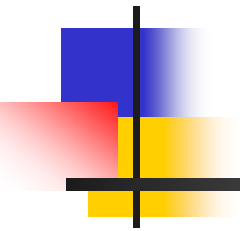


XML Parser

배 희호 교수
경복대학교
소프트웨어융합과

Data를 저장하는 방식

- Text File
  - Text File은 사람이 읽을 수 있는 형태로 저장되는 File
  - 보통 ASCII나 UTF-8 등의 문자 Encoding을 사용하여 Data를 저장
  - 특징
    - 사람이 직접 읽고 편집할 수 있음 (메모장, VS Code, Sublime Text 등으로 열 수 있음)
    - Data가 문자열 형태로 저장됨
    - 개행 문자(₩n)나 공백 등을 이용하여 Data 구분
    - 크기가 커질 수 있음 (같은 Data를 저장하더라도 Binary File보다 용량이 큼)
    - Text File은 정보를 쉽게 공유할 수 있지만, 서식과 같은 독특한 정보를 추가할 수 없다는 단점이 있음
    - 단순한 단어들의 나열만 가능

Data를 저장하는 방식

- Binary File
  - Binary File은 0과 1의 Binary Data로 저장되는 File
  - 사람이 직접 읽기는 어렵지만, Computer가 효율적으로 처리할 수 있음
  - 특징
    - 사람이 직접 읽을 수 없음
      (Text Editor로 열면 깨진 문자로 보임)
    - Data를 효율적으로 저장 가능
      (Text File보다 용량이 작음)
    - File 구조에 따라 Data를 저장해야 하므로 읽고 쓰는 방식이 정해져야 함
    - 주로 Image, Audio, 영상, 실행 파일(.exe) 등을 저장하는 데 사용됨
    - Binary File은 호환성이 없다는 단점이 있음

Data를 저장하는 방식

- Markup을 고려하는 이유
  - Structured Data(Data 구조화)
    - 일반 Text File이나 Binary File은 Data의 구조를 명확히 표현하기 어려움
    - Markup을 사용하면 Data가 계층적(Tree 구조)으로 정리되어 Data 간의 관계를 쉽게 파악할 수 있음
  - Human-Readable(Data의 가독성)
    - Binary File과 달리, Markup을 이용하면 Data를 사람이 읽기 쉬움
    - JSON, XML, YAML과 같은 Format은 사람이 직접 Data를 수정할 수도 있음
  - Data 교환의 용이성(Interoperability)
    - Markup Data는 다양한 Program, System, Platform에서 쉽게 교환할 수 있음
    - Web 개발, API, Database, 설정 File 등에 사용

Data를 저장하는 방식

- Markup을 고려하는 이유
  - Data의 유효성 검증(Validation)
    - Markup을 이용하면 Data의 유효성을 검증할 수 있음
    - 예) XML의 경우 DTD 또는 XSD를 사용하여 특정 규칙을 준수하는지 확인할 수 있음
  - 확장성과 호환성 (Scalability & Compatibility)
    - 새로운 Field를 추가하거나, 기존 Data를 변경해도 큰 문제 없이 확장할 수 있음
    - 다양한 환경에서 사용될 수 있도록 독립적인 Data 저장 방식을 제공

Data를 저장하는 방식

- Markup을 사용할 때의 단점
  - File 크기 증가
    - Tag가 많아지면 Data 크기가 커질 수 있음
  - 처리 속도
    - Parsing 과정이 필요하여 Binary File보다 속도가 느릴 수 있음
  - 복잡성 증가
    - 단순한 Data 저장에는 오히려 불필요한 복잡성이 추가될 수 있음

Markup

- Markup Language
  - Tag 등을 이용하여 문서나 Data의 구조를 명기하는 언어의 한 가지
- Tag
  - 원래 Text와는 별도로 원고의 교정 부호 및 주석을 표현하기 위한 것이었으나 용도가 점차 확장되어 문서의 구조를 표현하는 역할을 하게 됨
  - 문서의 골격에 해당하는 부분을 작성
- 일반적으로는 Data를 기술하는 정도로만 사용되기 때문에 Programming Language와는 구분됨

Markup

- Markup Language 구분
  - Presentational Markup
    - 전통적인 Word Processing System에서 사용되는 Markup의 종류
    - WYSIWYG(“what you see it what you get”) 효과를 생성하는 문서 Text 내에 포함된 Binary Code
    - 사용자들이 아래 절차적/기술적 Markup을 사용하면, 사용자에게 “현재(WYSIWYG)” 상태로 변환
  - Procedural Markup
    - Text에 포함되며, Text는 Program을 통해 Text를 처리하기 위한 지침을 제공
    - Processor는 마주친 지침에 따라 처음부터 Text를 통해 실행 될 것으로 예상
    - 예) troff, TeX, PostScript

Markup

- Markup Language 구분
  - Descriptive Markup(기술적 마크업)
    - 문서의 일부에 Label을 붙이는 데 사용
    - 문서의 고유한 구조를 어떤 특정한 처리나 변경으로부터 분리
    - 예) LateX, HTML, XML
    - 예) HTML의 태그
    - 시각적으로 가 아닌 개념적으로 자료를 설명하는 방식으로 쓰도록 권장
  - 경량화 마크업(lightweight markup)
    - 최근 Web Browser를 통해 형식화된 Text를 작성할 수 있도록 개발된 작고 표준화되지 않은 다수의 Markup
    - 예) 위키피디아에서 사용하는 위키 마크업

Markup

- Markup Language 종류
  - GenCode
  - troff / nroff
  - TeX
  - Scribe, GML, SGML
  - HTML
  - XML
    - XML 기반 프로그램 : RDF/XML, XForms, DocBook, SOAP, WOL(Web Ontology Language) 등
  - XHTML
  - XML 기반 HTML
  - JSON
  - YAML

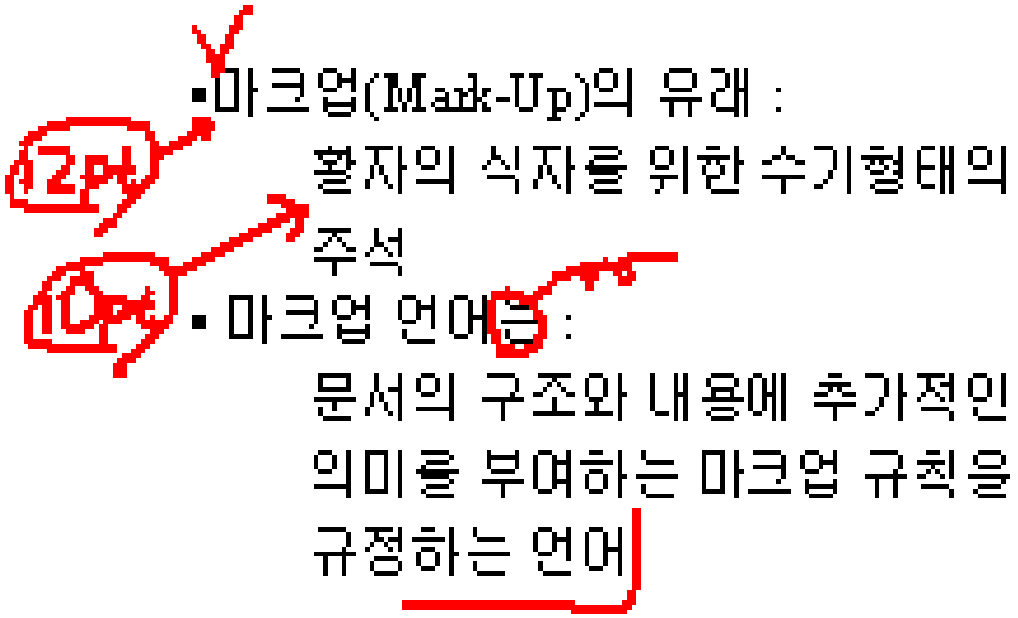
Markup

- Markup을 사용하는 대표적인 File Format

Format	설명	용도
HTML	Web Page 구조 정의	Web Page 제작
XML	Data 저장 및 교환용 Markup	설정 파일, Database, Web Service
JSON	경량 Data Format (JavaScript 기반)	API, Web 개발, Data 교환
YAML	사람이 읽기 쉬운 설정 File 형식	설정 File, DevOps, Kubernetes

Markup 언어

- Markup의 유래
  - 활자의 식자를 위한 수기 형태의 주석

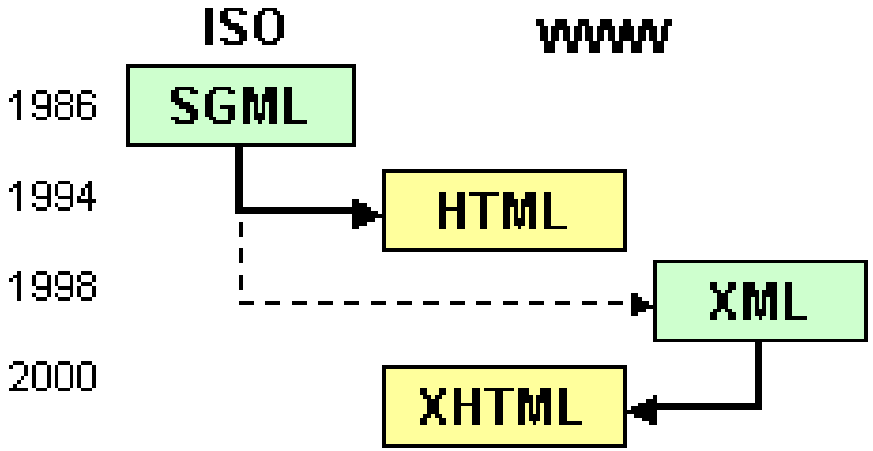


Markup 언어

- Markup
  - 문서 처리를 지원하기 위해 문서에 추가되는 정보
- Language
  - 정의된 구문과 문법을 따르는 기호의 집합
- Markup Language
  - 문서의 구조와 내용에 추가적인 의미를 부여하는 Markup 규칙을 규정하는 언어
  - 어떠한 문서 안의 정보가 어떻게 구조화 되었는가를 지정하는 언어
  - 문서에 포함된 문장, 표, 그림 등과 같은 내용에 대한 정보를 말하는 것이 아니라, 이들이 어떻게 배치되고, 어떠한 크기와 모양 등을 가지고 있는가 등의 정보를 의미
  - 기본적으로 문서의 구조 형식을 Tag를 사용하여 지정
  - 예) WordProcessor, RTF(Rich Text Format), TeX

