phải khai báo gói thư viện Junit trong biến classpath:

```
set classpath=%classpath%;.:junit.jar
```

```
javac TestPerson
```

#### Tổ chức các test vào các test suite:

Thông thường JUnit tự động tạo ra các Test Suite ứng với mỗi Test Case. Tuy nhiên, muốn tự tạo các Test Suite của riêng mình bằng cách tổ chức các Test vào Test Suite, JUnit cung cấp lớp junit.framework.TestSuite hỗ trợ việc tạo các Test Suite.

Khi sử dụng giao diện text hay graphic, JUnit sẽ tìm phương thức sau trong Test Case của bạn. JUnit sẽ sử dụng kỹ thuật reflection để tự động xác định tất cả các phương.

```
public static Test suite() { ... }
```

Nếu không thấy phương thức testXXX() trong Test Case của bạn, thêm chúng vào một Test Suite. Nó sẽ chạy tất cả các test trong suite này. Có thể tạo ra bản sao hành vi của phương thức suite() mặc định như sau:

```
public class TestGame extends TestCase
```

```
{
```

```

public static Test suite()
{
    return new TestSuite(TestGame.class);
}
```

```
}
```

Bằng cách truyền đối tượng TestGame.class vào constructor TestSuite, đang thông báo cho JUnit biết để xác định tất cả các phương thức testXXX() trong lớp đó và thêm chúng vào suite.

Đoạn mã trên không làm khác gì so với việc JUnit tự động làm, tuy nhiên có thể thêm các test cá nhân để chỉ chạy các test nhất định nào đó hay là điều khiển thứ tự thực thi.

### CODE

```
import junit.framework.*;
public class TestGame extends TestCase
{
    private Game game;
    private Ship fighter;
    public TestGame(String name)
    {
        super(name);
    }
    -
    public static Test suite()
    {
        TestSuite suite = new TestSuite();
        suite.addTest(new TestGame("testCreateFighter"));
        suite.addTest(new TestGame("testSameFighters"));
        return suite;
    }
}
```

Ngoài ra, có thể kết hợp nhiều suite vào các suite khác. Ví dụ sau sử dụng mẫu Composite.

Ví dụ:

```
public static Test suite()
{
    TestSuite suite = new TestSuite(TestGame.class); suite.addTest(new
    TestSuite(TestPerson.class)); return suite;
}
```

Bây giờ khi chạy Test Case này, bạn sẽ chạy tất cả các test bên trong lớp TestGame và lớp TestPerson.

### **Chạy các Test lặp đi lặp lại**

Trong một vài trường hợp, chúng ta muốn chạy một test nào đó lặp đi lặp lại nhiều lần để đo hiệu suất hay phân tích các vấn đề trực trắc. JUnit cung cấp cho chúng ta lớp junit.extension.RepeatedTest để làm được điều này. Lớp RepeatedTest giúp chúng ta thực hiện điều này một cách dễ dàng.

#### **CODE**

```
public static Test suite()
{
    //Chạy toàn bộ test suite 10 lần
    return new RepeatedTest(new TestSuite(TestGame.class), 10);
}
```

Tham số đầu tiên của RepeatedTest là một Test cần chạy, tham số thứ 2 là số lần lặp lại. Vì TestSuite cài đặt Interface Test nên chúng ta có thể lặp lại toàn bộ test như trên.

Ví dụ sau mô tả cách xây dựng một Test Suite mà trong đó các test khác nhau được lặp đi lặp lại khác nhau:

#### **CODE**

```
public static Test suite()
{
    TestSuite suite = new TestSuite();
    //Lặp lại testCreateFighter 100 lần
    suite.addTest(new RepeatedTest(new TestGame("testCreateFighter"),
        100));
    //Chạy testSameFighters 1 lần
    suite.addTest(new TestGame("testSameFighters"));
    //Lặp lại testGameInitialState 20 lần
    suite.addTest(new RepeatedTest(new TestGame("testGameInitialState"),
        20));
    return suite;
}
```

### **Test các Exception**

Phần trình bày sau sẽ xem xét đến một phần test cũng quan trọng không kém trong lập trình là test các exception.

Chúng ta sử dụng cặp từ khóa try/catch để bắt các exception như mong đợi, gọi phương thức fail() khi exception chúng ta mong đợi không xảy ra.

Trong ví dụ sau, constructor của lớp Person gửi ra Illegal Argument Exception khi cả 2 tham số của nó đều mang giá trị null. Test sẽ thất bại nếu nó không gửi ra exception.

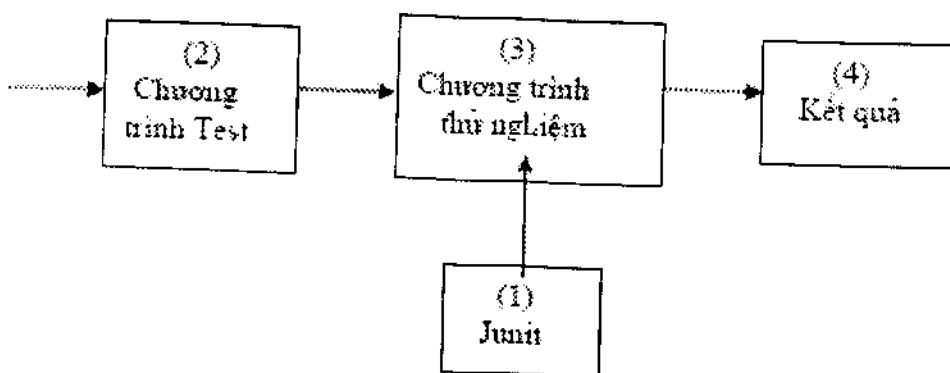
#### CODE

```
public void testPassNullsToConstructor() {
try {
Person p = new Person(null, null);
fail("Expected IllegalArgumentException because both args are null");
}
catch (IllegalArgumentException expected) {
//Bỏ qua phần không xử lý vì có nghĩa là test được chấp nhận
}
}
```

Chỉ nên sử dụng kỹ thuật này khi bạn mong đợi một exception xảy ra. Đối với các điều kiện lỗi khác bạn nên để exception chuyển sang cho JUnit. Khi đó JUnit sẽ bắt lấy và tường trình 1 lỗi test.

#### SƠ ĐỒ KIỂM THỬ

Sơ đồ biểu diễn cách test chương trình:



- Junit: công cụ dùng để test (tool Junit)
- Chương trình test
- Chương trình thử nghiệm: viết một TestCase để test chương trình trên.
- Kết quả: sau khi kiểm thử chương trình xong sẽ đưa ra kết quả là chương trình đúng hay sai.

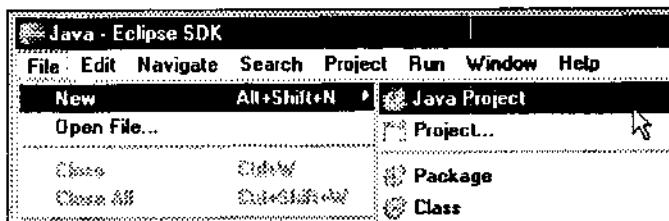
## LẬP KẾ HOẠCH KIỂM THỬ

- Bước 1: đầu tiên, ta phải tạo ra một dự án.
- Bước 2: add Junit 4.9 vào thư mục chứa đường dẫn của chương trình.
- Bước 3: mở chương trình test và tạo TestCase cho chương trình.
- Bước 4: chạy chương trình

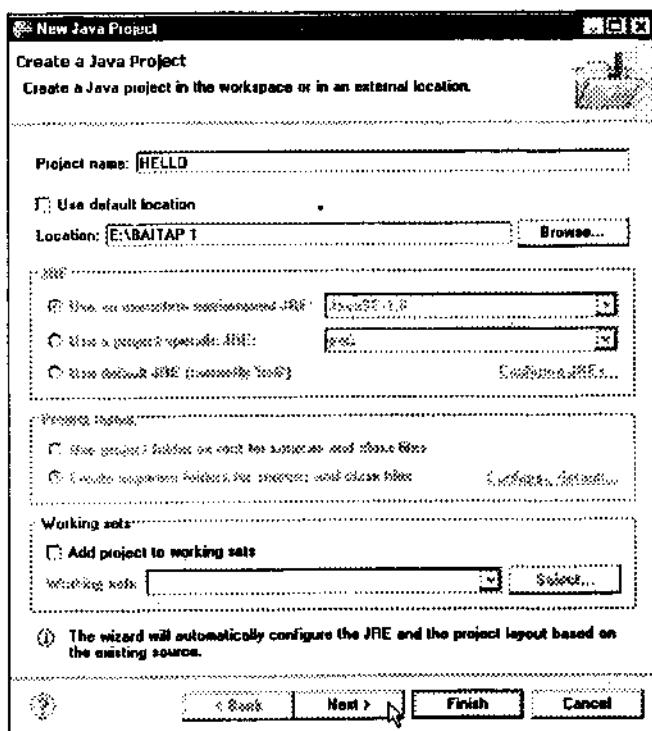
## QUÁ TRÌNH KIỂM THỬ

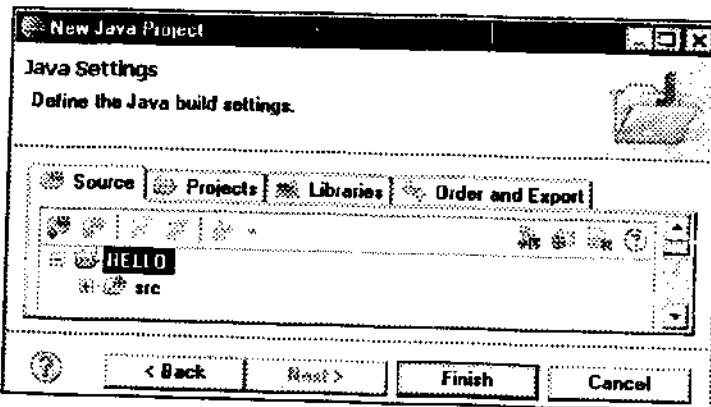
Bước 1: khởi tạo dự án mới.

Thực hiện cài đặt chương trình Eclipse, sau đó tạo mới một dự án bằng cách: khởi động chương trình, nhấp chọn menu **File > New > Java Project**.

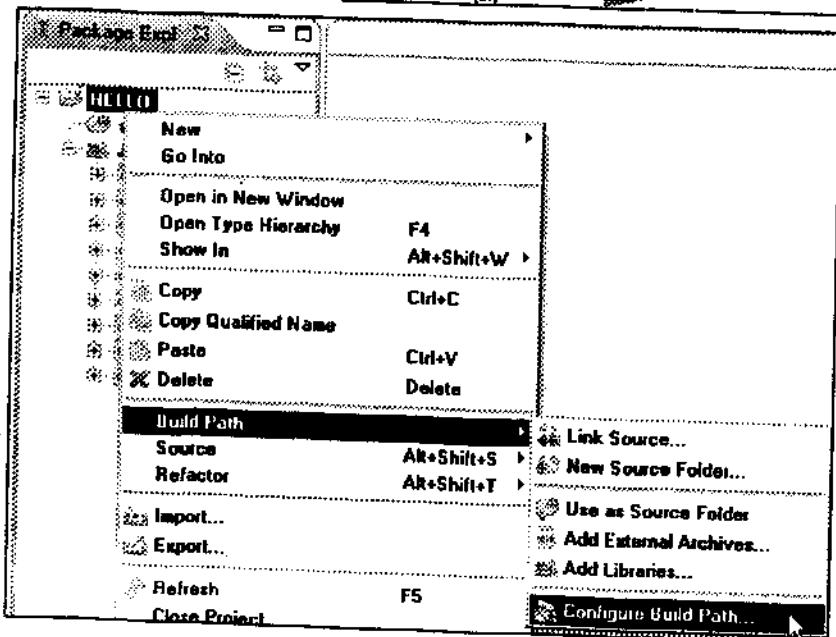
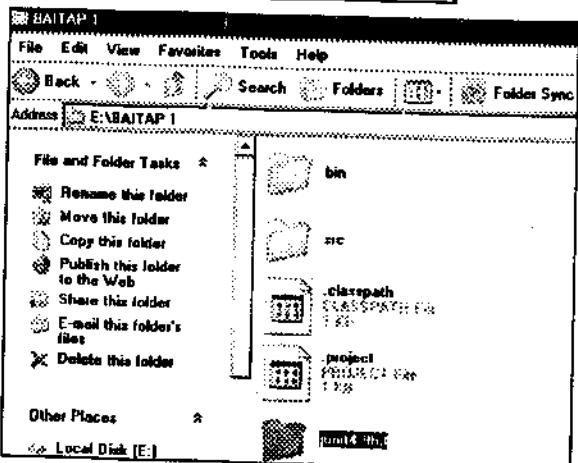


Hộp thoại **New Java Project** xuất hiện, nhập tên cho Project vào khung **Project name** và chọn đường dẫn đến thư mục bài tập cần lưu, sau đó nhấp nút **NEXT** cuối hộp thoại để tiếp tục bước khởi tạo. Hộp thoại mới sẽ xuất hiện thành phần Java settings, nội dung của phần source xuất hiện cùng với các tab làm việc như: **Projects**, **Libraries**, **Order** và **Export**. Sau đó nhấp nút **Finish** để kết thúc khởi tạo thư viện.

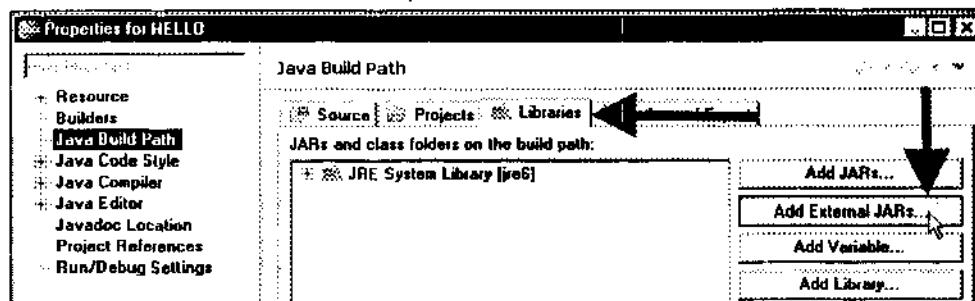




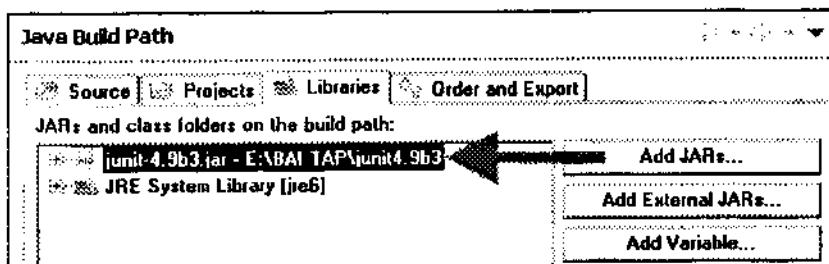
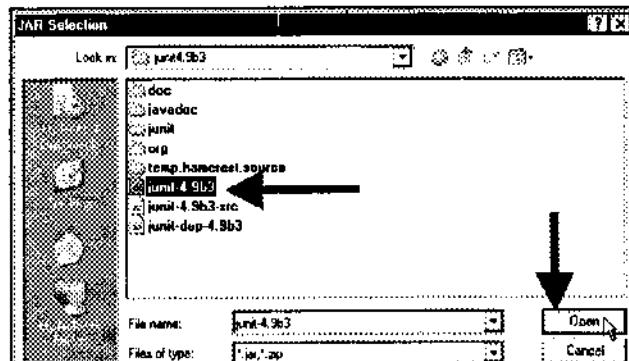
Thực hiện sao chép gói JUnit 4.9 vào thư mục bài tập của chương trình, đồng thời đưa dự án mẫu vào thư mục src. Quay lại chương trình Eclipse, trong khung Package Explorer, nhấp phải chọn Build Path > Configure Build Path.., phần lệnh này sẽ thêm thư viện và thực hiện gọi lại các thao tác xây dựng.



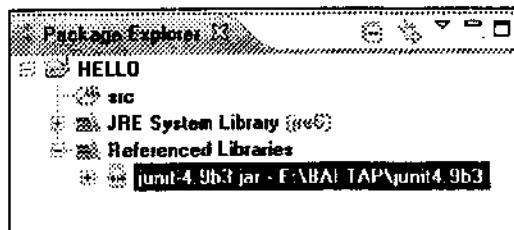
Hộp thoại Properties for... xuất hiện, nhấp chọn Tab Libraries > Add External JARs... để thêm thư viện cho Test Junit 4.9.



Hộp thoại JAR Selection xuất hiện, nhấp chọn đường dẫn tới thư mục chứa Junit trong file bài tập và chọn file junit 4.9b3, sau đó nhấp nút Open. Như vậy Junit đã được đưa vào tab thư viện của chương trình.

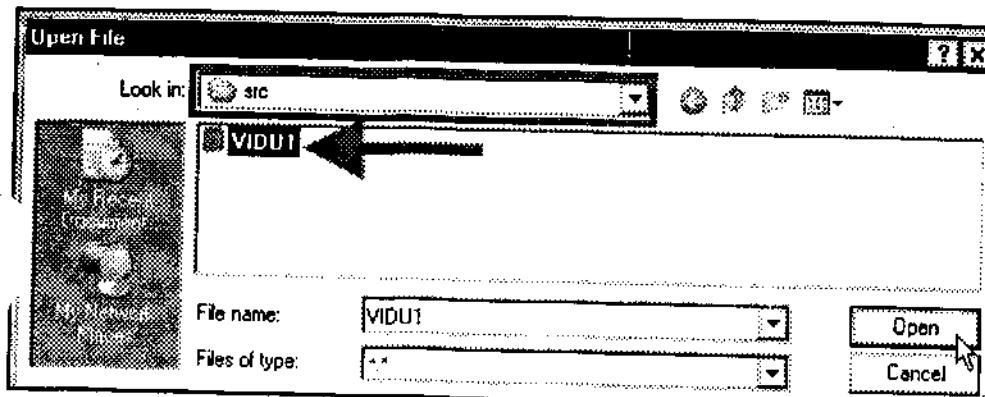


Đồng thời trên giao diện chương trình cũng xuất hiện một thành phần thư viện của Junit trong hộp thoại Package Explorer dưới dạng cây như hình bên:



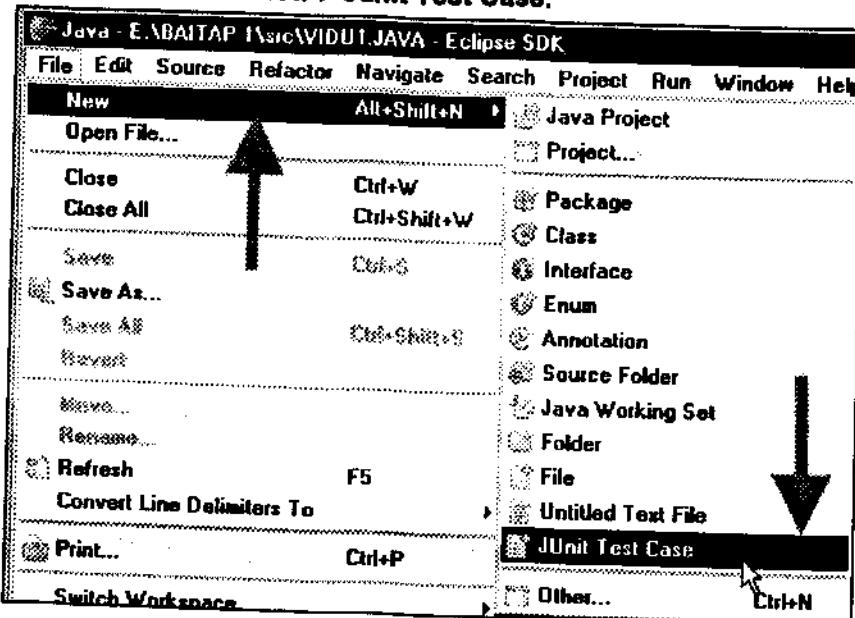
Chọn menu File > Open File để mở nội dung của của một dự án được xây dựng bằng Java. Hộp thoại Open File xuất hiện, hãy chỉ đường dẫn đến thư mục src và chọn tên file dự án.



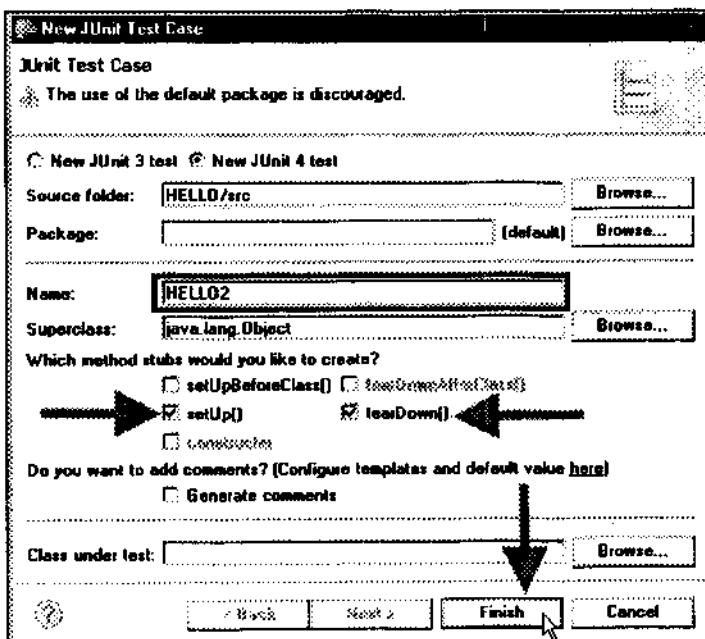


```
VIDU1.java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
    public static void say(String msg) {
        for (int i = 0; i < 3; i++) {
            System.out.println(msg);
        }
    }
}
```

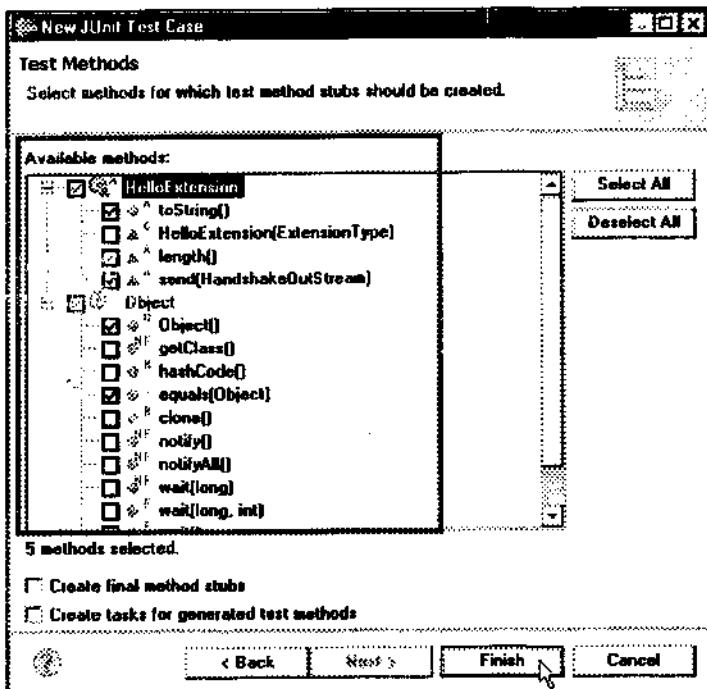
Bước kế tiếp, thực hiện tạo một **Test Case** thông qua thư viện Junit. Nhấp chọn menu **File > New > Junit Test Case**.



Hộp thoại **New JUnit Test Case** xuất hiện, thực hiện nhấp tên của Case là **HELLO2** vào khung Name. Nhấp chọn chế độ khởi tạo là **Setup()** và **Teardown()** sau đó nhấp nút **Finish**.



Hộp thoại mới xuất hiện, nhấp chọn các thành phần cần kiểm thử trong cây thư mục. Sau khi chọn các thành phần thì căn cứ vào đó, chương trình sẽ thực hiện kiểm tra các thành phần có liên quan. Nhập **Finish** để kết thúc các thiết lập.



File Test Case sẽ xuất hiện trong chương trình như hình dưới: trong đó thông báo tất cả các thành phần kiểm thử và đánh dấu các lỗi xuất hiện của chương trình theo từng phương pháp đã chọn.

```

VIDU1.JAVA HELLOTEST.java
import static org.junit.Assert.*;

public class HELLOTEST {

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

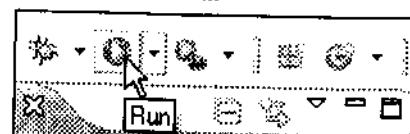
    @Test
    public void testToString() {
        fail("Not yet implemented");
    }

    @Test
    public void testLength() {
        fail("Not yet implemented");
    }

    @Test
    public void testSend() {
    }
}

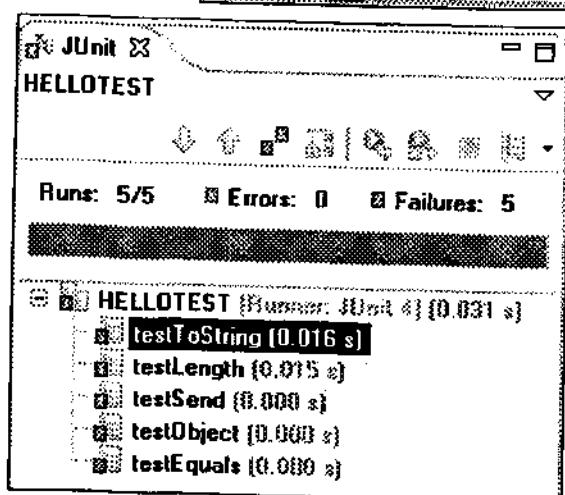
```

Nhấp nút Run và bắt đầu test bằng JUnit.



Kết quả các lỗi sẽ được tổng kết và thông báo một lần nữa tại cột JUnit như hình.

Để thực hiện sửa lỗi, hãy thực hiện trực tiếp trên chương trình và thực hiện kiểm thử bằng chương trình lần nữa.



## CHƯƠNG 14

# KIỂM THỬ VỚI SELENIUM 1.0

Selenium (thường được viết tắt là SE) là phần mềm mã nguồn mở, được phát triển bởi Jason Huggins, sau đó được tiếp tục phát triển bởi nhóm ThoughtWorks vào năm 2004. Phiên bản hoàn chỉnh mới nhất là 1.0.1 được phát hành vào 10/06/2009. Selenium kiểm tra tự động bằng cách sử dụng một công cụ để chạy thử nghiệm lặp lại đối với các ứng dụng đã định làm mục tiêu kiểm thử. Selenium tập hợp các công cụ phát triển mạnh, hỗ trợ nhanh chóng các kiểm thử tự động dành cho các ứng dụng Web. Đây là công cụ kiểm tra các ứng dụng Web rất hiệu quả. Ngoài ra, Selenium còn cung cấp một tập phong phú dùng cho các thử nghiệm chức năng đặc biệt đáp ứng các nhu cầu kiểm thử của một ứng dụng Web.

Sử dụng Selenium không phải luôn gặp thuận lợi khi tự động hóa các trường hợp kiểm thử. Có khi kiểm thử bằng tay lại thích hợp hơn trong trường hợp: giao diện của người dùng cho trình ứng dụng sẽ thay đổi trong tương lai gần, lúc đó bất kỳ quá trình tự động hóa nào cũng được viết lại. Lập trình viên không đủ thời gian để viết quá trình tự động hóa. Với thời gian ngắn, việc kiểm tra bằng tay có thể có hiệu quả hơn. Nếu một ứng dụng có yêu cầu thời gian thực hiện rất chặt chẽ mà lại không có chương trình tự động kiểm tra có sẵn, việc kiểm tra bằng tay là giải pháp tốt nhất.

Tuy nhiên, việc tự động kiểm thử có lợi thế để nâng cao hiệu quả lâu dài của một nhóm phần mềm kiểm tra quy trình.

## Kiểm tra tự động hỗ trợ:

- Tương xứng kiểm tra quy hồi.
- Phản hồi nhanh chóng để sửa chữa trong quá trình phát triển.
- Hầu như không có giới hạn cho sự thực hiện lặp đi lặp lại trường hợp thử nghiệm phần mềm.
- Tùy chỉnh báo cáo của các sai sót trong ứng dụng.
- Hỗ trợ cho Agile và phương pháp phát triển eXtreme.
- Tài liệu có tính logic cho các trường hợp thử nghiệm.
- Tìm các lỗi bị che giấu bằng cách kiểm tra bằng tay.

Selenium gồm có ba công cụ chính. Mỗi công cụ có một vai trò cụ thể trong việc hỗ trợ sự phát triển của tự động hóa kiểm tra ứng dụng web.

- Selenium là một công cụ hỗ trợ kiểm tra tự động cho các ứng dụng chạy trên nền Web. Selenium hỗ trợ kiểm tra hầu hết trên các trình duyệt phổ biến hiện nay như Firefox, Internet Explorer, Safari... cũng như các hệ điều hành chủ yếu như Windows, Linux, Mac...
- Selenium hỗ trợ nhiều ngôn ngữ lập trình như C#, Java, Perl, PHP, Python, Ruby...
- Selenium có thể kết hợp thêm với một số công cụ khác như Bromien, Junit nhưng với người dùng thông thường chỉ cần chạy tự động mà không cần cài thêm các công cụ bổ trợ.
- Selenium bao gồm một bộ các công cụ hỗ trợ kiểm tra tự động tính năng của ứng dụng Web bao gồm: Selenium IDE, Selenium Remote Control (RC), Selenium Core và Selenium Grid.
  - **Selenium IDE:** được phát hành dưới dạng một ứng dụng mở rộng (add-on) cho Mozilla Firefox phiên bản 2.0 trở lên. Công cụ cung cấp chức năng "Thu và Chạy lại-Record and Playback". Nhờ đó Tester có thể nhanh chóng tạo một bộ kịch bản kiểm tra (test script) bằng cách trực tiếp "thu" các thao tác của mình trên đối tượng cần kiểm tra thành một tập những câu lệnh "Selenese" (ngôn ngữ kịch bản được phát triển cho Selenium IDE và Selenium Core – có dạng văn bản HTML). Sau đó chạy lại các câu lệnh này để kiểm tra. Chức năng này rất hữu dụng, cho phép tiết kiệm thời gian viết kịch bản kiểm tra. Selenium IDE cho phép lưu kịch bản đã thu dưới nhiều loại ngôn ngữ lập trình khác nhau như Java, PHP, C#, Ruby, Perl hay Python.
  - **Selenium RC:** công cụ này có thể nhận các test script được thu bởi Selenium IDE, cho phép chỉnh sửa, cải tiến linh hoạt bằng nhiều ngôn ngữ lập trình khác nhau. Sau đó khởi động một trong các trình duyệt web được chỉ định để thực thi kiểm tra trực tiếp trên trình duyệt đó. Selenium RC còn cung cấp khả năng lưu lại kết quả kiểm tra.
  - **Selenium Grid:** thực hiện phương pháp kiểm tra phân bố, phối hợp nhiều kết quả của Selenium RC để có thể thực thi trên nhiều trình duyệt web khác nhau trong cùng một lúc. Selenium Grid cũng cho phép lưu lại kết quả kiểm tra.
  - **Selenium Core:** đã được tích hợp trong Selenium IDE. Selenium Core là một công cụ chạy các test script viết bằng Selenese. Thế mạnh của công cụ này là có thể chạy test script trên hầu hết các trình duyệt, nhưng lại yêu cầu được cài đặt trên máy chủ của ứng dụng hay website cần kiểm tra. Điều này là không thể khi người kiểm thử không có quyền truy cập đến máy chủ.

Có thể tải bộ công cụ của Selenium tại: <http://seleniumhq.org/download/>

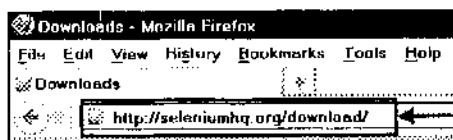
Selenium bao gồm một bộ các công cụ hỗ trợ kiểm tra tự động tính năng các ứng dụng web bao gồm: Selenium IDE, Selenium Remote Control (RC), Selenium Core và Selenium Grid. Yêu cầu hệ thống phải sử dụng hệ điều hành Win XP, Vista, Win 7.... và trình duyệt Mozilla Firefox 2.0 trở lên.

## 1 HƯỚNG DẪN CÀI ĐẶT SELENIUM IDE

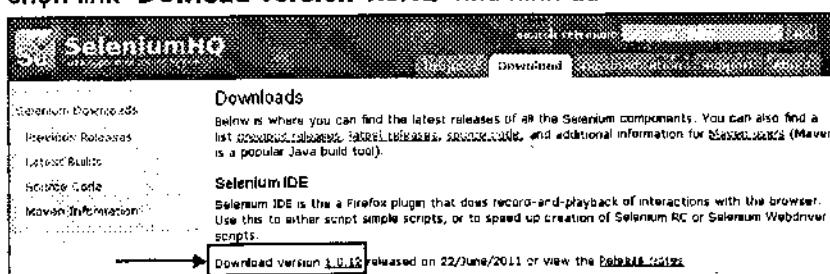
Selenium IDE (Integrated Development Environment) được phát hành dưới dạng phần mềm bổ trợ (add-on) của Firefox, cho phép test, edit và debug code. Selenium có thể sinh code tự động hoặc nạp các đoạn mã viết tay. Để cài đặt Selenium IDE bạn thực hiện các bước như sau:

**Bước 1:** tải Selenium IDE tại trang web có địa chỉ:

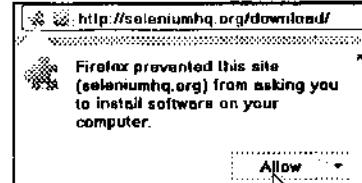
<http://seleniumhq.org/download/>



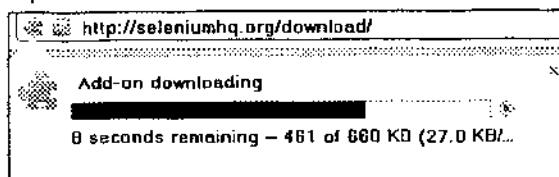
**Bước 2:** giao diện trang web xuất hiện, trong khung Selenium IDE bạn nhấn chọn link "Download version 1.0.12" như hình dưới.



