제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

URL 패턴을 이용한 동적 웹 컨텐트 출판 시스템의 설계

조재호, 구흥서
청주대학교 컴퓨터정보공학과
e-mail:{pm1200, hskoo}@chongju.ac.kr

Design of Dynamic Web Content Publishing System using URL Pattern

Jae-Ho Cho, Heung-Seo Koo
Dept. of Computer and Information Engineering,
Chongju University

요 약

기업의 웹사이트가 비즈니스의 중요한 부분으로 정착되면서 급격히 증가한 웹사이트의 컨텐트를 관리하기 위한 CMS의 필요성이 급증하였다. 그러나 CMS의 출판은 기존의 출판 방식과 비교하여 추가적인 처리과정으로 인한 성능 저하의 문제점이 있다. 본 논문에서는 XML 기반 CMS의 효율적인 동적 웹 컨텐트 출판을 지원하기 위하여 XML 기반의 CMS의 출판에서 필요한 컨텐트 저장소로부터 XML 객체의 추출과 XSLT를 이용한 HTML 변환의 추가적인 처리과정을 전처리하여 컨텐트 컴포넌트를 생성 및 캐싱함으로써 효율적인 출판을 지원하고, 동적 컨텐트 캐싱의 효율성을 높이기 위하여 URL 패턴을 이용한 페이지와 컴포넌트의 그룹화 관리를 지원하는 출판 시스템을 설계하였다.

1. 서론

오늘날 웹사이트가 기업 비즈니스의 중요한 부분으로 정착되면서 웹 컨텐트의 양적 증가와 다양화로 인하여 기존의 수작업 방식의 컨텐트 관리는 비효율성과 일관성 결여 문제가 나타날 수 있다. 그러므로 최근에는 컨텐트의 생성, 관리, 그리고 출판하는 일련의 과정을 자동화하는 CMS(Content Management System)의 필요성이 급증하고 있다. 대부분의 CMS는 데이터베이스와 같은 컨텐트 저장소(repository)에 저장된 동적 컨텐트를 템플릿(template)을 이용하여 출판한다. 따라서 CMS를 이용한 웹 컨텐트의 출판은 정적 출판 방식에 비하여 컨텐트 저장소로부터 컨텐트를 추출하기 위한 컨텐트 저장소 접근 및 검색과 템플릿을 이용한 컨텐트 변환의 추가적인 처리과정이 필요하기 때문에 컨텐트의 출판을 위한 서버의 부하와 지연시간을 가중시킨다. 이러한 CMS 출판의 문제점을 해결하기 위하여 대부분의 CMS가 캐싱 시스템(caching system)을 지원하고 있지만, 일반적으로 기존의 캐싱 시스템은 정적 컨텐트를 대상으로 하고 있기 때문에 동적 컨텐트가 증가하면 시스템의 성능이 저하된다[2].

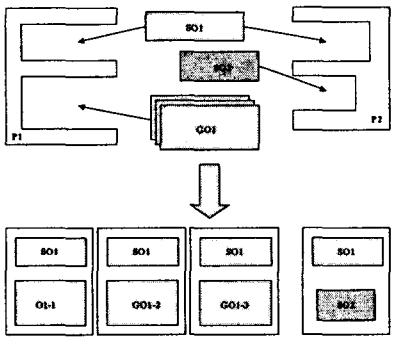
본 논문에서는 XML(eXtensible Markup Language) 기반의 CMS에서 효율적인 동적 컨텐트 출판을 지원하는 출판 시스템을 설계하였다. 본 논문에서 제안한 출판 시스템은 컨텐트 저장소로부터 XML 객체의 추출과정과 XML과 XSLT(eXtensible Stylesheet Language Transformation) 조합하여 HTML로 변환하는 과정을 전처리(pre-computing)하여 컨텐트 컴포넌트를 생성한다. 이러한 컨텐트 컴포넌트를 캐싱하고, 사용자 요청시 조립하여 출판함으로써 효율적으로 XML 기반의 동적 컨텐트 출판을 지원한다. 또한, 동적 컨텐트 캐싱시 요구되는 컨텐트의 일관성 유지를 위해서 필요한 XML 객체와 컨텐트 컴포넌트간의 종속정보를 효율적으로 관리하기 위하여 URL 패턴을 이용한 페이지와 컴포넌트의 그룹화 관리를 제안한다.

본 연구는 2002년 청주대학교 산·학·연 공동기술개발 컨소시엄 지원에 의한 것임.

