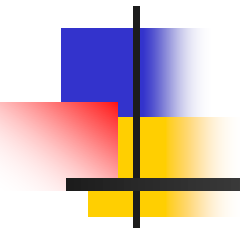


SAX Parser(이론)

배 희호 교수
경복대학교
소프트웨어융합과

SAX Parser

- Android에서 XML Data를 처리하는 또 다른 방식으로, Event 기반의 SAX(Simple API for XML) Parser를 제공
- SAX Parser는 Memory 효율성을 중시하며, XML 문서를 순차적으로 읽어 Event를 Trigger하여 Data를 처리

- DOM 방식 단점을 극복하는 대안으로 등장
  - DOM 단점
    - XML 문서가 커질 경우 Memory 소모가 큼
  - DOM 방식과 달리 XML 문서를 Memory에 Load하지 않으므로, Memory 사용량이 적어 큰 XML File을 처리하는 데 적합

SAX Parser

- Event-Driven Programming
  - Parsing Event가 발생하면, Parser는 해당 Handler의 해당 Method를 호출함
  - SAX는 문서를 읽고 발생하는 각 요소 또는 Text Block에 대한 Handler Method를 호출하여 처리함
  - Handler들은 표준 JAVA API로 구현
- XML 문서 전체를 처음부터 끝까지 차례대로 순차적으로 읽어 들이면서 처리하는 구조
  - 적은 Memory를 가진 System에서 XML 문서를 처리할 수 있으므로 이점

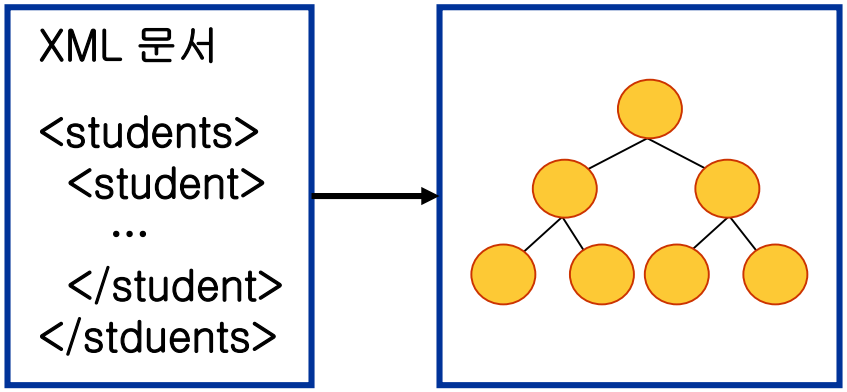
SAX Parser

SAX Parser 발전 과정

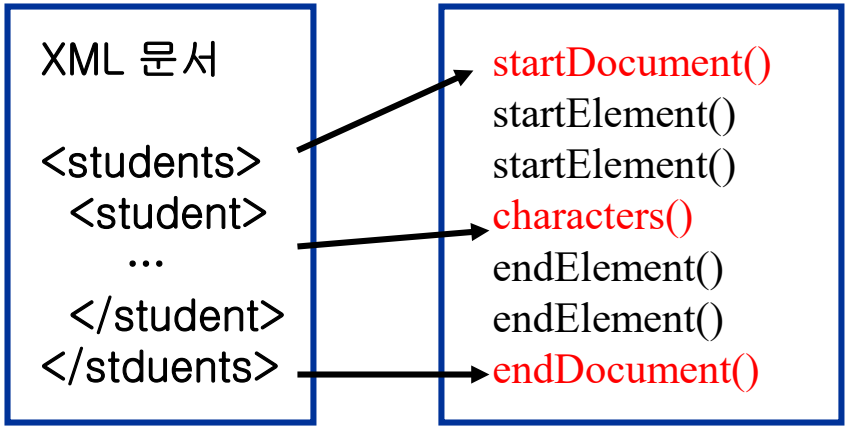
SAX 개발	개발 내용
개발 시작	• 1977년 12월(XML 1.0 초안이 채택된 직후) • Open Source 진영의 XML 개발자들이 개발 주도 • 다수의 XML Parser에 탑재되어 개발자들에게 폭넓게 인정 받은 표준 API
SAX 1.0 API	• 1998년 5월 발표 • 알프레드(Alfred), 데이비드 메긴슨(David Magginson), 팁 브레이(Tim Bray), 제임스 클락(James Clark) 등 참여, XML-DEV에서 개발 • Name Space를 지원하지 않음 • Bug가 있고, 기능 제약이 있음
SAX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

SAX Parser

- DOM 처리 방식(Tree 기반 API)



- SAX 처리 방식(Event 기반 API)
  - Parsing Event를 callback을 통해서 Application에 직접 전달하는 구조