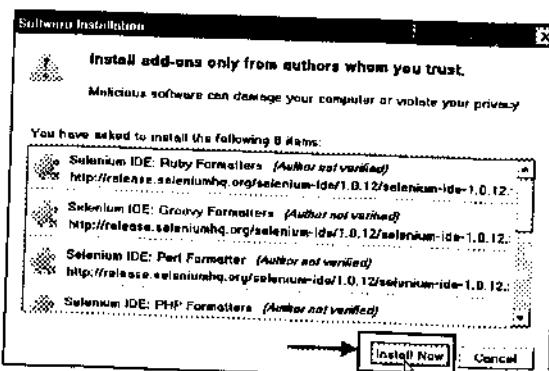
**Bước 3:** một thông báo cho biết Firefox ngăn chặn trang web này ([seleniumhq.org](http://seleniumhq.org)) yêu cầu bạn cài đặt phần mềm trên máy tính của bạn. Để cài đặt, bạn bấm vào nút Allow như hình bên.



**Bước 4:** một cửa sổ pop up xuất hiện thông báo quá trình downloading đang được thực hiện như hình dưới.

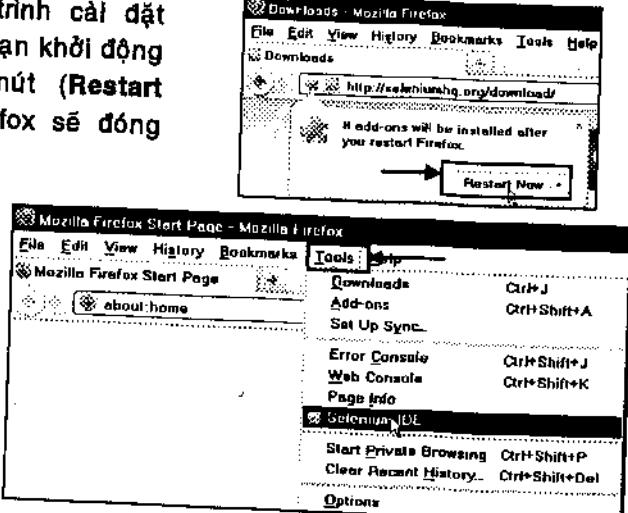


**Bước 5:** sau khi quá trình đếm ngược đã hoàn tất, nút Install sẽ hoạt động và bạn có thể nhấp chuột vào nó. Điều này sẽ cài đặt Selenium IDE như một Firefox add-on như hình trang bên.

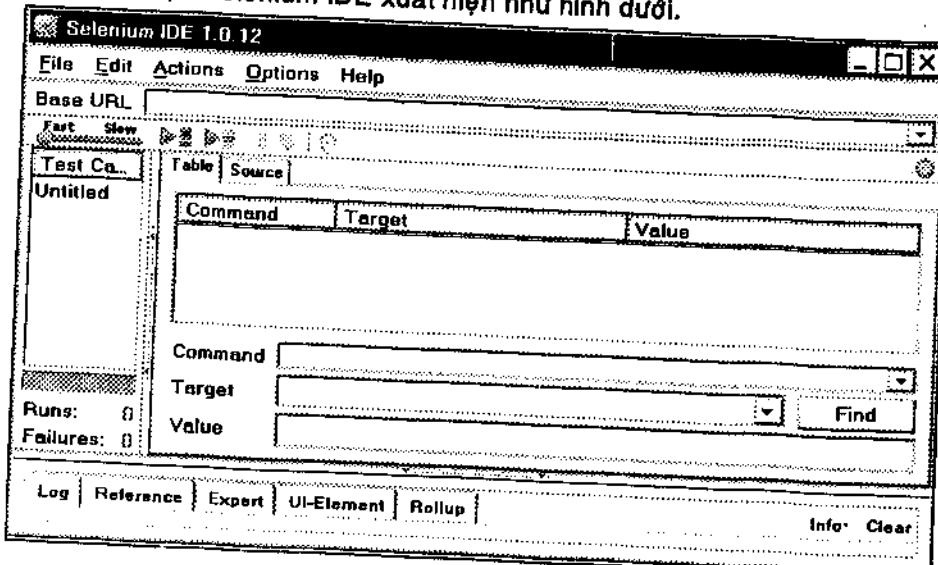


**Bước 6:** khi quá trình cài đặt hoàn tất, nó sẽ yêu cầu bạn khởi động lại Firefox. Nhấp vào nút (Restart Now) khởi động lại. Firefox sẽ đóng cửa và sau đó mở lại.

**Bước 7:** sau khi cài đặt hoàn tất, cửa sổ tiện ích sẽ hiển thị các Selenium IDE. Chọn Tools > Selenium IDE.



Giao diện Selenium IDE xuất hiện như hình dưới.



## 2 HƯỚNG DẪN CÀI ĐẶT SELENIUM RC

Để sử dụng Selenium RC, trước hết cần phải chắc chắn rằng máy tính đã cài sẵn Java JRE. Bạn có thể tải về tại <http://www.softpedia.com>, vì Selenium RC được phát triển trên ngôn ngữ Java nên có thể kiểm tra trên máy có hệ điều hành Mac, Linux, Windows.

### 2.1 Thiết lập proxy điều khiển từ xa.

**Bước 1:** tải về selenium RC từ website <http://seleniumhq.org/download/>

**Bước 2:** giải nén file vào thư mục do bạn đặt tên.

Sử dụng gói **selenium-server-standalone**.

**Bước 3:** mở cửa sổ điều khiển **Command Prompt** trỏ tới tập tin đã được giải nén ở bước 2.

**Bước 4:** chạy lệnh **java -jar selenium-server-standalone.jar** trong cửa sổ command ta có được kết quả như hình dưới.

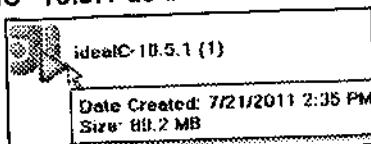
```
Microsoft Visual Studio 2008 Command Prompt - java -jar selenium-server-standalone.jar
C:\Development\selenium2\build>java -jar selenium-server-standalone.jar
22:16:57.893 INFO - Java: Sun Microsystems Inc. 1.4.2-b51
22:16:57.895 INFO - OS: Windows 7 6.1 and64
22:16:57.907 INFO - v2.0 (selenium-standalone) instance should connect to: http://127.0.0.1:4444/hub
22:16:58.026 INFO - Version Jetty/2.5.1.x
22:16:58.072 INFO - Started HttpContext[/selenium-server/WebDriver, /selenium-server]
22:16:58.072 INFO - Started HttpContext[/selenium-server, /selenium-server]
22:16:58.079 INFO - Started HttpContext[/, /]
22:16:58.079 INFO - Started HttpContext[/_/, _]
22:16:58.237 INFO - Started org.openqa.jetty.servlet.ServletHandler@2ca5d5
te
22:16:58.238 INFO - Started HttpContext[/wdm/_/]
22:16:58.245 INFO - Started SocketListener on 0.0.0.0:4444
22:16:58.245 INFO - Started org.openqa.jetty.Server@7721cdef
22:16:58.245 INFO - Started org.openqa.jetty.Listener@1e0a0000
```

### 2.2 Hướng dẫn cài đặt chương trình IDEA IntelliJ.

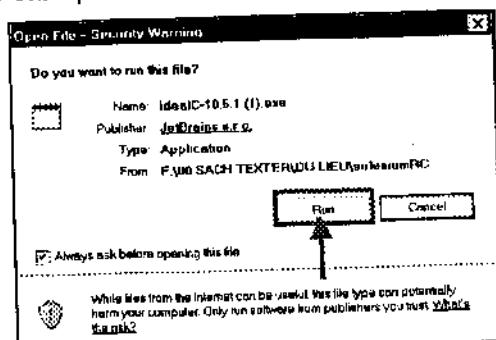
Có thể tải chương trình về từ địa chỉ sau:  
<http://www.jetbrains.com/idea/download/>.

Đây là chương trình hỗ trợ tất cả các công cụ giúp xây dựng một bài kiểm tra thành công. Sau đây là các bước cài đặt.

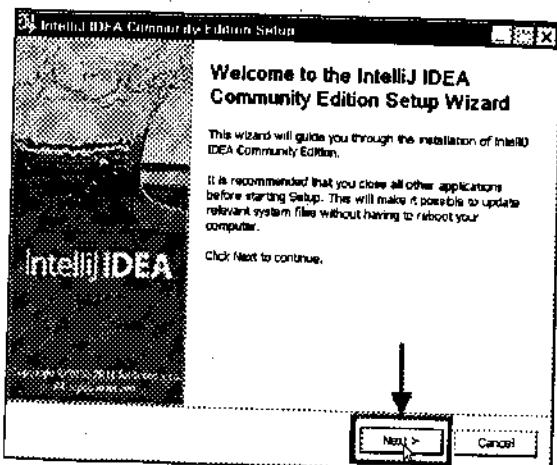
**Bước 1:** nhấp chọn chương trình **IDEAIC-10.5.1** để bắt đầu cài đặt.



**Bước 2:** hộp thoại **Open File – Security Warning** xuất hiện, nhấp chọn **Run** để cài đặt.

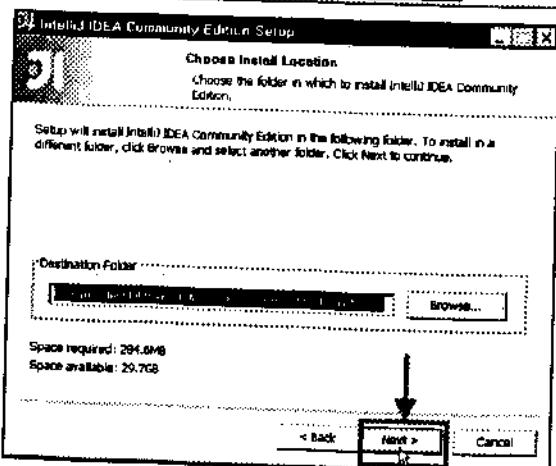


**Bước 3:** cửa sổ **Welcome to the IntelliJ IDEA Community Edition Setup Wizard** xuất hiện, nhấn **Next** để tiếp tục cài đặt.

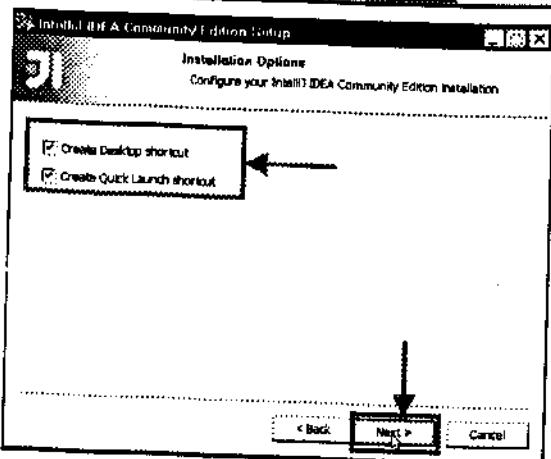


**Bước 4:** cửa sổ **Choose Install Location** xuất hiện, trong khung **Destination Folder** nhấp nút **Browse** để chọn thư mục cài đặt hoặc giữ nguyên đường dẫn mặc định.

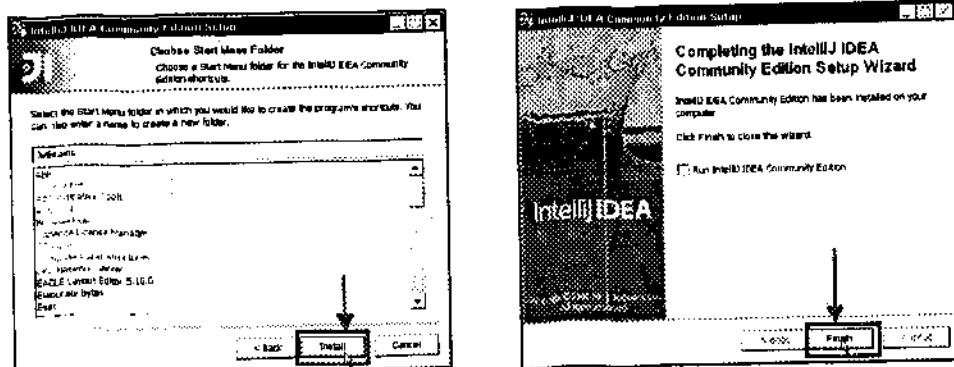
Nhấp **Next** để tiếp tục.



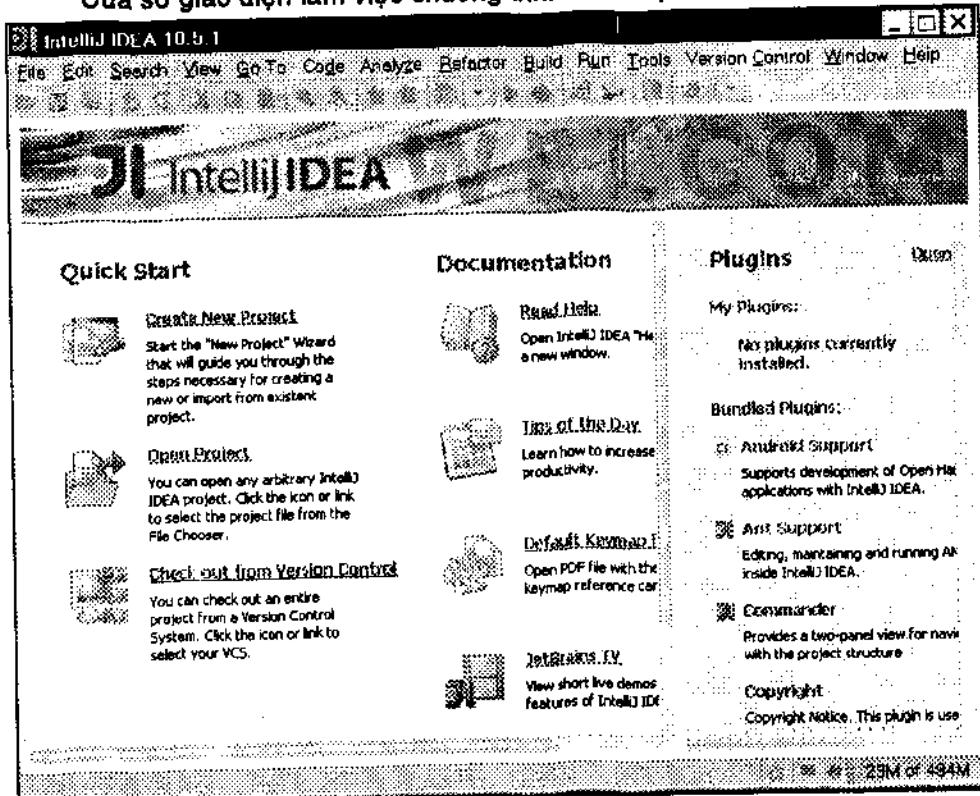
**Bước 5:** cửa sổ **Installation Options** xuất hiện, đánh dấu kiểm vào **Create Desktop shortcut** và **Create Quick Launch shortcut** sau đó nhấp **Next** để tiếp tục cài đặt.



**Bước 6:** cửa sổ **Choose Start Menu Folder** xuất hiện, nhấn **Install** để tiếp tục cài đặt, sau đó nhấn **Finish** trong cửa sổ mới để kết thúc.



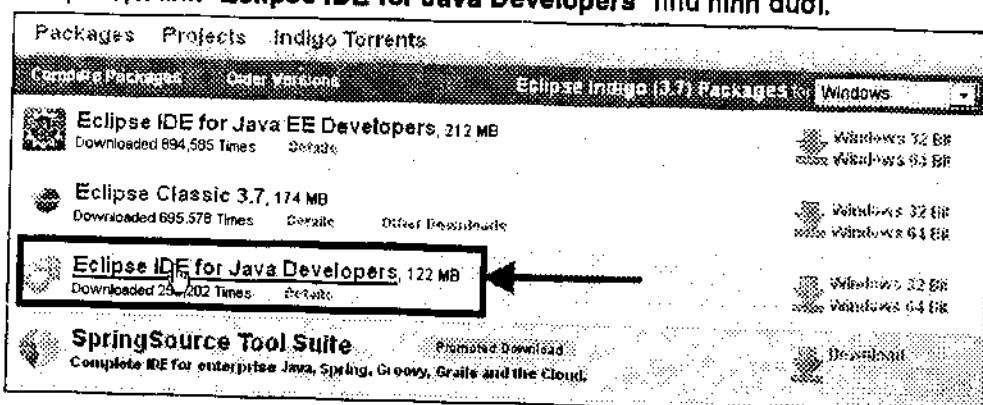
Cửa sổ giao diện làm việc chương trình xuất hiện như hình dưới.



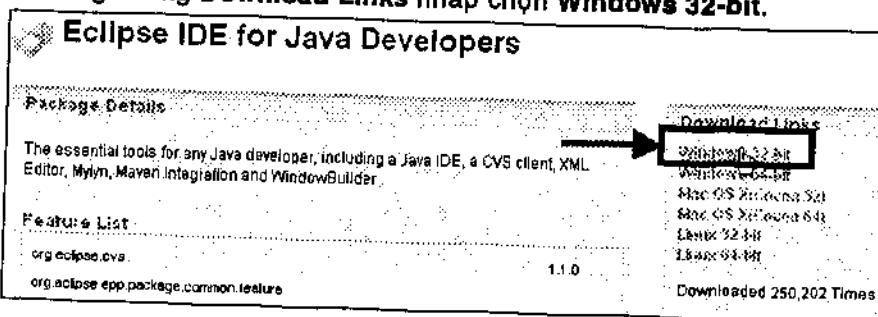
### 3 HƯỚNG DẪN CÀI ĐẶT ECLIPSE

Eclipse là một IDE đa ngôn ngữ được viết trên Java với một hệ thống plug-in mở rộng cho phép người dùng bổ sung nhiều tính năng. Eclipse là một trong những công cụ phát triển phần mềm mạnh mẽ hiện nay. Thực ra Eclipse là một môi trường tiêu chuẩn cho phát triển mã nguồn mở. Điểm mạnh nhất của Eclipse là chức năng Plug-in, Eclipse có thể sử dụng ngôn ngữ lập trình để tạo ra 58 loại Plug-in khác nhau.

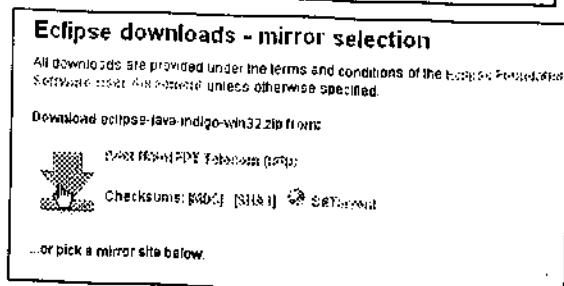
Bạn vào đường link "<http://www.eclipse.org>" để tải Eclipse về máy. Nhập chọn link "Eclipse IDE for Java Developers" như hình dưới.



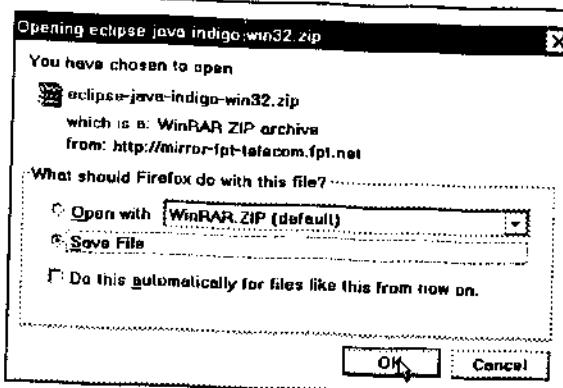
Trong khung Download Links nhấp chọn Windows 32-bit.



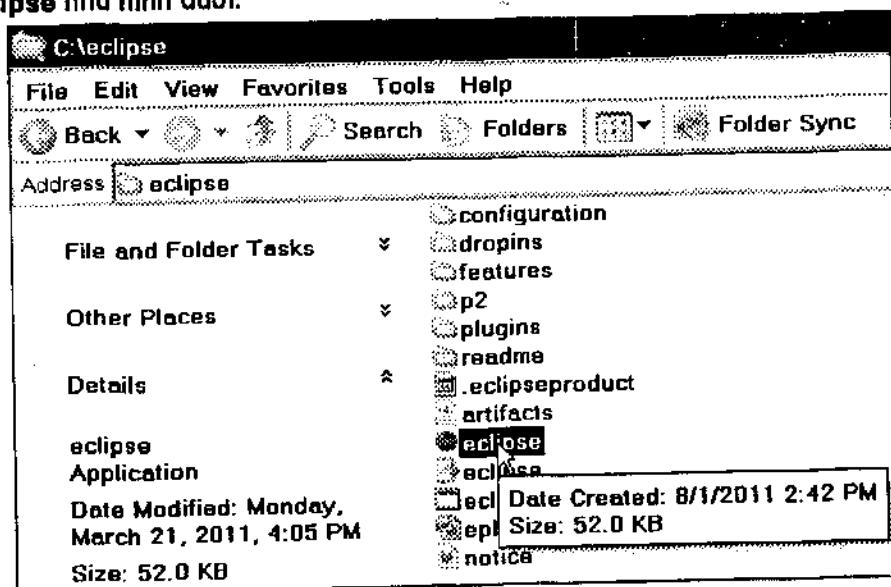
Trong phần Eclipse downloads – mirror selection, nhấp chọn biểu tượng mũi tên màu xanh để tải chương trình về máy.



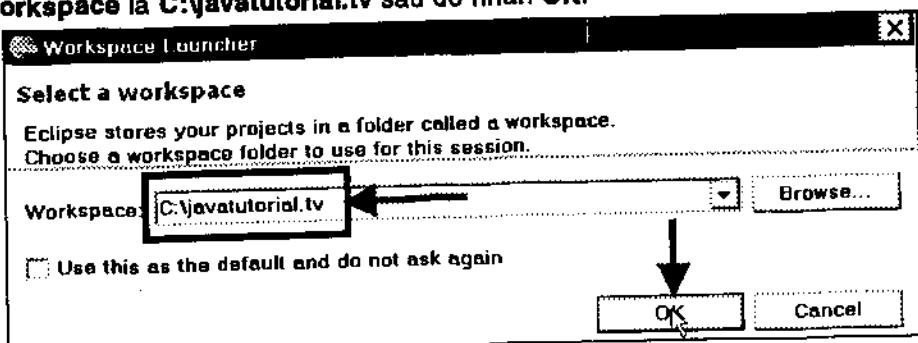
Hộp thoại Opening eclipse-jee-indigo-win-32.zip xuất hiện, nhấp chọn Save File sau đó nhấp OK.



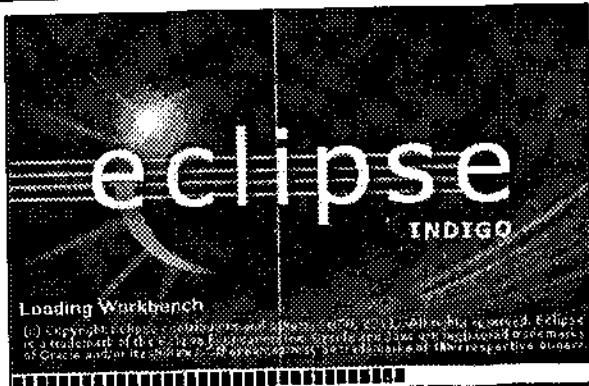
Sau khi tải file nén này về, bạn giải nén vào ổ C của máy và nhấn chọn **eclipse** như hình dưới.



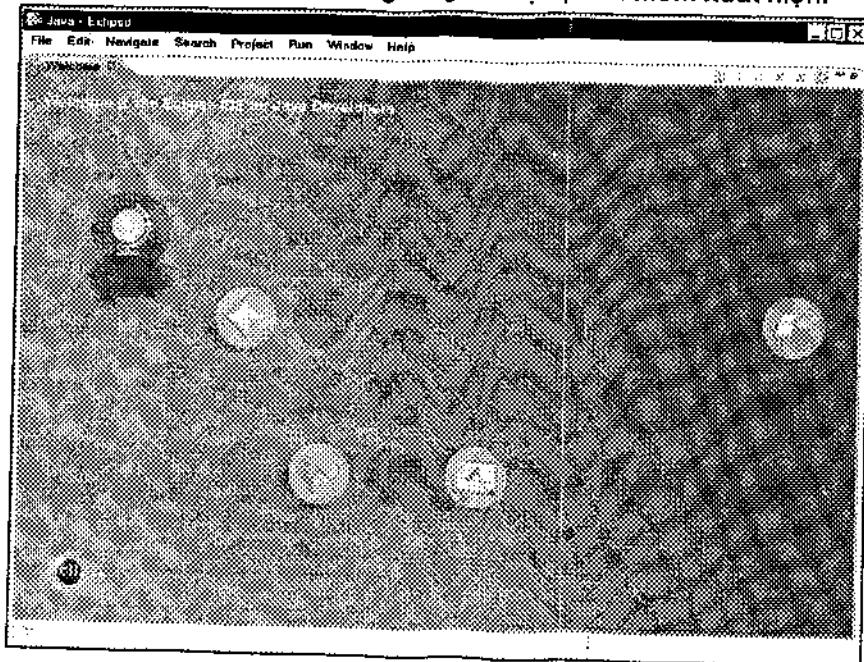
Hộp thoại **Workspace Launcher** xuất hiện, điền nội dung tại dòng **Workspace** là **C:\javatutorial.tv** sau đó nhấn **OK**.



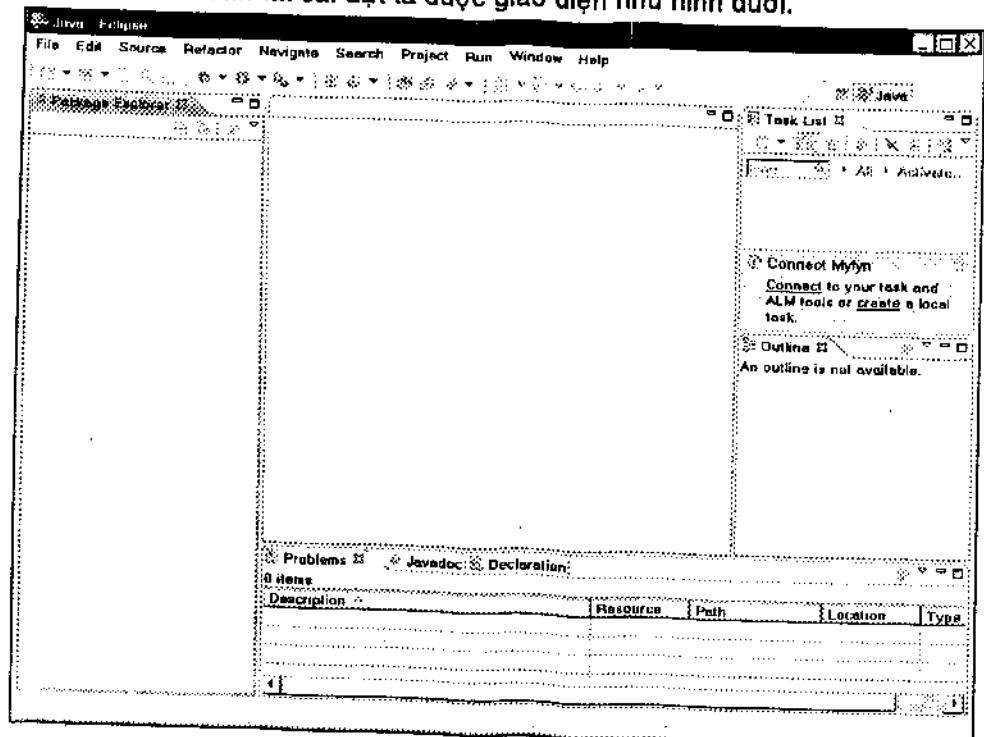
Chương trình  
Eclipse được khởi  
động như hình bên.



Cửa sổ Welcome chúc mừng và giới thiệu phần mềm xuất hiện.



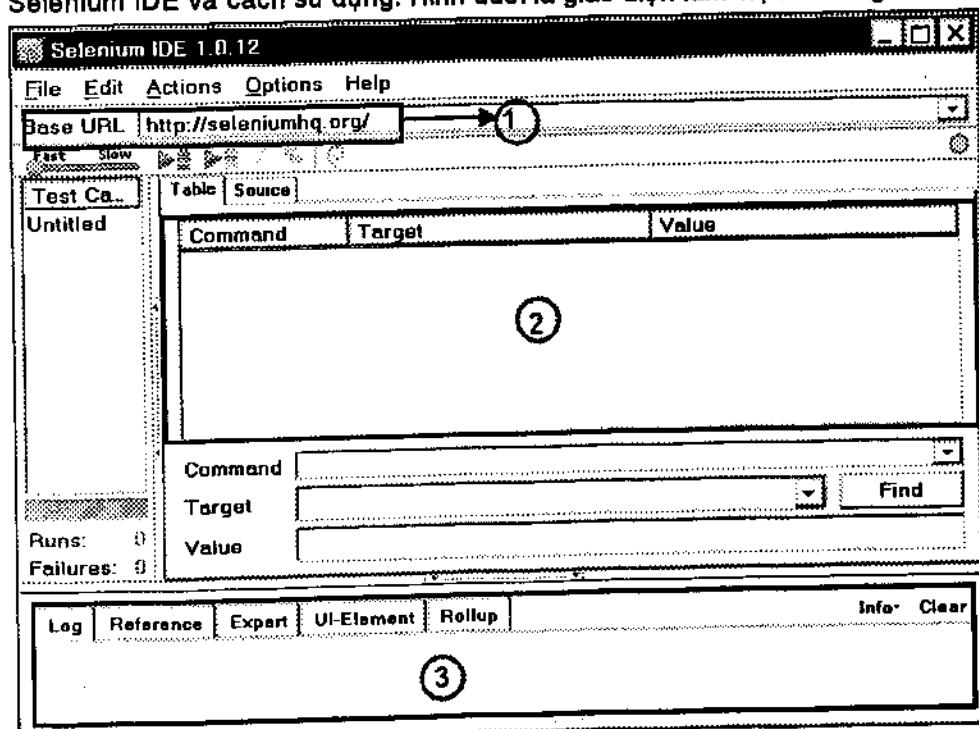
Sau khi hoàn tất cài đặt ta được giao diện như hình dưới.



## CHƯƠNG 15

# KIỂM THỬ WEBSITE VỚI SELENIUM IDE

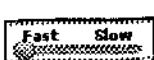
Chương 15 hướng dẫn các bạn làm quen với giao diện, các công cụ Selenium IDE và cách sử dụng. Hình dưới là giao diện làm việc chương trình.



1 – Uri cho ứng dụng web cần test.

2 – Danh sách các actions trong Test Case để thực hiện.

3 – Nơi thể hiện các sự kiện đã thực hiện bao gồm các lỗi và cảnh báo.



Điều khiển tốc độ test.



Chạy tất cả các sự kiện cần kiểm tra trên IDE.



Chạy thử nghiệm một sự kiện trên IDE.



Tạm dừng một thử nghiệm hiện đang chạy.



Bước qua một kiểm tra khi nó đã bị tạm dừng.



Nút ghi nhận hoạt động test.

**Thể hiện những dòng lệnh, bao gồm kết quả mong đợi.**

Command	<input type="text"/>
Target	<input type="text"/> <input type="button" value="Find"/>
Value	<input type="text"/>

Command	<input type="text" value="addLocationStr..."/>
	<input type="button"/>
	<input type="button"/>
	<input type="button"/>

Bao gồm danh sách các lệnh có sẵn cần thiết để tạo ra một thử nghiệm.

Target	<input type="text"/> <input type="button"/>
--------	---

Hộp văn bản cho phép nhập vào vị trí các yếu tố cần tương tác.

Find

Nhấp vào nút này để làm nổi bật ví trí cần tương tác có trên file.

Value

Giá trị textbox là nơi bạn đặt giá trị cần thay đổi.

Tham khảo các thể hiện sự kiện đã chọn trong command.

Log	Reference	Expert	UI-Element	Rollup	Info	Clear

❖ Các chức năng trong File:

- + Tạo Test-Case và Test-Suite mới.
- + Mở Test-Case và Test-Suite đã lưu.
- + Lưu Test-Case và Test-Suite theo định dạng html.
- + Xuất (Export) Test-Case và Test-Suite theo các định dạng mà Selenium hỗ trợ.
- + Thêm Test-Case.
- + Thoát khỏi chương trình.

❖ Các chức năng trong Edit:

- + Undo, Redo: thực hiện lại.
- + Cut, Copy, Paste, Delete: cắt, copy, dán, xóa.
- + Select All: chọn tất cả các command.

❖ Các chức năng trong Options:

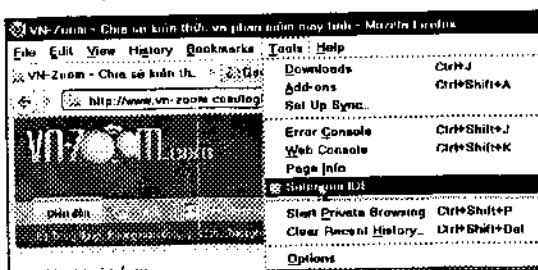
- + Options: lựa chọn một số tính năng như encoding của file, chọn lựu phần mở rộng của Selenium IDE.
- + Format: chọn định dạng của nguồn test-script.
- + Clipboard Format: chọn định dạng của Clipboard.
- ❖ Các chức năng trong Help:
- + Các thông tin tài liệu về Selenium IDE.

