웹의 표현 계층 지원 컴포넌트 설계 및 구현

○
이수일*, 권기현*, 이형봉**, 정연철***
* 삼척대학교 정보통신공학과
** 강릉대학교 컴퓨터공학과
*** 호남대학교 컴퓨터게임학과
e-mail: kweon@samcheok.ac.kr

Presentation Layer Component of Web Application Systems with Server Side Java Technology

Su-Il Lee*, Ki-Hyeon Kwon*, Hyung-Bong Lee**, Yun-Chul Jung***
* Dept of Info. & Comm. Engineering, Samcheok Natl. Univ.
** Dept of Info. & Computer Engineering, Kangnung Natl. Univ.
*** Dept of Info. & Computer Game Engineering, Hoham Univ.

요 약

웹 애플리케이션 개발에 있어 웹 디자이너와 소프트웨어 개발자의 역할을 분리하는 것은 소프트웨어 작업 능률을 높이고 생산성을 증대시키기 위해 요구되는 사항이며 웹 디자이너와 소프트웨어 개발자의 모듈의 응집도(cohesion)를 높이고 결합도(coupling)를 낮추기 위해 매우 중요하다. 본 논문에서는 기존에 JSP 기반에서 사용되었던 기법들을 살펴보고 JSP(Java Server Page) 작성 시 페이지 디자이너와 소프트웨어 개발자의 역할을 효율적으로 분리하기 위해 커스텀(custom) 태그를 사용하여 HTML 코드와 Java 로직을 분리하여 개선된 개발환경을 제공해 주는 COHALS(Component Of Html And Logic Separation) 컴포넌트를 제시한다.

1. 서론

소프트웨어 개발 시 중요한 고려사항 중의 하나는 각 모듈 간 응집도(cohesion)를 높이고 결합도(coupling)를 낮추는 데 있다. 특히, 웹 애플리케이션 개발 시 프리젠테이션과 로직을 분리하여 페이지 디자이너와 소프트웨어 프로그래머의 역할을 분리하는 방법에 대한 연구가 많이 진행되어 왔다[1-4]. 그러나 아직 완전한 형태의 역할 분리가 이루어지지 못하고 있으며 프로그래머가 페이지 개발에 개입하거나 페이지 디자이너에게 코드를 이해하는 능력을 요구하고 있는 경우가 많다. 최근에는 웹 애플리케이션을 개발하는데 있어 ASP, JSP, PHP와 같은 서버 측 스크립트 언어가 주류를 이루고 있으며 웹 개발에 있어 스크립트 언어를 사용하면 접근이 용이하다는 장점이 있으나, 디자인과 코드를 섞어서 사용하면서 여러 가지 문제점들이 발생한다.

주로 발생하는 문제점을 살펴보면 다음과 같다.

- 프로그래머와 디자이너가 하나의 파일을 사용하여 작업하기 때문에 동시에 작업할 수 없다.
- 프로그래머가 페이지 개발에 개입하거나 페이지 디자이너에게 코드를 이해하는 능력을 요구하고 있는 경우가 많다.
- 하나의 파일을 공유하는데 있어 버전 관리가 쉽지 않다.
- 프로그래머가 디자인을 디자이너가 코드를 실수로 변경할 수 있다.

이러한 문제점 때문에 디자인과 로직을 분리하려는 시도는 자연스러운 것이라고 할 수 있으며 이러한 연구에는 기존에 사용되어왔던 smalltalk의 MVC(Model View Controller) 모델[9], IBM의 MVP(Model View Precentor) 모델[3]과 아울러 오픈 소스 자카르타 프로젝트인 Velocity[4] 및 JSF(Java Server Faces)[7] 등이 있다.

본 논문에서는 JSP(Java Server Page) 작성 시 페

이지 디자이너와 소프트웨어 프로그래머의 역할을 효율적으로 분리하기 위해 HTML 코드와 자바 로직을 분리하여, 개선된 개발환경을 제공해 주는 컴포넌트를 제시한다.

2. 웹 클라이언트와 서버 상호작용

웹 클라이언트와 서버간의 상호작용 형태는 다음과 같이 구분할 수 있다.

2.1 정적인 문서의 처리 흐름

첫 번째 경우는 브라우저가 HTML 파일이나 이미지 파일 같은 정적 문서를 요청하는 경우이다. 이 경우 웹 서버는 클라이언트 요청을 처리하는데 있어 웹 애플리케이션 서버나 서블릿 컨테이너와 상호작용 할 필요가 없다.

