"빨리, 더 빨리…" 전자상거래 성공의 화두

전자상거래의 전장에서 살아남기 위해서는 기존 데이터 소스를 빠르고 안정적으로 활용할 수 있는 능력이 필요하다. 이런 필요성을 충족시키는 XML 데이터 서버의 흥미로운 옵션들을 살펴보자.
-편집자-

배심원단은 여전히 귀추를 주목하고 있지만 초기 보고서는 그다지 좋은 점수를 얻지 못한 것 같다. 바로 전자상거래 이야기다. 만일 그 배심원단이 고객이라면 점수는 더욱 낮아질 것이다. 작년 12월 31일자 타임지는 전자상거래에 대해 다음과 같이 일축하였다.

성탄 시즌의 마지막 날은 배달되지 못한 산더미 같은 패키지들과 고객들의 불평 불만으로 점철되어 버렸다. 시스템은 사용자들의 폭주로 인해 마비되었으며, 그 결과 주문서는 어디론가 행방불명 되었으며, 재고는 갑자기 바닥이 나 버리거나 창고 천장에 닿을 정도로 수두룩하게 쌓인 상태로 남겨지기도 하였다.

기업과 비즈니스를 생각할 때 확실히 지금의 상황은 전자상거래를 빼놓고 이야기할 수 없게 되었다. 모든 기업들이 전자상거래에 뛰어들고 있지만, 이것만으로는 충분치 않다. 그것은 돌아가야만 하며, 그것도 잘 돌아가야만 한다. 기업과 소비자간 거래이든 기업과 기업간 거래이든 상관없이 고객의 위치에 서게 되는 모든 소비자들은 자신들의 기대에 부응하지 못하는 서비스는 외면할 것이다.

아직 논란 많은 XML

그렇다면 전자상거래 영역에서 기업이 모든 잠재력을 발휘하기 위해 무엇이 필요한가? 아마도 이 질문에 대한 답변 중 하나로서 확장 마크업 언어인 XML을 들 수 있겠다. 당신은 XML을 범용 데이터 커뮤니케이션 포맷, HTML보다 더욱 향상된 표시 포맷, 데이터 저장 포맷 등과 같은 형식으로 다양하게 사용할 수 있다.

XML은 당당히 기술 사전의 한 자리를 차지하겠지만, 아직 논쟁이 많은 기술중 하나이기도 하다. XML을 이용한 새로운 제품의 출시 소식은 하루가 멀다하고 매일같이 들려온다. 하지만 구매에 앞서 이러한 제품들의 비교작업 조차 매우 힘든 것이 현실이다.

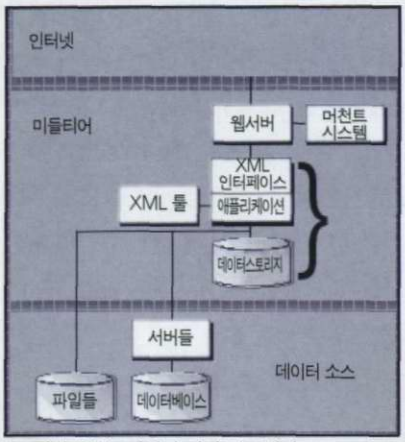
이 기사는 XML 기술 - XML 데이터 서버 - 이 가지는 비즈니스적, 기술적 가치의 일부 단면들을 살펴보려 하며, 나아가 XML 기반의 전자상거래 애플리케이션을 위한 기초적인 인프라스트럭처 구축에 이 기술이 어떻게 사용되는가 알아볼 것이다.

XML 데이터 서버를 사용한다는 것은 전략적 의도에 기초한다. 당신은 XML 데이터 서버를 사용해 웹 사이트와 전자상거래 애플리케이션을 보다 신속히 개발할 수 있으며, 사이트 자체의 응답 시간도 단축시킬 수 있다. 결국 핵심적인 요소는 속도이다. 고객은 느려터진, 때로는 응답조차 없는 전자상거래 사이트는 거들떠 보지도 않을 것이다. 따라서 가상의 전자공간에서 무언가를 하기 위해서는 기본이 속도이다.

서버 응답에 관한 문제점을 해결할 수 있는 가장 확실한 방법은 보다 빠른 하드웨어 플랫폼을 사용하는 것이다. 하지만 이 같은 하드웨어 솔루션은 너무 많은 비용이 들어가며 모든 문제를 해결할 수도 없다.