Để bắt đầu làm quen với chương trình kiểm thử Selenium IDE, bạn hãy thực hiện tiến trình kiểm thử đơn giản với các bước như sau:

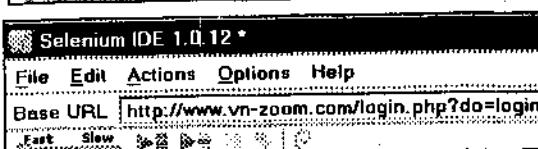
## 1 ỨNG DỤNG SELENIUM IDE TRONG MỘT TEST FORM ĐƠN GIẢN

Bạn thử test form đăng nhập cho một diễn đàn bất kì như hình dưới. Ví dụ ở đây chọn trang web của diễn đàn http://www.vn-zoom.com.

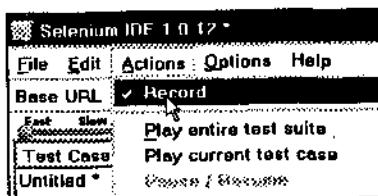
**Bước 1:** nhấp chọn Tools > Selenium IDE như hình bên.



Cửa sổ giao diện làm việc xuất hiện như hình bên:



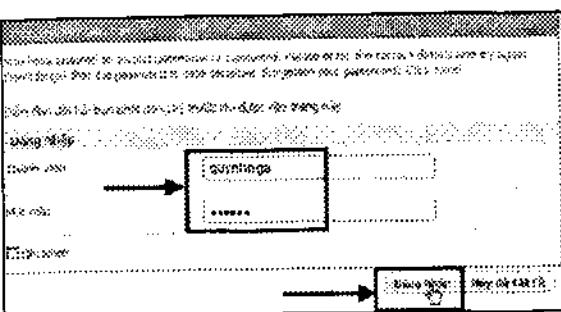
**Bước 2:** nhấp biểu tượng Record để ghi nhận lại các bước đã thực hiện trên form đăng nhập vào diễn đàn.



**Bước 3:** đăng nhập vào diễn đàn với tài khoản chưa đăng ký.

Thành viên: **quynhnga**

Mật mã: **123456**



Khi đăng nhập sai, gọi lại form đăng nhập cho phép bạn nhập lại thông tin tài khoản. Quay trở về giao diện Selenium IDE, trong tab Table xuất hiện các giá trị tương ứng ở Command, Target, Value.

Command	Target	Value
open	/login.php?do=login	
type	vb_login_username	quynhnhage
type	vb_login_password	123456
clickAndWait	name=input.button	

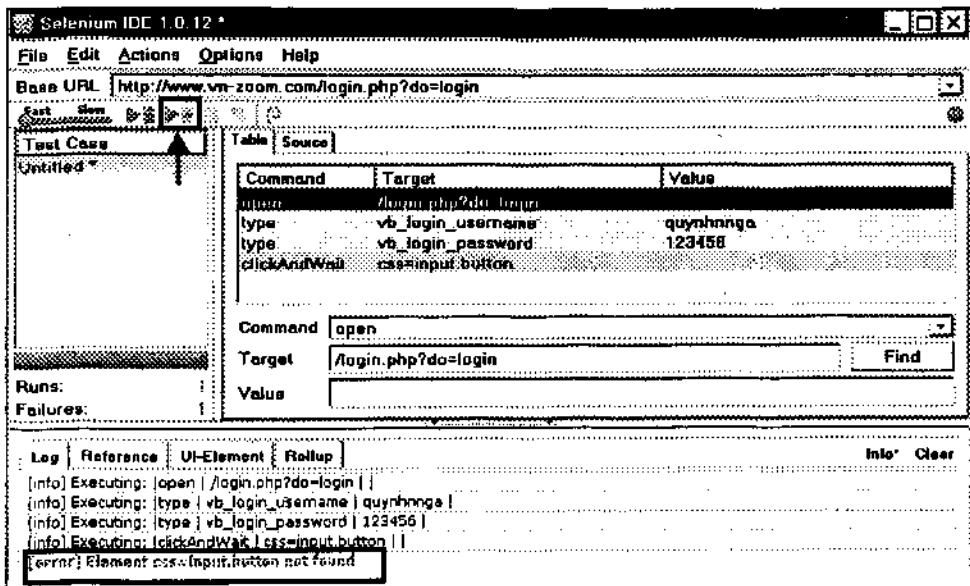
Bước 4: nhấp chọn biểu tượng kiểm tra toàn bộ trang web. Khi đó trong khung Test Case sẽ xuất hiện 2 giá trị với 2 màu khác nhau.

- + **Runs** (số lần chạy kiểm tra): hiển thị với giá trị bằng 1 được bôi thành màu xanh.
- + **Failures** (số lỗi có trong trang web): hiển thị với giá trị bằng 1 được bôi thành màu đỏ.

**Chú ý:** trong tab Table lúc này hiển thị lỗi tương ứng mà nó tìm được. Chương trình dừng ở dòng clickAndWait (được bôi màu gạch).

Runs:	1
Failures:	2

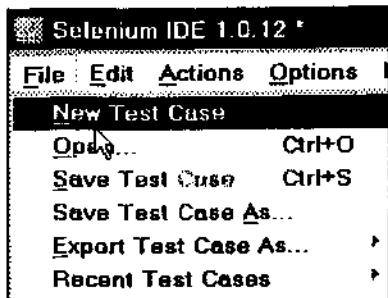
Bước 5: tiến hành test từng dòng có trên tab Table. Ví dụ: nhấp chọn dòng open và bấm vào biểu tượng cho phép text từng dòng. Kết quả thông báo được hiển thị trong tab Log như hình dưới.



Tương tự, bạn thực hiện test với một tài khoản đã đăng kí.

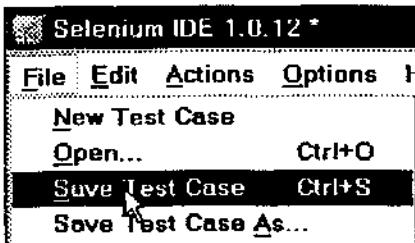
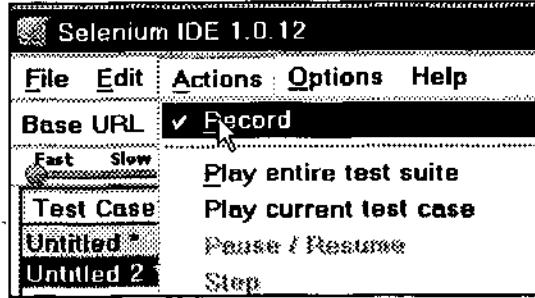
Nếu đăng nhập vào diễn đàn với một tài khoản đã được đăng kí sẽ xuất hiện form thông báo đăng nhập thành công.

**Bước 1:** trong giao diện Selenium IDE 1.0.12, chọn menu File > New Test Case để tạo một cuộc kiểm tra mới.



**Bước 3:** tiến hành test từng sự kiện, chọn menu File > Save Test Case để lưu lại kết quả dưới định dạng file php, java...

**Bước 2:** nhấp biểu tượng Record để ghi nhận lại các bước đã thực hiện trên form đăng nhập vào diễn đàn với tài khoản đăng kí.



Nếu muốn thêm sự kiện (command) của mình để kiểm tra trên Selenium IDE, bạn thực hiện các bước sau:

**Bước 1:** phải đảm bảo rằng quá trình kiểm tra đang tiến hành liên tục với các bước kiểm tra trước đó.

Command	Target	Value
open	/login.php?do=login	
type	vb_login_username	quynhng
type	vb_login_password	123456
clickAndWait	css=input.button	

Command: open  
Target: /login.php?do=login  
Value:

Run: 1  
Failures: 1

Log | Reference | UI-Element | Rollup

**Bước 2:** nhấp chuột phải vào sự kiện **open** > **Insert New Comment**.

- Table | Source |
- Command Target Value
- open /login.php?do=login
- type vb\_login\_username quynhng
- type vb\_login\_password 123456
- clickAndWait css=input.button
- Ctrl+X Ctrl+C Ctrl+V Del
- Insert New Command
- Insert New Comment**
- Clear All

**Bước 3:** một vùng không gian trống xuất hiện nằm giữa các dòng Selenium lệnh có trong Table.

Command	Target	Value
open	/login.php?do=login	
type	vb_login_username	quynhng
type	vb_login_password	123456
clickAndWait	css=input.button	

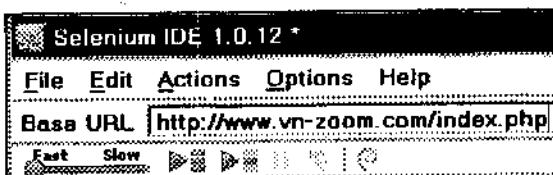
Nhấp chuột vào hộp văn bản và nhập vào nội dung chú thích để bạn có thể sử dụng nó trong tương lai.

Command type1  
Target  
Value

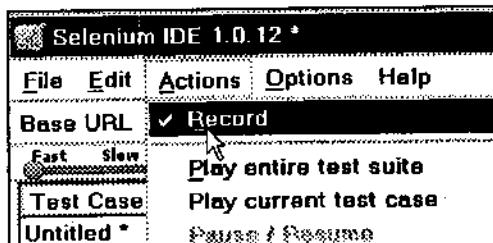
## 2 ỨNG DỤNG SELENIUM IDE TRÊN NHIỀU CỬA SỔ CỦA 1 WEBSITE.

Làm việc với nhiều cửa sổ trình duyệt có thể là một trong những điều khó khăn nhất để làm trong một thử nghiệm với Selenium. Trong ví dụ tiếp theo, bạn sẽ nhấp vào một yếu tố trên trang để mở ra một cửa sổ mới sau đó tương tác trên cửa sổ con và thực hiện kiểm tra trên cửa sổ cha.

**Bước 1:** khởi động Selenium IDE.



**Bước 2:** chọn Actions > Record để khởi động công cụ ghi lại các bước tương tác trên trình duyệt Firefox.



**Bước 3:** nhấp vào một yếu tố trên trang web để mở ra một cửa sổ mới.

Ví dụ nhấp vào menu Tá lả.

Quay trở về giao diện Selenium IDE, trong tab Table sẽ ghi lại các hành động bạn vừa mới tương tác như hình bên.

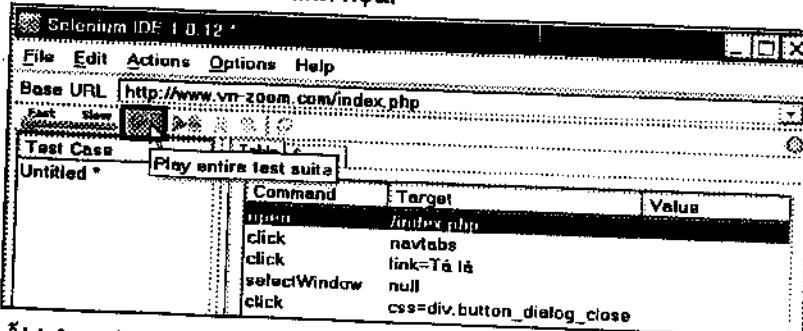


Selenium IDE 1.0.12 *																	
File Edit Actions Options Help																	
Base URL http://www.vn-zoom.com/index.php																	
	Test Case	Source															
Untitled *		<table border="1"> <thead> <tr> <th>Command</th><th>Target</th><th>Value</th></tr> </thead> <tbody> <tr> <td>open</td><td>/index.php</td><td></td></tr> <tr> <td>click</td><td>navtabs</td><td></td></tr> <tr> <td>click</td><td>link=Tá lả</td><td></td></tr> <tr> <td>click</td><td>link=Tá lả</td><td></td></tr> </tbody> </table>	Command	Target	Value	open	/index.php		click	navtabs		click	link=Tá lả		click	link=Tá lả	
Command	Target	Value															
open	/index.php																
click	navtabs																
click	link=Tá lả																
click	link=Tá lả																

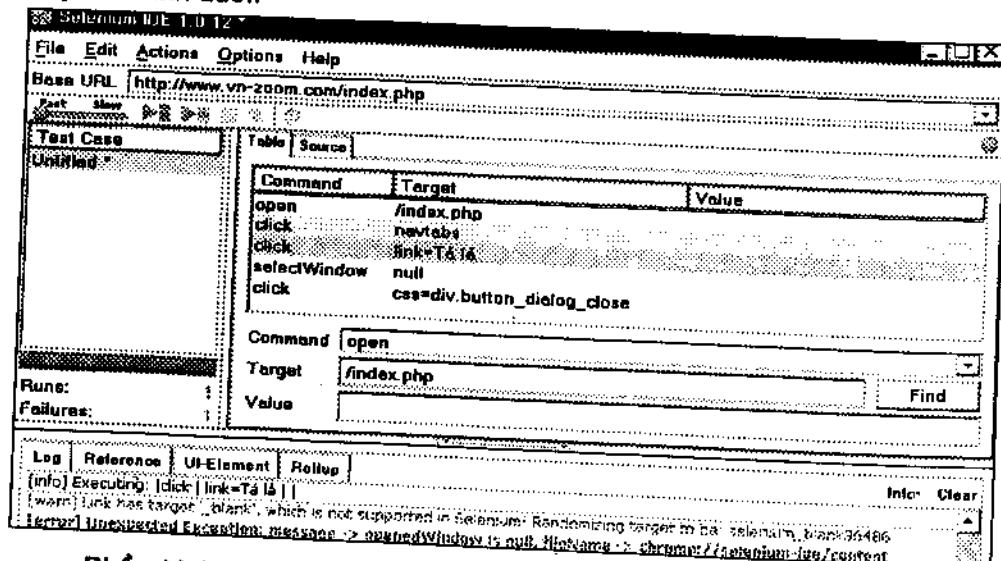
**Bước 4:** trên cửa sổ vừa mới được mở, bạn nhấp chọn một chức năng bất kỳ. Khi đó trong Table sẽ ghi lại hành động kế tiếp bạn vừa mới tương tác.

Table   Source		
Command	Target	Value
open	/index.php	
click	navtabs	
click	link=Tá lả	
selectWindow	null	
click	css=div.button_dialog_close	

**Bước 5:** đóng tất cả các cửa sổ liên kết và tiến hành kiểm tra các yếu tố đó trên cửa sổ chính. Nhấp chọn biểu tượng Play entire test suite để kiểm tra toàn bộ trang web như hình minh họa.

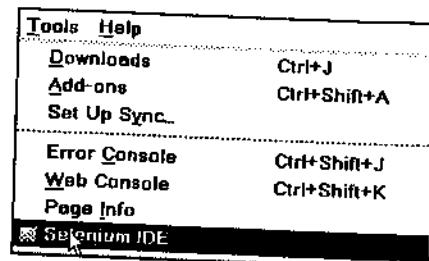


Lỗi trên các sự kiện của cửa sổ con sẽ được kiểm tra, quá trình kiểm tra diễn ra và dừng lại ở command click, target link=Tá lả. Giao diện làm việc lúc này như hình dưới.

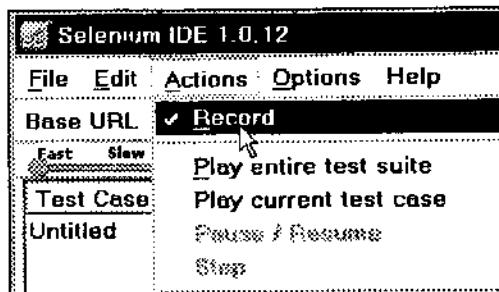


Phần trình bày sau trình bày cách kiểm tra các tương tác trên cửa sổ con, từ đó biết được khoảng thời gian chuyển đổi giữa các cửa sổ khi quá trình kiểm tra diễn ra.

**Bước 1:** khởi động trang web, từ menu Tools nhấp chọn Selenium IDE.



**Bước 2:** giao diện làm việc Selenium IDE xuất hiện, nhấp chọn Actions > Record để ghi nhận các bước thực hiện trên web.



**Bước 3:** nhấp chọn một liên kết để mở một cửa sổ mới và tương tác trên trang đó bằng cách chọn Open Link in New Tab.

**Bước 4:** quay trở lại trang chính, nhấp vào liên kết để mở cửa sổ thứ 2 và tương tác trên nó bằng cách chọn Open Link in New Tab.

Lúc này trên giao diện Selenium IDE ghi nhận lại các bước như hình:

Command	Target	Value
type	regusername	modquynhng
click	email	
click	email	
type	email	nga@gmail.com
click	emailconfirm	
click	emailconfirm	
type	recaptcha_response_field	nga@gmail.com
type	recaptcha_response_field	0.828 roceppr
click	cfield_B	
select	cfield_B	label=Việt Nam
click	css=#cfield_B > option[value=1]	
clickAndWait	css=input.button	
click	css=div.body_wrapper	

Run: 0  
Failures: 0

**Bước 5:** di chuyển cửa sổ được mở đầu tiên và đóng nó lại.

Ví dụ: bạn đang ở cửa sổ hiển thị bài viết, bây giờ bạn nhấp chọn một liên kết khác như tìm kiếm.

Sau khi cửa sổ này hiện lên, bạn sẽ đóng nó lại.

**Bước 6:** di chuyển cửa sổ thứ hai và sau đó đóng cửa sổ này lại.

**Bước 7:** di chuyển lại cửa sổ chính để xác minh một yếu tố trên trang đó. Ví dụ: trang đăng ký được mở trực tiếp trên cửa sổ chính thay vì chọn Open Link in New Tab.

**Bước 8:** chạy thử nghiệm trên cửa sổ chính để xem cách nó di chuyển giữa các cửa sổ với nhau.

Nhấp chọn biểu tượng cho phép kiểm tra toàn bộ thao tác trên trang web. Nó chạy qua cửa sổ 1 và đến cửa sổ thứ 2 thì bị dừng vì phát hiện ra lỗi pass đăng ký vào hệ thống.

Bạn chú ý đến dòng [error] Timed out after 30000ms, đây là thời gian chuyển giữa các cửa sổ trong trang web.

Giao diện sau khi kiểm tra như hình dưới.

Command	Target	Value
open	/	
type	password	123
click	passwordconfirm	
type	passwordconfirm	123
click	email	
click	regusername	
type	regusername	quynhngai
click	css=div.blockrow > p.description	
click	//form[@id='registerForm']/div/div..	
click	regusername	
type	regusername	quynhngakhoa
click	passwordconfirm	
type	regusername	modauvnhn
click	regusername	

Logs:

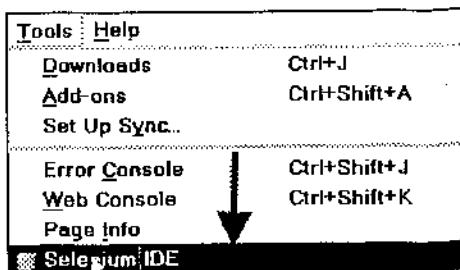
- [info] Executing: open //
- [error] Timed out after 30000ms
- [info] Executing: type | password | 123
- {error} Element password not found

### 3 SELENIUM KIỂM TRA VỚI CÁC ỨNG DỤNG AJAX

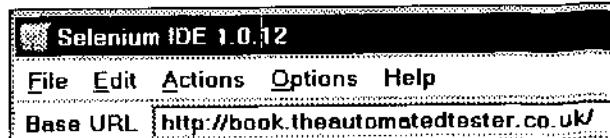
Phần trình bày sau thực hiện kiểm tra các trang web có sử dụng các ứng dụng viết bằng AJAX. (viết tắt từ Asynchronous JavaScript và XML).

**Bước 1:** chạy thử một trang web có sử dụng chương trình ứng dụng viết bằng AJAX.

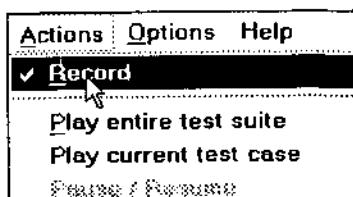
**Bước 2:** chọn menu Tools > Selenium IDE để khởi động chương trình kiểm thử.



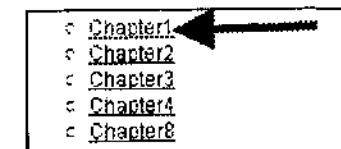
**Bước 3:** dán địa chỉ trang web cần kiểm tra vào khung Base URL.



**Bước 4:** chọn menu Actions > Record để tiến hành ghi lại các thao tác trên web.

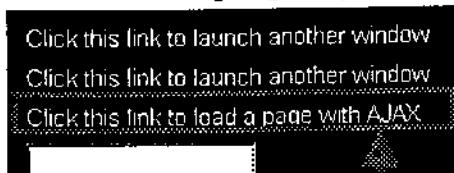


**Bước 5:** nhấp đúp chuột vào link chapter1 để mở một cửa sổ mới và thực hiện các thao tác kiểm tra trên nó.



Nhấp chọn vào đường link Click this link to load a page with AJAX để mở trang ứng dụng AJAX.

The following text has been loaded from another page on this site. It has been loaded in an asynchronous fashion so that we can work through the AJAX section of this chapter



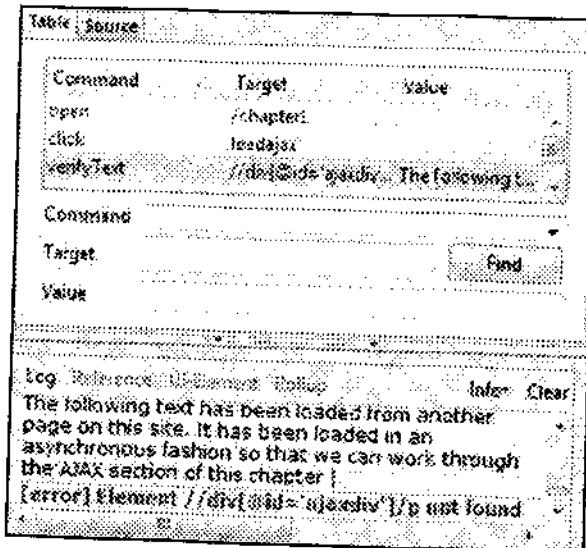
Click this link to launch another window  
Click this link to launch another window  
Click this link to load a page with AJAX

Một khung text xuất hiện như hình bên.

Mọi sự kiện diễn ra trên website đều được lưu lại trong tab Table.

Table	Source	Target	Value
Command	open	Target	/chapter1
click		Value	loadAjax
verifyText		Value	//div[@id='ajaxDiv... The following text ...

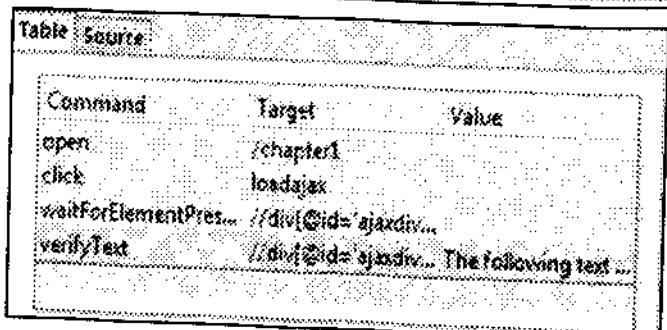
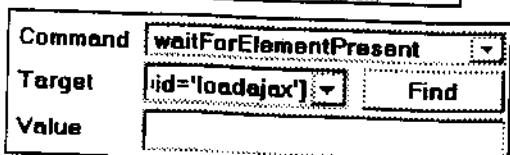
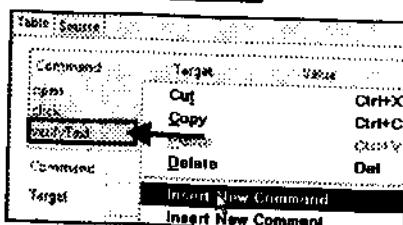
**Bước 6:** nhấp chọn biểu tượng cho phép kiểm tra toàn bộ thao tác trên trang web. Thông báo lỗi sẽ được đưa ra trong khung Log và trình kiểm tra dừng lại. Khi nhìn vào mục thông báo lỗi, bạn sẽ thấy văn bản đang chờ được xử lý. Vì khi quá trình kiểm tra diễn ra, văn bản không được nạp vào các DOM. Điều này cũng dễ hiểu, vì trước đó nó đã được yêu cầu trả lại dữ liệu từ máy chủ về trình duyệt.



Để khắc phục lỗi, bạn thêm một lệnh mới để kiểm tra hệ thống. Bạn nhấp vào một sự kiện trước khi phát sinh lỗi sau đó chọn Insert New Command.

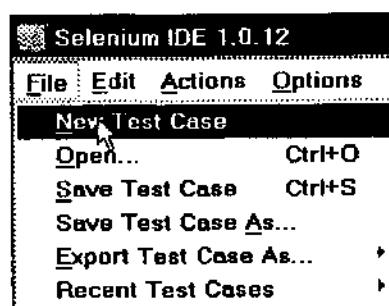
Nhấp chọn vào vị trí đó, trong khung Command nhấp chọn loại waitForElementPresent trong combobox với Target tương ứng.

Kiểm tra một lần nữa để thu được kết quả như hình bên.

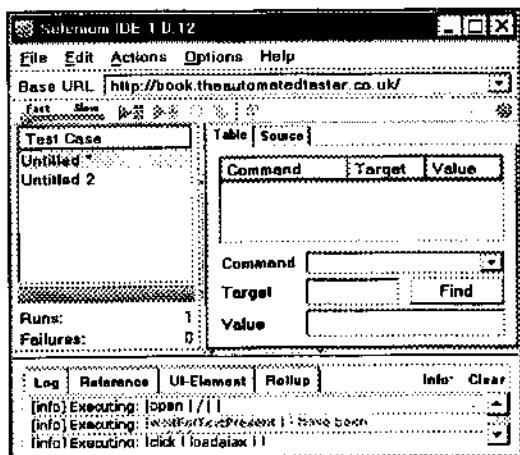
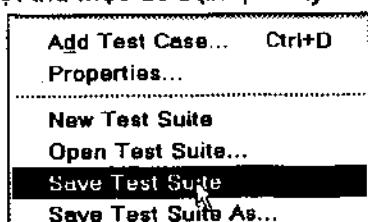


## 4 TẠO MỚI FILE KIỂM TRA

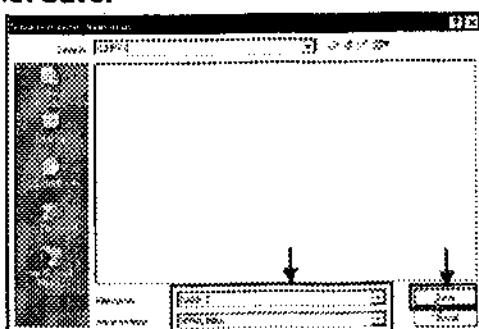
Bước 1: để tạo mới một dãy các phòng kiểm tra, từ giao diện Selenium IDE được mở trước đó chọn File > New Test Case.



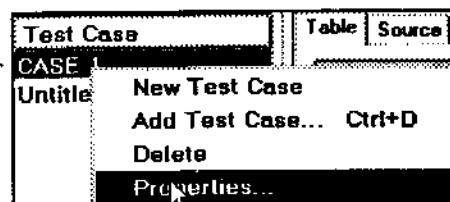
Selenium IDE mở một khu vực làm việc mới. Để lưu file này, chọn File > Save Test Suite hoặc Save Test Suite As... để lưu vào một thư mục do bạn quản lý.



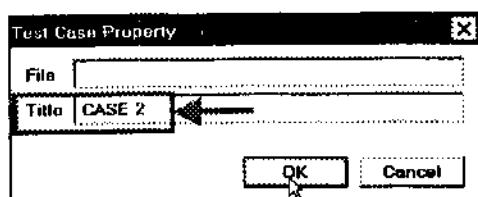
Đặt tên file, chọn kiểu lưu và nhấp nút Save.



Để thay đổi tên trực tiếp Test Case, bạn nhấp chuột phải vào Test Case cần đổi sau đó chọn Properties.



Hộp thoại Test Case Property xuất hiện, điền tên sau đó nhấp OK.

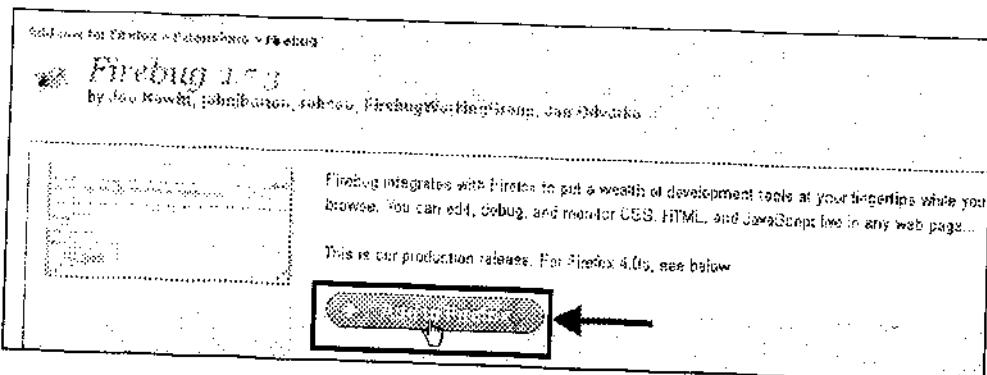


## 5 ĐỊNH VỊ CÁC YẾU TỐ TRÊN TRANG WEB

Phần trình bày sau giới thiệu bạn cách xác định các yếu tố trên trang web bởi ID, tên, liên kết, xpath, và CSS.

Bạn cần cài đặt chương trình Firebug 1.7.3 vào firefox theo địa chỉ tải về sau: <https://addons.mozilla.org/en-US/firefox/addon/firefinder-for-firebug/>.

Đây là chương trình giúp cho các nhà phát triển web tìm kiếm các yếu tố trên trang web bằng chức năng tìm kiếm.



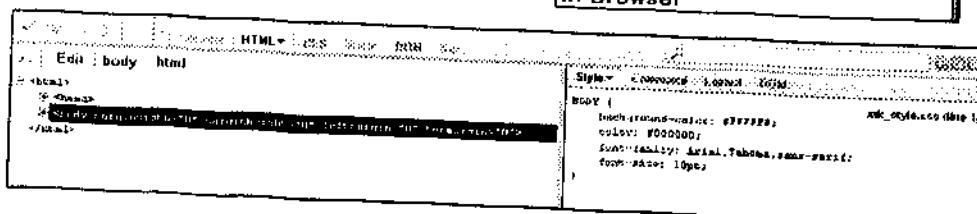
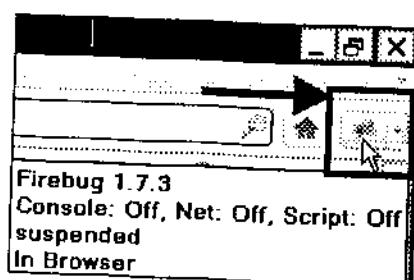
### 5.1 Xác định vị trí các yếu tố bởi ID

Trong tất cả các ứng dụng web hiện nay, các yếu tố sẽ có một ID cho tất cả các nút. Nhờ vậy Selenium có thể tìm một mục duy nhất (giống như tìm ID duy nhất) để hoàn thành các hành động cần thực hiện.

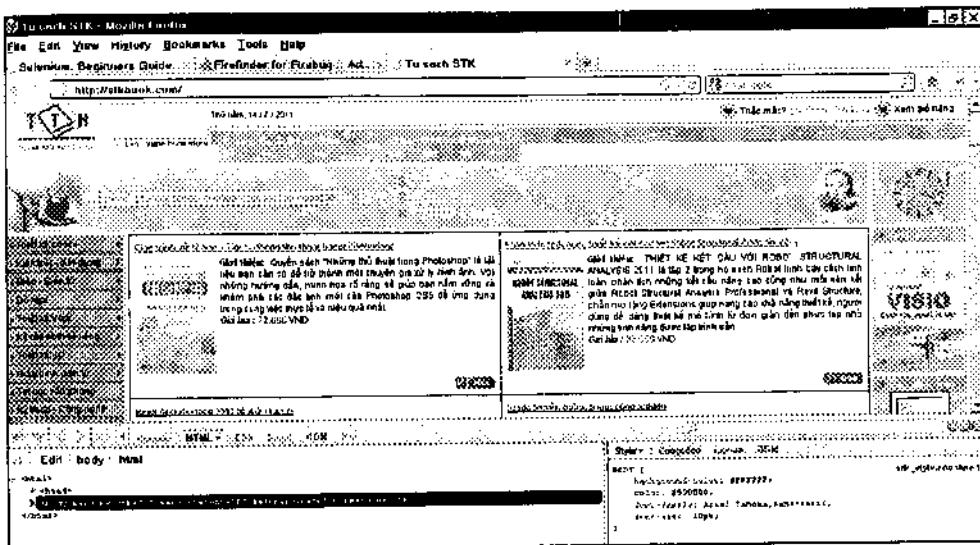
Các bạn theo các bước hướng dẫn sau để tìm yếu tố trên trang bằng ID với Firebug.

**Bước 1:** nhấp vào biểu tượng Firebug nằm ở góc phải, trên cùng của trình duyệt Firefox.

Đây là khung hiển thị các thao tác khi bạn di chuột trên trang web đang thực thi.