Markup 언어

- Web에서 대표적인 Markup 언어
  - SGML, HTML, XML
  - HTML은 SGML에 기반한 Markup 언어(SGML의 응용)
  - XML은 복잡한 SGML을 간소화한 Web 문서 작성 언어
    - XML은 구조화된 문서를 작성할 수 있어 HTML 언어의 단점을 개선
  - XHTML은 XML에 기반한 언어 – HTML과 동일한 기능
- SGML, XML, HTML 및 XHTML의 관계



Markup 언어

Markup 언어의 역사

언어	설명
GML (1970년대)	✓ General Markup Language의 약자 ✓ 1979년 IBM 연구원인 Dr.GoldFarb가 법률 문서를 관리하기 위하여 최초로 개발한 Mark-up 언어
SGML (1986년)	✓ Standard Generalized Markup Language의 약자 ✓ 미국 및 캐나다에서 공공 문서나 법률 문서를 관리하기 위한 목적이나 출판업계에서 사용할 목적으로 많이 사용 ✓ 문서의 구조를 정의하기 위한 많은 문법들을 제공하고 있는데 이러한 규칙들이 너무나 복잡하다는 단점을 가짐 ✓ 이후에 SGML은 HTML과 XML의 모태가 됨
HTML (1990년대)	✓ Hyper-Text Markup Language의 약자 ✓ 1991년 Tim Berners Lee에 의해 개발 ✓ SGML의 복잡한 기능을 제거하고, Internet상에서 쉽게 사용할 수 있는 Mark-up Language로 개발됨 ✓ 쉬운 사용 방법에 의하여 급속히 일반인들이 사용할 수 있게 되었고 그 결과 Internet이 일반화 되는데 큰 영향을 끼침

Markup 언어

- Markup 언어의 역사

언어	설명
XML (1998년)	✓ eXtensible Markup Language의 약자 ✓ Platform에 독립적이고 이 기종간의 정보교환이 가능하며 Data의 구조를 정의하기에 적합하도록 만들어졌으며, SGML과 같이 강력하고 HTML처럼 쉽고 가벼우며 실용적인 특성을 갖도록 SGML을 재정의한 것
JSON (JavaScript Object Notation) 2000년대	✓ XML과 경쟁하며 등장한 Data Format, 마크업 언어는 아님 ✓ 간결성과 가독성을 중점으로 개발되었음 ✓ 특징 ✓ 키-값 쌍으로 Data 표현 ✓ 구조가 간단하여 Parsing 속도가 빠름 ✓ Web과 Mobile Application에서 널리 사용
Markdown	✓ 간단한 마크업 언어로, Text 기반의 문서 형식화에 사용
HTML5 (2014년)	✓ HTML의 최신 표준으로, Web Application과 Multimedia 지원에 중점
YAML	✓ 설정 File 및 Data 직렬화에 사용되는 인간 친화적 마크업 언어

Markup 언어

- Markup 언어의 분류

순차적 마크업	✓ 문서 내용을 시각적으로 어떻게 표현할 것인지에 대한 정보를 제공 ✓ 예) 폰트의 크기, 색깔, 여백, 줄 간격 등에 대한 정보 ✓ Data와 함께 문서의 표현 정보가 한 곳에 있음 ✓ 대표적인 예 : 워드프로세서 문서
서술적 마크업	✓ 문서 내용의 구조에 대한 추가적인 정보를 제공 ✓ 예) 문서에서 어떤 부분이 제목, 이름, 작가, 본문인지 등에 대한 정보 제공 ✓ Data와 문서의 구조를 표현하는 정보가 한 곳에 있음 ✓ 대표적인 예 : SGML, XML, Latex 등

Markup 언어

- Markup 언어의 중요성
  - Data의 구조화와 의미 표현
  - Platform 간 Data 교환 표준 제공
  - 오늘날의 Web, Mobile, Data 중심 환경에서 핵심 기술로 자리 잡음
  - SGML은 문서 중심, XML은 Data 중심으로 발전, JSON은 간결성과 Web 친화성을 강화

SGML

- SGML(Standard Generalized Markup Language)은 Text File과 Binary File의 저장 방식의 문제점을 해결
- File에 풍부한 정보를 기록할 수 있도록 하기 위해서 Markup을 사용하고 호환이 가능하도록 하기 위해서 Text를 기반으로 하는 언어
- SGML은 다양한 문서들에 대한 효율적인 관리와 상호 교환, 다양한 매체로의 변환이 가능하다는 강점이 있음
- 역사가 깊고 매우 좋은 언어지만 배우기가 너무 어렵고 방대하다는 단점이 있음
- 개발자들이 이러한 SGML의 뛰어난 강점을 이용해서 만든 언어가 바로 HTML
- HTML은 현재 Internet의 부상에 기여한 바가 큼
- 이렇게 HTML이 자리매김을 하고 있었는데 XML이란 또 다른 Markup 언어가 등장한 이유가 뭘까?

SGML

- SGML(Standard Generalized Markup Language)
  - 문서에서 논리 구조와 내용 구조를 기술하기 위한 Meta Language

장점	단점
✓ 유연성 ✓ 개방 표준 ✓ 시스템이나 플랫폼에 독립적 ✓ 재 사용성	✓ S/W 개발의 어려움 ✓ Web에서 S/W 제한과 Instance 의 이식성 결여 ✓ 산업계의 지원 ✓ 논리구조 작성의 어려움 (DTD 작성 어려움)

HTML

- HTML(HyperText Markup Language)
  - W3C의 명세, Web 상에서 Hypertext 문서를 생성할 수 있는 간단한 Markup Language
  - SGML의 응용
  - ASCII Text 양식의 문서
  - HTML = SGML + DTD

장점	단점
✓ 이식성과 사용 편리 ✓ HTML Instance를 Web 에서 손쉽게 다운로드 가능	✓ 고정된 Tag 집합 ✓ 다양한 Page Format ✓ 임의 구조화 능력 부족 ✓ 효과적 검색 및 재사용 어려움

HTML

- HTML은 Data를 정보 형태로 가공해서 보낼 수 없음
  - Server와 Client간에 통신을 하게 되면 Server 측에서도 각각의 내용에 대한 Tag를 붙여서 표현하는 방법만 제시해서 보내줌
  - 이는 HTML은 단지 표현 형식만 첨가하였을 뿐 아무 의미 없는 단어들의 나열만 전달해주는 것이라고 볼 수 있음
    - 예) HTML이 '나주 배'를 요청하면 그게 먹는 배인지, 타는 배인지 하는 지능적인 검색이 불가능하다는 것
  - 이러한 문제점을 해결하기 위해서 개발자들은 HTML의 기존 기능인 Tag로 내용을 표현해주면서, 그 내용을 Database화(의미를 부여할 수 있도록)하기 위한 것이 XML

XML 소개

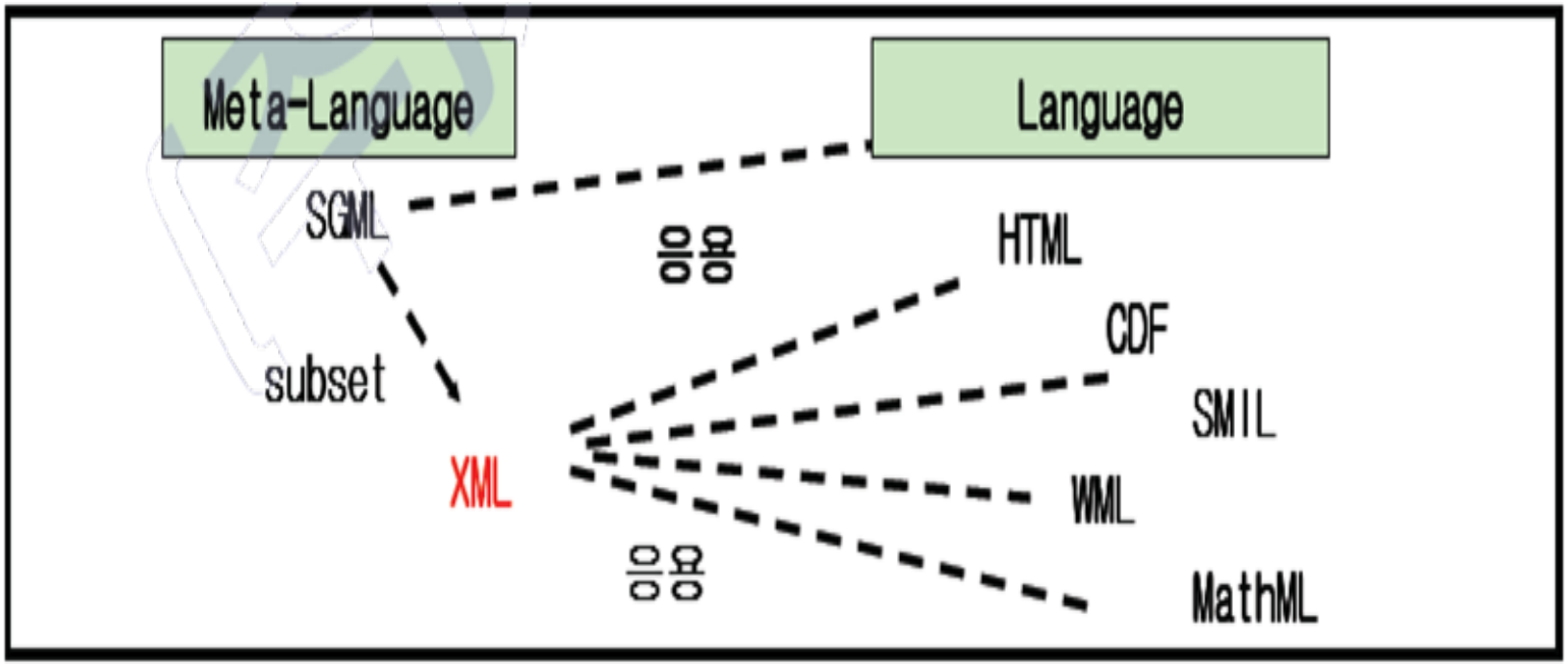
- XML의 정의
  - eXtensible Markup Language(확장 가능한 마크업 언어)의 약자
    - 확장 가능 : 기존에 없던 것을 새롭게 만듦
    - Mark-up : 문서의 논리적인 구조와 내용 기술
  - 1996년 W3C에서 제안하고 XML Working Group에 의해 개발됨
  - Web에서 구조화된 문서를 효율적으로 처리하도록 설계된 표준화된 Data 형식
  - SGML의 장점(구조화된 문서를 정의하여 Tag를 자유롭게 정의)과 HTML의 장점(Internet 상에서 손쉽게 Hypermedia 문서를 제공)을 모두 가질 수 있도록 제안된 ”Web 표준 문서 Format”
  - 확장 가능한 Markup 언어로써 Programming 언어가 아닌 Data를 Description하는 Meta Data 언어

