패턴 중심의 웹 테스트 자동화 프레임워크의 구현

(Implementation of Pattern-Driven Web Test Automation Framework)

나 종 채 † 정 희 수 ††
(JongChae Na) (Hyiesoo Jeong)

유 석 문 †††
(Seokmoon Ryoo)

요 약 빠르게 진화하는 웹 생태계에서 테스트는 안정성과 생산성향상을 위한 필수과정이다. 복잡하게 얽혀 있는 웹 콘텐트(content)와 유저 인터페이스(user interface)에 대한 테스트는 매우 중요하며, 작성된 테스트케이스는 자동화 되어 누구나 쉽게 반복적으로 수행될 때 가장 효과적이다. 하지만 현재까지 나와 있는 대부분의 도구들은 웹 요소에 대한 기술적 접근 가능성에 중점을 두고 있으며, 이해관계자들 간의 협업(collaboration), 작성된 테스트케이스의 재사용(reusability)에 대한 부분을 간과하고 있다.

본 논문에서는 웹 테스트 자동화에 있어 효율적인 테스트케이스 설계와 이를 공유하고 패턴화시켜 재 사용할 수 있는 테스트 프레임워크(framework)를 제안한다. 오픈 소스를 기반으로 제작된 본 프레임워크는 웹 테스트 자동화와 자동화된 테스트케이스를 지속적으로 수행할 수 있는 통합(integration) 환경을 제공한다.

키워드 : 테스트, UI, 패턴, 테스트 자동화

Abstract The web environment is evolving rapidly. Testing in the web based software is an essential process to improve stability and productivity. Testing of complex web contents and UI(user interface) is most important thing. Implemented test cases are efficient when they are automated and reusable. But, most of the testing automation tools are focused on technical accessibility and functions still. A collaboration of the persons concerned and reusability of implemented test case are ignored.

In this paper we propose an efficient way to design automated test case in web environment, and to share and pattern automated test cases we introduce testing framework called NTAF(NHN Test Automation Framework.) The NTAF is based on open source framework. It provides integrated testing environment that web testing cane be automated and managed continuously.

Key words : Test, UI, Pattern, Test Automation

1. 서 론

Mike Cohn은 테스팅 피라미드(Testing pyramid)[1]에서 UI 테스트는 단위 테스트(Unit Test), 사용자 인수 테스트(Acceptance Test)와 비교하여 대상 콘텐트의 다양성, 모호성, 복잡성으로 구현 및 유지 보수가 어려움을 설명하였다. 그럼에도 UI 테스트는 웹 서비스 기술의 발전과 더불어 중요성이 점차 높아지고 있다.

본 논문에서 소개하는 테스트 자동화 프레임워크는 웹 테스트 패턴(web test pattern)이라는 새로운 개념을 제안한다. 이를 통해 사용자들은 변화가 빠른 웹 환경에서 높은 생산성과 유지보수성을 갖는 통합 테스트 환경을 구축할 수 있게 된다. 제안된 프레임워크의 정식명칭은 엔탭 웹킷(NTAF WebKit)이다.

2. 배 경

NTAF WebKit은 오픈 테스트 자동화 도구인 FitNesse[2], NTAF(NHN Test Automation Framework)[3,4], WebDriver, Selenium[5]을 이용하여 구현되었다. FitNesse는 FIT(Framework for Integrated Test)[6,7]의 개념을 바탕으로 위키(Wiki) 페이지에 테스트 테이블을 작성하고 웹 브라우저에서 테스트를 수행 하는 테스트 자동화 프레임워크 이다. NTAF은 획일적으로 수행되던 FitNesse의 테스트 흐름(workflow)을 NTAF 키워드를 통해 제어하며 IBM의 STAF(Software Testing Automation Framework)[8]과 결합하여 분산 환경에서의 테스트 자동화(End-to-End Test Automation)가 가능 하도록 지원한다. 마지막으로 WebDriver는 웹 UI 테스트 자동화를 지원하는 도구로서, 최근에는 Selenium와 연동되어 서로간의 장, 단점을 보완하고 있다. WebDriver는 작성된 테스트케이스에 대한 크로스 브라우징(cross browsing) 테스트, 가상 브라우저의 빠른

