웹 프로젝트에서 디자인과 프로그램의 분리를 위한 ProDesigner 시스템 설계 및 구현

Design and implemention of ProDesigner System to separate Design from Program on Web Project

정 강 용*
Gang-Yong Jung

요 약

소프트웨어 공학 측면에서 웹 기반 프로젝트는 상당히 비효율적인 분야이다. 웹 기반 시스템을 개발하기 위해서는 사용자 인터페이스 디자이너가 먼저 화면 레이아웃을 설계한 후 그 화면의 레이아웃 소스를 가지고 다시 비즈니스 로직 개발자가 코딩을 하여 프로젝트를 완성한다. 유지 보수를 하는 경우에도 이와 비슷한 형태로 작업이 진행된다. 이러한 웹 기반의 소프트웨어 개발 방법은 JSP의 태그 라이브러리와 같은 다양한 형태로 제안되었으나 실제로 적용하는데 여러 가지 문제점이 있으며 기존의 C/S 기반의 소프트웨어 개발에 비해 생산성 측면에서도 매우 좋지 않다. 본 논문에서 제안한 웹 폼 시스템은 기존에 제안된 방법들이 해결하지 못한 사용자 인터페이스 디자인과 비즈니스 로직을 분리하여 웹 기반 소프트웨어의 개발 생산성을 향상시킬 수 있는 방안을 제시하였다.

Abstract

The web-based project is seriously not efficient area at software engineering. In order to develop a web-based system, UI(User Interface) designers usually first design display layout and then business logic developers bring to complete the coding of the display layout source. In the case of maintenance, it goes through the same process. This kinds of web-based software development method were proposed by the various methods same as the tag library of JSP. But there are many problems to apply them and they are low productive comparing to the C/S based software development method. WFS(Web Form System), which is suggested on this thesis, separates UI design from business logic on a web project and offers the better environment to develop web based software. Eventually WFS will improve the productivity to develop web based software.

Keyword : web design, business logic, web form system, template, web UI

Ⅰ. 서 론

1990년대 중반 인터넷의 대중화는 정보의 공유와 배포라는 측면에서 혁명적인 수단이 된 반면에 소프트웨어 개발 측면에서는 퇴보하는 결과를 가져왔다. 절차적 언어인 HTML(Hyper Text Mark-up Language)을 기반으로 하는 웹 기반의 개발 환경에서는 4세대 언어에서 이미 해결한 사용자 인터페이스(User Interface)와 비즈니스 로직을 구분하는 효과적인 방법을 찾지 못하여 사용자 인터페이스 디자인과 비즈니스 로직의 동시 개발이 어려우며 웹 기반의 소프트웨어 개발에 장애가 되고 있다. 따라서 웹 기반의 소프트웨어를 개발하는 경우 사용자 인터페이스 디자이너와 비즈니스 로직 개발자간의 효율적인 의사소통과 풍부한 협업 개발 능력이 필요하다. 이는 웹 기반의 소프트웨어가 체계적이고 효율적인 개발 방법을 이용해 진행되기보다는 경험에 의한 개발 방법에 의존한다고 할 수 있다. 이러한 문제점은 웹 기반의 소프트웨어를 개발하는데 있어서 비효율성이 증가하게 되는 요인이 되고 있다[5,9].

* 정 회 원 : 순천제일대학 컴퓨터과학과
jung740@hanmail.netb(제 1 저자)

웹 기반 프로젝트와 4GL을 이용한 프로젝트를 소프트웨어 공학적으로 분석하면 웹 기반 프로젝트의 개발 생산성이 훨씬 낮다. 웹 기반 프로젝트의 개발 생산성이 저조한 이유는 개발 방법 자체가 복잡하며, 사용자 인터페이스와 비즈니스 로직을 분리할 수 있는 표준화된 방법이 존재하지 않기 때문이다.

본 논문에서는 웹 환경에서 운영되는 정보 시스템의 개발 과정에서 발생하는 사용자 인터페이스와 비즈니스 로직을 효과적으로 분리할 수 있는 웹 폼 시스템을 설계, 구현하였으며 논문의 구성은 다음과 같다.

2장에서는 웹 개발 프로젝트의 특성을 살펴보고, 사용자 인터페이스와 비즈니스 로직을 분리하기 위하여 제안된 여러 가지 방안에 대하여 소개한다. 3장에서는 웹 개발 프로젝트의 특성을 파악하고 디자인과 비즈니스 로직을 분리하는 웹 폼 시스템을 구현하기 위한 필요성과 방법에 대하여 알아본다. 4장에서는 웹 폼 시스템을 설계하고 구현된 예를 살펴보고, 마지막으로 5장에서는 결론과 향후 발전 방향을 제시하였다.

