웹페이지 코드 분리를 위한 리마커블 처리기의 설계 및 구현

강동현, 이관용
한국방송통신대학교 평생대학원 정보과학과
e-mail: idongh@paran.com, kylee@knou.ac.kr

Design and Implementation of Remarkable Processor for Web page Code separation

Dong-Hyeon Kang, Kwan-Yong Lee
Dept. of Computer Science
Graduate School, Korea National Open University

요 약

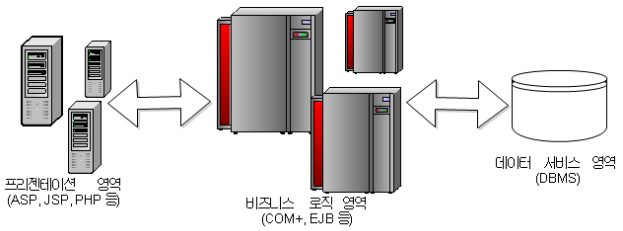
웹프로그래밍 환경에서 프리젠테이션 로직을 구성하는 서버처리 웹페이지에는 디자인을 위한 HTML 요소와 로직을 구현하기 위한 서버측, 클라이언트측 스크립팅 요소가 혼합되어 있다. 이와 같은 경우 스파게티 코딩에 의해 웹디자이너와 웹프로그래머간의 작업의 경계가 불분명해지고 디자인과 로직의 구성이 정렬되지 못하는 문제점이 발생할 수 있다. 본 논문에서는 이러한 문제점을 해결하고 웹페이지의 가독성과 재사용성을 향상시키기 위한 리마커블 코드 분리 모델을 제안한다. 이를 통해 웹페이지 수준에서 HTML 요소와 스크립팅 요소를 완전하게 분리하고 관련 연구와 차별화되는 플랫폼 독립적이고 언어 중립적인 웹페이지 모듈화 과정을 소개하고자 한다.

1. 서론

1.1 연구배경 및 목적

웹 기반의 시스템 환경은 그림 1과 같이 3계층으로 구성하는 것을 생각해 볼 수 있다.[1]

이와 같은 아키텍처를 바탕으로 할 때 프리젠테이션 로직과 비즈니스 로직의 분리는 가능하지만 프리젠테이션 로직안에 여전히 디자인 요소와 로직 요소가 혼재되어 있으므로 웹페이지 코드의 가독성을 높이고 디자인과 로직 개발 작업의 시간종속성을 개선하여 생산성을 향상시키는 연구가 필요하다.[2]



(그림 1) 3계층 웹 기반 시스템

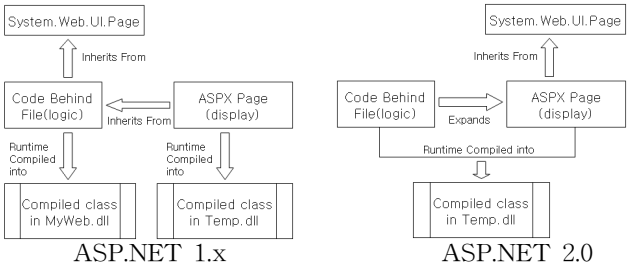
일반적인 서버처리 웹페이지에는 HTML 요소와 스크립팅 요소가 혼합되어 있으므로 스파게티 코딩에 의해 웹디자이너와 웹프로그래머간의 작업의 경계가 불분명해지고 디자인과 로직의 구성이 정렬되지 못하는 문제점이 발생할 수 있다.[3]

본 논문에서는 이러한 문제점을 해결하고 웹페이지 가독성과 재사용성을 향상시키는 리마커블 코드 분리 모델을 제안한다. 이를 통해 관련 연구와 차별화되는 플랫폼 독립적이고 언어 중립적인 웹페이지 모듈화 과정을 소개하는 것이 본 논문의 목표이다.