· 이 논문은 2010 한국컴퓨터종합학술대회에서 '패턴 중심의 웹 테스트 자동화 프레임워크의 구현'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : (주)NHN 생산성혁신팀 과장
monster@nhn.com
†† 비 회 원 : (주)NHN 생산성혁신팀 과장
shiningsoo@nhn.com
††† 비 회 원 : (주)NHN 지도지역서비스개발랩 랩장
seokmoon.ryoo@nhn.com

논문접수 : 2010년 8월 11일
심사완료 : 2010년 10월 27일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제12호(2010.12)

테스트 수행 등의 강력한 기능을 지원한다. NTAF WebKit은 이런 여러 오픈 소스 테스트 도구들을 융합하여 FitNesse의 협업성, NTAF의 분산환경 테스트, WebDriver를 통한 웹 UI 테스트 기능을 제공한다.

3. Architecture

NTAF WebKit은 FitNesse, NTAF, WebDriver이 결합된 구조로 이루어져 있다. 그림 1은 NTAF WebKit의 테스트 수행 순서에 기반한 구조이다.

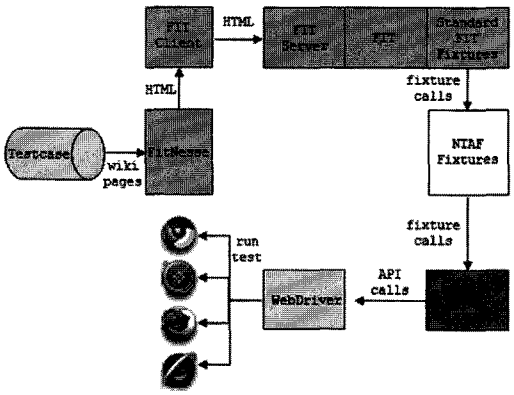

그림 1 NTAF WebKit Architecture

FitNesse의 위키 문법으로 작성된 테스트케이스는 HTML 형태로 변환되어 FIT Server에 전달된다. 전달된 테스트케이스는 대응되는 NTAF WebKit Fixture의 키워드로 호출되며 키워드 내에서 WebDriver의 API를 다시 호출하면 테스트케이스 내에 정의된 브라우저 별 테스트가 수행된다. 수행된 테스트 결과는 위키 페이지로 전달되어 테스트케이스 내에서 확인이 가능하다.

4. 위키를 통한 효율적인 테스트케이스의 작성

NTAF WebKit은 위키 페이지 상에서 빠르고 정확한 테스트케이스의 작성을 돕는 Smart Wiki Editor와 Wiki Test Box 기능을 제공한다.

4.1 Smart Wiki Editor

WebDriver를 이용한 코드 위주의 테스트케이스 작성에 비해 위키 기반의 작성방식은 통합 개발환경(IDE)이 지원 되지 않아 작성이 어렵고 생산성 또한 저하되는 단점이 있다. 자바 개발환경에서 가장 많이 사용되는 이클립스(eclipse)에서 테스트케이스를 작성할 경우 콘텐트 어시스트(Content Assist)기능을 이용하여 WebDriver의 API 리스트, 입력 인자(parameter) 등을 바로 확인할 수 있으며 잘못 입력된 키워드, 데이터에 대한 실시간 확인이 가능하다. 반면에 위키에서는 이런 지원 기능의 부재로 높은 생산성을 기대하기 어렵다. NTAF WebKit은 이런 단점을 보완하기 위한 두 가지 기능을 지원한다.

첫 번째 기능은 테스트케이스 작성의 생산성을 높여주는 키워드 추천 기능이다. 기존의 위키 페이지 입력 방식은 픽스처 내에 구현되어 있는 키워드를 개발코드 또는 문서에서 직접 확인해야 하는 번거로움이 있었다. NTAF WebKit은 키워드 추가 시 사용 가능한 키워드 목록이 위키에 실시간으로 표출되어 편리한 테스트 작성을 지원한다(그림 2).

