퍼지 및 지능시스템학회 논문지 2005, Vol. 15, No. 7, pp. 794-799

동적 컨텐츠 캐싱을 지원하는 XML 기반의 컨텐츠 관리 시스템의 구현

An XML-based Content Management System supporting Dynamic Content Caching

구흥서
Koo, Heung-Seo

청주대학교 컴퓨터정보공학과

요 약

본 논문에서는 효율적인 동적 컨텐츠 출판을 지원하는 XML 기반 웹 컨텐츠 관리 시스템인 EasyCM을 설계하였다. EasyCM은 효율적인 동적 웹 컨텐츠 출판을 지원하기 위해 XML 기반의 컨텐츠 관리 시스템의 출판에서 필요한 컨텐츠 저장소로부터 XML 객체의 추출과 XSLT를 이용한 HTML 변환의 추가적인 처리과정을 전처리하여 컨텐츠 컴포넌트를 생성하고 사용자 요청시 컨텐츠 컴포넌트를 최종 웹 페이지로 조립하여 출판함으로써 컨텐츠의 재사용성을 향상시켰다. 또한 EasyCM의 성능을 향상시키기 위해서 기반 데이터와 동적 컨텐츠 컴포넌트간의 종속정보를 Dependency Map으로 유지하여 동적 컨텐츠 캐싱을 지원한다. 이를 위해서 EasyCM은 XML 출판 프레임워크인 Cocoon2를 기반으로 하고, 동적 컨텐츠 캐싱이 가능하도록 Cocoon2의 캐싱 구조를 확장하였다. 본 논문에서 설계한 확장된 캐싱 시스템은 캐싱된 컨텐츠의 효율적인 갱신을 지원하기 위하여 동적 컨텐츠의 특성에 따른 두 가지 갱신 유형, 즉 즉시갱신과 지연갱신을 지원한다.

Abstract

In this paper, We describe the XML-based Web content management system that supports the efficient dynamic content publishing environment. EasyCM is designed based on Cocoon2 that is the XML publishing framework. We propose the publishing mechanism to support the efficient dynamic content publishing environment to expand into the available dynamic content caching to Cocoon2. Publishing mechanism of EasyCM draws XML object from content repository, associates XML with XSLT, creates and caches content components preprocessing HTML transformation process, and publish web pages constructed into cached content component. For supporting more efficient caching, EasyCM supports also content component update, two update method that is immediately-update and delay-update for updated content component.

Key Words : 컨텐츠, 컨텐츠 컴포넌트, 동적 캐싱, 동적 컨텐츠 출판, 컨텐츠 관리 시스템.

1. 서 론

오늘날 웹사이트가 기업 비즈니스의 중요한 부분으로 정착되면서 웹 컨텐츠의 양적 증가와 다양화로 인하여 기존의 수작업 방식의 컨텐츠 관리는 비효율성과 일관성 결여 문제가 나타날 수 있다. 그러므로 최근에는 컨텐츠의 생성, 관리, 출판 등 일련의 과정을 자동화하는 컨텐츠 관리 시스템(CMS : Content Management System)의 활용이 증가하고 있다. 컨텐츠 관리 시스템은 데이터베이스와 같은 컨텐츠 저장소(repository)에 저장된 동적 컨텐츠를 템플릿(template)을 이용하여 출판한다. 따라서 컨텐츠 관리 시스템을 이용한 웹 컨텐츠의 출판은 정적 출판 방식에 비해 컨텐츠 저장소로부터의 컨텐츠 추출시간과 템플릿을 이용한 컨텐츠 변환시간이 추가적으로 필요하기 때문에 서버의 부하와 지연시간을 가중시킨다. 이러한 컨텐츠 관리 시스템에서 나타나는 출판 과정의 성능저하 문제를 해결하기 위해 많은 컨텐츠 관리 시스템이 캐싱(caching) 기능을 지원하지만, 일반적으로 정적 컨텐츠를 대상으로 하므로 동적 컨텐츠가 증가하면 시스템 성능이 저하되는 문제가 있다[1].