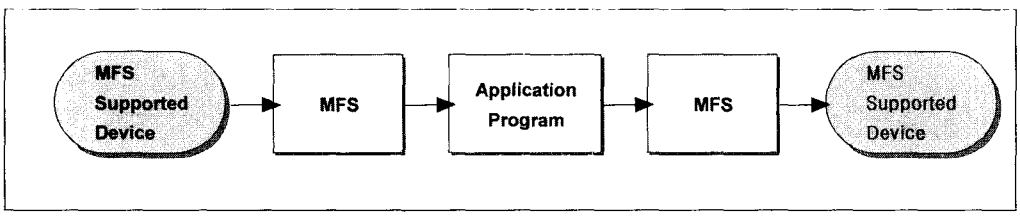
Ⅱ. 관련 연구

사용자 인터페이스와 비즈니스 로직을 분리하기 위한 방법은 CUI(Character User Interface) 기반의 메인 프레임 환경에서부터 시작되었다. 메인 프레임 환경에서 제안되었던 기술은 개발 기간을 단축하기 위하여 개발되었으며, C/S기반의 개발 환경에서는 표준화된 사용자 인터페이스 컴포넌트를 구현하여 사용자 인터페이스 디자이너의 역할을 최소화시켰다. 웹 기반의 환경에서는 사용자 인터페이스 디자이너의 역할이 부각되면서 다음과 같은 다양한 기법들이 제안되었다. 그러나 문제점들을 근본적으로 해결하지는 못하고 있다[1].

2.1 레거시 시스템에서의 분리 방법

IBM 3270과 같은 메인 프레임 기반의 텍스트 환경에서도 사용자 인터페이스와 비즈니스 로직을 분리하기 위한 노력은 있었다. 유능한 프로그래머는 모듈화나 간단한 사용자 라이브러리를 만들어 사용하였으며 점점 상용화된 개발 도구들에 의해서 보완되었다. IBM 메인프레임의 CUI기반의 사용자 인터페이스 환경에서 Message Format Services라는 시스템을 이용하여 화면의 입출력 부분을 별도의 파일로 작성하여 컴파일하고 코볼이나 PL/I으로 제작된 프로그램은 API(Application Programming Interface)를 이용하여 화면 입출력을 용이하게 설계, 개발할 수 있도록 하였다.

메인프레임 환경에서 클라이언트/서버 환경으로 변화되면서 클라이언트의 향상된 그래픽 성능을 이용한 GUI 기반의 통합 개발 도구들이 많이 등장하였다. Inprise사의 Delphi, Sybase사의 Powerbuilder, Microsoft사의 Visual Studio 등은 GUI 환경에서 화면 디자인을 쉽게 하여 프로그래머는 사용자 인터페이스 디자인보다는 프로그래밍에 전념할 수 있는 환경을 제공하였다. 그림 1은 IBM IMS 환경의 MFS를 이용한 메시지 포맷팅 과정을 보여주고 있다[9].



〈그림 1〉 IBM IMS환경의 MFS를 이용한 메시지 포맷팅

2.2 웹 환경에서 분리 방법

소프트웨어 개발 환경이 WWW 기반으로 이동되면서 가장 큰 변화는 절차적 마크업 언어인 HTML이 화면 표현을 위하여 사용된다는 것이다. HTML이 사용자 인터페이스를 위한 화면 레이아웃을 설계하는데 사용됨으로서 웹 개발 초기엔 프로그래머가 디자인에 관련된 마크업 언어를 이해하고 코딩할 수 있어야 했으며, 디자이너도 프로그래머가 사용하는 프로그래밍 언어를 알아야 했다. 또한, 4GL 언어와는 다르게 사용자 인터페이스 관련 디자인이 상당히 중요한 부분을 차지하게 되어 기존의 RAD(Rapid Application Development) 툴에서 제공되던 사용자 인터페이스 관련 레이아웃을 사용할 수가 없게 되었다. 이런 문제는 디자이너의 사용자 인터페이스 작업이 완료된 후 다시 프로그래머가 그 디자인을 포함하는 코드 안에 비즈니스 로직을 작성해서 포함시켜야 했으며, 사용자 인터페이스 디자인이 변경되게 되면 다시 디자이너가 변경 작업을 수행한 후 그 변경된 디자인을 포함하는 코드를 가지고 프로그램을 해야 하는 절차상의 문제를 야기하게 되었다.

이런 문제를 해결하기 위하여 다양한 방법들이 제안되었는데 서버 사이드 스크립트 기반의 템플리트 방식과 4GL 방식이 있으며 각각의 특징은 다음과 같다.

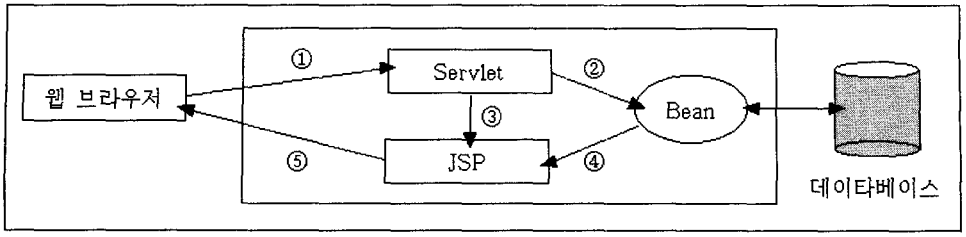
2.2.1 서버 사이트 스크립트 기반의 템플리트 방식

서버 사이드 스크립트 기반의 템플리트 방식은 웹 상에서 응용 프로그램을 개발하는데 필요한 스크립트 언어의 특징을 활용하는 방식이다.