2. 동적 웹 컨텐트 출판

2.1 컨텐트 재사용

대부분의 동적 웹 페이지는 다른 페이지와 공통으로 포함하는 메뉴, 헤더와 같은 공통 컨텐트가 존재한다. 따라서 페이지를 좀더 간단한 객체로 분할하면, (그림 1)과 같은 컨텐트의 재사용을 통하여 좀더 효율적인 동적 컨텐트 출판을 가능하게 한다.



(그림 1) 컨텐트 재사용

2.2 동적 컨텐트 캐싱

동적 컨텐트 캐싱의 가장 큰 이슈중의 하나는 기반 데이터(underlying data)와의 일관성 유지이다. 동적 컨텐트는 기반 데이터의 변경이 어떤 캐싱된 페이지에 영향을 미치는지 식별하기 어렵기 때문에 낡은 정보를 제공하거나 최신 정보를 갱신할 가능성이 있다. IBM의 1998년 동계 올림픽 사이트에서는 웹 페이지를 객체로 분할하고 각 객체와 기반 데이터와의 종속 관계를 ODG(Object Dependency Graph)를 이용한 DUP(Data Update Propagation) 알고리즘으로 점진적으로 갱신함으로써 일관성 문제를 해결하였다[2].

2.3 페이지 그룹화

DUP[2]와 같은 일관성 유지 기법들은 개별적인 페이지별로 기반 데이터와의 종속 정보를 유지해야 하기 때문에 컨텐트의 양이 증가하면 종속 정보가 급격히 증가하므로 종속 정보를 이용한 캐싱된 컨텐트의 관리가 어려워진다. 이러한 문제점을 해결하기 위하여 공통 URL 패턴을 이용한 페이지 그룹화를 통하여 종속 정보를 공유함으로써 캐싱의 효율성을 향상시킬 수 있다[4]. 그러나 페이지 그룹화는 페이지 단위로 종속정보를 유지하기 때문에 컨텐트 재사용성이 떨어지는 문제점이 있다.

3. 출판 시스템 설계

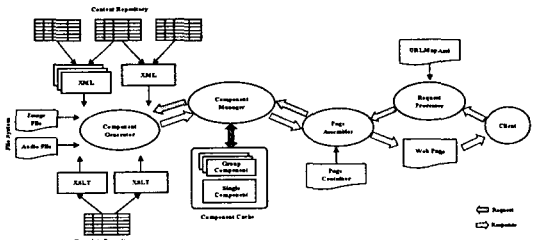
3.1 설계시 고려사항

현재 CMS의 구조에 관하여 구체적으로 합의된 표준은 아직 없지만 일반적으로 수집(collection) 시스템, 관리(management) 시스템, 그리고 출판(publishing) 시스템으로 나눌 수 있다[1]. 본 논문에서는 CMS를 위한 출판 시스템을 설계하였다.

본 논문에서 제안한 출판 시스템은 컨텐트와 스타일을 분리하기 위하여 XML을 기반 데이터 형식으로 사용하는 CMS를 대상으로 하였다. 따라서 기반 데이터로부터 XML 객체를 추출하는 과정과 XML 객체와 XSLT를 조합하여 HTML 형태로 변환하는 추가적인 과정이 필요하다. 이러한 추가적인 처리과정으로 인한 성능 저하의 문제점을 해결하고 컨텐트의 재사용성을 높이기 위하여 HTML 형태로 전처리된 컨텐트 컴포넌트를 이용하여 페이지를 분할하였다. 분할된 컨텐트 컴포넌트를 컴포넌트 캐시에 캐싱하고 페이지 요청시 필요한 컴포넌트를 조립하여 제공함으로써 동적 컨텐트 제공의 서버 부하와 지연 시간을 감소시킨다. 또한 URL 패턴을 이용하여 페이지와 컴포넌트를 그룹화하여 관리함으로써 동적 웹 페이지와 컴포넌트들의 효율적인 관리를 지원한다.

3.2 시스템 구조

동적 웹 컨텐트 출판 시스템의 구조는 (그림 2)와 같이 컴포넌트 생성기, 컴포넌트 관리자, 페이지 조립기, 그리고 요청 처리기의 4개의 주요 구성요소를 중심으로 컨텐트와 템플릿을 저장하는 저장소, 컴포넌트의 사본을 저장하는 컴포넌트 캐시, 컴포넌트로 조립하여 페이지를 생성하기 위한 페이지 컨테이너, 그리고 URL 패턴을 정의하는 URLMap.xml로 구성되어 있다.



(그림 2) 출판 시스템의 구조도

3.3 시스템 구성요소

이번 절에서는 동적 웹 컨텐트 출판 시스템의 4가지 주요 구성요소에 대해서 설명한다.