본 논문에서는 동적 컨텐츠 캐싱을 지원하는 XML (eXtensible Markup Language) 기반의 컨텐츠 관리 시스템을 설계, 구현한다. 본 논문에서 구현한 컨텐츠 관리 시스템은 XML 기반의 웹 애플리케이션 프레임워크인 Cocoon2[2]를 기반으로 하여 설계하였고, Cocoon2에서 동적 컨텐츠 출판시 나타나는 성능저하 문제를 해결하기 위해 Cocoon2를 동적 컨텐츠 캐싱이 가능하도록 확장한다.

본 논문에서 제안한 컨텐츠관리시스템의 동적 출판 메커니즘은 컨텐츠 재사용성의 향상과 출판과정의 성능 향상을 위해 컨텐츠 저장소로부터 XML 객체의 추출과정과 XML과 XSLT(XML Stylesheet Language Transformation)를 조합하여 HTML로 변환하는 과정을 전처리(preprocessing)하여 컨텐츠 컴포넌트로 미리 생성하여 캐싱한 다음, 사용자 요청이 발생하면 캐싱된 컨텐츠 컴포넌트를 조립하여 웹 페이지 형태로 출판한다. 또한, 좀더 효율적인 캐싱을 위하여 컨텐츠 컴포넌트의 갱신과 접근빈도의 특성에 따라 즉시-갱신과 지연-갱신의 두 가지 갱신유형을 지원한다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서

접수일자 : 2005년 12월 2일
완료일자 : 2005년 12월 7일

관련 연구로 동적 컨텐츠 캐싱에 관한 연구들을 살펴보고, 3장에서 본 논문에서 설계한 동적 컨텐츠 캐싱 기법을 설명하며, 4장에서는 3장의 Cocoon2 프레임워크 기반으로 설계한 웹 컨텐츠관리시스템과 동적 캐싱 기법이 구현된 웹 출판 메커니즘에 대해 설명한다. 그리고 5장에서 결론을 맺는다.

2. 관련연구

동적 컨텐츠 출판은 데이터베이스 액세스, 비즈니스 로직 수행 등의 추가 처리과정이 필요하기 때문에 일반적으로 정적 컨텐츠 출판보다 더 많은 서버 부하(load)로 인한 지연시간을 가진다. 따라서, 최근에는 동적 컨텐츠를 대상으로 하는 캐싱 시스템에 관한 연구[1,3,4,5,6]들이 활발히 진행되어 왔다. 이번 장에서는 동적 컨텐츠 캐싱에 관한 대표적인 3가지 기술에 대하여 살펴본다.

2.1 DUP 접근방법

DUP(Data Update Propagation)는 IBM이 1998년 동계 올림픽 공식 사이트의 성능향상을 위해 동적 웹 컨텐츠 캐싱 알고리즘으로 개발되었다[1]. DUP 알고리즘은 기반 데이터와 컨텐츠 객체의 종속성들을 유지하기 위해 그림 1과 같은 방향성 그래프인 ODG(Object Dependence Graph)를 사용하였다.

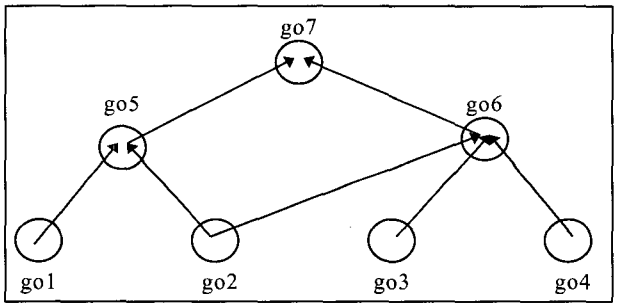

그림 1. ODG(Object Dependency Graph)의 예.
Fig. 1. Example for Object Dependency Graph.

DUP 알고리즘에서 ODG는 깊이(depth)가 3으로 제한되며, go1~go4 같은 단말 노드들은 데이터베이스에 저장된 기반 데이터를 나타내고, go7 같은 최상위 노드들은 웹페이지를 나타낸다. go5, go6 같은 중간 노드들은 가상 객체라고 하며 기반 데이터의 변경 정보를 실제 페이지에 효율적으로 전파(propagation)하는데 사용된다. 그리고 노드와 노드를 연결하는 에지(edge)는 각 노드들간의 종속성을 나타낸다.

