DocBook으로 글 쓰기

박용주
yongjoo@kldp.org

DocBook으로 글 쓰기
지은이 박용주

펴냄 $Date: 2001/08/18 13:08:06 $
Copyright © 2000, 2001 by 박용주

이 글은 DocBook DTD를 처음 사용하려는 사람들을 위한 안내서입니다.

여기서는 DocBook의 배경을 간략하게 알아보며, 그 사용법에 대해 실례를 들어가며 설명합니다. 특히 이 글은 DocBook에서 한글을 쓰는데 관한 내용을 중점적으로 다룹니다.

이 글의 많은 부분은 다른 문서에 의존하여 쓰여졌습니다. 이런 부분에서는 원문의 출처를 최대한 밝히려 노력하였습니다.

이 문서는 GNU Free Documentation License (http://www.gnu.org/copyleft/fdl.html) 버전 1.1 혹은 자유 소프트웨어 재단에서 발행한 이후 판의 규정에 따르며 저작권에 대한 본 사항이 명시되는 한 어떠한 정보 매체에 의한 본문의 전재나 발췌도 무상으로 허용됩니다.

고친 과정

고침 0.1 2000/05/23고친이 yongjoo
이 글이 첫 선을 보였습니다.
고침 0.2 2000/10/29고친이 yongjoo
전체적인 틀을 새로 짰으며 여러가지 새로운 내용이 많아졌습니다.
고침 0.2.22001/08/18고친이 yongjoo
몇가지 틀린 점을 고치고 KLDP 스타일시트에 대한 내용을 보충하였습니다.

차례

표 목록

예 목록

수식 목록

머리글

1. DocBook을 소개합니다!

DocBook은 좀더 아름답고 효율적인 방법으로 글을 쓰고 펴낼 수 있게 도와주는 하나의 문서 형식입니다. 이를 사용하면 문서의 겉모양에는 거의 구애받을 필요 없이 글 자체의 구조와 내용에만 집중할 수 있으며, 완성된 글은 다양한 포맷으로 자동 변환되므로 자신의 지식과 생각을 좀더 자유롭게 널리 펴낼 수 있습니다.

DocBook은 각각 SGML과 XML 표준에 따라 만들어진 두가지 형태가 있으며 흔히 태그(tag:꼬리표)라고 부르는 조판기호(markup)들로 이루어져 있습니다. 이들은 글의 전체적인 구성과 그 부분적인 특성들을 묘사합니다. 이를 해석하여 보기 좋은 다른 형식의 문서로 변환하기 위해서는 몇가지 외부적인 도구들의 힘을 빌려야 합니다.

작성된 문서는 여러가지 다른 형식으로 얼마든지 변환할 수가 있습니다. 예를 들면 HTML, XML, PS, PDF, RTF, TeX 같은 형식으로 만드는 것이 가능하며, 그 밖에 다른 형식으로도 만들 수 있습니다. 따라서 한번의 문서 작성만으로도 최대한 많은 사람들이 다양한 형식으로 정보를 얻을 수 있게 됩니다.

이진수화된 정보들은 손상받기 쉬우며 다루기도 힘듭니다. 그러나 DocBook으로 작성된 문서에는 이상한 깨진 문자(즉, 이진수 형식으로 저장된 정보)들이 없습니다. 따라서 문서의 내용을 직접 검색하고 색인을 만드는 일이 가능합니다. 또한 DocBook 문서는 그 구조와 내용이 명확하게 규정되어 있으므로 훨씬 지능적인 문서 검색과 가공이 가능합니다.

DocBook은 300가지가 넘는 태그를 사용하는데, 그 각각은 몇가지 속성을 가지고 있으며 여기에는 몇가지 값이 미리 가정되어 있기도 합니다. 태그의 구조와 종류를 정의해 놓은 것을 DTD라고 하는데 이들은 필요에 따라 수정되거나 새로 정의될 수도 있습니다.

다만, DocBook이 정의한 DTD에 조금이라도 어떤 수정이 가해졌다면 그것은 이미 DocBook이 아니라는 점을 명심하도록 합시다.

최신 버전의 DocBook은 DocBook 공식 홈페이지 (http://www.oasis-open.org/docbook/)에서 구할 수 있습니다.

> 참고**:** 이 내용은 DocBook HOWTO (http://www.ibiblio.org/godoy)의 Introduction을 수정 보완한 것입니다.

2. 이 글에 대해서

이 글은 원래 Jorge Godoy의 DocBook HOWTO에 기반하고 있었습니다. 여기에 몇가지 부족한 점을 보충하고 한글 관련 팁을 추가한 것이 이 글의 버전 0.1입니다. DocBook HOWTO의 원문은 Godoy’s DocBook SGML/XML reference page (http://www.ibiblio.org/godoy/)에서 구할 수 있습니다.

이제 Modular DocBook Stylesheet (http://docbook.sourceforge.net)에 한국어 지역화가 포함되면서 DocBook의 한글 사용에 많은 변화가 생겼습니다. 그래서 글의 전체 틀을 새로 짜고 많은 내용을 수정 보충하여 버전 0.2를 내놓게 되었습니다. 여기에는 Normal Walsh와 Leonard Muellner가 지은 DocBook: The Definitive Guide의 내용을 많이 참고하였습니다. 이 글의 원문은 DocBook: The Definitive Guide 홈페이지 (http://www.docbook.org)에서 볼 수 있습니다.

이 글에서는 DocBook의 모든 기능을 다루지 않습니다. 여기서 설명된 내용은 아주 간단한 경우만을 다루고 있습니다. 따라서 DocBook의 모든 것이 설명된 공식 참고서 'DocBook: The Definitive Guide'를 꼭 살펴보기 바랍니다.

또한 리눅스 한글 문서 프로젝트 (http://kldp.org)에서 운영하는 Docs 메일링리스트에 가입하면 DocBook에 관한 여러 정보를 얻을 수 있으며 질문과 토론을 같이 할 수 있습니다. 메일링리스트 가입은 열린 글짓기 홈페이지 (http://docs.kldp.org)를 참고하세요.

아직 이 글은 미흡한 부분이 많으며 군데군데 잘못된 내용이 있을 수도 있습니다. 또한 아직 완성되지 못한 곳도 있습니다. 이런 부분들은 앞으로 계속 보완해 나갈 예정입니다. 이 글에서 잘못된 부분이나 미흡한 점을 발견하면 지은이에게 바로 알려주시기 바랍니다.

3. 감사의 글

이 글은 많은 분들의 도움으로 이루어질 수 있었습니다.

- 모든 공개 문서 작업의 모태이자 DocBook 한글 작업의 기반인 리눅스 한글문서 프로젝트 (http://kldp.org)를 운영해 주시는 권순선 님께 감사드립니다.
- DocBook Technical Committee의 의장이자 거의 모든 DocBook 관련 프로젝트를 주도하는 Normal Walsh 님과 DocBook: The Defintive Guide의 공동 저자인 Leonard Muellner 님께 감사드립니다.
- 이글의 기반이 된 DocBook HOWTO를 집필한 Jorge Godoy 님께 감사드립니다.
- 무엇보다도, KLDP의 Docs 메일링리스트에서 많은 의견주시고 문제점을 알려주신 모든 분들께 감사드립니다. 이 문서와 KLDP 스타일시트는 여러분들의 도움이 없이는 만들어지지 못했을 것입니다.

1장. SGML과 XML, 그리고 DocBook

1.1. SGML, XML

DocBook을 사용해 글을 쓰기 위해서는 일단 SGML(Standard Generalized Markup Language)과 XML(eXtensible Markup Language)에 관한 약간의 이해가 필요합니다. 혹시 HTML(Hypertext Markup Language)에 대하여 알고 있다면, SGML을 이해하기가 좀더 수월할 것입니다.

HTML은 태그(tag:꼬리표)란 것을 조판기호(markup)로 사용해 글과 그림의 모습을 묘사합니다. 웹 브라우저는 이 조판기호들의 의미를 파악해서 실제로 우리가 보게 되는 웹 페이지를 화면에 보여줍니다. 이런 HTML과 같은 것을 일컫어 마크업 언어(markup language)라고 합니다.

SGML은 마크업 언어가 어떻게 정의되어야 하는지를 규정하는 국제적인 표준입니다. 그리고 이런 SGML 표준에 따라 특정 마크업 언어를 직접 정의해 놓은 것을 DTD(Document Type Definition)라고 합니다. 즉, HTML은 SGML의 규정에 따라 정의된 하나의 DTD입니다. 그러므로 'HTML 문서'란 것을 엄밀하게 말한다면 'HTML DTD를 사용하는 SGML 문서'라고 할 수 있겠습니다.

XML은 HTML을 좀더 지능적으로 향상시키기 위해 개발되었으며, 이미 많은 웹 브라우저들이 XML을 지원하고 있습니다. 그러나 XML은 단순히 HTML을 개량한 것은 아닙니다. XML은 마크업 언어가 어떻게 정의되어야 하는지에 대한 좀더 새롭고 쉬운 표준입니다. 즉, 인터넷 환경에 알맞은 새로운 버전의 SGML이라고 할 수 있습니다.

> 참고: 이 부분의 내용은 DocBook: The Definitive Guide (http://www.docbook.org/)의 'HTML and SGML vs. XML' 부분을 요약, 정리한 것입니다.

1.2. DocBook 이야기

우리가 사용하려고 하는 것은 DocBook DTD입니다. 이것은 1991년 Hal Computer Systems란 회사와 O'Reilly & Assosiates 출판사에 의해 만들어졌으며, 그 후, 이들로부터 Davenport Group이 DocBook을 위해 따로 조직되었습니다. 그리고 Davenport Group은 1998년에 OASIS(Organization for the Advancement of Strunctured Information Standards)로 이름을 바꾸었습니다.

DocBook DTD는 주로 기술적인 분야의 문서를 작성하기 위해 만들어졌으며, 앞서 예를 들었던 HTML DTD와는 확연히 다른 특징을 지니고 있습니다. HTML은 문서에 포함된 글이나 그림 등이 웹 브라우저 상에서 어떻게 보여야 할지에 대해 주로 정의합니다. 그러나 DocBook은 문서의 모양이 어떻게 보여야 하는지에 대해서는 전혀 정의하지 않습니다. 대신에, DocBook은 글의 논리적인 구조만을 다룹니다.

따라서 DocBook으로 작성한 문서는 글의 전체 구조와 세부적인 특성들(제목, 부제목, 본문, 주석 등등)을 더할나위 없이 명확하게 알 수 있게 됩니다. 바로 이 점 때문에, DocBook으로 작

성된 문서는 어떤 양식의 문서로도 자동적인 변환이 가능하며, 또한 DocBook 문서를 직접 검색하여 필요한 정보만을 뽑아낼 수도 있는 것입니다. 인터넷을 통해 글을 펴내는 데 있어, DocBook이 각광 받게된 것은 이와 같은 요인들 덕분입니다.

그런데, 이렇게 글의 논리 구조만을 다루고 있는 DocBook을 보기좋은 문서로 변환하기 위해서는 그 논리 구조를 구체적으로 어떤 모습으로 표현할 것인지를 따로 알려주어야 합니다. 이에 쓰이는 것이 스타일시트(stylesheet, 문서 양식)입니다. 또한 이 스타일시트에 사용되는 언어를 스타일시트 언어(stylesheet language)라고 하며, FOSIs DSSSL CSS XSL 등 여러가지가 있습니다. 스타일시트를 사용해 DocBook 문서를 변환하는 데 대해서는 3장에서 자세히 설명합니다.

DocBook DTD는 SGML과 XML 버전이 모두 사용 가능합니다. XML 버전은 SGML 버전보다 좀더 세련된 기능을 앞으로 제공할 것이지만 아직은 그 기능이 완전치 못하므로 이 글에서는 SGML 버전을 기준으로 설명합니다. 이 글에 없는 DocBook에 대한 새로운 정보와 XML 버전에 대한 도움말 등은 열린 글짓기 홈페이지 (http://docs.kldp.org)를 참고하시기 바랍니다.

2장. 글 쓰기

2.1. 준비

DocBook으로 글을 쓰기 위해서는 단순 텍스트 파일을 다룰 수 있는 문서 편집기를 사용하여야 합니다. Unix 환경이라면 Vi나 Emacs 등을 사용하면 되겠고, MS-Windows 환경이라면 메모장 류의 프로그램을 사용하면 됩니다.