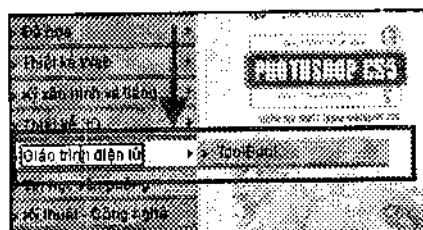
Hình trang bên là giao diện bạn thấy khi kiểm tra trang web của nhà sách STK tại địa chỉ <http://stkbook.com>.



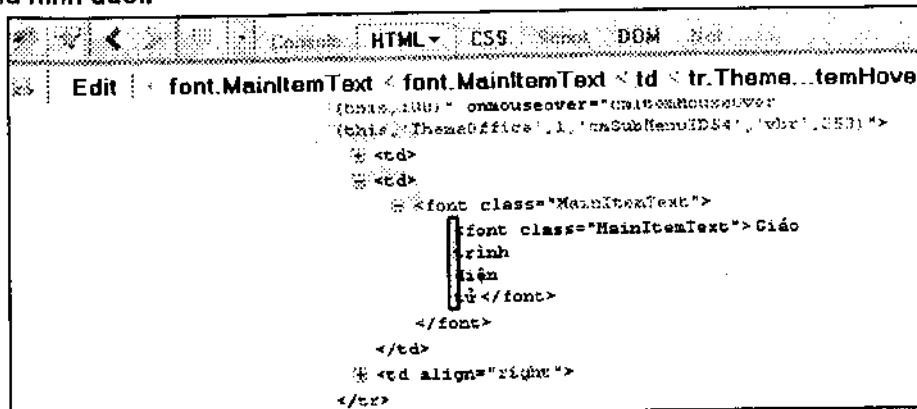
**Bước 2:** nhấp vào biểu tượng hình chữ nhật có hình mũi tên ở phía bên trái giao diện để kiểm tra một phần tử trên trang.

**Chú ý:** khi bạn đưa trỏ chuột tới một vị trí bất kì trên trang web, nó sẽ được đánh dấu.

Ví dụ trỏ chuột vào danh mục **Giáo trình điện tử**.



Lúc này trong khu vực của Firebug sẽ nổi bật các yếu tố bạn cần xem như hình dưới.



Làm nổi bật định dạng của nó trong file CSS.

```

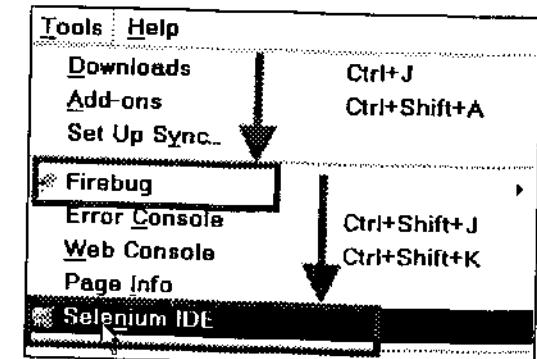
Style Computed Layout DOM
.MainItemText {
    color: #000000;
    font-family: Arial,Tahoma,Verdana;
    font-size: 9pt;
    text-decoration: none;
}
Inherited from font.MainItemText
.MainItemText {
    color: #000000;
    font-family: Arial,Tahoma,Verdana;
    font-size: 9pt;
    text-decoration: none;
}

```

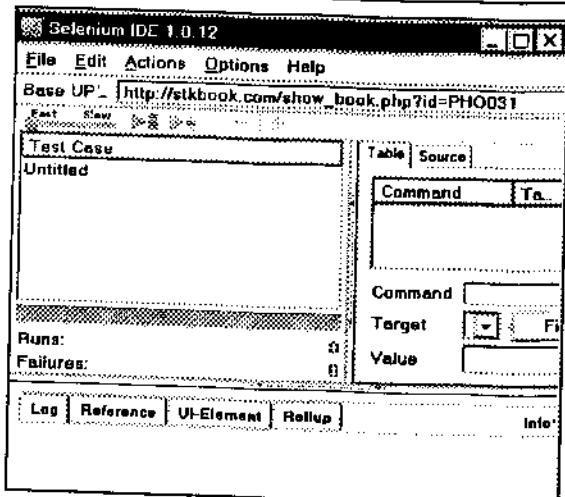
Để có thể nhìn thấy các thuộc tính đó trên Selenium bạn cần thực hiện các bước như sau:

#### Bước 1:

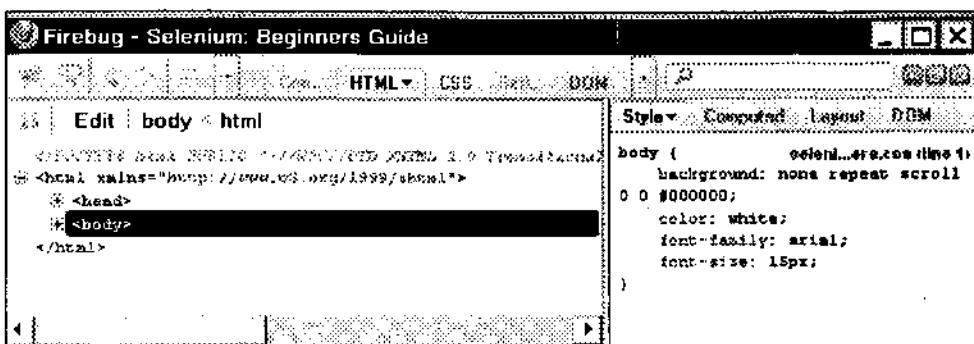
Trong trình duyệt trang web <http://book.theautomatedtester.co.uk/>, chọn menu Tools sau đó khởi động Selenium IDE và Firebug.



Giao diện Selenium IDE xuất hiện như hình bên.

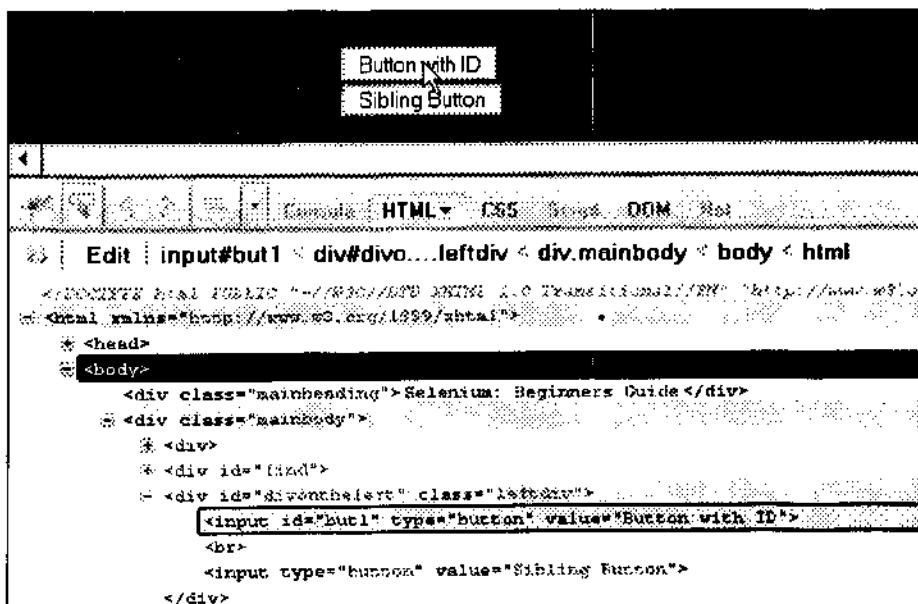
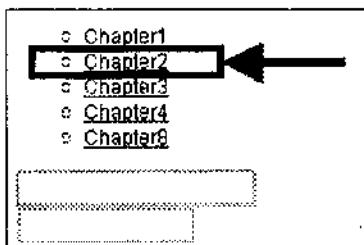


Hình trang bên là giao diện Firebug khi mở trên một cửa sổ mới tách rời trang web cần kiểm tra.



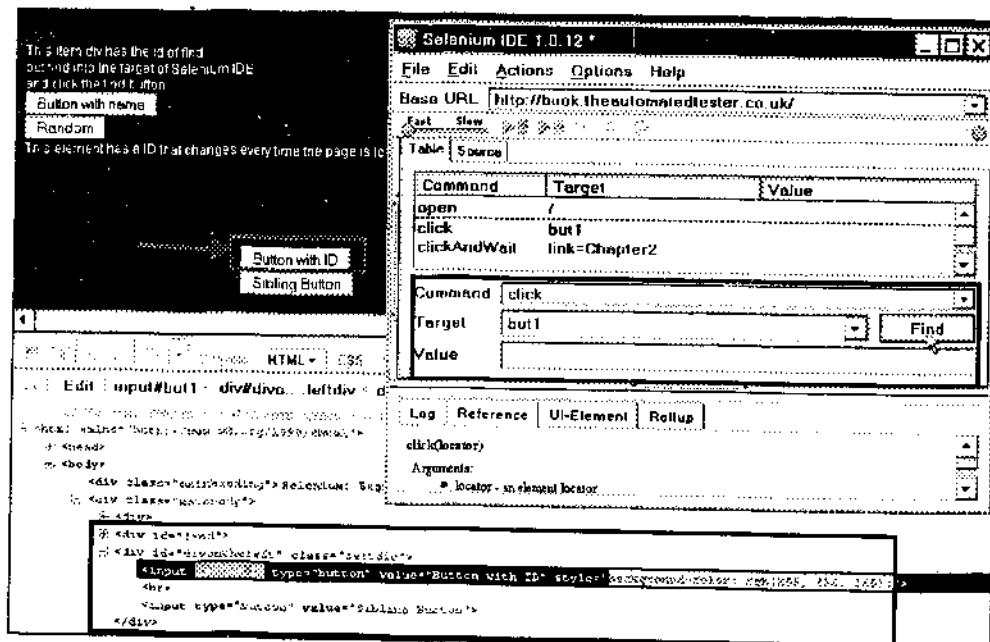
**Bước 2:** nhấp chọn Chapter2.

**Bước 3:** ví dụ bạn sử dụng chương trình Firebug và rê chuột vào nút Button with ID như hình minh họa.



Lấy giá trị thuộc tính của nó là `Id="but1"` để thực hiện việc tìm kiếm lỗi bằng ID trên Selenium IDE.

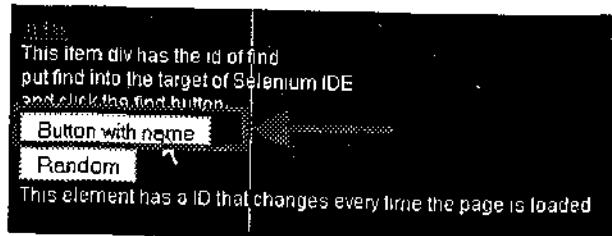
**Bước 4:** chọn lệnh trong khung Command và chọn lệnh thực thi trong Combobox Target tương ứng, sau đó nhấp nút Find tìm kiếm yếu tố này trên trang web. Nó sẽ đổi màu để làm nổi bật yếu tố cần tìm như hình trang bên.



## 5.2 Xác định vị trí các yếu tố bởi tên (name)

Các yếu tố không nhất thiết phải có tên, nó có thể có tên để chúng ta xác định nó trên trang web khi nhấp vào combobox Target. Tương tự như tìm kiếm với ID, bạn thực hiện các bước như sau:

**Bước 1:** nhấp chọn nút **Button with name**.



Sử dụng chương trình Firebug và nhấp chuột vào nút **Button with name** để có đoạn code sau:

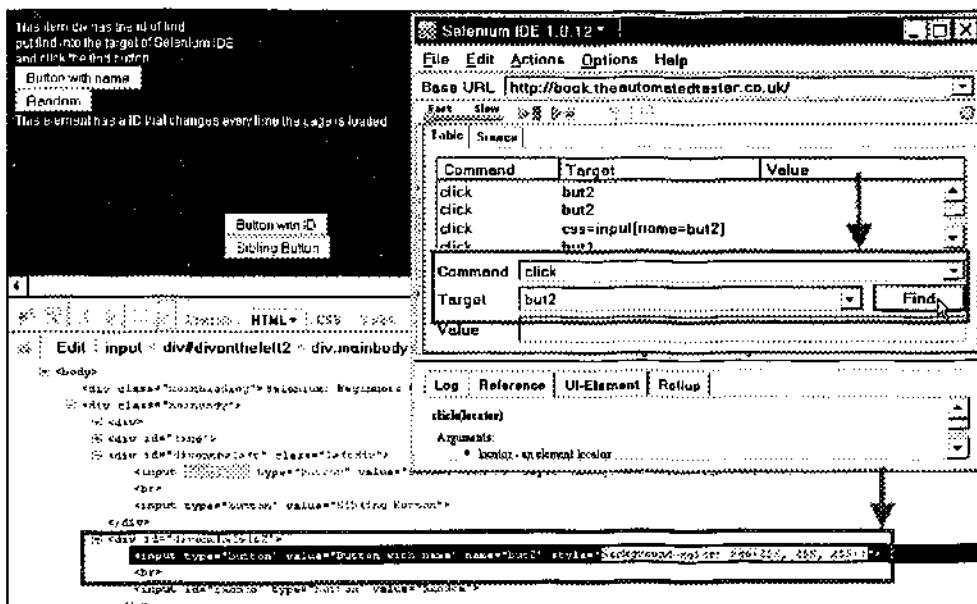
```

<div id="divonthaleft2">
  <input type="button" value="Button with name" name="but2">
  <br>
  <input id="random" type="button" value="Random" name="random">
</div>

```

Lấy giá trị thuộc tính của nó là **name="but2"** để thực hiện việc tìm kiếm lỗi bằng tên trên Selenium IDE.

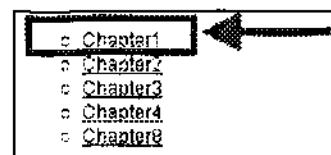
**Bước 2:** chọn lệnh trong khung **Command** và chọn lệnh thực thi trong combobox **Target** tương ứng, sau đó nhấp nút **Find** tìm kiếm yếu tố này trên trang web. Nó sẽ đổi màu để làm nổi bật yếu tố cần tìm như hình.



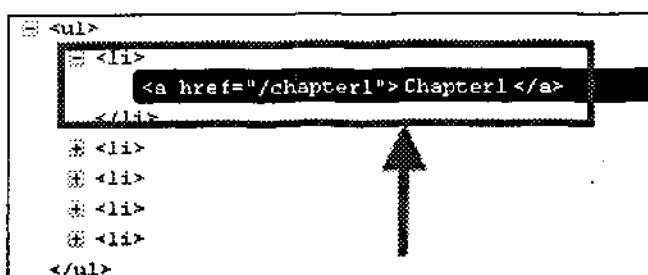
### 5.3 Xác định vị trí các yếu tố bằng liên kết

Yếu tố liên kết thường phổ biến trong các trang web, nó được dùng để tạo liên kết giữa các trang web với nhau. Từ đó người dùng có thể điều hướng trang web một cách dễ dàng.

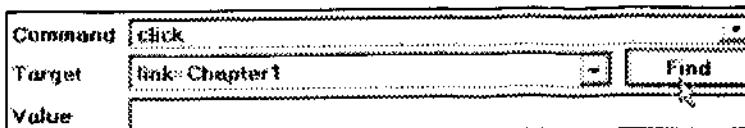
**Bước 1:** nhấp chọn link Chapter 1 có trên trang web.

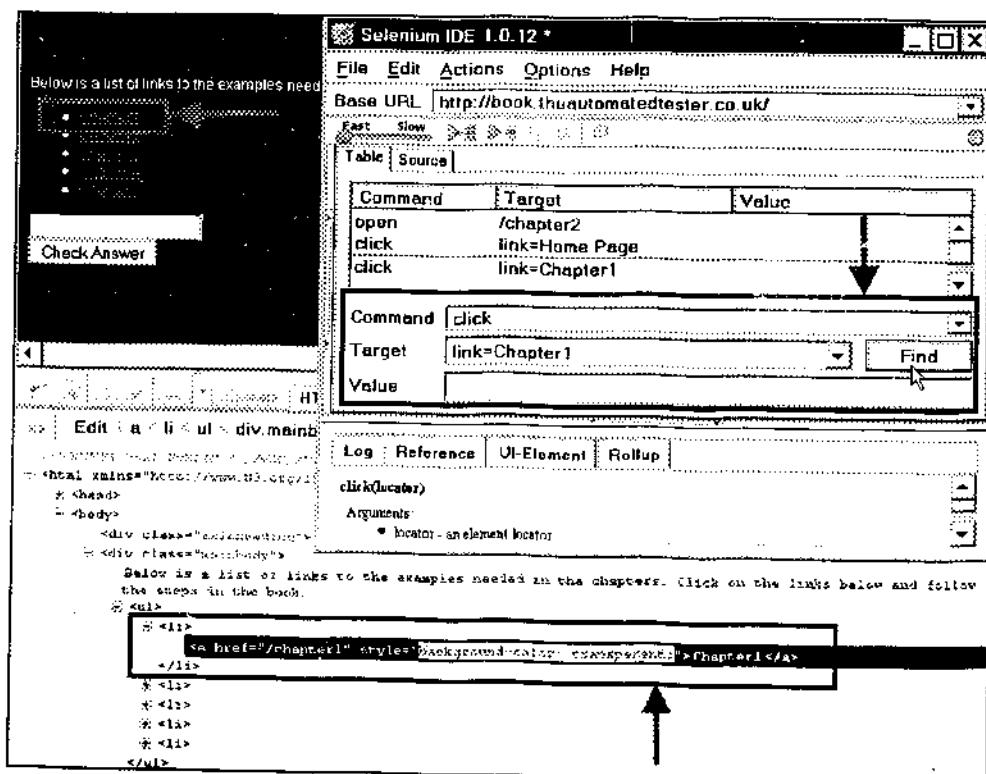


**Bước 2:**  
sử dụng chương  
trình Firebug và  
nhấp chuột vào  
link Chapter 1  
để có đoạn code  
như hình:



**Bước 3:** chọn lệnh trong khung Command và chọn lệnh thực thi trong combobox Target tương ứng, sau đó nhấp nút Find tìm kiếm yếu tố này trên trang web. Nó sẽ đổi màu để làm nổi bật yếu tố cần tìm như hình minh họa.

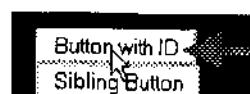




#### 5.4 Xác định vị trí các yếu tố bởi XPath

Phần trình bày sau hướng dẫn bạn làm việc Xpath, nó cho phép truy cập DOM như một tài liệu XML. Với Xpath chúng ta có thể thực hiện các truy vấn phức tạp trên trang.

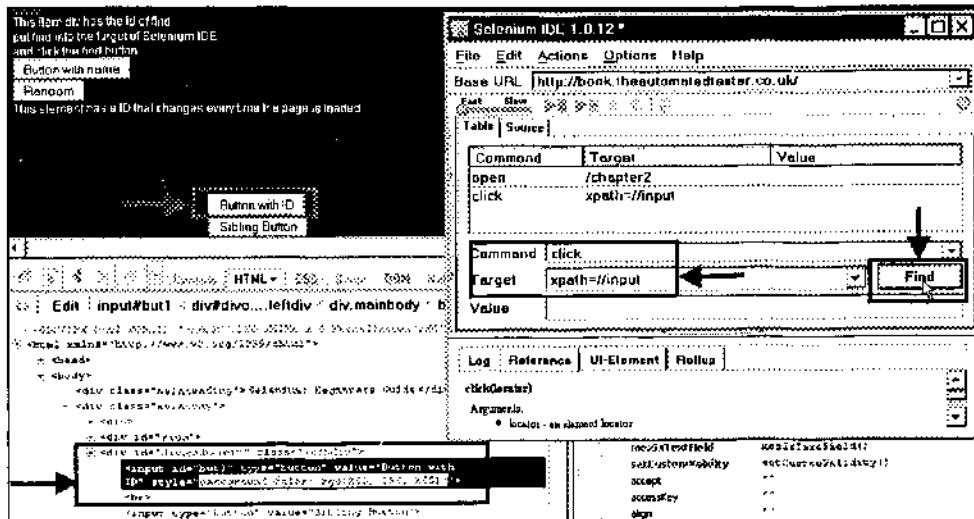
**Bước 1:** nhấp chọn nút Button with ID.



**Bước 2:** sử dụng chương trình Firebug và nhấp chuột vào nút Button with ID để có đoạn code như hình:

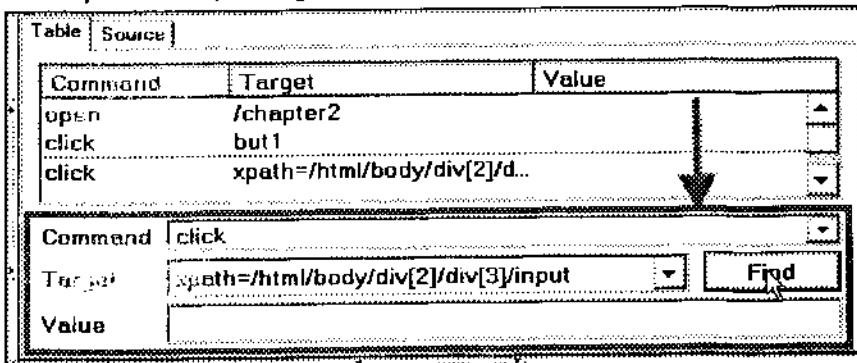
```
<div id="divontheleft" class="leftdiv">
    <input id="but1" type="button" value="Button with ID" style="background-color: #FFFF00; color: #0000FF;">
    <br>
    <input type="button" value="Sibling Button">
</div>
```

**Bước 3:** chọn lệnh trong khung Command và gõ lệnh thực thi là `xpath=/input` trong combobox Target, sau đó nhấp nút Find tìm kiếm yếu tố này trên trang web. Nó sẽ đổi màu để làm nổi bật yếu tố cần tìm như hình trang bên.



Tuy nhiên, kiểu truy vấn với cú pháp **xpath=//Input** sẽ phân tích toàn bộ cú pháp DOM cho đến khi gặp vấn đề bạn muốn tìm kiếm. Vì vậy, bạn có thể sử dụng đường dẫn trực tiếp, thay vì sử dụng // giờ / nhưng phải chắc chắn rằng các nút đầu tiên trong truy vấn là HTML. Trong khung Target gõ đường dẫn **xpath=/html/body/div[2]/div[3]/input** và nhấp nút Find để tìm kiếm vị trí của nó trên web.

Kết quả thu được cũng như hình ở bước 3.

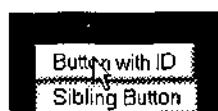


Đây là loại truy vấn XPath sẽ tìm các phần tử fraction nhanh hơn, nhưng nếu giao diện người dùng của bạn thay đổi nó có thể bị thất bại.

Ngoài ra bạn có thể sử dụng các thuộc tính phần tử trong truy vấn XPath. Nhưng điều này rất tốn kém và mất nhiều thời gian.

## 5.5 Xác định vị trí các yếu tố bởi CSS

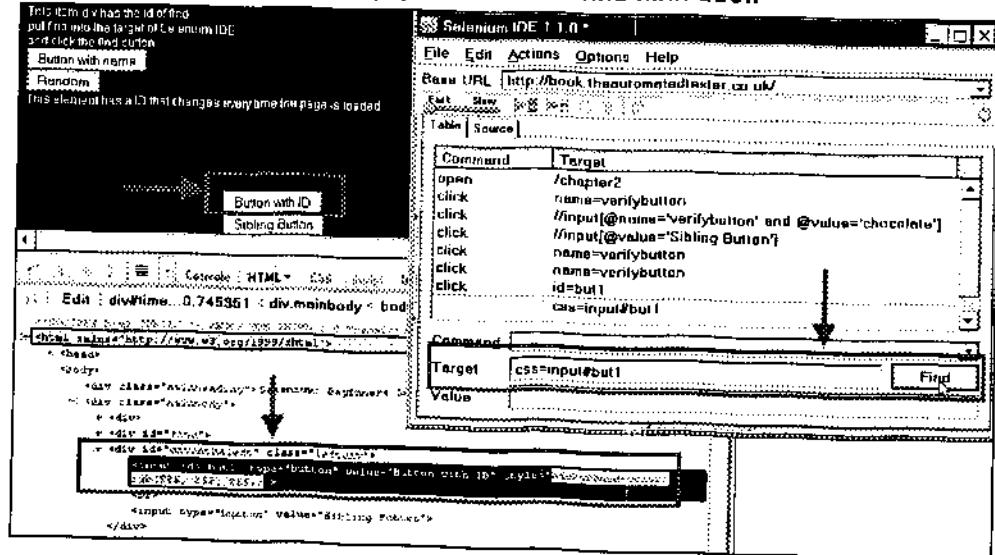
**Bước 1:** thao tác trên web, nhấp chuột vào nút **Button with ID**.



**Bước 2:** sử dụng chương trình Firebug và nhấp chuột vào nút **Button with ID** để có đoạn code như hình:

```
<div id="div-with-buttons" class="leftDiv">
    <input id="but1" type="button" value="Button with ID">
    <br>
    <input type="button" value="Sibling Button">
</div>
```

**Bước 3:** trong khung combobox **Target** gõ lệnh thực thi là **css=input#but1**, sau đó nhấp nút **Find** tìm kiếm yếu tố này trên trang web. Nó sẽ đổi màu để làm nổi bật yếu tố cần tìm như hình dưới.



Trên đây là cấu trúc đường dẫn khi bạn tìm kiếm theo ID trong CSS với dấu vào của nó là thẻ **input**. Ngoài ra còn có cấu trúc tìm kiếm theo tab đầu vào là DIV hay các tab khác.

Tìm các yếu tố theo phương pháp CSS là phổ biến nhất. Rất nhiều các truy vấn mà người lập trình tạo ra bắt đầu với một nút phân biệt bởi lớp CSS. Sau đó thông qua DOM để tìm các nút con, cháu tương ứng.

Khi nhấp nút **Find** (tìm kiếm), bảng **Quick Info** sẽ xuất hiện các thuộc tính CSS cơ bản thông báo cho người dùng biết các chi tiết liên quan đến yếu tố kiểm tra.

<b>Quick Info:</b>
nodeName: INPUT
id: but1
offsetWidth: 100
offsetHeight: 22
<b>Computed Style:</b>
width: 100px
height: 30px
position: static
top: auto
right: auto
bottom: auto
left: auto
margin-top: 0px
margin-right: 0px
margin-bottom: 0px
margin-left: 0px
color: #000000
display: inline
visibility: visible

## 6 SỬ DỤNG CÁC MẪU KIỂM TRA SẴN

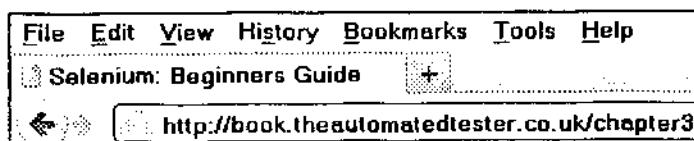
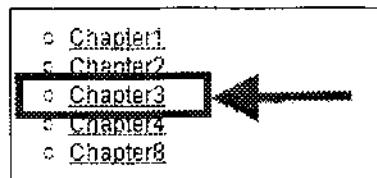
### 6.1 Verify Text, sử dụng exact:văn bản

Phần sau đây trình bày cách kiểm tra các văn bản trên trang web có đúng và phù hợp với trang web hay không.

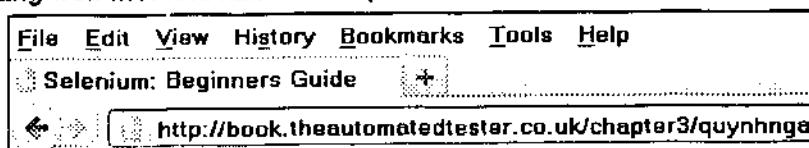
#### Sử dụng cú pháp exact:văn bản

Hãy chắc chắn trang web của bạn có thể tự tạo một trang mới với tên đường dẫn tự nhập và văn bản này sẽ được hiển thị trên trang.

**Bước 1:** nhấp chọn liên kết [link Chapter3](#) trên trang chính để mở cửa sổ mới. Trên thanh địa chỉ lúc này hiển thị đường dẫn tới trang web có tên là **Chapter3**.



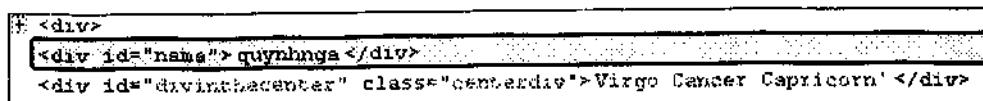
**Bước 2:** bạn gõ thêm tên muốn đặt vào tiếp sau đường dẫn trên để tạo một trang web mới như hình minh họa.



**Bước 3:** nhấp Enter để chạy trang web vừa mới đặt tên. Lúc này trên giao diện trang sẽ có văn bản xuất hiện với tên giống với nội dung bạn đặt tại thanh địa chỉ.



**Bước 4:** sử dụng Firebug và nhấp chuột vào text **quynhnga** để có đoạn code như hình: div có tên **quynhnga** đã xuất hiện trên trang web.



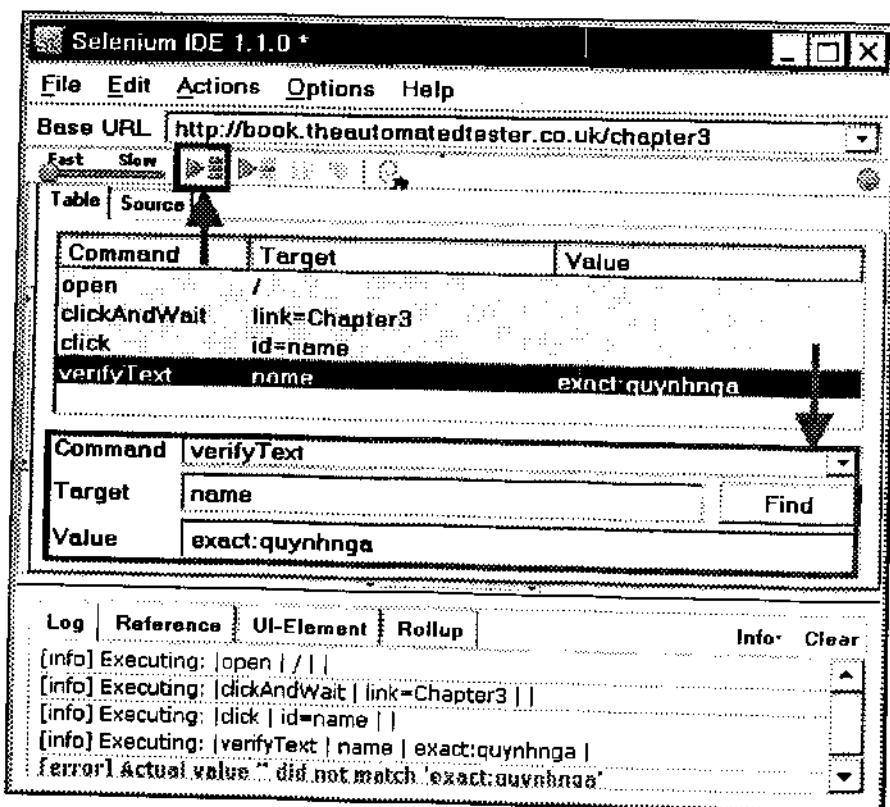
**Bước 5:** trong khung Command bạn thực hiện các bước sau:

Chọn sự kiện là **verifyText**.

Target = **name**

Value (giá trị) = **exact:quynhnga**.

Nhấp chọn biểu tượng kiểm tra text, lỗi thông báo hiện lên ở tab Log.

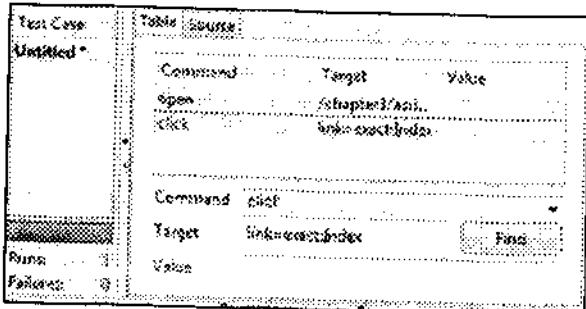


## 6.2 Sử dụng exact: on links

Kiểm tra theo đường liên kết

Trong khung Command chọn sự kiện là click, Target = link=exact:index.

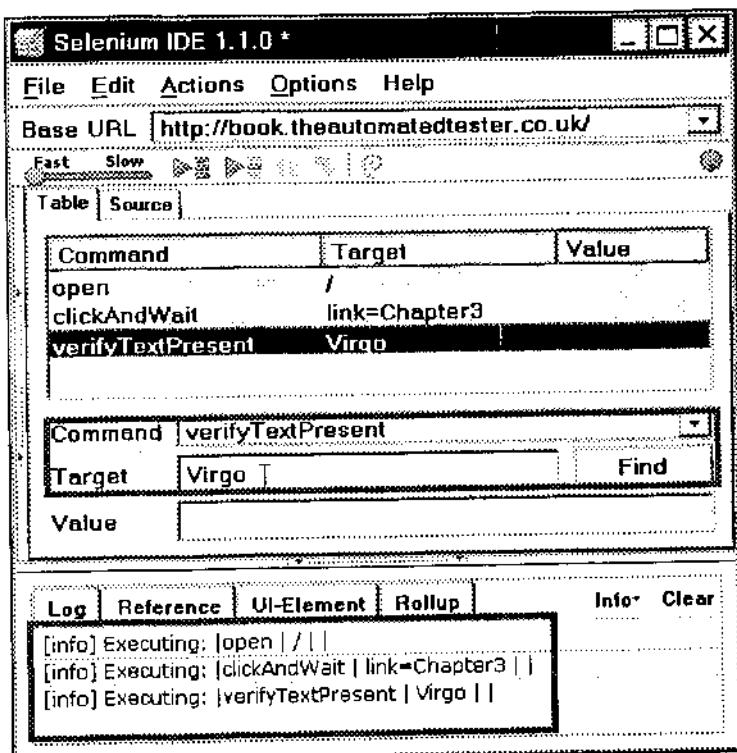
Sử dụng tương tự như phần 6.1



## 6.3 Sử dụng command = verifyTextPresent

Trước khi kiểm tra với sự kiện này, bạn phải chuẩn bị code trang web có hỗ trợ hiển thị tên người dùng và ngày hiện tại. Khi kiểm tra đúng nó sẽ trả về kết quả người dùng chính xác có trên màn hình.