XML 소개

- XML의 등장 배경
  - Data 교환의 표준화 요구
    - Internet과 Network의 확산
    - HTML의 한계(고정된 Tag, Page Layout) 극복
    - WWW의 전자 상거래를 위한 Business 영역으로 확대
  - Data 표현의 유연성과 구조화 요구
    - 계층적 Data 구조 필요
    - Data의 의미와 구조 분리
  - Platform 및 언어 독립적인 표준 요구
    - Platform 간 호환성
    - W3C의 표준화 노력
  - SGML의 복잡성을 단순화 필요
  - Data 상호운용성의 확립

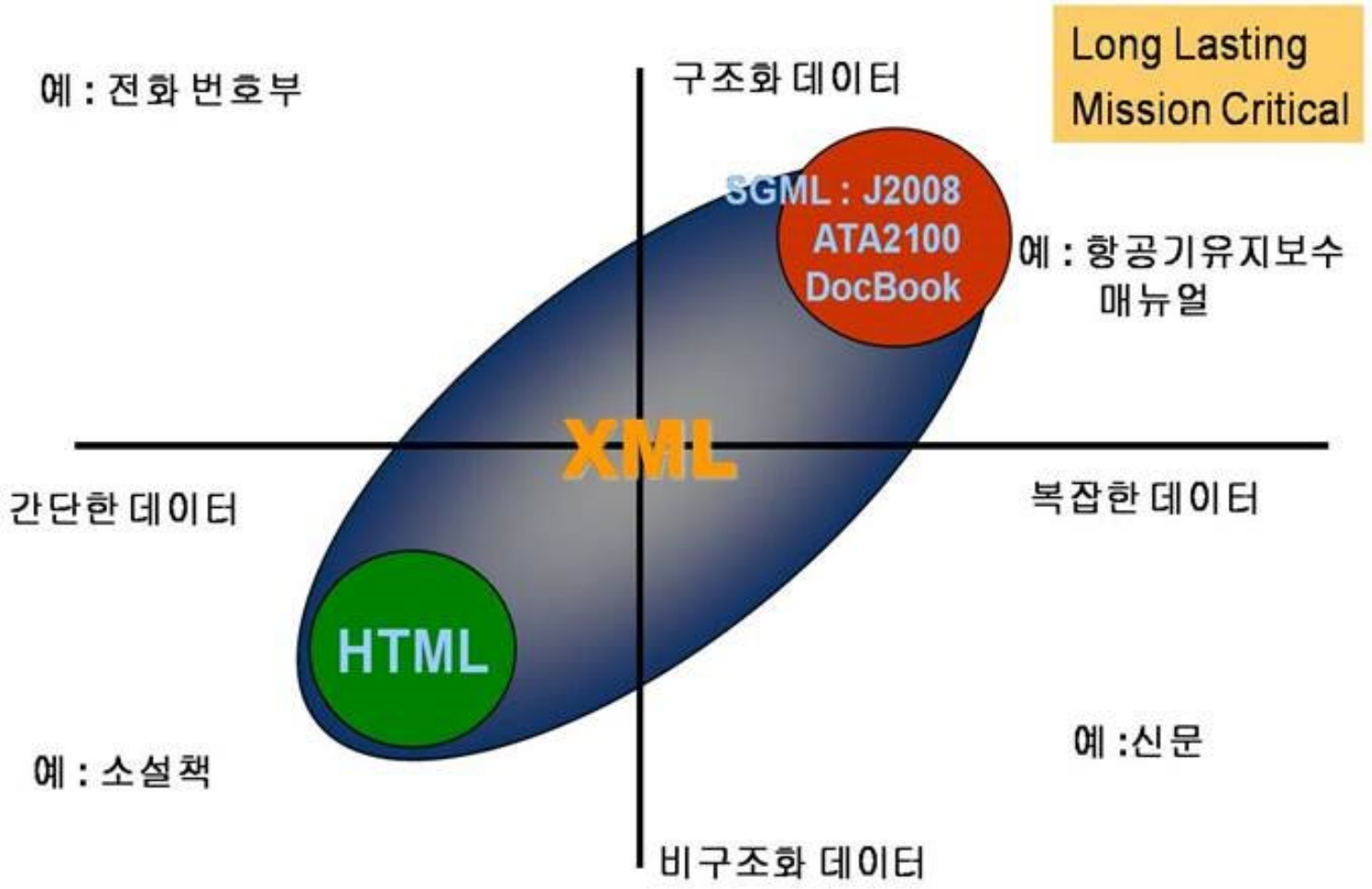
XML 소개

- XML의 관련 언어



XML 소개

- HTML, XML, SGML 비교



XML 소개

- 간단하고, SGML, HTML 경험에 기본 하였고, Tag의 제한이 없음
- Link의 자동 생성이 가능하며, 차세대 Hypertext 기능을 할 수 있고, 문서의 재사용이 용이함
- Web Site들의 간단한 System 관리를 제공

XML에 추가된 기능	SGML에서 계승된 기능	SGML에서 사라진 기능
Style Sheet XLINK, XPointer Query Language DOM, RDF Namespace, etc	문서형 선언 element 선언 문자 entity 외부 entity	생략 Tag, SGML 기능 논리적 결합 inclusion, exclusion NAME, NAMES NUMBER, NUMBERS NUTOKEN, NUTOKENS CURRENT etc

XML 소개

- XML(extensible markup language)
  - License – Free
  - Platform – Independent
  - 세계적 표준 언어
  - 하나의 XML Data를 다른 형태로 표현
    - 예) html, txt, wml , pdf 등
  - XML은 구조적이면서 Data를 쉽게 표현

XML 소개

- XML의 특징
  - 제품, Platform 독립성
    - HTML과 유사한 Text 기반 문서 Markup 언어이므로, Text 기반 S/W 도구를 비롯한 다양한 S/W 도구를 사용하여 처리할 수 있음
  - 단순성
    - SGML(Standard Generalized Markup Language)을 단순화한 문서 Mark-Up 언어로서, SGML의 일종 임
  - 확장성
    - 문서 Type 정의 기능을 통하여 다양한 XML 기반 문서 형식(새로운 Tag의 정의 등)을 정의하여 사용 가능
    - HTML 문서 Type도 문법에 약간의 제약을 가하면 XML 문서 Type 중 하나가 됨

XML 소개

- XML의 특징
  - 정확성
    - 주어진 응용에 가장 적합한 문서 Type 정의를 통하여 정보를 표현함으로써 문서 정보를 가장 적절히 표현할 수 있음
    - 이러한 특징은 문서 처리의 유연성과 효율성을 제공
  - 통일성
    - 다양한 XML 문서 Type을 정의하여 사용하더라도 기본 골격은 XML 문서 형식을 따름
    - 다양한 XML 응용 S/W에서 XML 처리기를 재사용할 수 있음

XML 소개

- XML의 특징
  - 출력 매체 독립성
    - 문서의 내용이 특정 출력 형식과는 독립적
    - HTML과는 달리 XML 문서가 최종적으로 출력되는 형태는 해당 XML 문서 Type을 지원하는 XML 처리기 응용에 달려있음
    - 문서의 내용을 다양한 형태(Table, Graph 등)로 출력할 수 있음
    - 문서의 내용을 다양한 형식의 다른 문서(HTML, 다른 XML 문서, RTF, MS Word, Postscript, PDF, TeX, LaTeX)로의 변환이 가능
    - XML 문서 출력은 Client측(Web Browser, Applet 등) 및 Server 측(Servlet 등)에서 이루어질 수 있음. 이때, 경우에 따라 HTML 문서 등 다른 문서 형식으로의 변환을 통하여 출력하는 방법이 종종 사용