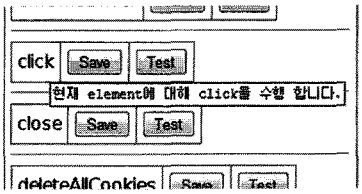

그림 2 적용가능 키워드 목록 표출

그림 3의 'get' 키워드는 테스트하고자 하는 대상 웹 주소(url)를 입력 받아 테스트 브라우저를 실행하는 키워드이며 사용자가 'get' 키워드 추가 후 테스트를 수행하면 주소 입력 창이 자동으로 추가 되는 예이다.

각 키워드 및 인자 값에 대해서는 툴 팁(tool tip)을 제공하고 있어 사용자는 별도의 문서 없이 테스트케이스를 작성할 수 있다.

그림 3 주소 값을 받는 키워드 호출

NTAF WebKit의 두 번째 기능은 동적 웹 페이지 콘텐트의 실시간 조회 기능이다. 웹 UI 테스트를 하기 위해서는 해당 웹의 요소(element), 요소의 속성(attribute), 속성 값(value)에 대한 데이터를 필요로 하게 된다. 이런 웹 데이터를 확인하기 위해서는 별도의 도구를 활용 하거나, 웹 페이지 소스를 일일이 확인해야 하는 번거로움이 있다. 하지만 생산성의 측면에서 볼 때 이렇게 작성하는 방식은 분리된 도구에서 오는 생산성 저하와 작성의 불편함을 수반하게 된다. NTAF WebKit은 테스트케이스 작성 시에 웹 콘텐트를 바로 확인이 가능하도록 지원한다. 그림 4는 WebDriver의 'findElement'키워드를 사용하여 테스트 페이지의 웹 구성요소를 동적으로 출력하는 화면이다.

앞의 일반 속성 이외에도 WebDriver가 지원하는 xpath기능을 편리하게 사용할 수 있도록 그림 5와 같이 자동으로 조합된 xpath 리스트를 구할 수도 있다.

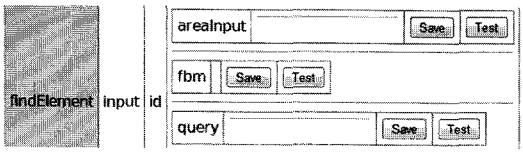

그림 4 input 요소의 id 값을 가지는 리스트 표출

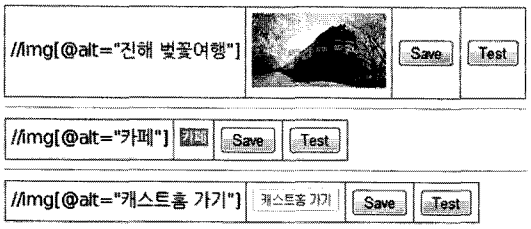

그림 5 img 요소의 xpath 값을 가지는 리스트 표출

동적 웹 구성요소를 직접 테스트케이스에서 확인하면서 테스트를 단계적으로 진행하는 방식은 웹 요소, 속성, 속성 값의 단계적 설정 과정을 통해 정확한 데이터의 선택이 가능해지고 작성된 테스트케이스의 100% 성공을 보장하는 장점을 갖는다. 이는 TDD(Test Driven Development) 방식으로 개발되는 단위테스트가 모두 성공할 수 밖에 없는 이유와 같다.

4.2 Wiki Test Box

Wiki Test Box(그림 6)을 활용하여 사용자는 작성된 테스트케이스에 대해 손쉽게 테스트 브라우저를 변경하여 손쉬운 크로스 브라우징 테스트를 수행할 수 있다. Wiki Test Box는 그 외에도 자주 쓰는 키워드를 등록하여 사용할 수 있는 키워드 즐겨찾기, WebDriver와 Selenium의 키워드를 손쉽게 추가할 수 있는 확장 기능을 지원한다.