Trong khung Command chọn giá trị verifyTextPresent, với Target = Virgo sau đó nhấp nút Find xem thực sự người dùng đó có tồn tại trên trang web hay không.

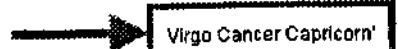


### 6.3 Sử dụng dấu sao \* để kiểm tra các từ hoặc chữ cái

Nếu trong trang web của bạn có sẵn các từ như trang web từ điển, để kiểm tra các chữ cái hoặc kí tự bắt đầu hoặc kết thúc của một từ nào đó bạn thực hiện các bước sau:

Ví dụ kiểm tra các từ bắt đầu bằng kí tự Cap như hình bên.

#### Selenium: Beginners Guide

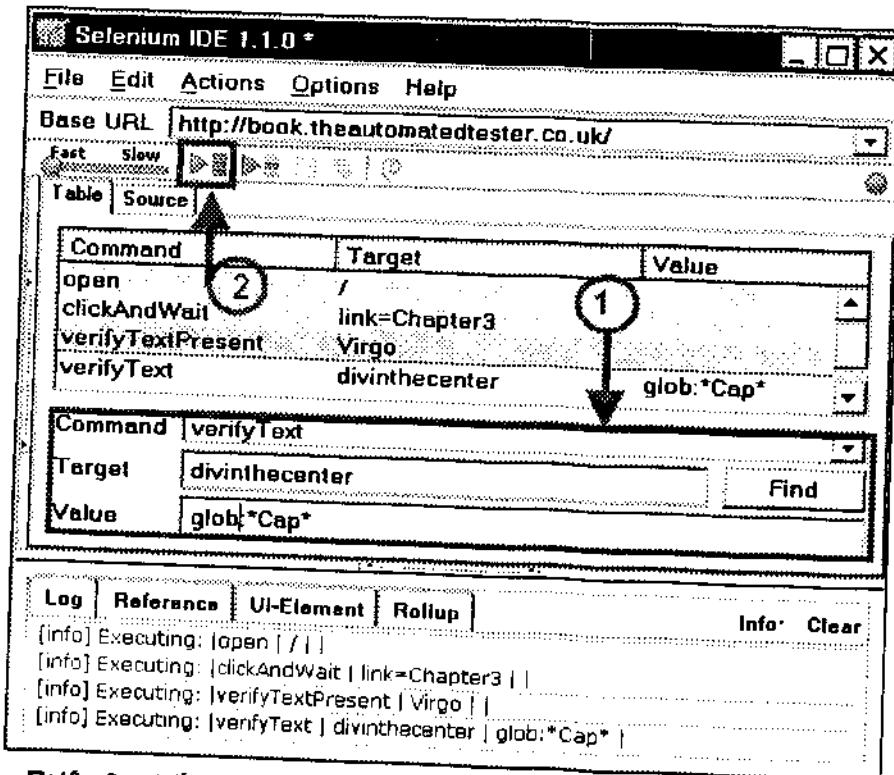


**Bước 1:** sử dụng chương trình Firebug và nhấp chuột vào text Capricorn để có đoạn code như hình.

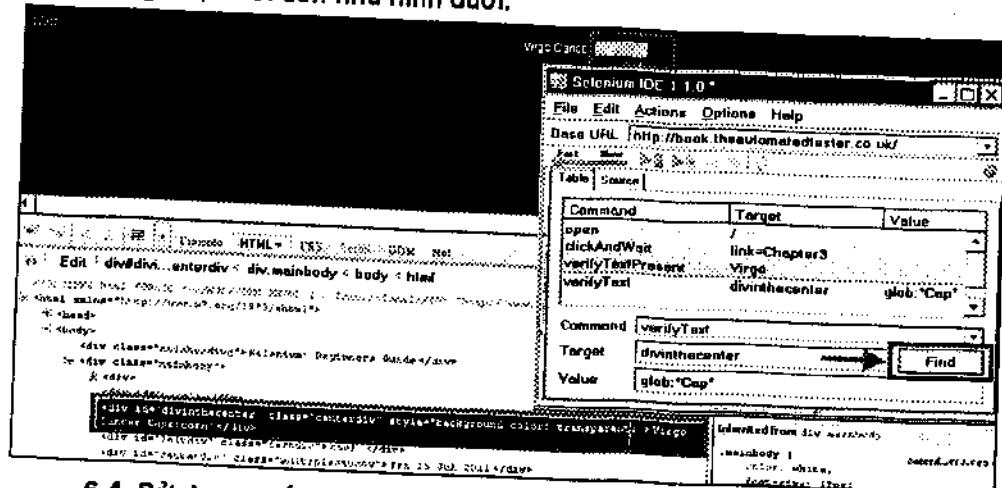
```
<div id="name"></div>
<div id="divinthecenter" class="centerdiv">Virgo Cancer Capricorn' </div>
<div id="leftdiv" class="leftdiv">cool </div>
<div id="centerdiv" class="multiplewindow">Fri 15 Jul 2011</div>
```

**Bước 2:** đầu tiên bạn hãy tạo Command mới.

Kế tiếp trong combobox của Command chọn sự kiện verifyText, với vị trí cần tương tác trong ô Target là divinthecenter, giá trị nhập vào theo cú pháp glob:**\*Cap\*** sau đó nhấp nút kiểm tra.



**Bước 3:** nhấp nút **Find** để tìm vị trí các từ bắt đầu với kí tự Cap có trên trang web. Tất cả các từ trùng với nội dung sẽ được đánh dấu, đồng thời đoạn code cũng được bôi đen như hình dưới.



#### 6.4 Sử dụng dấu chấm hỏi ? để kiểm tra từ

Bạn có thể sử dụng kí tự dấu chấm hỏi ? để kiểm tra các từ có từ kết thúc bằng các kí tự muốn tìm kiếm, ví dụ tìm kiếm các từ có kí tự kết thúc là ool.

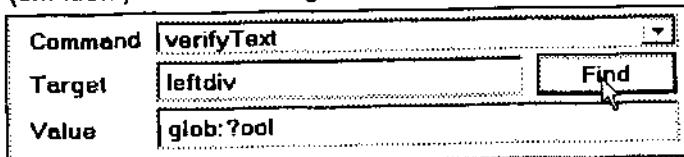
**Bước 1:** vào trang web <http://book.theautomatedtester.co.uk>

**Bước 2:** khởi động chương trình Selenium IDE, sau đó nhấp chọn đường link Chapter3.

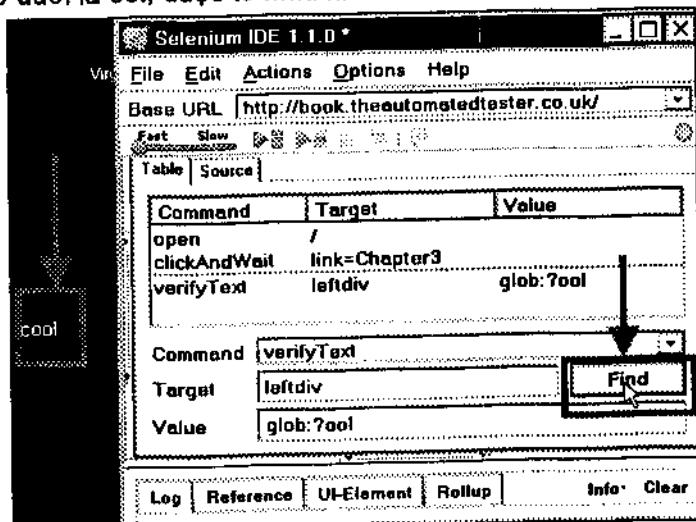
**Bước 3:** sử dụng chương trình Firebug và nhấp chuột vào text cool để có đoạn code như hình: div có id="leftdiv" xuất hiện như hình dưới.

```
<div>
<div id="navas"></div>
<div id="divinthecenter" class="centerdiv">Virgo Cancer Capricorn'</div>
<div id="leftdiv" class="leftdiv" style="background-color: transparent;">cool </div>
<div id="centerdiv" class="multiplevindow">Mon 16 Jul 2011 </div>
```

**Bước 4:** trong khung Command chọn sự kiện có sẵn là verifyText, diền vị trí Target (tìm kiếm) là leftdiv với giá trị cần tìm với cú pháp glob:?ool.



**Bước 5:** nhấp chọn nút Find (tim), trên trang web sẽ tự động nhảy tới vị trí các từ có đuôi là ool, được tô màu như hình dưới.



## 6.5 Sử dụng các biểu thức thông thường

Biểu thức thông thường được ứng dụng rộng rãi trong các ứng dụng. Để kiểm tra, bạn sử dụng cú pháp regexp: chuỗi.

### Kiểm tra ngày tháng năm.

Sử dụng số ký tự chữ cái tương ứng đã được quy định để kiểm tra bao gồm: 2 chữ cái xác định ngày của tháng, 3 chữ cái xác định tháng và 4 chữ cái xác định năm.

Các con số sẽ được bọc bởi cặp dấu ngoặc {}. Phương thức này dùng để kiểm tra xem nó có phù hợp với khuôn mẫu hay không.

**Bước 1:** vào trang web <http://book.theautomatedtester.co.uk>

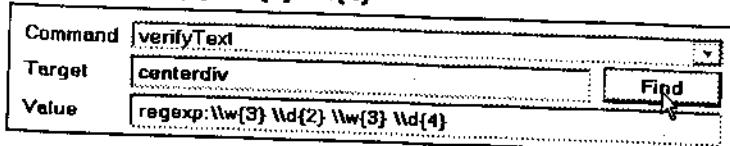
**Bước 2:** khởi động chương trình Selenium IDE, sau đó nhấp chọn đường link Chapter3.

**Bước 3:** sử dụng chương trình Firebug và nhấp chuột vào text hiển thị ngày tháng năm trên trang web để có đoạn code như hình:

```
<div>
<div id="name" style="background-color: transparent;"></div>
<div id="divinthecenter" class="centerdiv">Virgo Cancer
Capricorn' </div>
<div id="leftdiv" class="leftdiv" style="background-color: transparent;">cool </div>
<div id="centerdiv" class="multiplewindow" style="background-color: transparent;">Non 18 Jul 2011</div>
```

**Bước 4:** trong khung Command chọn sự kiện có sẵn là verifyText, diển ví trí Target (tìm kiếm) là centerdiv với giá trị cần tìm với cú pháp:

`regexp:\w{3} \d{2} \w{3} \d{4}`



**Bước 5:** nhấp chọn nút Find (tim), trên trang web sẽ tự động nhảy tới ví trí hiển thị ngày tháng năm được tô màu như hình dưới.

Command	Target	Value
open	/	
clickAndWait	link=Chapter3	
verifyText	leftdiv	glob:cool
click	id=centerdiv	
click	id=leftdiv	
verifyTextPresent	name	glob:[name]cool
verifyText	centerdiv	regexp:\w{3} \d{2} \w{3} \d{4}
click	id=centerdiv	regexp:\w{3} \d{2} \w{3} \d{4}

## 7 SỬ DỤNG JAVASCRIPT ĐỂ KIỂM TRA LỖI

### 7.1 Kiểm tra nhập văn bản được tạo bởi javascript.

Bước 1: vào trang web <http://book.theautomatedtester.co.uk>

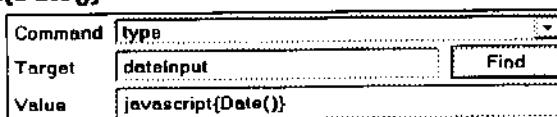
Bước 2: khởi động chương trình Selenium IDE, sau đó nhấp chọn đường link Chapter4.

Bước 3: sử dụng chương trình Firebug và nhấp chuột vào khung cho phép nhập văn bản trên trang web để có đoạn code như hình:

```
<div class="mainbody">
  <div>
    <div id="dateDiv">
      <input id="dateInput" type="text" style="background-color: #fff(255, 255, 255);">
    </div>
```

Bước 4: trong khung Command chọn sự kiện có sẵn là type, điền vị trí Target (tim kiếm) là dateInput với giá trị cần tìm có cú pháp:

**Javascript(Date())**



Bước 5: nhấp chọn nút Find (tìm), trên trang web sẽ tự động nhảy tới vị trí nhập văn bản được tô màu như hình dưới.

Command	Target	Value
open	/	
clickAndWait	link=Chapter3	glob:#tool
verifyText	leftdiv	glob:#tool
click	id=centerdiv	glob:[c]mp[uo]
click	id=leftdiv	regexp:\w{3}\d{2}\w{3}
verifyTextPre	centerdiv	glob:[c]mp[uo]
verifyText	centerdiv	regexp:\w{3}\d{2}\w{3}
click	id=centerdiv	glob:[c]mp[uo]
type	dateInput	javascript(Date())

Command: type  
Target: dateInput  
Value: javascript(Date())

Ngoài ra, bạn có thể sử dụng một lúc nhiều lệnh javascript để kiểm tra. Ví dụ: cùng lúc kiểm tra ngày và giờ như cú pháp trong hình.

Command	<input type="text" value="type"/>
Target	<input type="text" value="dateInput"/> <input type="button" value="Find"/>
Value	<input type="text" value="javascript{ d = new Date();d.getHours() }"/>

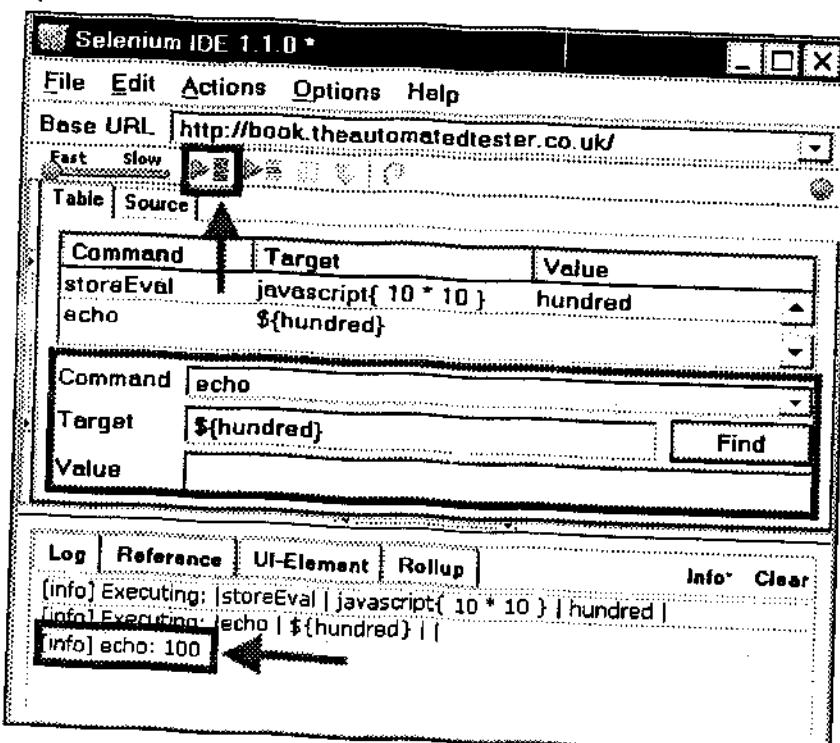
## 7.2 Kiểm tra thời gian lưu trữ kết quả trong 1 biến

Bước 1: khởi động chương trình Selenium IDE.

Bước 2: chọn sự kiện **storeEval** dùng để lưu trữ biến với Target **javascript{ 10 \* 10 }** và giá trị sử dụng là **hundred**.

Bước 3: chọn sự kiện **echo** (Thông báo) để hiển thị kết quả với Target là giá trị biến  **\${hundred}**

Bước 4: nhấp biểu tượng kiểm tra các sự kiện để cho kết quả như hình minh họa.

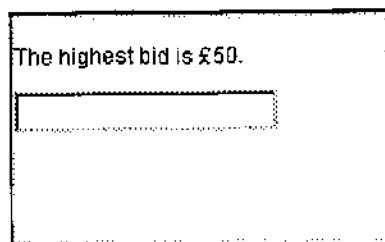


Ví dụ: bạn muốn kiểm tra một trang web đấu giá qua mạng. Sau khi kỳ đấu giá kết thúc, nó sẽ thông báo kết quả đấu giá cao nhất. Để thấy rõ việc kiểm tra này, sau khi chương trình kiểm tra xong nó sẽ tự động cộng thêm 5 và hiển thị lên vùng textbox.

Bước 1: vào trang web <http://book.theautomatedtester.co.uk>

**Bước 2:** khởi động chương trình Selenium IDE, sau đó nhấp chọn đường link Chapter4.

**Bước 3:** sử dụng chương trình Firebug và nhấp chuột vào khung textbox để có đoạn code như hình dưới.



```
<div id="auction" class="belowcenter">
  <p>
  <p>
    <input id="nextBid" type="text">
  </p>
</div>
```

**Bước 4:** chọn sự kiện **storeText** với target (vị trí) **bid**, có giá trị **bid**.

Chọn sự kiện **type**, vị trí **nextBid**, giá trị thêm vào có cú pháp:

**Javascript{ +storedVars['bid']+ 5}**

Kế tiếp bạn nhấp biểu tượng kiểm tra các sự kiện thêm vào có phù hợp với cấu trúc trang web hay không.

The screenshot shows the Selenium IDE 1.1.0 interface. At the top, there's a menu bar with File, Edit, Actions, Options, Help. Below it is a toolbar with Fast, Slow, and various icons. A dropdown menu for Base URL shows the URL <http://book.theautomatedtester.co.uk/>. The main area has tabs for Table and Source, with Table currently selected. A command table is displayed:

Command	Target	Value
storeText	bid	bid
type	nextBid	javascript{ +storedVars['bid']+ 5}

A large arrow points from the bottom of the table to a detailed view of the last row. This view shows the Command (type), Target (nextBid), and Value (javascript{ +storedVars['bid']+ 5}).

At the bottom of the interface, there are tabs for Log, Reference, UI-Element, Rollup, Info, and Clear. The Log tab is active, displaying the following log entries:

- [info] Executing: |storeText | bid | bid |
- [info] Executing: |type | nextBid | javascript{ +storedVars['bid']+ 5} |

Kết quả sau khi chạy kiểm tra các sự kiện như hình bên.

Hoặc bạn sử dụng sự kiện verifyEval để lưu trữ biến.

The highest bid is £60.

55

Table	Source									
<table border="1"> <thead> <tr> <th>Command</th> <th>Target</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>storeText</td> <td>bid</td> <td>bid</td> </tr> <tr> <td>verifyEval</td> <td>javascript{...}</td> <td> \${bid}</td> </tr> </tbody> </table>	Command	Target	Value	storeText	bid	bid	verifyEval	javascript{...}	\${bid}	
Command	Target	Value								
storeText	bid	bid								
verifyEval	javascript{...}	\${bid}								
<table border="1"> <thead> <tr> <th>Command</th> <th>verifyEval</th> </tr> </thead> <tbody> <tr> <td>Target</td> <td>javascript{ 5*10 }</td> </tr> <tr> <td>Value</td> <td> \${bid}</td> </tr> </tbody> </table>	Command	verifyEval	Target	javascript{ 5*10 }	Value	\${bid}	<input type="button" value="Find"/>			
Command	verifyEval									
Target	javascript{ 5*10 }									
Value	\${bid}									

### 7.3 Truy cập trình duyệt với javascript

Ví dụ: kiểm tra một trang web từ điển trực tuyến, bạn kiểm tra ngữ âm hiển thị như thế nào để có thể phát âm. Khi đó bạn cần phải lưu trữ các từ được tìm thấy sau đó áp dụng thuật toán cơ bản để phân chia các từ vào các phần khác nhau.

Sử dụng Browserbot để có quyền truy cập vào cửa sổ trình duyệt.

Cú pháp: var window = this.browserbot.getUserWindow();

Browserbot là các đối tượng JavaScript cho phép Selenium điều khiển trình duyệt. Nó sẽ ghi đè truy cập vào các cửa sổ, tài liệu, và các đối tượng quan trọng khác JavaScript của trình duyệt mà bạn có thể truy cập.

Sử dụng getUserWindow() để trả về đối tượng cửa sổ. Gọi XPCNativeWrapper xóa bỏ vỏ bọc Window trong Selenium mà Firefox tạo ra.

**Bước 1:** vào trang web <http://book.theautomatedtester.co.uk>

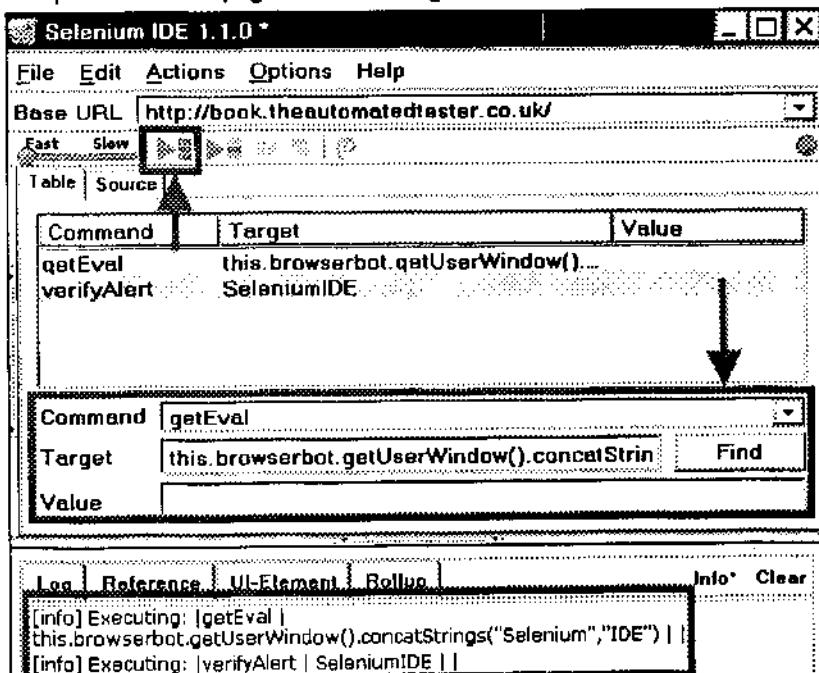
**Bước 2:** khởi động chương trình Selenium IDE, sau đó nhập chọn đường link Chapter4.

**Bước 3:** sử dụng Selenium và IDE khi truy cập trình duyệt.

Trong khung command chọn sự kiện có sẵn là getEval, điền vị trí target this.browserbot.getUserWindow().concatStrings("Selenium","IDE")

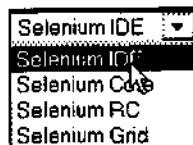
Gọi sự kiện verifyAlert để xác minh cảnh báo với vị trí Selenium IDE.

Nhấp vào biểu tượng kiểm tra trang web như hình dưới.



#### 7.4 Kiểm tra đánh giá Javascript với Browserbot

Ví dụ kiểm tra trong hộp combobox chứa 4 công cụ hỗ trợ kiểm tra tự động các tính năng của ứng dụng web.



**Bước 1:** sử dụng chương trình Firebug và nhấp chuột vào combobox để có đoạn code javascript như hình dưới.

```
function loadSelect(){
    $('#ajaxLoad').append("<option value='Selenium IDE'>Selenium IDE</option>" +
        "<option value='Selenium Core'>Selenium Core</option>" +
        "<option value='Selenium RC'>Selenium RC</option>" +
        "<option value='Selenium Grid'>Selenium Grid</option>");
}
```

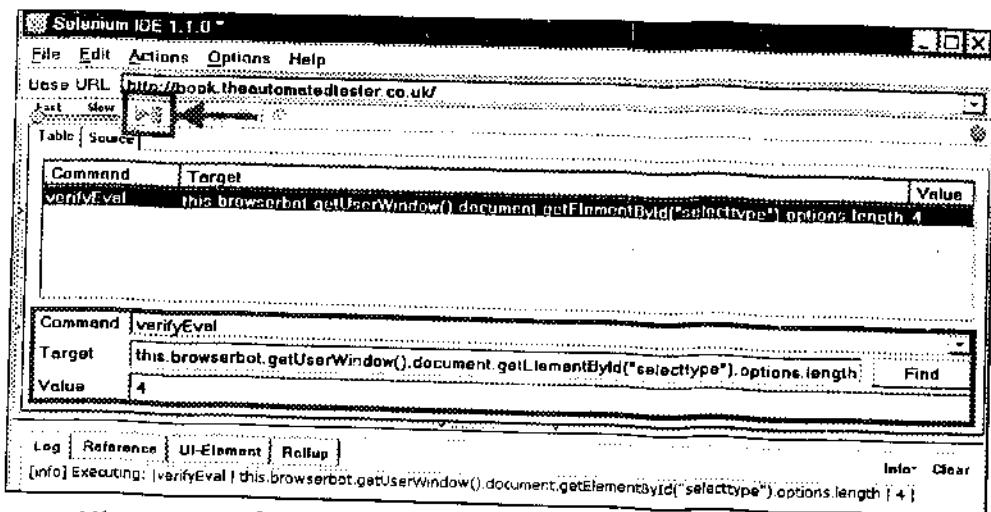
**Bước 2:** trong khung Command bạn thực hiện các bước sau:

Chọn sự kiện có sẵn là verifyEval.

Điền vị trí Target là:

`this.browserbot.getUserWindow().document.getElementById("select type").options.length`

Điền giá trị (value) bằng 4, sau đó nhấp biểu tượng kiểm tra sự kiện như hình trang bên.



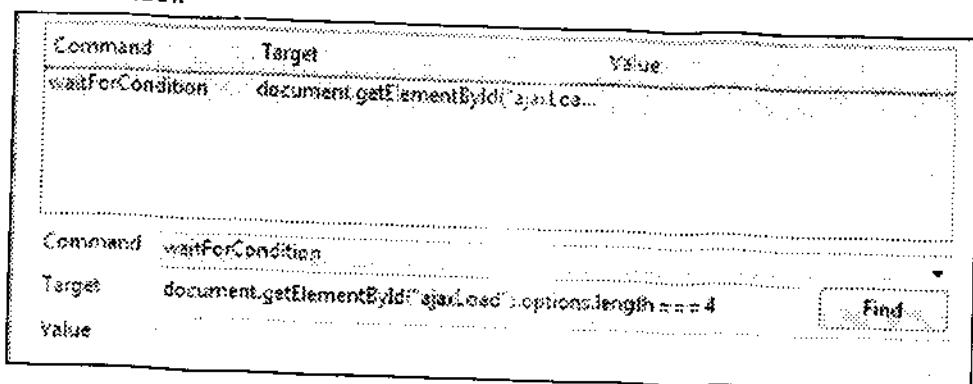
Ví dụ: bạn kiểm tra một ứng dụng phổ biến được xây dựng bởi AJAX trên các website và cơ sở dữ liệu.

Các bước kiểm tra được thực hiện như sau:

**Bước 1:** khởi động ứng dụng Selenium IDE.

**Bước 2:** vào trang web <http://book.theautomatedtester.co.uk>, sau đó nhập chọn link Chapter4.

**Bước 3:** sử dụng sự kiện waitForCondition và kiểm tra độ dài được chọn như hình dưới.



## 7.5 Kiểm tra các sự kiện khi thực hiện nhấp chuột.

Selenium cung cấp cuộc gọi API cho phép thực hiện các sự kiện gắn liền với các yếu tố trên cùng một trang.

Ví dụ:

```
<div id='fireEventDiv' onClick='alert("Alert Thrown")>
```

Khi người dùng nhấp nút gọi sự kiện fireEventDiv sẽ có một thông báo hiện ra, nó có tác dụng chống lại các sự kiện khác diễn ra như:

- + onFocus
- + onBlur
- + onChange
- + onSubmit
- + onMouseOut
- + onMouseOver

Để minh họa cho các yếu tố trên, bạn kiểm tra sự kiện rê chuột lên một đối tượng (onMouseOver).

**Bước 1:** sử dụng chương trình Firebug để có đoạn code javascript như hình dưới. Đây là bảng thông báo khi bạn rê chuột lên các ứng dụng có trên trang web.

```
t <p>
<div id="hoverOver" class="secondbutton" onmouseover="alert('on MouseOver
worked'), ""> Mouse over this </div>
```

**Bước 2:** trong khung Command chọn sự kiện có sẵn là fireEvent, di chuyển vị trí Target hoverOver với giá trị mouseover. Đưa ra sự kiện thông báo verifyAlert với giá trị on MouseOver worked sau đó nhấp nút kiểm tra.

Command	Target	Value
fireEvent	hoverOver	mouseover
verifyAlert	on MouseOver worked	

Command	Target	Value
verifyAlert	on MouseOver worked	

Log:

- [info] Executing: fireEvent | hoverOver | mouseover |
- [info] Executing: verifyAlert | on MouseOver worked | |

Việc tạo sự kiện như trên cho phép bạn rê chuột lên các sự kiện mà không hiển thị bảng thông báo được đề cập ở trước. Điều này sẽ rất có ích khi Selenium không thể truy cập được các lệnh.

## 8 MỞ RỘNG ỨNG DỤNG TIỆN ÍCH CHO NGƯỜI DÙNG

Selenium cho phép các nhà phát triển mở rộng ứng dụng để tạo ra các chức năng giúp việc kiểm tra dễ dàng hơn.

Nếu bạn đang tạo ra một phần mở rộng có thể sử dụng bởi nhiều người, cần chắc chắn rằng nó được lưu ở vị trí trung tâm, như vậy sẽ ngăn chặn được các vấn đề phát sinh trong tương lai khi người dùng sử dụng nó.

Ví dụ: bạn muốn sử dụng một đoạn mã được sử dụng bởi rất nhiều các cuộc kiểm tra khác nhau như `type | locator | javascript{....}`. Tuy nhiên nếu đoạn code javascript trên bị lỗi, khi đó bạn sẽ phải duyệt tất cả các cuộc kiểm tra trước đó có tái sử dụng đoạn code trên. Điều này sẽ làm cho chúng ta gặp khó khăn trong quá trình phát hiện lỗi. Để khắc phục yếu tố trên, Selenium cho phép bạn tạo một chức năng riêng và sau đó có thể sử dụng chúng suốt các đợt kiểm tra mà không bị gây khó khăn.

Trọng tâm của Selenium là phát triển dựa trên nền javascript nên để tạo ra một phần mở rộng, bạn cần tuân thủ theo cú pháp sau;

```
Selenium.prototype.doFunctionName = function(){
    .....
}
```

Yếu tố “do” nằm phía trước tên chức năng cho Selenium dùng để gọi thực hiện một lệnh cho một bước thực hiện.

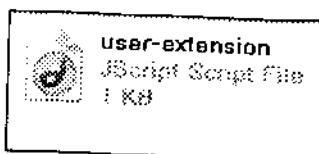
### 8.1 Cài đặt phần mở rộng chức năng cho Selenium

Bước 1: mở trình soạn thảo code bạn thường sử dụng, ví dụ Word, Notepad, Dreamweaver.....

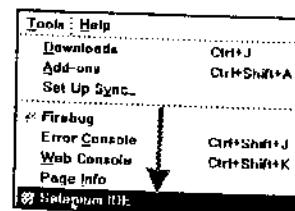
Bước 2: tạo một đoạn code chức năng với tên là Nothing:

```
Selenium.prototype.doNothing= function(){
    .....
}
```

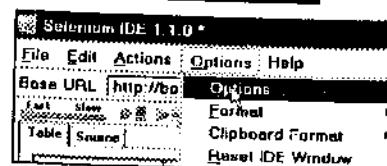
Lưu văn bản trên với định dạng.js (java)



Bước 3: khởi động trình ứng dụng Selenium.



Bước 4: từ menu Options  
nhấp chọn Options.. như hình.

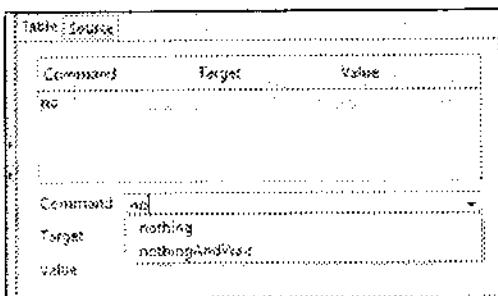
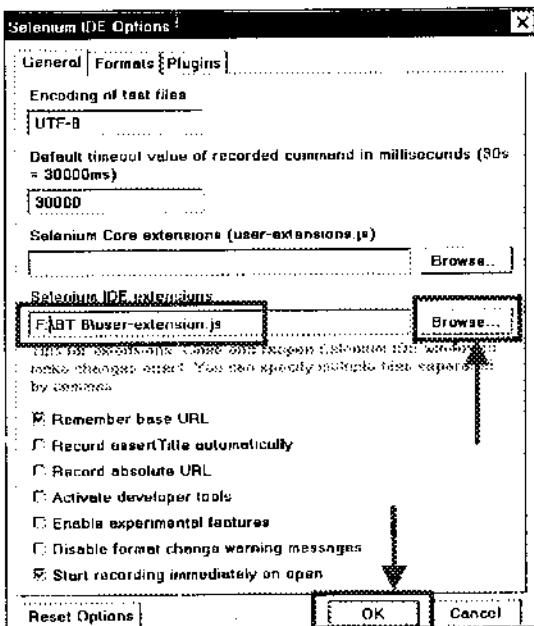


**Hộp thoại Selenium IDE Options xuất hiện, trong phần Selenium IDE extensions bạn nhấp chọn nút Browse để chọn đường dẫn đến file .js vừa tạo.**

Cuối cùng nhấp OK để cài đặt ứng dụng như hình.