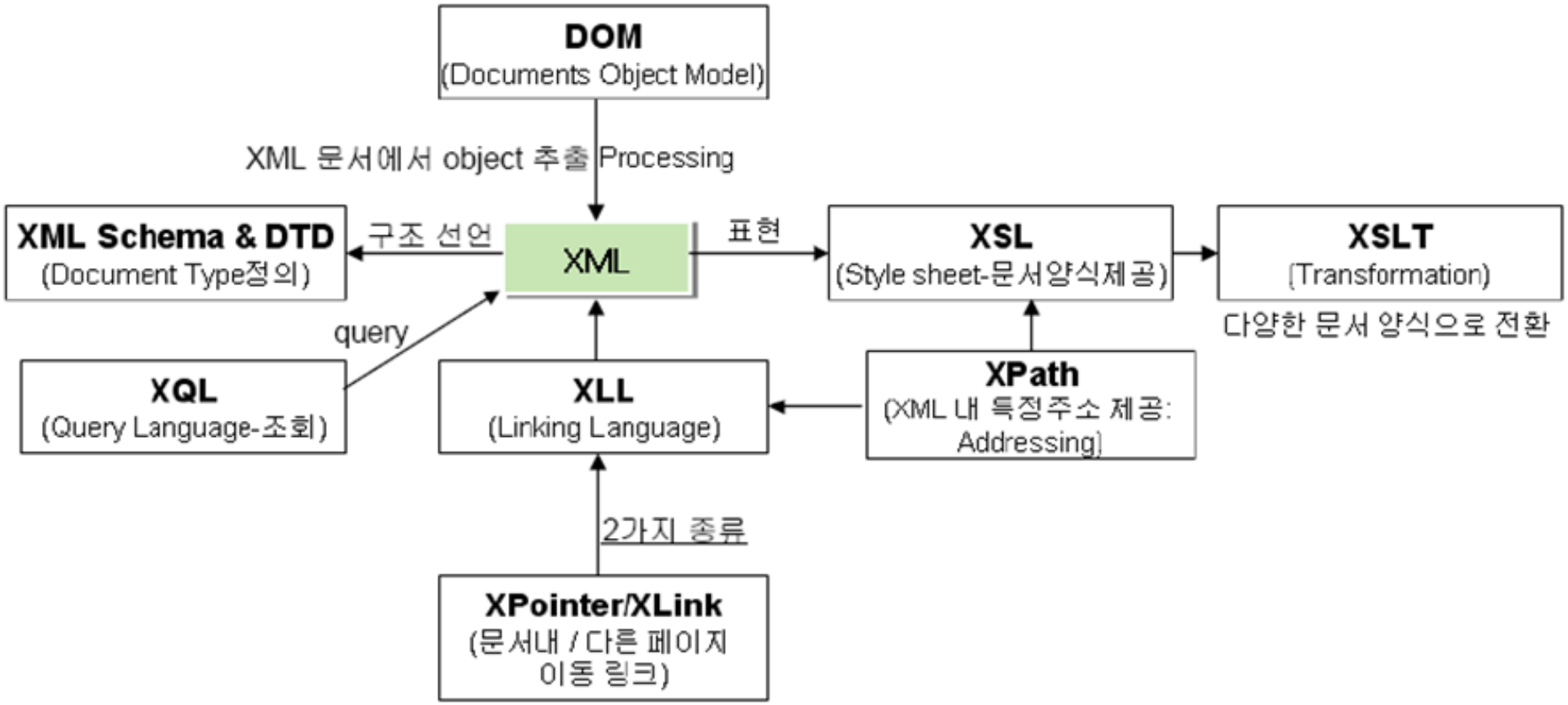
XML 소개

- XML 처리기
  - XML 문서를 읽어 들여 구문 분석한 후, 그 내용과 구조에 대한 접근을 제공하는 S/W Module
- XML 처리기 응용
  - XML 처리기를 내장(혹은 호출)하여 처리(편집, 검증, 저장, 전송, 검색, 구문 분석, 변환, Formatting 등) 기능을 제공하는 XML 처리기 응용 S/W

XML 문서
|
XML 처리기 + XML 처리기 응용

XML 소개

- XML의 구성도



XML 소개

구분	세부내용	특징
DTD	✓ Document Type Definition ✓ XML 문서의 형태를 일관된 구조로 정의하는 문서 ✓ XML 문서에 대한 논리, 물리적 구조 정의 ✓ XML 유효성 검증을 위해 필요	✓ 반복 정의 가능 : Tag의 중첩 순서의 정의(N회 반복)
XML스키마	✓ DTD보다 강력한 문서 구조, 내용, 의미 지원 ✓ Schema 자체가 XML 문법을 따름	✓ 사용자 정의 형식 ✓ 다양한 반복 형식 (최대, 최소)
XML Namespace	✓ DTD가 하나이상의 XML 문서를 참조 시 각 DTD를 위한 고유의 Namespace를 정의(각각의 DTD식별)	✓ URI형식
XSL	✓ eXtensible Style Language ✓ XML문서의 스타일 정보를 기술하지 위한 표준어로 사용 ✓ 다양한 출력 형식(HTML, WAP, PDF)을 정의하기 위한 포맷을 정의	✓ XSLT: 입력된 XML 문서를 원하는 출력 구조로 변환
XLL	✓ eXtensible Linking Language ✓ XML 요소 간이 연결 및 관계를 표시	✓ XLink: Hyper Link의 인식과 처리 ✓ XPointer: XML문서내의 요소에 대한 주소

XML 소개

- XML 문서 Type의 종류
  - XML 문서 Type은 다음과 같은 부류로 나뉘어질 수 있음
    - 개별적인 용도를 위해 정의한 새로운 문서 형식
    - 특정 분야에서 받아들여지는 표준 형식
    - 일반적으로 폭넓게 받아들여지는 표준 형식
  - XML 문서 Type 정의 모음
    - http://www.schema.net/
    - http://wdvl.com/Authoring/Languages/XML/Specifications.html
    - http://www.oasis-open.org/cover/xml.html

XML 소개

- Web 분야 문서 Type
  - XHTML (HTML 문서 Type의 XML Type 정의)
    - http://www.w3.org/TR/WD-html-in-xml/
  - Resource Description Framework (RDF)
    - 다양한 Web 기반 Meta Data의 통합
    - http://www.w3.org/RDF/
  - Platform for Internet Content Selection (PICS)
    - Label(Meta Data)과 Internet Contents와의 연동
    - http://www.w3.org/PICS/
  - Platform for Privacy Preferences (P3P)
    - http://www.w3.org/P3P/
  - Digital Signature (DSig)
    - http://www.w3.org/DSig/Overview.html
  - Web Interface Definition Language (WIDL)
    - http://www.w3.org/TR/NOTE-widl

XML 소개

- 정보 검색 분야 문서 Type
  - A Query Language for XML (XML-QL)
    - SQL 유사 질의어
    - http://www.w3.org/TR/NOTE-xml-ql/
- 사용자 Interface 분야 문서 Type
  - Extensible Forms Description Language (XFDL)
    - Form 기반 사용자 Interface 제공
    - http://www.w3.org/TR/NOTE-XFDL

XML 소개

- Graphics, Multimedia
  - Vector Markup Language (VML)
    - Vector Graphic
    - http://www.w3.org/TR/NOTE-VML
  - Scalable Vector Graphics(SVG)
    - 2차원 Graphics 표현(Vector Graphic 도형, Image, Text)
    - http://www.w3.org/Graphics/SVG/
  - Precision Graphics Markup Language (PGML)
    - 고 정밀 2차원 Graphics 표현(Font, 색상, 배치, 합성 등)
    - PostScript, PDF(Portable Document Format)에 공통된 Imaging Model 채용
    - http://www.w3.org/TR/1998/NOTE-PGML

XML 소개

- Graphics, Multimedia
  - X3D
    - 3차원 Graphic 표현
    - VRML의 확장 : http://www.web3d.org/fs_specifications.htm
  - Synchronized Multimedia Integration Language (SMIL)
    - 서로 독립적인 Multimedia 자료의 동기화된 Presentation
    - http://www.w3.org/AudioVideo/
  - JSML (Java Speech Markup Language)
    - http://java.sun.com/products/java-media/speech/forDevelopers/JSML/

XML 소개

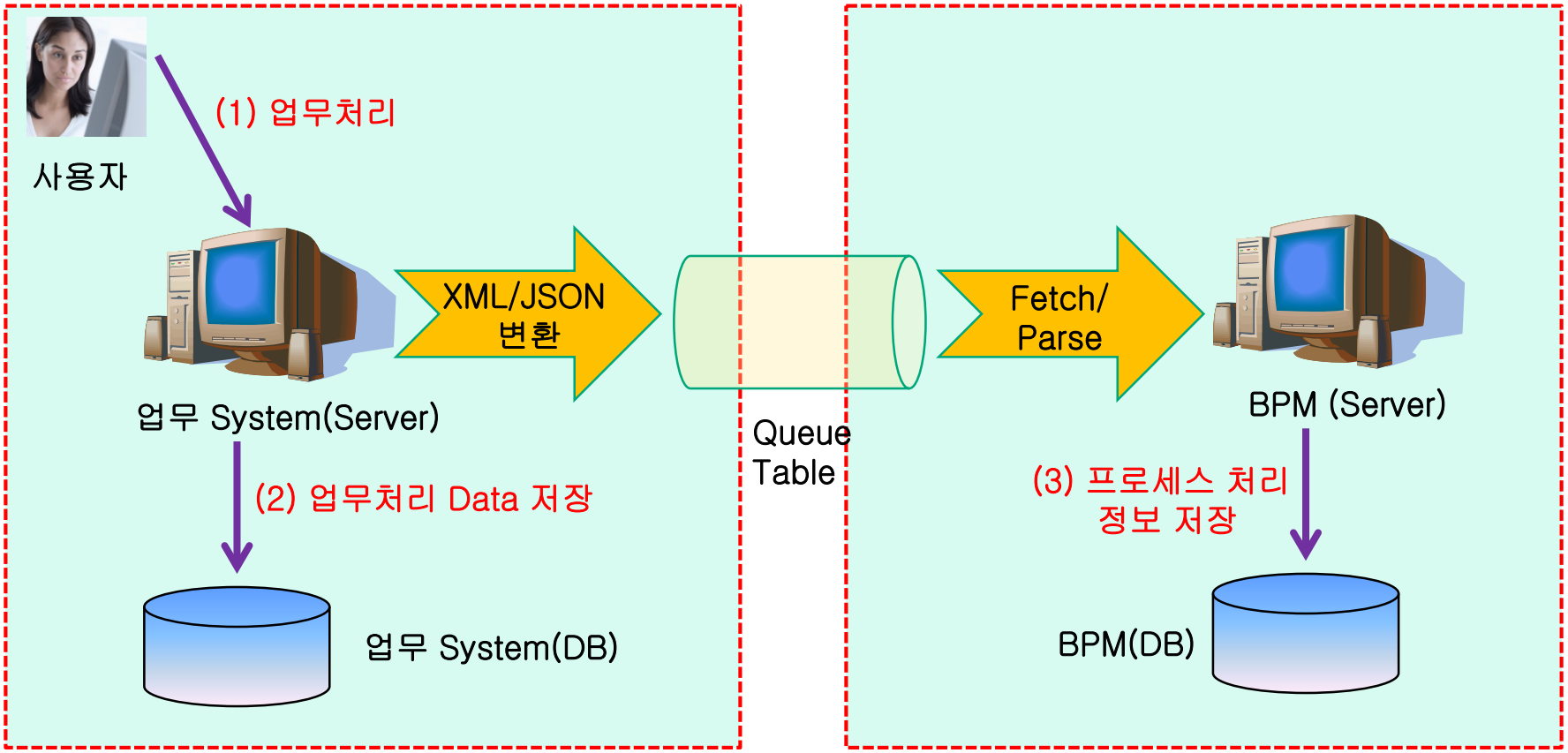
- 일반 문서, Presentation Type
  - TeXML
    - TeX과 유사
    - http://www.alphaWorks.ibm.com/formula/texml
  - XslSliderMaker
    - Slide Presentation
    - http://www.inria.fr/koala/XML/xslProcessor/slide.0.1.html