사이트 속도를 빠르게

그렇다면 대안은 무엇인가? 바로 보다 빠른 소프트웨어이다. XML 기술 - 더 정확히 말해서 XML 데이터 서버 - XML 인터페이스와 더불어 웹 사이트의 응답 속도를 보다 빠르게 할 수 있는 소프트웨어 아키텍처를 제공한다.



〈그림1〉 XML 데이터 서버 아키텍처.

XML 데이터 서버 아키텍처는 데이터 저장, 데이터 조정을 위한 애플리케이션 레이어, 중간 계층(middle tier) 상의 XML 데이터 표시를 지원한다 〈그림 1. 참조〉 XML 데이터 서버를 사용하는 것은 카탈로그를 출판하는 것과 흡사하지만 인쇄 대신에 XML 데이터 서버를 사용한다.

데이터 서버는 카탈로그 데이터가 고객이 원하는 데이터 포맷 - 다양한 종류를 지원하며, DTD(document type definition: 문서 원형 정의)도 지원함 - 으로 표시되도록 조작하며, 고객이 원할 경우 일반 HTML 형식으로도 데이터를 제공할 수 있다.

XML 데이터 서버 아키텍처는 세 개의 형태로 분류된다. 첫 번째는 독립형, 기존 소스로부터 데이터를 사용하는 형태, 멀티사이트형. 여기서 필자는 XML 데이터 서버가 가장 많이 사용되는 두 번째 형태를 설명하겠다.

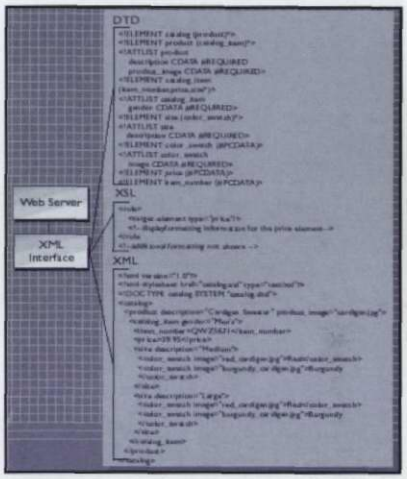
만약 당신의 기업이 제품이나 서비스에 관한 정보를 포함하는 데이터 소스를 가지고 있다면 이 두 번째 형태의 아키텍처를 사용하는 것이 좋다. 이러한 소스에 포함된 데이터는 보통 파일 시스템이나 DBMS에 저장되어 있다. 〈그림 1〉에서 볼 수 있듯이 중간 계층에 XML 데이터를 추가함으로써 이미지, 그래픽 이것들은 개발자의 PC와 같은 전혀 다른 시스템에 저장되어 있을 것이다.

XML 서버는 최하단에 위치

이 경우 각종 애플리케이션이 현재의 데이터 포맷에 기초하고 있기 때문에 기존 데이터를 XML 포맷으로 변환하는 것은 무의미하다. 또한 인터넷 쿼리는 데이터 프로세싱 작업에 문제를 일으킬 소지가 있기 때문에 기존 데이터 소스를 인터넷 쿼리에 노출시키는 것도 바람직하지 못하다.

〈그림 1〉을 주의해서 보면 알 수 있듯이 XML 데이터 서버는 웹 서버의 아랫쪽에 위치한다. 만약 기업이 온라인 결재방식을 채택하고 있다면 웹 서버는 대개 판매 시스템(merchant system)과 연결된다.

이러한 소프트웨어 아키텍처에서 중간 계층에 위치한 XML 인터페이스는 데이터 표시를 담당하는데, 이것은 DTD와 XSL(extensible style language)을 포함한 모든 정보를 웹 서버에 표시한다. 문서의 논리구조를 정의하고 문서 종류에 대한 문법을 제공하는 DTD는 컴퓨터를 통해 작성되는 전자 문서에서 문서의 구조, 레이아웃, 논리적 연결에 관련된 정보를 표시하기 위해 사용되는 표식인 태그를 정의한다.



〈그림 2〉 자체 정의된 DTD와 XSL 사용하기.

