AJAX를 위한 자바스크립트 시험 도구 제안

서광익[O] 최은만
동국대학교 컴퓨터 공학과
{bradseo[O], emchoi}@dgu.ac.kr

JavaScript Test Tool for AJAX

Kwang Ik Seo[O] , Eun Man Choi
Dept. of Computer Engineering, Dongguk Univ., Seoul, Korea

요 약

웹 플렛폼이 계속해서 발전하고 기술이 고도화 됨에 따라 이를 사용하기 위한 다양한 지원 기술과 프레임워크가 제안되고 있다. 그 중 최근 가장 관심이 집중되고 있는 브라우저 기반의 웹 어플리케이션 기법이 AJAX(Asynchronous JavaScript and XML)이다. AJAX를 구현하는 데에는 JavaScript가 중요한 기술적 요소라 하겠다. 본 논문에서는 사용 규모와 기능의 확대가 예상되는 JavaScript의 시험을 위한 도구를 개념적으로 제안했다. 기존에 제안된 도구나 수동적인 시험은 HTML에 시험을 위한 JavaScript를 삽입하여 브라우저를 통한 시험이 위주였다. 브라우저를 통한 방법은 스크립트를 삽입한 후 HTML을 불러와야 한다는 번거로움이 있고 시험을 위해 삽입된 JavaScript를 제거하는 과정에서 다른 오류가 유입될 수도 있다. 또한 브라우저에서 보여주는 결과만으로는 시험의 확인이 어렵다. 이에 본 논문은 AJAX 환경에서 더욱 사용이 확대되고 있는 자바스크립트를 시험하기 위한 자동 시험 도구 시스템을 제안한다

1. 서 론

웹 2.0과 더불어 이를 지원하는 기술인 AJAX에 대한 관심이 높아지고 있다. 특히 구글이나 아마존 그리고 국내의 유수한 업체들이 경쟁적으로 AJAX 개념을 각 사의 웹 페이지에 적용하여 사용자들에게 높은 편의성과 접근성을 제공하고 있다. 이러한 상황은 웹 어플리케이션과 관련된 업체들에게 커다란 영향을 미치면서 AJAX의 적용을 확대하고 있다.

AJAX를 구현하는데 있어서 핵심적인 개념은 바로 기존의 웹 방법과는 다르게 JavaScript를 이용하여 비동기(Asynchronous) 통신이 가능하도록 한다는 것이다. 또한 JavaScript는 AJAX 구현에 필요한 XHTML(extensible HTML), CSS(Cascading Style Sheets), DOM(Document Object Model), XML, XMLHttpRequest 등과 같은 주요 기술들을 통합하는데도 쓰인다[1]. 이러한 JavaScript의 역할은 기존의 서버 측에서의 서비스에 집중하던 관점을 AJAX 방법에서는 클라이언트 쪽으로 이동하고 있다. 서버의 서비스에만 의존하지 않고 클라이언트 측에서 JavaScript를 실행하여 더욱 빠른 성능을 제공한다. 따라서 AJAX에서의 JavaScript의 기능과 역할은 기존의 웹 어플리케이션 보다 더 확대될 것이고 크기 또한 커질 것이다.

이러한 발전의 기대 속에서 AJAX를 적용하는 웹 개발자는 JavaScript 실행에 대한 정확한 예측과 시험이 필요할 것이다. 만약 JavaScirpt의 실행이 실패한다면 AJAX의 핵심인 비동기 통신의 구현에 오류가 생길 것이고 비즈니스 로직을 포함하고 있는 JavaScirpt의 경우는 더 더욱 클라이언트 측과 서버 측 간의 거래에 치명적인 손실을 초래할 수도 있다. 따라서 JavaScript의 사용의 확대와 함께 이에 맞는 정밀한 시험 기법과 도구들이 필요하다.

이에 본 논문은 AJAX 적용 확대로 인한 JavaScript의 시험 및 연구사례에 대해 논하고 이를 지원하는 시험 도구를 제안한다. 논문의 진행 내용으로 제 2장에서는 AJAX와 기존의 웹 방식에 대한 차이점과 AJAX의 시험에 대한 배경 연구를 서술했다. 제 3장에서는 기존에 제안된 AJAX 시스템 시험 자동화에 대해서 서술했다. 4장에서는 제안된 시험 자동화 도구를 이용해서 실제 기능 시험을 한 후 그 결과에 대해 논했다. 5장에서는 기존의 AJAX 자동 시험 도구를 보완할 수 있는 새로운 시스템에 대한 제안을 설명하고 6장에서 결론을 정리했다.