XML 소개

- 수학, 과학, 예술
  - Mathematical Markup Language (MathML)
    - 수학, 과학 기호 및 수식 표현
    - http://www.w3.org/Math/
  - Chemical Markup Language
    - 화학 분자식
    - http://ala.vsms.nottingham.ac.uk/vsms/java/jumbo/cml12/cml/
  - Music Markup Language (MusicML)
    - http://tcf.nl/3.0/musicml/index.html

XML 소개

- BPM(Business Process Management) System 구축
  - 업무 System에서 발생된 BPM 연계처리 Data를 XML이나 JSON Format으로 변환



XML 소개

- RBD와 XML의 Mapping
  - RDB의 Data를 어떻게 XML로 Mapping할 것인가?

멀 선택하지..???.

CarList DB

name	CC	Company
소나타	2000	현대
스포티지	3000	기아

RDB를 Element와 Mapping

```
<CarList>
    <car>
        <name>소나타</name>
        <CC>2000</CC>
        <company>현대</company>
    </car>
    <car>
        <name>스포티지</name>
        <CC>3000</CC>
        <company>기아</company>
    </car>
</CarList>
```

◆ 장점
- DOM을 이용한 계층적 접근이 가능

◆ 단점
- 문서의 용량이 커짐

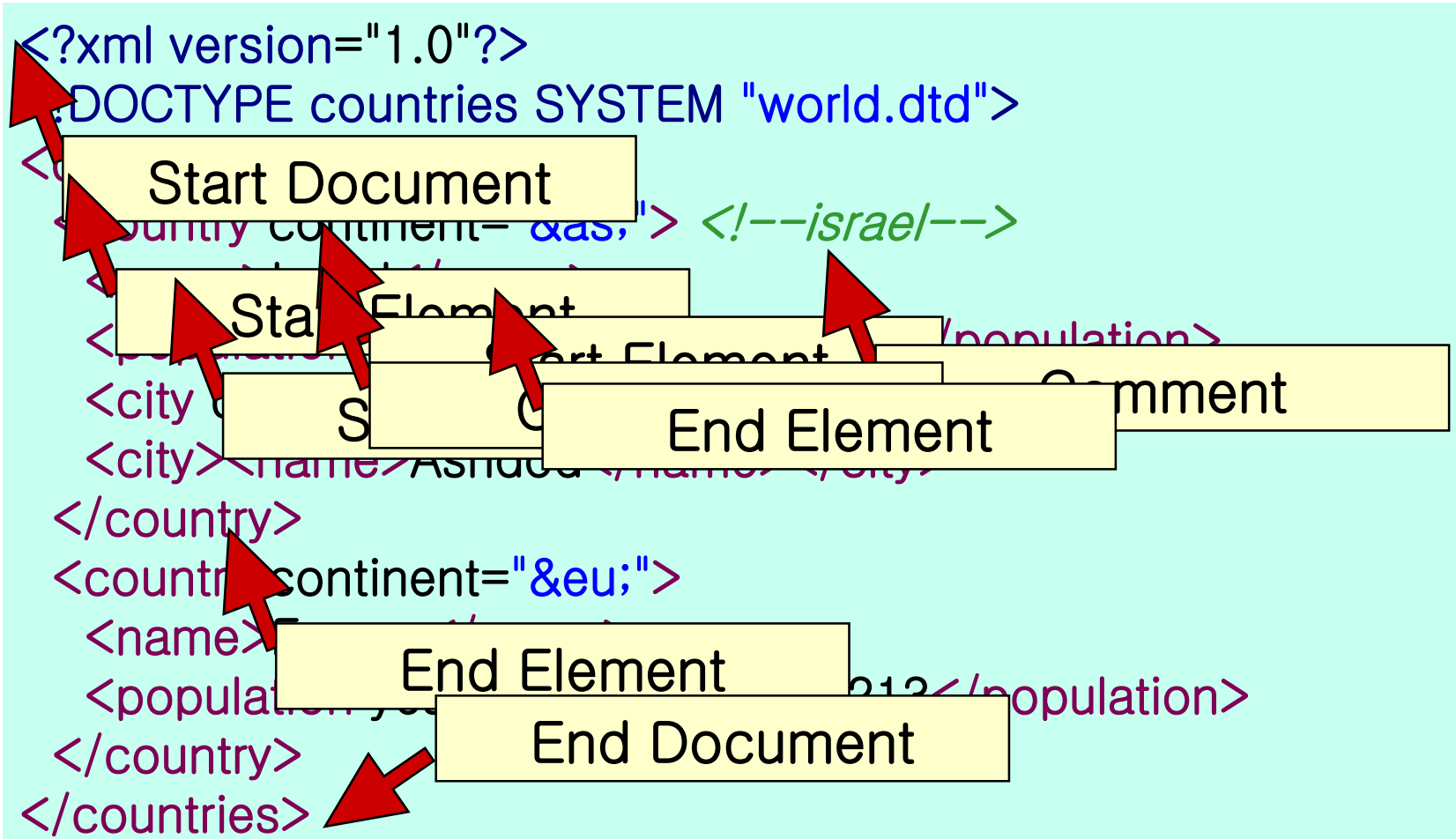
RDB를 Attribute와 Mapping

```
<CarList>
   <car name="소나타" CC="2000" company =" " 현대" />
   <car name="스포티지" CC="3000" company ="기아"/>
</CarList>
```

◆ 장점
- 문서의 용량을 줄일 수 있음

◆ 단점
- name을 이용한 Attribute의 검색만이 가능

XML 문서


<?xml version="1.0"?>
<!DOCTYPE countries SYSTEM "world.dtd">
Start Document
<!--israel-->
End Element
End Element
End Document
</country>
</countries>

XML 문서

```
<xml version="1.0" encoding="UTF=8" ?>
<Library>
  <book>
     <name isbn = “123456”>JAVA</name>
     <author>홍길동</author>
     <price>15000</price>
     <publish company=“한빛아카데미”>2015.7.20</publish>
  </book>
  <book>
      <name isbn = “345656”>Android 공부</name>
     <author>정재곤</author>
     <price>25000</price>
     <publish company=“이지스퍼블리싱”>2016.12.20</publish>
  </book>
  <book>
      <name isbn = “245621”>데이터베이스</name>
     <author>한만국</author>
     <price>23000</price>
     <publish company=“글로벌출판사”>2017.3.30</publish>
  </book>
</Library>
```

XML 문서

- Data 구조가 정의된 Text의 집합
- 문서의 구조를 임의의 DTD(Document Type Definition)를 선언하고 Tag를 정의함으로써 문서를 표현 가능
- XML Tag들은 문서의 구조 혹은 Data의 의미를 표현

XML 문법

- PCDATA(Parsed Character DATA)
  - PCDATA는 Parser(해석기)에 의해 해석이 될 Text XML 요소의 시작 Tag와 마침 Tag 사이에 위치한 Text
- CDATA(Character DATA)
  - CDATA는 Parser에 의해 해석이 되지 않을 Text
  - <![CDATA[로 시작하여,]]>로 끝나는 영역은 그대로 출력 '<', '>', '&' 같은 특수문자를 사용할 때 CDATA로 감싸면 됨
- Entities(엔티티)
  - XML에서 Entities는 변수와 비슷한 개념으로 생각할 수 있음
  - Entities는 문서 내에서 참조 될 수 있는 문자 집합의 단위
  - 예) HTML 문서에서 “ “가 Entities 참조의 한 예

XML 문법

```
<?xml version="1.0" encoding="utf-8" ?>
<booklist>
  <book>
    <title>[!CDATA< XML &amp;amp; VisualBasic  >]</title>
  </book>
</booklist>
```

- CDATA Section내의 문자 Data인 XML & VisualBasic 는 XML Parser에 의해 해석되지 않고 아래와 같이 바로 응용 프로그램(익스플로러)으로 전달되어 출력

```
<?xml version="1.0" encoding="utf-8" ?>
<booklist>
  <book>
    <title>[!CDATA[ XML &amp;amp; VisualBasic  ]]</title>
  </book>
</booklist>
```