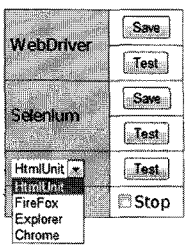

그림 6 Wiki Test Box를 이용한 브라우저 변경 테스트

5. 테스트케이스의 패턴화

5.1 웹 테스트 패턴

지금까지 제시되었던 많은 웹 UI 테스트 자동화 도구들은 프로그래밍 레벨에 근접하는 스크립트 또는 레코딩 방식으로 테스트를 수행한다. 스크립팅을 통한 테스트는 대상 플랫폼에 상관없이 실제 사용자의 작업과 동일한 이벤트를 일으킬 수 있어 테스트의 제약이 적다. 하지만 각각의 프레임워크에 특화된 스크립트를 익히는 것이 사용자에게 진입장벽으로 작용하게 되고 작성된 스크립트가 테스트의 의도를 명확하게 표현하기 어려운 단점을 갖는다.

레코딩 방식으로 테스트를 작성하는 경우 자체 지원 도구를 이용하여 다양한 웹 요소에 접근, 이벤트를 저장한다. 저장된 레코딩 데이터는 다음부터 테스트 자동화에 재사용 된다. 레코딩 방식은 사용법도 간단하고 특별히 도구에 대한 학습이 필요하지 않는 장점이 있다. 반면 레코딩 방식이 클라이언트(client) 이벤트 중심으로 이루어지기 때문에 서버(server) 상의 데이터를 가져오는 데는 제약이 존재한다.

앞서 설명한 두 방식의 웹 테스트 장점들을 수용하여 본 논문에서는 웹 테스트 패턴을 이용한 점진적 테스트 자동화 방식을 제안한다. 웹 테스트 패턴이란 다양한 웹의 콘텐트 속에서 공통되는 요소를 끄집어 내어 이를 일반화 한 것이다. 일례로 검색 사이트의 검색 과정을 보면, 우선 검색 창을 찾고 검색어를 입력한다. 다음으로 검색 버튼을 누르게 되고 검색된 결과가 예상한 결과와 일치하는지 확인(assertion)하는 과정을 거친다. 검색과 관련된 수순은 웹 사이트 마다 거의 동일하며, 단지 차이점은 해당 웹 요소의 식별자 또는 이벤트가 될 것이다. 물론 웹 테스트 패턴과 관련된 내용을 모든 웹 테스트에 완벽히 적용할 수는 없다. 이는 다양하고 복잡한 사용자의 요구를 수용하는 웹의 특성에 기인한다고 할 수 있다. 하지만, 웹의 기능 상당수가 공통된 패턴을 가지고 있고 이를 기반으로 동작하고 있다. NTAF WebKit은 이점에 착안하여 웹의 공통된 패턴을 끄집어내어 일반화 시킨 후 재사용할 수 있도록 지원한다.

5.2 웹 테스트 패턴 작성 및 적용 예

NTAF WebKit에서 지원하는 웹 테스트 패턴 기능을 이용하여 앞장에서 소개한 검색과 관련된 웹 테스트 패턴을 실제로 적용해 본다. 여기서는 국내 검색 포털 중 하나인 네이버(http://www.naver.com)의 메인 검색 창에서 '김연아' 검색어를 입력 후 결과를 확인하는 과정을 패턴화하도록 한다. 먼저 'htmlUnitDriver' 키워드를 추가한다. HtmlUnitDriver는 오픈소스인 HtmlUnit을 이용한 빠른 테스트 피드백(feedback)을 지원한다(그림 7).

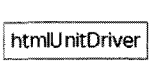

그림 7 HtmlUnitDriver 선택

다음으로는 테스트를 수행할 웹 페이지에 접근한다. WebDriver가 지원하는 'get' 키워드를 이용하여 그림 8

과 같이 인자 값으로 테스트 대상 주소를 입력한다.

그림 8 get 키워드를 사용한 테스트 페이지 로드

이제는 검색 창을 찾는다 Smart Wiki Editor를 이용하여 'findElement' 키워드를 선택한 후 id 값이 'query'인 검색 창(input)을 찾는다(그림 9).

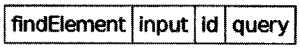

그림 9 id='query'인 input 요소 선택

검색 창을 성공적으로 찾았으면, 검색 창에 검색어 입력을 위해 'setText'키워드의 인자 값으로 '김연아'를 입력한다(그림 10).

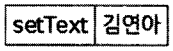

그림 10 검색어 입력

검색 버튼을 찾기 위해 다시 'findElement' 키워드를 이용하여 검색 버튼의 속성 값을 선택한다. 여기서는 조금 다른 방식으로 자동으로 완성되어 표출되는 xpath를 이용하여 검색버튼을 찾는다(그림 11).