만일 기반 데이터가 변경되면, 캐시 관리자로 변경사항이 전달되고 캐시 관리자는 VIB(Vertex Information Block)를 참조하여 변경된 기반 데이터의 종속성을 ODG 순회 알고리즘을 사용하여 계산한다. 그리고 그 계산결과를 기반으로 하여 가상 객체와 페이지를 변경하게 된다. 따라서 캐싱된 객체의 변경이 사용자 요청에 의해 수행되는 것이 아니라 데이터 변경시 수행되기 때문에 100%에 가까운 캐시 히트율이 가능하게 된다.

2.2 플래그먼트 기반 접근방법

IBM에서는 2000년 올림픽 사이트의 효율적이고 일관된 동적 웹 컨텐츠 출판을 위해 플래그먼트(fragment) 기반의 페이지 생성 기법을 사용하였다[3]. 플래그먼트 기반 접근방법은 동적 페이지 생성시 서버부하를 줄이기 위해 복잡한 웹 페이지를 좀더 간단한 플래그먼트를 조합하여 구성한다.

플래그먼트란 그림 2와 같이 웹 컨텐츠의 재사용을 고려하여 웹 페이지를 보다 작은 단위로 분할한 객체이다. 플래그먼트 기반 접근방법은 기반 데이터가 아닌 플래그먼트로부터 페이지를 조립하여 생성하기 때문에, 기반 데이터로부터 페이지 전체를 생성하는 경우 보다 작업부하를 줄여서 웹 컨텐츠 출판의 효율성을 높을 수 있다.

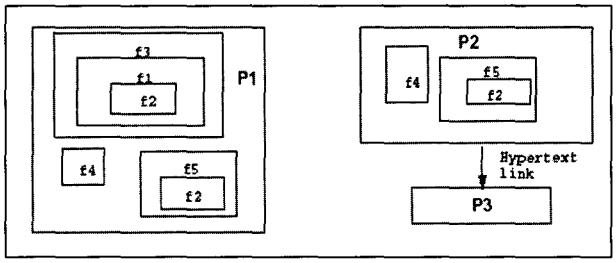

그림 2. 플래그먼트로 구성된 웹 페이지의 예.
Fig. 2. Example for Web Pages composed of Fragment.

플래그먼트로 구성된 복잡한 웹 페이지를 관리하기 위해 마크업 언어로 템플릿을 작성함으로써 웹 페이지와 플래그먼트의 포함관계를 정의한다. 이러한 포함관계는 그림 3과 같은 ODG(Object Dependence Graph)로 표현된다. ODG의 그래프 순환 알고리즘은 플래그먼트 변경이 웹사이트 전체에 전파되는 경로를 찾아낸다. ODG는 노드와 에지(edge)로 구성되는데, 각 노드는 페이지와 플래그먼트를 나타내고, 노드와 노드를 연결하는 에지는 객체들간의 포함관계를 나타내는 포함 에지(inclusion edge)와 하이퍼텍스트 링크를 나타내는 링크 에지(link edge)로 구성된다.

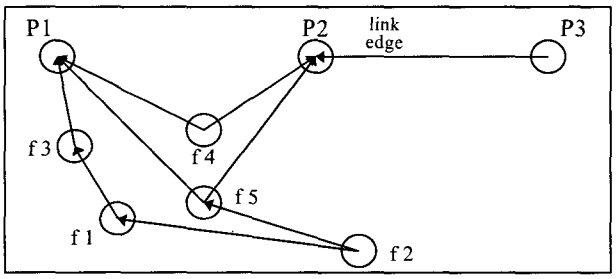

그림 3. ODG(Object Dependency Graph)의 예.
Fig. 3. Example for Object Dependency Graph

플래그먼트 기반 접근방법에서 하나 이상의 객체들의 집합 S가 변경되면 포함 에지를 따라 S로부터 도달할 수 있는 모든 노드들을 결정하기 위해 토폴로지컬 정렬(topological sort)을 사용한 깊이-우선(depth-first) 그래프 순환을 한다. 예를 들면, 그림 3의 그래프에서 P3, f4, f2의 변경후의 토폴

로지컬 정렬의 값은 P3, f4, f2, f5, P2, f1, f3, P1이 된다. 플래그먼트 기반 접근방법에서는 기반 데이터의 변경으로 인해 수행되는 갱신 작업의 일관성을 보장하기 위하여 토폴로지컬 정렬 순서에 의거한 점진적 갱신 기법을 사용한다.

