XML,인터넷시대 표준언어로 자리잡아

XML 구조는 기존의 HTML의 단점을 보완한 SGML의 복잡함을 보완하고 호환성이 있다는 점에서 주목을 끌고 있다. SGML은 마크업 언어의 표준으로 등장했지만 규모가 커서 처리해야 할 사항이 너무 많았다. 그리고 HTML은 단순하기는 하지만 사용자가 원하는 모든 작업을 수행하기에는 충분하지 못했다.
HTML에 새로운 태그, JavaScript, VBScript와 같은 스크립트 언어들을 추가함으로서 이러한 한계를 극복하려는 시도도 있었다. HTML은 이러한 확장은 가능했지만, 너무 많은 회사들이 난립해 웹상에서 혼란만을 초래하게 되었다. 이에 XML 구조의 장점과 필요성에 대해서 짚어 보았다. 〈편집자〉

김종민 (주)건지소프트 프로그래머

1. HTML, SGML, 그리고 XML

HTML, SGML 그리고 XML은 모두 ML이라는 단어를 포함하고 있다. ML이란 Markup Language의 약자로서 문서의 구조나 외양을 나타내기 위해서 사용되는 부가정보를 의미한다. 쉽게 이야기하면 문서의 글꼴, 글자크기, 진하게, 이탤릭체, 여백 .. 이러한 속성들을 설정해주는 것들을 마크업이라 한다. 실제로 우리는 한글이나 MS-Word와 같은 워드 프로세서를 사용할 때 많은 마크업을 쓰게 된다. 즉, 한글에서 글자의 크기, 글꼴, 페이지 여백 등을 설정하는 것을 모두 마크업이라고 보면 된다.

그러나 XML이나 SGML, HTML에서 사용되는 마크업은 위에서 설명한 마크업과는 차이가 있다. 워드프로세서와 같은 프로그램에서 사용되는 마크업은 주로 문서의 외양을 나타내기 위해서 사용된다고 할 수 있는 반면 XML, SGML, HTML에서 마크업은 문서에 구조적인 의미를 부여하기 위해 사용된다. 외양을 나타내기 위해 사용되는 대부분의 마크업은 특정 시스템이나 특정 S/W에 종속되어 있기 마련이다.

즉, 한글에서 편집한 것을 워드에서 불러올 경우 아직도 완벽하게 포맷을 바꾸어주지 못하는 것이 바로 특정 회사에 종속되어 있기 때문인 것이다. 그래서 1986년 국제 표준으로서 SGML(Standard Generalized Markup Language)가 등장하게 되었다. 여기에서 일반화된 표준(Standard Generalized)이란 것은 누구나 특정 회사제품의 마크업에 얽매이지 않고 자신의 마크업을 개발할 수 있으며 쉽게 다른 포맷으로 변경할 수 있음을 의미한다.

이것은 관계형 DBMS에서 DB 테이블의 각 필드의 이름, 크기, 데이터 타입과 같은 정보를 알고 있을 때 데이터베이스에 접근하고 삽입, 검색, 삭제 명령을 쉽게 사용할 수 있듯이, 문서의 구조, 즉 마크업을 알고 있다면 문서에서 할 수 있는 모든 작업들을 보다 효과적이고, 유용하게 처리할 수 있다는 것을 나타낸다.

그렇다면 먼저 HTML과 XML에서의 마크업이 어떤 문서 구조를 나타내는지 살펴보도록 하자.

1) HTML과 XML의 문서구조

문서의 구조를 파악하기 위해서 먼저 간단한 XML 문서 예제를 살펴보도록 하자.

다음의 예는 책을 구분하는 문서의 타입을 나타내는 엘리먼트, 즉 태그들이 정확하게 구조를 보여주고 있다.