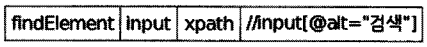

그림 11 검색버튼 선택

검색 버튼을 찾았으면, 검색 버튼을 클릭하기 위해 'click' 키워드를 삽입한다(그림 12).

그림 12 검색 버튼 클릭

마지막으로 검색결과 페이지에서 기대 값과의 일치 여부를 검증 한다. 이러한 비교 방법에는 다양한 방법이 존재하지만, 여기서는 'findElement'를 이용하여 그림 13과 같이 기대하는 제목의 웹 링크(link)가 존재여부를 확인한다.

그림 13 검색 결과의 확인

작성된 테스트케이스를 수행하면 그림 14와 같이 성공적으로 검색을 검증하는 테스트를 수행한다.

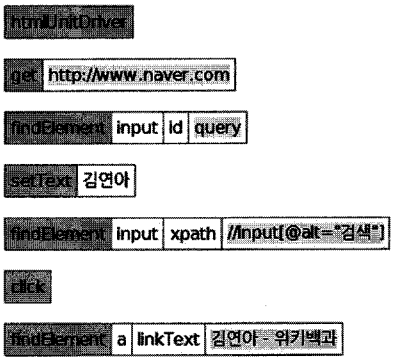

그림 14 테스트 수행 결과 확인

'htmlUnitDriver'로 작성된 테스트 케이스는 Wiki Test Box를 이용하여 별도의 테스트케이스 수정작업 없이 다양한 실제 브라우저 테스트가 가능하다. 다음은 작성된 테스트케이스에 대한 일반화 후 패턴화 과정을 진행한다. 앞서 작성된 테스트케이스는 해당 웹(네이버)에 테스트가 특화되어 있다. 일반화 작업을 위해 키워드의 가장 마지막 셀에 인자로 받는 요소의 속성 값을 모두 삭제한다. 이와 함께 패턴 사용시 테스트 드라이버의 자유로운 선택을 위해 'htmlUnitDriver'도 제거한다. 이런 방법으로 의존성(dependency)을 제거하면 일반화 과정이 완료된다(그림 15).

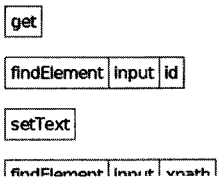

그림 15 테스트케이스의 일반화

일반화된 테스트케이스는 NTAF WebKit에서 지원하는 패턴 페이지에 별도로 저장한다. 다음으로 저장된 테스트 패턴을 새로 작성되는 테스트케이스에 적용한다. 테스트 대상 웹 사이트를 구글(http://www.google.com)로, 검색어는 '김연아'에서 '아사다 마오'로 변경한 후 패턴목록을 호출하고, 앞서 저장한 검색 패턴 항목(BasicSearch)을 로딩하여 해당 페이지에 적용한다.

패턴이 적용되었으면 FireFox를 이용한 테스트를 진행하기 위해 가장 상단에 'fireFoxDriver'를 추가 후 테스트를 수행한다. NTAF WebKit은 테스트 수행 도중 입력이 되지 않은 키워드의 필드가 있을 경우 바로 테스트를 중단하고 다음 값의 입력을 받도록 대기한다. 그림 16과 같이 'get' 키워드에서 테스트를 수행할 주소를 입력하도록 대기 상태에 머문다.

그림 16 get 키워드를 사용한 테스트 페이지 로드

http://www.google.com을 입력 후 테스트를 수행하면 다시 입력할 데이터가 남은 다음 키워드에서 멈춘다.

그림 17을 보면 네이버와 달리 구글은 검색 창의 id 값이 'q'이다. 해당 속성 값에 대한 테스트를 다시 진행 후 검색어로 이번에는 '아사다 마오'를 입력한다. 이런 과정을 반복하여 사용자는 패턴이 제안하는 흐름에 따라 대상 웹 페이지에 특화된 데이터를 선택 혹은 입력한다. 수행 결과는 그림 18과 같다.