2.3 Cachuma 접근방법

DUP과 플래그먼트 기반 접근방법에서 사용한 일관성 유지 기법은 각각의 컨텐츠 객체들과 기반 데이터 간의 종속정보를 개별적으로 유지하는 fine-grain ODG를 구성한다. 이러한 fine-grain 방식은 동적 컨텐츠의 양이 증가하면 종속 그래프의 크기 또한 급격하게 증가하기 때문에 종속성을 관리하고 처리하는데 많은 작업부하를 발생시켜 결과적으로 시스템 성능이 저하되는 문제점이 있다. Cachuma 접근방법[4]에서는 fine-grain 방식을 보완하여 사용자가 기반 데이터 집합들과 공통 URL 패턴 또는 클라이언트 정보를 공유하는 URL 클래스라고 하는 동적 페이지의 그룹 간에 coarse-grain 방식으로 종속정보를 지정하도록 하여 좀더 포괄적이고 유연한 캐싱을 실현한다.

Cachuma에서는 웹 페이지들을 페이지 URL과 클라이언트 정보를 기반으로 한 클래스로 그룹화 한다. 페이지 그룹화는 개별적인 페이지가 아닌 그룹-기반의 캐싱과 무효화를 허용한다. URL 클래스에서 페이지를 분류하기 위한 URL 정보는 네트워크 경로, 애플리케이션 이름, 그리고 매개변수가 있으며, 추가정보로는 클라이언트 IP 주소와 쿠키(cookie) 값 등의 클라이언트 정보를 이용한다.

캐싱된 페이지는 트리거에서 전송된 메시지에 의해 무효화될 수 있으며, 무효화는 세가지 유형으로 분리하여 처리한다. 간접 클래스 무효화, URL 클래스의 직접 무효화, 그리고 개별적인 페이지의 직접 무효화가 그것이다.

Cachuma에서는 효율적인 무효화 처리를 위해 URL 클래스를 예측-가능(predictable)과 예측-불가능(unpredictable)으로 구분한다. 그리고, 클래스의 특성에 따라서 예측-불가능한 URL 클래스들은 페이지 무효화 지연(lazy page invalidation) 방식으로 예측-가능 URL 클래스들은 페이지 전처리(page precomputing) 방식으로 각각 처리한다. 페이지 무효화 지연 방식은 해당 캐싱된 페이지를 즉시 무효화시키지 않고 사용자가 해당 페이지를 요청하면 무효화를 수행함으로써 무효화 비용을 최소화한다. 예측-가능 URL 클래스에 속하는 페이지의 경우는 무효화 메시지를 전송받으면 즉시 무효화 처리를 한다. 이 방식은 기반 데이터가 변경되면 즉시 무효화된 페이지를 미리 전처리하여 캐싱시킴으로써 캐시 히트율을 상당히 개선하다는 장점을 가진다.

3. 동적 컨텐츠 캐싱

3.1 컨텐츠의 재사용

본 논문에서는 동적 컨텐츠의 효율적인 출판을 위해 템플릿을 이용한 컨텐츠 재사용과 페이지 분할을 통한 컨텐츠 재사용의 두 가지 컨텐츠 재사용 기법을 사용한다. 템플릿을 이용하면 동일 컨텐츠에 서로 다른 템플릿을 적용하여 다양한 형태로 출판함으로써 컨텐츠의 재사용이 가능하다. 템플릿을 이용한 접근방법은 컨텐츠와 스타일이 분리되기 때문에 개발자와 디자이너의 독립적인 개발환경을 지원하여 신속한 개발을 가능하게 하고, 동일 컨텐츠를 서로 다른 형태의 페이지에서 공유하기 때문에 컨텐츠 변경에 따른 일관성을 보장할 수 있다.

페이지 분할을 통한 컨텐츠 재사용은 일반적으로 웹사이트의 페이지들이 메뉴, 헤더와 같은 공통 컨텐츠를 포함하는 특성을 이용하여 페이지를 좀더 간단한 객체로 분할하고 이러한 객체를 조립하여 페이지를 생성함으로써 공통 객체들을 재사용한다. 페이지 분할을 통한 컨텐츠 재사용은 페이지를 좀더 간단한 객체를 조립하여 생성하기 때문에 웹 페이지 생성비용을 감소하고, 분할된 컨텐츠의 변경에 따른 웹 페이지의 일관성이 보장된다. 그림 4는 페이지 분할을 통한 컨텐츠 재사용을 나타낸 그림이다.