1.2. 관련 연구와의 비교

1.2.1 Inheritance 접근법

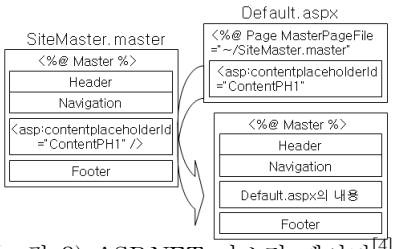
가. ASP.NET 코드 비하인드



(그림 2) ASP.NET 코드 비하인드 모델 비교[4]

ASP.NET 2.0에서는 이전 버전에 비해 향상된 코드 비하인드(Code Behind) 모델을 소개하였는데 이를 Code Beside라고도 부른다.[4]

나. ASP.NET 마스터 페이지



(그림 3) ASP.NET 마스터 페이지[4]

마스터 페이지(Master Page)는 공통 사용자 인터페이스의 재사용성을 개선한 것으로 시각적인 상속(Visual Inheritance)을 가능하게 하는 기술이다.[4]

1.2.2 Template 접근법

여기서는 기존의 서버처리 웹페이지 기술을 보완하기 위해 사용되는 Smarty의 웹페이지 구성을 예로 들었다. Smarty는 Template Engine이며 PHP를 위한 Template/Presentation Framework이다.[5]

<표 1> Smarty의 웹페이지 구성방식[5]

<table> {section name=art loop=$article} <tr> <td>{$article[art].headline}<td> <td>{$article[art].date}<td> <td>{$article[art].author}<td> </tr> {/section} </table>	<table> <tr> <td>How the west was won<td> <td>Dec 2, 1999<td> <td>John Wayne<td> </tr> <tr> <td>Team loses, Coach quits<td> <td>Feb 2, 2002<td> <td>John Smith<td> </tr> <tr> <td>Gourmet Cooking<td> <td>Jan 23, 1954<td> <td>Betty Crocker<td> </tr> </table>
루프를 갖는 template	template에서 생성된 출력물

1.2.3 웹페이지 구성 방식간의 비교

<표 2> 웹페이지 가독성과 재사용성 비교

<표 3> 플랫폼 독립성과 언어 중립성 비교

디자인파일 내용 기준	가독성(웹디자이너 입장)	재사용성
스파게티 코딩	X (로직 요소 노출)	X
include 코딩	X (로직 요소 노출)	△
Smarty	△ (자체 문법 구문 사용)	O
ASP.NET	△ (커스텀 태그 사용)	O
리마커블 모델	O (HTML 주석 사용)	O

리마커블 모델은 역공학과 모듈화 과정에서 디자인과 로직을 분리하고 전처리를 통해 최종 웹페이지를 생성하므로 플랫폼 독립적이고 언어 중립적인 접근법이다. 그러나 리마커블 처리기에 준하는 도구가 있어야만 적용할 수 있다는 단점이 있다.

디자인과 로직의 인터페이스	플랫폼 독립성	언어 중립성
include 코딩	X	X
Smarty	X	X
ASP.NET	X	O
리마커블 모델	O	O

1.3 연구범위 및 방법

반복적으로 재사용한다는 가정하에 웹페이지 예제 하나를 선정하여 역공학을 수행하고 모듈화 과정과 최종 웹페이지 생성을 통해 리마커블 모델과 리마커블 처리기의 타당성을 검토하고자 한다.

2. 논문의 제안

2.1 용어의 정의

2.1.1 리마커블 코드 분리 모델

웹페이지를 모듈화하여 효과적으로 재사용하기 위한 절차적 방법을 말하며 약칭하여 리마커블 모델이라 부른다. 리마커블 모델은 반복적으로 재사용되는 웹페이지에 적용하는 것을 원칙으로 한다.

2.1.2 리마커블 처리기

리마커블 처리기는 역공학기와 전처리기로 구성된다. 역공학기는 기존 웹페이지를 입력받아 디자인파일과 로직파일로 분리하여 출력시키는데 식별자로 HTML 주석(alpha태그, omega태그)을 포함시킨다.

