ISSN(Print): 2287-5905

http://dx.doi.org/10.3745/KTSDE.2012.1.1.001

Reengineering Template-Based Web Applications to Single Page AJAX Applications

Jaewon Oh† · Hyeon Cheol Choi†† · Seung Ho Lim††† · Woo Hyun Ahn††††

ABSTRACT

Web pages in a template-based web application (TWA) are automatically populated using a template shared by the pages with contents specific to the pages. So users can easily obtain information guided by a consistent structure of the template. Reduced duplicated code helps to increase the level of maintainability as well. However, TWA still has the interaction problem of classic web applications that each time a user clicks a hyperlink a new page is loaded, although a partial update of the page is desirable. This paper proposes a reengineering technique to transform the multi-page structure of legacy Java-based TWA to a single page one with partial page refresh. In this approach, hyperlinks in HTML code are refactored to AJAX-enabled event handlers to achieve the single page structure. In addition, JSP and Servlet code is transformed in order not to send data unnecessary for the partial update. The new single page consists of individual components that are updateable independently when interacting with a user. Therefore, our approach can improve interactivity and responsiveness towards a user while reducing CPU and network usage. The measurement of our technique applied to a typical TWA shows that our technique improves the response time of user requests over the TWA in the range from 1 to 87%.

Keywords : Single Page Application, AJAX, Web Template

1. 서 론

웹 페이지 개발 생산성을 높이기 위해 많은 웹 사이트가 웹 페이지의 골격을 나타내는 템플릿에 구체적인 내용을 채워 페이지를 생성한다[1]. 이러한 템플릿 기반 웹 애플리케이션(TWA)의 동작 방식이 Fig. 1에 나와 있다. 15행 <jsp:include> 태그에 지정된 경로에 따라 서로 다른 페이지가 생성된다. TWA를 통해 사용자는 일관된 UI를 제공받으며 정보를 쉽게 파악할 수 있다. 또한 코드 중복이 줄고 모듈화가 이루어져 유지보수와 재사용 수준이 향상된다. 그러나 TWA는 사용자의 요청 시 페이지 중 일부분만 변경될 필요가 있지만 전체 페이지가 새 페이지로 업데이트되는 문제가 있다.

위와 같이 TWA를 포함한 전통적인 웹 애플리케이션은 사용자와의 상호작용 모델로 사용자 요구마다 전체 인터페이스가 리프레시되는 멀티 페이지 애플리케이션(Multi-page application, MPA) 모델을 사용하고 있다. 이와 달리 단일 페이지 애플리케이션(Single page application, SPA)은 데스크톱 애플리케이션처럼 페이지 전환 없이 하나의 페이지 안에서 서비스를 제공하는 웹 애플리케이션이다[2]. SPA 페이지는 독립적으로 갱신이 가능한 컴포넌트들로 구성되며, 처음 페이지 로드 시 필요한 코드(HTML, JavaScript, CSS 등)를 다운로드한 후 페이지 전환 없이 현재 페이지에서 일부만을 갱신하며 사용자의 요구를 만족시킨다. 이때 사용자의 요청과 부분 업데이트 처리를 위해 AJAX[3]를 이용한다. 따라서 SPA는 페이지 전환으로 발생하는 네트워크와 CPU 사용 시간을 감소시키고, 비동기적인 페이지 업데이트로 사용자의 체감 성능을 향상시킬 수 있다.