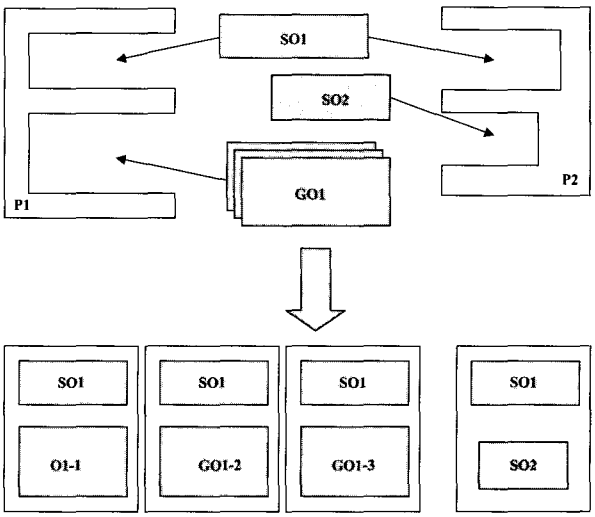

그림 4. 페이지 분할을 통한 컨텐트 재사용.
Fig. 4. Content Reuse using Page Fragmentation.

본 논문에서는 두 가지 컨텐츠 재사용 기법을 사용하여 그림 5와 같은 기반 데이터, 컨텐츠와 템플릿, 컴포넌트, 그리고 페이지의 4단계로 구성된 출판 메커니즘을 사용한다. 데이터베이스에 저장된 기반 데이터로부터 추출된 XML 컨텐츠에 XSLT 템플릿을 결합하여 컨텐츠 컴포넌트를 생성하고, 생성된 컨텐츠 컴포넌트들을 조합하여 최종 웹 페이지를 출판하는 과정을 거친다. 이때 컨텐츠 컴포넌트의 생성과정에서 단일 XML 컨텐츠에 서로 다른 XSLT 템플릿을 결합하여 서로 다른 형태의 컨텐츠 컴포넌트를 생성함으로써 템플릿을 이용한 컨텐츠 재사용 방법을 적용하였다. 또한 컨텐츠 컴포넌트를 조립하여 페이지를 생성하는 과정에서 동일 컨텐츠 컴포넌트를 여러 페이지에서 재사용 가능하도록 하여 컨텐츠를 재사용 한다.

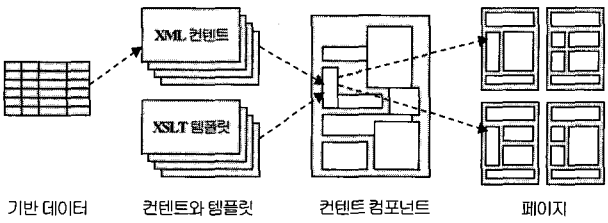

그림 5. 컨텐트 출판의 4 단계.
Fig. 5. Four Steps of Content Publishing.

3.2 캐싱 메커니즘

데이터베이스로부터 생성되는 동적 컨텐츠는 DB 접근과 비즈니스 로직 수행과 같은 추가적인 처리과정이 필요하므로 정적 컨텐츠에 비하여 서버의 부하를 가중시킨다. 또한 XML 컨텐츠를 웹으로 출판하려면 HTML 문서로의 변환과정이 필요하기 때문에 더욱 성능이 저하된다.

본 논문에서는 그림 5에서의 동적 컨텐츠의 생성과정(단계 1)과 XML 컨텐츠의 변환과정(단계 2)을 처리한 HTML 형태의 컨텐츠 컴포넌트를 캐싱하여, 사용자가 웹 페이지 요청시 캐싱된 컨텐츠 컴포넌트(단계 3)를 조립하여 신속하게 웹 페이지(단계 4)를 생성함으로써 웹 컨텐츠 출판성능을 향상시키는 방식을 사용한다. 이러한 방식은 재사용이 가능한 컨텐츠 컴포넌트를 캐싱하므로 캐시 저장공간을 절약할 수 있다. 캐싱된 컨텐츠 컴포넌트와 기반 데이터의 일관성 유지를 위하여 데이터베이스 테이블과 컨텐츠 컴포넌트 간의 종속정보를 ODG를 이용하여 표현한다. ODG의 깊이는 2로 제한되기 때문에 동적 컨텐츠의 증가에 따른 종속정보 관리비용 증가를 최소화하였다.