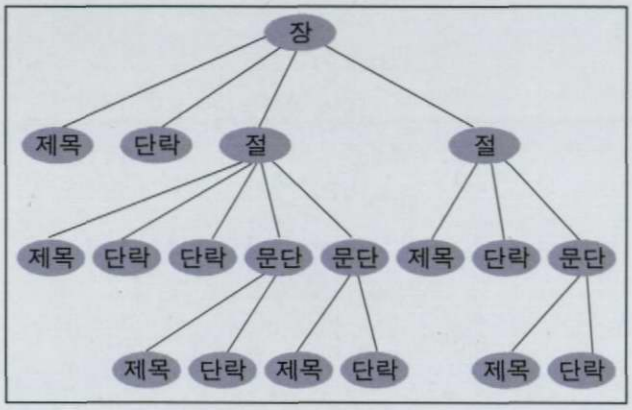
```
〈?xml version="1.0" encoding="ksc5601" ?〉
〈장〉
    〈제목〉
        장 제목
```

```
〈/제목〉
〈단락〉
        장의 내용 소개
〈/단락〉
〈절〉
        〈제목〉
                첫번째 절 제목
        〈/제목〉
        〈단락〉
                첫번째 절의 내용 1
        〈/단락〉
        〈단락〉
                첫번째 절의 내용 2
        〈/단락〉
        〈문단〉
                〈제목〉
                        첫번째 절의 문단의 제목
                〈/제목〉
                〈단락〉
                        첫번째 절의 문단의 내용
                〈/단락〉
        〈/문단〉
        〈문단〉
                〈제목〉
                        첫번째 절의 문단의 제목
                〈/제목〉
                〈단락〉
                        첫번째 절의 문단의 내용
                〈/단락〉
        〈/문단〉
〈/절〉
〈절〉
        〈제목〉
                두번째 절 제목
        〈/제목〉
        〈단락〉
                두번째 절의 내용 1
        〈/단락〉
        〈문단〉
                〈제목〉
                        두번째 절의 문단의 제목
                〈/제목〉
                〈단락〉
                        두번째 절의 문단의 내용
                〈/단락〉
        〈/문단〉
    〈/절〉
〈/장〉
```

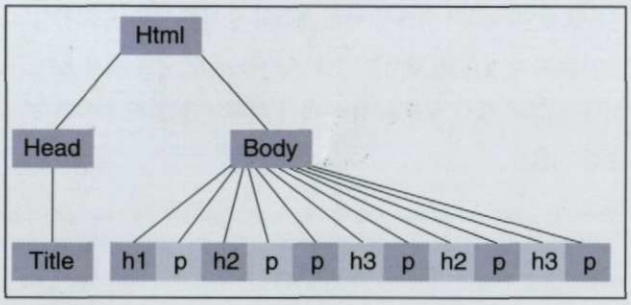
구조를 비교하기 위해서 같은 형태의 문서를 HTML로 작성해 보기로 하자. HTML에서는 종료 태그를 써도 되고 쓰지 않아도 된다. 하지만 일단 구분을 쉽게 하기 위해서 종료 태그까지 모두 포함하였다.

```
〈html〉
〈head〉
〈title〉문서 예제〈/title〉
〈/head〉
〈body〉
〈h1〉장 제목〈/h1〉
〈p〉장의 내용 소개〈/p〉
〈h2〉첫번째 절 제목〈/h2〉
〈p〉첫번째 절의 내용 1〈/p〉
〈p〉첫번째 절의 내용 2〈/p〉
〈h3〉첫번째 절의 문단의 제목〈/h3〉
〈p〉첫번째 절의 문단의 내용〈/p〉
〈h3〉첫번째 절의 문단의 제목〈/h3〉
〈p〉첫번째 절의 문단의 내용〈/p〉
〈h2〉두번째 절 제목〈/h2〉
〈p〉두번째 절의 내용 1〈/p〉
〈h3〉두번째 절의 문단의 제목〈/h3〉
〈p〉두번째 절의 문단의 내용〈/p〉
〈/body〉
〈/html〉
```

어떠한 차이가 있는지 다음 그래프를 참고해 보자. 먼저 XML 문서의 구조이다.



다음은 HTML 문서의 구조이다.



