J2EE 기반 웹 애플리케이션의 표현계층을 위한 설계 패턴 분류

김송주[0] 이창목 유철중 장옥배
전북대학교 컴퓨터과학과
songju@mail.chonbuk.ac.kr cmlee@cs.chonbuk.ac.kr {okjang, cjyoo}@moak.chonbuk.ac.kr

Classification of Design Pattern for Presentation Tier of Web Application based on J2EE

Song-Ju Kim[0], Chang-Mok Lee, Cheol-Jung Yoo, Oak-Bae Chang
Dept. of Computer Science, Chonbuk National University

요 약

J2EE 플랫폼에 기반 하는 웹 애플리케이션의 수는 최근 들어 급격히 증가하는 추세이다. 이러한 추세에 따라 웹 애플리케이션의 성능 향상을 위한 표준화가 부각되고 있다. 본 논문에서는 이러한 웹 애플리케이션의 표현계층에 대한 설계 패턴을 알아보고 이러한 패턴을 분류하여 표준화시키고자 한다. 이러한 분류는 웹 애플리케이션의 빠른 생성을 위한 표현 계층 자동생성 템플릿 설계의 바탕이 된다.

1. 서 론

J2EE는 썬마이크로시스템즈사가 제안한 다 계층 엔터프라이즈 애플리케이션을 개발하기 위한 플랫폼으로 엔터프라이즈 JavaBean, Servlet, JSP등으로 다 계층에 적용되어지는 여러 기술들을 이용하고 있다[1].

J2EE 기반으로 개발되는 웹 애플리케이션의 구조는 클라이언트 계층, 표현계층, 비즈니스 계층, 통합계층, 리소스 계층으로 이루어져 있다. 개발에 가장 중심이 되는 계층은 표현 계층과 비즈니스 계층이다. 표현 계층은 JSP와 Servlet 기술을 이용하여 구성되어지고 비즈니스 계층은 EJB와 다른 비즈니스 객체들을 이용하여 구성되어진다. 이러한 계층 중심적인 웹 애플리케이션의 구조는 각 계층에 따라 개발자의 역할을 달리하도록 하고 있다. 이는 웹 애플리케이션의 개발에 있어 개발자의 역할이 표현 계층의 개발을 담당하는 웹 컴포넌트 개발자역할과 비즈니스 계층을 담당하는 비즈니스 컴포넌트 개발자역할로 구분되어 있음을 의미한다. 이 중에서 실질적인 비즈니스 로직을 담당하는 비즈니스 컴포넌트 개발자 역할은 전체 애플리케이션 개발 중에 더욱 중요한 위치에 있다. 따라서 개발의 주도는 보통 비즈니스 컴포넌트 개발자가 이끌어나가는데, 비즈니스 컴포넌트 개발자가 자신의 컴포넌트를 점검하고 테스트하기 위해서는 웹 컴포넌트 개발자와 지속적이고 유기적인 상호협력이 필요하다. 웹 컴포넌트를 개발하는 개발자와 항상 협력해야만 한다. 이는 개발인력의 증가와 상호협력에 따른 개발의 비효율성이 나타날 수 있으며, 전체적으로 웹 애플리케이션의 개발 기간을 지연시킬 수 있다. 따라서 본 논문에서는 표현 계층의 웹 컴포넌트를 자동으로 생성해줄 수 있는 템플릿을 설계하기 위한 전 단계로서 표현 계층의 고려사항과 설계 패턴(Design Pattern)을 알아보고 패턴을 분류해보고자 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 표현 계층을 설계할 때 고려해야할 사항을 서술하고 3장에서는 표현 계층 설계 패턴에 대해 서술한다. 4장에서는 이러한 설계 패턴을 역할에 따라 분류해보고자 한다. 5장에서는 결론 및 향후 연구 방향을 제시한다.

