MVC 프레임워크 기반의 CMS 설계 및 구현

Design and Implementation of CMS using MVC Framework

이 준 희
(주)아이젠소프트

Lee Jun-Hee
Igensoft co., Ltd.

요약

웹 서비스는 WSDL 서비스 인터페이스와 서비스 구현 문서를 이용하여 정적으로 호출되거나 서비스 타입 정의와 UDDI를 통한 서비스 구현을 검색함으로서 동적으로 호출될 수 있다. 지금까지는 동시에 두 가지를 수행 할 수 없었다. 본 논문에서는 Model View Controller(MVC) 프레임워크를 사용한 정적/동적 웹 서비스를 지원하는 효율적인 멀티미디어 CMS(Contents Management Systems)를 설계하고 구현하였다.

Abstract

Web services can be invoked statically using a WSDL service interface and service implementation documents, or dynamically by retrieving the service type definitions and the service implementation via UDDI. But until now, couldn't do both at the same time. The Model View Controller pattern(MVC) supports both dynamic and static Web services. The MVC paradigm is a way of breaking an application, or even just a piece of an application's interface, into three parts: the model, the view, and the controller. Object-oriented methodologies have been applied to developing internet applications for increase reuse and expandability. I supposed a effective multimedia CMS(Contents Management Systems) using MVC.

1. 서론

객체지향 패러다임은 프로그램 이해와 유지보수가 용이하고 확장하기 쉽기 때문에 최근 프로그램 개발에 가장 많이 적용되고 있는 분야이다. 또한 재사용 기술의 전환을 가져오면서 단순히 원시코드를 재사용하는 수준이 아닌 소프트웨어 구조를 구성하는 프레임워크와 디자인 패턴[1, 2] 등을 재사용하여 설계 단계의 생산성을 높일 수 있게 되었다.

특히 멀티미디어 콘텐츠 저작 기술이 발전하면서 콘텐츠는 점점 대용량화되고[3], 특히 웹사이트의 콘텐츠 양이 많아지고 사이트가 복잡해지면서 다양한 콘텐츠를 효율적으로 관리하기 위해서 객체지향 패러다임을 이용한 CMS의 설계 및 구현은 매우 필요하게 되었다.

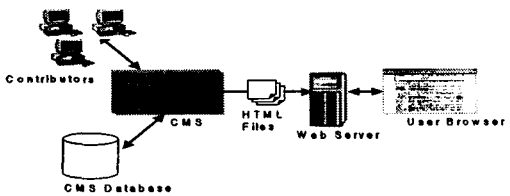
MVC 패러다임은 애플리케이션을 나누는 방식이다. 또는, 하나의 애플리케이션 인터페이스를 세 부분(모델, 뷰(view), 컨트롤러)로 나눈다.

모델(model)은 엔터프라이즈 데이터와 액세스를 관리하는 비지니스 규칙을 나타내고 데이터를 업데이트한다. 뷰(view)는 모델의 콘텐츠를 만든다. 모델을 통해 엔터프라이즈 데이터에 액세스하여 데이터가 어떻게 나타나야 하는지를 정한다. 모델이 변할 때 구현 영속성을 유지하는 것도 뷰가 할 일이다. 이를 위해 push model을 이용한다. pull model의 경우 뷰는 최신 데이터를 검색할 때 모델 호출을 담당한다. 한편 컨트롤러는 뷰를 가지고 있는 인터랙션을 액션으로 바꿔서 모델에 의해 수행될 수 있도록 한다.

모델에 의해 수행된 액션에는 비지니스 프로세스의 활성화 또는 모델 상태 변경이 포함된다. 컨트롤러는, 사용자 인터랙션과 모델 액션의 결과에 근거하여 적절한 뷰를 선택함으로서 반응한다.

1.1 CMS 분류

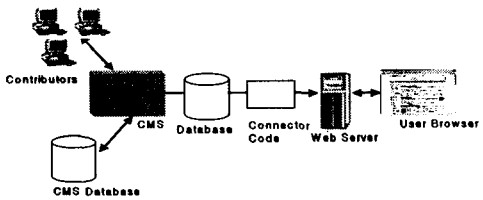
CMS를 정적 페이지 방식, 동적 페이지 방식, Hybrid 방식으로 크게 분류하여 주요 특징을 설명하면 아래와 같다. 그림 1은 정적 페이지 방식의 CMS 구조를 보여준다.