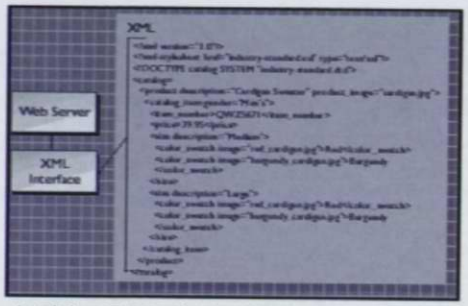
XSL은 XML 문서를 위한 스타일 시트(디스플레이 장치에 DTD요소와 그 컨텐츠를 포맷하는 방법을 알려주는 명령어 리스트)를 기술하는 언어로서 폰트 크기와 종류, 간격, 기타 디자인 사항과 같은 정보를 포함하고 있다. 결국 최종 표시는 웹 서버와 XML, DTD, XSL간의 상호연계에 의해 결정된다〈그림 2. 참조〉.

XML을 이용한 확실한 옵션은 모든 스펙을 제어할 수 있는 폐쇄형 시스템을 만드는 것이다. 이 시스템은 DTD와 XML 데이터, 그리고 문서 표시를 일일이 지정하는 XSL에 대한 모든 제어를 가능케 한다.

폐쇄형 시스템과 관련된 단점은 인터넷상에서 급증하고 있는 자동 툴 - 이 툴에는 각종 검색 엔진과 쇼핑과 제품 비교에 이용되는 다양한 보츠(bots)등이 포함됨 - 이 데이터의 의미를 이해하지 못하다는 점에 있다.

모든 문서 스펙 제어 가능

예를 들어 XML은 일련의 이름 값과 태그가 한 쌍으로 구성된 언어 구조를 가지고 있는데, 〈제품 명칭=브라운 스웨터〉에서 제품 명칭은 태그이며 브라운 스웨터는 값이다. 이 경우 표준 이름 값을 사용하는 시스템에서는 제품 명칭 태그를 제품이라는 태그



〈그림3〉 표준 DTD와 XSL 사용하기

로 인식한다. 이 때문에 일부 업계 그룹에서는 XML 태그의 의미와 이름에 대한 합의점을 끌어내기 위해 연구하는 중이다.

만약 당신이 합의된 태그를 사용하고자 한다면 업계 표준이나 관련된 모든 그룹이 동의하는 DTD와 XSL을 사용하기 바란다. 당신이 이러한 표준 스펙을 사용하게 되면, 정보 표시에 필요한 것은 오직 XML 파일 뿐이다〈그림 3 참조〉).

〈그림 2〉와 〈그림 3〉를 비교해보면 알 수 있듯이, 표준 DTD를 사용하게 되면 작업 프로세스가 굉장히 간단해진다. 언뜻 보면 표준 DTD는 그림 2의 하단과 비슷해 보이지만, 분명한 구분의 기준은 관련 그룹들이 태그의 이름과 의미에 동의하고 있는지 여부이다.

확실히 업계 표준의 DTD를 채택하는 것이 가장 무리없는 방법인 것 같다. 그러나 표준 DTD 하나만으로 충분치 못할 때가 있다. 만약 하나 이상의 표준 DTD가 존재한다면 어떻게 될 것인가? 당신이 자체 정의된 DTD를 표준 DTD에 추가한다면 어떻게 될 것인가?

이를테면 그림 3의 경우, 〈color_swatch〉 태그는 표준 DTD에서 표시되지 않기 때문에 〈color〉 태그를 사용해야만 한다. 또한 표준 〈color〉 태그는 JPEG 파일을 표시하는 image 속성을 갖지 못할 수도 있다.

'업계표준 · 자체 DTD' 동시 사용

따라서 그 대안으로 업계 표준용 DTD와 자체 정의된 DTD를 같이 사용하는 것이다. 즉 표준 DTD가 지원하지 못하는 기능이 있을 경우, 자체 정의된 DTD를 사용하여 정보를 표시하는 것이다.

그렇게 되면 〈Color〉 태그가 지원하지 못하는 이미지 표시를 위해 〈color swatch〉 태그를 사용할 수 있다. 같은 결과를 보여줄 수 있는 방법으로서 표시 포맷과 저장 포맷을 분리시키는 방법이 있으며, 여기서 저장 형식은 DBMS를 통해 사용할 수 있다.

이제 애플리케이션을 살펴볼 차례이다. XML 데이터의 사용을 위해서 데이터 표시와 더불어 애플리케이션 레이어가 동작하게 되는데, XML을 사용하는 대다수 애플리케이션들은 객체 지향 프로그래밍 언어인 자바나 C++로 작성된다. 따라서 애플리케이션 레이어에서 객체를 사용하는 방법을 선택하는 것이 중요하다.

