웹 어플리케이션의 사용자 인터페이스 테스팅 자동화에 대한 연구

권영호[0] 최은만
동국대학교 컴퓨터공학과
e-mail : {yhkwon, emchoi}@dgu.ac.kr

A Study on User Interface Testing Automation for Web Applications

Young Ho Kwon[0] Eun Man Choi
Dept. of Computer Engineering, Dongguk University

요 약

인터넷의 폭발적인 성장은 웹 어플리케이션의 증가에 영향을 주었다. 웹 관련 기술의 발전과 다양한 비즈니스 요구사항은 웹 어플리케이션을 점점 더 복잡하게 했고, 따라서 품질과 신뢰성이 중요해졌다. 이 논문에서는 웹 사용자 인터페이스의 테스팅 자동화에 대한 방법을 제시한다. HTML 페이지의 객체와 자바스크립트의 객체를 매핑시켜, 사용자 입력 폼에 대한 테스트 케이스를 자동으로 생성해서 UI 테스팅을 자동화하는 방법을 보여준다. 또한 제안된 방법으로 테스팅 자동화 사례를 보여준다.

1. 서론

인터넷과 웹의 급속한 성장은 기존의 전통적인 어플리케이션이 웹 기반 어플리케이션으로 이행하게 했다. 이는 인터넷을 매개로 한 전자상거래, 엔터테인먼트, 교육 등 우리의 일상을 변하게 하고 있다. 또한 웹의 즉시성(immediacy)은 빠른 제품 개발 요구와 반면에 좋은 품질, 신뢰성을 기대하고 있다. 그러나 전통적인 클라이언트/서버 어플리케이션에 비해 웹 어플리케이션이 점점 더 복잡해지고 있으며, 다양한 웹 기술을 가지고 테스팅과 품질 관리를 하기가 더욱 어려워 졌다[1][2][3].

웹 어플리케이션을 테스트하는 것은 전통적인 소프트웨어와 여러 면에서 차이가 난다. 네트웍 상에서 수행되며 웹 브라우저 기반의 클라이언트와 서버측 컴포넌트로 구성되어 있다. 다양한 웹 기술의 등장, 다중의 환경과 플랫폼이 존재하기 때문에 테스팅을 수행하기가 쉽지 않다.

이 논문에서는 날로 증가하는 웹 어플리케이션을 테스트하는 다양한 방법 중에서 사용자 인터페이스 테스팅을 자동화하는 방안에 대해 제안한다. 여기서는 대부분의 웹 어플리케이션이 웹 브라우저 인터페이스에서 사용자의 입력부분과 마우스의 누름에 대한 반응이 테스트의 주된 관심사로 생각한다. 따라서 사용자 입력 폼(Form), 즉 텍스트박스, 라디오 버튼, 체크박스, 선택박스 등에 대한 테스트 자동화 방안을 제시한다.

2. 관련 연구

2.1 웹 기술과 테스팅

웹 어플리케이션에 관련된 기술에는 인터넷 표준인 HTML, XML, SGML 등이 있고, ASP, JSP, PHP, CGI, Perl 등 서버쪽 스크립트 언어와 클라이언트쪽 스크립트 언어인 JavaScript, VBScript 등이 있다. 또한 COM/DCOM, CORBA, EJB 와 같은 컴포넌트 기술이 있으며, TCP/IP, HTTP, FTP 등과 같은 네트웍 관련 기술, 데이터베이스 접속 등 많은 기술들이 집약되어 있다. 이것은 웹 어플리케이션을 테스트하는데 어렵고 복잡하게 만드는 것들이다.

웹 어플리케이션을 테스트하는 방법은 웹 페이지가 정적이냐 동적이냐에 따라서 달라질 수 있는데, 정적인 경우는 구문 검사와 링크 검사로 가능하지만 동적으로 서버측에서 생성할 경우에는 단위 테스트, 통합 테스트, 시스템 테스트가 필요하다[4].

2.2 웹 어플리케이션 테스팅 도구

웹 어플리케이션을 테스트하는 도구들이 많이 상용화되어 있다. HTML 구문 검사와 하이퍼링크 검사기[5][6]가 있으나 이것들은 웹 어플리케이션에서 중요한 기능성을 테스트할 수 없다.