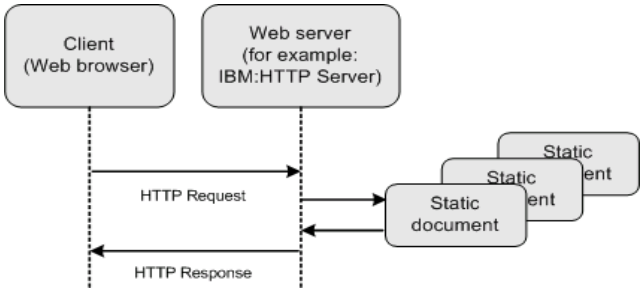

그림 1. 정적 문서의 처리 흐름

2.2 Servlet 경우의 처리 흐름

두 번째 경우는 브라우저가 자바 서블릿을 포함하는 웹 리소스를 요청한 경우이다. 자바 서블릿은 웹 서버가 자바 프로그래밍 언어를 사용하여 서버 상에서 태스크를 수행하도록 한다. 서블릿들은 효율적이어서 CGI와 서버 측 JavaScript 같은 오래된 기술보다 메모리와 프로세싱 파워를 덜 사용한다.

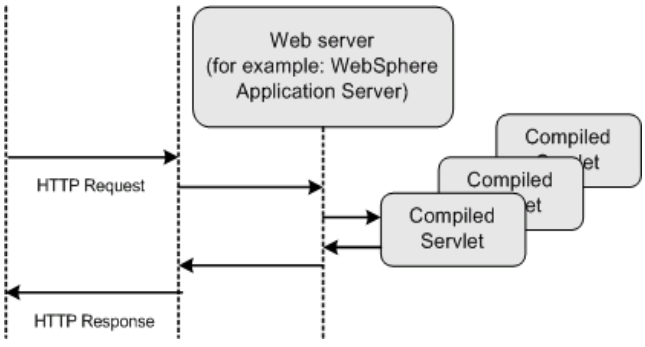

그림 2. 서블릿의 처리 흐름

2.3 JSP 경우의 처리 흐름

세 번째 경우는 브라우저가 JSP 페이지를 포함하는 웹 페이지를 요청하는 경우이다. JSP 페이지는 정보를 디스플레이 하는 작업을 쉽게 하고 동적 콘텐트를 정적 페이지와 분리하는 것을 돕는다. 웹 페이지 디자이너는 HTML 라이브러리에 있는 다른 태그들처럼 JSP 태그를 사용한다. JSP 프로그래머는 JSP 프로그래밍 스펙에 따라 태그를 구현한다.

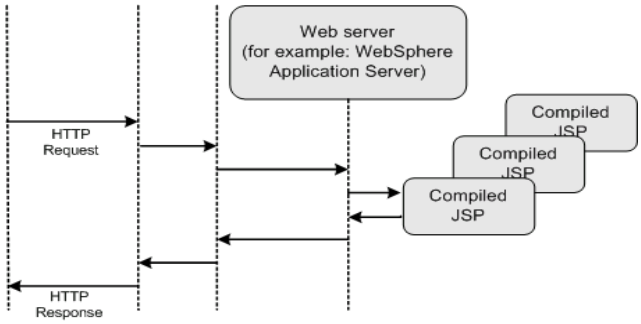

그림 3. JSP의 처리 흐름

웹 컨테이너의 관점에서 볼 때 JSP 페이지는 자바 서블릿과 밀접하게 관련되어 있다. 텍스트 기반 JSP 페이지는 웹 컨테이너에 의해 자바 구현으로 변환된다. 웹 컨테이너는 자바 구현을 찾아 자바 서블릿 같은 구현을 처리하고 코드를 구동하여 프로세싱 결과를 클라이언트에 리턴한다. 많은 레이어와 재지향이 있는 듯이 보이지만 디스패칭은 빠르고 사용자에게도 투명하다. 서블릿과 마찬가지로 자주 요청되는 JSP 페이지는 서버 메모리에 캐싱 된다.

2.4 태그라이브러리와 커스텀태그

JSP는 JSTL(Java Standard Template Library) 같은 표준 라이브러리와 일명 커스텀(custom) 태그라고 하는 스스로 작성한 라이브러리도 사용할 수 있다. 일반적으로 커스텀 태그는 특정 문제 도메인을 위해 사용된다.

3. JSP와 COHALS의 처리 구조 비교

3.1 JSP의 처리 구조

그림 4는 JSP의 처리 구조이다. 이 구조는 JSP 내에 Logic이 포함되어 있어 구현이 복잡하며 JSP 상태에서의 디자인 수정이 어렵다. 또한, Logic 변경 시 JSP 유지보수가 어려운 단점이 있다.