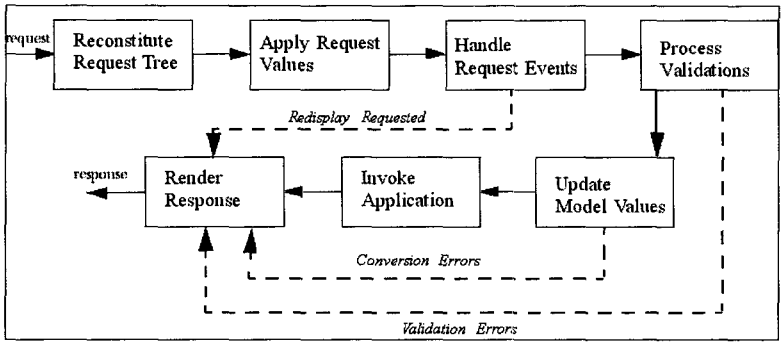
WWW(World Wide Web)에서 애플리케이션을 개발하는 기본 기술인 CGI(Common Gateway Interface) 방식으로 웹 애플리케이션을 개발하는 경우 비즈니스 로직과 사용자 인터페이스 부분이 전혀 분리되지 않고 하나의 비즈니스 로직 개발 코드 안에 포함된다. 이런 문제를 해결하기 위하여 제안된 것이 서버 사이트 스크립트로서 대표적인 서버 사이드 스크립트는 ASP(Active Server Page), JSP(Java Server Page), PHP(Personal Home Page Tools) 등이 있다. 서버 사이드 스크립트는 전체 코드 안에 비즈니스 로직과 사용자 인터페이스 관련 코드 부분을 분리하여 표현할 수 있다. 그러나 별도의 코드로 완벽하게 분리되는 것이 아니고 동일한 코드 안에서 단순하게 비즈니스 로직과 사용자 인터페이스 관련 부분의 영역만을 구분한다. 이 방식은 비즈니스 로직을 개발하는 개발자의 부담을 줄여 주었으나 사용자 인터페이스와 비즈니스 로직을 완전하게 분리하지는 못한다. 이런 문제를 해결하기 위해 ASP에서는 DLL (Dynamic Link Library) 파일, JSP에서는 Java Beans, PHP에서는 Smarty(the compiling template engine for PHP)와 같은 템플리트 기법을 도입하여 비즈니스 로직과 사용자 인터페이스 부분을 분리할 수 있다. 그러나 이러한 방법들은 특정 스크립트 언어에 종속되며, 개발자와 디자이너가 템플리트 기법에 대한 사전 지식이 있어야 하며, 개발자나 디자이너 둘 중에 한 파트의 작업자가 다른 파트의 작업자에 비해 더 많은 역할을 수행하여야 한다. 그림 2는 JSP에서 Beans를 이용한 프로그램의 구조이다. 클라이언트의 웹 브라우저에서 웹 서버로 처리를 요청하면 Servlet이 그 요구를 받아 Bean으로 처리를 위임하고 Bean은 데이터베이스에 접속하여 비즈니스 로직을 처리하고 JSP는 Bean으로부터 그 결과를 받아 웹 브라우저에게 전달한다[2,6,7,8].

2.2.2 4GL 방식

4GL 언어가 애플리케이션 분야에 폭 넓게 사용된 것은 화면 디자인을 하는데 필요한 여러 가지 컴포넌트들을 정형화 시킨 후, 그 컴포넌트에 속성을 부여하여 개발자별로 손쉽게 GUI 기반의 응용 프로그램을 개발할 수 있기 때문이다. 이 방식은 Microsoft에서 제안한 웹 폼, SUN에서 제안



〈그림 2〉 Bean을 이용하는 JSP 프로그램 구조



〈그림 3〉 JSF의 라이프사이클

한 JSF(Java Server Faces) 방식이 있다. 웹 폼이나 JSF는 Visual Basic이나 Delphi 등과 같은 4GL 개발 툴에서 사용하던 방식과 비슷하게 정형화된 사용자 인터페이스 컴포넌트를 활용하여 사용자 인터페이스 디자이너의 역할을 최소화하였다. 웹 기반에서 4GL 방식의 기법을 활용한 방식은 아직은 개발이 진행 중이기 때문에 폭넓게 사용되고 있지 않다. 그림 3은 JSF의 라이프사이클을 보여준다[8,10].

Ⅲ. 디자인과 비즈니스 로직의 분리

3.1 웹 개발 프로젝트의 특성

기존에 사용되어온 메인프레임 환경이나 C/S기반의 개발 환경에서의 정보 시스템 프로젝트가 웹 개발 프로젝트에 적용할 수 없거나, 개발 생산성이 저조한 이유는 웹 프로젝트 자체가 여러 기술들이 복합되어 매우 복잡하며, 사용자 인터페이스와 비즈니스 로직을 분리할 수 있는 표준화된 방법이 존재하지 않고, 다음과 같은 웹 개발 프로젝트의 특수성 때문이다.