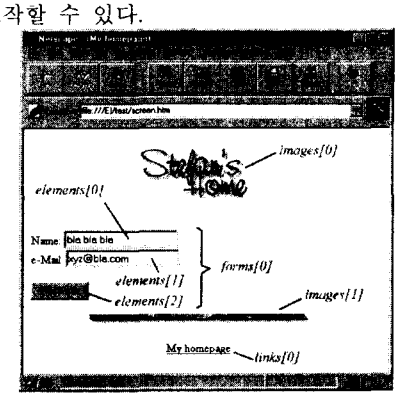
최근에는 자동화된 웹 테스팅 도구들이 나왔는데 성능 시험 및 부하 시험 위주로 되어 있다[7][8].

3. 웹의 사용자 인터페이스 테스트

3.1 HTML 문서의 브라우저 표현

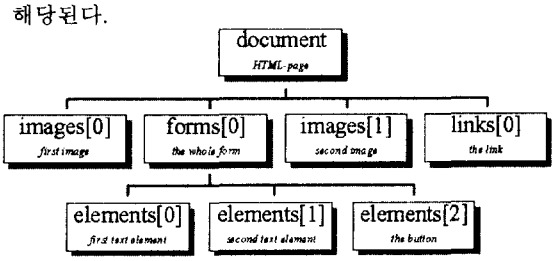
HTML 문서가 웹 브라우저에 로딩될 때, 그 문서의 서로 다른 구성요소는 브라우저 내장 객체로 표현된다. 예를 들면, 하나의 이미지 객체는 HTML 문서의 <img> 태그로부터 생성된다. 브라우저 내장 객체는 HTML 페이지의 구조를 나타내는 계층구조를 하고 있다. 자바스크립트는 웹 페이지상에 나타나는 모든 구성요소를 하나의 계층구조로 조직화한다[9]. 모든 요소는 하나의 객체로 보여지게 된다. 각 객체는 속성과 메소드를 가질 수 있다. 이러한 자바스크립트를 이용하면 쉽게 HTML 문서

객체를 조작할 수 있다.



[그림 1] 브라우저의 HTML 문서 표현[9]

HTML 객체는 [그림 1]에서 보는 것처럼 브라우저 상에 표현된다. 자바스크립트의 관점에서 보면, 브라우저 윈도우는 하나의 윈도우 객체이다. 윈도우 객체에는 메뉴바, 툴바, 상태바와 같은 여러 가지 요소들이 포함된다. 하나의 윈도우안에서 하나의 HTML 문서를 불러들일 수 있는데 이때 로딩되는 웹 문서가 하나의 도큐먼트 객체에 해당된다.



[그림 2] HTML 문서 객체의 계층구조[9]

3.2 HTML 객체에 접근 방법

여기서 서로 다른 객체에 대한 정보와 그 객체를 조작하는 방법에 대해 알아보면, 먼저 해당 객체에 접근하는 방법을 알아야 하는데 [그림 2]의 계층구조에서 각 객체에 붙여진 이름을 볼 수 있다. HTML 문서의 첫번째 이미지에 접근하려면 계층구조의 맨 위의 도큐먼트 객체에서 시작해서 첫번째 이미지는 images[0]를 통해 참조가 된다. 결국 자바스크립트를 이용하면 document.images[0]와 같이 접근할 수 있다.

입력란에 어떤 문자열이 입력되었는지 알려면 입력 객체의 속성과 메소드를 보면 된다. 입력 텍스트 객체의 속성은 value 이다. 객체가 많은 문서에서는 계층구조의 순서상의 숫자로 구분한다면 혼동이 올 수 있을 것이다. 이 문제를 피하기 위해 객체의 이름으로 대신 사용할 수 있다.

```
<form name= " myForm" >
Name : <input type=" text" name=" myText" value=" " >
...
```

[그림 3] HTML 입력 폼 예제

[그림 3]의 예제에서 forms[0]는 'myForm'으로 불려진다 다시 말해서 name = document.forms[0].elements[0].value; 는 name = document.myForm.myText.value; 로 쓸 수 있다.