웹 애플리케이션 재공학 분야에서 2000년대 중반까지 화두는 레거시 애플리케이션을 웹 애플리케이션으로 재공학하는 연구였고, 최근에 레거시 MPA를 SPA로 재공학하는 연구가 시작되었다. [2]는 UI 관점에서 MPA의 웹 페이지들을 분석하고 클러스터링하여 SPA의 페이지를 구성하는, 독립적으로 갱신 가능한 UI 컴포넌트를 식별해낸다. 빅뱅 방식으로 재공학하는 [2]와 달리 [4]는 리팩토링을 적용해 UI를 조금씩 개선하는 점진적인 방식을 취한다. [5]는 오프라인 모드, SPA와 같은, RIA(Rich Internet Application)가 지니는 특징을 정리하고 레거시 웹 애플리케이션을 RIA로 재공학하기 위한 프레임워크를 제안한다. 위 연구를 포함하여 MPA를 SPA로 재공학하는 기존 연구는 주로 UI 재설계에 초점을 맞추어 ASP, JSP, PHP와 같은 서버 사이드 코드를 고려하지 않는다. 그러나 페이지 부분 업데이트를 위해서는 AJAX 요청의 응답으로 갱신에 필요한 데이터만 전송하도록 서버 사이드 코드를 수정해야 한다.

본 논문은 Java 기반 TWA를 SPA로 변환(transformation)하기 위한 재공학 방법을 제안한다. 제안 기법은 우선 페이지 전환을 제거하기 위해 HTML 요청을 AJAX 요청으로 변환한다. 아울러 갱신에 필요한 데이터만 주고받기 위해 처음으로 웹 페이지를 로드할 때를 제외하고 템플릿이 전송되지 않아야 한다. 이를 위해 클라이언트 측면에서는 HTML, JavaScript 코드에서 템플릿 요청 코드를 제거한다.

※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2011-0023224).
† 종신회원 : 가톨릭대학교 컴퓨터정보공학부 조교수
†† 준 회 원 : 가톨릭대학교 컴퓨터정보공학부 학사과정
††† 정 회 원 : 한국외국어대학교 디지털정보공학과 조교수
†††† 정 회 원 : 광운대학교 컴퓨터소프트웨어학과 부교수
논문접수 : 2012년 6월 27일
심사완료 : 2012년 8월 3일
* Corresponding Author : Woo Hyun Ahn(whahn@kw.ac.kr)

서버 측면에서는 JSP, Servlet 코드에서 템플릿 전송 코드를 제거한다.

2장에서 제안하는 재공학 기법을 설명한다. 3장에서 성능 검증을 위해 전형적인 TWA를 대상으로 실험한 결과를 보여준다. 4장에서 결론을 제시한다.

```
1: <%@page contentType="text/html; charset=euc-kr" %>
2: <HTML>
3:   <HEAD> <TITLE> 뉴 출판사 </TITLE> </HEAD>
4:   <BODY>
5:     <H1> 뉴 출판사 </H1>
6:     <TABLE border=1>
7:       <TR>
8:         <TD width=190 valign=top>
9:           <A HREF="template.jsp?CONT_PATH=intro.html">회사 소개</A><BR>
10:          <A HREF="books-info">책 정보</A><BR>
11:          <A HREF="template.jsp?CONT_PATH=BBSInput.html">게시판 글쓰기</A><BR>
12:          <A HREF="bbs-list">게시판 글읽기</A><BR>
13:        </TD>
14:        <TD valign=top width=650>
15:          <jsp:include page="${param.CONT_PATH}" />
16:        </TD>
17:      </TR>
18:    </TABLE>
19:    <H5>Copyright@ 2010-2012 뉴 출판사(주)</H5>
20:  </BODY>
21:</HTML>
```

A. Template written in JSP (template.jsp)

게시판 글쓰기((a)11행) 링크 선택 시 BBSInput.html을 동적으로 포함하여 생성한 페이지

B. Web page generated from template.jsp template

Fig. 1. Example of template-based web application

```
1~8: Fig. 1A 1-8 라인과 동일
9:   <A onclick="getIntro()">회사 소개</A><BR>
10:  <A onclick="getBookInfo()">책 정보</A><BR>
11:  <A onclick="getBBSInput()">게시판 글쓰기</A><BR>
12:  <A onclick="getBBSList()">게시판 글읽기</A><BR>
13-14: Fig. 1A 13-14 라인과 동일
15:  <span id="contents"></span>
16-21: Fig. 1A 16-21 라인과 동일
```