본 논문에서 구현한 캐싱 시스템은 XML 기반의 웹 애플리케이션 프레임워크인 Cocoon2를 확장하여 효율적인 동적 컨텐츠 출판 메커니즘의 구현하였다. Cocoon2의 현재 버전은 정적 컨텐츠 캐싱만을 지원하기 때문에 Cocoon2의 캐싱 시스템을 확장하여 동적 컨텐츠 캐싱이 가능하도록 하였다. 그림 6은 확장된 캐싱 구조의 알고리즘으로 동적 컴포넌트의 캐싱 및 응답 과정과 갱신 과정이다.

본 논문에서는 캐싱된 동적 컴포넌트 갱신의 효율성을 향상시키기 위해 Cachuma 접근방법의 페이지 무효화 지연(lazy-invalidation)과 페이지 전처리 기술로부터 동적 컴포넌트의 특성에 따른 즉시-갱신과 지연-갱신을 제공한다. 즉시-갱신은 사용자의 요청이 있기 전에 미리 캐싱된 컨텐츠를 갱신하기 때문에 캐시 히트율을 향상시키고 유효화 과정을 생략할 수 있다. 지연-갱신은 기반 데이터의 변경이 사용자 요청보다 빈번한 경우에 불필요한 갱신과정을 생략하기 때문에 서버의 부하를 감소시켜 성능향상의 장점이 있다.

3.2 확장된 캐싱 시스템

본 논문에서는 XML 기반의 컨텐츠 관리 시스템인 EasyCM 개발하기 위하여 먼저 XML 출판 프레임워크인 Cocoon2를 확장하여 효율적인 동적 컨텐츠 출판 메커니즘의 설계하였다. Apache Cocoon2는 정적 컨텐츠 캐싱만을 지원하기 때문에 Cocoon2를 확장하여 동적 컨텐츠 캐싱이 가능하도록 하였다.

확장된 캐싱 시스템의 구조는 그림 7과 같이 캐시 모듈이 확장된 Cocoon2를 이용하여 EasyCM과 통합되어 있다. 그림 7에서 EasyCM의 Component 관리 기능으로 생성된 동적 컴포넌트는 동적 컨텐츠 캐싱을 위한 Caching Dynamic Generator에 의해서 Cocoon Cache System에 캐싱된다. 이때 동적 컴포넌트의 종속정보는 동적 컨텐츠 캐싱을 관리하는 DynamicCacheManager에 의하여 dependencyMap에 등록된다. EasyCM의 Data 관리 기능으로 기반 데이터가 변경되면 EasyCM은 캐싱된 컴포넌트를 관리하기위한 CCM (Component Control Message)를 생성하여 CCM Manager.xsp를 통하여 CCMGenerator로 전달된다. CCM Generator는 DynamicCacheManager에게 해당 CCM에 적합한 작업을 요청하여 캐싱된 컴포넌트를 갱신한다.

```
단계 1  동적 페이지 요청
단계 2  파이프라인이 CachingDynamicGenerator에 동적 페이지에 사용된 컴포넌트 요청
단계 3  CachingDynamicGenerator가 CachingDynamicManager에 요청 컴포넌트전송
단계 4  CachingDynamicManager가 전송된 데이터에 대하여 유일키 생성
단계 5  if (dependencyMap에 저장된 키와 같지 않은가?)
            CachingDynamicGenerator가 새로운 컨텐츠 생성
            CachingDynamicManager가 컨텐츠 캐싱
        end if
단계 6  CachingDynamicManager가 유효한 메시지를 파이프라인에 전송
단계 7  파이프라인이 캐시에서 해당 컨텐츠 컴포넌트를 검색하여 사용자에게 반환
```

(a) 사용자 요청시
(a) In User Request

```
단계 1  기반데이터 변경
단계 2  CachingDynamicManager로 변경 테이블 정보 전송
단계 3  for (dependencyMap의 모든 테이블)
            if (dependency-Table이 변경 되었는가?)
                dependency-Table과 관련된 컴포넌트 일괄갱신
            end if
        end for
```

(b) 데이터 갱신시
(b) In Data Update

그림 6. 동적 컨텐츠 컴포넌트를 위한 캐싱 알고리즘.
Fig. 6. Caching Algorithms for Dynamic Content Components.

