FitNesse와 STAF을 결합한 테스트 자동화 프레임워크의 구현

(Implementing an Automated Testing Framework through the Integration of FitNesse and STAF)

나 종 채 † 오 영 은 ††
(JongChae Na) (Youngeun Oh)

유 석 문 †††
(Seokmoon Ryoo)

요 약 최근 들어 소프트웨어의 복잡도와 규모의 증가와 함께 테스트의 중요성 또한 증대되고 있다. 테스트는 QA(Quality Assurance) 기간에 한정되지 않고 개발 기간에 꾸준히 적용 되어 자주 수행될수록 높은 품질 향상을 기대할 수 있다. 하지만, 지금까지 대다수의 테스트는 QA 과정에 국한 되어 반복적이고 비 능률적인 매뉴얼 테스트 방법을 위주로 진행 되고 있다. 이는 코드 변경에 대해 효과적으로 회귀 테스트(Regression Test)를 수행할 수 없음을 의미하며 결과적으로 테스트를 자주 수행할수록 비용이 급격하게 증가하는 문제점을 가지고 있다.

본 논문에서는 이러한 문제점을 극복하기 위한 테스트 자동화 프레임워크(Framework)를 제안하고자 한다. 제안된 테스트 프레임워크는 테스트 테이블을 바탕으로 테스트를 가시적으로 설계할 수 있는 기능을 제공하며 스크립트 및 코드 작성을 최소화하여 적은 비용으로 빠른 시간 안에 테스트 자동화를 구현할 수 있도록 도와준다.

· 이 논문은 제36회 추계학술발표회에서 '효과적 테스트 자동화 프레임워크의 구현'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : NHN 생산성혁신팀 과장
monster@nhn.com
†† 비 회 원 : NHN 웹플랫폼개발팀 과장
iceize@nhn.com
††† 비 회 원 : NHN 생산성혁신팀 부장
seokmoon.ryoo@nhn.com

논문접수 : 2009년 12월 23일
심사완료 : 2010년 2월 18일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제5호(2010.5)

키워드 : 테스트 자동화, 사용자 인수 테스트, FitNesse, STAF

Abstract As developers and testers today we all suffer from increasing project complexity, the risks of late defect discovery, repetitive manual processes, and the risk of release delays. In this paper, we introduce an effective framework for automated testing to help solve such problems. Those that are new to testing do not need to delve into complex automation tools or test scripts. This framework helps automate the distribution, execution and results analysis of test cases. It also aids communication among the various stakeholders, using tables for representing tests and for reporting the results of automatically checking those tests. This paper describes the practices and benefits of using the proposed framework.

Key words : Test Automation, Acceptance Test, FitNesse, STAF

1. 서 론

소프트웨어 개발에 있어 품질보증은 언제나 어려운 과제이다. 소프트웨어 출시 후 발견되는 버그(Bug)는 고객에게 막대한 손실을 초래할 수 있으며 해당 버그에 대한 수정 비용 또한 매우 높은 특징을 갖는다. 그러므로 관련 문제를 전체 개발주기에서 보다 빨리 발견하여 적은 비용으로 수정하는 것이 개발 생산성과 품질을 높이고 비즈니스 기회를 극대화하는 첩경이다. 개발 산출물에 대한 품질 보증의 활동으로 가장 널리 사용되고 있는 방법이다. 효과적인 테스트를 위하여 다양한 도구 및 방법론이 소개되었지만 실 개발 환경에서는 여전히 테스트의 50% 이상이 비능률적인 수동 테스트에 의존하고 있는 실정이다[1]. 이는 테스트 자동화 초기에 투입되는 비용이 과도하거나 해당 기술을 익히고 적용하기까지 많은 시간을 필요로 하는 문제점과 유지보수에 많은 비용이 들어가게 되는 문제가 주요한 원인이다.

테스트 자동화를 적용하는 좋은 방법 중 하나는 범용적이고 이미 검증된 테스트 방법론과 자동화 프레임워크를 사용하는 것이다. 테스트 자동화 프레임워크를 사용하면 자동화를 위한 부수적인 개발이 필요 없으며 작성된 테스트케이스는 SUT(System Under Test)와의 의존성(Dependency)이 낮아져 변경에 의한 영향을 최소화할 수 있다. 하지만, 프레임워크를 사용하기 위해서는 테스트 도구 및 스크립트 작성을 위한 기술 습득이 필요하고 별도의 커스터마이징(Customizing) 작업 등의 자동화를 위한 추가적인 시간과 노력이 필요하게 되어 결과적으로 테스트 자동화를 적용하지 못하게 하는 주요한 걸림돌로 작용한다.