**Bước 5: Restart lại ứng dụng Selenium IDE.**

**Bước 6: trong khung Command, bạn gõ vào lệnh mới vừa tạo. Khi đó lệnh mới có sẵn sẽ xuất hiện như hình bên.**



## 8.2 Sử dụng biến Selenium trong phần mở rộng

Ví dụ 1: bạn đang kiểm tra một ứng dụng đòi hỏi nhập một số ngẫu nhiên vào textbox. Khi đó, bạn cần phải tạo ra một phần mở rộng để lưu trữ kết quả trong một biến.

**Bước 1: mở file user-extension.js đã được tạo ra trước đó.**

**Bước 2: tạo ra một chức năng có tên là storeRandom với đoạn code có cú pháp:**

```
Selenium.prototype.doStoreRandom = function(variableName)
{
    random = Math.floor(Math.random()*100000000);
    storedVars[variableName] = random;
}
```

**Bước 3: lưu lại file user-extension.js**

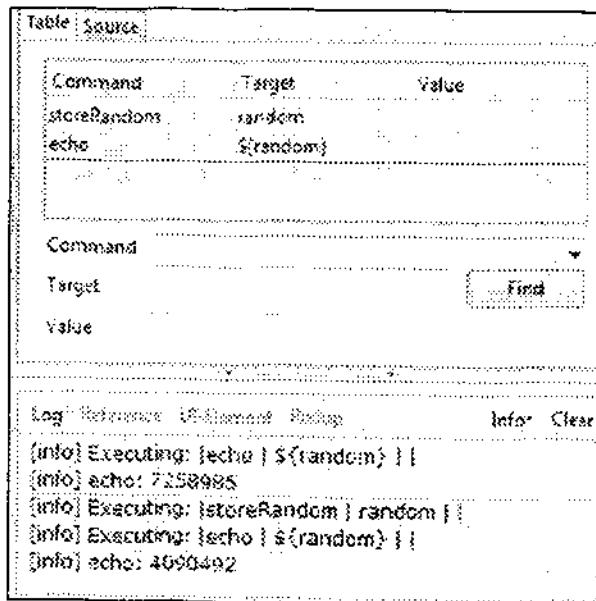
**Bước 4:** Restart lại ứng dụng Selenium IDE.

**Bước 5:** cài đặt phần mở rộng vừa tạo cho Selenium có tên là **storeRandom**, nó là biến dùng để giữ giá trị ngẫu nhiên (như đã trình bày ở 8.1).

**Bước 6:** gọi sự kiện **storeRandom** và đưa ra thông báo kết quả biến random nhận được. Nhấp nút kiểm tra sự kiện để có kết quả như hình dưới.

Sau mỗi lần chạy chương trình, giá trị biến được lưu trước đó bị ghi đè bởi giá trị sau đó.

Ví dụ 2: kiểm tra một code tính toán ngày, giá trị biến dùng để lưu trữ kết quả và không bị ghi đè sau mỗi lần chạy chương trình.



**Bước 1:** khởi động chương trình soạn thảo.

**Bước 2:** tạo ra một chức năng có tên là **doTypeTodaysDate** với đoạn code có cú pháp:

```
Selenium.prototype.doTypeTodaysDate = function(locator)
```

```
{
```

```

var dates = new Date();
var day = dates.getDate();
if (day < 10)
{
    day = '0' + day;
}
month = dates.getMonth() + 1;
if (month < 10)
{
    month = '0' + month;
}
var year = dates.getFullYear();
var prettyDay = day + '/' + month + '/' + year;

```

```
this.doType(locator, prettyDay);
}
```

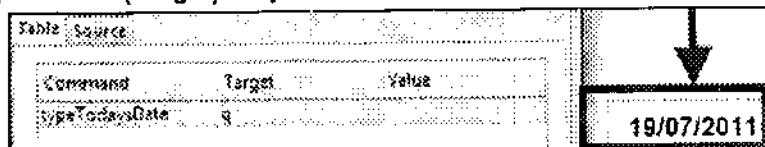
**Bước 3:** lưu file với tên **doTypeTodaysDate.js**, cài đặt và restart lại ứng dụng Selenium IDE.

Sử dụng chương trình Firebug và nhấp chuột vào khung text để có đoạn code như hình:

**Bước 4:** chạy thử ứng dụng.

Gọi sự kiện cần kiểm tra có tên là **TypeTodaysDate**.

Vị trí trả tới (Target) là **q**.



**Ví dụ 3:** cần tương tác với một yếu tố trên trang web và cách duy nhất để truy cập vào nó là thông qua một API Javascript.

Để thực hiện điều này, bạn cần phải truy cập đến yếu tố Browserbot như đã đề cập trong các phần trước, bằng sự kiện **getEval** đã chạy các đoạn mã trong Selenium.

Bây giờ bạn có thể sử dụng các lệnh đơn giản để giảm bớt độ phức tạp trong ví dụ trước đó. Để định dạng ngày tháng năm, bạn sử dụng cấu trúc dd/mm/yyyy.

**Bước 1:** khởi động chương trình soạn thảo.

**Bước 2:** tạo ra một chức năng có tên là **doCheckDate** với đoạn code có cú pháp:

```
Selenium.prototype.doCheckDate = function()
{
    var dates = new Date();
    var day = dates.getDate();
    if (day < 10){
        day = '0' + day;
    }
    month = dates.getMonth() + 1;
    if (month < 10){
        month = '0' + month;
    }
    year = dates.getFullYear();
    return day + '/' + month + '/' + year;
}
```

```

        }

        var year = dates.getFullYear();
        var prettyDay = day + '/' + month + '/' + year;
        this.browserbot.getUserWindow().checkDate(prettyDay);
    }

```

**Bước 3:** lưu file với tên doCheckDate.js, cài đặt và restart lại ứng dụng Selenium IDE.

Sử dụng chương trình Firebug và nhấp chuột vào khung text để có đoạn code như hình:

**Bước 4:** tạo mới sự kiện là verifyText với vị trí là Answer và giá trị Correct.

**Bước 5:** chạy thử ứng dụng ta được kết quả như hình bên.

Command	Target	Value
checkDate		
verifyText	Answer	Correct

## 9 CÁCH TẠO RA MỘT FIREFOX ADD-ON BÌNH THƯỜNG

**Bước 1:** tạo thư mục có tên là Book, tất cả mọi thứ có sau đó đều chứa trong thư mục này.

**Bước 2:** tạo một tập tin gọi là install.rdf cho phép cài đặt tiện ích.

```

<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:em="http://www.mozilla.org/2004/em-rdf#>

```

```
<Description about="urn:mozilla:install-manifest">
<!-- needs to be in the format of an email address, but should be an actual
email address -->
<em:id>your email address</em:id>
<!-- has to be lowercase -->
<em:name>name of addon</em:name>
<em:version>1.0</em:version>
<em:creator>Your Name</em:creator>
<em:description>
A quick plugin for Selenium IDE
</em:description>
<em:type>2</em:type>
<!--Preferences -->
<em:optionsURL>
chrome://book/content/view/options.xul
</em:optionsURL>

<!-- its a firefox plugin -->
<em:targetApplication>
<Description>
<em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
<em:minVersion>1.5</em:minVersion>
<em:maxVersion>4.0.*</em:maxVersion>
</Description>
</em:targetApplication>

<!-- this is an Se-IDE plugin,
so we need to specify it as a requirement -->
<em:requires>
<Description>
<em:id>{a6fd85ed-e919-4a43-a5af-8da18bda539f}</em:id>
<em:minVersion>1.4</em:minVersion>
```

```

<em:maxVersion>1.*</em:maxVersion>
</Description>
</em:requires>
</Description>
</RDF>

```

**Bước 3:** tạo tập tin mới có tên là **chrome.manifest**, cho phép Mozilla Firefox Add-on thêm các chức năng.

```

content book chrome/content/
locale booken-US chrome/locale/en-US/
overlay chrome://selenium-ide/content/selenium-ide-common.xul
chrome://book/content/view/optionsOverlay.xul

```

**Bước 4:** tạo thư mục có tên là **content** nằm trong thư mục cha **Book**.

**Bước 5:** trong thư mục **content** tạo một thư mục có tên là **view**.

**Bước 6:** trong thư mục có tên là **view**, bạn tạo tập tin với tên là **optionsOverlay.xul** cho phép tạo thanh trình đơn thả xuống.

```

<?xml version="1.0"?>
<?xmlstylesheet href="chrome://global/skin/" type="text/css"?>
<overlay id="selenium_overlay" xmlns=
"http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
<menupopup id="options-popup">
<menuitem label="Book Addon options" accesskey="P" oncommand=
>window.open('chrome://book/content/view/options.xul',
'Book Options','chrome=yes,,centerscreen=yes');;" />
</menupopup>
</overlay>

```

**Bước 7:** tạo tập tin có tên là **options.xul**

```

<?xml version="1.0"?>
<?xmlstylesheet href="chrome://global/skin/" type="text/css"?>
<prefwindow id="book-prefs" title="Book Options" width="520"
height="200"
xmlns= "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

```

```
<!-- Pref Pane -->
<prefpane id="book-panel" label="Book Options">
<preferences>
<preference id="pref_perform" name="extensions.selenium-ide.book.
book-options" type="bool" />
</preferences>
<!--Window layout so we can see how it all works together -->
<tabbox>
<tabs>
<tab label="General"/>
</tabs>
<tabpanel flex="1" >
<tabpanel>
<vbox flex="1">
<hbox align="center">
<!--if you use & at the beginning and ; at the end it will allow you to
localize if you create a folder called locale in the root and place in it
different languages
-->
<label control="name" value="A Checkbox"/>
<checkbox preference="checkbox_perf" id="checkbox" />
</hbox>
<spacer height="100" />
</vbox>
</tabpanel>
</tabpanel>
</tabbox>
</prefpane>
</prefwindow>
```

Bước 8: trong tập tin options.xul có thể thêm vào các đoạn Javascript để khi bạn nhấp vào một nút nào đó, nó sẽ chạy tự động.

```
<!-- reference some Javascript -->
<script type="application/x-javascript" src="chrome://selenium-
ide/content/ api.js"/>

<html:script type="application/javascript"> var ide_api = new API();
ide_api.addPluginProvidedFormatter("CheckBox",
"Friendlier Checkbox name", "chrome://book/content/code/file-holding-
info.js");
</html:script>
```

**Bước 9:** tạo thư mục con có tên **code** nằm trong thư mục **content**.

**Bước 10:** trong thư mục **code**, tạo tập tin chứa đoạn mã:

```
load('chrome://book/content/code/other-javascript.js');
```

**Bước 11:** sau khi hoàn thành tất các bước trên bạn đóng gói file theo định dạng **.zip**, kế tiếp đổi đuôi thành **.xpi**.

Phần trình bày trên là các bước cơ bản để có thể tạo một ứng dụng Add-on đơn giản trên Firefox. Phản hướng dẫn ứng dụng Selenium IDE đến đây là kết thúc.

Trong chương tiếp theo bạn sẽ làm quen các ứng dụng với Selenium Remote Control (RC).

## CHƯƠNG 16

# KIỂM THỬ WEBSITE VỚI SELENIUM RC

Chương 16 hướng dẫn bạn làm quen với Selenium Remote Control trong điều khiển từ xa. Đây là một ứng dụng phổ biến nhất của Selenium, nó cho phép các nhà phát triển viết các chương trình kiểm tra để thử nghiệm trên các trình duyệt khác nhau.

Các bạn sẽ tìm hiểu các vấn đề sau trong chương 16:

- Giới thiệu Selenium RC.
- Kiểm tra chạy thử nghiệm Selenium RC.

## I GIỚI THIỆU SELENIUM RC

Selenium RC (viết tắt từ **Selenium Remote Control**): có thể nhận các Test script thu được bởi Selenium IDE, cho phép chỉnh sửa, cài tiến linh hoạt bằng nhiều ngôn ngữ lập trình khác nhau. Sau đó khởi động một trong các trình duyệt web được chỉ định để thực thi kiểm tra trực tiếp trên trình duyệt đó. Selenium RC còn có khả năng lưu lại kết quả kiểm tra. Trong chương 15 bạn đã làm quen với ứng dụng Selenium IDE, tuy nhiên nó chỉ chạy trên trình duyệt Firefox. Với Selenium RC, chương trình có thể chạy trên các trình duyệt khác như Explorer, Mozilla Firefox, Google Chrome và Opera.

Selenium Remote Control được phát triển đầu tiên bởi Patrick Lightbody, với nhu cầu kiểm tra trên tất cả các trình duyệt mà không cần phải cài đặt Selenium Core trên máy chủ web. Nó được phát triển như một proxy ở giữa ứng dụng thử nghiệm và code tương ứng. Selenium Core luôn đi kèm với Selenium RC thay vì được cài đặt ở máy chủ web. Điều này giúp các nhà phát triển có thể tương tác trực tiếp với proxy.

Bạn có thể sử dụng ngôn ngữ lập trình viết các đoạn code để kiểm tra thay vì sử dụng các bài kiểm tra HTML bằng Selenium IDE. Như vậy sẽ giúp bạn tận dụng hết tất cả các lợi thế của ứng dụng và phá bỏ hầu hết các khuôn mẫu có sẵn.

## II KIỂM TRA CHẠY THỬ NGHIỆM SELENIUM RC VỚI INTELLIJ IDEA

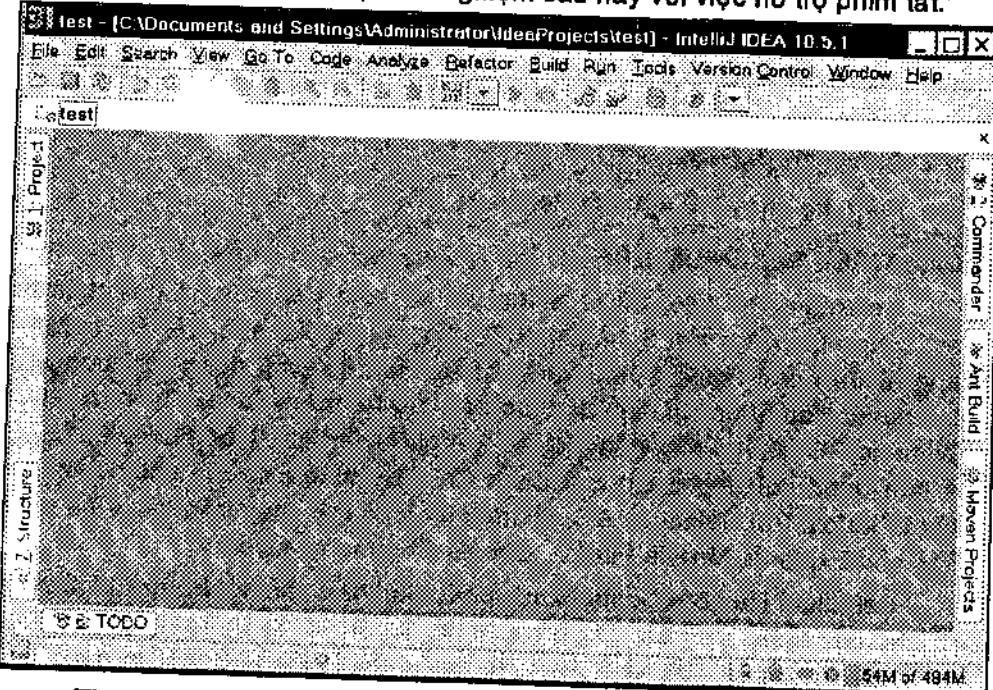
Phần trình bày sau sẽ trình bày cách chuyển đổi trình Selenium IDE thành một ngôn ngữ lập trình. Bạn có thể viết các bài kiểm tra để tận dụng tối đa các ứng dụng của Selenium và thế mạnh của ngôn ngữ lập trình này.

Để có một trình IDE viết bài kiểm tra cho các cuộc thử nghiệm, bạn có thể sử dụng chương trình IntelliJ IDEA.

### ❖ Giới thiệu chương trình IntelliJ IDEA

IntelliJ IDEA được nhiều nhà phát triển Java và các chuyên gia về Java IDE công nhận đây là chương trình tốt trên thị trường. Với tính năng hàng đầu, IntelliJ IDEA làm giảm thời gian lập trình Java và tăng cường đáng kể năng suất của họ. IntelliJ IDEA tạo ra một môi trường thuận lợi nơi mà tất cả các thành viên trong nhóm có thể làm việc với nhau hiệu quả.

Phần này bạn chỉ cần nắm sơ lược các công cụ cơ bản để giúp cho việc tạo bài kiểm tra cho các cuộc thử nghiệm sau này với việc hỗ trợ phím tắt.



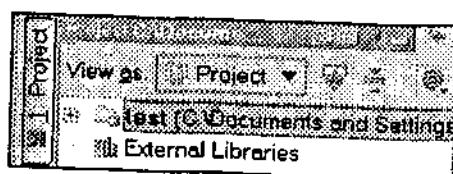
Thanh tiêu đề.

Thanh trình đơn.

Thanh công cụ.

### Project

Cho phép quan sát cấu trúc dự án từ các điểm khác nhau.

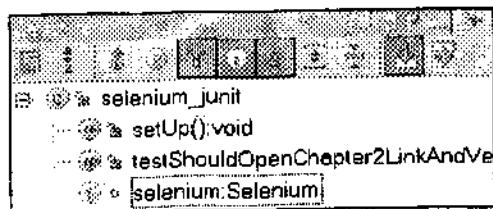


Dưới đây là bảng giới thiệu từng công cụ.

Biểu tượng	Chức năng
View as	Sử dụng danh sách thả xuống để hiển thị dự án của bạn như là một cây thư mục, thiết lập một phạm vi, hoặc gói.
	Di chuyển một tập tin, nút tương ứng trong cửa sổ.
	Đóng tất cả các nút (Ctrl+Subtract or Ctrl-Minus)
	Mở hộp thoại có thể tạo và sửa đổi.
	Thay đổi tên danh sách lựa chọn.
	Hiển thị danh sách dự án.
	Hiển thị các thư viện lưu trữ bên ngoài dự án.

### Structure

Hiển thị cấu trúc một tập tin hiện đang mở trong trình soạn thảo, hoặc lựa chọn trong cửa sổ dự án.



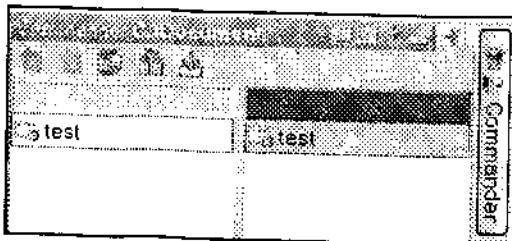
Dưới đây là bảng giới thiệu từng công cụ.

Biểu tượng	Tùy chọn	Chức năng
	PHP	Hiển thị phân cấp PHP.
	HTML View	Hiển thị phân cấp PHP.
	Sort by visibility	Sắp xếp khả năng hiển thị.
	Sort alphabetically	Sắp xếp theo bảng chữ cái.
	Group methods by defining type	Chọn tất cả các phương pháp thực hiện như ghi đè, nhóm lớp...
	Show properties	Nhấp vào nút này để hiển thị getters, setters trong cây thư mục.
	Show Inherited	Hiển thị các phương pháp thừa kế từ lớp hiện tại.

	Show Includes	Hiển thị tất cả báo cáo.
	Show fields	Hiển thị tất cả trường trong cây thư mục.
	Show Constants	Hiển thị các hằng số.
	Show non-public	Hiển thị tất cả các thành viên không được phổ biến.
	Ctrl + Add hoặc Ctrl + Equals	Mở rộng tất cả các nút.
	Ctrl + Subtract hoặc Ctrl + Minus	Thu gọn các nút.
	Autoscroll to source	Tự động di chuyển từ các nút lựa chọn trong cửa sổ công cụ cấu trúc vào dòng tương ứng của mã nguồn trong trình soạn thảo.
	Autoscroll from source	Tự động di chuyển từ dòng tương ứng của mã nguồn trong trình soạn thảo vào các nút lựa chọn trong cửa sổ công cụ cấu trúc.

### Commander

Hiển thị cấu trúc phân cấp của một dự án trong hai khung tương tự nhau và cung cấp một giao diện tiện dụng cho các tập tin và thư mục quản lý. Mỗi cửa sổ commander chỉ hiển thị một mức độ phân cấp.

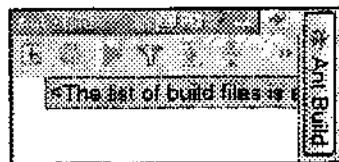


Dưới đây là bảng giới thiệu từng công cụ.

Biểu tượng	Phím tắt	Chức năng
	Ctrl + Alt + Left Ctrl + Alt + Right	Di chuyển lịch sử hoạt động trong cửa sổ.
	Ctrl + U	Trao đổi nội dung giữa 2 khung.
	Alt + F6	Đồng bộ hóa 2 khung.
		Nội dung tập tin được chọn tự động mở trong trình soạn thảo.

**Ant build**

Dùng để xây dựng kịch bản, kiểm soát hành vi và chạy kiểm tra các dự án.

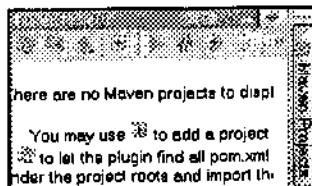


Dưới đây là bảng giới thiệu từng công cụ.

Biểu tượng	Chức năng
	Thêm tập tin xây dựng.
	Loại bỏ tham chiếu đến tập tin được lựa chọn từ các dự án.
	Chạy các mục tiêu.
	Chỉ hiển thị mục tiêu chính.
	Mở rộng, thu gọn các nút.
	Hiển thị các tập tin được chọn.

**Maven Projects**

Cho phép xem các dự án, nguồn tải về, javadoc và thực hiện các giai đoạn đã được xây dựng. Cửa sổ công cụ hiển thị các nút cho từng dự án.

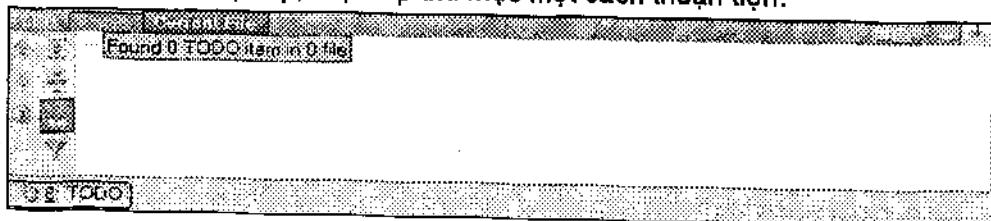


Dưới đây là bảng giới thiệu từng công cụ.

Biểu tượng	Chức năng
	Đồng bộ hóa dự án.
	Khởi động nguồn, đọc cấu trúc.
	Tải nguồn tài liệu.
	Thêm dự án, thư viện.
	Chạy giai đoạn lựa chọn sau khi xây dựng.
	Chuyển sang chế độ offline.

	Bật chế độ tùy chỉnh kiểm tra, bỏ chạy thử nghiệm.
	Hiển thị sự phụ thuộc của dự án hoặc thư viện trong một cửa sổ.
	Đóng các nút trong dự án.
	Cấu hình dự án hiện trong hộp thoại.
	Trợ giúp.
	Hiển thị các trình đơn được chọn.

**TODO:** cho phép, sắp xếp thư mục một cách thuận tiện.



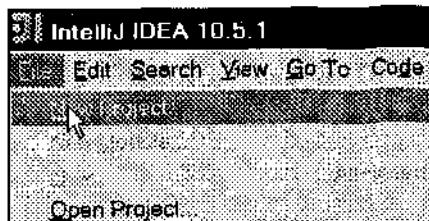
Dưới đây là bảng giới thiệu từng công cụ.

Biểu tượng	Phím tắt	Chức năng
	Ctrl + Alt + Up	Điều hướng đến mục trước.
	Ctrl + Alt + Down	Điều hướng đến mục tiếp theo.
	F1	Trợ giúp.
	Ctrl + Add hoặc Ctrl + Equals	Mở rộng tất cả các nút.
	Ctrl + Subtract hoặc Ctrl + Minus	Thu gọn tất cả các nút.
		Chuyển đổi các Autoscroll sang chế độ nguội.
		Lựa chọn bộ lọc từ danh sách.
	Ctrl + D	Hiển thị theo nút hoặc module tương ứng.
	Ctrl + P	Hiển thị theo gói tương ứng.

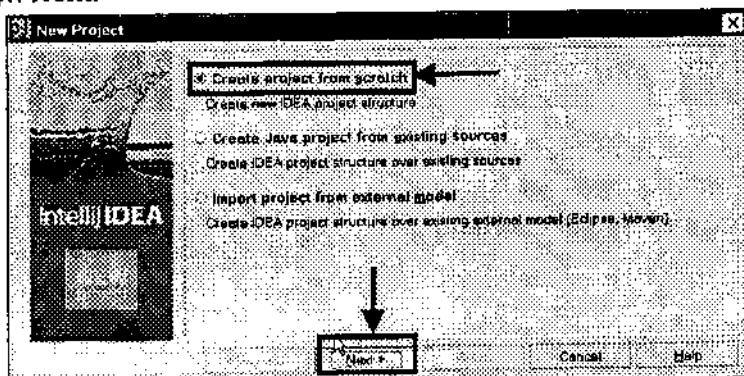
	Ctrl + F hay Alt + F3	Các nút hiển thị theo danh sách ngang bằng nhau.
--	-----------------------	--

### Cách tạo mới một dự án kiểm tra.

**Bước 1:** khởi động chương trình IntelliJ IDEA 10.5.1. Chọn File > New Project để tạo một cuộc kiểm tra mới.

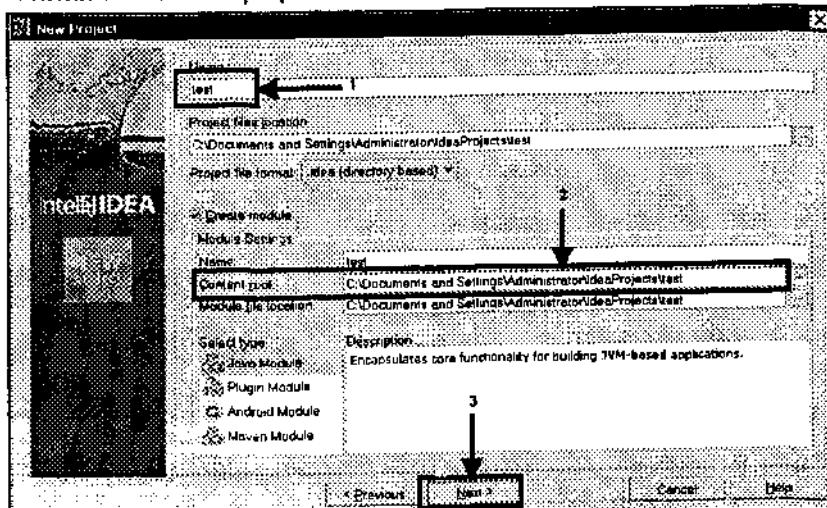


**Bước 2:** đánh dấu check vào dòng Create project from scratch sau đó nhấp chọn Next.



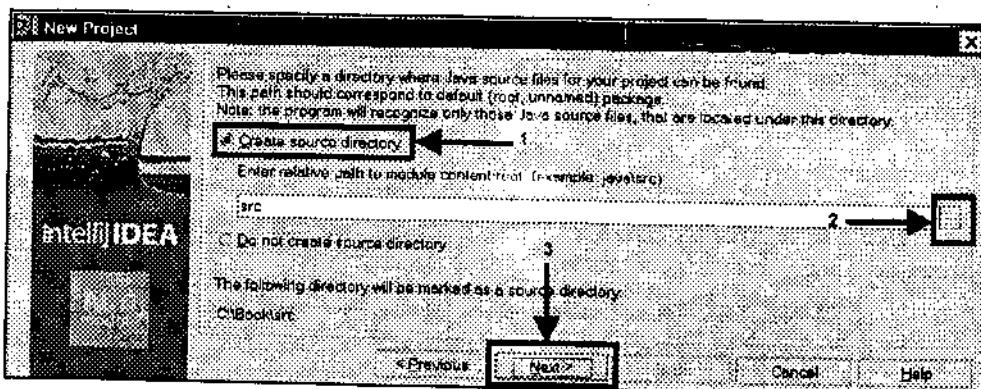
Chọn thư mục mặc định, bạn có thể tạo lại thư mục gốc.

**Bước 3:** trong ô name đặt tên là test, chọn thư mục gốc trong ô Content root và nhấn Next để tiếp tục

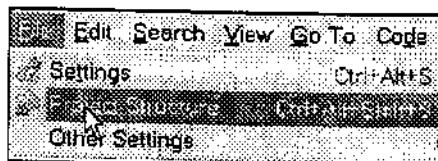


**Bước 4:** thực hiện các bước theo thứ tự được đánh số như trong hình.

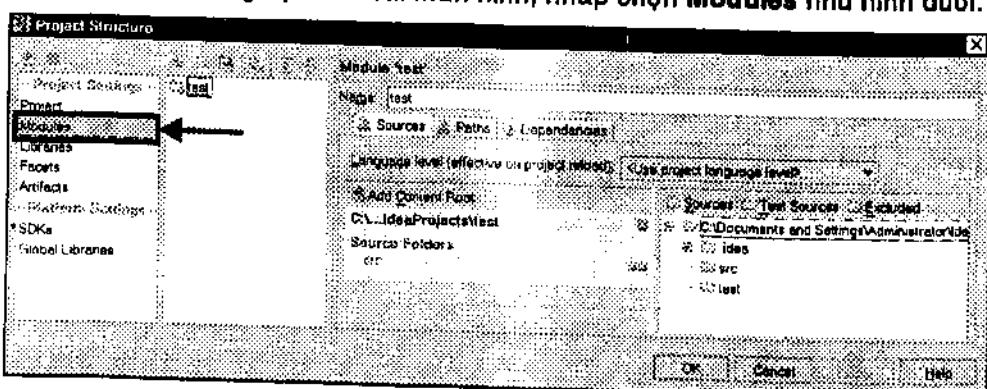
- 1 Đánh dấu check vào trước mục **Create source directory**.
- 2 Chọn đường dẫn tương đối đến thư mục gốc.
- 3 Nhấn **Next** để tiếp tục tạo mới bài kiểm tra.



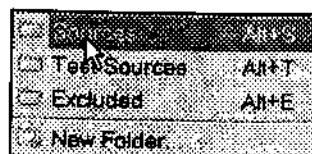
**Bước 5:** chọn **File > Project Structure** để mở một dự án cần thiết lập. Cửa sổ **Project Structure** xuất hiện



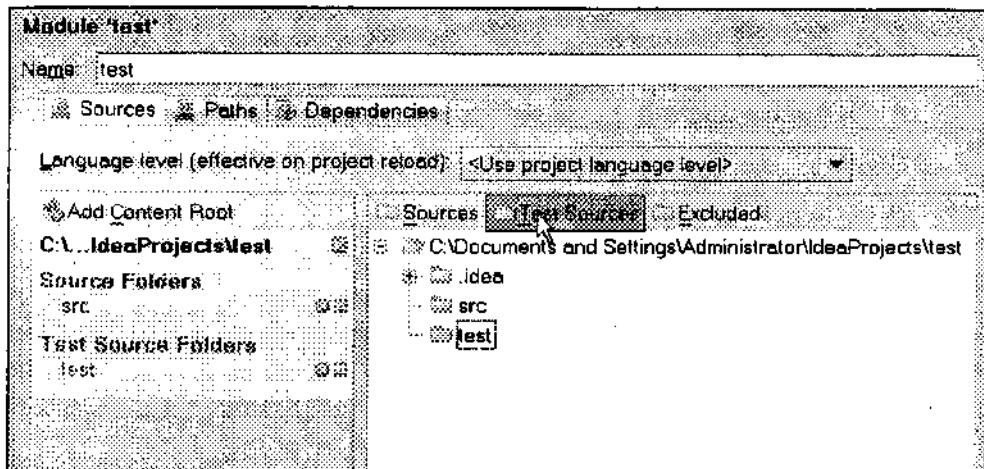
**Bước 6:** trong cột bên trái màn hình, nhấp chọn **Modules** như hình dưới.



**Bước 7:** nhấp chuột phải vào thư mục thử nghiệm mà bạn tạo ra trong cây thư mục ở phía bên phải của hộp thoại. Các chức năng hỗ trợ test xuất hiện như hình trang bên.