본 연구는 정보통신진흥원 대학기초연구(2001-168-2)지원 과제의 일부임

2. 표현 계층 고려 사항

표현 계층을 설계하고자 할 때에는 여러 가지 고려할 사항들이 있다[2]. 그 예로는 보안 문제, 데이터 통합 문제, 관리 문제, 확장성 문제 등이 있다. 이러한 문제 중에서 반드시 고려해야 하는 문제는 세션(session) 관리, 클라이언트 접근 제어, 유효성이 있다. 이에 관한 사항을 자세히 언급하면 다음과 같다.

◆ 세션 관리(Session Management)

세션이란 클라이언트와 서버간의 다중 요청에 걸친 대화의 내용을 간직하는 연결을 의미한다. 하지만 웹 브라우저와 웹서버가 사용하는 HTTP 프로토콜은 세션 지향적이지 않다. 따라서 표현 계층의 근간을 이루는 Servlet과 JSP는 세션이라는 개념을 가지고 있다. 이러한 세션을 관리하는 것은 웹 컴포넌트 개발할 때 중요한 고려사

항이라고 볼 수 있다.

◆ 클라이언트 접근 제어(Controlling Client Access)
클라이언트가 서버에 접근할 때 접근 사용자의 권한은 상당히 중요하다. 접근 권한이 없음에도 접근이 가능하다면 다중 사용자 환경인 웹에서 여러 가지 보안 문제가 노출되게 된다. 이러한 문제를 해결하기 위해서는 사용자의 역할에 따라 서로 다른 수준으로 접근해야 할 것이다. 따라서 이에 대한 방비책 또한 중요하다.

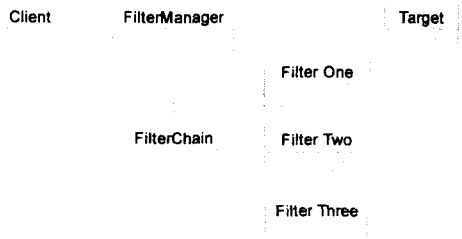
◆ 유효성(Validation)
유효성의 검사는 서버 측과 클라이언트 측의 서로 다른 유효성 원칙에 의하여 이루어진다. 클라이언트 측에서는 잘못된 입력이 되어 있는 지를 확인하는 JavaScript 등을 이용하여 상위 레벨의 유효성 검사를 한다. 서버측면의 유효성 검사는 좀더 포괄적인 유효성 검사를 의미한다.

3. 표현 계층 설계 패턴

표현계층 설계에는 여러 가지 패턴들이 적용될 수 있다. 이러한 패턴들은 기존의 디자인 패턴을 J2EE플랫폼에 맞게 변형화 된 것이다. 또한 이러한 패턴의 공통점은 표현계층이 웹 계층의 여러 요청 또는 응답을 처리하기 위한 메커니즘이라는 것이다. 이러한 각 패턴들의 예는 다음과 같다[2].

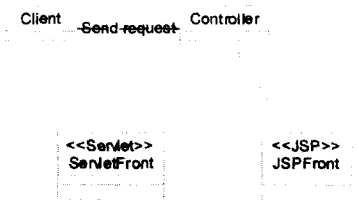
3.1 Intercepting Filter

표현 계층의 요청 처리 메카니즘은 자료 처리를 위해 요구되어지는 여러 가지 타입들을 처리하기 위한 서로 다른 많은 요청을 받는다. 몇몇의 요청들은 적절히 취급 가능한 컴포넌트들에 간단히 전달되어지는 반면에 간단히 전달하기 전에 미리 조절되고, 적절한 전처리가 필요하기도 한다. 이러한 클라이언트 측면의 웹 요청과 응답을 위해 전처리 그리고 후처리 과정이 필요하다. 이러한 문제를 해결하기 위한 패턴이다.