본 논문에서는 이와 같은 테스트 자동화 적용의 어려움을 극복할 수 있는 테스트 자동화 프레임워크를 소개한다.

2. 배 경

본 프레임워크는 Fit/FitNesse[2-4]와 STAF[5]이라는 두 오픈 소스(Open source)를 기반으로 개발되었다. Fit(Framework for Integrated Test)[2,3]은 Ward Cunningham에 의해 고안된 테스트 테이블 기반의 자동화 테스트 프레임워크이며, FitNesse[3,4]는 Fit의 개념을 바탕으로 위키(Wiki) 페이지 내에 테스트 테이블을 구성하고 웹 브라우저에서 테스트를 수행할 수 있도록 지원한다. FitNesse를 통해 고객과 테스터 그리고 개발자는 좀더 명확하게 대상 개발 산출물을 이해할 수 있고 다양한 기대 값에 대한 비교를 수행할 수 있다. 결과적으로, 작성된 테스트케이스에 대한 고객들의 요구사항이 얼마나 잘 반영 되었고 정확하게 수행되는지에 대한 빠른 검증이 가능하게 되어 사용자 테스트에 있어 가장 효율적으로 사용할 수 있게 된다.

STAF(Software Testing Automation Framework)[5]은 재사용이 가능한 컴포넌트들로 이루어진 다중 플랫폼(Multi-platform), 다중 언어(Multi-language)를 지원하는 오픈 소스 프레임워크로서 분산 환경에서의 테스트 환경 구축 및 실행, 리포팅(Reporting)이 용이한 장점을 가지고 있다.

본 논문에서 소개하는 프레임워크는 앞서 설명한 FitNesse의 테스트 테이블을 통한 테스트 설계 유연성(Flexibility)과 STAF 서비스의 재사용성(Reusability) 및 확장성(Extensibility)을 결합하여 개발 되었다. 이에 따라 정형화된 다른 도구와 달리 범용적인 테스트케이스 구축에 많은 이점을 가지게 되며 '테스트 환경 구축, 테스트케이스 설계, 수행, 결과 보고, 테스트 환경의 초기화'라는 일련의 통합 테스트(Integration Test) 과정에 폭 넓게 적용할 수 있게 된다.

3. 프레임워크의 구조

이 논문을 통해 설명되는 프레임워크의 정식 명칭은 NTAF(NHN Test Automation Framework)이며 내부 구성도는 그림 1과 같다. 그림 1에서 회색으로 표현된 NTAF Fixture가 실제 NTAF의 구현부이며, FitNesse 및 STAF의 라이브러리를 바탕으로 개발되어 있다.

3.1 Flow Control

NTAF은 FitNesse를 통해 구현된 위키 테스트케이스 테이블에 대한 흐름제어(Flow control)를 가능하게 하는 다양한 키워드를 제공하고 있으며 해당 키워드를 이용해 사용자는 동적인 테스트 설계 작업을 할 수 있게 된다. 표 1은 NTAF이 제공하고 있는 키워드에 대한 설명이다.

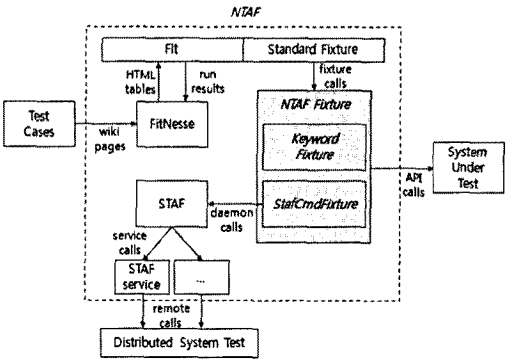

그림 1 NTAF Architecture

표 1 NTAF Keyword Fixture

키워드 명	액션
LOOP	테스트 테이블을 조건만큼 반복 수행
ITERATE	리스트 데이터만큼 테이블 반복 수행
BREAK	LOOP, ITERATE 키워드를 조건에 따라 종료
CONTINUE	LOOP, ITERATE 키워드의 시작 부분으로 이동
PARALLEL	테스트 테이블을 병렬 수행
PARALLELITERATE	리스트 데이터만큼 테이블 병렬 수행
IF/ELSEIF/ELSE	조건에 따라 테이블 선택 수행
LOG	테스트 수행과정의 로그 정보 수집
TIMER	정해진 시간만큼 테이블 지속 수행
VAR	심볼 변수에 데이터 할당 및 연산
ASSERTION	심볼 변수에 대한 비교 수행