애플리케이션 레이어에서 객체를 사용하는 방법에는 다양한 옵션이 존재하는데, 개인이 개발한 간단한 자체 프로그램에서부터 CORBA, COM+/MTS나 EJB(Enterprise JavaBean)와 같은 복잡한 애플리케이션에 이르는 선택 범위를 가지게 된다. 하지만 기사의 주제상 애플리케이션 객체만을 고려할 것이며 다양한 객체 모델은 논의의 대상에서 제외될 것이다

애플리케이션 객체에서 XML을 사용하는 한 가지 방법은 고유 포맷으로 객체를 직접 저장하는 것이다. 따라서 이 방법은 XML의 변환 작업이 필요 없어진다. XML을 직접 사용함으로써 얻는 이점은 매핑이 필요 없다는 점이고, 단점은 XML 데이터가 분리 저장되기 때문에 객체 모델이 손상을 입을 수 있다는 점이다.

하지만 이러한 단점에도 불구하고 만일 XML 데이터를 객체에서 캡슐화하는 작업을 필요로 하지 않는다면 XML을 객체로부터 분리하는 것은 타당한 귀결임에 틀림없다.

그러나 XML 데이터를 애플리케이션 객체에서 캡슐화할 필요가 있다면 이 같은 방법은 도움이 되지 않는다. XML 데이터의 캡슐화는 객체 모델과 데이터를 연결시키며 결국 데이터 조정은 보다 쉬워진다.

이런 경우 XML은 정보 표시와 교류에만 사용될 것이다. 이러한 방법이 제공하는 장점은 객체 모델이 손상을 입지 않고 원래 모습 그대로 보존된다는 점이며, 단점은 XML이 애플리케이션과 인터페이스 레이어 사이에서 매핑 작업을 거쳐야 한다는 점이다. 하지만 자바와 C++같은 프로그래밍 언어에서 XML과 객체 모델은 유사성을 가지기 때문에 매핑과 관련된 문제점은 그다지 심각하지 않다.

다양한 저장방법의 존재

XML 데이터의 저장 방법은 세 가지가 있다: 파일 시스템, DBMS, 원시 저장. 이 기사에서는 DBMS가 가지는 범용성 - 더욱이 DBMS는 안전한 멀티유저 액세스와 업데이트를 보장하는데 필요한 ACID 특성(원자성, 일관성, 독립성, 지속성)을 제공하기까지 한다 - 때문에 이것만을 다루도록 하겠다. 여기서 알아볼

DBMS는 세 가지로 분류된다: RDBMS, ODBMS, OR 매핑 기반의 DBMS.

RDBMS 기반의 데이터 서버. RDBMS 기반의 데이터 서버는 웹 사이트가 크고 고성능 쿼리와 업데이트 작업이 필요한 경우에 파일 기반의 시스템보다 훨씬 뛰어난 성능을 제공한다. 〈그림 1〉에서 보듯이 데이터 서버는 중간 계층에서 RDBMS를 사용한다.

기존 데이터베이스나 파일시스템의 데이터는 RDBMS 포맷으로 변환되어 저장된다. 모든 쿼리와 업데이트 작업은 기존 데이터 소스가 아닌 RDBMS를 중심으로 직접 이루어진다. 업데이트 작업의 경우 일정한 간격을 두고 기존 데이터 소스에 데이터를 쓰는 방법과 한가한 시간에 몰아서 쓰는 방법, 두 가지가 있다.

표준 스펙 사용하면 간단해져

XML 데이터를 RDBMS와 같이 사용할 때 유의해야 할 사항이 중간계층의 관계형 모델과 XML 모델의 차이점이다. 알다시피 XML 모델은 객체 모델인 반면, RDBMS는 관계형 모델이다. 이 차이점은 결국 임피던스 불일치를 일으킬 수 있으며 상당량의 매핑 레이어를 요하게 된다 매핑 레이어가 클수록 개발 시간은 오래 걸리며 성능도 저하된다 .

ODBMS 기반의 데이터 서버. 동일한 데이터베이스 관리 원리가 RDBMS와 ODBMS 기반의 데이터 서버에 적용된다. 두 시스템간 중요한 차이점은 XML 모델이 객체 모델과 유사한 반면 관계형 모델은 그렇지 않다는 점이다.