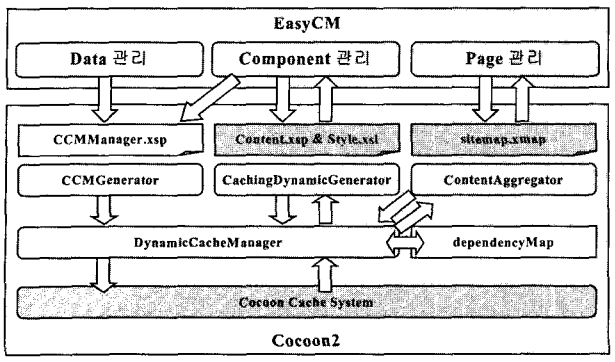

그림 7. 확장된 캐싱 시스템 구조.

Fig. 7. Extended Caching System Architecture.

확장된 캐싱 시스템의 주요 구성요소는 DynamicCache Manager를 중심으로 CCM(Component Control Message)을 처리하는 CCMManager.xsp와 CCMGenerator, 동적 컴포넌트를 캐싱 가능하도록 하는 CachingDynamicGenerator, 컨텐츠 컴포넌트에서 생성된 컨텐츠를 조립하여 웹 페이지를 생성하는 ContentAggregator로 구성되어 있다.

4. 컨텐츠 관리 시스템의 설계

Cocoon2는 XML 출판에 필요한 컨텐츠의 생성, 처리, 변환 그리고 배포하는 일련의 과정을 sitemap.xmap라는 설정 파일을 통하여 손쉽게 정의함으로써 XML 출판의 유용성을 향상시킨다. 하지만 sitemap.xmap 작성과 Cocoon2의 파이프라인 처리에 대한 전문적인 지식이 필요하다는 문제점이 있다. 본 논문에서는 Cocoon2에 대한 전문적인 지식이 없는 개발자가 간단하게 Cocoon2를 이용하여 웹사이트를 개발할 수 있는 웹 인터페이스 컨텐츠 관리 시스템인 EasyCM을 개발하였다.

4.1 시스템 구조

EasyCM의 구조는 그림 과과 같이 DB와 관련된 기능을 수행하는 Data Manager와 Cocoon2와 연동하여 웹 출판에 관련된 작업을 수행하는 Component Manager와 Page Manager로 구성되어 있다.

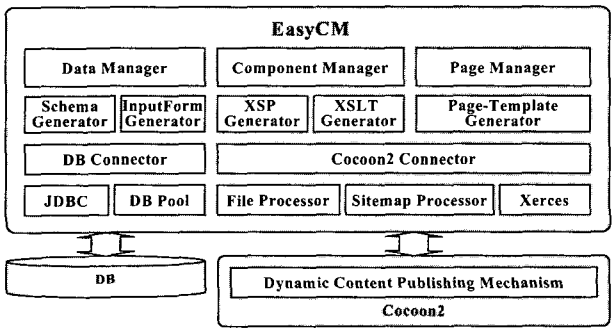

그림 8. 웹 CMS의 시스템 구조.

Fig. 8. Architecture of Web CMS.

Data Manager는 DB 스키마를 생성하는 Schema Generator와 생성된 DB 스키마를 이용하여 자동으로 데이터 입력폼을 생성하는 InputForm Generator를 포함하며 Wrapper 클래스인 DB Connector를 이용하여 DB의 접속 및 관련 작업을 수행한다. Component Manager는 XSP 코드를 자동 생성하는 XSP Generator와 XSLT 코드를 자동 생성하는 XSLT Generator를 포함하고, Page Manager는 XPT(eXtensible Page Template) 코드를 자동 생성하는 Page-Template Generator를 포함한다. Component Manager와 Page Manager는 래퍼(wrapper) 클래스인 Cocoon Connector를 이용하여 Cocoon2 시스템 내부의 관련 작업을 수행한다.

4.2 웹 출판 메커니즘

EasyCM에서는 이러한 4단계의 출판 과정을 Cocoon2 내부의 Content Aggregator를 이용하여 구현하였다. 그림 8은 Content Aggregator를 이용한 출판과정을 나타낸 그림이다.