SAX Parser

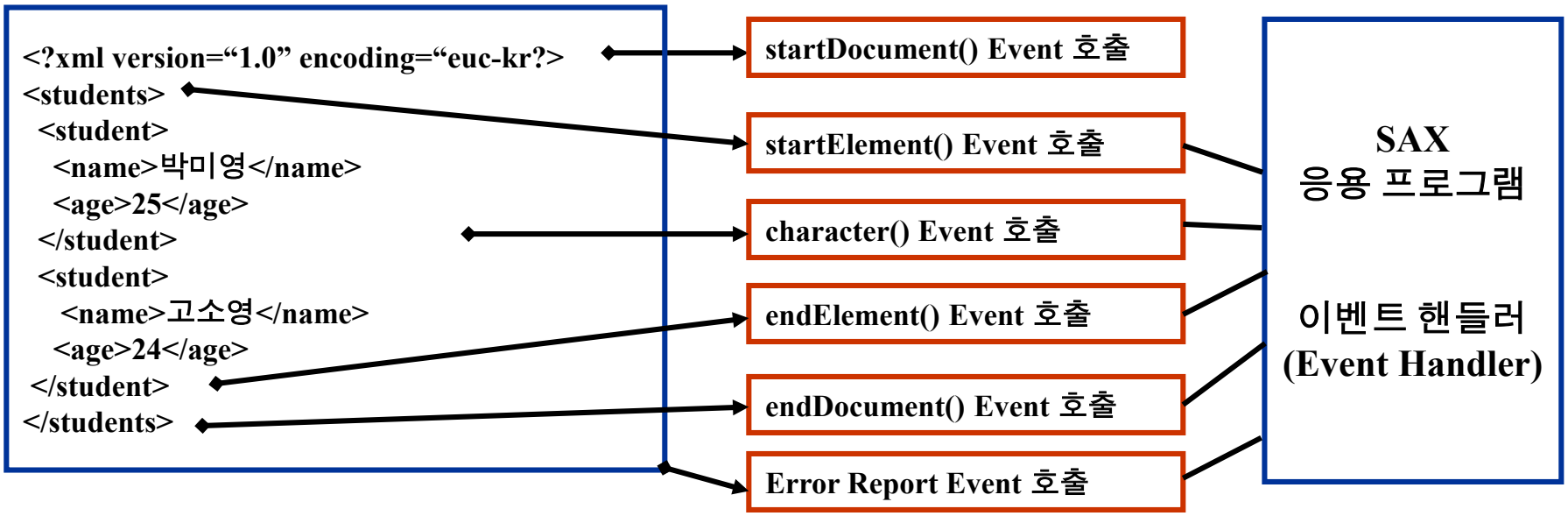
- DOM Parser는 문서의 모든 내용을 Memory에 Tree 형태로 펼친 후 읽기 때문에 속도가 대단히 빠르며 임의의 Node를 여러 번 읽을 수 있다는 장점이 있음
- SAX는 문서를 순서대로 읽으면서 Event를 발생시키는 방식이라 Memory를 거의 사용하지 않으며 기동 속도가 빠름
  - SAX는 읽기만 하는데 비해 DOM은 Node를 삽입할 수도 있다는 차이점이 있으나 Mobile 환경에서는 주로 XML 문서를 읽기만 하므로 큰 의미는 없음
- 두 Parser는 장단점이 뚜렷하게 구분되므로 읽고자 하는 문서의 성격에 맞는 Parser를 선택하면 됨
- SAX Parser를 초기화하는 방법은 DOM Parser와 거의 유사하며, BuilderFactory로부터 Builder를 생성하고 Builder로부터 Parser Object를 얻음

SAX Parser

- SAX Parser의 주요 특징
  - Event-Driven 방식
    - XML 문서를 읽으면서 특정 Event(Tag 시작, Tag 끝, Text 등)가 발생할 때 Callback Method가 호출
  - Memory 효율적
    - XML File을 한 번에 Memory에 Load하지 않고 Streaming 방식으로 처리
  - 단 방향 Parsing
    - XML 문서를 순차적으로 처리하며, 과거 Data를 참조하거나 문서를 역방향으로 탐색할 수 없음