NTAF 키워드는 혼용(Nested) 사용 및 심볼변수를 통한 테이블 간의 데이터 공유가 가능하다. 이러한 키워드를 활용함으로써 테스트케이스에 대한 코딩, 스크립트 작업을 최소화할 수 있도록 시나리오 테스트를 구현할 수 있다.

3.2 STAF과의 연동

그림 1에 기술된 StafCmdFixture는 STAF의 내부 및 외부 서비스를 NTAF의 위키페이지에서 사용할 수 있도록 지원한다. StafCmdFixture를 이용하여 원격 장비에 대한 제어가 가능하며 분산환경 테스트 환경 구축, 테스트 수행, 결과 수집, 초기화라는 일련의 과정을 위키페이지 내에 모두 기술하여 테스트의 시작과 끝을 아우르는 종단 테스트 자동화(End-To-End Test Environments)를 구현할 수 있다. 그림 2는 StafCmdFixture를 사용하여 원격 장비에 대해 자바 컴파일 수행 명령을 내린 후, 생성된 바이너리의 수행 결과 값을 비교하는 예제이다.

그림 2의 녹색(음영부분) 영역은 테스트 수행 결과와 예상 값이 일치하여 테스트가 성공하였을 경우에 나타난다.

StafCmdFixture				
service	location	request	submitMarshall()	response()
PROCESS	172.16.189.129	START SHELL COMMAND javac HelloWorld.java WORKDIR C:\ RETURNSTDOUT STDERRTOSTDOUT WAIT	0	blank
PROCESS	172.16.189.129	START SHELL COMMAND java -cp . HelloWorld WORKDIR C:\ RETURNSTDOUT STDERRTOSTDOUT WAIT	0	Hello World!

그림 2 NTAF을 이용한 분산환경 테스트 성공 예

PROCESS	172.16.189.129	START SHELL COMMAND java -cp . HelloWorld WORKDIR C:\ RETURNSTDOUT STDERRTOSTDOUT WAIT	0	Hello World? *expected* Hello World! *actual*

그림 3 NTAF을 이용한 분산환경 테스트 실패 예

예상 값과 실제 값이 일치하지 않을 경우에는 그림 3과 같이 붉은색 마킹(진한 음영부분)과 함께 예상 값과 실제 값을 모두 표시해 준다.

4. 한글로 서술된 테스트케이스

FitNesse의 가장 큰 장점은 설계 된 테스트케이스의 높은 가시성을 통한 이해자 간의 협업의 극대화에 있다. 하지만, FitNesse는 영어를 기반으로 개발 되어 한글을 사용한 테스트케이스 설계에 많은 제약을 가지고 있다. NTAF은 FitNesse의 이러한 단점을 보완하여 한글을 이용한 테스트케이스의 설계가 가능하도록 지원하고 있다. 그림 4는 한글을 이용한 간략한 테스트케이스의 예제이다.

eg.fixture.거래내역								
check	가진돈	철수	10000					
check	가진돈	영희	5000					
누가	철수	누구에게	영희	빌려준돈	2000	매달이자율	0.5	
check	가진돈	철수	8000					
check	가진돈	영희	7000					
check	갚을돈	2000						
누가	영희	누구에게	철수	갚은돈	1500	몇달뒤	2	
check	가진돈	철수	9520					
check	가진돈	영희	5480					
check	갚을돈	500						
누가	영희	누구에게	철수	갚은돈	500	몇달뒤	1	
check	가진돈	철수	10022.5					
check	가진돈	영희	4977.5					
check	갚을돈	0						

그림 4 한글로 작성된 NTAF 테스트케이스

이러한 장점을 활용하여 테스트케이스를 테스트 대상을 위한 가이드 문서로 활용할 수 있으며 테스트케이스에 대한 여러 이해자 간의 공유가 용이하게 된다.