문자 Data로 사용할 수 없는 < 문자나 & 문자도 CDATA Section 내에서는 마음대로 사용할 수 있으며 공백 문자의 길이도 보존 됨

XML 문법

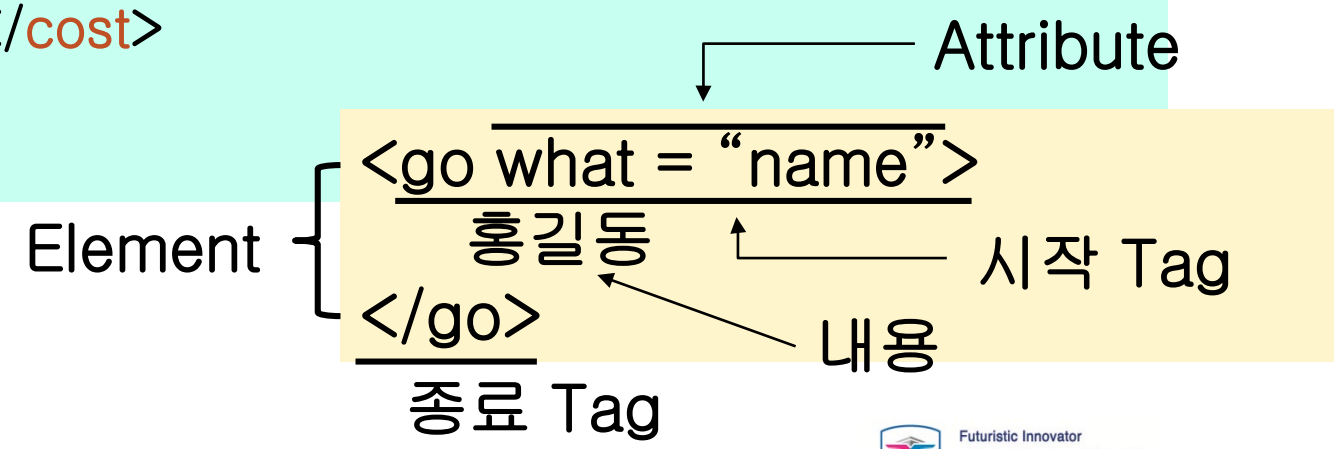
```
<?xml version="1.0" encoding="utf-8" ?>
<booklist>
   <book>
      <title>XML &amp;amp; VisualBasic</title>
   </book>
</booklist>
```

- title Tag 내용인 XML &amp amp VisualBasic 문자열은 PCDATA 이며, XML Parser는 이것을 해석한 후 내장된 Entities를 참조하여 &amp를 &로 치환한 후 아래처럼 익스플로러에 출력

```
<?xml version="1.0" encoding="utf-8" ?>
<booklist>
   <book>
      <title>XML & VisualBasic</title>
   </book>
</booklist>
```

XML 문서 구조

```
<?xml version="1.0" encoding="UTF-8"?>
<shop city="서울" type="마트">
    <food>
        <name>귤</name>
        <sort>과일</sort>
        <cost>3000</cost>
    </food>
    <food>
        <name>상추</name>
        <sort>야채</sort>
        <cost>2000</cost>
    </food>
</shop>
```



- Element(원소)
  - Element는 XML 문서에서 핵심 구성 요소
- Tag(태그)
  - Tag는 Element를 구성하기 위해 사용
  - <Element 이름>과 같은 시작 Tag가 원소의 시작
  - </Element 이름>같은 종료 Tag가 원소의 마지막을 표시
- Attribute(속성)
  - Attribute는 Element에 대한 부가적인 정보를 제공
  - Attribute는 Element의 시작 Tag 안에 위치
  - Attribute는 <Name-Value> 쌍으로 나타냄

XML 문서 구조

- XML 문서에 단 하나만이 존재하는 Root Element를 생성
- 이 Root Element는 XML 문서에 존재하는 모든 Element의 조상(Ancestor) 요소가 됨

```
<shop city="서울" type="마트">
```

XML 문서 구조

- Root Element는 Child Element로 2개의 <food> Element를 가짐

```
<shop city="서울" type="마트">
    <food>
        …
    </food>
    <food>
        …
    </food>
</shop>
```

XML 문서 구조

- 첫 번째 <food> Element는 <name> Element, <sort> Element, <cost> Element의 총 3개의 Child Element를 가짐

```
<food>
    <name>귤</name>
    <sort>과일</sort>
    <cost>3000</cost>
</food>
```

XML 문서 구조

- Element와 Attribute의 차이점

```
<student>
    <name>홍길동</name>     <!-- XML Element  -->
    <year>3</year>
    <major>컴퓨터공학</major>
</student>
```

```
<student name="홍길동">     <!-- XML Attribute  -->
    <year>3</year>
    <major>컴퓨터공학</major>
</student>
```

  - Attribute는 여러 개의 값을 가질 수 없으며, Element처럼 손쉽게 확장할 수 없음
  - Attribute은 XML Tree에 포함되지 않기 때문에 다양한 용도로 활용할 수가 없음

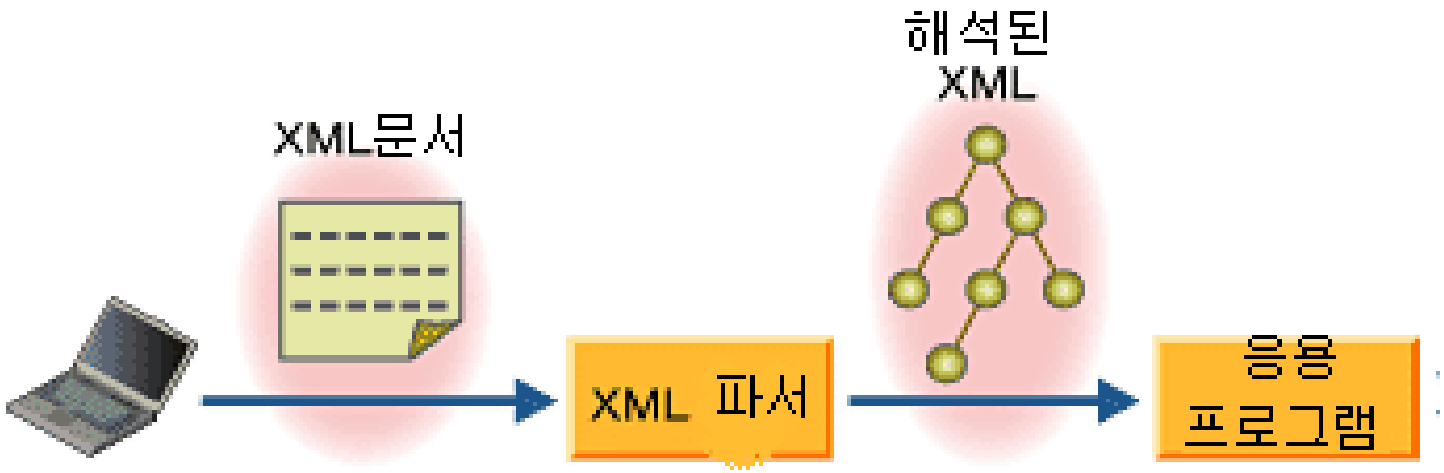
Parsing의 정의와 필요성

- Parsing이란?
  - 주어진 문장을 분석하거나 문법적 관계를 해석하는 것
  - (Syntactic) Parsing(파싱)은 일련의 문자열을 의미 있는 Token(토큰)으로 분해하고 이들로 이루어진 Parse Tree(파스 트리)를 만드는 과정을 말함

- Parsing의 필요성
  - Programming이 간편
  - Platform 독립적(Programming 언어에 구애 받지 않음)
  - 필요한 Data를 빠르게 처리 가능
  - Web 상의 XML이 수정되어도 Application Program을 변경하지 않아도 됨

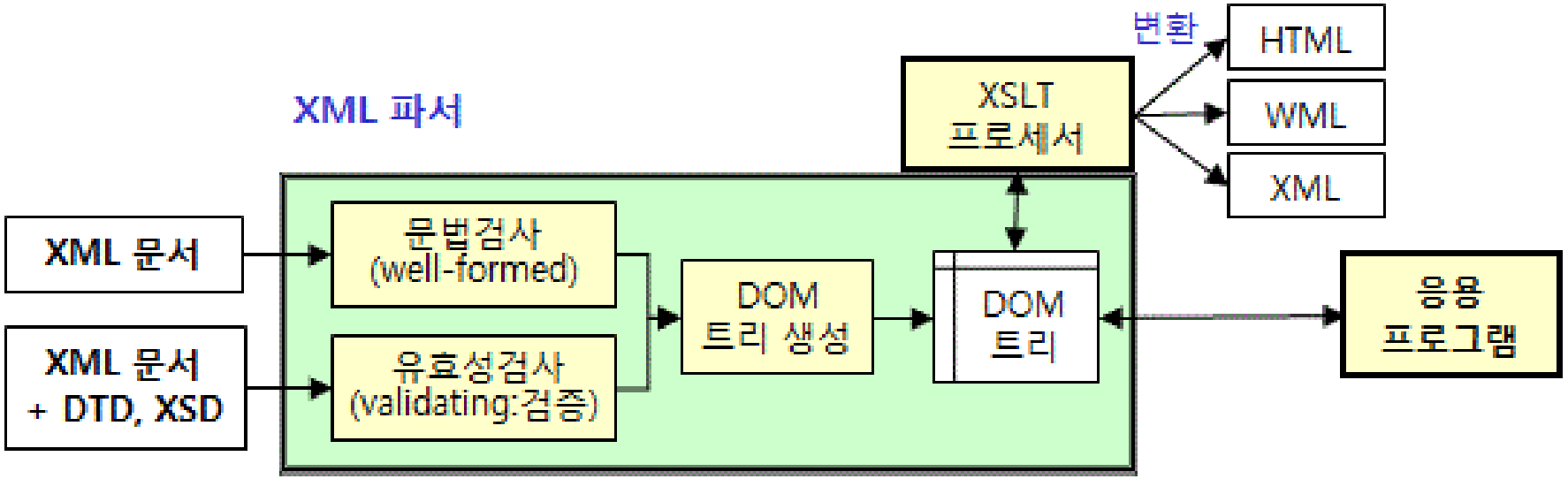
XML Parser

- XML로 구성된 Data들을 Application Program이 유용하게 쓰기 좋은 형태로 만들어 주는 것이 XML Parser
- XML 문서의 구조와 의미를 파악하는 구문 해석기
- XML Parser 종류
  - Tree 기반의 Parser(Tree 기반 Interface와 연계)
  - Event 기반의 Parser(Event 기반 Interface와 연계)



XML Parser

- XML Parser의 역할
  - 문서의 오류 검사(well-formed) : XML 문법 검사
  - 문서의 유효성 검증(validation) : XML 문서가 DTD에서 정의한 대로 작성되었는지 검사
  - Document Tree 생성
  - Application Program Interface