전처리기는 이러한 식별자를 이용하여 디자인파일과 로직파일을 최종 웹페이지로 생성한다. 이러한 특징은 XML 스키마에서 주석문 <annotation>의 <appinfo>를 사용해 application에 정보를 제공할 수 있도록 한 것과 비슷한 가치를 추구하는 것이다.[6]

주석을 뜻하는 단어 Remark에 착안하여 가독성을 높이는 기능적 주석이라는 상징적 의미로 리마커블을 본 논문의 모델과 처리기의 수식어로 사용한다.

2.1.3 Code Beside

본 논문에서는 ASP.NET에서 유래한 Code Beside라는 용어를 Code Separation과 동의어로 간주하여 사용하고 있으며[7] 이러한 개념을 ASP.NET 이외의 플랫폼에도 적용할 수 있도록 지원한다는 플랫폼 독립적 특징과 언어 중립적 기능성을 표현하고자 리마커블 코드 분리라는 명칭으로 확장하여 사용한다.

2.1.4 리마커블 처리기의 특수태그

alpha태그, omega태그는 디자인파일과 로직파일 각각에 대응 정의되고 cramp태그, bracket태그는 로직파일에 정의되어 최종 웹페이지 생성에 관여한다.

<표 4> 리마커블 처리기의 특수태그

기호	특수태그의 명칭
<!--a[?]-->	alpha태그.(코드블록의 시작. ?는 코드블록의 *keyword)
<!--o-->	omega태그.(코드블록의 끝)
<<ao>>	cramp태그.(디자인파일에서 사용자 컨트롤 속성값 추출)
<[ao]>	bracket태그.(디자인파일 **keyText를 로직파일에 삽입)

*keyword : 해당 코드블록의 기능을 잘 설명하는 이름을 부여.
**keyText : keyword로 참조하는 해당 코드블록의 내용.

2.2 웹페이지 모듈화 과정

2.2.1 역공학 대상 선정

디자인 측면이나 로직 측면에서 향후에 반복적으로 재

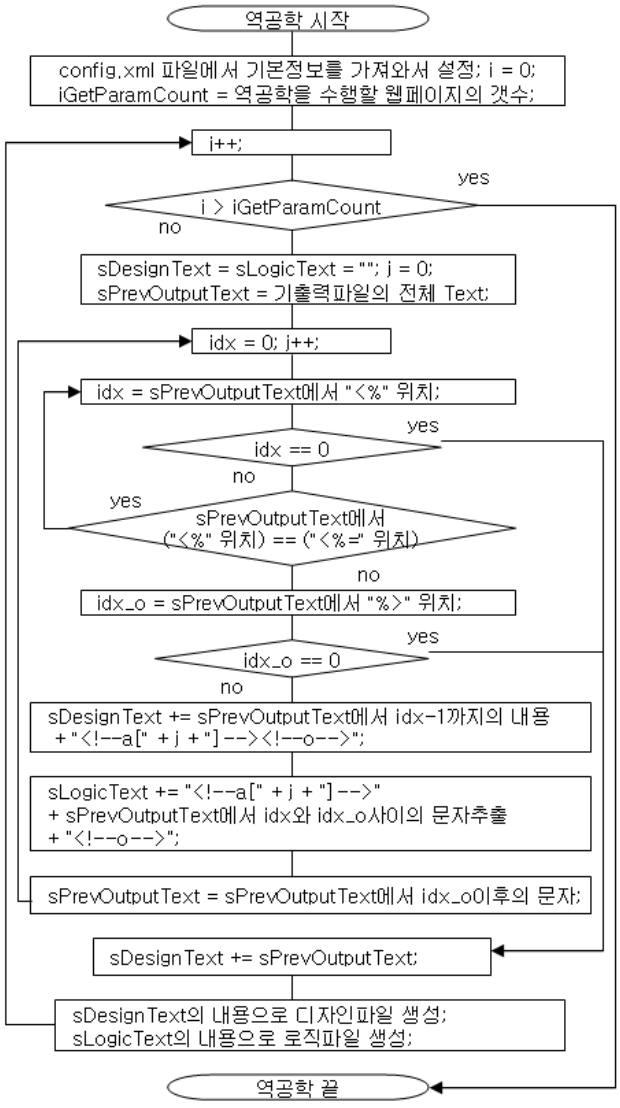
사용하게 될 웹페이지를 선정한다.