2. 배경 연구

2.1. AJAX와 기존 웹 방식의 차이점

기존의 웹 방식은 브라우저가 웹서버에서 응답을 보내줄 때까지 기다리는 동기적 통신이었다. 이러한 특성

때문에 웹어플리케이션에서 어떤 기능을 수행하려면 언제나 페이지 전체를 로딩해야 하기에 아무리 작은 정보라 하더라도 웹서버에 일단 요청을 보내면 전체 HTML을 다 운받아야 작업이 끝난다. 반면, AJAX는 브라우저가 웹서버에 요청을 보내면 응답이 올 때까지 기다릴 필요가 없다. XMLHttpRequest 객체의 상태 체크를 통해 웹서버로부터 응답이 왔을 때를 알 수 있고, 이 때 이에 맞는 처리를 해주면 되는 것이다. 이런 방식으로 페이지 전체를 로딩할 필요 없이 HTML의 일부분만을 변경할 수 있다

그림 1은 전통적인 웹 어플리케이션 모델과 AJAX 웹 어플리케이션 모델을 구조이다. 전통적인 웹 어플리케이션 모델과 달리 AJAX 웹 어플리케이션은 AJAX 엔진이 브라우저 클라이언트에 포함되어 있다. AJAX 엔진은 자바스크립트를 실행하며 HTML 문서 처리 엔진이 만든 객체와 자바스크립트가 연동할 수 있도록 제어하여 서버 측에 대한 서비스 없이도 클라이언트 메모리에서 프로그램이 가능하도록 지원한다.

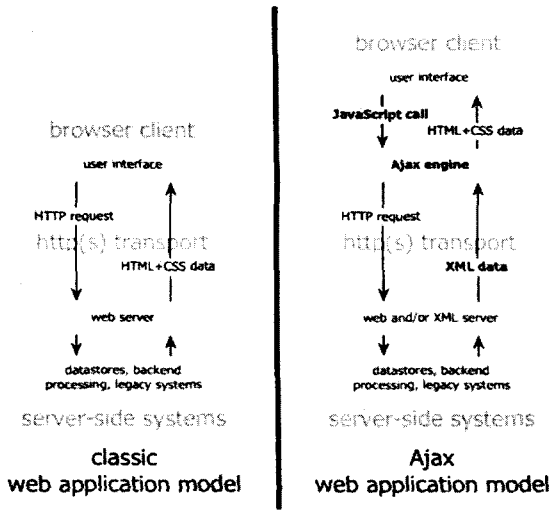

그림 1 웹 어플리케이션 모델

2.2. 스크립트 웹 어플리케이션 시험

웹 어플리케이션을 개발하기 위해 서버 측에서 사용되는 프로그래밍 언어는 C나 Java와 같은 컴파일 언어(Compiled Language)와 스크립트 언어(scripting Language)로 나눌 수 있다[2]. 이 중 스크립트 언어가 컴파일 언어보다 구현 방법이 단순하여 더 배우기 쉽고 개발 비용도 저렴한 것으로 알려져 있다. 그러나 이것이 신뢰도가 더 낮다는 것을 의미하지는 않는다. Prechelt의 연구[3]에서는 일곱 개의 컴파일 언어와 스크립트 언어를 비교 시험하였다. 그리고 스크립트로 구현된 시스템이 C++이나 JAVA로 구현된 시스템 보다 신뢰성이 낮지 않다는 것을 밝혔다. 이와 같은 사실은 어플리케이션에 대한 시험의 시간과 노력은 기대하는 소프트웨어의 품질의 수준에 의해 달라진 다는 것을 의미한다[4].

2.3 JavaScript 컴포넌트 시험

HTML에 내장되어 있는 JavaScript 컴포넌트를 시험하기 위해 xUnit[5]와 같은 형태의 프레임워크를 사용할 수 있다. 이러한 프레임워크는 JavaScript의 작동을 자동화하여 시험 결과를 단위 및 기능 시험을 할 수 있도록 지원한다.

JavaScript를 시험하기 위한 프레임워크는 Scriptaculous와 JsUnit 그리고 IeUint과 nUnit 같은 프레임워크가 있다. 이들은 단위 시험 프레임워크를 포함하고 있고 모두 비슷한 기능과 시험을 지원한다. 테스트 케이스는 메소드들의 시퀀스를 갖고 있는 객체로 구현된다.