XML Parser

- XML은 Web Service의 기본 Data Format으로 Server와 Client의 중요한 통신수단
- Server는 Client의 요청을 받아들여 처리하고, 그 결과를 XML로 반환하며 Client는 XML을 분석하여 처리 결과를 얻음

- XML 문서 Parsing에 사용되는 3가지 Parser
  - DOM(Document Object Model) Parser
  - SAX(Simple API for XML) Parser
  - XML Pull Parser (상대적으로 이해하기 쉬움)

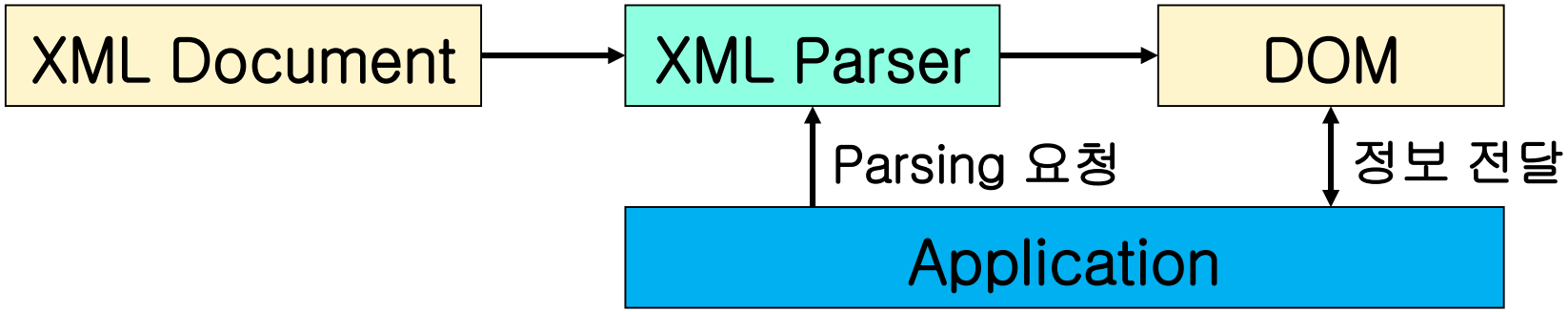
XML Parser

- DOM(Document Object Model)
  - Tree Based API Spec
  - Platform, Programming 언어에 독립적인 Interface
  - 문서의 논리적 구조, 동적 접근, 동적 제어 방법을 정의
  - Element, Contents의 조회, 추가, 수정, 삭제

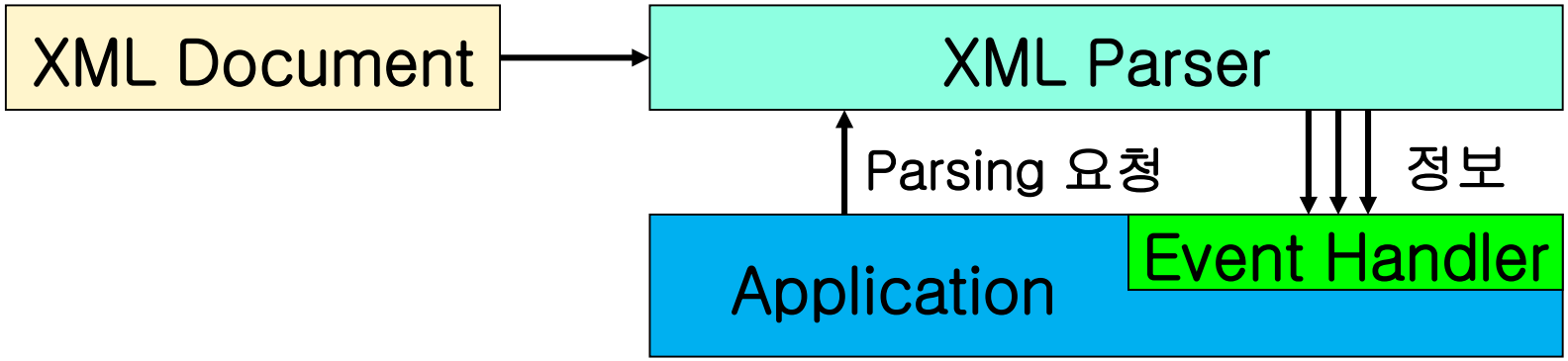
- SAX(Simple API for XML)
  - Event Based의 XML Parsing을 위한 표준 Interface
  - XML-Dev Mailing List 회원들의 제안으로 만들어 졌고, 그 대표 주자가 David Magginson
  - 1998년 5월 11일 SAX 1.0 Release
  - 2000년 5월 5일 SAX 2.0 Release
  - 용량이 큰 XML 문서에서 원하는 Contents를 추출하는데 효율적
  - Read Only

XML Paser

- DOM 접근 방식



- SAX 접근 방식



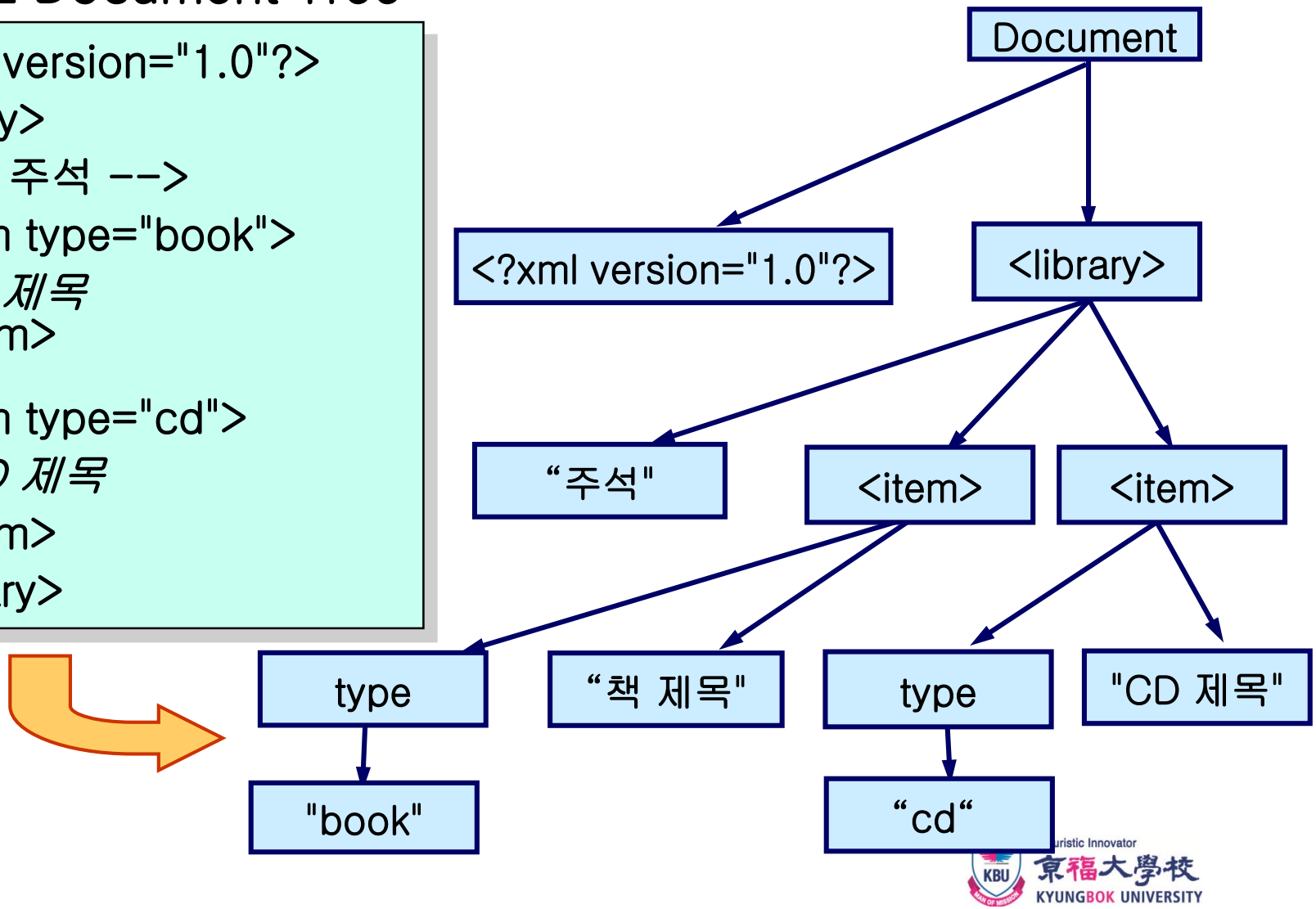
XML Parser

- DOM(Document Object Model) Parser
  - DOM은 WEC에 의해 주관되는 표준 XML 처리 방법
  - DOM은 XML만을 위한 것이 아니라 XML과 같은 Mark-up 형식을 따르는 모든 문서를 처리하기 위한 표준 API
  - DOM은 HTML뿐만 아니라 Style Sheet를 위한 API도 함께 포함하고 있음
  - DOM은 XML 문서를 처리하기 위해 XML 문서를 읽고 Memory에 Loading. Memory에 Loading된 XML은 Tree 구조를 가지며, Tree 기반의 API를 이용해서 XML 문서를 처리할 수 있음
  - Tree 형식으로 문서를 읽어서 전체 구조를 파악한 후 정보를 구하는 방식
  - Element를 모두 Tree 구조로Memory에 넣어두고 사용함
  - DOM은 버전으로 구분되지 않고 Level에 의해 구분되는데, 현재 Level 3 일부가 권고안으로 확정된 상태

XML Parser

- XML Document Tree

```
<?xml version="1.0"?>
<library>
  <!-- 주석 -->
  <item type="book">
     책 제목
  </item>

  <item type="cd">
     CD 제목
  </item>
</library>
```