5. 다양한 도구들과의 연계

개발 단계에서 테스트를 작성하고 수행하기 위해서는 통합 개발 환경에 대한 지원은 필수적 이다. 이를 위하여 NTAF은 Eclipse, Maven, Hudson에 대한 Plug-in을 제공하고 있다. NTAF Plug-in들을 활용하여 개발자/QA/테스터들은 개발 프로세스 전반에 걸쳐 손쉽게 테스트를 작성하고 실행할 수 있으며 이에 대한 피드백을 받을 수 있다. 이와 더불어 검증이 어려운 동시성(Concurrency) 문제를 검증할 수 있는 도구도 함께 제공 한다.

5.1 Maven plug-in

NTAF은 최근에 각광을 받고 있는 자바 빌드(Build) 자동화 도구 중 하나인 Maven plug-in을 제공하고 있다. NTAF Maven plug-in은 NTAF 서버의 실행 및 테스트 수행, 결과 보고 리포트 작업을 담당한다. NTAF Maven plug-in은 Java.net[6]의 Maven 저장소(Repository)에 배포되어 있으며 Maven 기반의 프로젝트라면 의존성(Dependency) 추가만으로 사용이 가능해진다.

5.2 Hudson plug-in

NTAF을 통해 작성된 테스트케이스는 회귀 테스트를 통해 개발 산출물에 대한 주기적인 검증 작업을 수행한다. NTAF은 CI(Continuous Integration) 서버 중 하나인 Hudson[7]에 대한 plug-in을 제공하고 있다. 이를 통해 테스트의 주기적인 수행은 물론, 결과 저장 및 빌드 별 테스트 현황 지표 확인이 가능하다. 그림 5는 NTAF Hudson plug-in의 사용 예이다.

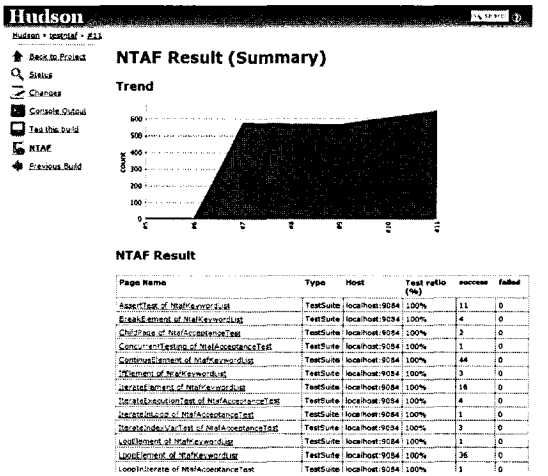

그림 5 NTAF Hudson plug-in

5.3 Eclipse plug-in

NTAF은 대부분의 자바 개발 프로젝트에서 가장 널리 사용되고 있는 개발환경인 Eclipse[8]에 대한 plug-

in을 별도로 제공하고 있다. 개발자는 자신의 로컬 컴퓨터에서 개발 중인 개발 코드에 대해 NTAF Eclipse plug-in을 사용하여 테스트케이스를 손쉽게 설계하고 테스트할 수 있게 된다. 그림 6은 Eclipse 상에서 동작 중인 NTAF Eclipse plug-in이다.