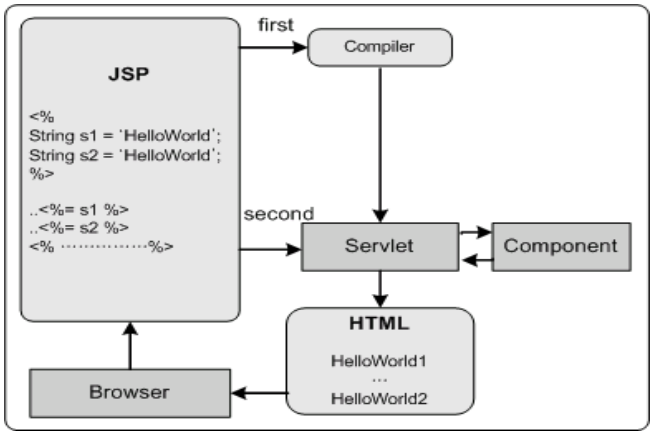

그림 4. JSP의 구조

3.2 COHALS의 처리구조

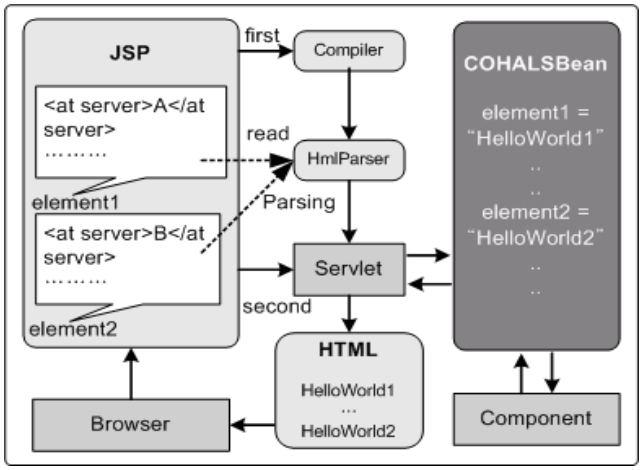

그림 5. COHALS의 구조

그림 5는 본 논문에서 제안한 COHALS(Component Of Html And Logic Separation)의 처리 구조이다. 이 구조는 COHALS 태그를 이용해서 HTML과 로직을 분리할 수 있도록 작성한 컴포넌트이다. 기존의 태그 라이브러리는 이미 생성할 코드를 포함하고 있으나 COHALS는 HTML을 동적으로 읽어 들여 HTML 코드를 생성하는 방법을 사용한다. 마치 ASP.NET의 처리 방식(Code Behind 방식)과 유사한 방법으로 볼 수 있다.

COHALS의 장점은 다음과 같다.

- COHALS 태그를 삽입한 JSP는 일반 HTML과 동일하게 작동하여 디자이너에 의한 디자인 수정작업이 용이하다.
- JSP 디자인 변경 시 개발자가 JSP를 재작성할 필요가 없다.
- 개발자는 Business Logic에만 집중할 수 있다.
- HTML과 Logic이 분리되어 프로젝트 관리가 용이하다.
- 개발기간 단축 및 비용 절감 효과가 있다.

4. 프로토타입 구현

4.1 COHALS API

각 COHALS API와 기능은 다음과 같다.

- com.mod.cohals.fx.TagComponentAdapter : ServerEvent 객체를 이용하여 session처리 및 page view 처리 메소드 제공
- com.mod.cohals.fx.ServerEvent : JSP의 정보(request,response,COHALS 태그 정보 등)를 얻기 위한 메소드 제공
- com.html.element.HtmlTag : JSP의 HTML 태그 중 Table 태그 이외의 태그를 핸들링하기 위한 메소드 제공한다.
- com.html.element.Table : JSP의 HTML 태그 중 Table 태그를 핸들링하기 위한 메소드 제공
- com.html.element.TableCell : JSP의 HTML 태그 중 Table 태그의 <TD> 태그를 핸들링하기 위한 메소드 제공
- com.html.element.TableRow : JSP의 HTML 태그 중 Table 태그의 <TR> 태그를 핸들링하기 위한 메소드 제공

4.2 COHALS 태그와 빈(Bean) 클래스

COHALS 태그는 JSP에서 Logic을 분리시키는 태그로 JSP와 1:1 맵핑되어 있는 클래스를 COHALS 태그로 선언하고 Logic 수행 후 결과 값이 출력되길 원하는 곳에 COHALS 태그를 선언하여 결과 값을 표시하게 된다.

정의한 COHALS 태그의 prefix는 at이고 태그는 server이다. 태그 핸들러의 scope는 BEAN_PAGE, BEAN_SESSION, BEAN_APPLICATION으로 정의되어 있으며 각각 페이지 스코프, 세션(브라우저) 스코프, 애플리케이션 스코프를 가진다. 로직은 HelloWorld 클래스에 의해 처리되며 결과는 ref 값인 a1에 의해 구해져 출력된다.