XML Parser

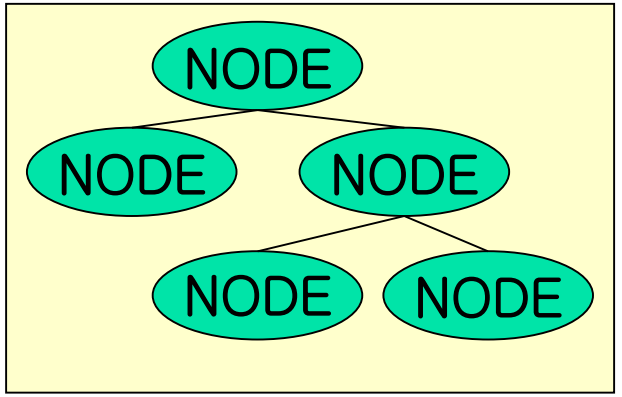
- Tree-Based

```
<?xml version="1.0" …?>
<연구실 이름 = "xmlab">

  <담당교수>
    <이름>홍길동</이름>
  </담당교수>

  <학생>
    <이름>이몽룡</이름>
    <과정>석사2학기</과정>
  </학생>

</연구실>
```

Parsing

DOM Parser

DOM Tree 구조



XML Parser

- DOM(Document Object Model) Parser 장단점
  - 장점
    - Memory에 Tree 구조로 정보가 들어 있기 때문에 한번 Parsing 해두면 아무 때나 얻고 싶은 Element에 대한 정보를 얻을 수 있음
    - 특정 Node에 대한 임의 접근이 자유로움
    - 원하는 Node를 몇 번이고 읽을 수 있음
    - 문서의 수정도 가능
  - 단점
    - Memory의 소모가 다른 방법보다 많음
    - 처음 시작이 다소 느림
      (문서 전체를 Loading해야 하므로)

XML Parser

- DOM Spec의 구분
  - DOM Level 1
    - XML과 HTML 문서 처리를 위한 API
  - DOM Level 2
    - Core – XML과 같은 Mark-up 언어를 처리하기 위한 API
    - Views – 문서의 표현 방법을 위한 API
    - Events – Node에 대한 Event를 처리하기 위한 API
    - Style – CSS와 관련된 API
    - Traversal and Range – XML 문서 탐색에 관련된 API
    - HTML – HTML 문서 처리를 위한 API
  - DOM Level 3
    - Core – XML과 같은 Mark-up 언어를 처리하기 위한 API
    - Load and Save – XML 문서의 저장과 Loading에 관련된 API
    - Validation – 문서의 유효성 검사를 위한 API 제공

XML Parser

- SAX(Simple API for XML) Parser
  - SAX Parser는 DOM Parser가 XML 문서를 Memory에 Loading하기 때문에 처리 속도가 느리고, XML 문서의 크기가 클 경우 많은 양의 Memory를 필요로 하는 단점을 극복하기 위해 만들어진 것
  - DOM은 XML 문서를 Memory에 Loading한 후 처리하는 방식을 사용하는 반면, SAX는 XML 문서를 읽어가면서 Event를 호출하여 처리하는 Event Based 처리 방식
  - 현재 SAX는 2.0까지가 발표된 상태이며, 이는 http://www.saxproject.org에서 확인할 수 있음
  - XML 문서를 처리하기 위한 필요한 Event를 등록하면 SAX Parser가 XML 문서를 읽으면서 동시에 등록된 Event에 해당하는 부분을 만났을 때 Event가 호출되는 방식으로 동작

XML Parser

- SAX(Simple API for XML) Parser
  - Event 기반의 Parser로 문서의 시작과 끝, Element의 시작과 끝, Element의 내용 등 Element Tag의 이름에 따라 각각을 처리하는 메소드를 두어 Parsing 함
  - SAX Parser를 초기화 하는 방법은 DOM Parser와 거의 유사하며, BuilderFactory부터 Builder를 생성하고 Builder로부터 Parser 객체를 얻음
  - SAX는 문서를 순서대로 읽으면서 Event를 발생시키는 식이라 Memory를 거의 사용하지 않으며 기동 속도가 빠름
  - SAX는 읽기만 하는데 비해 DOM은 Node를 삽입할 수도 있다는 차이점이 있으나 Mobile 환경에서는 주로 XML 문서를 읽기만 하므로 큰 의미는 없음
  - SAX Parser와 DOM Parser는 장단점이 뚜렷하게 구분되므로 읽고자 하는 문서의 성격에 맞는 Parser를 잘 선택해야 함

XML Parser

- Event-Based Parsing

```
<?xml version="1.0" …?>
<연구실 이름 = "xmlab">

  <담당교수>
    <이름>홍길동</이름>
  </담당교수>

  <학생>
    <이름>이몽룡</이름>
    <과정>석사2학기</과정>
  </학생>

</연구실>
```

SAX Parser

<학생>을 찾으면 저를 불러 주세요!!!

Method

Call-Back!!!

화면에 "찾았다"를 출력한다.

Parsing

"찾았다"

XML Parser

- SAX(Simple API for XML) Parser 장단점
  - 장점
    - Line 단위로 Parsing하기 때문에 Parsing하는데 적은 Memory 소요
  - 단점
    - Parsing시 그냥 지나갔던 Element 정보를 다시 얻고 싶으면 다시 Parsing해야 함
- DOM은 Memory를 많이 사용하지만 성능이 좋고, SAX는 느리지만 Memory를 거의 사용하지 않음

XML Parser

SAX Parser 발전 과정

SAX 개발	개발 내용
개발 시작	✓ 1977년 12월(XML 1.0 초안이 채택된 직후) ✓ Open Source 진영의 XML 개발자들이 개발 주도 ✓ 다수의 XML Parser에 탑재되어 개발자들에게 폭넓게 인정받은 표준 API
SAX 1.0 API	✓ 1998년 5월 발표 ✓ 알프레드(Alfred), 데이비드 메긴슨(David Magginson), 팁 브레이(Tim Bray), 제임스 클락(James Clark) 등 참여, XML-DEV에서 개발 ✓ NameSpace를 지원하지 않음 ✓ Bug가 있고, 기능 제약이 있음
SAX 2.0 API	✓ 2000년 5월 발표 ✓ SAX 1.0의 핵심 API는 거의 변경하지 않고, 필요한 클래스와 Interface만 추가

XML Parser

- XML Pull Parser
  - DOM, SAX 외에도 Android는 XmlPullParser라는 것을 제공
  - SAX와 같이 Event 기반의 Parser이지만, SAX와 달리 문서에 대해 모두 Parsing을 하지 않고도 특정 부분까지의 Parsing 내용을 활용할 수 있음
  - 장점
    - 원하는 부분을 Parsing할 수 있음
    - 상대적으로 이해하기 쉬움
  - 단점
    - SAX의 단점을 가지며 SAX보다 약간 느림

XML Parser

구분	DOM	SAX
파싱방법	Tree-Working기반	Event기반
처리방법	메모리에 로딩 Tree 자료 구조화	순차적인 이벤트 발생 처리
장점	문서 구조 동적 변경 가능 복잡한 처리 연산 가능 문서 생성 및 편집 가능	XML 문서의 크기에 관계없이 파싱 가능 자신만의 데이터 구조 생성 가능 XML 문서의 일부만 처리 가능 단순하고 속도가 빠름
단점	메모리 요구량이 많음 처리 속도가 느림	문서의 구조에 대한 정보 파악이 어려움 문서 생성, 편집이 불가능
적용분야	구조적 접근이 필요한 경우 특정 부분으로 이동할 경우 문서 정보를 쉽게 파악하고자 하는 경우	문서 일부분만 읽는 경우 유효성 처리, 데이터 변환이 필요한 경우 동일 오류 처리가 필요한 경우 Element를 일부 추출하는 경우
구조변경	데이터 구조 변경 가능	데이터 구조 변경 불가
접근방식	Random Access 방식	Streaming 방식

XML Parser

Parser	특징	장점	단점
DOM (Document Object Model) Parser	Element를 모두 Tree구조로 메모리에 넣어두고 사용함	메모리에 Tree구조로 정보가 들어있기 때문에 한번 파싱해두면 아무때나 얻고 싶은 Element에 대한 정보를 얻을 수 있음	문서가 커지면 메모리를 치나치게 많이 소모함
SAX (Simple API for XML) Parser	이벤트기반의 파서로 문서의 시작과 끝. Element의 시작과 끝, Element의 내용 등 Element Tag의 이름에 따라 각각을 처리하는 메소드를 두어 파싱함	라인 단위로 파싱하기때문에 파싱하는데 적은 메모리 소요	파싱시 그냥 지나갔던 Element의 정보를 얻고 싶으면 다시 파싱해야 함
XML Pull Parser	SAX와 같이 이벤트 기반의 파서이지만, SAX와 달리 문서에 대한 모든 파싱을 하지 않고도 특정부분까지의 파싱 내용을 활용할 수 있음	원하는 부분을 파싱할수있음	SAX의 단점을 가지며 SAX보다 약간 느림

XML Parser

- 주요 Parsers
  - Apache Xerces
  - IBM XML4J
  - Oracle XML Parser
  - Sun Microsystems Project X (!=JAXP)
  - JamesClark's XP
  - OpenXML
  - MicroSoft's MSXML
  - …

XML Parser

- JAXP(Java API for XML Processing)
  - SUN에서 제안한 표준 JAVA API

- JAXP의 필요성
  - 각 Vendor의 Parser가 각기 조금씩 다른 API를 제공
  - J2EE기반에서 사용할 표준 XML API 필요
  - JAXP는 각 Vendor가 제공하는 Parser를 사용할 수 있는 추상적인 층(abstract layer)를 제공

XML Parser

- XML4J(Xerces-J)
  - IBM
  - TX-Parser로 연구 시작
  - Apache Project에 참여
  - 현재 Apache Xerces-J를 Test하고, 배포
  - JAVA로 쓰여진 대표적인 XML Parser
  - Parser Download
    - http://www.alphaworks.ibm.com/formula/로 가서 XML4J를 Download
  - JAVA 환경이 필요
    - JDK 설치 확인
    - CLASSPATH에 XMLParser Class가 설정되었는가 확인 필요