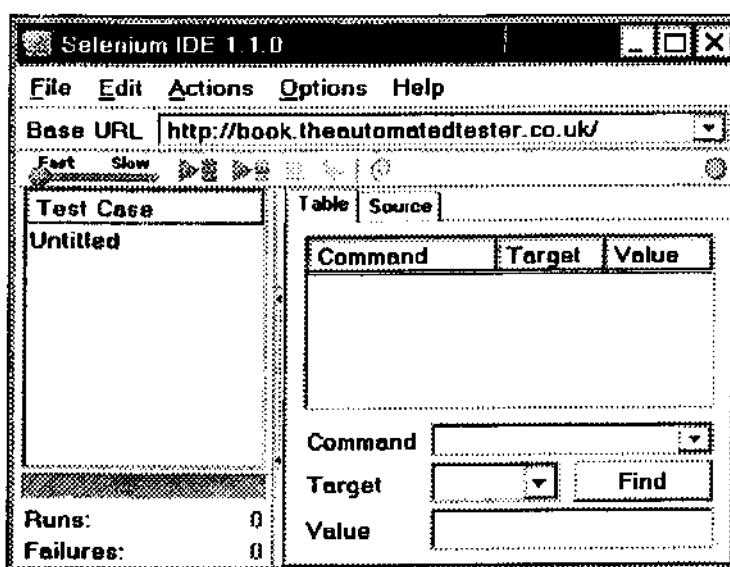
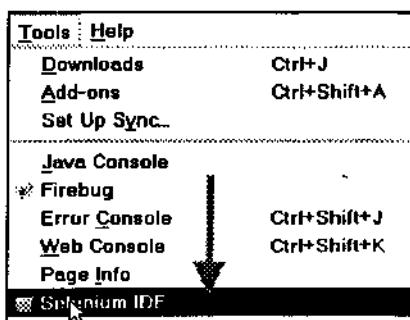


**Bước 8:** nhấp vào nút **Test Source**, thư mục kiểm tra sẽ chuyển sang màu xanh như hình trang bên.



**Bước 9:** Chọn Tools > Selenium IDE khởi động ứng dụng Selenium IDE.

Cửa sổ giao diện làm việc chương trình Selenium IDE xuất hiện như hình dưới.

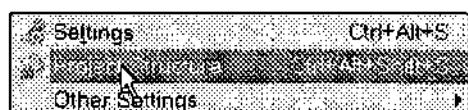


**Bước 10:** tiến hành kiểm tra một đoạn HTML sau đó lưu lại với tên file testcase1.

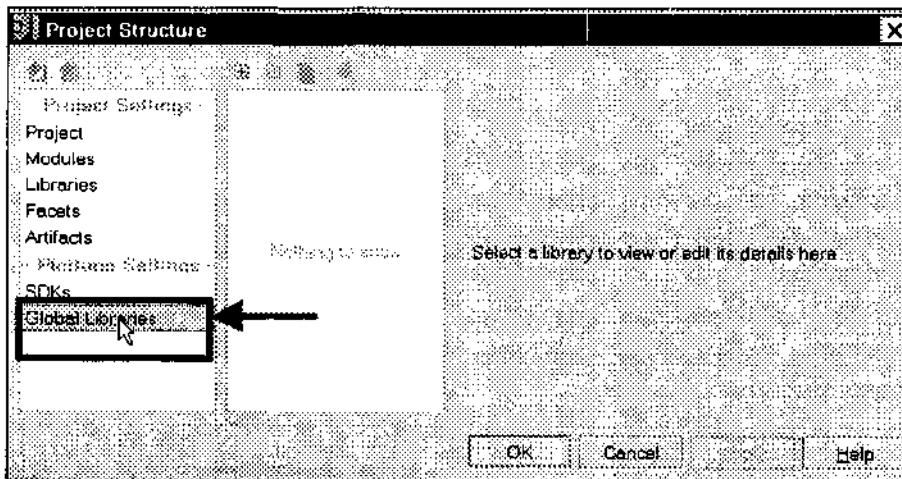
Nếu có sẵn bạn có thể lấy file đã thực hiện kiểm tra trước đó để thực hiện cuộc kiểm tra.

### 1 Kiểm tra ứng dụng trên web “book.theautomatedtester.co.uk”.

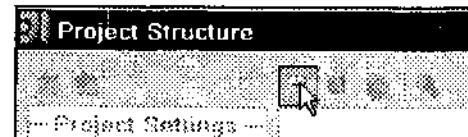
**Bước 1:** trên cửa sổ giao diện làm việc chương trình IntelliJ IDEA chọn File > Project Structure.



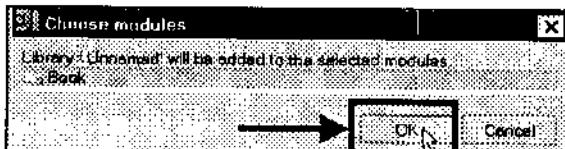
### Bước 2: nhấp chọn Global Libraries như hình dưới



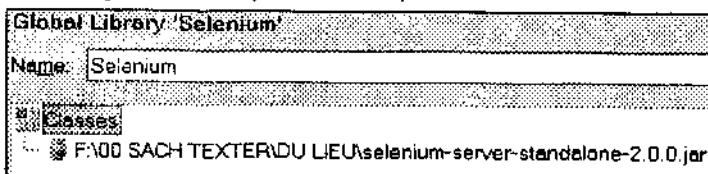
**Bước 3:** nhấp chọn biểu tượng dấu thập nằm ở phía trên gần thanh tiêu đề để tạo một thư viện mới.



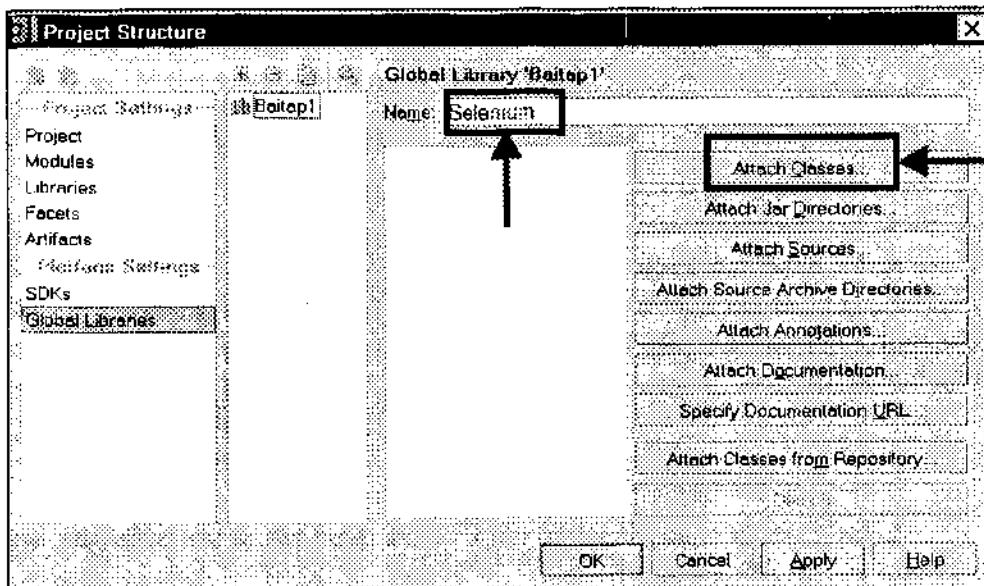
Cửa sổ **Choose modules** xuất hiện, chọn thư mục đặt thư viện khi được tạo mới, nhấn OK để chấp nhận như hình dưới.



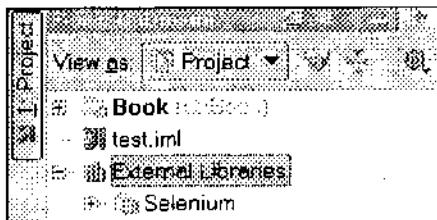
### Bước 4: trong ô Name đặt tên thư viện là Selenium.



Nhấp chọn nút Attach Classes để thêm file selenium.jar vào chương trình IntelliJ IDEA.

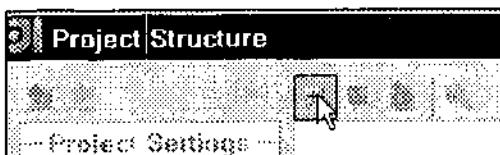


Nhấn chọn tab **Project**, trong thư mục **Book** xuất hiện thư mục **Selenium** sau khi cài đặt như hình.

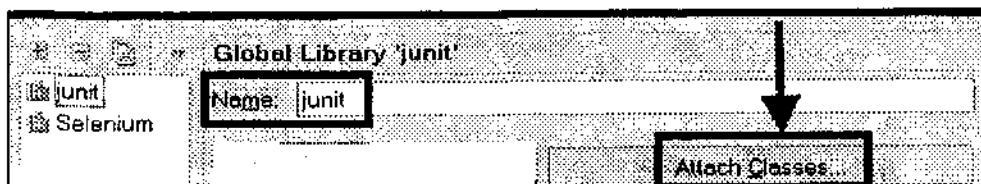


Tương tự, thực hiện các bước như trên cho việc cài đặt junit.

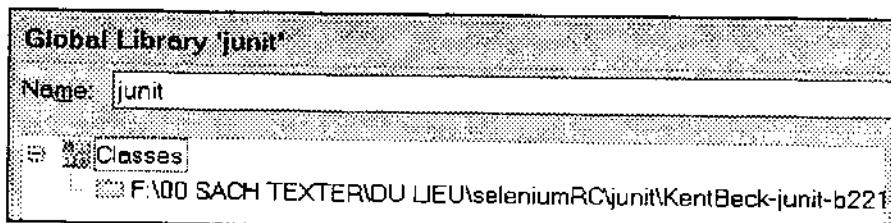
**Bước 5:** nhấp chọn biểu tượng dấu thập nằm ở phía trên gần thanh tiêu đề để tạo một thư viện mới.



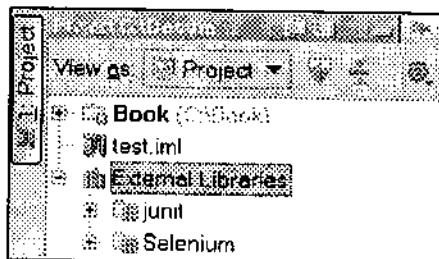
**Bước 6:** trong ô **Name** đặt tên thư viện là **junit**, nhấn chọn nút **Attach Classes** để thêm thư mục junit (đã được tải về máy trước đó) vào chương trình IntelliJ IDEA.



Sau khi nhấn nút **Attach Classes**, nhấp nút **Apply > OK** để áp dụng vào chương trình như hình trang bên.



Nhấn chọn tab **Project**, trong thư mục **Book** xuất hiện thư mục **junit** sau khi cài đặt như hình.

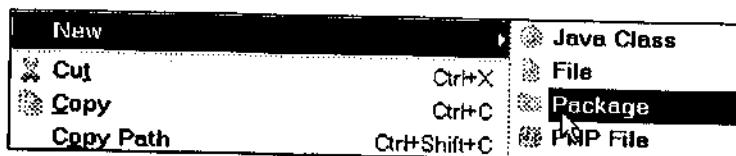


**Bước 7:** nhấp chuột phải vào tập tin Java được tạo ra bởi Selenium IDE và nhấp vào "Run" testcase1.

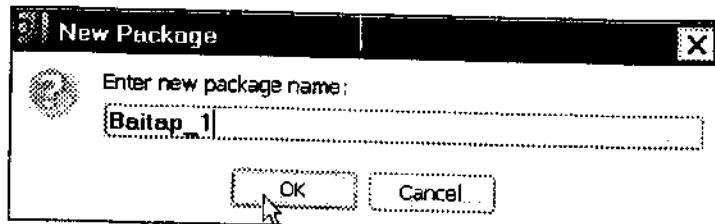
Trong phần thực hành này, bạn hãy thử nghiệm trên trình duyệt Firefox. Ngoài ra bạn vẫn có thể thực hiện trên các trình duyệt khác.

## 2 Tạo một ví dụ trên IDEA

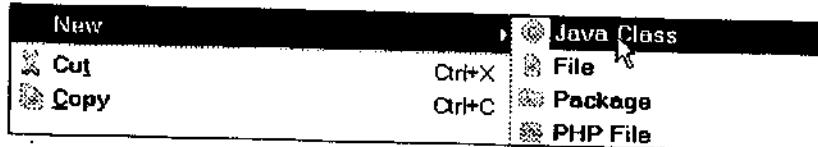
**Bước 1:** để tạo mới lớp java trong IDEA, bạn nhấn chọn tab **Project** và nhấp vào dự án muốn xây dựng. Ví dụ vào thư mục **Book[Books]**, nhấp chuột phải vào thư mục đường dẫn **src > New > Package** để tạo mới gói kiểm tra như hình.



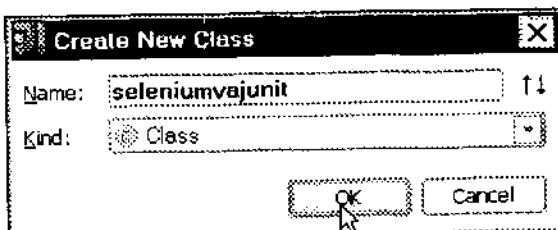
Hộp thoại **New Package** xuất hiện, đặt tên gói là **Baitap\_1** sau đó nhấn **OK** để tạo mới.



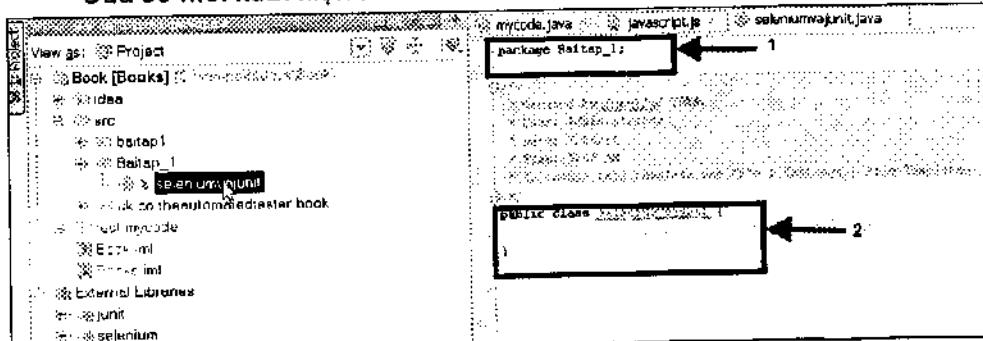
Chọn gói **Baitap1** > nhấp chuột phải > **New > JavaClass** để tạo mới lớp java.



**Bước 2:** hộp thoại Create New Class xuất hiện, trong khung Name bạn điền tên lớp là seleniumvajunit sau đó nhấp OK.



Cửa sổ mới xuất hiện như hình dưới.



- 1 Gói đang tạo code kiểm tra.
- 2 Lớp cho phép tạo mới code kiểm tra.

**Bước 3:** bạn thêm thư viện selenium vào lớp java vừa mới tạo bằng đoạn code:

```
import com.thoughtworks.selenium.*;
```

Chú ý: sử dụng dấu \*; để kết thúc một câu lệnh.

```
Package Baitap_1;
import com.thoughtworks.selenium.*;
public class seleniumvajunit {
```

**Bước 4:** gọi thư viện Selenium bởi biến selenium bởi dòng lệnh:

**Selenium selenium;**

```
import com.thoughtworks.selenium.*;
public class seleniumvajunit {
    Selenium selenium;
}
```

**Bước 5:** thiết lập một selenium mới với tên hàm `setUp()`.

Đầu tiên bạn phải khởi tạo đối tượng bao gồm 4 thông số:

- Tên máy chủ điều khiển từ xa.
- Cổng chạy ứng dụng.
- Trình duyệt sử dụng.
- Tên trang web thử nghiệm.

```
import com.thoughtworks.selenium.*;
String serverHost, int serverPort, String browserStartCommand, String browserURL;
CommandProcessor processor
    processor = new DefaultCommandProcessor();
    processor.setServerHost("www.theautomatedtester.co.uk");
    processor.setServerPort(4444);
    processor.setBrowserStartCommand("chrome");
    processor.setBrowserURL("http://book.theautomatedtester.co.uk/");
    processor.setTimeout(30000);
    selenium = new DefaultSelenium(processor);
    selenium.start();
}
```

Dưới đây là đoạn code thực hiện:

```
selenium = new DefaultSelenium("localhost",4444,"*chrome",
"http://book.theautomatedtester.co.uk");
```

Kế tiếp bạn gọi phương thức `start()`.

```
public void setUp(){
    selenium= new DefaultSelenium("localhost", 4444, "*chrome", "http://book.theautomatedtester.co.uk");
    selenium.start();
    selenium.setTimeout(30000);
    selenium.setAlertHandler(new AlertHandler() {
        public void alert(String message) {
            System.out.println(message);
        }
    });
    selenium.start();
    selenium.stop();
}
```

Hàm `setUp()` thu được như hình dưới.

```
public void setUp(){
    selenium= new DefaultSelenium("localhost", 4444, "*chrome", "http://book.theautomatedtester.co.uk");
    selenium.start();
}
```

**Bước 6:** tạo một thử nghiệm mới với tiền tố `test`.

Ví dụ: `testShouldOpenChapter2LinkAndVerifyAButton()`

Bạn có thể thêm các lệnh selenium trong thử nghiệm mới tạo. Gọi phương thức `open()` để gọi đường dẫn đến trang `Chapter2`, với thời gian load trang 3000 giây khi nút `but1` được gọi.

```
public void testShouldOpenChapter2LinkAndVerifyAButton() {
    selenium.open("/");
    selenium.click("link=Chapter2");
    selenium.waitForPageToLoad("3000");
    Assert.assertTrue(selenium.isElementPresent("but1"));
}
```

### Code kiểm tra như hình dưới.

```

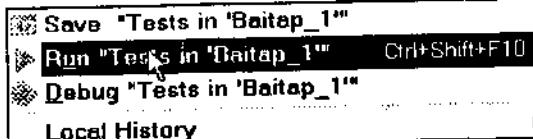
selenumvajunit.java
package Baitep_1;

import org.openqa.selenium.*;
import org.junit.Assert;
import com.thoughtworks.selenium.*;

public class Baitep_1 {
    Selenium selenium;
    public void setUP(){
        selenium= new DefaultSelenium("localhost", 4444, "chrome", "http://book.thinkautomated.com");
        selenium.start();
    }
    public void runSeleniumScript() throws Exception{
        selenium.open("/");
        selenium.click("link=Chapter 2");
        selenium.waitForPageToLoad("3000");
        Assert.assertTrue(selelium.isElementPresent("but1"));
    }
}

```

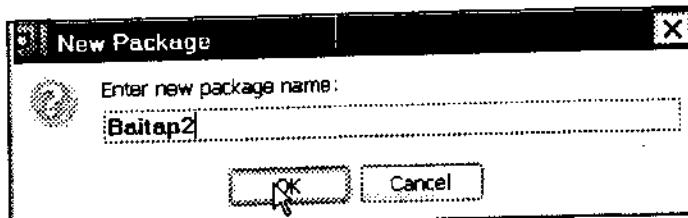
**Bước 7:** đưa chuột đến gói  
Baitep1 > nhấp chuột phải >  
Run “Tests in ‘Baitep\_1’” để  
chạy chương trình.



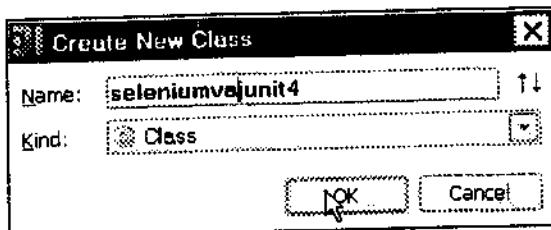
### 3 Tạo ví dụ selenium với junit4.

Junit4 là khuôn khổ kiểm tra đơn vị, sau đây là các bước hướng dẫn.

**Bước 1:**  
tạo mới gói thử  
nghiệm trong  
IDEA, đặt tên  
Baitep2 sau đó  
nhấp OK.



**Bước 2:** hộp thoại Create  
New Class xuất hiện, trong  
khung Name bạn điền tên lớp  
là seleniumvajunit4 sau đó  
nhấn OK. Giao diện xuất hiện  
như hình trang bên.



**Bước 3:** thêm thư viện selenium, junit vào trong lớp java, code được sử dụng như sau:

```
Import com.thoughtworks.selenium.*;
```

```
Import org.junit.*;
```

```
seleniumvajunit.java
package Baitap2;

import com.thoughtworks.selenium.*;
import org.junit.*;

seleniumvajunit4.java
```

**Bước 4:** khai báo biến selenium.

Selenium selenium;

```
public class seleniumvajunit4 {
    Selenium selenium;
}
```

Bạn tạo một phương pháp kiểm tra mới trong lớp **seleniumvajunit4** với đoạn code như sau:

Sử dụng chú thích **@Before** để bắt đầu đoạn code kiểm tra.

```
public class seleniumvajunit4 {
    Selenium selenium;
    @Before
    public void setUp() {
        selenium = new DefaultSelenium("localhost", 4444, "chrome", "http://huongtuts.com/test-selenium-viet-nam");
        selenium.start();
    }
}
```

**Bước 5:** tương tự như trên bạn cũng khởi tạo hàm với 4 thông số và gọi phương thức start() để khởi động Selenium.

```
public class seleniumvajunit4 {
    Selenium selenium;
    @Before
    public void setUp() {
        selenium = new DefaultSelenium("localhost", 4444, "chrome", "http://huongtuts.com/test-selenium-viet-nam");
        selenium.start();
    }
}
```

**Bước 6:** sử dụng chú thích **@Test** để bắt đầu đoạn code kiểm tra.

```

    @Test
    public void testShouldOpenChapter2LinkAndVerifyAButton(){
        selenium.open("/");
        selenium.click("link=Chapter2");
        selenium.waitForPageToLoad("3000");
        Assert.assertTrue(selenium.isElementPresent("but1"));
    }
}

```

**Bước 7:** sau khi đã khởi động được trình duyệt, bạn cần phải đóng nó lại sau khi cuộc thử nghiệm hoàn tất bằng cách tạo nên một **@After** dùng để gọi hàm **stop()**.

**@After**

```

public void tearDown(){
    selenium.stop();
}

```

**@After**

```

public void tearDown(){
    selenium.stop();
}

```

Tập tin thử nghiệm của bạn như hình.

```

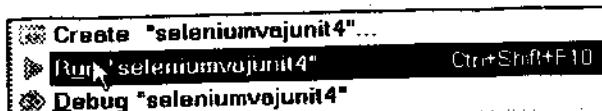
seleniumvajunit.java
seleniumvajunit4.java
package Baithap2;

import com.thoughtworks.selenium.*;
import org.junit.*;

public class seleniumvajunit4 {
    Selenium selenium;
    @Before
    public void setUp(){
        selenium = new DefaultSelenium("localhost",4444,"chrome", "http://book.tutsplusatutsgo.com.vn");
        selenium.start();
    }
    @Test
    public void testShouldOpenChapter2LinkAndVerifyAButton(){
        selenium.open("/");
        selenium.click("link=Chapter2");
        selenium.waitForPageToLoad("3000");
        Assert.assertTrue(selenium.isElementPresent("but1"));
    }
    @After
    public void tearDown(){
        selenium.stop();
    }
}

```

Nhấp Run để chạy thử nghiệm.

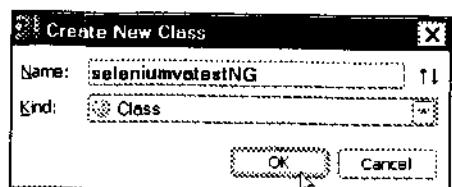


Với cuộc thử nghiệm ở trên, bạn thấy: sau khi trình duyệt khởi động, chương trình được bật lên và sau đó tắt liền. Điều này giúp cho bạn có thể bật Selenium lên một lần nữa với môi trường trình duyệt sạch cho các cuộc thử nghiệm tiếp theo.

#### 4 Tạo ví dụ Selenium với TestNG

TestNG là một thử nghiệm rất phổ biến cho Java. Nó có thể được mở rộng hơn so với JUnit4.

**Bước 1:** tạo lớp mới để viết code thử nghiệm. Trong gói **Baitap2** nhấp chuột phải chọn **New class**, trong cửa sổ **Create New Class** đặt tên lớp là **seleniumvatestNG** sau đó nhấp **OK**.



**Bước 2:** add thư viện Selenium vào lớp.

```
seleniumvajunit.java   seleniumvajunit4.java  seleniumvatestNG.java
package Baitap2;

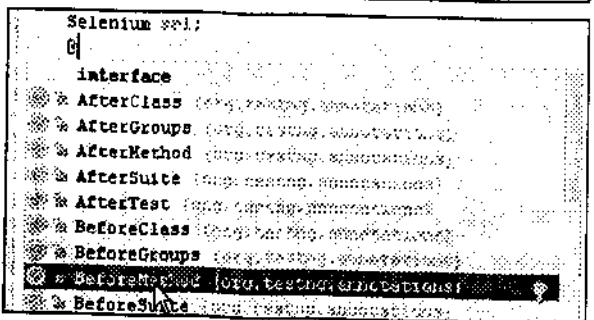
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;

public class seleniumvatestNG {
}
```

**Bước 3:** trong lớp **seleniumvatestNG**, bạn khởi tạo biến **sel** để có thể sử dụng thư viện Selenium.

```
public class seleniumvatestNG {
    Selenium sel;
}
```

**Bước 4:** sử dụng chú thích bởi **@BeforeMethod()** để đảm bảo vòng kiểm tra sẽ chạy.



**Bước 5:** tương tự như trên, bạn cũng khởi tạo hàm với 4 thông số và gọi phương thức start() để khởi động Selenium.

```
public class seleniumvatestNG {
    Selenium sel;
    @BeforeMethod(alwaysRun = true)
    public void setUp(){
        sel = new DefaultSelenium("localhost", 4444, "firefox", "http://book.theautomatedtester.org/easy");
        sel.start();
    }
}
```

**Bước 4:** kiểm tra trang web.

@Test

```
public void testShouldOpenTheRootOfSite(){
    sel.open("/");
}
```

**Bước 5:** tạo một đoạn chương trình mới với chú thích @AfterMethod để đóng kiểm tra.

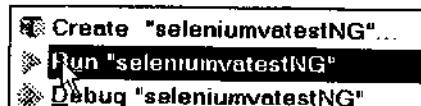
@AfterMethod

```
public void tearDown(){
    sel.stop();
}
```

Sau khi hoàn thành các bước bạn có code như sau:

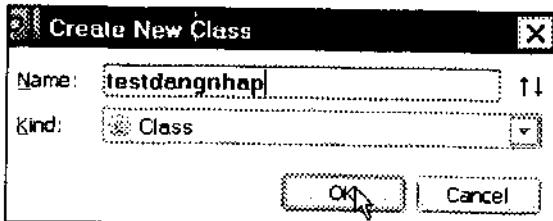
```
seleniumvajunit.java  seleniumvajunit4.java  seleniumvatestNG.java
package BaiTap2;
import com.thoughtworks.selenium.DefaultSelenium;
import com.thoughtworks.selenium.Selenium;
import org.testng.annotations.*;
public class seleniumvatestNG {
    Selenium sel;
    @BeforeMethod(alwaysRun = true)
    public void setUp(){
        sel = new DefaultSelenium("localhost", 4444, "firefox", "http://book.theautomatedtester.org/easy");
        sel.start();
    }
    @Test
    public void testShouldOpenTheRootOfSite(){
        sel.open("/");
    }
    @AfterMethod
    public void tearDown(){
        sel.stop();
    }
}
```

**Bước 6:** nhấp chuột phải vào file java và chọn Run để chạy chương trình kiểm tra.



### 5 Thủ nghịch với trang web yêu cầu đăng nhập.

**Bước 1:** trong gói **Baitap2**, bạn tạo mới lớp java có tên là **testdangnhap** sau đó nhấp OK.



**Bước 2:** add thư viện Selenium vào lớp, trong lớp **testdangnhap** khởi tạo biến sử dụng selenium có tên là **selenium**.

```

seleniumvajunit.java      seleniumvajunit4.java      testdangnhap.java

package Baitap2;

import org.junit.*;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.ie.IEDriverServer;
import org.openqa.selenium.opera.OperaDriver;
import org.openqa.selenium.phantomjs.PhantomJSDriver;
import org.openqa.selenium.safari.SafariDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import com.thoughtworks.selenium.*;

public class testdangnhap {
    Selenium selenium;
}

```

**Bước 3:** tạo các hàm bắt đầu **setUp()** và hàm kết thúc **tearDown()**.

```

public class testdangnhap {
    Selenium selenium;
    WebDriver driver;
    public void setUp(){
        selenium = new DefaultSelenium("localhost", 4444, "chrome", "http://book.theguestcoffeeshop.com");
        selenium.start();
    }
    public void tearDown(){
        selenium.stop();
    }
}

```

**Bước 4:** để kiểm tra chính xác trên một trang web, bạn cần sử dụng dòng lệnh **getTitle()** để thấy rõ tiêu đề trang và kết hợp thời gian gọi đến trang khác chậm.

Giả sử bạn điều chỉnh sai không chỉ đường link đến trang **Chapter2** thì đoạn chương trình sau sẽ điều chỉnh nó đi đúng hướng.

```

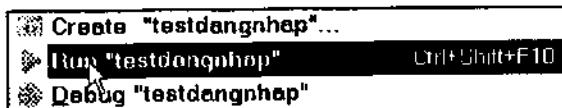
@Test
    public void testShouldOpenChapter2LinkAndVerifyAButton(){
        if (!"Page 2".equals(selenium.getTitle())){
            selenium.open("/chapter2");
            selenium.waitForPageToLoad("30000");
        }
    }
}

```

Bổ sung các đoạn code còn lại để hoàn thành bài kiểm tra.

`Assert.assertTrue(selenium.isElementPresent("but1"));`

Nhấn Run để chạy thử nghiệm.



### 6 Tạo phương thức mới thay cho các dòng lệnh.

Giả sử bạn đang kiểm tra trên một trang web duy nhất, và khi đó bạn có một vài cuộc thử nghiệm. Các bài kiểm tra có thể sử dụng một hoặc nhiều đoạn code, điều này sẽ khá khó khăn khi gặp phải trường hợp bị lỗi. Nếu muốn thay đổi nội dung trên trang web thì bạn phải di qua tất cả các bài kiểm tra. Để sửa lỗi này, bạn có thể thay đổi các dòng lệnh bằng các phương thức thay thế.

Ví dụ: cùng sử dụng một đoạn code để kiểm tra 2 sự kiện nhấn nút trên một trang web.

**Bước 1:** di chuyển đến thư mục chứa trang web.

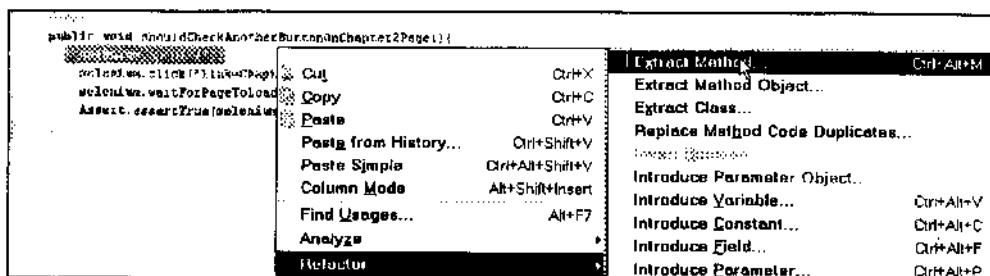
```

@Test
public void shouldCheckButtonOnChapter2Page(){
    selenium.open("/");
    selenium.click("link=Chapter2");
    selenium.waitForPageToLoad("30000");
    Assert.assertTrue(selenium.isElementPresent("but1"));
}

@Test
public void shouldCheckAnotherButtonOnChapter2Page(){
    selenium.open("/");
    selenium.click("link=Chapter2");
    selenium.waitForPageToLoad("30000");
    Assert.assertTrue(selenium.isElementPresent("verifybutton"));
}

```

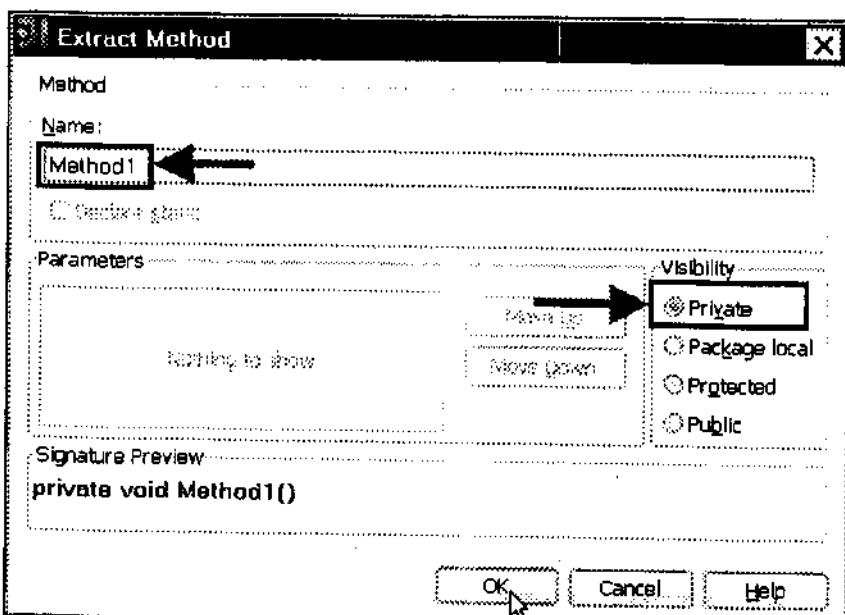
**Bước 2:** đánh dấu dòng lệnh muốn thay thế cấu trúc sau đó nhấp chuột phải và chọn Refactor > Extract Method như hình trang bên.



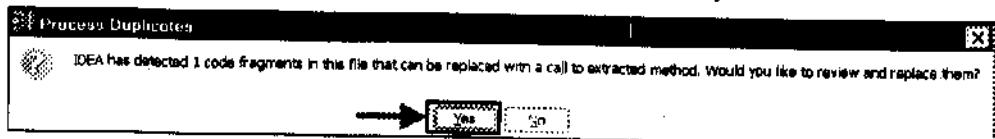
**Bước 3:** hộp thoại Extract Method xuất hiện, trong ô Name bạn điền tên cho phương thức là Method1. Trong khung Visibility, chọn chế độ sử dụng cho phương thức như:

- **Private:** chỉ sử dụng trong một hàm.
- **Package local:** gói địa phương.
- **Protected:** chọn chế độ bảo vệ.
- **Public:** các hàm khác có thể sử dụng.