한글 DocBook 파일의 인코딩으로는 euc-kr(KSC 5601)이나 utf-8(유니코드 [1])를 쓸 수 있습니다 MS-Windows에서는 KSC 5601에 정의되지 않은 글자들도 확장 완성형을 사용해 표현하는데, 이런 확장 완성형 글자들은 문서 변환시 결국 깨져버리므로 사용하면 안됩니다. KSC 5601에 정의되지 않은 글자를 사용할 필요가 있을 때는 유니코드(utf-8)를 사용하면 됩니다.

문서 편집기들 중에는 특별히 DocBook 지원 기능을 갖춘 것도 있습니다. Vim과 Emacs에서는 DocBook 태그를 인식하여 예쁘게 색을 입혀주는 기능을 사용할 수 있습니다. 또한 Emacs에서는 적절한 태그를 자동으로 삽입해 주는 기능도 사용할 수 있습니다. Vim과 Emacs는 Unix나 MS-Windows에서 모두 사용할 수 있으므로 더욱 편리합니다. [2]

2.2. 마크업 언어의 기본 익히기

마크업 언어는 기초요소**(element)**, 속성**(attribute)**, 실체요소**(entity)**로 구성됩니다.

2.2.1. 기초요소

기초요소는 글의 전체 구조와 세부적인 특성들을 묘사합니다. 대부분의 기초요소들은 그 해당 영역을 열고 닫는 한 쌍의 태그를 필요로 합니다.

```
<para>
  그리고 "우리는 ...을 안다(we know ...)"란 말을 함부로
  하여서는 안된다. 왜냐하면, 당연히 우리는 모르기 때문이다.
  그러나 우리는 아는 것처럼 행동해야 하는 것이다.
</para>
```

위의 예는 한 문단의 시작과 끝을 나타내는 기초요소 `para`의 사용법을 보여줍니다. 여는 태그인 `<para>`과, 닫는 태그인 `</para>`가 서로 한 쌍을 이루는 것을 알 수 있습니다.

그러나 기초요소들 중에는 여는 태그만 사용하는 것도 있습니다. 참조를 위해 사용되는 `xref` 같은 경우가 이에 해당됩니다.

```
<para>
  이렇기 때문에, 컴퓨터를 끄기 전엔 반드시 적절한
  셧다운 절차를 밟아야만 하는 것이고(
  <xref linkend="boots-and-shutdowns">
```

```
    참조 ), 마운트한 플로피를 빼기 전엔 꼭 언마운트를
    해야하는 것이다.
  </para>
```

위의 예에서 참조할 글의 위치를 나타내는 기초요소 xref는 단지 여는 태그인 <xref linkend="boots-and-shutdowns">만 사용하고 있습니다.

2.2.2. 속성

기초요소들은 속성을 가질 수 있습니다. 예를 들어, 기초요소 xref는 linkend라는 속성을 가집니다. 위의 예에서는 이 값이 boots-and-shutdowns로 지정되어 있습니다.

```
<chapter id="boots-and-shutdowns">
<title>부팅과 셧다운</title>
  <para>
    여기서는 리눅스 시스템이 시작될 때와 멈춰질 때 어떤 일이
    진행되는지를 설명할 것이며, 또한 그것이 제대로 진행되려면 어찌
    해야하는지에 대해서도 알아볼 것이다. 만일, 이 때 적절한 과정이
    수행되지 못한다면, 파일들이 손상을 입거나 지워질 수도 있다.
  </para>
</chapter>
```

모든 기초요소들은 id라는 속성을 가질 수 있습니다. 위의 예에서는 기초요소 chapter의 id 속성 값이 boots-and-shutdowns로 지정되어 있습니다. 따라서 바로 이 부분이 <xref linkend="boots-and-shutdowns">가 가리키는 참조 글의 위치입니다.

2.2.3. 실체요소

실체요소는 어떤 특정한 데이터 덩어리에 이름을 붙여주기 위해 사용됩니다. 실체요소는 내부 실체요소와 외부 실체요소로 나눌 수 있습니다.[3]

내부 실체요소는 다음과 같이 선언합니다.

```
<!ENTITY ora "O'Reilly &amp; Associates">
```

이렇게 내부 실체요소를 선언하고 나서, 문서에 &ora;라고 써넣게 되면 이 부분은 O'Reilly & Associates라는 문자열로 치환되게 됩니다. 이와 같이 &는 특수한 의미를 가진 문자이므로, 이 문자를 그대로 문서에 표시하기 위해서는 &라고 써주어야 합니다.

외부 실체요소는 다음과 같이 선언합니다.

```
<!ENTITY ch01 SYSTEM "ch01.sgm">
```

이렇게 실체요소를 선언하고 나서, 문서에 &ch01;이라고 써넣게 되면 그 곳에 ch01.sgm 파일의 내용 전체가 삽입됩니다. 이 방법은 문서를 여러 파일에 나누어서 작성할 때 유용합니다.

위의 예에서 SYSTEM은 큰 따옴표 사이의 내용을 시스템 식별이름[4]으로 해석하라는 지시어입니다.

또한 외부 실체요소에는 특수 문자를 위한 실체요소가 있습니다. 방금 위에서 사용된 &가 바로 특수 문자 실체요소입니다. 이 밖에도 무척 많은 수의 특수 문자 실체요소들이 정의되어 있으므로 DocBook: The Definitive Guide (http://www.docbook.org/tdg/html/docbook.html)의 DocBook Character Entity Reference (http://www.docbook.org/tdg/html/refchar.html)를 꼭 살펴보시기 바랍니다.

2.2.4. 주의할 점

DocBook의 기초요소와 속성은 대소문자를 가리지 않습니다. 따라서 `<para>`와 `<pArA>`는 같은 것으로 간주됩니다. 그러나 실체요소는 대소문자를 가리므로 주의하여야 합니다.

그러나 XML과의 호환성을 고려한다면, 모든 기초요소와 속성들을 소문자로 일관되도록 하는 것이 좋습니다. 또한 닫는 태그를 사용할 때 생략형을 사용하지 않도록 하고[5] 속성은 꼭 큰 따옴표로 묶도록 합니다.

> 참고**:** 이 내용은 DocBook: The Definitive Guide (http://www.docbook.org/)의 'Elements and Attributes'를 많이 참고하였습니다.

2.3. 글 쓰기의 시작

모든 SGML 문서는 문서 형식 선언(Document Type Declaration)으로 시작되어야 합니다. 이것은 문서에서 어떤 DTD를 사용하며, 그 문서의 최상위 기초요소(root element)가 무엇인지를 알려줍니다. DocBook 문서의 전형적인 문서 형식 선언은 다음과 같습니다.

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN">
```

이 선언에서는 문서의 최상위 기초요소가 `<book>`이라는 점을 알려주고 있습니다. 또한 문서에서 사용하는 DTD를 큰 따옴표 사이의 내용을 통해 알려주고 있습니다. PUBLIC은 큰 따옴표 사이의 내용이 공식 식별이름[6]이란 것을 알려주기 위한 지시어입니다.

이렇게 문서 형식 선언을 하고 나면, 그에 맞는 최상위 기초요소의 여는 태그를 써넣는 것으로부터 글 쓰기가 시작됩니다.

2.3.1. 최상위 기초요소

기초요소들은 계층 구조로 조직화 되어 있습니다. 문서의 최상위 기초요소란 것은 그 문서에서 사용된 기초요소들 중 가장 상위 계층의 기초요소를 가리키는 말입니다.

DocBook에서 계층구조의 최상위는 `set`입니다. `set`은 다시 많은 수의 `book`을 포함하며, `book`은 다시 많은 수의 `chapter`를 포함합니다. 그리고 `chapter`는 많은 `section`을 포함하는

데, section은 자기자신을 다시 포함할 수가 있으므로 계속 section으로 세분화할 수가 있습니다. section 아래에는 para가 있으며 이것은 하나의 문단을 나타내는 것입니다.

만약 최상위 기초요소를 chapter로 지정한다면, 그 보다 상위에 있는 book과 set을 사용하지 않고 그 하위 요소만으로 글의 구조를 만들겠다는 뜻이 됩니다. 즉, 책 논문 안내문 등, 문서 특성에 맞는 어떤 최상위 기초요소를 사용하느냐에 따라 그에 걸맞는 하위 구조를 가진 문서를 작성할 수 있게 됩니다.

최상위 기초요소에는 보통 chapter, article, book 정도만을 사용하지만 그 밖에 set, section, para 등 다른 기초요소들도 필요에 따라 사용할 수 있습니다. chapter는 간단한 안내문 정도의 글을 작성하는 데 적당하며, article은 짧은 논문이나 보고서 수준의 글을 쓰는데 알맞습니다. 사실 article은 계층구조의 위치상에서 chapter와 같은 급에 속하는 것이지만, chapter를 좀더 독립된 수준의 문서로 만들기 위해 따로 정의된 것입니다. book은 독립된 책으로 펴낼 수 있을 정도의 구조를 갖추어야 하는 글을 위한 것으로서 preface, appendix 등과 함께 많은 수의 chapter를 포함합니다. 또한 set는 많은 수의 book을 포함하게 되어 있으므로, 여러 책들의 모음집을 펴내는데 알맞습니다.

2.3.2. chapter의 구조

아래의 예는 간단한 chapter의 구조를 보여주고 있습니다.

chapter에는 제목(title)이 있어야 하며, 문단의 시작과 끝을 나타내는 <para>와 </para> 사이에 글의 내용을 써넣어야 합니다.

예 **2-1. Chapter**의 간단한 예

```
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<chapter lang="ko">
  <title>이 곳에 제목을 씁니다</title>
  <para>
    첫번째 문단입니다.
    이 곳부터 글의 본문을 씁니다.
    띄어쓰기를        두 칸        이상 하거나
    줄바꿈을 하더라도

    모두 한 칸 띄어쓰는 효과만을 냅니다.
  </para>
  <para>
    두번째 문단입니다.
    첫번째 문단과는 줄바꿈을 통해
    구별되어 보이게 됩니다.
    문단 안에서는 아무리 줄을 바꾸어도
    한칸 띄어쓰는 효과만을 냅니다.
  </para>
  <para>
    세번째 문단입니다.
```

```
    이런 식으로 계속 문단을 추가하여
    글을 써내려 가면 됩니다.
  </para>
</chapter>
```

<chapter lang="ko"> 부분을 눈여겨 보기 바랍니다. 이 부분에서 chapter의 lang 속성이 ko로 지정되어 있는데, 이것은 이 chapter가 한국어로 씌어져 있다는 점을 나타냅니다. 이 속성이 제대로 지정되어 있어야만 한국어 환경에 알맞도록 문서가 변환됩니다.[7] lang 속성이 지정된 기초요소의 하위 구조는 모두 이 속성을 갖게 됩니다.

chapter를 세분화하기 위해서는 section을 사용하면 됩니다. section은 또다른 section으로 계속 세분화할 수 있습니다. section에도 title이 있어야 하며 역시 para를 사용해 문단을 나눕니다.

예 **2-2. Chapter**의 일반적인 예

```
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<chapter lang="ko">
  <title>라면의 종류</title>
  <para>
     라면의 종류에는 어떤 것이 있을까요?
  </para>

  <section>
    <title>떡 라면</title>
    <para>
      떡 라면은 가래떡을 어슷하게 썰어 넣은 라면입니다.
    </para>
    <para>
      가래떡이 쫄깃하게 씹히는 맛이 일품이죠.
    </para>
  </section>

  <section>
    <title>김치 라면</title>
    <para>
      김치 라면은 김치를 썰어 넣은 라면입니다.
    </para>
    <para>
      얼큰한 김치 맛이 라면과 잘 어울리죠.
    </para>
  </section>

  <section>
    <title>이상한 라면</title>
    <para>
      이상한 라면들도 있습니다.
```

```
        </para>

        <section>
          <title>우유 라면</title>
          <para>
            우유 라면은 우유를 부어 끓이는 라면입니다.
          </para>
          <para>
            우유는 라면 맛을 고소하고 부드럽게 해줍니다.
          </para>
        </section>

        <section>
          <title>카레 라면</title>
          <para>
            카레 라면은 카레를 얹어 끓이는 라면입니다.
          </para>
          <para>
            카레의 향기는 라면의 감칠맛을 더욱 풍부하게 해줍니다.
          </para>
        </section>