일단 그림에서 보이는 것처럼 XML로 된 문서가 그룹화가 잘 되어 있다. 그렇다면 왜 이런 그룹화가 필요한가? 만약 다른 문서에 위 문서의 첫 번째 절을 포함하려 한다면 XML에서는 〈절〉을 찾아서 거기서부터 〈/절〉까지 복사를 하면 된다. 그러나 HTML에서는 〈h2〉를 찾았다 하더라도 〈/h2〉가 그저 같은 라인에 있을 뿐 하나의 절을 포함하고 있지는 않다. 즉, 〈h2〉는 쉽게 찾을 수 있더라도 어디가 절의 끝인지를 쉽게 알 수가 없다는 것이다.

이것은 〈절〉 내의 〈문단〉을 복사하고자 할때도 마찬가지이다.

그렇다면 왜 문서의 일부분을 잘라내고 복사하는 것이 필요한가? 그것은 문서를 목적에 따라 다양하게 재사용할 수 있기 때문이다. 예를 들어 차의 엔진에 대한 설명서가 상세하게 존재한다고 생각해 보자. 그리고 그 설명서는 XML로 이루어져 다른 포맷으로 쉽게 변경이 가능하다고 가정하자. 그렇다면 우리는 차 엔진의 세부 설명서를 가지고 다음과 같은 것을 만들 수가 있다.

- 차 엔진에 대한 사용 설명서
- 차 엔진을 조립하는 방법
- 차 엔진이 고장났을 때 수리하는 방법

여기에 다음과 같이 다른 미디어로 출판하는 것도 가능하다.

- 차 엔진에 대한 CD-ROM 버전
- PDA에서 사용할 수 있는 형태
- 웹에서 사용할 수 있는 형태.

이것은 같은 정보에서 여러가지 형태의 문서를 생성할 수 있다는 것이다. 또한 문서의 내용이 수정되었을 경우 추가로 생성된 문서를 모두 바꾸는 것이 아니라 핵심 글, 즉 위에서는 차 엔진의 세부 설명서만 바꾸면 된다.

HTML은 다른 포맷으로 쉽게 변환될 수 없는 구조이기 때문에 여러가지 목적으로 사용되는 많은 정보를 저장하기에는 부족하다.

2) XML의 필요성

SGML은 마크업 언어의 표준으로 등장했지만 규모가 커서 처리해야 할 사항이 너무 많았다. 그리고 HTML은 단순하기는 하지만 사용자가 원하는 모든 작업을 수행하기에는 충분하지 못했다. 어떤 사람들은 HTML에 새로운 태그, JavaScript, VBScript와 같은 스크립트 언어들을 추가함으로서 이러한 한계를 극복하려 했다. HTML의 이러한 확장은 가능했지만, 너무 많은 회사들이 이러한 것에 뛰어들어서 웹상에서 혼란만 초래하게 되었다.

그래서 대안으로서 SGML에서 복잡하고 잘 사용되지 않는 부분을 제거하고 구현하기 쉽도록 만들어진 것이 바로 XML이다.

XML에서 다시 정의된 특성들은 다음과 같다.

1) 모든 문서에서는 지정된 문자만 사용되어야 한다. 예를 들어 모든 XML 태그는 반드시 〈로 시작해야 하고 〉로 끝나야 한다. 그리고 모든 엔터티 참조는 &로 시작해서 ;로 끝나야 한다.

2) Empty XML 엘리먼트는 일반적인 엘리먼트 사이에 아무런 내용을 가지고 있지 않은 시작태그와 끝태그의 구조를 가지거나 다음과 같이 /로 끝나는 형태로 표시해야 한다. 예를 들면

〈img src=테스트〉〈/img〉

〈img src=테스트/〉 두가지 모두 허용된다.

3) 엘리먼트를 생략하는 것이 허용되지 않는다. HTML에서는 종료태그를 생략하는 것이 허용되지만 XML에서는 그럴 수 없다.

4) DTD를 포함하지 않는 문서도 작성할 수 있다. (그러나 XML의 기능을 최대한 활용하려면 DTD를 포함해 사용하는 것이 좋다.)

이렇게 많은 예외사항을 단순화 함으로서 XML 문서를 파싱해서 처리해야 할 작업이 많이 줄어들게 되고 이것은 XML과 관련된 소프트웨어를 개발하는 것이 더욱 쉬워진다는 것을 의미한다.