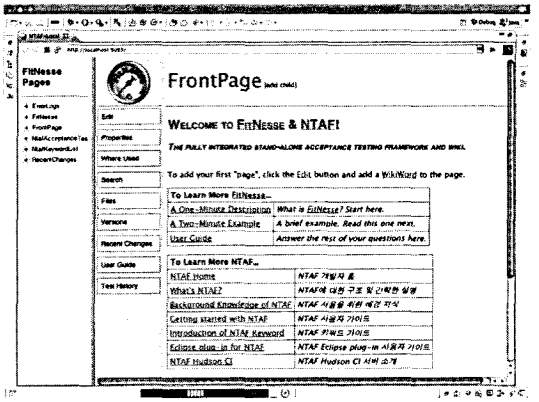

그림 6 NTAF Eclipse plug-in

5.4 동시성 테스트 지원

쓰레드 간의 공유 자원에 의한 동시성 문제는 개발자 및 테스터에게 불규칙적인 발생 메커니즘으로 많은 고민을 안겨주고 있다. 동시성 문제는 발견도 힘들뿐만 아니라 재현도 어려워 개발 및 QA 단계에서 발견 되지 못하고 사용자에 의하여 발견 되는 경우가 대부분이다. NTAF은 동시성 오류 재현을 돕는 도구인 ConTest [9]를 연동하여 동시성 오류를 쉽게 재현할 수 있도록 하였으며, 부가적으로 쓰레드 별 테스트 커버리지(Test Coverage) 측정, 디버깅(Debugging) 기능을 제공하고 있어 사용자는 별도의 설정 작업 없이 자신이 작성한 테스트케이스에 대한 동시성 테스트를 수행할 수 있다.

6. NTAF을 활용한 테스트 구현 예제

NTAF을 활용한 테스트 구현 예제로 3.2장의 예제를 확장하여 그림 7과 같이 테스트 전략을 작성하였다. (본 사례는 NTAF을 통한 분산환경하에서의 효율적이고 빠른 테스트 환경 구성 및 테스트케이스의 작성/수행을 설명하기 위한 예로서 NTAF 구성의 실질적인 사례가 아님을 미리 밝혀 둔다.)

본 사례는 Windows, Linux, Mac으로 구성된 Multi-Platform 분산환경에서 테스트를 수행하며, ①~③번 과정을 통해 테스트 환경을 구축하고, ④, ⑤번 작업으로 테스트를 수행한다. 마지막으로 ⑥번 과정을 통해 각 테스트 장비에 설치된 테스트 관련 데이터를 초기화하게 된다.

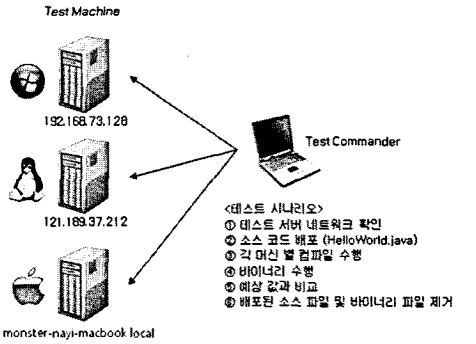

그림 7 분산환경 내 다중 OS에 대한 테스트 시나리오

테스트는 ①~⑥번까지가 순차적으로 수행되며 테스트 환경은 매 테스트 마다 초기화 되어 이전 테스트 결과의 간섭으로 인한 테스트 실패를 사전에 차단할 수 있다. 그림 8은 NTAF의 ITERATE 키워드를 이용하여 테스트 장비의 수만큼 반복을 수행하는 예이다.

ITERATE 키워드의 내부에는 StafCmdFixture를 사용하여 분산환경 테스트 장비의 상태를 확인할 수 있도록 PING 테스트를 수행한다. PING 테스트 수행 결과는 IF 키워드를 통해 평가되어 다음 테스트케이스의 수행 여부를 결정한다. 테스트를 수행할 각 장비 별 배포 위치가 OS 별로 다르기 때문에, VAR 키워드를 통해 장비 별 적절한 소스코드의 배포 위치를 설정한다.

다음으로 그림 9에서는 FS(File System) 서비스를 사용하여 원격 배포를 수행한 후, PROCESS 서비스를 통해 컴파일 및 테스트를 수행한다. 테스트 종료 후에는 다시 FS 서비스를 이용하여 배포된 파일을 삭제하게 된다.

테스트 수행 후 결과는 앞의 NTAF 예와 같이 작성

start_iterate	{var:$machine$}, {in:192.168.73.128, monster-nayi-macbook.local, 121.189.37.212}, {indexvar:i}

그림 8 ITERATE 키워드를 통한 장비 별 반복 수행 예

StafCmdFixture

service	location	request	submitMarshall()	response()
FS	local	COPY FILE $local_locate$/HelloWorld.java TODIRECTORY $locate$ TOMACHINE $machine$	0	
PROCESS	$machine$	START SHELL COMMAND javac HelloWorld.java WORKDIR $locate$ RETURNSTDOUT STDERRTOSTDOUT WAIT	0	
PROCESS	$machine$	START SHELL COMMAND java -cp . HelloWorld WORKDIR $locate$ RETURNSTDOUT STDERRTOSTDOUT WAIT	0	Hello World!