      </section>

    </chapter>
```

'이상한 라면' section이 '우유 라면' section과 '카레 라면' section으로 다시 세분화되어 가는 구조를 눈여겨 보시기 바랍니다. section은 이런 방식으로 계속 세분화가 가능합니다.

작은 정보: section의 구조를 명확히 하기 위해서 <sect1>, <sect2>, <sect3>.. 식으로 계층구조를 만들 수도 있습니다. 즉 <section> 대신 <sect1>을 사용하며 <sect1>이 세분될 때는 <sect2>, <sect2>가 세분될 때는 <sect3>를 사용하는 방식으로 계속 세분화를 할 수 있습니다.

이 방식은 계층 구조를 한눈에 알아볼 수 있는 장점이 있지만 구조가 복잡한 글에서는 그 구조를 변경할 때마다 모든 기초요소를 일일이 수정해 주어야 하므로 아주 곤란해 질 수 있는 단점이 있습니다. 그러므로 자신에게 편리한 방식을 선택해서 사용하면 되겠습니다.

2.3.3. article의 구조

article은 기본적으로 chapter와 같은 하위구조를 가집니다. 그러나 article은 articleinfo라는 기초요소를 포함하고 있어야 합니다. 이 밖에도 article은 몇가지 확장된 하위구조를 가질 수 있습니다.

> **주의**
>
> DocBook V3.1 이하에서 사용되던 `artheader`가 V4.0부터는 `articleinfo`로 이름을 바꾸었습니다. 그러므로 V4.0부터는 더 이상 `artheader`를 사용해선 안 됩니다.

`articleinfo`는 글 과 글쓴이에 대한 다양한 정보를 체계적으로 정리하기 위한 기초요소입니다. 다음은 간단한 `article`의 예입니다.

예 **2-3. article**의 간단한 예

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<article lang="ko">

  <articleinfo>
    <title>신용장 거래시 주의 사항</title>

    <author>
      <surname>홍</surname>
      <firstname>길동</firstname>

      <affiliation>
        <orgname>사고팔고 물산</orgname>
        <orgdiv>미주 뉴욕 지부</orgdiv>

        <address>
          <email>gildong@sagopalgomulsan.com</email>
        </address>

      </affiliation>

    </author>

  </articleinfo>

  <abstract>
    <para>
      이 글은 신용장 거래시 발생할 수 있는 문제점과 그 대책에
      대해 설명합니다.
    </para>
  </abstract>

  <section>
    <title>클레임 관련</title>
    <para>
      신용장 거래는 서류상의 거래이므로 신용장을 통해 물품의
```

```
      하자에 대한 클레임을 제기할 수 있는 방법이 없습니다.
    </para>
    <para>
      그렇기 때문에 계약서에 클레임 해결조항을 규정해야 함은
      물론 물품의 하자가 발견되면 즉시 검정기관을 통해
      survey report를 만들어 증빙을 확보하고 계약서에 따른
      클레임 제기절차를 밟아야 합니다.
    </para>
  </section>

  <section>
    <title>중개무역시 유의점</title>
    <para>
      중개무역 거래시 신용장을 통해 결제하는 경우에는 상업송장
      및 환 어음, 포장명세서 등은 중개인 명의로 대체가 가능합니다.
    </para>
    <para>
      그러나 선하증권 및 원산지 증명서의 경우는 중개인이 직접
      만들 수 없으므로 신용장 상에 'Third party documents are
      acceptable'이라는 문구가 있어야 Nego시 하자가 발생하지
      않습니다.
    </para>
  </section>

</article>
```

articleinfo가 포함하고 있는 기초 요소들을 주의깊게 살펴보기 바랍니다. articleinfo는 title과 author를 포함하고 있으며 author는 surname firstname affiliation을 포함하고 있습니다.

author는 글쓴이에 관한 정보를 담기 위한 기초요소입니다. 글쓴이의 성은 surname이 나타내며 이름은 fitstname이 나타냅니다.

참고: 한국어 지역화가 포함되지 않은 Modular DocBook Stylesheet V1.58 이하 버전을 사용하게 되면 성과 이름이 서양식으로 뒤바뀌어 표현되니 주의해야 합니다.

작은 정보: 만일 성과 이름을 구별하여 띄어쓰기를 해야 할 필요가 있는 경우에는 Modular DocBook Stylesheet의 한국어 지역화 파일(common/db11ko.dsl)을 약간 수정하여야 합니다. 파일의 주석문을 읽어보기 바랍니다.

affiliation에는 글쓴이의 주소(address), 소속단체(orgname), 소속부서(orgdiv)에 관한 정보가 들어갑니다. address에는 보통 email 기초요소를 사용해 전자우편 주소를 적어 넣습니다.

초록(abstract)에는 글에 대한 설명이나 짧은 요약을 써 넣습니다. section 이하의 구조는 앞서 살펴본 내용과 같습니다.

2.3.4. book의 구조

book 기초요소는 한 권의 책을 펴내기에 알맞은 아주 다양한 하위구조를 가지고 있습니다. 아래의 예는 극히 기본적인 구조만을 사용한 경우입니다.

예 **2-4. book**의 간단한 예

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<book lang="ko">

  <bookinfo>
    <title>대우주에의 도전</title>
    <pubdate>1983년 4월 20일</pubdate>

    <author>
      <surname>조</surname>
      <firstname>경철</firstname>

      <affiliation>
        <orgname>경희대학교</orgname>
        <jobtitle>부총장</jobtitle>
      </affiliation>

    </author>

    <abstract>
      <para>
        우리들은 우주시대에 살고 있다고 하지만,
        우리나라는 그 혜택만 입는 데에 만족하고
        있는 현상을 우리들은 어떻게 생각하여야 할까?
      </para>
      <!-- 원문에서 중간 부분을 생략하였습니다 -->
      <para>
        우리들에게 &lt;꿈과 자신감&#62;을 필자는
        심어주려고 전력투구한 작품이다.
      </para>
    </abstract>

  </bookinfo>

  <preface>
    <title>서문<title>
    <para>
      &lt;나는 자랑스러운 태극기 앞에 국력부강에
      혼신의 노력을 다했다&#62;는 말은 우리들
      누구나 다 할 수가 있다.
    </para>
    <!-- 원문에서 중간 부분을 생략하였습니다 -->
```

```
  <para>
    나는 믿는다 &lt;뿌린 씨는 거두리라.&#62;
  </para>
</preface>

<chapter>
  <title>대우주에의 도전</title>
  <para>
    우리 인간이 지구를 정복할 수 있었던 가장 큰
    원동력은, 여러가지 이유를 들 수 있겠지만, 역시
    &lt;사고능력&#62;을 월등하게
    가진 것과 손발이 기술사회를 이룩하기 위해
    적당하게 우리 몸에 달려 있었기 때문이라
    할 수 있을 것이다.
  </para>
  <!-- 원문에서 중간 부분을 생략하였습니다 -->
</chapter>

<chapter>
  <title>세 사람의 선각자</title>

  <section>
    <title>우주여행의 아버지, 티올코프스키<title>
    <para>
      모스크바의 남서부쪽에 200Km쯤 떨어진 곳에
      자리잡은 가루가라는 마을에 수학교사
      콘스탄틴 티올코프스키가 살고 있었다.
    </para>
    <!-- 원문에서 중간 부분을 생략하였습니다 -->
  </section>

  <section>
    <title>근대 로케트의 아버지, 가다드<title>
    <para>
      우주여행의 이론은 티올코프스키에 의해 확립되었지만,
      다음에 필요한 것은 그런 우주여행에 필요한
      로케트를 실제로 만들어 보는 일이다.
    </para>
    <!-- 원문에서 중간 부분을 생략하였습니다 -->
  </section>

</chapter>

  <!-- 원문의 나머지 부분은 생략하였습니다 -->

<appendix>
  <title>부록<title>

  <section>
```

```
  <title>E.T.를 찾자<title>
  <para>
    달이 없는 맑은 밤하늘...... 도시 등불이 없는
    들판에나 산 혹은 바다 위에서 쳐다보는 하늘은
    정말로 아름답다.
  </para>
  <!-- 원문에서 중간 부분을 생략하였습니다 -->
</section>

<section>
  <title>과학이 계산한 E.T.의 수<title>

  <informalequation>
    <alt>
      N=R<subscript>*</subscript>f<subscript>p</subscript
      >n<subscript>e</subscript>f<subscript>e</subscript
      >f<subscript>i</subscript>f<subscript>c</subscript>
    </alt>

    <mediaobject>
      <imageobject>
        <imagedata fileref="ETnumber.eps" format="eps">
      </imageobject>
      <imageobject>
        <imagedata fileref="ETnumber.gif" format="gif">
      </imageobject>
      <imageobject>
        <imagedata fileref="ETnumber.bmp" format="bmp">
      </imageobject>
      <textobject>
        <phrase>N=R<subscript>*</subscript>f<subscript
        >p</subscript>n<subscript>e</subscript>f<subscript
        >e</subscript>f<subscript>i</subscript>f<subscript
        >c</subscript>L</phrase>
      </textobject>
    </mediaobject>

  </informalequation>

  <para>
    위의 식은 1970년 여름 NASA의 에임스 연구센터에
    모인 우주생물, 화학, 천문학자들이 모여 연구발표회를
    한 자리에서 유명한 칼 세이간 교수가 발표한
    &lt;우주 문명사회의 수&#62;에 관한
    공식이다.
  </para>
  <!-- 원문에서 중간 부분을 생략하였습니다 -->
</section>
```

```
    <!-- 원문의 나머지 부분은 생략하였습니다 -->

  </appendix>

</book>
```

book은 bookinfo를 포함하고 있어야 합니다. bookinfo의 역할은 articleinfo와 기본적으로 같습니다. bookinfo 안에 새롭게 등장한 pubdate 기초요소와 jobtitle 기초요소를 눈여겨 보기 바랍니다. pubdate은 글을 펴낸날, jobtitle은 글쓴이의 직위를 알려주기 위한 기초요소입니다.

위의 예에서는 book이 초록(abstract)과 서문(preface), 두 개의 장(chapter) 그리고 부록(appendix)으로 구성되어 있습니다. preface와 appendix의 하위구조는 chapter의 경우와 같습니다.

그런데 appendix의 두번째 section에는 상당히 복잡한 구조의 informalequation이라는 기초요소가 보입니다. 이 것은 수식의 표현을 위한 기초요소인데 이에 대해서는 4.1.5절에서 자세히 설명하겠습니다.

한편, 위의 예에서는 DocBook 문서 내에서 주석(comment)을 사용하는 방법도 보여주고 있습니다. 주석은 <!-- --> 사이에 적어 넣어야 합니다. 물론 주석 처리된 내용은 변환된 문서에 전혀 나타나지 않게 됩니다.

2.3.5. set의 구조

다음은 set의 간단한 예입니다. 여러개의 book을 포함합니다.

예 **2-5. set**의 간단한 예

```
<!DOCTYPE set PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<set>

  <setinfo>
    <title>The Perl Series</title>
    <corpauthor>O'Reilly &amp; Associates, Inc.</corpauthor>
  </setinfo>

  <book>

    <bookinfo>
      <title>Learning Perl</title>
    </bookinfo>
    ...
  </book>

  <book>
```

```
  <bookinfo>
    <title>Programming Perl</title>
  </bookinfo>
  ...
</book>

<book>

  <bookinfo>
    <title>Advanced Perl Programming</title>
  </bookinfo>
  ...
</book>