"게시판 글쓰기" 클릭 시 getBBSInput() 이벤트 핸들러가 작동하여 AJAX 요청 발생됨. 추후 비동기적으로 전달되는 AJAX 응답을 가지고 id가 contents인 span 요소 업데이트됨. Fig. 1B와 URL 제외하고 페이지 look&feel 동일함

Fig. 2. Example of single page application

Table 1. Addresses classified according to link types

	직접	간접
정적	<A HREF="template.jsp?CONT_PATH=Intro.html">	<A HREF="books-info">
동적	<A HREF="template.jsp?CONT_PATH=BBSItemView.jsp?SEQ_NO=${BBS_LIST.seqNo[cnt]}">	<A HREF="bbs-list?LAST_SEQ_NO=${BBS_LIST.seqNo[BBS_LIST.listSize-1]}">

2. 재공학 기법

기존 Java 기반 TWA를 SPA로 재공학하기 위해 제안하는 기법을 절차 순으로 각 절에서 설명한다.

2.1 갱신 컴포넌트 식별 및 아이디 부여

우선 템플릿에서 독립적으로 갱신 가능한 UI 컴포넌트를 식별할 필요가 있다. TWA에서 웹 페이지는 템플릿에 해당 페이지에 특화된 데이터를 결합하여 생성된다. 이 결합을 위해 일반적으로 템플릿에서 <jsp:include> 표준 액션을 사용한다. 따라서 <jsp:include> 수행 결과가 차지하는 영역을 독립적으로 갱신 가능한 컴포넌트로 볼 수 있다. 페이지 리프레시 대신 현재 페이지 부분 갱신을 위해 이 영역을 혹은 태그로 변환하고 id 값을 설정한다. 이유는 사용자와의 상호작용 시 혹은 영역만을 변경하는 단일 페이지를 구현하기 위해서이다. 이러한 변환 과정이 Fig. 1A의 15행, Fig. 2의 15행에 나와 있다.

2.2 하이퍼링크를 onclick 이벤트 핸들러로 변환

페이지 전환을 제거하기 위해 A 태그의 링크를 AJAX을 활용하는 이벤트 핸들러로 변환한다. Fig. 1A의 9-12행, Fig. 2의 9-12행에 변환 과정이 나와 있다.

이 때, 링크를 템플릿 사용 방식에 따라 두 가지로 나눌 수 있다. 템플릿 주소에 CONT_PATH 파라미터를 붙인 형태(Fig. 1A 11행)와 템플릿이 아닌 서블릿을 호출하는 주소 형태(Fig. 1A 10행)이다. 앞으로 전자를 템플릿 직접 이용, 후자를 템플릿 간접 이용이라 칭한다. 후자의 경우 서블릿 처리 과정 중에 페이지 생성을 위해 템플릿을 사용한다.

링크는 또한 주소 생성 시점 기준으로 나눌 수 있다. 첫 번째는 정적 링크로써 페이지의 주소가 실행 전에 결정되는 유형이고, 두 번째는 동적 링크로써 실행 시 생성되는 유형이다. 동적 링크의 예로 게시판에서 게시글 보기 서비스를 제공하기 위하여 실행 시에 생성한 게시글의 id를 이용해 생성하는 링크를 꼽을 수 있다. 따라서 본 논문은 총 4가지 경우(Table 1 참조)에 따른 변환 방법을 고려한다.

2.3 템플릿 직접 이용하는 정적 링크 변환

이 링크 형식은 "템플릿주소?Include되는 페이지 주소"이다. 페이지 전환을 제거하기 위해 A 태그의 HREF 속성을 삭제하여 Fig. 3A와 같이 변환한다. 한편, 현재 페이지 부분 업데이트에 필요한 데이터만 얻기 위해 HREF 속성에 CONT_PATH 값으로 지정되어 있던 Intro.html 주소를 이벤트 핸들러 코드에서 XMLHttpRequest 객체의 open 함수 인자로 사용한다(Fig. 3B 4행). 만약 Fig. 3A에서 1행 대신 <A id="intro">와 같은 방법을 사용하면 Fig. 4와 같은 이벤트 핸들러 등록 함수가 필요하다.

```
<A HREF= "template.jsp?CONT_PATH=Intro.html" >
```