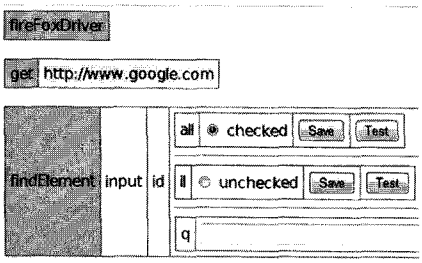

그림 17 id='q'인 input 요소 선택

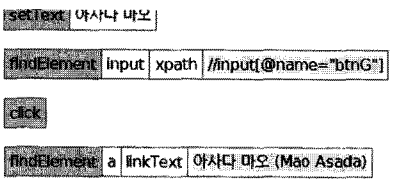

그림 18 패턴을 적용한 테스트 수행

6. 결 론

본 논문에서 기술하고 있는 '웹 테스트패턴'은 테스트 작성과 재활용을 위한 효과적인 대안이다. 개인의 지식이 조직 내에 공유되고 축적될 수 있는 협업과 발전을 위한 모델로 활용될 수 있다. 지금까지 나온 대다수의 웹 UI 테스트 도구들은 변화하는 웹 기술에 대응하는 다양한 기능, 레코딩 방법 등에 주안점을 맞추어 개발이 이루어져 왔다. 물론 각 도구들의 기술적인 접근 또한 의미가 있지만, 테스트대상에 대한 정확한 이해와 그들이 갖는 공통의 영역을 도출하는 기술도 매우 중요하다. 이런 지식들이 계속 쌓여 점차 일반화 되고 패턴화되면 결국 다양한 테스트 템플릿(template)을 제공할 수 있게 된다. 테스트 템플릿은 웹 개발자들에게 공유되어 이를 고려한 웹 개발 생태계가 조성될 수 있다. 테스트 패턴의 일반화는 궁극적으로 테스트하기 쉬운(testable) 개발 코드로 이어진다. NTAF WebKit은 테스트의 본질적 관점에 대한 접근을 시도한 프레임워크이다. 테스트를 수행하는 대상은 한 명의 개인이 만들어 내는 산출물이 아닌 여러 이해 관계자들이 관여된 공동의 창작물이다. 때문에 테스트는 모두가 공감할 수 있고 공유될 수 있는 기반이 마련된 후에 설계되어야 한다. 그리고 자동화되어 설계된 테스트케이스는 일회성이 아닌 지속적인 수행 과정을 거칠 때 진정한 가치를 가지게 된다. NTAF WebKit이 가지는 가치는 여기에 있으며 주안점을 두고 있는 웹 테스트 패턴의 커뮤니티화를 통해 기존의 테스트 도구들이 가진 한계를 극복하는 시도를 하고 있다.

NTAF WebKit은 2010년 초에 정식 릴리즈가 되어 NHN에서 웹 UI 테스트 자동화의 용도로 사용되고 있다. 네이버 개발자 센터[9]를 통해 오픈 소스로 공개된 NTAF의 확장 모듈로 등록되어있어, 누구나 쉽게 사용할 수 있으며 개발 참여도 가능하다.

참 고 문 헌

[1] Mike Cohn, "The forgotten layer of the test automation pyramid," Mike Cohn's Blog - Succeeding with Agile, http://blog.mountaingoatsoftware.com/
[2] FitNesse, "FrontPage," http://www.fitnesse.org/
[3] J. Na, Y. Oh, S. Ryoo, "Implementing an Automated Testing Framework through the Integration of FitNesse and STAF," *Journal of KIISE : Computing Practices and Letters*, vol.16, no.5, pp.581-585, May. 2010. (in Korean)
[4] NTAF, "NHN Test Automation Framework," http://dev.naver.com/projects/ntaf/
[5] Selenium, "Browser automation framework," http://code.google.com/p/selenium/
[6] Ward Cunningham, "Fit: Framework for Integrated Test," http://fit.c2.com/
[7] Rick Mugridge, and Ward Cunningham, "Fit for Developing Software," Prentice Hall, 2005.
[8] STAF, "Software Testing Automation Framework (STAF)," http://staf.sourceforge.net/
[9] NAVER 개발자 센터, http://dev.naver.com/