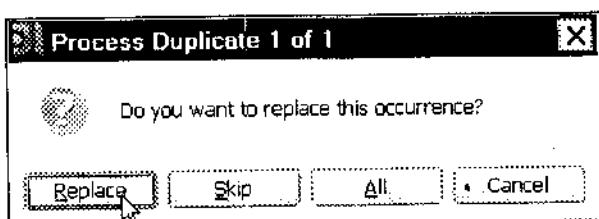
Nhập OK để tạo mới.



**Bước 4:** hộp thoại Process Duplicates thông báo dòng lệnh đã được thay thế bởi một phương thức mở rộng, nhấn Yes để thay thế.



**Bước 5:** hộp thoại Process Duplicates 1 of 1 thông báo hỏi bạn muốn thay thế dòng lệnh không, nhấp Replace để thay thế.



Phương thức thay thế cho dòng lệnh được phát sinh như hình dưới.

```

public void shouldCheckAnotherButtonOnChapter2Page(){
    Method1();
    selenium.click("link=Chapter2");
    selenium.waitForPageToLoad("30000");
    Assert.assertTrue(selenium.isElementPresent("verifybutton"));
}

private void Method1() {
    selenium.open("/");
}

```

### 7 Kiểm tra biến cookie trên một trang web.

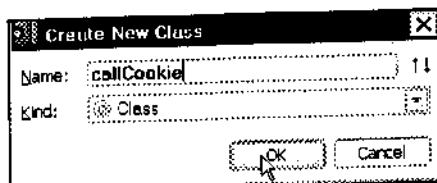
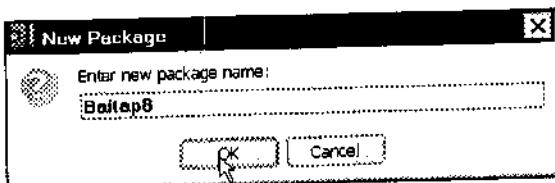
Giả sử bạn có sử dụng phương thức lưu trữ số lần truy cập trang web bởi một người dùng bằng biến cookie. Kế tiếp, bạn tạo một phương thức kiểm tra để có thể tải trang và nhận giá trị cookie. Sử dụng giao thức `getCookie(name)` lấy về biến cookie. Giao diện trang Chapter8 khi chạy trên trình duyệt web như hình dưới.



Bây giờ bạn thực hiện các bước kiểm tra trên trang web Chapter8.

**Bước 1:** tạo mới gói kiểm tra, nhấp chuột phải vào `src > New > Package`. Trong cửa sổ `New Package` đặt tên `Baitap8` sau đó nhấp OK.

**Bước 2:** tạo mới lớp java có tên là `callCookie` sau đó nhấp OK.



**Bước 3:** thêm thư viện Selenium vào lớp Java sử dụng như hình dưới.

```

class callCookie {
    public void main() {
        // code here
    }
}

import com.thoughtworks.selenium.*;

```

**Bước 4:** tạo các hàm bắt đầu **setUp()** và hàm kết thúc **tearDown()**.

Khai báo biến để sử dụng thư viện Selenium.

Sử dụng chú thích **@Before** và **@After** cho mỗi hàm khởi tạo.

```

public class visitorCount {
    Selenium selenium;
    @Before
    public void setUp(){
        selenium = new DefaultSelenium("localhost", 4444, "*chrome", "http://book.tutorialspoint.com/testng/index.htm");
        selenium.start();
    }
    @After
    public void tearDown(){
        selenium.stop();
    }
}

```

**Bước 5:** tạo một thử nghiệm với tên hàm là **ShouldGetACookie()** sau đó sử dụng phương thức **getCookieByName("visitorCount")**. Trong đó **visitorCount** là giá trị mà bạn cần nhận được.

Code kiểm tra như hình dưới.

```

@Test
public void ShouldGetACookie(){
    selenium.open("/chapter8");
    String cookie = selenium.getCookieByName("visitorCount");
    Assert.assertEquals("Should be 1","1",cookie);
    selenium.open("/chapter8");
    cookie = selenium.getCookieByName("visitorCount");
    Assert.assertEquals("Should be 2","2",cookie);
}

```

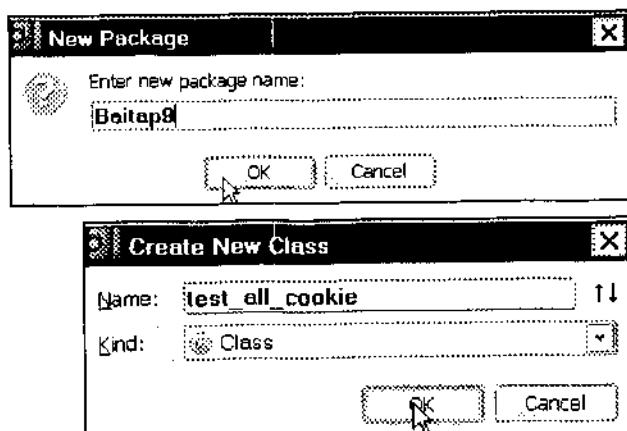
**8 Kiểm tra biến cookie trên tất cả các trang web.**

Các trang web sẽ lưu trữ các thông tin khác nhau trên các cookie khác nhau. Chính vì vậy khi sử dụng phương thức **getCookie()**, giá trị trả về là một chuỗi tập tin cookie được ngăn cách nhau bởi dấu chấm phẩy.

Các bước thử nghiệm được thực hiện tương tự ở phần 7:

**Bước 1:** tạo mới gói kiểm tra, nhấp chuột phải vào **src > New > Package.** Đặt tên **Baitap9** sau đó nhấp **OK**.

**Bước 2:** tạo mới lớp java có tên là **test\_all\_cookie** sau đó nhấp **OK**.



**Bước 3:** thêm thư viện selenium và junit vào lớp java sử dụng như hình dưới.

```

callCookie.java | test_all_cookie.java
-----
package Baitap9;

import com.thoughtworks.selenium.*;
import org.junit.*;

public class test_all_cookie {
}

```

**Bước 4:** tạo các hàm bắt đầu **setUp()** và hàm kết thúc **tearDown()**.

Khai báo biến để sử dụng thư viện selenium.

Sử dụng chú thích **@Before** và **@After** cho mỗi hàm khởi tạo.

```

public class test_all_cookie {
    Selenium selenium;
    @Before
    public void setUp(){
        selenium = new DefaultSelenium("localhost", 4444, "firefox", "http://zoonk.thetestforsite.com.vn/");
        selenium.start();
    }

    @After
    public void tearDown(){
        selenium.stop();
    }
}

```

**Bước 5:** tạo hàm kiểm tra là **should GetAllCookiesOnThePage()** sau đó sử dụng phương thức **getCookie()**.

Code kiểm tra như hình dưới.

```
Given
public void should GetAllCookiesOnThePage(){
    selenium.open("/Chapter8");
    selenium.click("secondCookie");
    String[] cookies = selenium.getCookie().split(";");
    Assert.assertEquals("Should be 2 cookies", 2, cookies.length);
    Assert.assertEquals("Should be " + anyValue(), anyValue(), selenium.getCookieByName("secondCookie"));
}
```

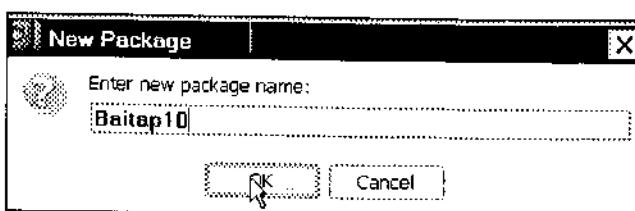
### 9 Xóa biến cookie trên trang web.

Điều này được sử dụng khi người dùng mong muốn các giá trị cookie không còn lưu trên trang web. Bạn có thể áp dụng điều này để làm sạch trình duyệt khi thực hiện nhiều lần các dợt kiểm tra trên 1 trang web. Để xóa một cookie, bạn gọi phương thức `deleteCookie`.

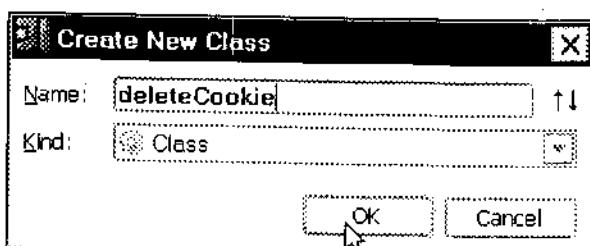
Các bước kiểm tra thực hiện như sau: vẫn thực hiện trên trang web:

<http://book.theautomatedtester.co.uk/chapter8>.

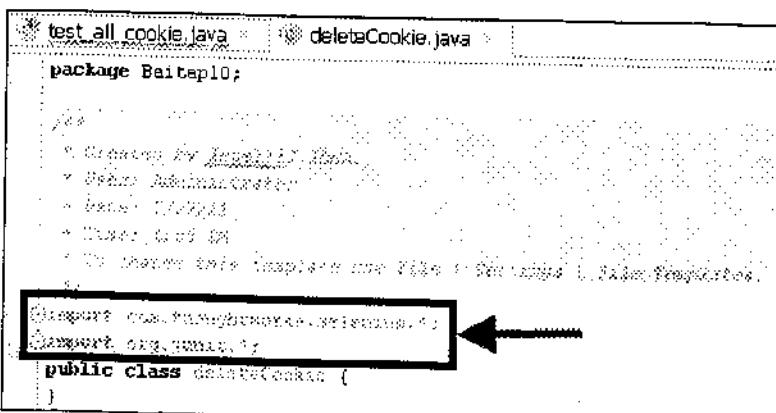
**Bước 1:** tạo mới gói kiểm tra, nhấn chuột phải vào `src > New > Package`. Đặt tên `Baitap10` sau đó nhấp **OK** trong cửa sổ mới.



**Bước 2:** tạo lớp java mới có tên là `deleteCookie` sau đó nhấn **OK**.



**Bước 3:** thêm thư viện `selenium` và `junit` vào lớp java sử dụng như hình minh họa.



**Bước 4:** tạo các hàm bắt đầu **setUp()** và hàm kết thúc **tearDown()**.

Khai báo biến để sử dụng thư viện selenium.

Sử dụng chú thích **@Before** và **@After** cho mỗi hàm khởi tạo.

```
public class Chapter8Test {
    Selenium selenium;
    @Before
    public void setUp(){
        selenium = new DefaultSelenium("localhost", 4444, "chrome", "http://book.theautomatedtester.co.uk/");
        selenium.start();
    }
    @After
    public void tearDown(){
        selenium.stop();
    }
}
```

**Bước 5:** các tập tin cookie được tạo trên trang web với đường dẫn "**path:/**". Để xóa các tập tin cookie, bạn thực hiện như sau:

```
selenium.deleteCookie("visitorCount","path=/");
```

Kế tiếp bạn sẽ mở lại trang Chapter8, lấy về giá trị của visitorCount và xác nhận giá trị là 1.

Tạo hàm kiểm tra có tên **shouldDeleteACookieOnThePage()**.

Code kiểm tra như hình dưới.

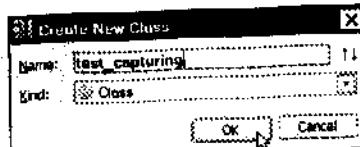
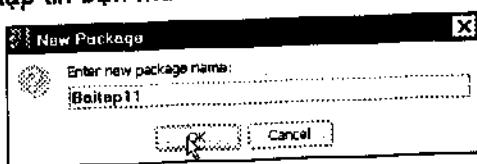
```
public void shouldDeleteACookieOnThePage(){
    selenium.open("/Chapter8");
    selenium.deleteCookie("visitorCount","path=/");
    selenium.open("/Chapter8");
    String cookie = selenium.getCookieByName("visitorCount");
    Assert.assertEquals("Should be 1","1",cookie);
}
```

## 10 Chụp màn hình trong Selenium.

Chụp màn hình trong Selenium được giới hạn trong trình duyệt Mozilla Firefox và Internet Explorer do chúng có các thư viện Selenium. Để chụp toàn bộ các sự kiện xảy ra trên màn hình máy tính, bạn sử dụng phương thức **captureScreenshot(file)**, với file là tên tập tin bạn muốn lưu.

**Bước 1:** tạo mới gói kiểm tra, nhấn chuột phải vào **src > New > Package**. Đặt tên **Baitap11** sau đó nhấn **OK**.

**Bước 2:** tạo mới lớp java có tên là **test\_capturing** sau đó nhấn **OK**.



**Bước 3:** thêm thư viện selenium và junit vào lớp java sử dụng như hình minh họa.

```
test_capturing.java
1 package Baitap11;
2
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.chrome.ChromeDriver;
5 import org.junit.Test;
6 import org.junit.Before;
7 import org.junit.After;
8
9
10 import java.util.concurrent.TimeUnit;
11 import org.openqa.selenium.JavascriptExecutor;
12
13 public class test_capturing {
14 }
```

**Bước 4:** tạo các hàm bắt đầu `setUp()` và hàm kết thúc `tearDown()`.

Khai báo biến để sử dụng thư viện selenium

Sử dụng chú thích `@Before` và `@After` cho mỗi hàm khởi tạo.

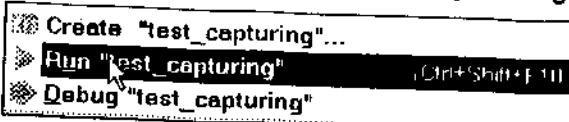
```
public class test_capturing {
    WebDriver selenium;
    @Before
    public void setUp(){
        selenium = new DefaultSelenium("localhost", 4444, "chrome", "http://book.theautomatedtester.co.uk/");
        selenium.start();
    }
    @After
    public void tearDown(){
        selenium.stop();
    }
}
```

**Bước 5:** tạo hàm kiểm tra có tên `shouldTakeAScreenShot()`, gọi phương thức `open()` trả đến trang chapter8. Kế tiếp sử dụng `captureScreenshot()` để chụp và lưu màn hình với tên file là `003.png`.

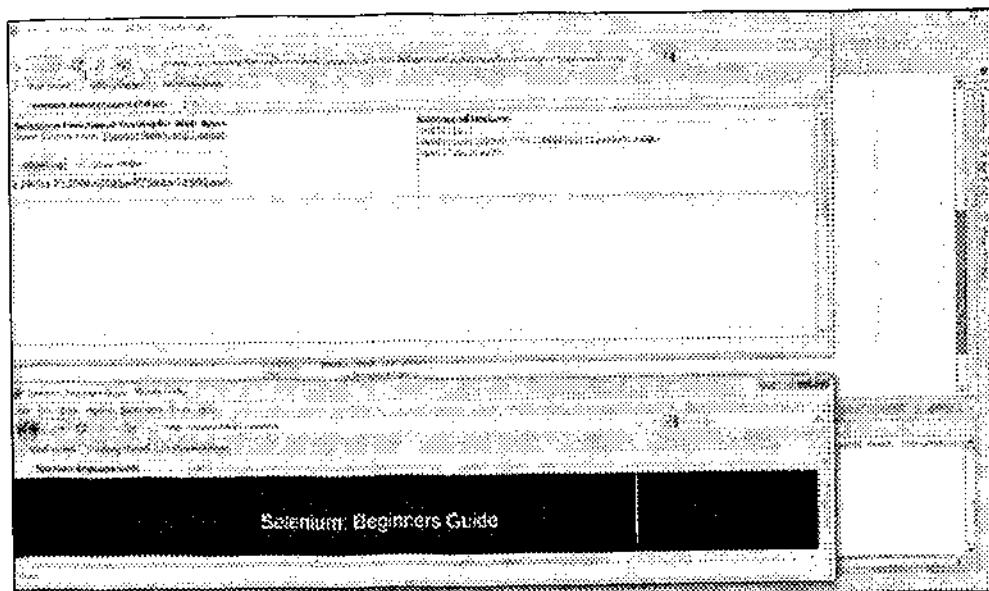
Code xử lý như hình dưới.

```
Class
public void shouldTakeAScreenShot(){
    selenium.open("/chapter8");
    selenium.captureScreenshot("003.png");
}
```

**Bước 6:** nhấp chọn Run “`test_capturing`” để chạy chương trình kiểm tra.



Kết quả ảnh chụp màn hình thu được như hình trang bên.



Selenium còn cung cấp chức năng lưu trữ ảnh ở dạng chuỗi. Khi đó bạn cần sử dụng phương thức `captureScreenshotToString()`. Các bước thực hiện kiểm tra từ bước 1 đến bước 4 tương tự như trên. Sang bước 5, bạn thay bởi đoạn code như sau:

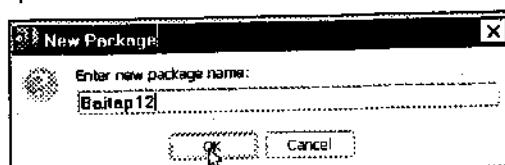
```
public void shouldTakeAScreenshotAndReturnAScreenshot() {
    selenium.open("/chapter8");
    String screenshot = selenium.captureScreenshotToString();
    System.out.println(screenshot);
}
```

## 11 Chụp một phần màn hình máy tính.

Giả sử bạn đang làm việc với một trang web có chiều dài lớn hơn chiều dài màn hình máy vi tính. Nếu muốn xem hết thì bạn cần đến sự hỗ trợ của thanh trượt dọc. Tuy nhiên, khi sử dụng cuộc gọi nó sẽ không hỗ trợ điều này cho bạn. Chính vì thế khi lưu ảnh sẽ không có ảnh toàn bộ trang đang thử nghiệm. Để khắc phục điều này, Selenium đã hỗ trợ phương thức `captureEntirePageScreenshot` qua 2 tham số. Tham số đầu tiên dùng để lưu tập tin, tham số thứ hai dùng để thay đổi hình nền của HTML.

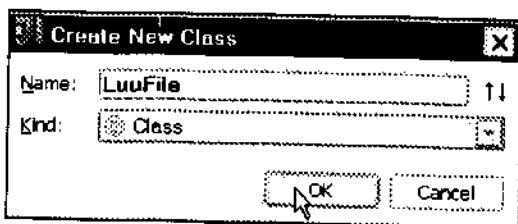
Các bước kiểm thử được thực hiện như sau:

**Bước 1:** tạo mới gói kiểm tra, nhấp chuột phải vào `src > New > Package`. Đặt tên `Baitap12` sau đó nhấp OK.



**Bước 2:** tạo mới lớp Java có tên là LuuFile sau đó nhấp OK.

**Bước 3:** thêm thư viện selenium và junit vào lớp Java sử dụng như hình minh họa.



LuuFile.java

```
package BaiTap12;

// Created by JUnit 4.12 IDEA
// User: KhanhNhat
// Date: 1/23/23
// Time: 9:34 AM
// To change this template, choose Tools | Templates | File | Open Source.
```

```
import com.thoughtworks.selenium.*;
import org.junit.*;

@SeleniumSetup(url = "http://check.themutantkittener.co.uk/")
public class LuuFile {
    @Test
    public void test() {
        // Test code here
    }
}
```

**Bước 4:** tạo các hàm bắt đầu `setUp()` và hàm kết thúc `tearDown()`.

Khai báo biến để sử dụng thư viện selenium.

Sử dụng chú thích `@Before` và `@After` cho mỗi hàm khởi tạo.

```
public class LuuFile {
    Selenium selenium;
    @Before
    public void setUp(){
        selenium = new DefaultSelenium("localhost", 4444, "chrome", "http://check.themutantkittener.co.uk/");
        selenium.start();
    }

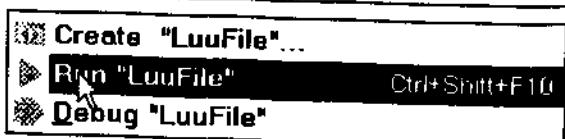
    @After
    public void tearDown(){
        selenium.stop();
    }
}
```

**Bước 5:** tạo hàm kiểm tra có tên `shouldTakeAScreenShotEntirePage()`, gọi phương thức `open()` trả đến trang chapter8 và lưu file màn hình chụp được.

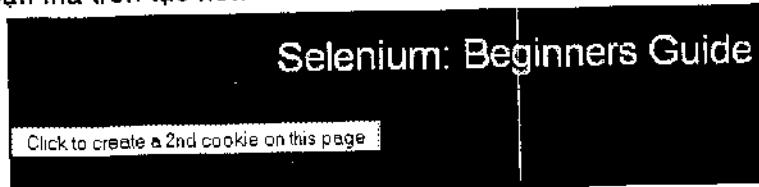
Code xử lý như hình dưới:

```
public void shouldTakeAScreenShotEntirePage(){
    selenium.open("/Chapter8");
    selenium.captureEntirePageScreenshot("/home/autotesteditester/entirepage.png");
}
```

**Bước 6:** nhấn chọn Run "LuuFile" để chạy chương trình kiểm tra.



Đoạn mã trên tạo nên ảnh như sau:

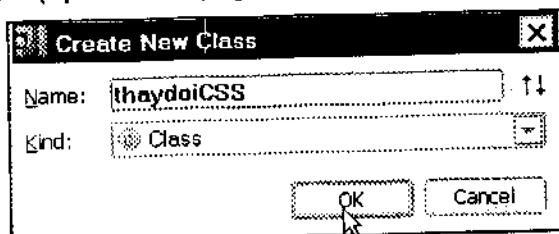


Kế tiếp, bạn viết code thay đổi giao diện của HTML thông qua việc xử lý mã CSS. Sử dụng phương thức:

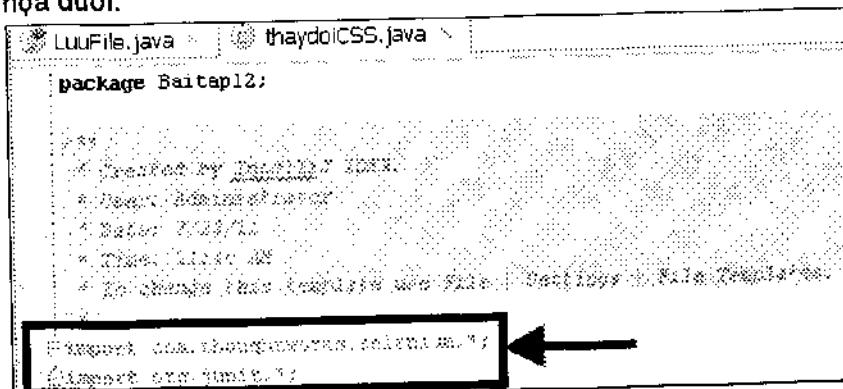
`captureEntirePageScreenshot("/path/to/file.png","background=#ccffdd");`

#### Bước 1: trong gói

Baitap12 bạn tạo mới lớp java có tên là **thaydoiCSS** sau đó nhấn OK.



**Bước 3:** thêm thư viện selenium và junit vào lớp java sử dụng như hình minh họa dưới.



**Bước 4:** tạo các hàm bắt đầu **setUp()** và hàm kết thúc **tearDown()**.

Khai báo biến để sử dụng thư viện selenium

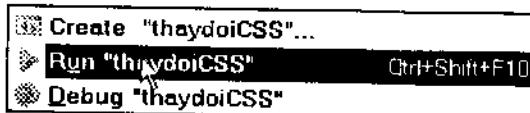
Sử dụng chú thích **@Before** và **@After** cho mỗi hàm khởi tạo.

```
public class thaydoiCSS {
    Selenium selenium;
    WebDriver driver;
    public void setUp(){
        selenium = new DefaultSelenium("localhost", 4444, "firefox", "http://www.thinkandtestwithselenium.com/");
        selenium.start();
    }
    @After
    public void tearDown(){
        selenium.stop();
    }
}
```

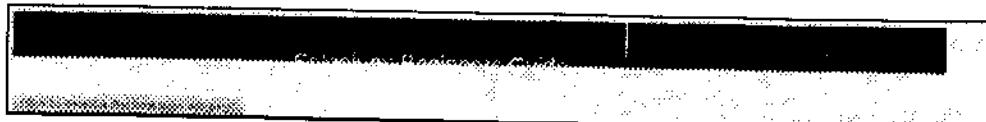
**Bước 5:** tạo hàm kiểm tra có tên:

shouldTakeAScreenShotEntirePageChangingBackground(), gọi phương thức open() trả đến trang chapter8 và thay đổi giao diện trang web bởi CSS như hình dưới.

```
public void shouldTakeAScreenShotEntirePageChangingBackground()
{
    selenium.open("chapter8");
    selenium.captureEntirePageScreenshot("chapter8/chapter8EntirePageChanged.png", "background:#cccccc");
}
```

**Bước 6:** nhấn chọn Run “thaydoiCSS” để chạy chương trình kiểm tra.

Đoạn mã trên tạo nên ảnh như sau:

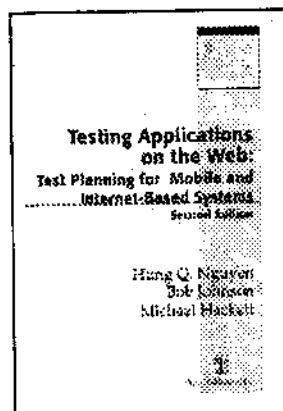


**PHỤ LỤC****GIỚI THIỆU SÁCH KIỂM THỬ PHẦN MỀM**

Sách liên quan đến chủ đề kiểm thử phần mềm tiếng Việt hầu như không có, chỉ có tài liệu Testing Applications on the web: Test Planning for Mobile and Internet-Based System của các tác giả Hung Q. Nguyen, Bob Johnson và Michael Hackett 3 chuyên gia về lĩnh vực kiểm thử phần mềm trên thế giới đã được dịch sang tiếng Việt.

Trong đó tác giả Hung Q. Nguyen (Nguyễn Quốc Hùng) là một trong những người tiên phong trong lĩnh vực kiểm thử và là nhà sáng lập công ty Logic Gear (công ty hàng đầu trong lĩnh vực kiểm thử phần mềm).

Bản dịch tiếng Việt sách Testing Applications on the web: Test Planning for Mobile and Internet-Based System gồm 2 tập.



Nội dung của bộ sách cung cấp kiến thức và kỹ thuật ứng dụng về công nghệ web từ các giải pháp thương mại điện tử, điện thoại thông minh và các thiết bị di động...

- Tập 1 dành cho những ai bắt đầu làm quen với kiểm thử phần mềm.
- Tập 2 sẽ tập trung vào các công cụ kiểm thử cụ thể.

Hiện 2 tập này không thấy bán tại các nhà sách.

Bạn đọc có thể tham khảo những bài viết về phần cứng cũng như phần mềm liên quan đến kiểm thử phần mềm tại nhiều trang web trên mạng.

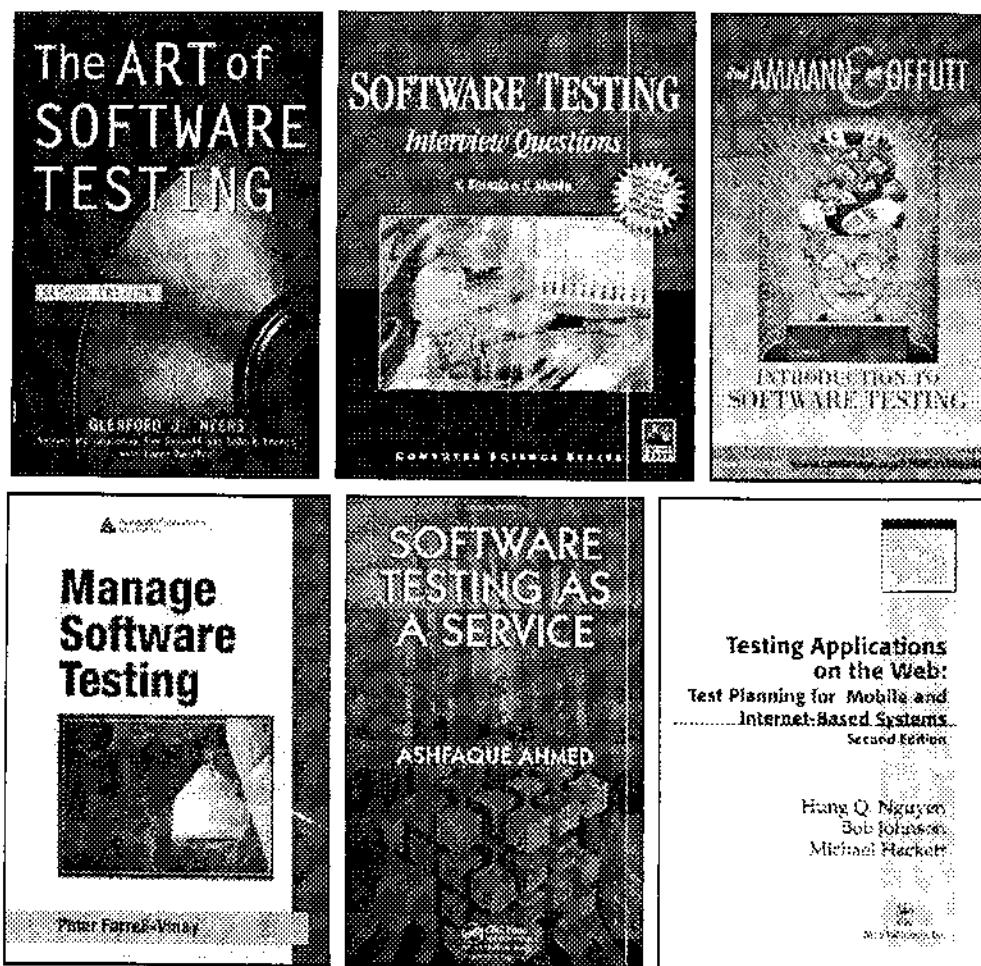
Một trong những trang web lớn (tiếng Việt) các bạn nên tham khảo là:

- <http://www.testingvn.com>.
- <http://www.vietnamesetestingboard.org/>
- <http://www.logigear.com/>

Các bạn có thể tham khảo bài viết cách tải dữ liệu (ebook, phim học, chương trình) với utorrent do tủ sách STK biên soạn để tải về các tài liệu, chương trình, phim liên quan đến kiểm thử phần mềm.

- <http://www.mediafire.com/view/?4mq6yqw8xz8hf!>

Hình dưới là một số sách các bạn có thể tải về với chương trình utorrent để tham khảo.



Các bạn cũng có thể tải các tài liệu trên theo đường dẫn sau:

<http://www.mediafire.com/myfiles.php#myfiles>

# MỤC LỤC

# KIỂM THỬ PHẦN MỀM

NỘI DUNG	TRANG
GIỚI THIỆU:	3
CHƯƠNG 1: TỔNG QUAN VỀ PHẦN MỀM	7
CHƯƠNG 2: TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM	29
CHƯƠNG 3: CHIẾN LƯỢC KIỂM THỬ	17
CHƯƠNG 4: CÁC GIAI ĐOẠN KIỂM THỬ	23
CHƯƠNG 5: CÁC KỸ THUẬT KIỂM THỬ	29
CHƯƠNG 6: KỸ THUẬT KIỂM THỬ HỘP TRẮNG	35
CHƯƠNG 7: KỸ THUẬT KIỂM THỬ HỘP ĐEN	51
CHƯƠNG 8: XÂY DỰNG TEST CASE	63
CHƯƠNG 9: ỨNG DỤNG KIỂM THỬ	79
CHƯƠNG 10: CHƯƠNG 10: KIỂM THỬ VỚI QUICKTEST PROFESSIONAL 10.0	87
CHƯƠNG 11: BÀI TẬP KIỂM THỬ WEBSITE	125
CHƯƠNG 12: LÀM VIỆC VỚI ỨNG DỤNG	131
CHƯƠNG 13: KIỂM THỬ VỚI JUNIT 4.9	179
CHƯƠNG 14: KIỂM THỬ VỚI SELENIUM 1.0	197
CHƯƠNG 15: KIỂM THỬ WEBSITE VỚI SELENIUM IDE	207
CHƯƠNG 16: KIỂM THỬ WEBSITE VỚI SELENIUM RC	251
CHƯƠNG 17: GIỚI THIỆU SÁCH KIỂM THỬ PHẦN MỀM	301
MỤC LỤC	303

# KIỂM THỬ PHẦN MỀM

## TRẦN TƯỜNG THỤY – PHẠM QUANG HIẾN

*Chịu trách nhiệm xuất bản*

**Nguyễn Thị Thu Hà**

**Biên tập:** Phan Kim Thương, Lâm Thị Mai

**Trình bày:** Trần Tường Thụy

**Thiết kế bìa:** Công ty TNHH Thương mại STK

---

## NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

**Trụ sở:** 18 Nguyễn Du, TP. Hà Nội

Điện thoại: 04-3577 2139, 3577 2140; Fax: 04-3557 9858

E-mail: nxbttt@mic.gov.vn; Website: [www.nxbthongtintruyenthong.vn](http://www.nxbthongtintruyenthong.vn)

**Chi nhánh TP. HCM:** 8A đường D2, phường 25, Q. Bình Thạnh, TP. Hồ Chí Minh

Điện thoại: 08-3512 7750, 3512 7751; Fax: 08-3512 7751

E-mail: cnsg.nxbttt@mic.gov.vn

**Chi nhánh Đà Nẵng:** 42 Trần Quốc Toản, quận Hải Châu, TP. Đà Nẵng

Điện thoại: 0511-3897 467; Fax: 0511-3843 359

E-mail: cndn.nxbttt@mic.gov.vn

---

Số đăng ký kế hoạch xuất bản: 1064-2012/CXB/2-394/TTTT

Số quyết định xuất bản: 403/QĐ-NXB TTTT ngày 20/12/2012

In 1.200 bản, khổ 16x24 cm, tại Công ty Cổ phần In Việt Nam

In xong và nộp lưu chiểu tháng 3/2013.