첫째, 웹 개발 프로젝트의 기반이 되는 HTTP 프로토콜의 가장 큰 특징은 비연결성이다. 즉, 웹 브라우저가 서버에 연결하고 정보를 요청하면, 서버는 연결된 브라우저에 처리된 정보를 제공하고 연결이 끊어진다는 것이다. 이런 구조에서는 지속적이고 안정적인 상태 지속을 할 수 없기 때문에 쿠키나 세션과 같은 별도의 방법을 이용하여 상태를 지속할 수 있으나 이 방법은 보안 측면이나 안정성 측면에서 보완해야 할 점이 많다[1,2].

둘째, 시스템 구성의 복잡성에 있다. C/S 기반의 4GL 툴로 개발된 소프트웨어의 경우 사용자 컴퓨터의 클라이언트 프로그램과 서버의 데이터베이스만이 작업을 수행하지만 웹 기반의 프로젝트의 경우 클라이언트 프로그램인 웹 브라우저와 서버의 데이터베이스, 그리고 서버의 데이터베이스를 가공하여 사용자의 클라이언트 프로그램에 데이터를 전송하는 웹 애플리케이션이 필요하다. 이런 복잡한 구조의 시스템의 효율적인 구성을

〈표 1〉 기존 방법의 문제점

단 점	내 용
1. 시스템 환경 종속	.Net 플랫폼과 J2EE 등 각기 다른 환경 사용
2. 웹 개발 언어 종속	JSP, ASP, PHP 등 언어에 따라 사용 툴이 상이함
3. 상이한 기술 습득	Beans, Web Form, Smarty 등 상이한 전문 기술 필요
4. 기타 문제점	디자인과 로직 분리만을 위해 Heavy Platform 사용

위해서는 기존의 개발 방법보다 훨씬 더 많은 노력을 요구하게 된다.

셋째, 기존의 4GL 툴은 표준화된 사용자 인터페이스 컴포넌트를 구현하여 별도의 사용자 인터페이스 디자이너가 없어도 애플리케이션을 구현할 수 있었으나 웹 기반 애플리케이션은 표준화된 사용자 인터페이스 컴포넌트가 없기 때문에 개발 과정에서 사용자 인터페이스 디자이너의 지원이 필수적이다. 사용자 인터페이스 디자인과 비즈니스 로직의 동시 개발에 대한 직관적인 개발 방법이 존재하지 않기 때문에 사용자 인터페이스 디자이너의 작업 후 비즈니스 로직 개발자의 작업이 진행되며, 디자이너와 프로그래머가 하나의 소스를 가지고 작업을 하고, 소스에 사용자 인터페이스 디자인과 비즈니스 로직이 혼재되어 있기 때문에 가독성이 저하되고, 수정 과정에서 난이도가 높아지며 소스의 안전성 유지에도 문제가 있다.

웹 기반 애플리케이션의 비연결성에 대한 제약점은 플러그인이라는 기술을 활용하여 극복되었으며 웹 기반 시스템이 가지는 복잡성은 WAS (Web Application Server) 등의 표준화된 개발 방법을 사용하면서 극복할 수 있었다. 그러나 사용자 인터페이스 디자인과 비즈니스 로직의 분리에 대한 연구는 지금도 진행 중이며 아직 완벽한 해결 방법이 없다[5,9].

3.2 디자인과 비즈니스 로직 분리의 필요성

앞장에서 언급한 기존의 방법들에는 공통적인 단점들이 있으며, 그 내용은 다음과 같다. 첫째, 각 벤더들은 각자 기존의 웹 운영과 개발 도구에 종속된 방법들을 제공하기 때문에 다른 웹 프로그래밍 언어나 웹 운영 환경과 호환이 되지 않는다. 둘째, 각 환경마다 기술의 습득이 어려우며 다른 개발 환경 및 운영 환경에 적용하기가 어렵다. 셋째, 기존의 디자인 전문 툴을 모두 활용하기가 어렵다. 넷째, 디자인과 로직의 분리만을 위해서 Heavy Platform을 적용하여야 하므로 도입과 유지보수 비용이 많이 든다. 표 1은 기존 방법의 문제점을 항목별로 분류해 놓은 것이다.

이러한 단점을 해결하기 위하여 본 논문에서는 사용자 인터페이스 디자이너와 비즈니스 로직 개발자를 분리시켜 사용자 인터페이스 디자이너와 비즈니스 로직 개발자와의 의견 충돌이나 소스 코드 버전 관리를 수월하게 할 수 있는 방안을 연구하였다. 즉, 사용자 인터페이스 디자이너가 사용자 인터페이스 관련 디자인이 끝난 후 개발이 진행되는 문제를 해결하여, 동시에 개발을 진행할 수 있도록 함으로써 소프트웨어 생산성을 향상시킬 수 있는 방법론을 제시하였다.