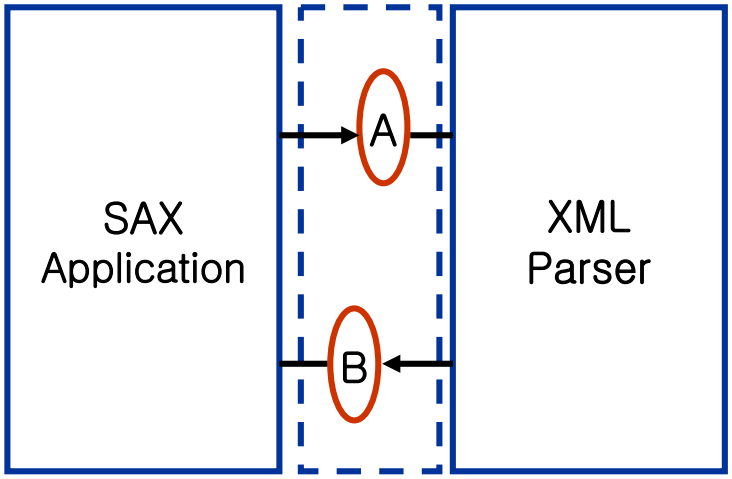
SAX Parser

SAX 처리 구조

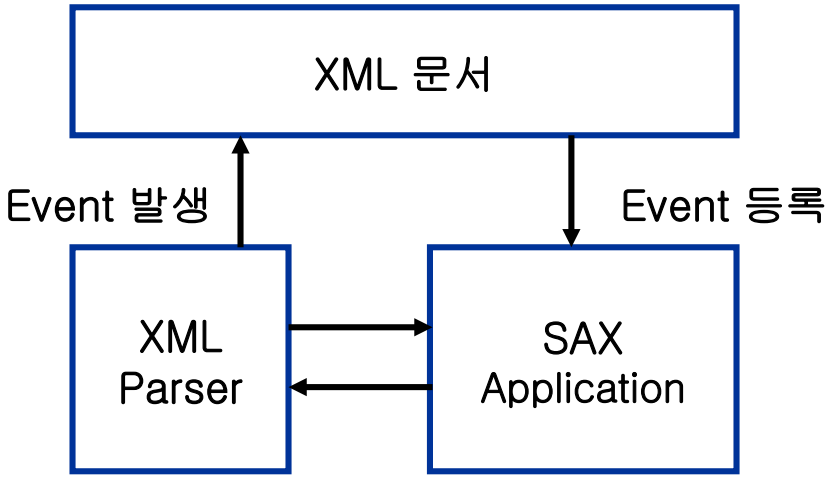


SAX Parser

- SAX Interface 구조와 Event Programming



Interface



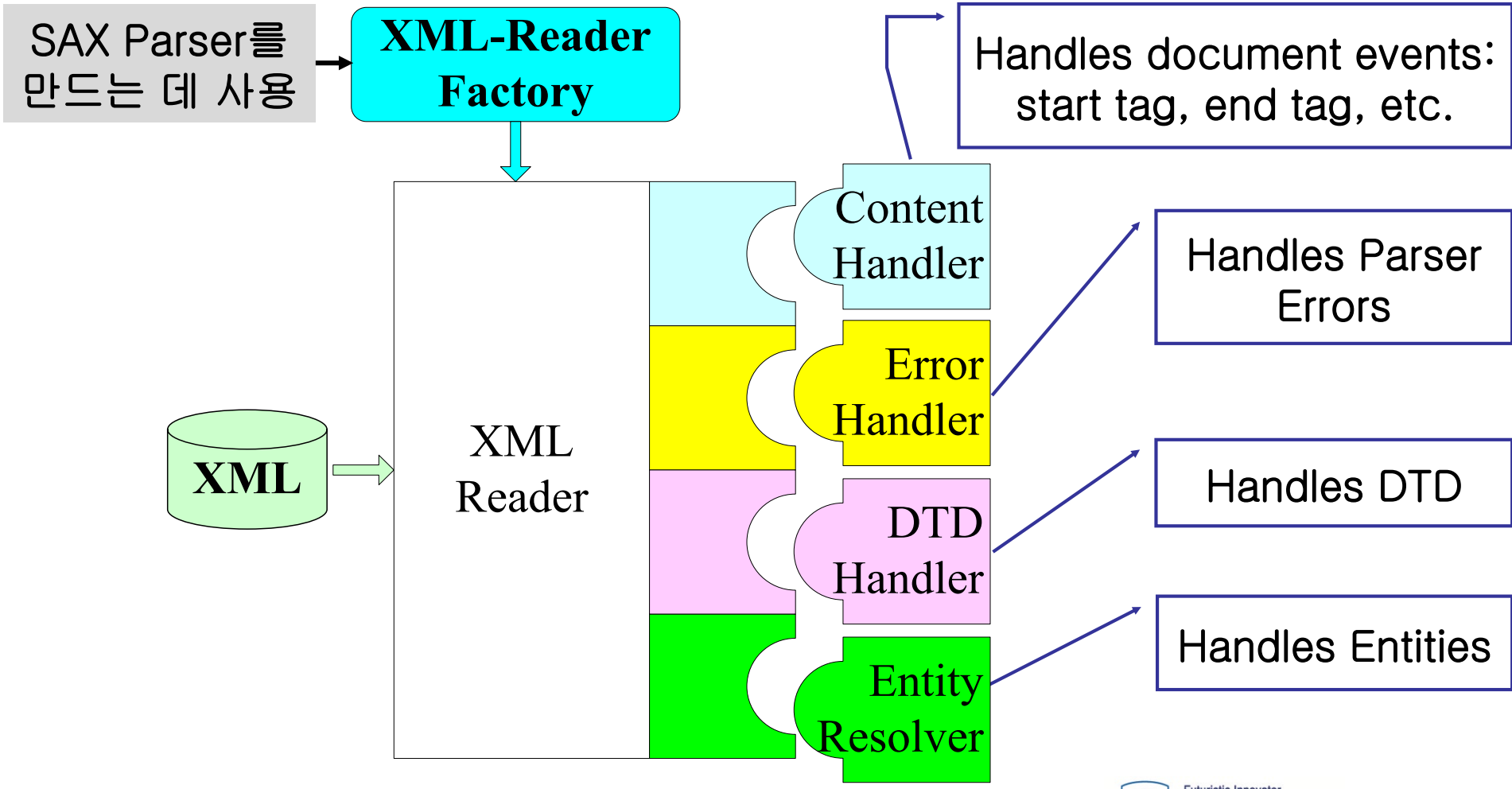
Event Handler 실행

- A: XML Parser가 구현, XML Application이 호출하는 Interface
- B: XML Application이 구현하고, XML Parser가 Event로 호출하는 Interface = CallBack Interface

SAX Parser

- Parser Object로부터 Reader Object를 구하고 Reader에 내용 Handler를 부착하면 문서를 읽으면서 정보가 발견될 때마다 Handler로 Event를 보내줌
  - Handler로 전달되는 Event는 Method 이름으로 쉽게 알 수 있음

SAX Parser



SAX Parser

- SAX 기본 패키지(org.xml.sax) Interface

인터페이스	내용
Attributes	요소 시작 태그 또는 빈 요소 내의 속성을 표현
ContentHandler	XML 문서 Parsing중에 마크업이나 문자 Data를 만났을 때 호출되는 Event Handler의 인터페이스
DTDHandler	XML 문서를 Parsing중에 DTD에 포함된 노테이션 선언이나 파싱되지 않은 개체 선언을 만났을 때 호출되는 Event Handler의 인터페이스
EntityResolver	외부 DTD 서브셋, 외부 파라미터 개체 참조, 내부 파서 파라미터 개체 참조 등을 만났을 때 호출되는 Event Handler의 인터페이스
ErrorHandler	XML 문서를 파싱하는 도중에 만난 파서 에러를 처리하는 Event Handler의 인터페이스
Locator	SAX Event가 어느 개체의 어느 위치에 발생 했는지의 정보를 접근하는데 필요한 메소드들을 정의하는 인터페이스
XMLFilter	대개 SAX 파서 개발자가 개발
XMLReader	XMLReader를 상속하는 인터페이스