3.3 웹 사용자 인터페이스 테스팅 자동화

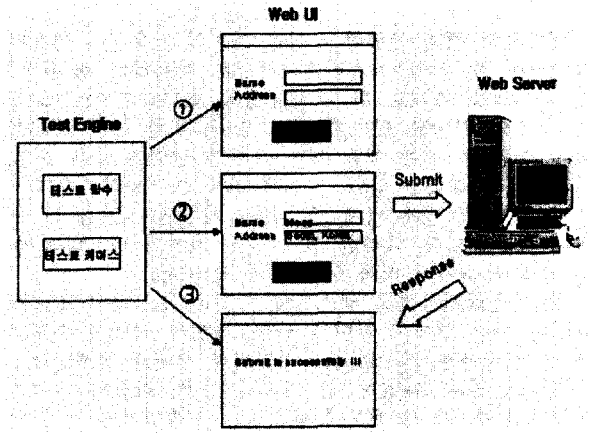
웹 어플리케이션의 사용자는 웹 브라우저의 인터페이스로 웹 서버와 상호작용을 한다. 웹 테스팅이 자동화되지 않은 경우에는 어플리케이션의 단위 테스트(unit test) 위주의 수동 테스팅(manual testing)을 하게 되는데, 이는 많은 시간적인 노력과 테스터가 오류를 범할 수 있다.

이 논문에서는 사용자의 입력 부분이 되는 폼 태그에 대한 부분을 테스팅 자동화를 하려고 한다.

아후 같은 데이터베이스 검색 시나리오를 생각해 보자.

1. 사용자는 검색 버튼을 누른다.
2. 키워드 입력 양식으로 돌아간다.
3. 사용자는 검색어(키워드)를 넣고 전송 한다.
4. 검색 결과를 본다.

이 경우에, HTML 문서에는 2 개의 입력 이벤트(검색 버튼 누름과 키워드 입력)가 일어난다. 추가로 검색 결과로 많은 HTML 문서가 생성된다.



[그림 4] 웹 인터페이스 테스팅 자동화

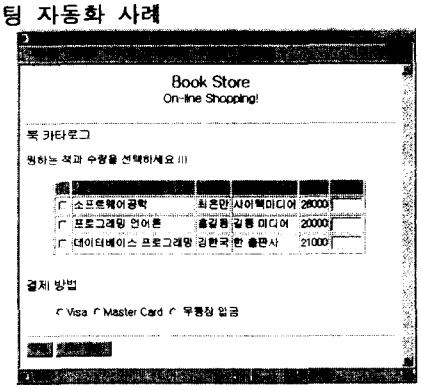
수동으로 테스트할 경우에 데이터베이스 검색 단계별로 많은 반복과정과 매번 다른 테스트 케이스(키워드 값)를 필요로 한다. 그러나 사용자 입력 이벤트에 대해 자바스크립트를 이용해 테스팅을 자동화 할 수 있다. [그림 4]는 테스팅 자동화 과정을 보여주고 있다. 이 과정의 테스트 시나리오를 설명하면,

1. 입력 폼과 전송 버튼을 누르는 이벤트에 대해 자바스크립트를 이용해서 테스트 엔진을 만든다. 테스트 엔진은 테스트 함수와 테스트 케이스로 구성된다.
2. ①에서 테스트 함수가 테스트할 HTML 문서를 호출하고, 테스트 횟수를 설정한다.
3. ②에서 텍스트 필드를 임의의 값으로 채운 후, 폼을 웹 서버로 전송(submit)한다.

4. 처리 결과를 가진 HTML 문서를 받을 때까지 기다린다(response).
5. ③에서 처리 결과가 성공적인지 검사한다.

이렇게 테스트가 한 번 자동화 되면, 다른 테스트 케이스를 가지고 여러 번 수행이 가능해 진다.