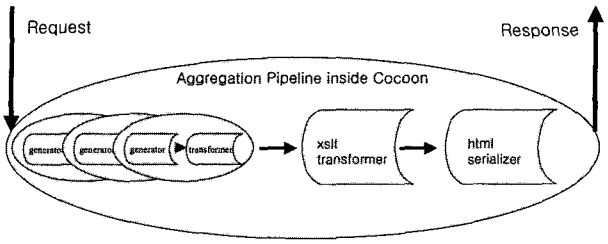

그림 9. Cocoon2의 Aggregation 파이프라인.

Fig. 9. Aggregation Pipeline of Cocoon2.

그림 9에서 Aggregation 파이프라인은 내부에 다른 파이프라인을 포함한다. 이렇게 Aggregation 파이프라인에 포함된 파이프라인들을 이용하여 컨텐츠 컴포넌트를 생성하고, 컨텐츠 컴포넌트들을 XSLT로 작성된 페이지 템플릿을 이용하여 최종 웹페이지로 조합하여 출판한다.

EasyCM의 메뉴는 컨텐츠 각각의 단계를 관리하는 기능들을 중심으로 하여 기반 데이터의 DB 스키마와 데이터를 관리하는 Data 관리, 컨텐츠 컴포넌트를 생성하기 위한 컨텐츠와 템플릿 관리와 컨텐츠와 템플릿을 조합하여 생성된 컨텐츠 컴포넌트를 관리하는 Component 관리, 그리고 컴포넌트들과 페이지 템플릿을 조합하여 페이지를 생성 및 관리하는 Page 관리로 구성된다.

5. 결 론

본 논문에서는 효율적인 동적 웹 컨텐츠 출판을 지원하는 XML 기반의 컨텐츠 관리 시스템인 EasyCM을 개발하였다. EasyCM은 웹 페이지를 컨텐츠 컴포넌트라고 하는 재사용 가능한 부분으로 분할하여 사용자 요청시 페이지 템플릿을 이용하여 페이지로 조립하여 출판함으로써 컨텐츠 재사용을 향상시켰다. 그리고 EasyCM의 출판 성능을 향상시키기 위하여 XML 출판에 필요한 추가적인 처리과정을 전처리한 컨텐츠 컴포넌트를 캐싱함으로써 웹 출판에 필요한 추가적인 처리과정을 제거하였다. 또한 동적 컴포넌트 캐싱의 효율성을 향상시키기 위해 즉시-갱신과 지연-갱신을 사용하였다

EasyCM은 XML 컨텐츠 출판 과정의 각 단계인 기반 데이터, XML 컨텐츠와 XSLT 템플릿, 컨텐츠 컴포넌트, 그리고 최종 웹 페이지의 생성 및 관리 기능을 통하여 사용자에게 자동화된 컨텐츠 관리를 제공한다. 따라서 신속한 웹사이

트 개발 및 개발비용 절감과 개발된 웹사이트의 유지보수 비용을 절감이 기대된다.

참 고 문 헌

[1] J. Challenger, A. Iyengar, and P. Dantzig, "A scalable system for consistently caching dynamic web data", Proceedings of Infocom 99, March 1999.

[2] Apache Cocoon, http://cocoon.apache.org/.

[3] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed. "A publishing system for efficiently creating dynamic web content", Proceedings of INFOCOM 00, March 2000.

[4] H. Zhu and T. Yang, "Class-based cache management for dynamic web content", Proceedings of INFOCOM 01, April 2001.

[5] M. Mikhailov and C. E. Wills, Change and relationship-driven content caching, dstribution and assembly, Technical Report WPI-CS-TR-01-03, Computer Science Department, WPI, March 2001.

[6] Vegard Holmedahl, Ben Smith, and Tao Yang, Distributed Web Server", Proc. of the Seventh IEEE International Symp, on High Performance Distributed Computing, July 1998.

[7] Apache Avalon Project, http://jakarta.apache.org/avalon

[8] Bob Boiko, Content Management Bible, Hungry Minds, 2001.

저 자 소 개

구흥서(Koo, Heung-Seo)

2003년 : 인하대학원 전산학과(이학박사).

2004년~현재 : 청주대학교 컴퓨터정보공학과.

관심분야 : 지능형 데이터베이스, 정보 모델링, 웹 애플리케이션 프레임워크.

Phone : 043) 229-8492

Fax : 043) 229-8432

E-mail : hskoo@cju.ac.kr