▶▶ 그림 1. 정적 페이지 방식

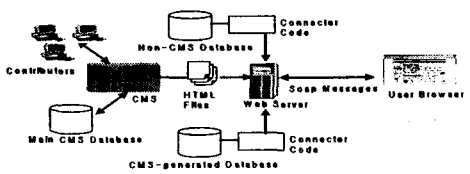
정적 페이지 방식은 운영 서버측의 부하를 줄일 수 있으나 다양한 기기 및 동적인 페이지 처리가 어렵다는 특징이 있다. 그림 2는 동적 페이지 방식의 CMS을 사용한 예를 보여준다.

이 방식은 콘텐츠 동기화가 용이하나 운영 서버측에 부하를 많이 준다.



▶▶ 그림 2. 동적 페이지 방식

그림 3은 동적 페이지 방식과 정적 페이지 방식을 동시에 사용하는 Hybrid CMS이다.



▶▶ 그림 3. Hybrid 방식

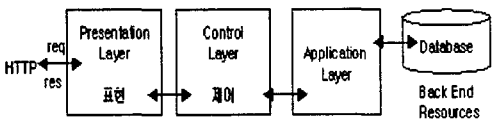
Hybrid 방식은 관리가 동적과 정적 페이지 방식의 장점을 모두 가지나 기술상의 구현이 어려운 점이 있다.

현재 각기 다른 방식으로 구현된 CMS는 일관된 콘텐츠 저장 및 관리 정책의 부재로 통합적인 콘텐츠 관리 및 Time-to-Web 지연과 콘텐츠의 질적인 저하를 가져오는 문제점이 있다. 따라서 본 논문에서는 위와 같은 문제점을 해결하기 위해서 멀티미디어 콘텐츠를 효율적으로 처리하고 코드 재사용의 용이화를 위해서 MVC 프레임워크에 기반한 Hybrid 방식의 CMS을 설계하고 구현하였다.

2. 본론

웹 애플리케이션이 어떤 복잡도를 가지고 있든지 간에, 애플리케이션 아키텍처는 아래의 세 가지 논리 계층으로 나누어 질 수 있다.

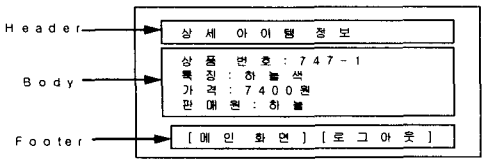
1) 애플리케이션 로직 계층: 애플리케이션의 데이터를 관리하고 백-엔드 리소스를 사용하거나 계산을 수행하는 부분으로서, 모델(model)이 된다.
2) 프리젠테이션 계층: 룩 앤 필(look and feel)을 조정하고 실행 결과를 보여주는 프론트-엔드로서, 뷰(view)가 된다.
3) 컨트롤 계층: 애플리케이션의 흐름을 제어하는 부분으로 컨트롤러(controller)가 된다.



▶▶ 그림 4. 웹 애플리케이션의 논리 계층

위의 세 계층은 MVC 패턴과 흡사한 구조를 가진다.

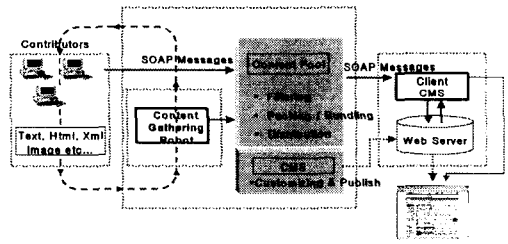
CMS는 페이지 분해를 통해 복잡도를 떨어뜨리고 관리의 용이를 위해서 그림 5와 같은 카탈로그 정보를 갖는 CMS 페이지에서 공통적인 상단 부분(header, 페이지 이동 패널과 로고가 포함된), 구체적인 항목 정보, 그리고 페이지를 마무리하는 하단 부분(footer)으로 나누어 설계될 수 있다.