XML은 SGML을 단순화시킨 것에 그치지 않고 SGML에서 제공되던 많은 문서의 처리방식들도 XML 관련 기술로서 스펙으로 만들어 나가고 있다. XLink, XPointer, XSL.. 이러한 것들은 나중에 HTML 웹페이지 보다 XML 문서를 더욱 가치있게 만들어주는 원동력이 될 것이다.

2. XML과 관련된 기술들

앞에서 우리는 XML이 문서의 구조를 정의할 수 있는 방식을 제공하고 그러한 구조를 사용하는 것과 정보를 분리해서 일부분만 보여주는 것의 장점을 보았다. XML을 사용하면 여러가지의 보다 융통성있고 가치있는 문서를 만들 수 있다고 했다. 그렇다면 언제 그러한 문서를 만들 수 있는가? 만약 XML이 HTML을 대체할 수 있으려면 최소한 기존의 HTML에서 제공되던 CSS(Cascading Style Sheet) 와 같은 것을 제공해야 한다. 또한 HTML보다 더 나아지려면 사용자들이 그것을 가지고 무엇을 할 수 있는지를 파악할 수 있도록 해야 한다.

그래서 등장한 것이 바로 XLink, XPointer,그리고 XSL이다. XLink는 HTML에서의 하이퍼링크보다도 많은 기능들을 제공한다. XLink에는 엘리먼트들 사이를 이동하기 위한 다양한 방식이 있다. HTML에서는 〈a〉태그를 통해서 현재 문서에서 다른 문서로 이동하거나 현재 문서 내에 name속성이 정해진 엘리먼트로 이동할 수 있는 하이퍼링크가 유일한 방식이었다.

그러나 XML에서 XLink와 XPointer는 이동하고자 하는 곳이 새로운 XML 문서, 또는 XML 문서의 엘리먼트, 특정 문자열, 심지어 개개의 문자들로도 이동할 수 있는 방식을 제공한다.

XSL은 XML엘리먼트들의 외양을 정의할 수 있도록 한다. 즉, 글꼴, 글자 크기, 굵기, 이탤릭체, 여백, 그리고 문서의 디자인에 영향을 미치는 다른 요소들,, 이것은 CSS나 스크립트 언어가 제공하는 기능보다도 훨씬 다양하다. 문서 내용을 특정 엘리먼트나 속성에 맞추어 정렬 할 수도 있고, 문서의 데이터, 구조, 또는 다른 조건에 맞추어 서로 다르게 보여줄 수도 있다.

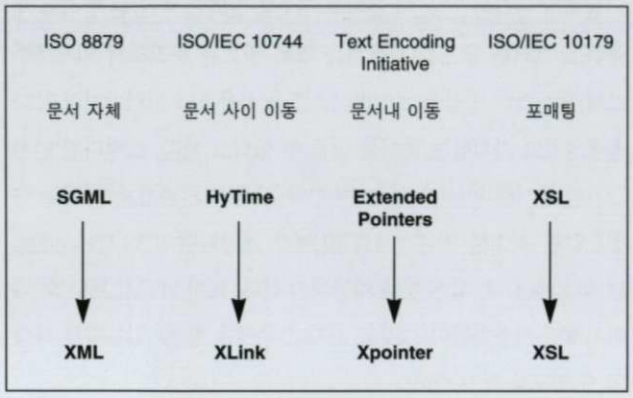
1) XLink, XPointer

HTML은 〈a href〉를 이용해서 단어나 구를 클릭하면 같은 문서의 특정 태그나 다른 문서로 이동하도록 구현되어 있다. 다음과 같은 형태를 많이 보아왔을 것이다.

〈a href=www.geongi.co.kr〉건지소프트 홈페이지〈/a〉

SGML에서는 HyTime(Hypermedia/Time based Structuring language)이 ISO 표준 ISO/IEC 10744으로서 문서 내에서 링크와 관련된 것들을 정의하는 다양한 방식들을 제공하고 있다. 시간에 기반을 둔 구조적 언어이기 때문에 단순한 텍스트와 정적인 그

림들 뿐만 아니라 오디오 클립이나 비디오 클립의 특정 위치로 이동하는 것까지도 표현할 수 있다. HyTime은 하이퍼미디어와 멀티미디어 정보를 표현하기 위해서 SGML에서 사용하는 표준 구성으로 디자인되었다.