</set>
```

book의 하위구조는 생략하였습니다.

주석

1. MS-Windows 2000에 포함된 메모장은 기본적으로 utf-8을 지원합니다. 그 밖에도 많은 에디터들이 MS-Windows 환경에서의 유니코드를 지원합니다. Unix 환경에서 쓸 수 있는 유니코드 에디터로는 yudit (http://www.yudit.org/) 등이 있습니다.
2. DocBook 지원 기능을 갖춘 편집기에 대한 상세한 내용은 아직 준비 중입니다.
3. 이 밖에 매개변수 실체요소라는 것이 있습니다. 이에 대해서는 예 C-2에서 간략히 설명합니다.
4. 이에 대해서는 B.1절에서 자세히 설명합니다.
5. <para>...</> - 이런 생략은 SGML에선 가능하지만 XML에선 허용되지 않습니다.
6. 이에 대해서는 B.1절에서 자세히 설명합니다.
7. 한국어 지원이 포함된 Modular DocBook Stylesheet V1.59 이상을 사용해야만 합니다.

3장. 문서 변환하기

3.1. 준비

완성된 DocBook 문서를 다른 포맷으로 변환하기 위해서는 몇가지 사전 준비가 필요합니다.

3.1.1. 필요한 파일들

우선, DocBook DTD가 기본적으로 있어야 합니다. DocBook DTD는 DocBook 공식 홈페이지 (http://www.oasis-open.org/docbook)에서 받을 수 있습니다. 변환할 문서가 사용하고 있는 DocBook 버전에 맞는 DTD가 필요합니다.

또한 DocBook DTD는 `ISO entity set`을 필요로 합니다. 이것은 oasis-open (http://www.oasis-open.org/cover/ISOEnts.zip)에서 받을 수 있습니다. 이 파일들은 DocBook DTD에 포함된 카탈로그 파일의 내용에 맞게 그 이름을 고쳐서 같은 디렉토리 안으로 복사해 넣어야 합니다. 카탈로그 파일에 대해서는 B.1절에서 상세히 설명합니다.

그리고 DocBook 문서에 적용시킬 스타일시트가 필요합니다. 현재 DocBook DTD를 위한 스타일시트로는 Normal Walsh의 Modular DocBook Stylesheet가 거의 유일합니다. 그 밖의 몇가지 다른 스타일시트들은 모두가 Normal Walsh의 것을 약간 변형시킨 것에 불과합니다. Modular DocBook Stylesheet는 DocBook Open Repository (http://sourceforge.net/projects/docbook)에서 받을 수 있습니다. 특히, 제대로 된 한글 문서를 만들기 위해서는 한글 지역화가 추가된 버전 1.59 이상의 것을 사용해야 합니다.

한편, 좀더 깔끔하고 완성도 높은 한글 문서를 만들기 위해서는 KLDP 스타일시트를 사용해야만 합니다. 이것은 Modular DocBook Stylesheet 위에서 돌아가는 수정된 스타일시트이며 Modular DocBook Styleshee의 한글 지역화도 KLDP 스타일시트와 함께해야만 그 기능을 제대로 발휘할 수 있습니다. KLDP 스타일시트는 KLDP CVS (http://cvs.kldp.org/cgi-bin/cvsweb/docs/KLDPstyle/)에서 구할 수 있으며 자세한 사용법에 대해서는 리눅스 한글문서 프로젝트 (http://kldp.org)와 열린 글짓기 홈페이지 (http://docs.kldp.org)를 함께 참고하시기 바랍니다.

마지막으로, DTD와 스타일시트를 분석하여 다양한 포맷의 문서를 생성시켜 줄 프로그램이 필요합니다. 현재 이런 프로그램으로는 James Clark의 Jade가 거의 유일합니다. 다른 프로그램들은 Jade를 좀더 편리하게 사용할 수 있도록 도와주는 데 불과한 경우가 대부분입니다. Jade는 James Clark's Home Page (http://jclark.com)에서 받을 수 있습니다. jade는 Unix 환경은 물론 MS-Windows에서도 사용이 가능합니다.

최신의 리눅스 배포본에는 KLDP 스타일시트를 제외한 모든 구성요소들이 이미 패키지로 만들어져 있습니다. 이런 패키지들을 설치한 경우라면 각각의 파일들에 대해서 특별히 신경쓸 필요가 없을 것입니다. [1]

3.1.2. SGML_CATALOG_FILES의 설정

DocBook 문서 변환을 준비하는 데 있어 중요한 것이 SGML_CATALOG_FILES 환경 변수의 설정입니다.

카탈로그(CATALOG) 파일에는 DocBook 문서를 처리하는 데 필요한 여러가지 파일들의 상세한 위치가 적혀 있는데, 환경 변수 SGML_CATALOG_FILES는 바로 이 카탈로그 파일의 위치를 알려주는 환경 변수입니다.

Unix 환경에서 bash 쉘을 사용하는 경우라면, SGML_CATALOG_FILES는 보통 다음과 같은 방식으로 설정해 줍니다. 카탈로그 파일의 구체적인 위치는 시스템마다 다를 수 있으므로 확인이 필요합니다.

```
$ export SGML_CATALOG_FILES="/usr/lib/sgml/catalog"
```

MS-Window 환경에서는 다음과 같은 식으로 설정해 줍니다. 물론 카탈로그 파일의 구체적인 위치는 각자 다를 것입니다.

```
> set SGML_CATALOG_FILES=c:\docbk41\docbook.cat
```

만일 DocBook DTD와 Jade 패키지를 각각 따로 받아 설치한 경우라면, 각각에 포함된 카탈로그 파일의 경로를 모두 알려 주어야 합니다. MS-windows 환경의 경우라면 다음과 같은 식으로 해야 할 것입니다.

```
> set SGML_CATALOG_FILES=c:\docbk41\docbook.cat;c:\jade\catalog
```

사실, 잘 만들어진 리눅스 패키지로 파일들을 설치한 경우라면 SGML_CATALOG_FILES 설정 자체가 필요 없을 수도 있습니다. 좋은 리눅스 배포본들은 모든 sgml 관련 카탈로그 파일들이 하나로 통합되어 있으며, 그 위치를 기본값으로 설정하여 컴파일된 Jade 패키지를 제공하기 때문입니다.

3.2. Jade 사용하기

DocBook 문서를 다른 형식의 문서로 변환하는 데는 Jade를 사용합니다. 현재 `HTML`, `RTF`, `TeX`, `DVI`, `PS` 형식의 문서로 잘 변환됩니다.

Jade 패키지에는 문서 작업에 관련된 몇가지 유틸리티와 설명서도 들어 있습니다.

3.2.1. SP_ENCODING의 설정

Jade가 한글을 올바로 처리하도록 하기 위해서는 우선 SP_ENCODING 환경변수를 정확히 설정해 주어야 합니다.

DocBook 문서를 euc-kr 인코딩으로 작성했다면 SP_ENCODING의 값은 `euc-kr`로 지정해 주어야 합니다. utf-8 인코딩으로 작성했다면 SP_ENCODING의 값은 `utf-8`로 지정합니다.

다음은 MS-Windows 환경에서 SP_ENCODING의 값을 `euc-kr`로 지정하는 방법을 보여줍니다.

```
> set SP_ENCODING=euc-kr
```

3.2.2. 문법 검사하기

DocBook으로 쓰여진 문서의 문법을 검사하는 데는 nsgmls를 사용합니다. nsgmls는 Jade 패키지에 포함되어 있습니다.

nsgmls는 문서 종류 선언부에 지정된 DTD가 어디에 있는지를 카탈로그 파일을 통해 알아냅니다. 그리고 그 DTD를 기준으로 삼아 주어진 파일의 문법이 틀린 데가 없는지 검사합니다.

```
$ nsgmls -s test.sgml
```

`s` 옵션은 nsgmls가 에러메시지만을 출력하도록 하기 위한 것입니다. 작성한 문서에 아무런 문제가 없다면 nsgmls은 그저 조용히 종료될 것입니다.

nsgmls가 카탈로그 파일을 읽어들어는 과정에서 다음과 같은 에러를 연달아 뿜어내는 경우가 있습니다. 이것은 nsgmls 패키지가 `DTDDECL`이라는 카탈로그 지시어를 지원하지 않기 때문입니다. 그러므로 그냥 무시하면 됩니다.

```
> nsgmls: DTDDECL catalog entries are not supported
```

3.2.3. Jade의 사용법

Jade의 기본적인 사용법은 다음과 같습니다. MS-Windows의 경우를 예로 들었습니다.

```
> jade -t sgml -d c:\docbook\html\docbook.dsl test.sgml
```

`t` 옵션에서는 어떤 형식의 문서로 변환할 것인지를 지정합니다. 위의 예에서는 `sgml`로 지정되어 있습니다. `d` 옵션에서는 적용할 스타일시트의 정확한 위치를 지정합니다. 시스템마다 스타일시트들을 보관하는 디렉토리의 위치가 다르므로 확인이 필요합니다. `Debian GNU/Linux` 같은 경우는 그 위치가 `/usr/lib/sgml/stylesheet/dsssl/docbook/nwalsh/html/docbook.dsl`입니다.

유용한 옵션으로 `V` 옵션이 있습니다 이 것을 사용하면 해당 스타일시트에 미리 정의되어 있는 내용들에 변화를 줄 수 있습니다. 예를 들어 Modular DocBook Stylesheet는 최상위 기초요소가 `article`인 경우에 차례를 만들어 넣지 않는 것을 기본으로 하고 있습니다. 이럴 때 `-V %generate-article-toc%` 옵션을 주게 되면 `article`에서 차례를 볼 수 있게 됩니다. [2]

3.3. HTML로의 변환

DocBook 문서는 아마 html 형식의 문서로 변환되는 경우가 가장 많을 것입니다. html 파일로 변환된 문서는 웹 브라우저로 간편히 볼 수 있습니다.

html로의 변환 방법은 이미 앞서 보여진 예에 나타난 것과 같습니다.

```
> jade -t sgml -d c:\docbook\html\docbook.dsl test.sgml
```

html 문서란 HTML DTD를 사용하는 sgml 문서입니다. 따라서 문서의 형식은 sgml로 해야합니다. 대신 스타일시트는 html 디렉토리에 있는 것을 사용합니다.

KLDP 스타일시트를 사용한다면 다음과 같이 해야 합니다. [3]

```
> jade -t sgml -i html -d kldp.dsl#html article.sgml
```

HTML로 변환하기 전에 SP_ENCODING 환경 변수가 제대로 설정되어 있는지 확인해야 합니다. 이 변수가 제대로 설정되어 있지 않으면 스타일시트의 한글 지역화로부터 삽입되는 한글들이 제대로 출력되지 않습니다. 즉, '차례' '다음' '이전' 등의 한글 지역화가 &#ddddd; 형식의 특수문자 실체요소 형식으로 표현되어 lynx나 w3m 등의 텍스트 브라우저에서 문제를 일으킵니다.

보통, 출력될 html 문서의 파일명을 특별히 지정할 필요가 있을 것입니다. 이때는 `book` `article` `chapter` `section` 기초요소의 여는 태그뒤에 다음과 같은 태그를 덧붙여 주면 됩니다.

```
<book lang=ko><?dbhtml filename="MyBook.html">
```

KLDP 스타일시트에서는 각 기초요소의 id 속성이 파일이름이 되도록 되어 있으므로 굳이 ?dbhtml 형식을 사용할 필요는 없습니다. [4]

3.4. RTF로의 변환

RTF 파일은 MS-Windows에 기본적으로 포함된 워드패드 프로그램이나, MS-Word으로 읽고 편집할 수 있습니다. RTF로 변환된 파일은 인쇄용으로 손색이 없습니다.

RTF 파일로는 다음과 같이 변환합니다. `print` 디렉토리에 있는 스타일시트를 사용한다는 점을 눈여겨 보기 바랍니다.

```
> jade -t rtf -d c:\docbook\print\docbook.dsl test.sgml
```

KLDP 스타일시트를 사용한다면 다음과 같이 합니다.

```
> jade -t rtf -d kldp.dsl#print article.sgml
```

변환하기 전에 SP_ENCODING 환경 변수가 올바로 지정되어 있어야 합니다. 이 변수가 제대로 잡혀 있지 않으면 한글에 이상한 군더더기가 붙어 출력됩니다.

3.5. TeX, DVI, PS로의 변환

본격적인 인쇄물이 필요한 경우라면, TeX 형식의 문서로 변환하는 것이 좋습니다. TeX은 뛰어난 기능을 가진 문서 조판 언어입니다. TeX 형식의 파일에서 실제로 인쇄물을 얻기 위해서는, 다시 DVI, PS로의 변환이 필요합니다.

변환하기 전에 우선 SP_ENCODING 환경 변수가 올바로 잡혀 있어야 합니다. 그렇지 않으면 한글 지역화로부터 삽입되는 한글들이 깨져 표현됩니다.

우선 TeX 문서로의 변환은 다음과 같이 합니다.

```
> jade -t tex -d c:\docbook\print\docbook.dsl test.sgml
```

KLDP 스타일시트를 사용한다면 다음과 같이 합니다.