↓

```
1: <A onclick= "getIntro()" >
```

A. HTML/JSP

```
1: function getIntro(){
2:    var spanElement=document.getElementById( "contents" );
3:    var xhr=new XMLHttpRequest();
4:    xhr.open( "GET" , "Intro.html" );
5:    xhr.onreadystatechange=function(){
6:       if(xhr.readyState!=4) return;
7:       spanElement.innerHTML = xhr.responseText;
8:    }
9:    xhr.send(null);
10:}
```

B. JavaScript

Fig. 3. Static link directly using template

```
window.onload = initPage;
function initPage(){
      intro=document.getElementById( "intro" )
            .addEventListener( "click" , getIntro, false);
}
```

Fig. 4. Event handler registration

```
<A HREF= "books-info" >
```

↓

```
<A onclick= "getBookInfo()" >
```

A. HTML/JSP

```
function getBookInfo(){
   var spanElement=document.getElementById( "contents" );
   var xhr=new XMLHttpRequest();
   xhr.open( "GET" , "books-info" );
   xhr.onreadystatechange=function(){
      if(xhr.readyState!=4) return;
      spanElement.innerHTML = xhr.responseText;
   }
   xhr.send(null);
}
```

B. JavaScript

```
RequestDispatcher dispatcher =
request.getRequestDispatcher( "template.jsp?CONT_PATH=BooksI
nfoView.jsp" );
```

↓

```
RequestDispatcher dispatcher =
request.getRequestDispatcher( "BooksInfoView.jsp" );
```

C. Servlet

Fig. 5. Static link indirectly using template

2.4 템플릿 간접 이용하는 정적 링크 변환

이 링크는 Fig. 5A의 books-info처럼 템플릿을 호출하는 서블릿의 URL을 의미한다. A 태그의 HREF을 삭제하여 Fig. 5A, 5B와 같이 변환한다. 이 서블릿은 비즈니스 로직을 수행한 후 템플릿을 이용하여 페이지를 업데이트한다. 이를 위해 호출할 페이지 주소를 지정하고 RequestDispatcher를 얻는다. 변경할 데이터만 웹브라우저로 전송하기 위해 이 주소 값에서 템플릿 부분을 삭제하며, 수정 과정은 Fig. 5C와 같다.

2.5 템플릿 직접 이용하는 동적 링크 변환

예를 들면 게시판에서 게시글을 볼 때 URL이 실행 시 생성되는 게시글의 id 값을 지니는 경우이다. Fig. 6A에서 ${BBS_LIST.seqNo[cnt]}가 이러한 id이다. Fig. 6처럼 변환한다. 주목할 점은 id를 이벤트 핸들러로 전달한다는 것이다.

```
<A
HREF= "template.jsp?CONT_PATH=BBSItemView.jsp?SEQ_NO=${BB
S_LIST.seqNo[cnt]}" >
```

↓

```
<A onclick= "getBBSItem(${BBS_LIST.seqNo[cnt]})" >
```

A. HTML/JSP

```
function getBBSItem(seqNo){
   var addr= "BBSItemView.jsp?SEQ_NO=" +seqNo;
   var spanElement=document.getElementById( "contents" );
   var xhr=new XMLHttpRequest();
   xhr.open( "GET" ,addr);
   xhr.onreadystatechange=function(){
      if(xhr.readyState!=4) return;
      spanElement.innerHTML=xhr.responseText;
   }
   xhr.send(null);
}
```

B. JavaScript

Fig. 6. Dynamic link directly using template

2.6 템플릿 간접 이용하는 동적 링크 변환

템플릿이 아닌 서블릿 호출 시 서블릿 인자 값이 호출 때마다 달라 주소 값이 동적으로 생성되는 경우이다. Fig. 7A

```
<A
HREF= "bbs-list?LAST_SEQ_NO=${BBS_LIST.seqNo[BBS_LIST.listSi
ze-1]}" >
```