SGML과 같이 HyTime은 구현하기는 어렵지만 매우 강력하다는 평가를 얻었다. 이러한 HyTime의 많은 특성들이 XML 워킹 그룹에 XML문서에서 엘리먼트 사이의 링크 이동을 위한 많은 아이디어를 제공해 주었다. 또한, XML은 TEI(Text Encoding Initiative)에 의해서도 영향을 받았다. TEI는 학술적인 연구를 위한 고전 작품들을 표시하기 위한 표준으로서 SGML의 DTD로 개발된 것이다.

TEI의 확장된 포인터에 대한 개념은 XML 워킹 그룹에서 XPointer에 적용되었다. Xpointer 는 HTML의 〈a href〉링크보다 다양하게 링크를 정의할 수 있는 방법을 제공한다. XPointer는 TEI에서 사용된 확장된 포인터에 대한 내용을 다듬어서 이론적인 부분보다도 실용적인 현실성에 기반을 둔 여러가지 장점을 제공하게 된다.

XLink는 XML 문법을 사용하여 객체들 사이의 관계를 정의하기 위한 스펙이다. 오늘날 HTML에서 사용하는 단방향의 하이퍼링크 뿐만이 아니라 여러가지 방향이나 타입으로 링크가 가능하도록 하기 위해서 XML문법을 사용한다. 링크가 복잡해짐으로서 사용자는 더욱 많은 곳으로 자유롭게 이동할 수 있다. 링크 소스와 타겟을 지정할 때 HTML 앵커 태그가 제공하는 것보다 다양한 방식으로 링크를 표현할 수 있다.

이동할 링크의 목적지는 다음과 같이 여러가지가 가능하다.

- 팝업 윈도우
- 두번째 브라우저에서 하이라이트된 텍스트
- 메일 박스의 이메일 메시지
- HTML 방식 〈a href〉 태그 이용 현재 링크된 엘리먼트를 다른 문서로 교체하는 형태

Xlink는 HyTime으로부터 각각의 목적에 의해서 링크의 특성을 다르게 표현할 수 있다. (예를 들면, 각주 참조, 인용, 또는 용어해설 참조와 같은 것을 링크로 표현이 가능하다.) 그리고 XML이나 SGML이 데이터 부분과 보여주는 부분이 구분되어 있듯이 XLink도 링크의 구현부분과 링크가 이동해야 할 의미부분이 분리되어 있다.

링크의 특성을 서술하는데 있어서 XLink와 함께 XPointer는 많은 강력한 새로운 능력을 제공한다.

HTML 〈a href〉 태그가 전체 문서나 지정된 name 속성을 가진 〈a〉 태그로 링크하는 반면에 XPointer는 타겟 문서 내에 원하는 어떤곳으로도 위치를 변경할 수 있다. 타겟문서에서 미리 엘리먼트의 링크를 지정할 필요가 없이 XPointer는 4번째 장의 2번째 리스트 중에서 세 번째 리스트로 이동해라와 같은 문법을 제공한다 또는 심지어 아이템의 세 번째 문자까지도 지정할 수 있다.

전체 문서나 문서의 하나의 엘리먼트로 이동하기를 강요하는 대신에 여러분은 링크 리소스의 시작과 끝으로 타겟 문서 내의 어느 곳이든 두 개의 포인터를 지정할 수 있다. 이것으로 여러분은 특정한 인용구, 서술문, 한 단어, 또는 어떤 범위의 텍스도 지정할 수 있다.

XLink의 확장된 링크는 선택할 수 있는 리스트를 팝업 윈도우 형태로 보여주든가, 또는 여러 명에게 선택가능한 모든 순서의 리

스트를 보여주거나, 링크와 관련된 리소스의 집합을 찾을 수 있도록 해 준다. 이러한 것의 가능성을 XML 응용프로그램을 개발하는 개발자의 상상력에 달려 있다.