1) 요청 처리기

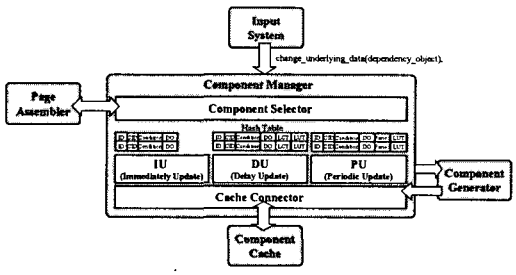
요청 처리기(request processor)는 URL 패턴을 이용한 동적 페이지의 그룹화를 지원하기 위하여 사용자가 요청한 URL을 파싱한 다음, URLMap.xml에서 정의한 URL 패턴과 대응되는 페이지 컨테이너를 페이지 조립기로 전송한다.

2) 페이지 조립기

페이지 조립기(page assembler)는 먼저 요청 처리기로부터 전송된 페이지 컨테이너를 파싱하여 페이지의 구성 컴포넌트 ID와 매개변수를 컴포넌트 관리자로 전송하여 해당 컴포넌트를 요청한다. 컴포넌트 관리자로부터 반환된 컴포넌트들을 조립하여 최종 페이지를 생성한다.

3) 컴포넌트 관리자

컴포넌트 관리자(component manager)는 페이지 조립기에서 요청한 컴포넌트 ID와 매개변수를 이용하여 해당 컴포넌트를 컴포넌트 캐시로부터 검색하여 반환한다. (그림 3)은 컴포넌트 관리자의 내부 구조를 나타낸다.



(그림 3) 컴포넌트 관리자의 내부 구조

컴포넌트 관리자는 CMS에서 전송하는 (그림 4)와 같은 CCM(Component Control Message)을 이용하여 컴포넌트 캐시에 저장된 컴포넌트의 생성, 갱신, 삭제 등의 관리 작업을 수행한다. 관리 작업 중에서 컴포넌트의 생성 및 갱신 작업은 컴포넌트 생성기에 dependency_object와 template_ID를 전송하여 컴포넌트의 생성 및 갱신을 요청하고 반환된 컴포넌트를 컴포넌트 캐시에 추가 및 교체함으로써 이루어진다.

```
• create_group_component(component_ID, template_ID, dependency_object, period);
• create_single_component(component_ID, template_ID, dependency_object, period, condition);
• delete_component(component_ID);
• add_group_component(component_ID, condition);
• delete_group_component(component_ID, condition);
• change_underlying_data(dependency_object, condition);
```

(그림 4) CCM(Component Control Message)

4) 컴포넌트 생성기

컴포넌트 생성기(component generator)는 컴포넌트 관리자로부터 전송된 dependency_object와 template_ID 정보를 이용하여 CMS의 컨텐트 저장소로부터 XML 객체를 생성하고, 템플릿 저장소로부터 XSLT 객체를 추출하여 Apache의 Xalan[6] XSLT 처리기를 이용하여 HTML 형태의 컴포넌트를 생성하여 컴포넌트 관리자로 반환한다.

4. URL 패턴을 이용한 페이지와 컴포넌트 그룹화

대부분의 동적 페이지는 동일 스크립트를 이용하여 서로 다른 조건 매개변수 또는 네트워크 경로명에 따라 동일 스타일과 구조로 상이한 컨텐트를 제공한다. 하지만 기존의 출판 시스템에서 동적 페이지들은 일단 출판되어 캐싱되면 개별적인 페이지로 인식하기 때문에 일관성 유지를 위한 기반 데이터와의 종속 정보를 페이지별로 관리한다. 따라서 페이지수가 증가하면 캐싱된 컨텐트 관리의 효율성이 급격히 떨어지는 문제점이 있다. 이러한 문제점을 해결하기 위하여 본 논문에서는 URL 패턴을 이용한 페이지와 컴포넌트의 그룹화 관리를 제안한다.

4.1 페이지 그룹화

URL 패턴을 이용한 페이지 그룹화는 그룹화된 페이지의 스타일과 구조 정보를 하나의 페이지 컨테이너 파일에 정의함으로써 저장 공간을 절약하고, 페이지 전체의 스타일과 구조의 변경 작업에서 일관성을 보장한다.