3.4 테스팅 자동화 사례



[그림 5] 온라인 서점 웹 어플리케이션

[그림 5]와 같은 온라인 서점에서 책을 주문하는 부분을 테스팅 자동화하는 과정을 보자. 사용자는 주문할 책을 선택하고 수량과 결제방법을 선택해서 주문 버튼을 누르면, 웹 어플리케이션은 주문에 대한 결과를 사용자에게 다시 보여준다.

```
<form method=post action="http://localhost/shop/
onlineShop.php">
 <table border=1>
  <tr bgcolor=skyblue><th><th>책이름<th>저자<th>출판사<th>
가격<th>수량
  <tr><td><input type=checkbox name=bookitem value="B1">
      <td>소프트웨어공학 <td> 최은만 <td> 사이텍미디어
<td> 28000
      <td><input type=text size=5 name='Q1'>
...
     <input type=radio name=payment value="P1">Visa
     <input type=radio name=payment value="P2">Master
Card
     <input type=radio name=payment value="P3">무통장 입금
  <hr>
  <input type=submit value="주문">
  <input type=reset value="다시 작성">
</form>
```

[그림 6] 온라인 서점 입력 폼 부분 HTML 소스

```
function submitAShoppingReq() {
  var tmp = Math.round(Math.random()*10);
  var payment = tmp%3;
  Win.document.forms[0].reset();
  Win.document.forms[0].bookitem[0].checked=true;
  Win.document.forms[0].bookitem[1].checked=true;
  Win.document.forms[0].bookitem[2].checked=true;
  Win.document.forms[0].Q1.value=
Math.round(Math.random()*300);
  Win.document.forms[0].Q2.value=
Math.round(Math.random()*200);
  Win.document.forms[0].Q3.value=
Math.round(Math.random()*100);
  Win.document.forms[0].payment[payment].checked = true;
  Win.document.forms[0].submit();
}
```

[그림 7] 테스트 함수 부분

[그림 7]는 [그림 6]에 대한 테스트 함수이다. 실제 테스팅은 입력 폼의 각각의 파라미터(수량, 결제 방법 등)에 임의의 난수를 발생시켜 주문 처리를 시뮬레이션한다. 그리고 이것은 온라인 서점 어플리케이션에 넘겨져서 처리 결과를 기다리게 된다. 테스팅 횟수를 지정해서 여러 번 테스트를 할 수 있다.

자동화된 테스팅 기법을 알아보았다. 그러나 다른 어플리케이션의 사용자 인터페이스 부분을 위한 테스트 함수를 그에 맞게 다시 만들어야 하는 단점은 있지만 한 번 테스트 스크립트를 만들면 자동으로 여러 번 테스팅을 수행할 수 있다는 장점이 있다.

4. 결론

웹 어플리케이션의 사용자 인터페이스 테스팅의 자동화 방법에 대해 제시하였다. 웹 브라우저 내장 객체와 자바스크립트 객체 사이의 매핑을 통해 사용자 입력 폼에 대한 테스팅을 자동화 할 수 있다는 것을 보여주었다 테스트 케이스를 자바스크립트로 생성하는 랜덤 테스팅 방법을 사용하였다.

향후 연구로는 다양한 사용자 인터페이스에 대한 테스트 스크립트를 자동으로 생성하는 생성기에 대한 연구, 사용자 이벤트를 기록하고 재생해서 테스트에 사용하게 할 수 있게 하는 것과 테스트 결과 보고서를 자동 생성하는 연구가 진행되어야 하겠다.

참고문헌

[1] Robert L. Glass, "Has Web Development Changed the Meaning of Testing? ", StickyMinds.com column, Dec 18, 2000.
[2] Edward Miller, "WebSite Testing", Software Research Inc., 2000.http://www.soft.com/eValid/Technology/White.Papers/website.testing.html
[3] Andrea MacIntosh, Wolfgang Strigel, "The Living Creature - Testing Web Applications", QA Labs Inc., Jun 1, 2000.
[4] 최은만, 권영호, "웹 기반 소프트웨어의 테스트 모델에 관한 연구", 한국정보처리학회 춘계 학술발표논문집 제 8 권 제 1 호, pp. 197-200, 2001.
[5] W3C HTML Validation Service, http://validator.w3.org
[6] Doctor HTML, http://www2.imagiware.com/RxHTML/
[7] Hung Q. Nguyen, *Testing Applications on the Web*, John Wiley & Sons, chap. 17, pp. 337-348, 2001.
[8] imbus test lab., "Test Tool List", 2001. http://www.imbus.de/engl/testlabor/tool-list.html
[9] Stefan Koch, "VOODOO'S INTRODUCTION TO JAVASCRIPT Version 2.5", 1996-1998., http://rummelplatz.uni-mannheim.de/~skoch/js/tutorial.htm