3.3 디자인과 비즈니스 로직의 분리를 위한 기본 조건

사용자 인터페이스와 비즈니스 로직의 분리를 위한 개발 방법론이 갖추어야할 기본적인 기능들은 다음과 같다. 본 논문에서 제안한 웹 폼 시스템은 이러한 기본 조건들을 만족하도록 설계 및

구현되었다.

첫째, 플랫폼에 독립적이어야 한다. 마이크로소프트사의 .Net이나 Sun사의 J2EE와 같은 플랫폼에 종속되지 않고 어느 플랫폼에나 쉽게 적용될 수 있어야 한다. 웹 애플리케이션을 개발하는데 있어서 특정 플랫폼에 종속된다는 것은 웹이 추구하는 기본 방향과 일치하지 않으며, 특정 기술에 종속되는 경우 개발 시스템 자체에 제약이 된다.

둘째, 기존의 디자이너와 프로그래머가 쉽게 사용할 수 있도록 하여야 한다. 즉, 완전히 새로운 디자인 도구를 사용하거나 새로운 언어를 습득하지 않고, 약간의 API와 디자인 규칙만으로 사용할 수 있어야 한다.

셋째, 기존의 웹 개발 프로그래밍 언어에 종속적이지 않고 모든 언어에서 동일한 방법으로 사용할 수 있어야 한다. JAVA 환경에서 개발하던 프로그래머가 PHP 환경에서 새로운 시스템을 개발하려 할 때 JAVA에서 사용하던 방법을 그대로 사용할 수 있어야 한다.

넷째, 단순히 사용자 인터페이스와 비즈니스 로직을 분리하기 위해서 높은 시스템 사양을 요구하거나 시스템에 많은 부하가 발생하여서는 안 된다. 마이크로소프트사의 .NET과 같은 경우 높은 사양의 시스템을 요구하기 때문에 일반적으로 사용되기에는 문제가 있다.

다섯째, 현재 널리 사용되고 있는 보편적인 디자인과 프로그램 도구들을 계속 사용할 수 있어야 한다. 다시 말해서 제안된 방법이 HTML 태그 중에 "<TABLE>"을 사용할 수 없다는 식의 제약 조건을 가져서는 안 된다.

여섯째, 사용자 인터페이스 디자이너와 비즈니스 로직 개발자가 동시에 작업 가능하도록 하여야 한다. Macromedia사의 드림위버는 사용자 인터페이스 디자이너와 개발자가 같은 화면을 통해 개발 작업이 가능하지만 동시에 개발을 진행할 수 없다는 단점은 해결하지 못하였다.

Ⅳ. 웹 폼 시스템의 설계 및 구현

본 논문에서 구현한 웹 폼 시스템은 3장에서 설명한 기존 방법의 문제점들과 분리를 위한 기본 조건들을 토대로 사용자 인터페이스 디자인과 비즈니스 로직을 분리하여 작업이 가능하도록 하였으며, Publish되는 과정에서 각자의 작업 결과를 조합시켜 최종 사용자의 웹 브라우저에 출력되도록 하였다.

웹 폼 시스템은 사용자 인터페이스 디자이너와 비즈니스 로직 개발자가 각자의 개발 툴로 각자가 작업한 소스를 가지고 별도로 작업하고, 최종 사용자가 해당 페이지를 보려고 할 때 웹 폼 시스템이 이것을 처리하여 클라이언트 브라우저에 그 결과를 전달한다. 그림 4는 이러한 과정과 웹 폼 시스템의 구성 요소를 보여준다.

4.1 웹 폼 시스템의 구조

웹 폼 시스템은 그림 4와 같이 폼 관리 모듈, 폼 컴파일러, 폼 Pool, 폼 엔진, 매칭 모듈과 폼 드라이버로 구성되어 있으며 개발 언어는 JDK 1.3을 사용하였다.

4.1.1 폼 관리 모듈

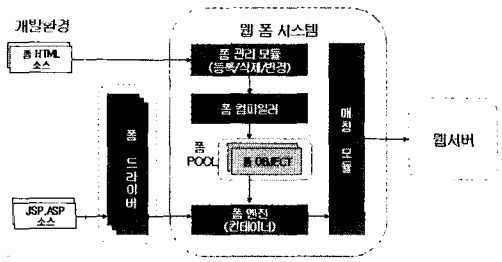
폼 관리 모듈은 사용자 인터페이스 디자이너가 생성한 화면 레이아웃 소스를 관리한다. 즉, 사용자 인터페이스 디자이너는 드림위버, 나모 웹 에디터와 같은 기존의 디자인 도구를 사용하여 화면을 디자인하고 그 결과로 생성되는 화면 소스를 폼 관리 모듈을 이용하여 웹 폼 시스템에 등록할 수 있다. 또한 화면 소스의 관리를 위해 수정 및 삭제할 수 있다. 화면 소스를 웹 폼 시스템의 폼 관리 모듈에 적재함으로서 각각의 화면들을 체계적으로 관리할 수 있으며, 유지 보수 업무를 쉽게 할 수 있다는 장점이 있다.