SAX 기본 패키지(org.xml.sax) 클래스

클래스	내용
InputSourse	XML 문서를 구성하는 외부 개체 하나를 표현하는 클래스
SAXException	SAX Parser의 작동 중에 발생하는 예외 클래스
SAXNotRecognizedException	SAXException을 상속하는 Class
SAXNotSupportedException	SAX Parser가 알 수 없는 특징이나 속성을 요청할 때 발생
class SAXParseException	SAXException을 상속하는 Class

SAX 확장 Package(org.xml.sax.ext) 인터페이스

인터페이스	내용
DeclHandler	XML 문서를 Parsing하는 도중 DTD에 포함된 개체 선언, 요소 목록 선언, 개체 선언을 만났을 때 호출되는 Event Handler의 인터페이스
LexicalHandler	XML 문서를 Parsing하는 도중에 만나는 마크업 중 ContentHandler에 보고되지 않는 마크업을 처리하기 위한 Event Handler의 인터페이스

구현 클래스 패키지(org.xml.sax.helpers) 클래스

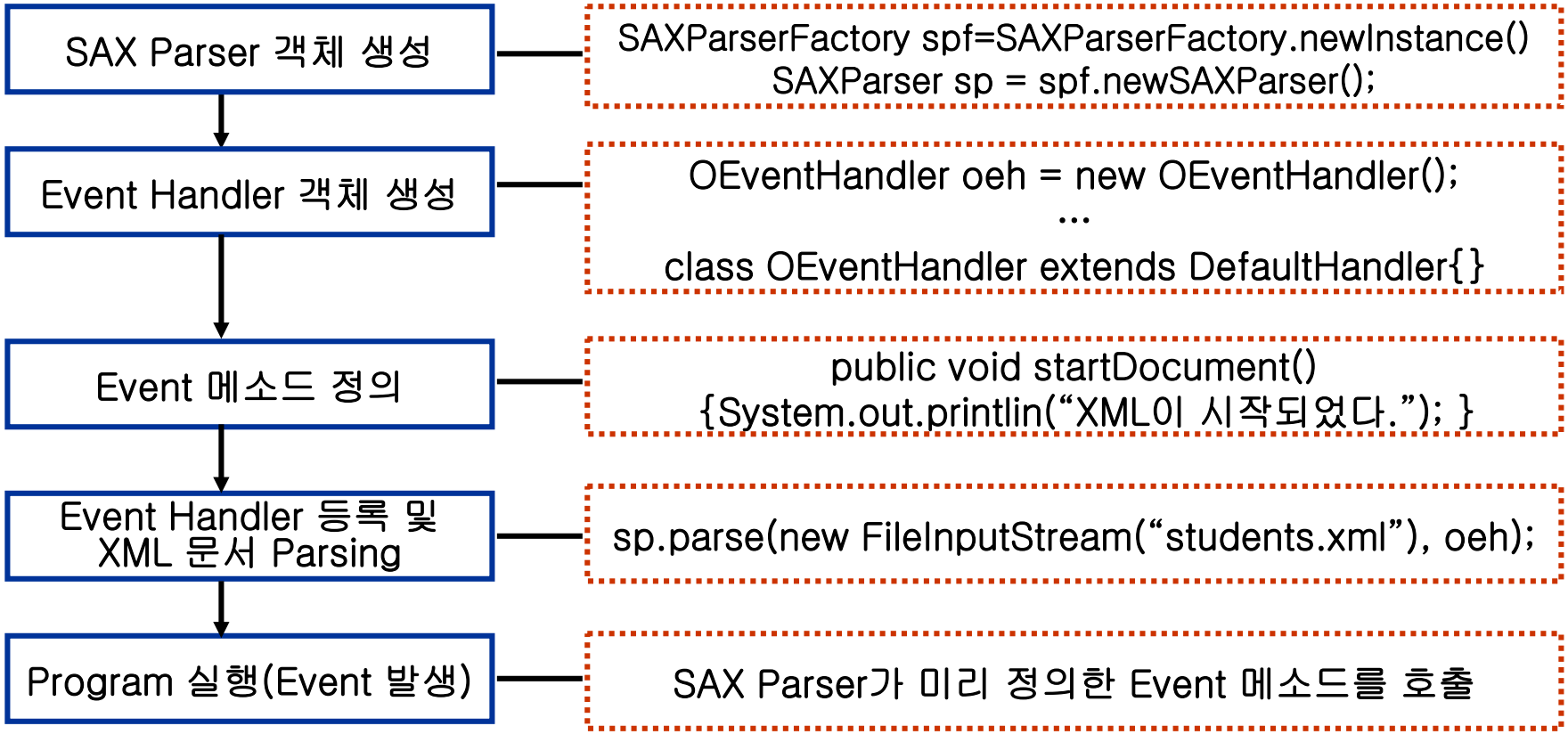
클래스	내용
AttributesImpl	Attributes 인터페이스의 구현 클래스
DefaultHandler	ContentHandler, DTDHandler, EntityResolver, ErrorHandler 인터페이스를 모두 구현하는 클래스
LocatorImpl	Locator 인터페이스의 구현 클래스
NamespaceSupport	SAX 파서가 사용하는 네임스페이스 로직을 캡슐화한 클래스
ParserAdapter	SAX 1.0의 인터페이스인 Parser를 XMLReader로 사용할 수 있도록 하는 클래스
XMLFilterImpl	XMLReader 인터페이스와 DocumentHandler 인터페이스 구현
XMLReaderAdapter	XMLFilter 인터페이스와 EntityResolver, DTDHandler, ContentHandler, Errorhandler 인터페이스를 모두 구현하는 클래스
XMLReaderFactory	XMLReader를 SAX 1.0의 인터페이스인 Parser로서 사용할 수 있도록 하는 클래스