▶▶ 그림 5. CMS 구성 요소 분해

2.1 제안 시스템

제안 시스템에서 사용되는 데이터베이스는 기존의 관계형 데이터베이스를 XML 기술과 통합하여 구현하였다. 멀티미디어 콘텐츠 정보를 XML로 처리함으로써 웹으로 쉽게 서비스할 수 있어 시간과 장소에 상관없이 멀티미디어에 대한 접근을 용이함으로 웹 서비스를 지원하도록 하였다. 그림 6은 MVC 프레임워크 기반의 CMS의 전체적인 구조를 보여준다.



▶▶ 그림 6. MVC 프레임워크 기반 CMS

웹 서비스는 플랫폼과 언어에 관계없이 표준 인터넷 프로토콜에 기반한 재사용 가능한 컴포넌트를 말하며, TCP/IP와 같은 인터넷 기술을 사용하여 응용 프로그램이 어떤 작업을 수행하는 프로시저나 함수들의 집합을 의미한다. 웹 서비스에서 각 장치나 애플리케이션이 이해할 수 있는 언어로서 부각되는 것이 XML로써 플랫폼과 언어에 독립적인 방법으로 통신하게 해준다.

웹 서비스는 XML을 기본 데이터로 표시하며, 데이터 인코딩 표준인 SOAP라는 불리는 RPC를 통한 메시지 형식을 정의하는 프토토콜을 사용한다.

SOAP[4]은 XML과 HTTP를 기반으로 네트워크 상에 존재하는 각종 컴포넌트간의 호출을 효율적으로 할 수 있게 하는 통신 프로토콜로서 이를 이용하여 .NET 플랫폼 및 어떤 애플리케이션과도 서로 호출할 수 있다.

SOAP 클라이언트는 HTTP를 통해 웹 서비스로 SOAP 메시지를 전송한다. 웹 서버는 메시지를 받아서 SOAP 서버에게 위임하며, SOAP 서버는 맞는 웹 서비스를 차례로 호출한다.

현재 웹 서비스는 다양한 분야에 활용이 되고 있으며 특히 Excel에서 웹 서비스 활용은 웹 사이트[5]에서 weather를 검색해서 사용할 수 있다.

XML 웹 서비스 호출은 동기 호출과 비동기 호출로 구분 할 수 있는데 각각의 장단점을 가지고 있다. 동기 XML 웹 서비스 호출은 코딩이 간단한 반면, 비동기 호출은 좀 더 복잡한 코딩이 요구된다. 동기 호출은 XML 웹 서비스가 오랜 시간동안 응답하지 않으면 응용 프로그램이 멈추게 된다. 반면에 비동기 호출은 XML 웹 서비스 서비스 호출이 진행되는 동안 사용자는 응용 프로그램과 상호 작용할 수 있다.

모바일 기기에서 XML 웹 서비스를 사용하는 경우 몇 초 동안의 초기 지연이 발생한다. 비동기 호출을 만들면 이러한 지연 현상이 있어도 응용 프로그램의 사용자 인터페이스가 멈추지 않는다. 이러한 지연은 런타임이 XML 웹 서비스에 대한 연결 세부 사항을 설정하고 그 세부 사항을 캐시할 때 발생한다. 이런 경우 응용 프로그램이 응답하지 않는 현상이 종종 발생한다. 연속으로 XML 웹 서비스를 호출하면 성능이 크게 향상된다.

비동기 XML 웹 서비스 호출을 만들려면 동기 호출을 만드는 것보다 좀 더 많은 코드가 필요하지만, 비동기 호출은 결과적으로 보다 나은 응답 속도로 응용 프로그램을 수행할 수 있다. 제안한 시스템에서는 비동기 호출을 통한 XML 웹 서비스를 이용하였다.

한편, XML 문서를 오브젝트로 처리하기 위해서 DOM[6-8]을 사용하였다.

DOM(Document Object Model)은 여러 클래스를 사용되지만 시스템에서 기본적으로 사용된 클래스는 아래와 같다.