4.1.2 폼 컴파일러

폼 컴파일러는 폼 관리 모듈을 통하여 등록된 웹 화면 소스를 컴파일하여 폼 OBJECT를 생성한 후 폼 POOL에 등록한다. 최종 사용자에게 요청한 페이지를 전달하는 시점에 화면을 컴파일하여 전송하는 방식이 아니고 미리 폼 엔진이 폼을 컴파일하여 사용자의 요청에 즉각 응답할 수 있도록 하였다. 매칭 모듈에서의 사용자 인터페이스 디자이너가 등록한 화면 소스와 비즈니스 로직 개발자의 데이터를 직접 결합시키려면 사용자가 화면을 요청할 때마다 매번 화면 소스를 파싱하여야 하므로 성능과 효율성에 문제가 있다. 그러므로 폼 컴파일러는 화면 소스를 매칭 모듈이 용이하게 사용할 수 있는 데이터 구조로 변화시켜 폼 OBJECT 형식으로 저장하는 역할을 수행한다.

4.1.3 폼 POOL

폼 POOL은 폼 컴파일러에 의해 생성된 폼 OBJECT들을 저장하고 관리하는 모듈이다. 폼 POOL에 있는 폼 OBJECT의 구조는 표 2와 같으며 폼 OBJECT는 HEADER 부분과 BODY 부분으로 구성되어 있다. HEADER는 폼에 대한 이름과 변경될 항목과 변경되지 않을 항목의 개수 정보가 저장되어 있으며, BODY는 정의되어 있는 FV(FVta Agent)들에 대한 순번 정보, Type 정보, 내용 정보 등의 기본 정보가 포함되어 있다.



〈그림 4〉 웹 폼 시스템의 구조

4.1.4 폼 엔진

폼 엔진은 폼 POOL에 저장되어 있는 폼 OBJECT를 주기억장치로 로드해서 폼 매칭 모듈이 즉시 사용할 수 있도록 한다. 웹 폼 시스템의 성능 향상을 위한 서버 역할을 하는 부분으로 보조기억장치에 있는 폼 OBJECT를 주기억장치에 상주시킴으로써 웹 서버에 의해서 요청되는 처리를 매칭 모듈이 빠르게 처리할 수 있도록 한다. 시스템이 대규모일 경우는 폼 OBJECT들의 수가 많아져 폼 POOL의 사이즈가 커지므로 변형된 LRU(Least Recently Used) 알고리즘을 적용하여 관리한다. 최근에 사용된 폼 OBJECT는 주기억장치에 계속 적재될 수 있도록 하고 가장 오래 전에 사용된 폼 OBJECT들은 주기억장치에서 삭제시킨다. 그러나 사용 빈도수가 작지만 중요한 화면의 경우 관리자가 영구히 상주할 수 있는 기능을 제공한다.

4.1.5 매칭 모듈

매칭 모듈은 사용자 인터페이스 디자이너가 작

〈표 2〉 폼 오브젝트 구성

폼 OBJECT 항목		설 명	비고
HEADER	이름	폼 OBJECT의 이름 (화면 ID)	
	적재 구분	주기억장치에 적재 방법("L":LRU, R:상주)	
	변경 FV 수	화면에서 변경되는 FV 수	
BODY	FV SEQ	화면 FV들의 순번	반복되는 항목
	FV TYPE	FV의 속성 (속성값 : 고정, 변경)	〃
	FV 내용	FV에 표시될 내용	실행시 할당됨

업한 화면 소스를 컴파일한 폼 OBJECT와 비즈니스 로직 개발자가 작성한 웹 프로그램에서 제공되는 가변 데이터를 결합시켜 주는 역할을 수행하며, 웹 폼 시스템의 핵심 기능을 수행한다. 표 2의 폼 OBJECT 구조에서 폼 BODY 부분의 FV 항목들은 실제 데이터와 매핑하는 역할을 수행한다. 항목 TYPE의 속성 값이 '고정'이면 화면에 변경 없이 출력되고 항목 TYPE의 속성 값이 '변경'이면 웹 프로그램으로부터 전달받은 데이터로 변경시켜 화면에 출력한다.

4.1.6 폼 드라이버

웹 폼 시스템은 다양한 플랫폼에서 실행되는 프로그램 환경과 인터페이스를 한다. 단일한 환경의 웹 폼 시스템이 다양한 프로그램 환경에 대응하기 위하여 필요한 것이 폼 드라이버다. 폼 드라이버는 각각의 프로그램 환경에 맞추어 제작되고 단지 웹 폼 시스템과의 인터페이스를 위한 게이트웨이 역할을 한다. 새로운 프로그램 환경이나 변경된 환경에 대한 처리를 위하여 폼 드라이버를 추가 또는 변경만 하면 웹 폼 시스템은 변경하지 않아도 됨으로 웹 폼 시스템과 프로그램 환경과의 독립성을 유지할 수 있다.

4.2 웹 폼 시스템의 구현 및 적용

4.2.1 웹 폼 시스템의 구현 방안

사용자 인터페이스와 비즈니스 로직을 구분하기 위하여 웹 폼 시스템에서 구현되어야 하는 기능은 다음과 같다.