JAXP의 SAX 파서 구동 API 패키지(javax.xml.parser) 클래스

클래스	내용
DocumentBuilder	DOM Parser를 추상화한 클래스
DocumentBulderFactory	DOM Paser를 생성하는데 필요한 API를 정의한 추상 클래스
SAXParser	SAX Parser를 추상화한 클래스 XMLReader 구현 클래스의 레퍼(wrapper) 역할을 함
SAXParserFactory	SAX Parser를 생성하는 데 필요한 API를 정의한 추상 클래스
ParserConfigurationException	Parser 생성 중에 Parser가 지원하지 못하는 구성 설정 요청을 발견하였을 때 발생하는 예외 상황
FactoryConfigurationError	ParserFactory 구성에 문제 있을 때 발생하는 에러 Parser를 구현하는 클래스 파일이 존재하지 않을 때 발생

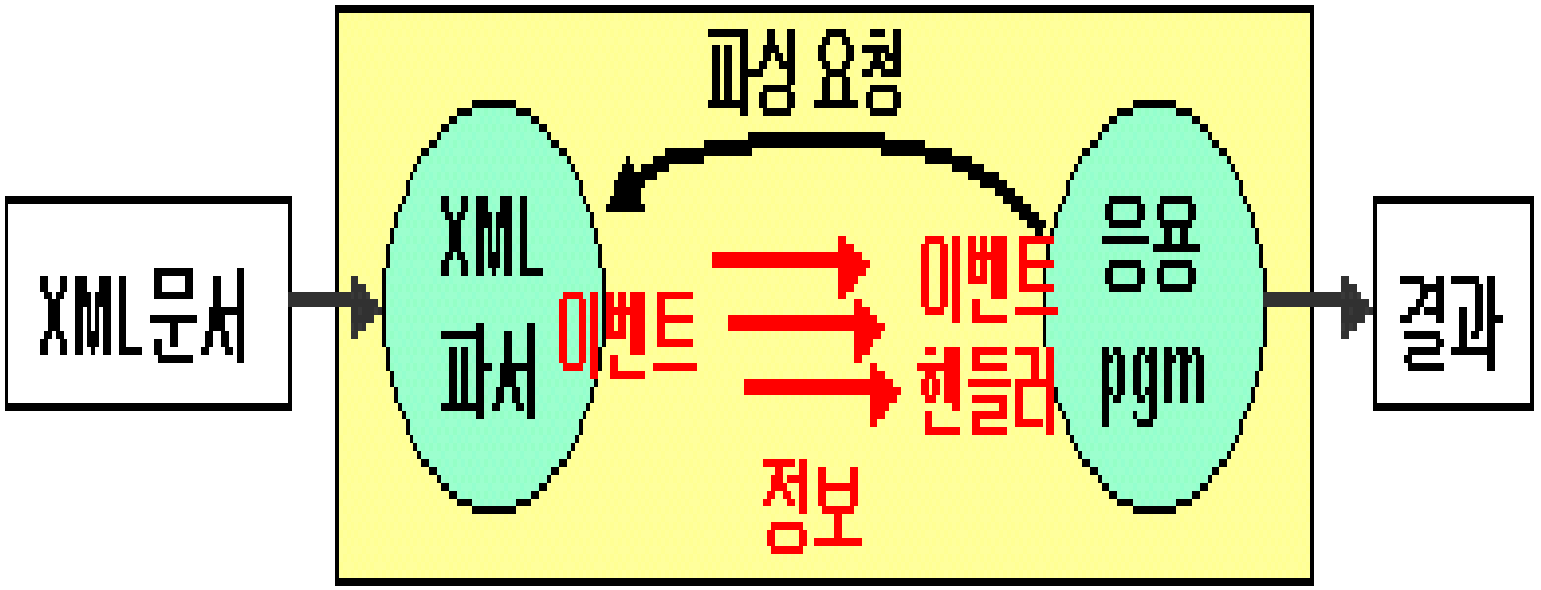
SAX Parser

SAX Progam 구조



SAX Parser

- SAX Parsing 방법



- 마크업(Mark-Up)의 유래 :
  환자의 식자를 위한 수기형태의 주석
- 마크업 언어 :
  문서의 구조와 내용에 추가적인 의미를 부여하는 마크업 규칙을 규정하는 언어

SAX Parser

- SAXParser를 사용하는 기본적인 형태
  - SAXParserFactory Object를 얻음
  - SAXParser Object를 생성
  - XMLReader Object를 얻음(선택)
  - DefaultHandler를 상속받은 Handler Class를 정의
  - XMLReader의 Handler를 지정하고, XMLReader의 parse() Method에 InputSource를 던지면 해당 XML의 Data가 Handler를 통해 Parsing 됨
  - Parsing된 Data를 필요한 위치에 사용

  - DefaultHandler Class
    - SAX를 통해 XML을 Parsing하기 위한 Event Handler를 포함한 Class

SAX Parser

- DOM과 SAX는 구분되는 특징

구분	DOM	SAX
Parsing 기반	Tree 기반	Event 기반
Data 접근 방식	랜덤	순차
Memory 사용 방식	Data 크기에 비례해서 증가	Data 크기와 상관없이 일정 Memory 사용
적합한 Data	경량 Data	경량/대용량 Data
Data 재사용	가능	불가능 (재 파싱해서 사용할 수 있음)