Scriptaculous 프레임워크[6]의 경우 testStringMatch()와 같은 ' test' 라는 어두를 포함하고 있는 메소드로부터 시험이 시작된다. 그리고 이 시험은 TestRunner 객체에게 넘겨지게 되고 setup()과 테스트하고자 하는 메소드 그리고 teardown() 메소드의 순서대로 호출이 되면서 시험이 행해진다. JsUnit 프레임워크[7]는 클라이언트 측의 JavaScirpt를 위한 단위 시험 프레임워크이다. 기본적으로 JUnit와 연동이 가능하도록 되어있고 다양한 브라우저와 OS 그리고 하드웨어에 사이에서 자동 시험이 가능하도록 되어있다. nUnit[8]는 다른 프레이워크과 동일한 기능과 시험을 제공하고 .NET 환경에서의 시험을 지원한다.

3. AJAX의 시스템 시험 자동화

AJAX로 구현된 시스템을 시험하기 위한 시험 자동화 환경은 테스트 엔지니어에게 많은 자원을 절약해 준다. 시스템 시험은 사용자가 실제로 시스템을 사용하는 것과 같이 시스템의 전반적인 기능을 작동시키면서 시험을 한다. 시스템 시험의 종류 중 주요한 항목은 시스템의 기능을 확인하는 기능 시험뿐만 아니라 성능과 견고성과 같은 품질도 시험 대상이 된다. 이러한 시스템 시험을 수동으로 하기보다 간편하게 자동으로 수행한다면 많은 비용을 줄일 수 있다. 특히 그림 2와 같이 로봇 스타일의 시험 자동화 도구는 브라우저를 호출하고 브라우저와 상호작용하는 작업을 생성하기 위해 OS의 API를 사용한다[9].

브라우저 컨트롤러는 AJAX 어플리케이션과 통신을 수행하고 어플리케이션을 시험하거나 시험 결과를 평가한다. 이때 브라우저에 내장되어 있는 JavaScript를 시험할 수 있다. 시험하는 방법은 HTML에 JavaScirpt를 시험하기 위한 테스트 하니스(Harness)와 같은 JavaScript 모듈을 작성한다. 그리고 브라우저 컨트롤러를 이용하여 테스트 하니스를 포함하고 있는 HTML을 브라우저로

실행한다. 시험 결과는 브라우저의 출력 결과를 통해 테스트 엔지니어가 확인할 수 있다.

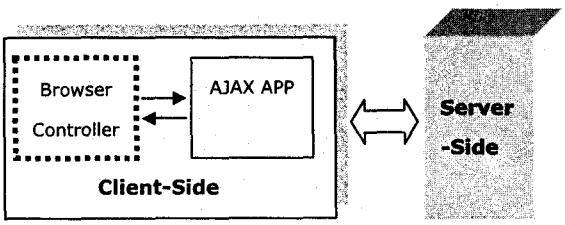

그림 2 브라우저 컨트롤러

4. JavaScript의 기능 시험

JavaScript는 구현 환경과 실행되는 환경이 JAVA와 같은 컴파일 언어와 차이가 있지만 컴파일 언어와 같이 다양한 프로그래밍과 기능의 구현이 가능하다. 이러한 기능들을 시험하기 위해서는 HTML 내부에 시험을 위한 JavaScript를 삽입할 수 있다. 본 논문에서는 JavaScript의 기능 시험을 위해 큐(Queue)를 시험하고 있는 JavaScript의 예를 보이고 있다. 표 1은 기능 시험을 위해 삽입된 JavaScript의 일부를 발췌하고 설명을 덧붙였다. 시험을 위해 구현된 파일 ' QueueTest.html' 은 같은 폴더에서 큐를 구현하고 있는' queue.js' 라는 파일을 불러온다. 그리고 ' QueueTest.html' 의 내부에 있는 JavaScript가 ' queue.js' 의 큐 기능을 시험한다.