↓

```
<A
onclick= "getBBSList(${BBS_LIST.seqNo[BBS_LIST.listSize-1]})" >
```

A. HTML/JSP

```
function getBBSList(seqNo){
  var addr= "bbs-list?LAST_SEQ_NO=" + seqNo;
  var spanElement=document.getElementById( "contents" );
  var xhr=new XMLHttpRequest();
  xhr.open( "GET" ,addr);
  xhr.onreadystatechange=function(){
    if(xhr.readyState!=4) return;
    spanElement.innerHTML=xhr.responseText;
  }
   xhr.send(null);
}
```

B. JavaScript

```
RequestDispatcher dispatcher =
request.getRequestDispatcher( "template.jsp?CONT_PATH=BBSLi
stView.jsp" );
```

↓

```
RequestDispatcher dispatcher =
request.getRequestDispatcher( "BBSListView.jsp" );
```

C. Servlet

Fig. 7. Dynamic link indirectly using template

의 ${BBS_LIST.seqNo[BBS_LIST.listSize-1]}가 이러한 인자이다. 이 형식은 A 태그의 링크를 onclick 이벤트 핸들러로 대체한다. 템플릿을 간접적으로 이용하므로 서블릿에서 Dispatch되는 부분 또한 변환한다. 수정 과정은 Fig. 7에 나와 있다.

3. 실험 및 결과 분석

검증을 위해 [6]에서 소개하는, 전형적인 TWA를 대상으로 실험하였다. 실험 정보가 Table 2에 나와 있다.

성능 지표로 응답시간을 사용한다. 이를 위해 구체적으로 TWA는 onload time(HTTP 요청부터 페이지가 모두 로드되어 onload 이벤트가 발생할 때까지 시간)을 이용한다[7]. 한편, SPA는 페이지 일부 업데이트의 경우 onload 이벤트가 발생하지 않으므로, 요청부터 DOM tree 업데이트 완료까지의 시간을 사용한다. 예를 들면, Fig. 3A 1행부터 Fig. 3B 7행 실행 종료까지의 시간을 의미한다. 10번 수행하였고 측정 결과(평균)가 Fig. 8, 9에 나와 있다. 첫 페이지를 제외하면 기존 TWA보다 SPA의 성능이 응답시간 측면에서 1~87% 개선되었다. 몇 가지 분석 내용을 정리하면 Table 3과 같다.

Table 3. Experimental result analysis

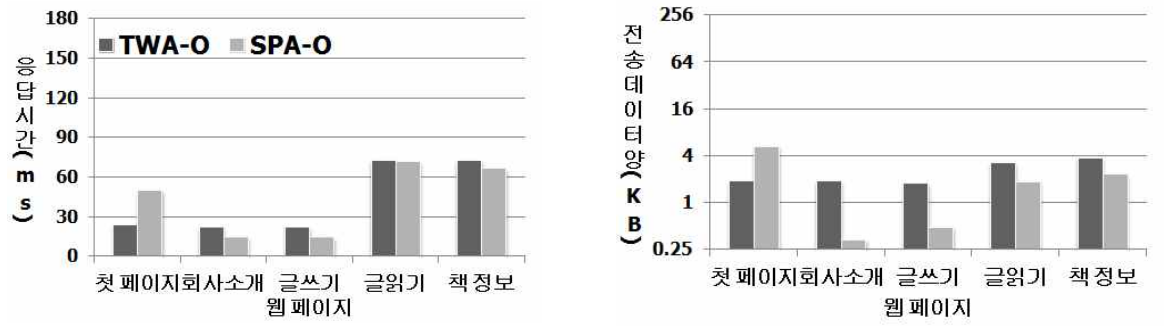
Item	Finding
SPA-O vs SPA-M	첫 페이지 로드 시 이미지 유무로 응답시간 차이 크지만 이 후 상호작용 시에는 부분 업데이트만 동일하게 이루어지므로 전송 데이터 양과 응답시간이 비슷함
TWA-O vs TWA-M	매번 전체 페이지 갱신이어서 이미지 사용 시 응답시간이 떨어짐
SPA vs TWA	SPA가 첫 페이지 로드 시 TWA에 비해 느림. JavaScript 로드 및 처리 때문. 그러나 일단 로드 후에는 TWA에 비해 응답시간이 향상됨.
글읽기, 책정보 응답시간	데이터양에 비해 응답시간이 느린 이유는 JDBC로 DB에 접근해서 데이터를 획득하는 자바빈 객체의 수행 시간 때문.