SAX Parser

- 주요 Class와 구성
  - Android에서 SAX Parsing을 수행하기 위해 다음 Class를 사용
  - SAXParserFactory
    - SAXParser Object를 생성하는 Factory Class
  - SAXParser
    - XML 문서를 Parsing하는 핵심 Class
  - DefaultHandler
    - XML 문서의 Event(Tag 시작, 끝, Text 등)를 처리하기 위한 Handler Class를 상속받아 사용

SAXParserFactory 클래스

- SAXParserFactory는 JAVA에서 SAX(Simple API for XML) Parser를 생성하기 위한 Factory Class
- 이 Class는 javax.xml.parsers Package에 속하며, SAX Parser를 생성하고 구성하는 데 사용
- SAXParserFactory를 통해 생성된 Parser는 XML 문서를 Event 기반으로 처리하며, Memory 효율적이고 Streaming 방식으로 XML Data를 다룸
- Source Code의 변화 없이도 다른 Vender의 Parser를 사용할 수 있게 하기 위해 Apache Group의 Xerces에서 제공하는 기능
- XML 문서 처리에 필요한 SAXParser를 생성, SAXParser가 Parsing하는데 필요한 여러 선택 사항을 만드는 역할

SAXParserFactory 클래스

- Method

메소드	설명
static SAXParserFactory newInstance()	SAXParserFactory 객체를 생성시키는데 필요한 static 메소드
abstract SAXParser newSAXParser()	SAXParser 객체를 생성
void setValidating(boolean)	Parsing 할 XML 문서의 유효성을 검사할 것인지 지정 (기본값은 false)
void setNamespaceAware (boolean)	Parser가 Namespace를 지원할 것인지 지정
abstract setFeature (String name, boolean)	SAX 2.0에서 지원하는 특정 기능을 사용할 것인지 지정
boolean isNamespaceAware()	현재 Factory가 Namespace를 지원하는지 확인
boolean isValidation()	현재 Factory가 XML 문서 유효성 확인을 지원하는지 확인
boolean getFeature (String name)	현재 Factory가 SAX 2.0에서 지원하는 특정 기능을 사용하는지 확인

SAXParser 클래스

- SAXParser Class는 JAVA의 javax.xml.parsers Package에 포함된 XML Parser로, SAX(Simple API for XML) Parsing을 수행하는 데 사용
- 이 Class는 Event 기반의 XML Parsing을 제공하며, Streaming 방식으로 XML Data를 처리하여 Memory 효율적
- SAXParser Object의 생성은 SAXParserFactory.newSAXParser() Method 사용
- SAXParser Object가 XML 문서를 Parsing하려면 Event 처리기인 DefaultHandler로 Event가 전달되어 미리 정의된 Event Method 호출
- SAXParser의 Method 중 대부분은 parse() Method의 Overload로 이루어져 다양한 Source에서 XML 문서 처리가 가능

SAXParser 클래스

- Method

메소드	설명
void parse (File f, DefaultHandler dh)	명시된 DefaultHandler를 사용하여 XML File을 파싱
void parse(InputSource is, DefaultHandler dh)	명시된 DefaultHandler를 사용하여 InputSource에서 XML을 파싱. InputSource로는 InputStream, Reader, String이 가능
void parse(InputStream is, DefaultHandler dh)	명시된 DefaultHandler를 사용하여 InputStream에서 XML을 파싱
void parse(Sting uri, DefaultHandler dh)	명시된 DefaultHandler를 사용하여 URI에서 XML을 파싱
void setProperty(String name, boolean value)	SAX 파서가 제공하는 특정 기능 사용 여부를 지정
boolean getProperty (String name)	SAX 파서가 제공하는 특정 기능 사용 여부를 확인
boolean isNamespaceAware()	SAX 파서가 네임스페이스를 지원하는지 확인
boolean isValidating()	SAX 파서가 문서 유효성 검사를 하는지 확인

XMLReader 인터페이스

- SAX Parser를 위한 기본적은 기능 요구사항을 선언하는 Interface
- SAX2를 사용하는 JAVA Program은 XMLReader Interface를 구현하는 Class를 시용해서 XML 문서를 Parsing할 수 있음
- 표준화된 API, XML Name Space 동기화된 Method 지원

XMLReader 인터페이스

- Method

메소드	설명
ContentHandler getContentHandler()	현재 컨텐츠 핸들러를 반환
DTDHandler getDTDHandler()	현재 DTD 핸들러를 반환
EntityResolver getEntityResolver()	현재 개체 리졸버(resolver)를 반환
ErrorHandler getErrorHandler()	현재 에러 핸들러를 반환
boolean getFeature(String name)	특성값을 반환
Object getProperty(String name)	속성값을 반환
void parse(InputSource input)	XML 문서를 파싱
void parse(String systemId)	시스템 식별자(URI)로 지시된 XML 문서를 파싱
void setContentHandler (ContentHandler handler)	컨텐츠 핸들러를 설정

XMLReader 인터페이스

- 메소드

메소드	설명
void setDTDHandler (DTDHandler handler)	DTD 핸들러를 설정
void setEntityResolver (EntityResolver resolver)	개체 리졸버(resolver)를 설정
void setErrorHandler (ErrorHandler handler)	에러 핸들러를 설정
void setFeature (String name, boolen value)	특성값을 설정
void setProperty (String name, Object value)	속성값을 설정

ContentHandler 인터페이스

- SAX에서 XML 문서의 Event를 받는 Interface
- DefaultHandler Class와 같은 Method를 가지고 있음
- XML Parser가 문서를 Parsing하는 동안에 ContentHandler Interface에 정의된 Method들이 자동적으로 호출