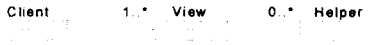
3.2 Front Controller

웹 계층은 각 사용자들의 다중 요청들을 제어하거나 중개자 역할을 할 수 있어야 한다. 이러한 제어 메커니즘은 중앙 집중형으로 설계되거나 혹은 그 반대의 방법으로 관리되어 진다. 이러한 문제를 해결하기 위한 패턴이다. 이 패턴은 시스템 서비스, 정보 검색, 표현계층 관리, 그리고 웹 계층의 메뉴처리들을 위해 통합된 처리를 지원한다. 통합된 처리를 위해 특히 이 패턴은 중앙 처리를 위한 접근 시작점을 제공한다.



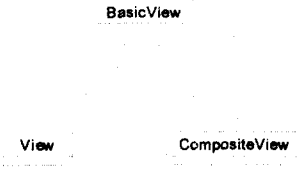
3.3 View Helper

동적인 비즈니스 데이터 처리를 필요로 하는 표현계층의 내용을 생성한다. 즉, 하나의 View에 동적인 데이터를 JavaBean Helper Class를 이용하여 화면의 내용을 동적으로 생성해낸다.



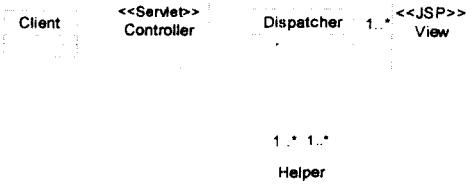
3.4 Composite View

단일 View페이지로 웹 페이지를 구성하는 것이 아니라 여러 개의 다중 하위View를 이용하여 여러 데이터를 표현하는 페이지를 만들어 낸다.



3.5 Service to Worker

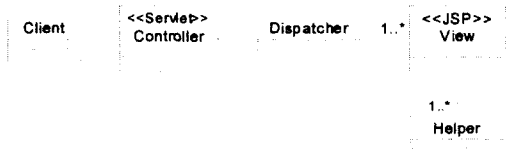
생성되어진 표현계층의 내용으로부터 비즈니스 데이터에 접근하고 실행시켜주는 흐름을 제어하고자 할 때 사용한다. 이는 후에 언급될 Dispatcher View와 비슷하지만 Controller가 Helpers들에 대한 정보 조작을 위임받는다는 것이다.



3.6 Dispatcher View

앞에 서술된 Service To Worker 패턴과 유사하다. Service to Worker 패턴과 다른 점은 Controller가

Helper들에 대한 정보 조작에 대해 위임받지 않는다는 점이다.



서술된 패턴들을 정리하면 다음 표1과 같다.

표 1 설계 패턴의 종류

Pattern Name	Synopsis
Intercepting Filter	클라이언트 요청에 대한 전처리 및 후처리를 용이하게 해준다.
Front Controller	클라이언트 요청의 운용 관리를 위해 중앙집중적인 제어기를 제공한다.
View Helper	표현계층의 형태와 관련되어있지 않은 로직을 헬퍼(Hepler) 컴포넌트로 캡슐화 시켜준다.
Composite View	최소단위의 하위 컴포넌트들을 모아 하나의 표현계층을 생성한다.
Service To Worker	Front Controller패턴과 View Helper패턴과 함께 전달자 컴포넌트를 조합한다.
Dispatcher View	표현계층에 주안점을 두고 Front Controller패턴과 View Helper패턴과 함께 컴포넌트를 조합한다.

4. 표현 계층 설계 패턴 분류

표현 계층에 사용되어지는 기술은 JSP, Servlet, JavaBean이 있다. 이 기술들은 각각에 따라 주어진 역할이 달라진다. 각 기술의 역할과 특징을 살펴보면 첫째, JSP는 화면의 레이아웃을 만드는데 사용되며 그 특징으로 HTML을 생성하고 결과물을 산출해낸다. 둘째, Servlet은 경계역할을 한다. 즉 JavaBean과 JSP 기술 사이를 이어주는 역할을 한다. 이의 특징은 요청을 받아들이고, 입력 데이터를 체크하고, JavaBean 객체를 생성하여 JSP 객체를 호출한다. 마지막으로 JavaBean은 데이터를 수집하거나 유지하는 역할을 하며 이의 특징은 특정 시나리오를 수행하고, 결과 데이터를 수집하고 그 데이터를 유지한다. 이를 요약하면 다음 표2와 같다[3].