2.2.2 역공학 분리 모델 - 스크립트릿 분리하기

역공학 대상으로 선정한 웹페이지를 역공학기에 입력하여 역공학을 수행한다. 그 결과 역공학 분리 모델의 디자인파일과 로직파일을 출력으로 얻는다.

```
<?xml version="1.0"?>
<remarkable_reverse_engineer>
  <design_file_ext>html</design_file_ext>
  <logic_file_ext>cod</logic_file_ext>
  <output_file_ext>jsp</output_file_ext>
  <scriptlet_begin>&lt;%</scriptlet_begin>
  <scriptlet_end>%&gt;</scriptlet_end>
  <expression_begin>&lt;%=</expression_begin>
</remarkable_reverse_engineer>
```

<표 5> 역공학기 config.xml 파일의 설정값 예제



(그림 4) 역공학기 알고리즘 - 순서도

2.2.3. 제1 분리 모델 - keyword 부여하기

역공학 분리 모델의 디자인파일과 로직파일을 제1 분리 모델로 정제하는 작업이다. 역공학 결과, 숫자로 부여된 keyword를 의미있는 keyword로 수정한다. 클라이언트 로직의 분리도 고려하되 관련 있는 코드블록끼리는 병합한다. 역공학 및 제1 분리 모델에서는 디자인파일에 서버측 표현식이 존재하는 경우 로직파일이 디자인파일에 로직종속성을 갖는다.

2.2.4. 제2 분리 모델 - 사용자 컨트롤 만들기

제1 분리 모델의 디자인파일과 로직파일을 제2 분리 모델로 정제하는 작업이다. 제1 분리 모델의 로직종속성을 해소하기 위해 해당 코드블록을 사용자 컨트롤로 만든다. 제2 분리 모델에서는 로직파일에 사용자 컨트롤이 존재하므로 디자인파일이 로직파일에 디자인종속성을 갖는다.

2.2.5. 제3 분리 모델 - 식별자기반 완전 분리

제2 분리 모델의 디자인파일과 로직파일을 제3 분리 모델로 정제하는 작업이다. 제2 분리 모델의 디자인종속성을 해소하기 위해 해당 코드블록을 로직치환용 식별자로 연결하는 작업을 실시한다. 제3 분리 모델은 HTML 요소와 스크립팅 요소의 분리라는 관점에서 완전 분리를 실현한 모델이다. 그러나 로직치환용 식별자가 디자인파일을 참조하므로 로직파일이 디자인파일에 식별자종속성을 갖는다.

2.2.6. 순수 분리 모델 - keyword기반 완전 분리

순수 분리 모델은 디자인파일의 내용은 건드리지 않고 로직파일만 재사용하는 방법이다. 전처리기는 초기 디자인파일에 alpha태그가 정의되어 있지 않으면 순수 분리 모델로 동작하며 디자인파일에서 로직파일 keyword와 일치하는 내용을 찾아 그 자체를 로직 요소로 치환한다. 순수 분리 모델은 디자인파일을 순수하게 유지한 상태에서 완전 분리를 실현한 모델이다. 그러나 로직파일 keyword가 디자인파일을 참조하므로 로직파일이 디자인파일에 keyword종속성을 갖는다. 순수 분리 모델에서도 cramp태그와 bracket태그를 로직파일에 정의하여 사용할 수 있다.