첫째, 사용자 인터페이스를 구현하는데 사용되는 HTML과 비즈니스 로직이 구현되는 프로그램 소스를 분리할 수 있어야 한다. 단순하게 HTML과 프로그램 소스를 따로따로 작업할 수 있도록 영역을 지정하는 것이 아니라 별도로 동시에 개발이 진행될 수 있도록 HTML과 프로그램 소스를 분리할 수 있어야 한다.

둘째, HTML을 그대로 유지해야 한다. Macromedia사의 ColdFusion 서버의 경우 별도의 CFML이라는 마크업 언어를 이용해서 사용자 인터페이스와 비즈니스 로직을 구분할 수 있도록 하였으며 다른 시스템들 또한 별도의 언어를 사용할 수 있어야만 개발이 가능하다는 단점이 있다. 본 논문에서 설계 구현한 웹 폼 시스템은 별도의 언어를 이용하지 않고 순수한 HTML만을 이용한다.

셋째, 사용자 인터페이스 디자이너와 비즈니스 로직 개발자가 동시에 개발이 진행될 수 있도록 필요한 규약을 제정해야 한다. 규약은 직관적이어야 하며 J2EE나 .NET와 같은 복잡한 방식이 아닌 디자이너나 개발자가 손쉽게 사용이 가능해야 한다.

4.2.2 웹 폼 시스템 적용 사례

웹 폼 시스템을 구현하여 실제로 적용하는 개발 과정은 표 3과 같다. 시스템 설계 단계에서 시스템 설계자는 사용자 인터페이스 디자이너와 비즈니스 로직 개발자가 구현 단계에서 인식할 수 있도록 화면 설계 과정에서 화면 레이아웃 부분은 HTML을 그대로 사용하고 데이터가 나타날 부분에 대해서만 웹 폼 시스템에서 제안한 FV (Form Variable, 식별자)를 지정한다. FV는 사용자 인터페이스 디자이너와 비즈니스 로직 개발자가 동시에 업무를 진행할 수 있도록 하는 매개체로서 그림 9와 같이 화면 레이아웃 디자인에 @@1

〈표 3〉 단계별 작업과정

설계 단계		구현 단계 (코딩, 유지 보수)	
설계자	식별자를 포함한 화면 레이아웃 설계	디자이너	HTML 편집기로 화면 디자인과 식별자 삽입
		프로그래머	비즈니스 로직에 의한 결과 값을 API를 통하여 식별자에 할당

과 같이 @@와 숫자로 조합된 것이다. @@는 일반 문서에서는 나타나지 않고 매칭 모듈만이 해석할 수 있는 특수 패턴의 인식자로 작용한다. FV처리를 일반적인 템플릿 엔진에서 하는 것처럼 새로운 문법이나 태그를 추가하는 방식을 취하지 않고 가시적이고 특수한 문자열을 조합을 선택한 것은 다음과 같은 이유이다. 일반적인 웹 디자인 도구를 이용하여 디자인하면서 별도의 조작 없이 FV를 쉽게 삽입하고 확인할 수 있어야 한다. HTML을 변형시키면 웹 에디터가 소스를 인식을 하지 못하거나 소스를 해석하지 못 할 수 있다. 구현 단계에서 사용자 인터페이스 디자이너는 설계된 레이아웃을 참조하여 기존의 HTML 편집 툴을 사용해서 데이터가 표시될 부분에 대해서는 FV를 표기하며, 그와는 별도로 개발자는 개발 툴을 사용해서 FV에 해당하는 비즈니스 로직만 작성하면 된다.

그림 5는 사용자 인터페이스 디자이너가 작업한 게시판 화면으로 디자이너는 설계단계에서 작성된 화면 레이아웃을 웹 디자인 도구를 사용하여 작성하고 비즈니스 로직과 연결되어 가변적으로 대치되어야할 부분에 FV만 삽입시키면 된다.

표 4는 비즈니스 로직 개발자가 그림 5에서 지정한 FV를 대치하기 위한 부분을 코딩한 예제이

〈표 4〉 코딩 예제

```
//=============================================================
// Initialize Form
//=============================================================
   myform.setFileName("/data/devel/project1/myform/freeBoardList.htm");

//=============================================================
// Substitute VF with Data
//=============================================================
// Form Header
//-------------------------------------------------------------
   myform.setListData(0,sNo);                  // 첫번째번호
   myform.setListData(1,sTotal);               // 전체리스트
   myform.setListData(2,sStartPage);     // 시작페이지
   myform.setListData(3,sTotalPage);     // 전체페이지
   myform.setListData(4,"1");                  // 첫페이지
   if ( session.getAttribute("s_id") != null )  // login id
        myform.setListData(5,(String)session.getAttribute("s_id"));
   else
        myform.setListData(5,"guest");

//-------------------------------------------------------------
// Form List
//-------------------------------------------------------------
   for( iListRow=1; iListRow<iMaxCount; iListRow )
   {
        myform.setTableData(iListRow, 0, s_num                );
        myform.setTableData(iListRow, 1, s_name               );
        myform.setTableData(iListRow, 5, email                );
        myform.setTableData(iListRow, 2, s_title + sNewMessage );
        myform.setTableData(iListRow,21, s_num                );
        myform.setTableData(iListRow, 3, "[" + strdate.substring(0,10) + "]" );
        myform.setTableData(iListRow, 4, read_count           );
   }

//=============================================================
// Print Form
//=============================================================
   if (!myform.printForm())
        out.println("prinitForm() Error<p>");
```

〈표 5〉 웹 폼 시스템 함수

함수명	처리 내용
setFileName	HTML 화면 템플릿을 지정한다.
setListData	VF에 값을 전달한다.
setTableData	2차원 배열로된 VF에 값을 전달한다.
printForm	해당 템플릿을 VF와 결합하여 클라이언트로 전송한다.