```
> jade -t tex -d kldp.dsl#print article.sgml
```

변환된 TeX 문서는 Jadetex으로 컴파일하여야 합니다. Jadetex은 Unix에서 tetex 패키지를 설치하거나 MS-Windows에서 fptex을 설치한 경우라면 같이 포함되어 있을 것입니다. Jadetex은 LaTeX 위에서 돌아가는 매크로입니다.

Jadetex이 한글을 처리할 수 있도록 하려면 우선 hlatex 패키지가 함께 설치되어 있어야 합니다. hlatex 설치에 관해서는 한글 latex 홈페이지 (ftp://ftp.linguistik.uni-erlangen.de/pub/HLaTeX/hlatex-ko.html)를 참고하기 바랍니다.

한편, Jade가 생성해낸 TeX 문서에는 한글이 유니코드 형식으로 표현되어 있습니다. 그런데 아직 hlatex 패키지는 유니코드를 다루지 못하므로, 유니코드 형식으로 표현된 글자들을 다시 KSC 5601에 정의된 글자들로 변환하여 hlatex에 넘겨주어야 합니다. 이 때 필요한 파일이 hcharacters.sty (http://docs.kldp.org/Stuff/hcharacters.zip)입니다. 이 파일을 `/usr/share/texmf/tex/jadetex/base` 아래에 복사해 넣어둡니다. TeX이 설치된 위치는 시스템마다 다를 수 있으므로 확인이 필요합니다. MS-Windows에 fptex가 설치된 경우에는 `c:\tex\texmf-local\tex\jadetex` 또는 `c:\tex\texmf\tex\jadetex`와 같은 디렉토리에 복사해 넣어둡니다. 물론 TeX이 설치된 위치는 각자 다를 것이므로 확인이 필요합니다. 파일을 복사해 넣은 후에는 꼭 **mktexlsr** 명령을 한번 실행해서 TeX에게 새로 설치된 파일이 있음을 알려주어야 합니다.

> **주의**
>
> 아직 hlatex 패키지는 KSC 5601에 정의된 문자 밖에는 다루지 못합니다. 따라서 DocBook 문서를 utf-8로 작성하였을 경우에 KSC 5601에 정의되지 않은 한글을 사용하였다면 그 글자는 제대로 표현되지 않습니다. 대신 그 자리에는 해당 유니코드의 십진수 값이 인쇄됩니다.

이제 Jade가 생성한 TeX 문서를 편집기로 열어 `\usepackage{hfont}` `\usepackage{hcharacters}` 이 두줄을 파일의 맨 처음에 삽입합니다. [5] 이 것은 jadetex에게 hfont.sty와 hcharacters.sty를 사용하여 한글을 처리하도록 지시합니다.

```
\usepackage{hfont}
\usepackage{hcharacters}
\FOT{2}\Seq%
{\def\HeadingLevel%
...
```

준비가 다 되었습니다. 드디어 Jadetex으로 DVI 파일을 생성합니다. 생성된 .dvi 파일은 xdvi나 windvi로 보고 인쇄할 수 있습니다. 그러나 DVI 파일에는 글꼴이 내장되어 있지 않으므로 dvi 프로그램과 글꼴이 제대로 설치된 사람만 파일을 보고 인쇄할 수 있습니다.

```
> jadetex test.tex
```

작은 정보: Jadetex은 꼭 세번 반복해서 실행시켜야 합니다. 이렇게 하지 않으면 차례의 페이지 번호가 제대로 표현되지 않습니다.

DVI 파일은 다시 포스트스크립트(PS) 파일로 변환할 수 있습니다. PS 파일에는 글꼴이 내장되어 있으므로 누구나 볼 수 있고 인쇄할 수 있습니다. PS 파일은 gv나 gsview32[6], 또는 Photoshop 등으로도 보거나 인쇄할 수 있습니다. 변환에 사용되는 dvips 프로그램은 TeX 패키지에 포함되어 있습니다.

```
> dvips -o test.ps -O -0.5cm,1cm test.dvi
```

소문자 o 옵션에서는 출력될 PS 파일의 이름을 정해 줍니다. 이 옵션을 지정해주지 않으면 PS 파일이 그대로 프린터에 출력되므로 주의해야 합니다. 대문자 O 옵션은 출력될 PS 파일의 여백을 조정해 줍니다. 문서가 한쪽으로 치우쳐 출력된다면 이곳에 적당한 값을 넣어줍니다. 옵션은 x,y 좌표로 표현하며 cm나 inch 단위를 쓸 수 있습니다. 마이너스 부호도 사용 가능합니다.

주석

1. DocBook 문서 변환을 위해 필요한 파일들을 설치하는 보다 자세한 방법에 대해서는 아직 준비 중입나다.
2. `v` 옵션의 다양한 사용방법에 대해서는 아직 준비 중입니다.
3. KLDP 스타일시트는 Modular DocBook 스타일시트에 기반하고 있으므로 그 위치를 정확히 알려주어야 합니다. kldp.dsl을 열어서 다음과 같은 부분을 자신에 맞게 수정하여 스타일시트의 위치를 알려줍니다.

   ```
   <!ENTITY % print "IGNORE">
   <!ENTITY docbook.dsl SYSTEM
   ```

```
"/usr/lib/sgml/stylesheet/dsssl/docbook/nwalsh/html/docbook.dsl"
CDATA dsssl>
]]>
<!ENTITY % print "INCLUDE">
<![%print;[
<!ENTITY docbook.dsl SYSTEM
"/usr/lib/sgml/stylesheet/dsssl/docbook/nwalsh/print/docbook.dsl"
CDATA dsssl>
]]>
```

여기서 '/usr/lib/sgml/stylesheet/dsssl/docbook/nwalsh/html/docbook.dsl' 부분과 '/usr/lib/sgml/stylesheet/dsssl/docbook/nwalsh/print/docbook.dsl' 부분을 자신의 시스템에 설치된 Modular DocBook 스타일시트의 경로에 맞게 수정합니다.

4. KLDP 스타일시트에 대해서는 열린 글짓기 홈페이지 (http://docs.kldp.org)를 참고하세요.
5. 이것은 `\usepackage{hfont,hcharacters}` 이렇게 한 줄로 써도 됩니다.
6. 이것은 gv의 MS-Windows 버전입니다.

4장. 더욱 다채로운 기능들

참고: 이 장의 내용은 DocBook HOWTO (http://www.ibiblio.org/godoy/)의 'Writing with DocBook elements'를 기반으로 하여 DocBook: The Definitive Guide (http://www.docbook.org)의 'Reference'를 참고해 씌어졌습니다.

4.1. 흔히 쓰이는 기초요소들

앞에서는 문서의 전체 구조와 이런 구조를 위한 기초요소들을 주로 살펴보았습니다. 이제는 DocBook의 세부적인 기초요소들 중 일반적으로 흔히 쓰이는 것 몇가지에 대해서 알아보겠습니다. 이런 기초요소들은 종류에 따라 특정 문맥 사이에서만 사용이 가능한 경우도 있다는 점을 주의하기 바랍니다.

작은 정보: 작성된 sgml 문서를 어떤 종류의 문서로 변환하느냐에 따라, 각 마크업이 적용된 모습은 조금씩 다르게 나타날 수 있습니다. 그러므로 원하는 종류의 문서로 변환된 후의 모습이 어떠한지를 꼭 확인해 볼 필요가 있습니다.

4.1.1. 지은이에 대한 정보

author

지은이에 대한 정보를 넣기 위해 사용됩니다. 이 태그 사이에는 다음과 같이 여러 정보가 올 수 있습니다.

```
<!DOCTYPE author PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

  <author lang="ko">
    <surname>홍</surname>
    <firstname>길동</firstname>

    <contrib>도적질을 총지휘하였습니다.</contrib>

    <affiliation>
      <jobtitle>의적 두목</jobtitle>
      <orgname>의적단</orgname>
      <orgdiv>수뇌부</orgdiv>

      <address>
        <email>gildong@honggildongzzang.com</email>
      </address>

    </affiliation>
```

```
</author>
```

editor

편집자를 위한 기초요소로서 author와 같은 위치(articleinfo나 bookinfo 안)에서 같은 구조(surname, firstname, affiliation)를 가집니다.

othercredit

이것은 지은이나 편집자는 아니지만 글의 완성에 크게 기여한 사람들을 위한 기초요소입니다. othercredit도 author와 같은 위치(articleinfo나 bookinfo 안)에서 같은 구조(surname, firstname, affiliation)를 가집니다.

특별히 번역자임을 알려주기 위해선 othercredit의 role 속성을 translator로 주도록 합니다. 문서를 변환하는데 기여한 경우에는 role 속성을 converter로 주면 되겠습니다.

```
<othercredit role="translator">
```

firstname surname othername

이름을 넣기 위해 사용됩니다. `firstname`에 이름 `surname`에 성이 들어가야 합니다. `othername`에는 별명이나 세례명 등을 넣을 수 있습니다.

Modular DocBook Stylesheet의 예전 버전을 사용하면 lang 속성과 상관없이 성과 이름의 순서가 서양식으로 뒤바뀌어 표현되는 문제가 있었습니다. 그러나 스타일시트 버전 1.59부터 한국어 지역화가 포함되면서 이 문제는 해결되었습니다.

최상위 기초요소의 lang 속성을 'ko'로 주었을 때, 성과 이름의 순서가 우리와 다른 서양식 이름을 사용하는 경우에는 상위 기초요소(author, editor, othercredit)의 lang 속성을 'en'으로 주어야 이름이 서양식으로 표현됩니다.

address

주소를 넣기 위해 사용됩니다. 보통 `email`을 사용하여 전자우편 주소를 넣습니다.

email

전자우편 주소를 넣기 위해 사용됩니다. 이는 다음과 같이 나타납니다.

`<address@domain>`

affiliation

소속을 나타냅니다.

jobtitle orgname orgdiv

각각 직책의 이름, 부서의 이름, 조직의 이름을 나타냅니다.

contrib

author, editor, othercredit 각각의 사람들이 구체적으로 어떤 기여를 하였는지 적어줍니다.

4.1.2. 글의 세부적인 속성

citation

참고문헌을 위한 것입니다. 다음과 같이 나타납니다.

[reference]

blockquote

인용문을 위한 것입니다.

```
<blockquote>
  <attribution>Text Author</attribution>
  <para>Quote Text.</para>
</blockquote>
```

다음과 같이 나타납니다.

Text Author

Quote Text.

emphasis

강조를 위한 것입니다. 다음과 같이 나타납니다.

text

보통 강조는 기울인 글씨체(italic)나 굵은 글씨체(bold)로써 하는 것이 일반적입니다. 그러나 DocBook에서는 기울인 글씨체를 사용한 강조가 기본으로 되어 있습니다. 굵은 글씨체를 사용해서 강조를 하려면 스타일시트에 추가 설정을 하여야만 합니다. KLDP 스타일시트에서는 굵은 글씨체가 기본값으로 되어 있습니다.

footnote

각주(페이지의 아래에 달리는 주석)를 달기 위한 것입니다.

```
<para>
  각주의 예를 보고 싶으세요?

    <footnote>
      <para>
        이것은 각주의 예로서 달아본 주석입니다.
      </para>
    </footnote>
```

```
</para>
```

이것은 다음과 같이 나타납니다.

- 각주의 예를 보고 싶으세요? [1]

ulink

이것은 URL 주소를 써넣기 위한 것입니다.

```
<ulink url="http://kldp.org">리눅스 한글문서 프로젝트</ulink>
```

다음과 같이 나타납니다.

- 리눅스 한글문서 프로젝트 (http://kldp.org)

xref

참조를 위한 것입니다. 해당 id 속성이 지정된 기초요소를 참조하게 됩니다.

```
<section id="inserting-pictures">
 ...
</section>

<section id="example">
 ...
  <para>
          그림 삽입에 대한 더욱 자세한 내용은
    <xref linkend="inserting-pictures">을 참고하기 바랍니다.
  </para>
</section>
```

다음과 같이 나타납니다.

- 그림 삽입에 대한 더욱 자세한 내용은 4.2절을 참고하기 바랍니다.

4.1.3. 나열하기

itemizedlist

항목들이 번호가 붙지 않은 상태로 나열됩니다.

```
<itemizedlist>

  <listitem>
    <para>item</para>
  </listitem>

  <listitem>
```

```
    <para>item</para>
  </listitem>

</itemizedlist>
```

다음과 같이 나타납니다.

- item
- item

orderedlist

번호가 매겨져 나열됩니다.

```
<orderedlist>

  <listitem>
    <para>item</para>
  </listitem>

  <listitem>
    <para>item</para>
  </listitem>

</orderedlist>
```

다음과 같이 나타납니다.

1. item
2. item

segmentedlist

항목 첫머리에 제목이 순차적으로 나타납니다.

```
<segmentedlist>
  <title>Binary to decimal conversion</title>
    <segtitle>Binary</segtitle>
    <segtitle>Decimal</segtitle>

    <seglistitem>
      <seg>00</seg>
      <seg>0</seg>
    </seglistitem>

    <seglistitem>
      <seg>01</seg>
```

```
      <seg>1</seg>
    </seglistitem>

    <seglistitem>
      <seg>10</seg>
      <seg>2</seg>
    </seglistitem>

  </segmentedlist>
```

이것은 다음과 같이 나타납니다.

Binary to Decimal Conversion

Binary: 00
Decimal: 0

Binary: 01
Decimal: 1

Binary: 10
Decimal: 2

variablelist

각 항목의 제목이 따로 분리되어 나타납니다.