2.2.7. 각 분리 모델의 포함관계

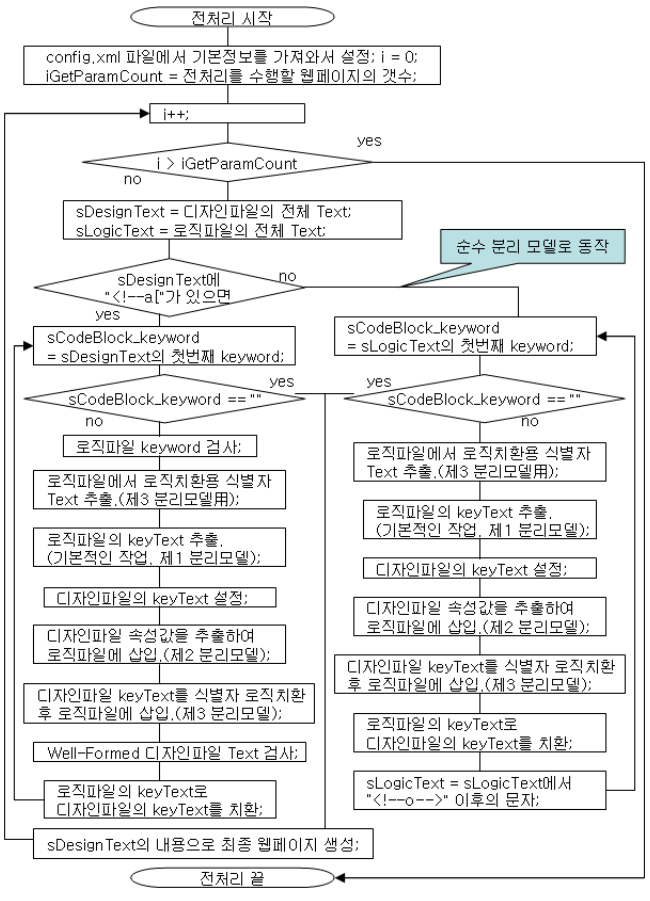
역공학 ⊂ 제1 ⊂ 제2 ⊂ 제3 ⊂ 순수

2.3 최종 웹페이지 생성

지금까지 살펴 본 각 분리 모델의 디자인파일과 로직파일은 전처리기의 입력으로 사용된다. 전처리기는 디자인파일과 로직파일을 하나로 결합한 최종 웹페이지를 출력물로 생성한다.

```
<?xml version="1.0"?>
<remarkable_preprocessor>
  <design_file_ext>html</design_file_ext>
  <logic_file_ext>cod</logic_file_ext>
  <output_file_ext>jsp</output_file_ext>
  <master_design_file>
    <design_file_name>resultList</design_file_name>
    <logic_path_pattern>*/test/resultList*</logic_path_pattern></master_design_file>
  <master_logic_file>
    <design_path_pattern>*/test/resultView_ajax</design_path_pattern><logic_file_name>resultView_ajax</logic_file_name>
    <design_path_pattern>*/test/resultView*</design_path_pattern><logic_file_name>resultView_master_logic</logic_file_name>
  </master_logic_file>
</remarkable_preprocessor>
```

<표 6> 전처리기 config.xml 파일의 설정값 예제

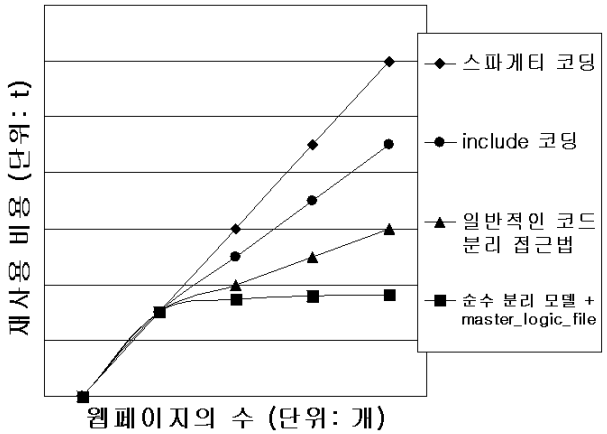


(그림 5) 전처리기 알고리즘 - 순서도

2.4 타당성 검토

웹페이지 재사용 측면에서 스파게티 코딩이나 include 코딩은 웹페이지의 수가 증가할수록 웹페이지를 분석하고 Copy & Paste 하기 위한 비용이 리마커블 모델에 비해 상대적으로 더 많아진다.