4. 결 론

본 논문은 Java 기반 TWA를 SPA로 재공학하는 방법을 제안한다. 이 기법은 페이지 전환을 제거하기 위해 HTML 요청을 AJAX 요청으로 변환한다. 아울러 갱신에 필요한 데이터만 주고받기 위해 처음으로 웹 페이지를 로드할 때를 제외하고 템플릿이 전송되지 않아야 한다. 이를 위해 HTML, JavaScript, JSP, Servlet 코드를 수정한다. 성능 측정 결과 기존 TWA보다 재공학된 SPA의 성능이 응답시간 측면에서 최대 87% 향상되었다. 향후 연구로는 제안 기법을 지원하는 도구의 개발을 꼽을 수 있다.

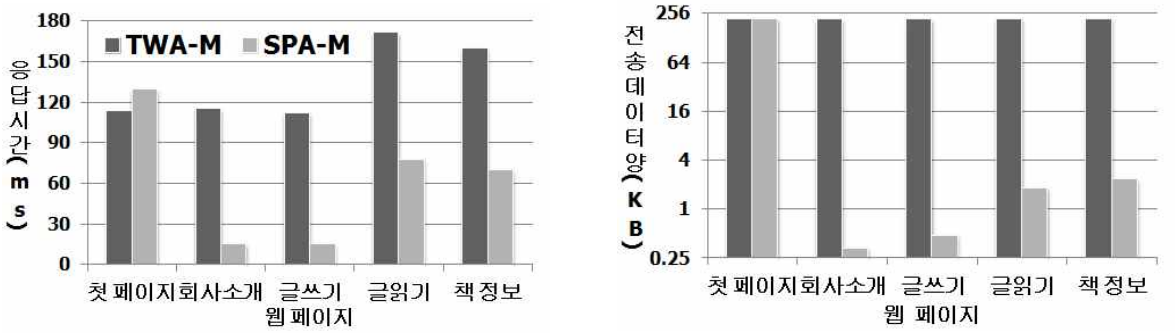
Table 2. Experimental testbed

TWA & SPA	환경 정보		
		web server	client
▣ TWA-O: 출판사 웹 애플리케이션으로 회사소개와 책 정보, 게시판(글쓰기와 글읽기) 페이지 제공. 아울러 "첫 페이지"가 제공되며 실험 시 가장 먼저 접근하는 페이지이며 템플릿만 보여줌. ▣ TWA-M: TWA-O 웹 페이지 크기가 10KB 미만이어서 실제 웹 사이트 크기 반영위해 TWA-O 템플릿에 210KB 이미지를 추가한 웹 애플리케이션 ▣ SPA-O, SPA-M: TWA-O, TWA-M을 제안 기법 적용하여 재공학한 것	CPU	AMD 애슬론II-X2 250	Intel i5 760 2.8GHz
	RAM	DDR3 4GB	DDR3 8GB
	OS	Windows 7 32Bit	windows 7 64bit
	DB	Cubrid 2008 R4.1	
	container	Tomcat7.0	
	browser		chrome 19.0.1084.56



A. Response time B. Transferred data

Fig. 8. Performance of TWA-O, SPA-O



A. Response time B. Transferred data

Fig. 9. Performance of TWA-M, SPA-M

참 고 문 헌

[1] Chulyun Kim et al., "TEXT: Automatic Template Extraction from Heterogeneous Web Pages," IEEE TKDE, Vol.23, No.4, 2011.