ODBMS 기반의 데이터 서버는 한마디로 뛰어난 성능을 필요로 하고 상당한 작업 불륨이 예상되는 복잡한 대형 웹 사이트에 적합하다. 그 이유는 XML 모델과 객체 모델간의 유사성 덕택에 RDBMS 기반의 데이터 서버에 비해 보다 나은 성능이 약속되기 때문이다.

데이터 서버는 중간 계층에서 ODBMS를 사용하며, 기존 데이터베이스나 파일 시스템의 데이터는 ODBMS가 사용하는 포맷으로 변환된다. ODBMS 데이터 서버에서는 객체가 애플리케이션과 데이터 저장, 양쪽 모두에 사용되므로 애플리케이션 레이어와 데이터 저장 사이에 매핑 레이어가 필요없게 된다.

애플리케이션이 필요로 하는 객체는 ODBMS에 직접 저장된다. 결국 매핑 레이어의 부재로 인해 RDBMS 기반의 데이터 서버에 비해 훨씬 뛰어난 성능을 제공할 수 있게 된다. 또한 모든 ODBMS 기반의 데이터 서버는 객체 캐쉬처리 기능을 제공하여 동일한 객체를 자주 반복 사용하는 웹 사이트의 반응 속도를 더욱 빠르게 만든다.

쿼리와 업데이트 작업은 RDBMS와 마찬가지로 기존 데이터 소스가 아닌 ODBMS를 중심으로 직접 이루어지며, 업데이트 방법도 RDBMS와 똑같이 일정한 간격을 두고 쓰는 방법과 한가로운 시간에 몰아서 쓰는 방법, 두 가지가 있다.

OR(object/relational) 기반의 데이터 서버. OR 기반의 데이터 서버는 데이터가 중간 계층에 저장되지 않는다는 점에서 위의 두 가지 데이터 서버와 구분된다. 위의 두 데이터 서버에서는 변환 작업이 배치 기반에서 일어나지만, OR 데이터 서버에서는 모든 변환 작업이 하위 계층에 위치하는 데이터베이스나 파일 시스템에서 이루어진다.

이러한 방식은 파일 시스템이나 데이터베이스에서 변환된 모든 정보를 인터넷 사용자가 즉시 이용할 수 있다는 장점을 제공한다. 하지만 XML 구조로 데이터를 변환하는 작업이 실시간으로 이뤄져야 하므로 리소스와 시간의 관점에서 볼 때 성능 저하가 필연적으로 예상된다.

대형 사이트에 적합한 ODBMS

그리고 RDBMS와 마찬가지로 임피던스 불일치 현상이 일어날 수 있지만, 캐쉬 기능을 지원하기 때문에 캐쉬 처리된 데이터가 비교적 자주 읽혀질 경우 불일치 현상이 상당량 줄어들게 된다. 만일 동일한 데이터 아이템이 10-100번 정도 읽혀진다면, 이러한 OR 기반의 데이터 서버의 성능은 ODBMS 기반의 데이터 서버의 성능에 육박할 수 있다.

현재 대부분의 OR 기반 데이터 서버들이 ODBMS의 객체 캐쉬 기능과 유사한 캐쉬 기능을 지원하고 있다. 그렇지만 문제는 캐쉬 동기화이다. 만약 데이터가 하위 계층의 RDBMS에서 변경된다면 캐쉬 처리된 데이터는 쓸모 없게 된다. 따라서 캐쉬 동기화 문제는 하위 계층의 RDBMS에서 발생하는 변경 사항에 대해 캐쉬 처리된 데이터를 항상 최신으로 유지해 해결할 수 있다.

많은 옵션들이 존재함에도 불구하고 당신의 고객들이 원하는 기대치에 부응하기 위해 어떤 것을 정확히 선택해야 하는 문제는 여전히 힘든 사안이다. 기존 데이터 소스를 이용할 수 있으면서도 웹 사이트의 반응 속도를 보다 빠르게 할 수 있는 방법을 찾고 있는 사람이 있다면 여기에 소개된 정보들이 그에게 적절한 참고 자료가 되기를 바란다. 두 마리 토끼를 모두 좇고 있는 사냥꾼이라면 반드시 XML 데이터 서버에 관해 여기 소개된 기능들을 유념해야 할 것이다.