Ex.) JSP Code using COHALS Tag :

```
<%@ page contentType="text/html; charset=euc-kr"%>
<%@ taglib uri="RunAtServer.tld" prefix="at" %>
<at:server scope="BEAN_PAGE" className="HelloWorld" />
<html>
<body>
<at:server ref="a1">abc</at:server>
</body>
</html>
```

COHALS BEAN class는 JSP에서 COHALS 태그로 선언된 부분의 Logic을 처리할 Java class로 JSP와 1:1로 매핑되는 클래스이다. COHALS Library의 TagComponentAdapter를 상속 받아서 구현한다.

JSP에서 <at:server ref="a1">abc</at:server>와 대응되는 element객체가 ServerEvent로 전달되어 처리가 된다.

ServerEvent 객체에는 Servlet 관련 Session, Request, Response 정보 등을 가져올 수 있는 메소드와 현재 COHALS 태그가 위치한 곳에 데이터를 보여줄 수 있는 메소드들이 제공된다.

Bean class 내용에서 tag_handler 메소드가 태그 핸들러의 처리 내용이 되며 ref의 값이 “a1”인 경우 "Hello World !!!!!" 문자열을 출력하도록 하고 있다. ref의 값이 “a1”이 아닌 경우에는 TagComponentAdapter에 의해 BODY 내용이 그대로 출력되어 "abc"가 출력된다. 또한, 태그의 몸체에 테이블이 있는 경우에는 html 태그 핸들러에 의해 테이블 내용이 처리된다.

Ex.) HelloWorld.java

```
public class HelloWorld extends TagComponentAdapter
{
  public boolean pageInit(ServerEvent e){
        return true;
  }
  public void tag_handler(String ref, ServerEvent event){
        if(ref.equals("a1"))
          e.setContent("Hello World !!!!!");
  }
}
```

5. 결론 및 향후 연구방향

본 논문에서는 JSP(Java Server Page) 작성 시 페이지 디자이너와 소프트웨어 프로그래머의 역할을 효율적으로 분리하기 위해 커스팀(custom) 태그를 사용하여 HTML 코드와 Java 로직을 분리하여 개선된 개발환경을 제공해 주는 COHALS(Component Of Html And Logic Separation) 컴포넌트를 개발하였다.

COHALS 태그를 사용하면 JSP 디자인 변경 시 개발자가 JSP를 재작성할 필요가 없으며 개발자는 비즈니스 로직에만 집중할 수 있다. 따라서 HTML과 로직이 분리되어 프로젝트 관리가 용이하고 개발기간 단축 및 비용 절감 효과가 있다. 향후 연구로 프레젠테이션 부분과 서버 양측에 템플릿 언어 적용에 대한 연구가 필요하다.

참고문헌

[1] A. Saimi, T. Syomuram H. Suganuma, I. Ishida, "Presentation layer framework of Web application systems with server-side Java technology," Computer Software and Applications Conference, 2000. COMPSAC 2000. The 24th Annual International, pp.473-478, Oct. 2000.

[2] Yu Ping, K. Kontogiannis, "Transforming Legacy Web Applications to the MVC Architecture," Software Technology and Engineering Practice, 2003. Eleventh Annual International Workshop on, pp.133-142, Sept. 2003.

[3] Mike Potel, "MVP: Model-Viewer-Presenter," Available at http://www-128.ibm.com/developer-works/java/library/j-mvp.html. 2000.

[4] “Velocity,” Available at http://jakarta.apach-e.org/velocity/index.html. 2003.

[5] M. Jacyntho, D. Schwabe, G. Rossi, “A Software Architecture for Structuring Complex Web Applications,” In International World Wide Web Conference(www2002), 2002.

[6] K. Iijima, J. Ivins, “An Alternate Three-Tiered Architecture for Improving Interoperability for Software Components,” In International World Wide Web Conference(www2003), 2003.

[7] By Qusay H. Mahmoud, "Developing Web Applications with JavaServer Faces," Available at http://java.sun.com/developer/technical-Articles/GUI/JavaServerFaces/. 2004.

[8] S. H. Cheon, G. H. Kweon, H. J. Choi, “Developing a Automatic Components Creating System in Distributed Environment,” Korea Digital Context, Vol.2, 2001

[9] Steve Burbeck, “Application Programming in S mallTalk-80 : How to use Model View Controller(MVC),” Available at http://st-www.cs.uiuc.edu/users/march/st-docs/mv.html. 1992.

[10] Chung, S., Yun-Sik Lee, "Modeling Web applica-tions using Java and XML related technologies," Proceedings of the 36th Annual Hawaii International Conference on System Science, Jan., 2003.