DefaultHandler 클래스

- DefaultHandler는 JAVA에서 SAX(Simple API for XML) 기반의 XML Parsing을 수행할 때 사용하는 기본 Handler Class
  - ContentHandler, DTDHandler, EntityResolver, ErrorHandler Interface를 모두 구현하는 Class
- 이 Class는 org.xml.sax.helpers.DefaultHandler Package에 포함되어 있으며, SAX Event(Tag 시작, 끝, Text Data 등)를 처리하기 위한 Method를 제공
- DefaultHandler는 SAX Parsing Event를 처리하는 기본 구현을 제공하며, 필요에 따라 특정 Method를 Override하여 사용자 정의 Event 처리가 가능
- 일반적으로 Programmer는 XML 문서를 처리하기 위해서 DefaultHandler Class로부터 상속받는 Class를 작성하고, DefaultHandler에서 필요한 Method를 Overriding 함

DefaultHandler 클래스

- Method

메소드	설명
void characters(char[] ch, int start, int length)	✓ 요소의 문자 Data를 만나면 characters() 메소드가 호출 ✓ Text 내용은 ch 배열의 start Index에서 length개의 문자에 해당됨
void endDocument()	✓ 문서가 끝나는 경우에 호출
void endelement(String uri, String localName, String qName)	✓ 요소가 끝나는 경우 호출 ✓ localName은 XML 네임스페이스의 접두어를 제외한 태그 이름을 의미하고, qName은 XML 네임스페이스의 접두어를 포함한 태그 이름을 의미
void endPrefixMapping (String prefix)	✓ 네임스페이스 매핑이 끝나는 경우에 호출
void error(SAXParseException e)	✓ 복구할 수 있는 파서 에러가 발생할 때 호출
void fatalError(SAXParseException e)	✓ 치명적인 파싱 에러가 발생하는 경우 호출

DefaultHandler 클래스

- Method

메소드	설명
void ignorableWhitespace (char[] ch, int start, int length)	✓ 요소 내용에서 공백문자를 무시하라는 내용이 있을 때 호출
void notationDecl(String name, String publicId, String systemId)	✓ 노테이션 선언이 있을 때 호출
void processingInstrcution (String target, String data)	✓ 처리 명령어(PI)가 있을 때 호출
InputSource resolveEntity (String publicId, String systemId)	✓ 외부 개체를 검색
void setDocumentLocator (Locator locator)	✓ Locator 객체를 설정
void skippedEntity(String name)	✓ 스킵하는 개체가 있는 경우 호출
void startDocument()	✓ 문서가 시작되는 경우 호출

DefaultHandler 클래스

- Method

메소드	설명
void startElement(String uri, String localName, String qName, Attributes attributes)	✓ 요소가 시작되는 경우 호출 ✓ localName은 XML 네임스페이스의 접두어를 제외한 태그 이름을 의미하고, qName은 XML 네임스페이스의 접두어를 포함한 태그 이름을 의미하며, attributes는 태그에 선언된 속성들을 의미
void startPrefixMapping (String prefix, String uri)	✓ 네임스페이스 매핑이 시작되는 경우 호출
void unparsedEntityDecl(String name, String publicId, String systemId, String notationName)	✓ 파싱되지 않은 개체 선언이 있는 경우 호출
void warning (SAXParseException e)	✓ 파싱 경고가 있을 때 호출

Attributes 인터페이스

- XML 속성 목록을 위한 Interface Type

메소드	내용
int getIndex(String qName)	XML 1.0 이름을 이용해서 속성의 인덱스 번호를 검색
int getIndex(String uri, String localName)	네임스페이스와 로컬 이름을 이용해서 속성의 인덱스 번호를 검색
int getLength()	목록에 있는 속성 개수를 검색
String getLocalName(int index)	인덱스를 이용해서 속성의 로컬 이름을 검색
String getQName(int index)	인덱스를 이용해서 속성의 XML 1.0 이름을 검색
String getType(int index)	인덱스를 이용해서 속성 타입을 검색
String getType(String qName)	XML1.0 이름을 이용해서 속성 타입을 검색
String getType(String uri, String localName)	네임스페이스와 로컬 이름을 이용해서 속성 타입을 검색
String getURI(int index)	인덱스를 이용해서 속성의 네임스페이스를 검색
String getValue(int index)	인덱스를 이용해서 속성값을 검색
String getValue(String qName)	XML1.0 이름을 이용해서 속성값을 검색
String getValue(String uri, String localName)	네임스페이스와 로컬 이름을 이용해서 속성값을 검색

Locator 인터페이스

- SAX Application에 위치 정보를 제공하는데 이용
- 대부분의 SAX Parser는 Locator 타입의 Object를 지원

메소드	내용
int getColumnNumber()	현재 이벤트가 끝나는 곳의 컬럼 번호를 반환
int getLineNumber()	현재 이벤트가 끝나는 곳의 줄 번호를 반환
String getPublicId()	현재 문서의 이벤트를 위한 PUBLIC 식별자를 반환
String getSystemId()	현재 문서의 이벤트를 위한 SYSTEM 식별자를 반환

ErrorHandler 인터페이스

- SAX Parser가 XML 문서를 Parsing하는 중 발견하지 못한 Error와 Warning를 Handling하기 위한 특별한 Interface
- SAX Parser가 Error를 만났을 때 SAX Process를 중지할 것인지, Error를 무시하고 계속 진행 할 것인지 결정
- Error 처리를 위한 3가지 Event Method 제공
  - fatalError() 메소드
  - error() 메소드
  - warning() 메소드