```
<variablelist>

  <varlistentry>
    <term>Entry 1</term>

    <listitem>
      <para>Description</para>
    </listitem>

  </varlistentry>

  <varlistentry>
    <term>Entry 2</term>

    <listitem>
      <para>Description</para>
    </listitem>

  </varlistentry>

</variablelist>
```

이것은 다음과 같이 나타납니다.

Entry 1

Description

Entry 2

Descripton

simplelist

단순히 열을 맞추어 나열됩니다. 속성에 type과 columns가 지정된 것을 눈여겨 보기 바랍니다.

```
<simplelist type="horiz" columns="3">
  <member>1</member>
  <member>2</member>
  <member>3</member>
  <member>4</member>
  <member>5</member>
  <member>6</member>
</simplelist>

<simplelist type="inline">
  <member>A</member>
  <member>B</member>
  <member>C</member>
  <member>D</member>
  <member>E</member>
  <member>F</member>
</simplelist>
```

이것은 다음과 같이 나타납니다.

1	2	3
4	5	6

A, B, C, D, E, F

4.1.4. 컴퓨터에 관련된 속성

keycap

키의 이름을 나타낼 때 사용합니다. 이는 다음과 같이 나타납니다.

F1

keycode

키의 코드를 나타낼때 사용합니다.

keycombo

키를 누르는 조합을 나타낼 때 사용합니다.

```
<keycombo>
  <keycap>Ctrl</keycap>
  <keycap>S</keycap>
</keycombo>
```

이는 다음과 같이 나타납니다.

Ctrl-S

guimenu

GUI 프로그램의 메뉴를 나타낼 때 사용합니다.

guimenuitem

GUI 프로그램 메뉴의 세부항목을 나타낼 때 사용합니다.

menuchoice

메뉴를 어떤 순서로 선택해야 하는지를 나타낼 때 사용합니다.

```
<menuchoice>

  <shortcut>
    <keycombo><keycap>Ctrl</keycap><keycap>S</keycap></keycombo>
  </shortcut>

  <guimenu>파일</guimenu><guimenuitem>저장</guimenuitem>

</menuchoice>
```

다음과 같이 나타납니다

파일→저장 (**Ctrl-S**)

mousebutton

마우스 버튼의 이름을 나타냅니다.

`<mousebutton>left</mousebutton>`

command

명령어를 나타냅니다. 다음과 같이 나타납니다.

command

application

응용프로그램의 이름을 나타냅니다. 이는 다음과 같이 나타납니다.

application

filename

파일의 이름을 나타냅니다.

```
<filename>filename</filename>
```

특별히 디렉토리의 이름을 나타낼 경우에는 다음과 같이 합니다.

```
<filename id="directory">directory</filename>
```

4.1.5. 수식의 표현

equation

DocBook 자체만으로는 수식의 표현이 취약합니다. 새로운 버전의 DocBook에서는 MathML과 결합하여 수식을 표현할 수 있다고 합니다. 일단은 복잡한 수식을 깔끔히 나타내기 위해 그림 파일을 사용할 수 있습니다. equation은 수식을 그림으로 대체하고 그 그림을 간단히 설명할 수 있도록 해줍니다.

참고: 그림 파일을 위해 사용된 `mediaobject`와 `imageobject`에 대해서는 4.2절에서 설명합니다.

> **주의**
>
> DocBook v3.1 버전까지 그림 삽입을 위해 사용되었던 `graphic` 기초요소는 V4.0 버전부터는 사용하지 않는 것이 좋습니다. V5.0에서는 `graphic` 기초요소 자체가 사라지게 될 것입니다.

다음과 같이 사용합니다.

```
<!DOCTYPE equation PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<equation>
  <title>페르마의 마지막 정리</title>
  <alt>x^n + y^n &ne; z^n &forall; n &ne; 2</alt>

  <mediaobject>
    <imageobject>
      <imagedata fileref="fermat.eps" format="eps">
    </imageobject>
    <imageobject>
      <imagedata fileref="fermat.gif" format="gif">
    </imageobject>
```

```
      <imageobject>
        <imagedata fileref="fermat.bmp" format="bmp">
      </imageobject>
      <textobject>
        <phrase>x^n + y^n ≠ z^n ∀ n ≠ 2</phrase>
      </textobject>
    </mediaobject>

  </equation>
```

다음과 같이 나타납니다.

수식 **4-1.** 페르마의 마지막 정리

informalequation

equation과 같지만 title을 쓰지 않고 수식만을 표현합니다.

inlineequation

equation과 같지만 수식을 문장에 포함시켜 보여줍니다. 제목은 사용하지 않습니다.

참고**:** 그림 파일을 위해 사용된 inlinemediaobject와 imageobject에 대해서는 4.2절에서 설명합니다.

> **주의**
>
> DocBook v3.1 버전까지 문장 중간에 그림을 넣기 위해 사용되었던 inlinegraphic 기초요소는 V4.0 버전부터는 사용하지 않는 것이 좋습니다. V5.0에서는 inlinegraphic 기초요소 자체가 사라지게 될 것입니다.

```
<!DOCTYPE para PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<para>
  아인슈타인의 상대성 이론에서 다음 식이 가장 유명합니다:

  <inlineequation>
    <alt>e=mc^2</alt>
```

```
    <inlinemediaobject>
      <imageobject>
        <imagedata fileref="emc2.eps" format="eps">
      </imageobject>
      <imageobject>
        <imagedata fileref="emc2.gif" format="gif">
      </imageobject>
      <imageobject>
        <imagedata fileref="emc2.bmp" format="bmp">
      </imageobject>
      <textobject>
        <phrase>e=mc^2</phrase>
      </textobject>
    </inlinemediaobject>

  </inlineequation>

</para>
```

다음과 같이 나타납니다.

아인슈타인의 상대성 이론에서 다음 식이 가장 유명합니다:

superscript subscript

어깨 글자와 아래 글자를 넣을 때 사용합니다.

```
H<subscript>3</subscript>O<superscript>+</superscript>
```

다음과 같이 나타납니다.

H_3O^+

4.1.6. 기타

Glossary

용어 해설을 하기 위한 것입니다.

```
<glossary>
  <glossentry>
    <glossterm>어떤 용어</glossterm>

    <glossdef>
```

```
      <para>여기에 용어 해설을 씁니다.</para>
    </glossdef>

  </glossentry>
</glossary>
```

이의 실례는 이 글의 *용어 해설* 부분을 보기 바랍니다.

4.2. 그림 넣기

출판을 하기 위한 문서라면 그림은 거의 필수적으로 들어가야만 합니다.

TeX 형식을 사용하는 경우에는 포스트스크립트 이미지가 필요하며, HTML의 경우라면 브라우저가 인식할 수 있는 JPEG, GIF, PNG 같은 이미지가 필요합니다. RTF로 변환하는 경우라면 PNG 이미지가 좋습니다.

주의

DocBook V3.1까지는 그림을 넣기 위해 `<graphic>`을 fileref 속성과 함께 사용할 수가 있었습니다. 그러나 V5.0에는 아예 이들 기초요소 자체가 사라질 것이므로, `graphic`과 `inlinegraphic`는 더 이상 사용하지 않는 것이 좋습니다. 다만 V3.1 이하를 사용하는 문서를 이해하기 위해 아래에 한가지 예를 보입니다.

예 **4-1.** 그림 넣기

```
<!DOCTYPE figure PUBLIC "-
//OASIS//DTD DocBook V3.1//EN">

<figure>
  <title>그림 제목</title>
  <graphic fileref="images/file"></graphic>
</figure>
```

V4.0과 4.1에서 굳이 사용한다면 다음과 같이 여는 태그만을 사용해야 합니다.

예 **4-2.** 그림 넣기

```
<!DOCTYPE figure PUBLIC "-
//OASIS//DTD DocBook V4.1//EN">

<figure>
  <title>그림 제목</title>
  <graphic fileref="images/file">
</figure>
```

다음과 같이, 여러가지 형식의 그림 파일들은 `<imageobject>`에 의해 포장됩니다. 이렇게 하면 문서를 변환할 때 해당 문서 형식에 맞는 그림 파일을 골라내어 쓸 수 있게 됩니다.

예 **4-3. `<imageobject>`**의 사용법

```
<!DOCTYPE figure PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<figure>
  <title>그림의 제목</title>

  <mediaobject>

    <imageobject>
      <imagedata fileref="images/file.eps" format="eps">
    </imageobject>

    <imageobject>
      <imagedata fileref="images/file.jpg" format="jpg">
    </imageobject>

    <textobject>
      <phrase>이 곳에는 예를 들기 위한 그림이 있습니다</phrase>
    </textobject>

    <caption>
      <para>그림에 대한 설명(안 넣어도 됩니다)</para>
    </caption>

  </mediaobject>

</figure>
```

사용 가능한 파일 형식은 다음과 같습니다. BMP, CGM-BINARY, CGM-CHAR, CGM-CLEAR, DITROFF, DVI, EPS, EQN, FAX, GIF, GIF87A, GIF89A, IGES, JPEG, JPG, LINESPECIFIC, PCX, PIC, PS, SGML, TBL, TEX, TIFF, WMF, WPG.

이런 방식을 사용하면 프로그램을 사용해 문서를 처리하기가 훨씬 수월해 집니다. `<imageobject>`는 적당한 파일이 나올 때까지 계속 테스트되며, 만일 사용 가능한 파일이 하나도 없는 경우에는 `<textobject>`가 사용됩니다. DocBook `5.0`이 나오게 되면 예 4-3과 같은 방식만이 사용되며, `<graphic>` 기초요소는 아예 없어질 것이라고 합니다.

`<figure>`는 그림의 표현 양식을 지정해줍니다. `<figure>` 대신에 `<informalfigure>`를 사용한다면 그림 제목은 안 달아도 됩니다.

`<figure>`에는 float라는 속성이 있습니다. 이 속성을 `0`으로 지정하게 되면 그림의 위치가 문서 상에서 고정되어 정확히 원래 위치에서만 나타나게 됩니다. 그러나 이 속성이 `1`로 지정된다면 그림이 좀더 보기 좋은 위치를 찾아 자리를 잡을 수 있게 됩니다(이 위치는 사용하는 스타일시트에 의해 결정됩니다).

4.3. 표

표를 사용하면 복잡한 정보를 효과적으로 표현할 수 있습니다.

작고 간단한 표를 나타내기 위해서는 4.1.3절에서 설명한 `<simplelist>`를 사용하는 것이 쉽고 깔끔합니다. 물론, DocBook에는 좀더 틀을 갖춘 표를 만들기 위한 기초요소 `<table>`도 준비되어 있습니다. [2]

예 4-4은 `<table>`을 사용해 만든 가장 간단한 표의 실례입니다. `<tgroup cols="6">`에서 열의 개수를 미리 지정해 두고 있는 점을 주의하기 바랍니다. 행의 개수는 `<row>`를 추가하기만 하면 됩니다.

예 **4-4.** 간단한 표

```
<!DOCTYPE table PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

   <table>
     <title>표준곡선을 그리기 위한 흡광도 측정</title>

     <tgroup cols="6">
       <tbody>

         <row>
           <entry>Hemolysis</entry>
           <entry>12.5</entry>
           <entry>25</entry>
           <entry>50</entry>
           <entry>75</entry>
           <entry>100</entry>
         </row>

         <row>
           <entry>O.D</entry>
           <entry>0.215</entry>
           <entry>0.439</entry>
           <entry>0.840</entry>
           <entry>1.08</entry>
           <entry>1.50</entry>
         </row>

       </tbody>
     </tgroup>

   </table>
```

표 **4-1.** 표준곡선을 그리기 위한 흡광도 측정

Hemolysis	12.5	25	50	75	100
O.D	0.215	0.439	0.840	1.08	1.50

다음과 같이 다채로운 모양의 표도 만들 수 있습니다.

예 **4-5.** 복잡한 표

```
<!DOCTYPE table PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

   <table frame="all">
     <title>Sample Table</title>

     <tgroup cols="5">
       <colspec colname="column1">
       <colspec colname="column2">
       <colspec colname="column3">
       <colspec colnum="5" colname="column5">
       <spanspec namest="column1" nameend="column2" spanname="span-
horiz" align="center">
       <spanspec namest="column2" nameend="column3" spanname="span-
horiz-vert" align="center">

       <thead>
         <row>
           <entry spanname="span-horiz">
              <foreignphrase>Span</foreignphrase> horizontal
           </entry>
           <entry>Heading 2</entry>
           <entry>Heading 3</entry>
           <entry>Heading 4</entry>
         </row>
       </thead>