표 1 기능 시험 JavaScript

기능 시험을 위한 소스	설명

testEmptyQueue: function() { with(this) {
assertEqual(0, q.size);
var items = q.items();
assertEqual(0, items.length);
}},
	큐의 초기 상태를 검증하기 위한 시험 메소드

testSequence: function() { with(this) {
q.admit("first");
q.admit("second");
assertEqual(2, q.size);
 	
assert(util.membersEqual(["second","first"], q.items()));
 	
assertEqual("first", q.serve());
assertEqual(1, q.size);
assert(util.membersEqual(["second"], q.items()));
 	
q.admit("third");
assertEqual(2, q.size);
- 중략 -
}},
	큐를 구현하고 있는 다양한 메소드를 호출하면서 시험하는 메소드

.....	.....

그림 3은 기능 시험을 수행할 JavaScript를 내장하고 있는 ' QueueTest.html' 파일을 구동한 결과이다. 구동 후 브라우저에서는 시험 결과나 과정에 대한 정보가 하나도 없다. 이러한 방법은 결국 테스트 엔지니어가 브라우저의 출력을 통해서만 시험할 수 있게 한다. 따라서 3장에서 설명한 브라우저 컨트롤러를 사용한 시험 결과와 수동으로 테스트 하니스를 포함한 HTML을 브라우저를 통해 출력시키는 것과 다르지 않다. 다만 그 과정을 브라우저 컨트롤러가 자동화 해 줄 뿐이다.

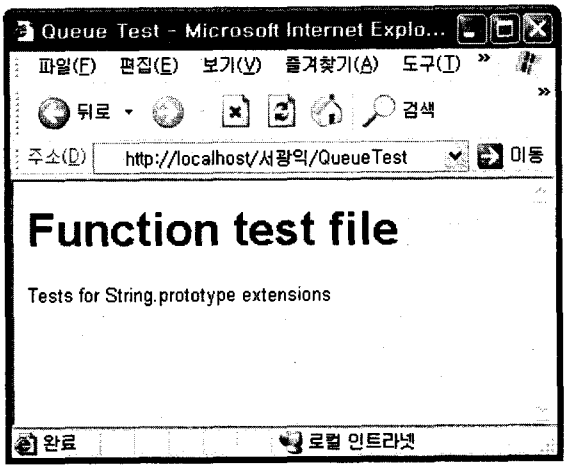

그림 3 큐를 시험한 결과

이러한 방법은 JavaScript 모듈간의 통합 시험 및 모듈간의 상호 작용으로 발생하는 기능에 대한 정확한 시험을 불가능하게 한다. 왜냐하면 HTML에 내장되어 있는 JavaScript 모듈간의 호출 순서 및 조합으로 특정 기능이 구현될 때 호출 순서에 대한 정확한 시험 사례를 추출하기 어렵기 때문이다. 만약 규모가 큰 AJAX 어플리케이션에서는 내장된 JavaScript의 호출 순서에 따라 서로 다르게 작동하는 여러 기능을 시험할 필요가 있을 것이다. 그러나 기존의 브라우저 컨트롤러를 이용한 자동 시험은 HTML 내부에 있는 JavaScript간의 호출 순서를 추적하기 어렵다. 따라서 단순한 브라우저의 출력 결과만으로는 모듈간의 호출 순서를 이용한 시험 사례의 시험을 할 때는 실제 브라우저 작동 시 JavaScript의 호출 순서가 시험사례의 명세대로 호출되었는지 확인하기 어려워 진다.

이에 본 논문에서는 브라우저 컨트롤러를 통한 AJAX 어플리케이션의 시험이 아닌 Ajax 엔진을 독립적으로 접근하여 JavaScript를 시험하는 도구를 제안한다.

5. JavaScript 기능 시험 도구

3장과 4장에서 논한 바와 같이 단순히 브라우저를 호출하는 브라우저 컨트롤러를 이용한 시험은 정밀한 JavaScript의 기능 시험을 어렵게 한다. 또한 테스트 하니스를 삽입하여 시험하는 방법은 시험이 끝난 후 이를 제거하는 과정에서 오류가 유입될 수도 있다. 따라서 브라우저를 이용하지 않으면서 내장된 JavaScript만 독립적으로 시험할 수 있는 JavaScript를 시험하는 도구에서 제공한다면 이러한 문제점들을 해결할 수 있을 것이다.

그림 4은 테스트 하니스를 포함하고 있는 시험 도구가 AJAX 엔진과 통신하고 있는 과정을 개괄적으로 설명하고 있다. 자동 시험 도구는 AJAX 엔진과 통신하면서 HTML에 내장되어 있는 JavaScript 추출하고, 추출된 JavaScript의 리스트를 이용하여 시험 사례를 작성한다.