ErrorHandler 인터페이스

- fatalError() Method
  - 치명적인 Error
  - XML 문서가 정형식 문서가 아닐 때 발생하는 Error
    - XML Parser는 치명적인 Error가 발생하면 Parsing 작업을 종료
    - SAXParseException 예외를 발생시킴

메소드	내용
int getColumnNumber()	예외가 발생한 곳의 컬럼 번호를 반환
int getLineNumber()	에러가 발생한 곳의 줄 번호를 반환
String getSystemId()	예외가 발생한 개체의 시스템 ID를 반환
String getPublicId()	예외가 발생한 개체의 퍼블릭 ID를 반환

ErrorHandler 인터페이스

- error() Method
  - XML 문서가 유효성을 위반할 때 발생
  - DTD 문서와 유효성을 체크하는 Parser를 사용할 때 발생
  - 단, 유효성을 Check하지 않은 Parser에서도 XML시작부에서 version 속성이 1.0이 아닐 때도 발생
  - XML 문서를 처리하는 Program은 Error를 처리하는 Routine을 작성하는 경우에 유효성 Check에서 Error Message를 출력
  - Error 처리 Routine을 작성하지 않은 경우, 유효하지 않은 문서라도 Message를 출력하지 않음
  - Error 처리 Routine은 error() 메소드

ErrorHandler 인터페이스

- warning() 메소드
  - SAX Parser가 DTD를 처리하는 동안에 발생
  - Error는 아니지만 Parsing에 장애가 생기는 문제를 발견했을 때 호출되는 메소드
  - Warning이 발생하면, warning() 메소드가 호출

  - SAX에서 치명적인 Error, Error, Warning을 처리하려면 DefaultHandler의 메소드를 Override
  - DefaultHandler는 Error 처리를 위한 ErrorHandler 인터페이스를 구현하기 때문임

SAX Parser

- SAX Parsing의 장점
  - Memory를 차지하는 공간이 적음
    - XML Data를 모두 Memory에 적재하지 않고 Line(Start Tag – End Tag)단위로 Marking
  - 대용량 XML을 Parsing해도 속도가 빠름

```
<?xml version="1.0" encoding="utf-8" ?>
<car>
1라인 <sedan type =“대형” cc =“3000cc” price=“4000”> graduer </sedan>
2라인 <sedan type =“중형” cc =“2000cc” price=“3000”> sonata </sedan>
</car>
```

SAX Parser

- SAX Parsing의 단점
  - 지나간 Element를 읽어오기 위해서는 처음부터 다시 읽어야 함
  - 특정 Element에 대해서 어떻게 동작할지 사용자가 직접 정의

XML Parser

SAX 기반 Parser

종류	내용
JAXP (Java API for XML Processing)	Sun Microsystems에서 제공하는 Parser
	http://java.sun.com/xml/jaxp/index.html
XML4J(XML Parser for Java)	IBM에서 제공하는 Parser
	http://www.alphaworks.ibm.com/tech/xml4j
Xerces(Xerces Java Parser)	Apache에서 제공하는 Parser
	http://xml.apache.org
Oracle의 XML Parser	Oracle에서 제공하는 Parser
	http://technet.oracle.com/tech/xml/content.html
제임스 클락 XP	제임스 클락(James Clark)이 작성한 XML Parser
	http://www.jclark.com/xml/xp/
크림슨 파서	Apache 그룹에서 작성한 XML Parser. J2SDK1.4.0에 포함
	http://xml.apache.org/crimson/index.html

SAX Parser

- SAX API Programming 절차
  - 3개의 Package를 import 함 (자동으로 처리 됨)
    - org.xml.sax
    - org.xml.sax.helpers
    - javax.xml.parsers
  - Event Handler Class를 설계하여 Event Handler를 등록
  - Parser Object를 생성
  - Event Handler로 Event Object 생성
  - Parser Object에 Event Object를 적용시켜서 XML 문서 Parsing

SAX Parser

- SAXParser Parsing 시작

```
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();
XMLReader reader = parser.getXMLReader();
SaxHandler handler = new SaxHandler();
                        -> 파싱을 수행해주는 객체 생성
reader.setContentHandler(handler);
InputStream istream = new ByteArrayInputStream(
                                                xml데이터.getBytes("utf-8"));
reader.parse(new InputSource(istream));
```

SAX Parser

- Parsing을 시작하면 호출되는 Method
  - DefaultHandler Class로부터 상속 받아서 5개의 Method 재정의

```
public void startDocument () {}
public void endDocument() {}
public void startElement(String uri, String localName,
                                    String qName, Attributes atts) {}
public void endElement(String uri, String localName,
                                                  String qName) {}
public void characters(char[] chars, int start, int length) {}
```

SAX Parser

- Event Handler Class 설계
  - DefaultHandler Class로부터 상속받아 Event Handler Class를 설계

```
class MyHandler extends DefaultHandler {
    public void startDocument(){          //문서의 시작
    }
    public void startElement(){           //요소의 시작
    }
    public void endElement( … ){          //요소의 끝
    }
    public void endDocument( … ){         //문서의 종료
    }
}
```

SAX Parser

- Parser Object 생성
  - 추상 클래스 SAXParserFactory의 newSAXParser( ) Method를 사용하여 생성

```
SAXParserFactory factory =
                        SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();
```

SAX Parser

- Event Object 생성
  - 앞에서 설계한 MyHandler Class로 Object 생성

```
MyHandler handler = new MyHandler();
```

SAX Parser

- 속성의 이름과 값 알아내기
  - startElement() 메소드는 속성에 대한 정보를 Attributes를 통해 알아냄

Attributes의 getLength() 메소드는 속성의 개수

Attributes의 getQName() 메소드는 속성의 이름

Attributes의 getValue() 메소드는 속성의 값