표 2 기술의 역할

기술	역할
JSP	레이아웃
Servlet	요청처리
JavaBean	데이터의 수집과 유지

앞에서 서술된 패턴들은 위의 기술들을 사용하는 설계 패턴이다. 현재 비즈니스 계층을 구성하는 주요 요소로는 EJB 컴포넌트를 들 수 있다. 비즈니스 계층은 많은 사람들의 관심으로 인하여 상당히 많은 표준화가 되어 있으며 이를 바탕으로 비즈니스 컴포넌트를 자동으로 생성해주는 CASE 도구의 수도 많다. 그러나 현재 표현 계층에 대한 심층적인 분석이나 표현 계층을 테트스 용 이외의 목적으로 자동 생성해주는 도구는 아직 없다. 따라서 본 논문에서는 표현 계층에 사용되는 패턴을 역할에 따라 분류하여 표준화를 이루고자 한다. 이러한 분류는 각 패턴을 구성하는 주요기술을 분석하여 역할에 따라 레이아웃, 요청처리, 데이터 표현으로 나누어 분류하여 보았다.

각 패턴을 구성하는 주요기술과 역할에 따라 분류한 표는 다음 표3과 같다.

표 3 기술에 따른 설계 패턴 분류

역할	패턴	주요기술
레이아웃	Composite View	JSP
요청 처리	Intercepting Filter Front Controller	Servlet
데이터 표현	View Helper Service to Worker Dispatcher View	JSP,Sevlet,JavaBean

표3의 분류에 사용된 각 역할들은 차후 연구될 표현계층 자동 생성 템플릿 분류에 바탕이 되는 핵심 요소이다.

5. 결론 및 향후 연구

J2EE 기반의 웹 애플리케이션은 역할 기반으로 개발되어진다. 이는 개발자들이 자신의 역할에 충실하게 개발하도록 하여 보다 빠르게 애플리케이션을 개발 할 수 있도록 하는데 중점을 둔 구조이다. 그러나 현실적으로 각 역할을 담당한 개발자들이 애플리케이션을 만들어 나갈 때 서로의 지속적인 상호작용을 요구함으로서 개발의 비효율성을 일으킬 수 있다. 또한 현재 J2EE 기반의 웹 컴포넌트의 구조에 대해 명확한 표준이 없는 상황이다. 본 논문에서는 위에 제기된 문제를 해결할 수 있는 표현계층의 근간 요소인 웹 컴포넌트를 작성할 때 고려해야 할 사항과 패턴에 대하여 알아보고 이를 정형화 할 수 있도록 특징을 뽑아 분류하여 보았다. 향후 연구 과제로는 본 논문에서 언급한 분류를 바탕으로 J2EE 기반 웹 애플리케이션의 표현 계층을 구성하는 웹 컴포넌트에 대한 템플릿을 설계하고자 한다. 이러한 템플릿이 설계되면 웹 컴포넌트의 표준화뿐만 아니라 J2EE 기반의 웹 애플리케이션을 빠르게 작성할 수 있다.

참고 문헌

[1] Sun Microsystems, inc., "Designing Enterprise Applications with the Java 2 Platform, Enterprise edition," v.12, 1999.

[2] D. Alur, J. Crupi, D. Malks, "Core J2EE PATTERNS," Prentice Hall, 2001.

[3] A. Saimi, T. Syomura, H. Suganuma, I. Ishida, "Presentation layer framework of web application systems with server-side Java technology," IEEE, 2000.