URLMap.xml은 URL 패턴을 정의하기 위한 XML 형식의 설정 파일이다. 다음 (그림 5)는 온라인 서점 사이트 URL 패턴의 일부를 작성한 예제이다. (그림 5)에서 도서 정보 페이지들이 공유하는 공통 URL과 와일드 카드로 표현된 조건 매개변수를 조합하여 "*/book/infor/book-detail.html?isbn=*"과 같은 URL 패턴을 작성할 수 있다. 이 URL 패턴에 부합하는 URL은 모두 "book-infor/book-detail.htm"의 페이지 컨테이너로 매핑함으로써 모든 도서 정보

페이지들의 요청을 단일 페이지 컨테이너로 집중시킨다.

```
<URL-match pattern="*/book/info/book-detail.html?isbn=*">
    <page-container src="book-info/book-detail.html"
     type="group"/>
    <parameter type="String" name="class" value="{1}"/>
    <parameter type="String" name="ISBN" value="{2}"/>
</URL-match>
```

(그림 5) URLMap.xml의 예

페이지 컨테이너는 페이지의 레이아웃을 정의하는 일종의 템플릿 파일이다. 페이지 컨테이너는 (그림 6)과 같이 작성하며, <Insert-Component>라는 확장 태그를 사용하여 구성 컴포넌트의 위치와 속성들을 정의한다.

```
<html>
 <body>
  <Insert-Component ID="header" Type="Single" Change="Static">
   <template ID="header"/>
  </Insert-Component>
  <table><tr>
    <td><table><tr>
  <Insert-Component ID="field-search" Type="Group" Change="RSt">
    <template ID="sidebar">
    <parameter name="parameter11.name" value="parameter11.value"/>
  </Insert-Component>
      </tr></table></td>
                    . . .
  </body>
</html>
```

(그림 6) 페이지 컨테이너 book-detail.html의 예

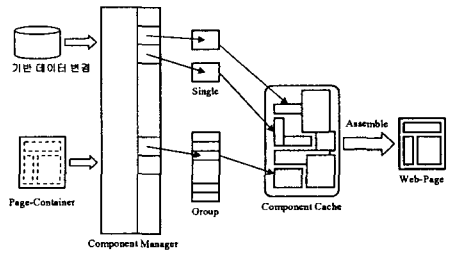
그룹화된 페이지의 레이아웃이나 구성요소가 변경되면 각각의 페이지를 변경하지 않고 하나의 페이지 컨테이너 파일만 변경하면 그룹화된 모든 페이지에 적용되므로 묵시적으로 일관성 유지가 보장된다.

4.2 컴포넌트 그룹화

컴포넌트의 그룹화는 일관성 유지를 위한 종속 정보를 공유하여 저장 공간을 절약하고, 그룹 컴포넌트의 일괄적인 변경작업을 지원한다.

(그림 6)과 같은 페이지 컨테이너에서 컴포넌트의 Type 속성이 "Group"이면 컴포넌트가 그룹 컴포넌트임을 나타낸다. 그룹 컴포넌트는 해시 테이블에 일관성 유지를 위한 종속 정보인 dependency_object와 ID를 공유하고, 각각의 그룹 컴포넌트는 개별적인 해시 테이블에 그룹 컴포넌트 중에서 개별적인 컴포넌트를 식별하기 위한 condition과 컴포넌트 캐시에 저장된 컴포넌트의 인덱스인 CID를 유지한다. 다음 (그림 7)은 해시 테이블을 이용한 컴포넌트의 처리 과정을 나타낸다.

컴포넌트의 그룹화 관리는 CCM을 통하여 일괄적으로 관리된다. 예를 들어, 만약 컴포넌트의 스타일이 변경되어 그룹 컴포넌트 전체를 갱신해야 하면 change_underlying_data(object_dependency, condition); 메시지에서 condition 값을 "all"로 하여 컴포넌트 전체를 일괄적으로 갱신할 수 있다.



(그림 7) 컨텐트 컴포넌트의 처리과정

5. 결론 및 향후연구과제

본 논문에서는 XML 기반의 CMS의 효율적인 동적 웹 컨텐트 출판을 지원하는 출판 시스템을 설계하였다. 설계한 출판 시스템은 XML 출판에 필요한 추가적인 처리과정을 전처리하여 생성한 컨텐트 컴포넌트를 캐싱하고, 효율적인 동적 컨텐트 캐싱을 위하여 URL 패턴을 이용한 페이지와 컴포넌트의 그룹화 관리를 지원한다. 따라서 CMS의 특성상 나타나는 출판의 성능 저하와 동적 컨텐트 증가로 인한 캐싱의 성능 저하를 해결할 수 있을 것으로 기대된다. 향후연구과제로는 설계한 출판 시스템을 구현하고 기존의 출판 시스템과의 성능평가를 수행하는 것이다.

참고문헌

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

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

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

[4] H. Zhu and T. Yang, Class-based cache management for dynamic web content, In *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] Xalan XSLT Processor, http://xml.apache.org/xalan-j.