[2] Ali Mesbah et al., "Migrating Multi-page Web Applications to Single-page Ajax Interfaces," Proc. of the 11th CSMR, 2007.

[3] Jesse James Garrett, "Ajax: A New Approach to Web Applications," Adaptive Path, 2005.

[4] Gustavo Rossi et al., "Refactoring to Rich Internet Applications. A Model-driven Approach," Proc. of the 8th ICWE, 2008.

[5] Roberto Rodriguez-Echeverria et al., "Modernization of Legacy Web Applications into Rich Internet Applications," Proc. of the 11th intl conf. on Current Trends in Web Engineering, 2012.

[6] Y. Kim, "JSP & Servlet for Java Programmers," Chapter 13, Hanbit Media, 2010.

[7] Dong-Min Kang, "Web Performance Optimization : Today and Tomorrow," Communications of KIISE, Vol.30, No.5, 2012.

오 재 원

e-mail : jwoh@catholic.ac.kr

2004년 서울대학교 전기컴퓨터공학부 (박사)

2004년~2007년 삼성전자 기술총괄 소프트웨어연구소 책임연구원

2007년~현 재 가톨릭대학교 컴퓨터정보공학부 조교수

관심분야 : 소프트웨어 공학, 웹 공학, 시스템소프트웨어

최 현 철

e-mail : choihc7@naver.com

2007년~현 재 가톨릭대학교 컴퓨터정보공학부 학사과정

관심분야 : 웹 공학, 소프트웨어 공학

임 승 호

e-mail : slim@hufs.ac.kr
2008년 KAIST 전기 및 전자공학과(박사)
2008년~2010년 삼성전자 메모리사업부 책임연구원
2010년~현 재 한국외국어대학교 디지털 정보공학과 조교수
관심분야 : 임베디드시스템, 파일시스템, 플래시메모리 소프트웨어

안 우 현

e-mail : whahn@kw.ac.kr
2003년 KAIST 전자전산학과(박사)
2003년~2005년 삼성전자 기술총괄 소프트웨어연구소 책임연구원
2006년~현 재 광운대학교 컴퓨터소프트웨어학과 부교수
관심분야 : 운영체제, 임베디드시스템, 시스템소프트웨어, 시스템보안

단일 페이지 AJAX 애플리케이션을 위한 템플릿 기반 웹 애플리케이션 재공학 기법

오 재 원[†] · 최 현 철[††] · 임 승 호[†††] · 안 우 현[††††]

요 약

템플릿 기반 웹 애플리케이션(TWA)은 웹 페이지들의 공통부분을 단일 코드 파일(템플릿)로, 각 페이지에 특화된 부분은 별도의 파일로 관리한다. HTTP 요청이 발생하면 이 두 종류 파일을 동적으로 조합하여 웹 페이지를 생성한다. 이를 통해 사용자에게 일관된 UI을 제공하며 코드 중복을 제거해 유지보수성을 향상시킨다. 그러나 TWA는 사용자의 요청 시 현재 페이지에서 바뀌어야 할 부분이 일부지만 페이지 전환이 이루어져 갱신이 불필요한 데이터까지 전송되고 처리되는 문제가 있다. 본 논문은 Java 기반 TWA를 대상으로 이 문제를 해결한다. 제안 기법은 페이지 전환을 제거하기 위해 HTML에 있는 하이퍼링크를 AJAX를 이용하는 JavaScript 이벤트 핸들러로 변환한다. 또한 갱신에 필요한 데이터만 반환하기 위해 JSP, Servlet 코드에서 템플릿 전송 코드를 제거한다. 따라서 이 기법은 페이지 전환으로 발생하는 네트워크와 CPU 부하를 감소시키며, 비동기적인 페이지 업데이트로 사용자의 체감 성능을 향상시킨다. 성능 검증을 위해 전형적인 TWA를 대상으로 이 기법을 적용하였다. 성능 측정 결과 기존 TWA보다 응답시간을 최대 87% 개선하였다.

키워드 : 단일 페이지 애플리케이션, AJAX, 웹 템플릿