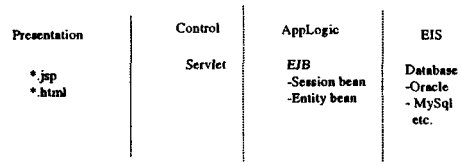
- XMLDOMNamedNodeMap: XML 문서의 요소가 갖는 속성을 보관하는데 사용
- XMLDOMNode: XML 문서의 요소(노드)를 보관하는 오브젝트 클래스
- XMLDOMNodeList: XML 문서의 노드의 목록을 보관하는 오브젝트의 클래스
- DOMDocument: XML 문서 전체를 보관한 오

브젝트의 클래스

XML 문서를 여러 형식으로 가공하기 위해서 XSL을 사용되는데 XSL(XSL Transformation) 프로세서를 사용하여 CMS 접속 클라이언트의 종류에 따라서 서비스하도록 하였다.

2.2 시스템 구현

제안 시스템은 J2EE(Java™ 2 Platform, Enterprise Edition) 사용하고 애플리케이션의 재사용성과 확장성을 고려해서 MVC 설계를 통해서 웹 서비스를 지원하도록 분산된 웹 서버와 연동하여 각종 콘텐츠를 제공할 수 있도록 하였다. 그림 7은 J2EE의 아키텍처를 나타낸다.



▶▶그림 7. J2EE 아키텍처

2.3 결과 고찰

복합 페이지 뷰 패턴은 콘텐츠 요소를 최대한 잘게 쪼개어 재사용성을 높이고 프리젠테이션 설계와 개발의 과정을 쉽게 만들고 있다. 중복적으로 사용되는 정보를 분해하고 이것을 제각기 파일에 넣어 둠으로써 얻는 이점은 두 가지이다.

첫째로, 공통적인 코드를 재사용함으로써 콘텐츠 생성 속도와 디자인 변경 속도가 향상되었고 둘째로, 구체적인 동작 구현과 리소스의 관리를 각각의 담당 부분에 맡기기 때문에 사이트의 품질 향상과 유지비용 감소를 얻을 수 있었다.

특히 분산환경에서 웹 서비스 기반 CMS는 통합적인 관리를 제공하기 때문에 매우 우수한 사이트 품질 향상과 오디오, 비디오 데이터 등의 멀티미디어 데이터 처리의 효율적인 가공 및 인터넷을 통한 다양한 포맷을 통한 배포가 가능하였다.

3. 결론

기존 CMS는 한 번 가공할 때마다 작업과 관련된 비용을 발생시키는데, 이러한 구조는 결국 콘텐츠에 대한 최적의 관리를 가로막는 장애가 되고 있다.

본 논문에서 제안한 MVC 프레임워크 기반의 CMS는 첫째, 콘텐츠를 한번 생성시켜서 XML 포맷으로 저장하고 이 콘텐츠를 여러 곳에서 재사용하여 컴포넌트 환경의 장점인 콘텐츠 관리의 효율성을 가져오고 개발 생산성의 증가 효과를 기대할 수 있다.

둘째, XML 기반의 CMS는 콘텐츠의 각종 메타데이터들을 정교하게 관리할 수 있는 방법을 제공함으로써 콘텐츠 서비스의 품질을 향상시킬 수 있다.

셋째, 콘텐츠의 내용과 Presentation을 분리시켜 관리하므로 다양한 기기에 맞게 콘텐츠를 제공할 수 있다.

향후 연구과제로는 좀더 체계적으로 웹 관련 컴포넌트들을 개발하여 다양한 환경에서 쉽게 적용할 수 있는 CMS 설계가 있어야 할 것이다.

▌참고문헌▐

[1] Sherman R. Alpert, Kyle Brown and Bobby Woolf, *The Design Patterns Smalltalk Companion*, Addison-Wesley, 1997.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[3] 홍석기, 류한영, "멀티미디어 타이틀의 정보구조화에 관한 연구", 울산대학교 논문집, 제39집, pp.591-605, 1994.

[4] SOAP 1.2 Spec., http://www.w3.org/TR/SOAP

[5] http://www.xmethods.com/

[6] http://www.jdom.org

[7] http://www.dom4j.org

[8] http://zvon.org/xxl/DOM2reference/