설령 include 코딩으로 모듈화를 하더라도 언어 특정적인 인터페이스를 위한 비용이 증가하게 된다.



(그림 6) 웹페이지의 수와 재사용 비용간의 관계

따라서 include 코딩을 보완하기 위한 몇몇 코드 분리 접근법이 등장하였는데 본 논문에서 제안하는 리마커블 모델은 주석기반의 문서화로 웹페이지의 가독성을 향상시킨다는 특징이 있다. 그리고 리마커블 처리기 기반의 웹페이지 모듈화로 언어 중립적인 인터페이스를 정의하여 웹디자이너와 웹프로그래머가 자신의 영역에 더욱 집중할 수 있도록 하였다.

특히 요구사항에 따라 재사용성을 극대화 할 수 있는 master_design_file과 master_logic_file 기능이 전처리기에 포함되어 있고 이러한 기능과 함께 디자인파일을 다른 플랫폼의 로직파일과 연결할 수 있도록 한 것은 관련 연구와 차별화되는 개념이다.

이와 같이 리마커블 모델은 웹페이지의 재사용성을 향상시켜 유지보수 비용을 절감하고 개발 생산성과 신뢰성을 높이고자 하는 것이므로 소프트웨어 품질에도 긍정적으로 기여할 수 있다.

3. 결론 및 향후 연구과제

프리젠테이션 로직을 구성하는 웹페이지에는 디자인 요소와 로직 요소가 혼합되어 있으므로 스파게티 코딩에 의해 웹디자이너와 웹프로그래머간의 작업의 경계가 불분명해지고 디자인과 로직의 구성이 정렬되지 못하는 문제점이 발생한다.

본 논문에서는 이러한 문제점을 해결하고 웹페이지의 가독성과 재사용성을 향상시키기 위한 리마커블 코드 분리 모델을 제안하였다. 이를 통해 웹페이지 수준에서 HTML 요소와 스크립팅 요소를 완전하게 분리할 수 있음을 표현하였고 관련 연구와 차별화되는 플랫폼 독립적이고 언어 중립적인 웹페이지 모듈화 과정을 소개하였다.

리마커블 모델을 실무에 적용하기 위해서는 업무의 재사용성을 검토하여 해당 요구사항이 모델 적합성을 갖는지 여부를 판단해야 한다. 또한 리마커블 모델을 웹 기반의 여러 업무 영역에 적용해 봄으로써 생산성의 향상이 실질적으로 나타나는지를 객관적으로 평가해 보는 폭넓은 연구의 필요성이 있다.

참고문헌

[1] 이상우, "JSP를 이용한 EJB기반의 웹 어플리케이션 구축", *삼양출판사*, 2001/11

[2] "Separation of Business Logic from Presentation Logic in Web Applications", 2003/7 paragoncorporation.com/ArticleDetail.aspx?ArticleID=21

[3] George Shepherd, "고급 ASP.NET 서버 측 컨트롤", *MSDN Magazine*, 2001/1 www.microsoft.com/Korea/MSDN/MSDNMAG/ISSUES/2001/visualprog0101

[4] 김태영, "웹 애플리케이션 개발의 진화, ASP.NET 2.0", *마이크로소프트웨어*, 2004/12

[5] "Smarty : Template Engine", http://smarty.php.net/

[6] 신민철, "기초에서 실무까지 XML 웹 서비스", *프리렉(이한디지털리)*, 2003/10

[7] Devin Rader, "Visual Studio 2005 Technical Articles - Where Did My Icons Go?", *MSDN Library*, 2005/5 msdn2.microsoft.com/en-us/library/ms379619(VS.80).aspx www.microsoft.com/korea/msdn/library/ko-kr/dnvs05/html/vs2005icons.aspx