       <tfoot>
         <row>
           <entry>Footing 1</entry>
           <entry>Footing 2</entry>
           <entry>Footing 3</entry>
           <entry>Footing 4</entry>
          <entry>Footing 5</entry>
        </row>
       </tfoot>

      <tbody>
         <row>
           <entry>Data11</entry>
           <entry>Data12</entry>
```

```
              <entry>Data13</entry>
              <entry>Data14</entry>
              <entry>Data15</entry>
            </row>
            <row>
              <entry>Data21</entry>
              <entry>Data22</entry>
              <entry>Data23</entry>
              <entry>Data24</entry>
              <entry morerows="1" valign="middle">
                 <foreignphrase>Span</foreignphrase> vertical
              </entry>
            </row>
            <row>
              <entry>Data31</entry>
              <entry spanname="span-horiz-
vert" morerows="1" valign="bottom">
                 <foreignphrase>Span</foreignphrase> duplo
              </entry>
              <entry>Data34</entry>
            </row>
            <row>
              <entry>Data41</entry>
              <entry>Data44</entry>
              <entry>Data45</entry>
            </row>
          </tbody>

        </tgroup>

      </table>
```

표 **4-2.** 복잡한 표

	Horizontal Span	**Heading 2**	**Heading 3**	**Heading 4**
Data11	Data12	Data13	Data14	Data15
Data21	Data22	Data23	Data24	Vertical Span
Data31	Double Span		Data34	
Data41	Data44	Data45		
Footing 1	Footing 2	Footing 3	Footing 4	Footing 5

4.4. 프로그램 코드 보여주기와 해설 목록 넣기

프로그램 코드를 보여주기 위해서는 `<programlisting>`을 사용합니다. 또한 컴퓨터 스크린

에 뿌려지는 내용을 보여주기 위해서는 <screen>을 사용하면 됩니다. 이 두 기초요소 안에서는 verbatim 환경이 적용되므로 소스상에서 띄어쓰기나 줄바꿈한 상태가 출력물에서도 그대로 나타나게 됩니다.

본격적인 컴퓨터 관련 문서를 작성하기 위해서는 프로그램의 코드에 해설을 효과적으로 덧붙여 줄 수 있는 기능이 필요합니다. DocBook은 프로그램의 코드를 문서에 삽입해주고, 그것을 코드의 특정 라인에 대한 해설 목록과 연계시켜주는 기능을 갖고 있습니다. 이 글의 B.1절에서 사용된 것이 바로 이 기능입니다.

사용된 코드는 다음과 같습니다. 만일 코드와 해설 목록을 연계해주는 기능이 필요없다면 <areaspec> 부분과 <calloutlist> 부분은 생략해도 됩니다.

```
<example id="sample-catalog">
  <title>카탈로그의 한 예</title>
  <programlistingco>
    <areaspec>
      <area coords="1" id="ex.catalogue.comment">
      <area coords="5" id="ex.catalogue.definition">
      <area coords="11" id="ex.catalogue.eof">
    </areaspec>
        <programlisting>
-- Catalogues for the Conectiva S.A. Style --

OVERRIDE YES

PUBLIC "-
//Conectiva SA//DTD books V1.0//EN" "/home/ldp/estilos/livros.dtd"

DELEGATE "-//OASIS" "/home/ldp/SGML/dtds/catalog.dtd"

DOCTYPE BOOK /home/ldp/SGML/dtds/docbook/db31/docbook.dtd

-- EOF --
        </programlisting>
    <calloutlist>
      <callout arearefs="ex.catalogue.comment">
        <para> 주석. 주석은 <quote>--</quote>로 시작과
        끝을 맺습니다. </para>
      </callout>
      <callout arearefs="ex.catalogue.definition">
        <para> 공식 식별이름
           <parameter class="option">"-
//Conectiva SA//DTD books V1.0//EN"</parameter>은
           시스템 식별이름
           <filename class="directory">/home/ldp/styles/books.dtd</filename>에 해
당된다는 뜻입니다. </para>
      </callout>
      <callout arearefs="ex.catalogue.eof">
        <para> 파일의 끝을 나타내는 주석입니다. </para>
```

```
          </callout>
        </calloutlist>
      </programlistingco>
    </example>
```

목록(list)들은 <example>나 <para> 없이도 문서에 그대로 삽입될 수 있습니다. 이것은 4.1.3절에 설명된 기초요소들도 마찬가지입니다.

<calloutlist>에 나열된 해설들은 <areaspec>에 지정된 위치에 따라 본문과 연계됩니다.

4.5. 색인 넣기

색인은 자동적으로 생성될 수 있습니다. 색인이 자동으로 생성되려면 문서에 적절한 마크업이 되어 있어야 합니다.

이런 마크업들을 외부 프로그램들이 처리하여 색인을 만들게 됩니다. 이런 프로그램들 중의 하나가 collateindex.pl (http://docs.kldp.org/Stuff/collateindex.pl)이라는 스크립트입니다. 이 프로그램을 사용하여 색인을 만드는 방법에 대해서는 C.1절를 보기 바랍니다.

색인들은 계층을 이루며 포개어집니다. 예 4-6에서 색인을 위한 마크업이 어떻게 작성되는지 볼 수 있습니다.

예 **4-6.** 색인을 만들기 위한 마크업

```
<indexterm>
  <primary>첫번째 계층</primary>
  <secondary>두번째 계층</secondary>
  <tertiary>세번째 계층</tertiary>
</indexterm>
```

특정한 장,절 기타 다른 곳을 지정하여서 색인을 만들 수도 있습니다. 이것은 zone 속성을 사용하면 가능합니다.

예 **4-7. zone** 속성의 사용

```
<section id="encoding-index">
  <title>색인 넣기</title>

  <indexterm zone="encoding-index">
    <primary>편집</primary>
    <secondary>색인</secondary>
  </indexterm>

  <para>
    색인은 자동적으로 생성될
    수 있습니다. 색인이 자동으로 생성되려면
    문서상에 적절한 마크업이 되어 있어야 합니다.
```

```
</para>
```

마크업에 속성 영역이 사용되었다면 그 색인 마크업은 문서의 어느 곳에 있어도 상관이 없습니다.

그러나 색인 마크업들을 알아보기 쉽도록 관리하려면, 색인이 참조하는 해당 부분 아래에 색인 마크업을 해두는 것이 좋습니다.

예 **4-8. `startofrange`와 `endofrange`을 속성으로 사용하는 방법**

```
<para>
  글을 쓰다 보면 이렇게
  <indexterm class="startofrange" id="example-band-index">
  <primary>examples</primary>
  <secondary>index</secondary>
  </indexterm>
  많은 양의 단락을 전부 지정하여 색인화할 필요가 있게 됩니다.
</para>
<para>
  이럴 때는 원하는 단락의 앞부분에서
  startofrange 속성을 사용하여 색인을 합니다.
</para>
<para>
  그리고 단락의 끝에서 endofrange 속성을
  사용한 마크업을 붙이면 됩니다
  <indexterm startref="example-band-index" class="endofrange">.
</para>
```

주석

1. 이것은 각주의 예로서 달아본 주석입니다.
2. DocBook의 table은 CALS Table Model로부터 기원한 것이라고 합니다.

부록 A. DocBook 설치와 사용

A.1. DocBook 설치와 설정

이 부분의 내용은 아직 준비 중입니다.

A.2. DocBook에 알맞은 편집기들

이 부분의 내용은 아직 준비 중입니다.

부록 B. 식별이름과 카탈로그

> 참고: 이 내용은 DocBook HOWTO (http://www.ibiblio.org/godoy/)의 ’Creating and modifying catalogues’ 부분을 번역한 것입니다.

B.1. 카탈로그의 작성과 수정

SGML은 식별이름(identifier)을 사용합니다. 식별이름라는 것은, 필요한 정보가 담겨있는 파일의 위치를 상징적으로 가리키고 있는 문자열들입니다.

식별이름에는 공식 식별이름(public identifier)과 시스템 식별이름(system identifier)이 있습니다. 공식 식별이름은 파일에 대한 정보를 규정에 따라 추상적으로 표현합니다. 이를 사용하면 어떤 종류의 시스템에서든지 같은 파일을 가리킬 수가 있습니다. 반면에, 시스템 식별이름은 파일의 위치를 지시하기 위해 시스템 상의 경로명을 그대로 사용하는 것이 보통입니다. 시스템 식별이름은 어떤 특정한 규정에 따르지 않으며, sgml 문서를 처리하는 프로그램에 의해 인식이 가능하기만 하면 됩니다.

공식 식별이름들이 가리키고 있는 파일의 구체적인 위치들은 별개의 파일에 정의되어 있는데, 이 파일을 카탈로그(catalogue)라고 합니다. SGML 문서를 처리하는 프로그램들은 이 카탈로그 파일에 의존해 SGML 문서 상에서 공식 식별이름들을 인식해내고 이들의 의미를 번역해냅니다.

또한, 카탈로그를 이용하면 여러가지 SGML 관련 파일들을 편리한 곳에 넣어두고 일괄적으로 관리할 수 있습니다(예를 들어, 자신의 홈 디렉토리나, `/usr/local/sgml`, 그 밖에 어떤 장소라도..). 문서를 직접 작성하지 않고 문서의 프로세싱과 “컴파일”만을 하는 주로 하는 경우라 하더라도 이런 기능은 편리한 점이 있습니다.

예 **B-1.** 카탈로그의 한 예

```
        -- Catalogue for the Conectiva Styles --            <10122>

        OVERRIDE YES

        PUBLIC "-
//Conectiva SA//DTD books V1.0//EN" "/home/ldp/styles/books.dtd" <10123>


        DELEGATE "-//OASIS" "/home/ldp/SGML/dtds/catalog.dtd"

        DOCTYPE BOOK /home/ldp/SGML/dtds/docbook/db31/docbook.dtd
                                                            <10124>
        -- EOF --
```

<10122> 주석. 주석은 "--"로 시작과 끝을 맺습니다.

<10123> 공식 식별이름 *"-//Conectiva SA//DTD books V1.0//EN"*은 시스템 식별이름 `/home/ldp/styles/books.dtd` 에 해당된다는 뜻입니다.

<10124> 파일의 끝을 나타내는 주석입니다.

위에 나타낸 예와 같이 카탈로그 파일이 작성되어 있다면, 다음과 같은 절차를 통해 식별이름과 해당 파일을 연관시키게 됩니다.

1. 문서 상에서 **PUBLIC**과 같은 지시어를 우선 인식합니다. 지시어는 뒤따르는 문자열이 식별이름임을 알려줍니다.
2. 식별이름의 유형을 파악합니다. 위의 경우처럼 **PUBLIC**이란 지시자에 뒤따르는 식별이름은 공식 식별이름입니다.
3. 공식 식별이름이 지시하고 있는 파일의 구체적인 경로명을 카탈로그에서 확인합니다.

B.2. 공식 식별이름 짓는 법

아래에 나타낸 식별이름을 주의깊게 보기 바랍니다.

```
"-//Conectiva SA//DTD books V1.0//EN"
```

이 식별이름은 그냥 멋대로 지어진 것이 아닙니다. 즉, 미리 정해진 규정에 따라 지어진 공식 식별이름입니다.

여기서 첫번째에 나타내어진 "-" 기호는 이 공식 식별이름이 정식으로 등록되어 있진 않다는 점을 나타냅니다. 사실, ISO나 IEEE 등에 관련된 경우를 제외하고는 정식 등록된 식별이름은 그리 많지 않습니다.

식별이름의 두번째 부분은, 그 해당 파일을 작성한 기구(단체)의 이름을 나타냅니다. 위의 경우에서는 Conectiva SA가 되겠습니다.

세번째 부분은 해당 파일이 어떤 종류의 문서이며(위 경우에는 DTD [1]문서) 그 이름과 버전이 어떻게 되는지를 알려줍니다.

마지막 부분은 해당 문서가 어떤 언어로 씌여졌는지를 나타냅니다. DocBook은 영어(English)로 씌여진 DTD이므로, 언어는 `EN`이 됩니다. 영문 두 글자로 사용 언어를 나타내는 방식은 ISO에서 권장하고 있습니다.

식별이름 짓는 규정에 대해서 좀 더 자세히 알고 싶은 분은 OASIS Technical Resolution 9401:1997 (Amendment 2 to TR 9401) (http://www.oasis-open.org/html/a401.htm)에 방문해 보시기 바랍니다.

B.3. 카탈로그에 쓰이는 지시어

카탈로그 파일에서 많이 쓰이는 지시어는 다음과 같습니다.

`PUBLIC`

지시어 `PUBLIC`은 공식 식별이름과 시스템 식별이름을 짝지워줄 때 쓰입니다.

`SYSTEM`

지시어 `SYSTEM`은 시스템 식별이름을 또 다른 시스템 식별이름과 짝지워 줍니다.