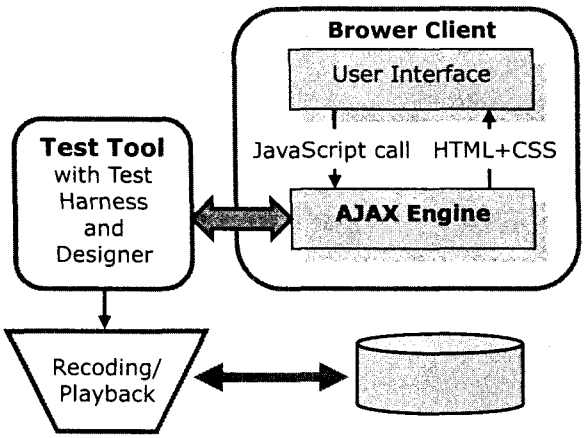

그림 4 기능 시험 도구

JavaScript를 시험하기 위해서 일단 시험 도구는 JavaScript를 HTML에서 추출한다. 그리고 시험 디자이너에 의해 시험 계획 및 시험 사례를 작성한 후 시험 하니스를 통해 JavaScript 함수를 호출하거나 실행된 함수에게 필요한 반환 값을 전달해 주는 스텁(stub)이나 드라이버(driver)의 작성을 도와준다. 또한 필요에 따라 함수를 검사할 수 있는 모듈을 삽입하면서 디자이너에 의해 작성된 시험 사례에 따라 자동으로 시험을 수행한다. 그리고 시험 과정을 저장하여 시험 자료를 시험 슈트(Suite)로 사용하거나 시험 재생이 가능하도록 한다.

6. 결론 및 향후 연구

웹 사이트를 운영하는 여러 업체에는 AJAX 를 도입하여 인터넷 사용자에게 다양한 기능과 UI를 제공하면서 고객을 위한 서비스에 집중하고 있다. 또한 이들의 성공적인 도입을 목격한 다른 업체들도 서둘러 AJAX의 효과에 대해 분석하고 적용을 검토하고 있다. 이러한 AJAX의 성공적인 원인에는 다양한 기술들의 결합이 있다. 그 중에서도 비동기 통신을 가능하게 하고 그 결과 더 빠른 속도로 사용자에게 다양한 서비스를 제공하게 하는 주요 기술은 JavaScript라 하겠다. 따라서 JavaScript를 사용하여 웹을 구현하는 사례는 AJAX를 통해 더 많아 질 것이고 과거보다 많은 역할과 기능을 제공할 것이다. JavaScript의 역할 확장과 더불어 철저하게 시험하고 검증하는 방법도 필요하다. 이에 본 논문에서는 기존에 제안된 JavaScript의 자동 시험 도구에 대한 단점을 서술했고 이를 보완하기 위한 대안이 될 수 있는 새로운 시스템의 개념을 설명했다. 새로 제안한 자동 시험 도구는 브라우저에 테스트 하니스를 삽입하는 것이 아니라 독립된 응용프로그램으로써 HTML에서 JavaScript를 추출하고 추출된 함수들을 이용하여 시험 사례를 설계한 후 자동으로 시험할 수 있는 시스템을 제안했다.

향후 제안한 시스템을 구현하기 위해 시험 사례를 설계하고 계획하는 디자이너와 테스트 하니스를 삽입하기 위한 구체적인 방법에 대해 연구할 예정이다.

참고문헌

[1] J. J. Garrett, Ajax: A New Approach to Web Application, www.adaptivepath.com/publications/essays/archives/000385.php, Feb 2005.

[2] S. Bedi and P. J. Schroeder, Observations on the Implementation and Testing of Scripted Web Applications, System Sciences, Proc. of the the Twenty-Sixhth Hawaii International Conference. vol. 3, pp. 513-522, 1993.

[3] L. Prechelt, " An Empirical Comparison of Seven Programming Languages," IEEE Computer, vol. 33, no. 10, pp. 23-29, 2000.

[4] C. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing: A Context Driven Approach. New York: John Wiley & Sons, Inc., 2002

[5] http://en.wikipedia.org/wiki/XUnit

[6] http://script.aculo.us

[7] http://www.jsunit.net

[8] http://www.nunit.org

[9] M. Mahemoff, Ajax Design Pattern, O' Reilly, June 2006, 1st Edition.