다. 비즈니스 로직 부분에서 처리해야될 부분은 크게 세 부분이다. 첫 번째 부분은 비즈니스 로직에서 사용하기 위하여 디자이너에 의해서 작성된 사용자 인터페이스 폼의 경로명을 setFileName() 함수로 지정한다. 두 번째 부분은 디자이너가 지정한 VF를 실제 대치되어야 할 내용을 setList Data()과 setTableData()으로 할당시킨다. setList Data()함수는 특정한 VF에 값을 할당만 시키지만 setTableData()란 함수는 그림 5와 같이 게시판에 글 제목과 같이 일련의 VF가 반복적으로 할당되어야 하는 부분에 사용한다. 이렇게 반복되어져야 할 부분은 디자이너가 지정해 주어야 하는 데 전제사항에서 기술한 바와 같이 기존의 HTML 문법을 변경시키지 않고 해결하기 위하여 인터페이스 디자인시 특정한 주석을 삽입하여 웹 폼 시스템에서 인식토록 하였다. 마지막으로 VF에 할당된 값을 인터페이스 디자이너에 의해 작성된 폼과 결합시키는 printForm()함수를 사용한다. 표 5는 웹 폼 시스템에서 사용하는 함수들이다.

V. 결론 및 연구 과제

웹을 기반으로 하는 소프트웨어는 사용자 인터페이스 디자이너의 비중이 증가되면서 비즈니스 로직 개발자와 사용자 인터페이스 디자이너의 개발 경험과 개발자와 디자이너의 원활한 의사소통이 필요조건이 되었다. 이런 문제점을 해결하기 위해서는 웹 기반의 소프트웨어에서 사용자 인터페이스 부분과 비즈니스 로직을 분리하여 서로의 간섭을 최소화시키는 것이 관건이지만 현재까지 제안된 방법들은 여러 가지 문제점들을 가지고 있다.

본 논문에서 구현한 웹 폼 시스템은 기존에 제안되었던 방안들보다 훨씬 쉬우면서도 효과적으로 사용자 인터페이스와 비즈니스 로직을 구분하여 개발할 수 있도록 하는 환경을 제공하고 있다. 웹 폼 시스템은 웹 기반 프로젝트에서의 RAD 툴을 사용하는 것처럼 개발 생산성을 증가시킬 수 있으며, 웹 프로젝트가 개발 완료된 이후 또는 유지보수 과정에서도 안전하고 용이하게 프로그램과 디자인을 변경시킬 수 있도록 하여 유지보수 비용도 절감할 수 있는 효과를 제공한다. 또한 웹 폼 시스템은 웹 서버나 웹 편집 도구에 대해 독립성을 가지고 있다.

앞으로의 연구 과제는 JSP로만 한정되어 있는 웹 프로그래밍 언어를 ASP나 PHP와 같은 다른 웹 프로그래밍 언어로 포팅 작업을 수행하여야 하며, 리스트 형태의 데이터를 효과적으로 처리하기 위한 FV를 개발하는데 있다.

참 고 문 헌

[1] Anders Kristensen, "Template Resolution in XML/HTML," HP Labs Technical paper, 1999.

[2] ASP, Microsoft사, http://www.microsoft.com

[3] Engin K. and Clemens K., "MyXML: An XML based template engine for the generation of flexible web content," Technical Univ. of Vienna, 2001.

[4] ITdata, http://www.itdata.co.kr/column/200212/

tech/tr08.asp
[5] Jessica Burdman, "Collaborative Web Development: Strategies and Best Practices for Web Teams," Addison Wesley Longman, 2000.
[6] Macromedia, http://www.macromedia.com/support/ultradev/building/arrow_aircraft7/
[7] PHP, http://smarthy.php.net/
[8] Sun's Javaserverfaces, http://java.sun.com/j2ee/javaserverfaces/docs/Introduction.html
[9] The history of Computing, http://ei.cs.vt.edu/~history
[10] Web forms, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconintroductiontowebforms.asp

◐ 저 자 소 개 ◑

정 강 용
1991년 Northern Illinois University MIS전공(석사)
2003년 순천대학교 컴퓨터과학과 전산학 박사
1992년~현재 : 순천제일대학 컴퓨터과학과 학과장 부교수
한국인터넷정보학회 학회지 편집위원
관심분야 : 웹프로그래밍, 인터넷정보처리, computer network
E-mail : jung740@hanmail.netbgkim@comp.ssu.ac.kr