```
SYSTEM "http://nexus.conectiva/utilidades/publicacoes/livros.dtd" "publicacoes/livros.dtd"
```

`SGMLDECL`

지시어 `SGMLDECL`은 적용될 필요가 있는 SGML 선언(declaration) 파일의 위치를 알려줍니다.

```
SGMLDECL "publishings/books.dcl"
```

`DTDDECL`

`SGMLDECL`과 비슷한 것으로서, `DTDDECL`도 적용될 필요가 있는 SGML 선언 파일의 위치를 알려줍니다. 다만 `DTDDECL`은 특별히 DTD와 관련된 선언 파일들을 다룹니다. 그러나 아쉽게도, 현재까지 이 지시어를 지원하는 자유 소프트웨어는 없습니다. 그렇지만 이 지시어를 사용함으로써 여러개의 카탈로그 파일을 쓸 수 있는 이점을 얻을 수는 있습니다.

```
DTDDECL "-//Conectiva SA//DTD livros V1.0//EN" "publicacoes/livros.dcl"
```

`CATALOG`

지시어 `CATALOG`는 카탈로그 안에 또 다른 카탈로그를 포함시킬 수 있도록 해줍니다. 이 방법을 쓰면, 카탈로그를 뜯어 고치지 않고서도 여러가지 독립적인 카탈로그를 함께 사용할 수 있습니다.

`OVERRIDE`

지시어 `OVERRIDE`는 공식 식별이름보다 시스템 식별이름이 더 우선권을 갖도록 할 것인지 아닌지를 결정합니다. 대부분의 시스템에서는 시스템 식별이름이 우선권을 갖도록 되어 있습니다.

`DELEGATE`

지시어 `DELEGATE`는 어떤 공식 식별이름의 집합을 따로 다른 카탈로그 파일에 의해서 해석될 수 있도록 해줍니다. 이 지시어는 해당 공식 식별이름을 인식하는 과정에 개입한다

는 점에서 CATALOG 지시어와는 차이가 있습니다.

```
DELEGATE "-//OASIS" "/usr/sgml/oasis/catalog"
```

위의 예에서, "-//OASIS"로 시작하는 모든 공식 식별이름은 모두 "/usr/sgml/oasis/catalog"에 의해서 해석됩니다.

DOCTYPE

만일, 어떤 유형의 문서인지를 알려주는 공식 식별이름이나 시스템 식별이름이 없는 SGML 문서가 있다면, DOCTYPE 지시어에 의해 지정된 DTD가 기본값으로 사용됩니다.

주석

1. 다음과 같은 종류의 문서들이 가능합니다: DTD, DOCUMENT, ELEMENTS, ENTITIES, 그리고 NONSGML.

부록 C. 효율적인 글쓰기

C.1. 색인 자동으로 넣기

DocBook이 색인을 위한 기초요소을 갖고 있긴 하지만, 이를 통해 색인을 자동적으로 생성해 주지는 못합니다. 그러나 collateindex.pl (http://docs.kldp.org/Stuff/collateindex.pl)을 사용하면 색인을 자동적으로 만들어 낼 수 있습니다 [1] 이 스크립트의 사용법은 다음과 같습니다.

1. jade에 `-V html-index` 옵션을 주고 *HTML* 스타일시트와 함께 컴파일합니다.

   ```
   $ jade -t sgml -d html/docbook.dsl -V html-index document.sgml
   ```

2. collateindex.pl (http://docs.kldp.org/Stuff/collateindex.pl)을 사용해 `index.sgml`을 컴파일합니다.

   ```
   $ perl collateindex.pl -o index.sgml HTML.index
   ```

이렇게 생성된 `index.sgml` 파일은 원래의 DocBook 문서에 포함되어야 합니다. 이렇게 하기 위해서 `index.sgml`을 외부 실체요소(external entity)로서 문서의 맨 앞에 선언해 줍니다.

예 **C-1.** 색인을 넣기 위한 외부 실체요소 선언

```
 <!doctype article PUBLIC "-//OASIS//DTD DocBook V3.1//EN" [

<!-- Insertion of the index -->
<!entity index SYSTEM "index.sgml">
]>
```

이제 문서상에서 `&index;`라고 써주면, 그 곳에 `index.sgml` 파일이 삽입될 것입니다. 색인을 위한 마크업을 작성하는 자세한 내용에 대해서는 4.5절을 참고하기 바랍니다.

> 참고**:** 이 내용은 DocBook HOWTO (http://www.ibiblio.org/godoy/)의 'Tools & Hints'에서 'Inserting indexes automatically' 부분을 번역한 것입니다.

C.2. 글을 지속적으로 보완해 나가기

글이란 원래, 일단 초안을 잡고 이것을 지속적으로 향상해 나가는 과정의 산물이라고 할 수 있습니다. 따라서, 전체 글 중에 아직 미완성된 부분을 실제로 변환된 문서에 포함되도록 할 것인지 말 것인지를 결정해야 할 경우가 많이 있습니다. 또한 완성된 문서를 부분적으로 보완해 나가는 과정에서도, 보완 수정된 부분을 최종적으로 포함시킬 것인지 결정해야 할 경우가 생깁니다.

DocBook은 이런 경우를 위해, 글의 특정 부분을 최종 변환되는 문서에 나타나도록 할 것인지 여부를 결정해 주는 실체요소(entity)를 가지고 있습니다. 이를 사용하면, 글의 내용을 향상시키기 위해 초안을 잡아둔 특정 부분을 보이지 않게 할 수가 있게 됩니다.

여기에 쓰이는 형식의 실체요소를 특별히 매개변수 실체요소(parameter entity)라고 부릅니다. 이것을 사용하면 여러 초안들을 일률적으로 다룰 수 있습니다.

예 **C-2.** 매개변수 실체요소의 사용

```
 <!entity % review "INCLUDE">
 ...
 <![%review;[
<para>이 문단은 실체요소 "review"가 "INCLUDE"로 선언되었을 때
문서상에 나타나게 됩니다.</para>
]]>
```

실체요소 `review`는 예 C-2에서 제시된 것처럼 여러 문장을 갖게 됩니다. 이들이 최종적으로 완성되어서 더이상 따로 취급될 필요가 없게 되었을 때는, 위에서 세번째 줄과 여섯번째 줄만을 지워버리면 그대로 전체 글과 통합됩니다.

완성되지 않은 글의 초안을 숨기고 싶을 때는 실체요소 선언부분의 `INCLUDE`를 `IGNORE`로 바꿔주기만 하면 됩니다.

> 참고**:** 이 내용은 DocBook HOWTO (http://www.ibiblio.org/godoy/)의 ’Tools & Hints’에서 ’Re-using parts of documents’ 부분을 번역한 것입니다.

C.3. 문서를 재사용하기

외부 실체요소(*external entity*)를 사용해 문서를 내용에 따라 여러개의 파일로 분할해 놓으면 다음에 문서를 재사용하기가 훨씬 편리해집니다.

즉, 라이선스나 회사의 정강 정책 같은 내용을 별개의 파일로 만들어 두면, 이것을 다른 문서에 포함시키는 방법으로 두고두고 반복 사용이 가능합니다.

이렇게 외부 실체요소를 사용하는 방법의 예가 예 C-1입니다.

> 참고**:** 이 내용은 DocBook HOWTO (http://www.ibiblio.org/godoy/)의 ’Tools & Hints’에서 ’Re-using parts of documents’ 부분을 번역한 것입니다.

주석

1. 색인에 대한 Norman Walsh의 글 (http://docbook.sourceforge.net/projects/dsssl/)에서 더 많은 정보를 얻을 수 있습니다.

용어 해설

참고: 이 내용은 DocBook HOWTO (http://www.ibiblio.org/godoy/)의 'Glossary'를 번역한 것입니다.

attribute - 속성

속성은 기초요소가 문서에 어떻게 표현될 것인지에 대해 좀더 특별한 정보를 갖게 해줍니다. 속성은 언제나 이름-값 형태로 짝을 짓도록 되어 있습니다. 즉, *id="identification"* 이런 형태가 됩니다. 이것은 속성 *id*가 값 *identification*을 갖는다는 뜻입니다.

Document Type Definition (DTD)

DTD는 기초요소(element)와 그 속성(attribute)들이 어떻게 조합되고 구조를 이루는지를 정의합니다. 이에 따라 현재 커서 위치에선 어떤 기초요소가 사용 가능하고 불가능한지를 알 수 있게 됩니다.

DSSSL

DSSSL은 Document Style Semantics and Specification Language의 줄임말이며 ISO 표준입니다(ISO/IEC 10179:1996). DSSSL은 SGML의 스타일시트를 작성하기 위한 국제적인 표준 언어입니다.

element - 기초요소

기초요소는 문서의 계층적인 구조를 정의합니다. 기초요소의 대부분은 한 쌍의 열고 닫는 태그로 이루어져 있는데, 이 태그 사이에 글의 내용이 들어가게 됩니다. 빈 기초요소(empty element)라는 것은 여는 태그만 사용하는 기초요소를 말합니다. 빈 기초요소는 글의 내용을 갖지 않습니다.

entity - 실체요소

실체요소라는 것은 이름으로 참조될 수 있는 데이터의 한 부분을 말합니다. 이들이 가리키는 것은 하나의 문자이거나 한 chapter, 또는 DTD의 선언문일 수도 있습니다. 실체요소에는 일반 실체요소(generic entity), 내부 실체요소(internal entity), 외부 실체요소(external entity), 매개변수 실체요소(parameter entity)가 있습니다.

external entity - 외부 실체요소

외부 실체요소는 외부의 문서 파일을 가리킵니다. 이것은 SGML 문서의 특정 위치에 외부 파일의 내용을 삽입해 넣고자 할 때 사용합니다. 보통, 법적 공지문이나 특별히 긴 예문 등 따로 분리해 둘 필요가 있는 문서를 위해 사용합니다.

generic entities - 일반 실체요소

"&"로 시작해 세미콜론으로 끝나는 문자열로 참조되는 실체요소를 일반 실체요소라고 합니다. 보통, 일반 실체요소는 문서 파일을 가리키며 DTD를 가리키진 않습니다. 일반 실체요소는 내부 실체요소와 외부 실체요소로 나눌 수 있습니다.

internal entity - 내부 실체요소

내부 실체요소는 문서 내부의 특정 부분을 가리킵니다. 반복적으로 같은 내용을 사용해야 할 때 유용합니다.

paremeter entity - 매개변수 실체요소

이것은 주로 DTD에서 사용됩니다. 실체요소의 이름은 퍼센트 표시(%)로 시작해 세미콜론으로 끝납니다.

float - 떠다님

그림, 표, 챠트, 옆줄 등이 고정된 위치에 있지 않고 알맞은 위치를 찾아 움직일 수 있을 때, 이들을 떠다님(float)이라고 말합니다. 예를 들어 떠다니는 그림은 페이지 상에서 알맞은 위치에 배치시킬 수 있으며 그림이 너무 크다면 아예 다음 페이지로 넘길 수도 있습니다.

processing instruction - 처리 지시문

처리 지시문은 문서를 처리하는 프로그램에 전달되며, "<?"로 시작합니다. 예를 들면 문서를 어떤 이름의 HTML로 변환시킬지 다음과 같이 지정할 수 있습니다. `<?dbhtml filename="file.html">`

SGML

Standard Generalized Markup Language. 플랫폼 독립적인 전자 문서를 작성하기 위한 마크업 시스템이 어떻게 구성되어야 하는지를 정의한 국제 표준(ISO8879)입니다.

tag - 태그**(**꼬리표**)**

SGML의 기초요소들은 구분을 위해 "<"와 ">"로 둘러싸여서 글에 삽입되는데, 이들을 모두 뭉뚱그려서 태그라고 부릅니다. 예를 들어 **`<title>`**는 제목을 써넣을 때 맨 앞에 쓰는 태그입니다.

XML

eXtensible Markup Language. SGML의 하위 산물이지만, 새로운 버전의 SGML이라고 볼 수도 있습니다. 특별히 인터넷을 위한 SGML입니다.

XSL

XML Style Language. XSL이란, SGML에서의 DSSSL 스타일시트에 해당하는 것으로서 XML을 위한 것입니다. 사실 XSL도 또한 XML 문서입니다.