StafCmdFixture

service	location	request	submit()	response()
FS	$machine$	DELETE ENTRY $locate$HelloWorld.java CONFIRM	0	
FS	$machine$	DELETE ENTRY $locate$HelloWorld.class CONFIRM	0	

그림 9 배포 및 테스트 수행/초기화 작업 예

된 테스트케이스 내에서 바로 확인할 수 있다.

7. 결 론

개발된 기능의 동작 여부를 판단할 수 있는 명확하게 작성된 테스트는 품질의 척도이자 안전장치이다. 이를 바탕으로 개발자는 기 개발된 기능에 오류를 발생 시키지 않고 수정(Refactoring)을 할 수 있으며 QA 및 테스터는 제품의 안정성을 보장할 수 있다. 작성된 테스트케이스가 이해하기 어렵게 되어 있다면 테스트케이스는 정보 전달 수단이 아닌 또 하나의 복잡한 개발 스펙(Specification)이 되어 의사 소통 비용을 증가 시키게 된다. 이런 문제점을 해결하기 위해서 테스트케이스는 누구나 쉽게 이해할 수 있도록 작성되어야 하며 개발 관련자 모두가 편리하게 공유하고 실행할 수 있어야 한다. NTAF은 개발자, QA, 테스터, 고객 모두가 쉽게 적용 가능한 프레임워크이며 테스트케이스를 훌륭한 참조 문서(Reference Document)로 만들어 준다. 이를 통해 개발자는 코드에 대한 검증을, QA는 개발자가 작성한 테스트케이스에 대한 부정적(Negative) 테스트케이스의 보강을, 테스터는 반복적으로 수행되어 왔던 테스트에 대한 자동화된 환경을 구축할 수 있으며, 고객(기획자 혹은 사용자)은 이해하기 쉬운 테스트케이스를 통해 자신이 요구했던 기능에 대한 손쉬운 검증이 가능하게 된다. 또한, NTAF을 통해 그간 단순하고 반복적으로 수행해 왔던 비 자동화 테스트 요소에 대해 자동화가 가능하게 되고 이는 회귀 테스트 시에 좀 더 가시적인 성과가 된다. 자동화 되어 작성된 테스트케이스로 인해 개발자는 약간의 기능 변경 시에도 발생할 수 있었던 side-effect로 인해 주저했던 부분들로부터 해방될 수 있고 이는 결과적으로 개발 기간의 단축 및 릴리즈(Release)된 제품의 안정성을 보장하게 되어 서두에서 언급한 비용 감소의 효과를 가져오게 된다.

NTAF은 2009년 2월 NAVER 개발자 센터[10]에 GPL v2 오픈 소스로 공개되었으며, 2009년부터 NHN 내부적으로 수행해 오고 있는 Quality Practice 활동의 일환으로 테스트 분야에 활발히 사용되고 있다. 또한 최근에는 NTAF과 Selenium[11]의 연동을 통한 웹 UI 테스트 자동화 방안을 연구 중에 있다.

참 고 문 헌

[1] Eunha Kim, Jong Chae Na, and Seokmoon Ryoo, "Implementing an Effective Test Automation Framework," *IEEE DOI 10. 1109/COMPSAC*, pp.534-538, 2009.

[2] Ward Cunningham, "Fit: Framework for Integrated Test," Available: http://fit.c2.com/

[3] Rick Mugridge, and Ward Cunningham, "Fit for Developing Software," Prentice Hall, 2005.

[4] FITNESSE, "FrontPage," Available: http://www.fitnesse.org/

[5] SOURCEFORGE.NET, "Software Testing Automation Framework (STAF)," Available: http://staf.sourceforge.net/

[6] Java.net, "The Source for java Technology Collaboration," Available: http://java.net/

[7] Hudson: an extensible continuous integration engine, Available: https://hudson.dev.java.net/

[8] Eclipse.org home, Available: http://www.ecli-pse.org/

[9] ConTest - A Tool for Testing Multi-threaded Java Applications, Available: http://www.haifa.ibm.com-/projects/verification/contest/index.html/

[10] NAVER Developer Center, Available: http://dev.naver.com/

[11] Selenium - Browser automation framework, Available: http:// code.google.com/p/selenium/