일단 브라우저들이 링크정보의 새로운 방식들을 지원할 수 있다면 XLink의 특징은 XML 시스템에서 가장 흥미진진한 많은 것을 제공할 것이다. 왜냐하면 XML이나 XSL은 이미 몇 년 전에 해왔던 출판시스템과 유사한 기능들을 제공하지만 XLink는 완전히 새로운 것들을 처리 할 수 있는 방식을 제공하고 있기 때문이다.

2) XSL

Extensible Stylesheet Language (XSL)는 XML 문서에서 엘리먼트의 비주얼한 겉모습을 정의할 수 있도록 한다. 이러한 것을 포매팅(Formating)이라 하는데 이것은 다음에 기반을 두고 있다.

- 엘리먼트의 위치 : 예를 들면 리스트들 중에서 첫 번째 엘리먼트에 한가지 스타일을 적용하고, 리스트의 나머지 엘리먼트에는 다른 스타일을 적용한다.
- 엘리먼트의 종류 : 예를 들면 장 엘리먼트의 내부에 타이틀 엘리먼트에는 한 가지 스타일을 적용하고 섹션 엘리먼트의 내부에 타이틀 엘리먼트에는 다른 스타일을 적용한다.
- 엘리먼트의 속성값 : 예를 들면 속성값에 따라 다른 스타일을 적용한다.

또한 일반적인 텍스트도 사용할 수 있으며, 매크로의 정의, 그리고 다른 강력한 스크립트 언어도 사용할 수 있도록 제공된다.

XSL은 또한 SGML과 결합해서 작업할 수 있도록 디자인되어 있다. DSSSL(Docment Style Semantics and Specification Laguage)은 SGML문서의 처리과정을 정의하고 있다. 처리과정이라 함은 글꼴, 글자 크기, 문자 여백 그리고 색깔 과 같은 것들을 정의하는 스펙을 의미한다.

DSSSL은 HyTime과 같이 웹상에서 문서를 받는 브라우저에서 구현해야 할 것이 너무 많았다. DSSSL-O라 불리우는 단순화된 버전을 만들기 위한 노력이 있었고 후에 DSSSL-O가 XSL로 변경되었다.

언뜻 보면, XSL은 DSSSL과 전혀 비슷하지 않다. DSSSL은 스키마에 뿌리를 두고 있고 LISP 프로그래밍 언어를 이용해 관계된 프로그래밍 구조들을 그룹화하는 것을 나타낸다. 그러나 XSL은 XML과 비슷하게 보인다.

W3C표준에 따르면 HTML문서로 할 수 있는 것은 XML 문서에서도 모두 할 수 있어야 한다. 그래서 XSL을 디자인할 때 CSS에서 XSL로 기계적인 매핑이 가능하도록 하는 것이 중요하다. 즉, CSS에서 XSL로 변경하는 프로그램을 짜기 쉬워야 한다는 것이다.

왜 CSS를 XSL로 변경해야 하는가? CSS 스타일은 매우 적은 코드에서 빠르게 포매팅을 할 수 있다. 그리고 종종 웹에서 전달되는 문서의 공통적인 포매팅의 시작포인트로서 이상적이다. 그러나 온라인 출판 시스템이나 웹이 더욱 복잡하게 성장함에 따라 CSS보다 더 복잡한 것을 요구했다. 예를 들면 CSS는 엘리먼트를 정렬할 방법이 없다. 또, 많은 문서에서 일부분만 추출하여 또다른 문서를 생성할 수도 없다.

3. 정리

이상으로 XML과 XML 관련된 기술들을 살펴보았다. XML이 모든 문제의 해결책이 될 수는 없다. 그러나 XML은 CAD/CAM이나 문서관리, 물품전시, 워드프로세서, 스프레드시트, 데이터베이스에 이르는 프로젝트들을 위한 더 나은 해결책이 될 것임은 분명하다. XML의 구조는 관계형 데이터베이스 테이블과 스프레드시트 정보, 기타 복잡한 문서들을 충분히 저장할 수 있을 정도로 유연하다. 웹자체가 표현 엔진에서 업무 도구로 변모하기 위해서 더욱 발전해야 하고 XML과 여러가지 관련 기술들이 발전할 때 빨리 확산될 수 있을 것이다.