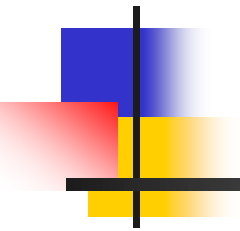


XML PullParser(이론)

배 희호 교수
경복대학교
소프트웨어융합과

XML PullParser

- DOM, SAX외에도 Android는 XML PullParser를 제공
  - Android에서 기본적으로 지원되는 XML Parser는 XML PullParser 임
- XML PullParser는 Android에서 XML Data를 효율적으로 처리하기 위한 기본 API
  - Android는 XML PullParser를 사용하여 SAX 및 DOM보다 XML File을 구문 분석하는 것이 좋음
  - 이는 Memory 효율성이 중요한 Mobile Application에서 XML 문서를 읽고 Parsing하는 데 사용됨
- DOM Parsing 방식보다 Memory 효율적이고, Event 기반 Parsing 방식(SAX와 유사)을 사용하여 DOM 방식과 SAX 방식에 비해 가볍고 간단하게 사용할 수 있음
  - 이는 특히 큰 XML File을 처리할 때 유용

XML PullParser

- XML PullParser는 Android SDK 내에 Interface로 정의되어 있으며, [XMLPULL V1 API]에서 정의한 규칙들과 API들을 포함하고 있음
  - XML PullParser가 Interface로 정의되어 있다는 것은, XML PullParser에 XML Parsing과 관련된 Method가 정의되어 있다는 것이고, XML PullParser의 참조를 통해 관련 Method를 사용할 수 있다는 의미임
  - XML Parsing을 위한 Method의 실질적인 구현은 XML PullParser에 포함되어 있지 않다는 것을 의미하기도 함
- XML PullParser는 SAX(Simple API for XML) 방식으로 동작
  - XML PullParser는 전체적인 동작 방식은 SAX와 유사하되 Event 발생 시마다 Handler를 호출하지 않고 대신 Loop를 돌며 다음 Event를 직접 조사하는 방식임

XML PullParser

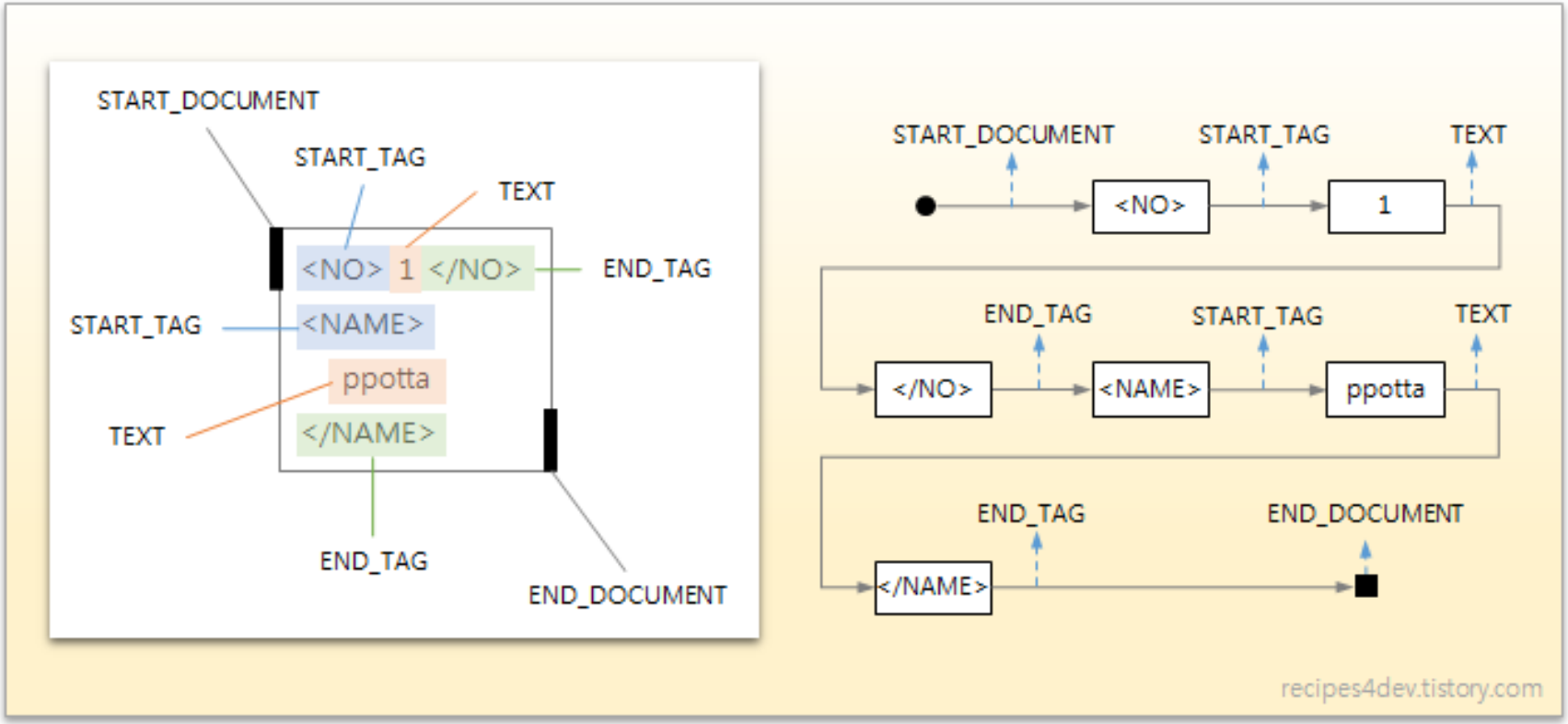
- XML PullParser 특징
  - Event 기반 Parsing
    - SAX Parser와 달리 Event Handler를 작성하지 않아도 되므로 간편함
    - XML 문서를 읽는 동안 Event(START_TAG, TEXT, END_TAG 등)가 발생하며 이를 처리함
    - 문서의 일부분만 작업해야 할 경우엔 SAX Parser보다 빠르고 단순하게 처리 가능
  - 경량 Parser
    - XML 전체를 Memory에 Load하지 않고 Streaming 방식으로 처리하므로 큰 XML File을 다룰 때 적합
  - 순차적 접근
    - XML File의 처음부터 끝까지 순차적으로 읽으므로, XML 문서의 특정 Data만 추출하거나 필요한 부분만 처리할 수 있어 Code를 간결하게 작성할 수 있음

XML PullParser

- XML PullParser 특징
  - Event 기반 API 제공
    - XML 문서를 순차적으로 읽으면서 각 Event에 따라 작업 수행
  - Event Type
    - START_DOCUMENT : 문서의 시작
    - END_DOCUMENT : 문서의 끝
    - START_TAG : Tag의 시작
    - END_TAG : Tag의 끝
    - TEXT : Tag 안의 Text
  - 직접적인 XML 속성 접근
    - Tag의 속성에 빠르게 접근 가능

XML PullParser

- XML PullParser 분석 진행 방법(속성)
  - 일반적으로 XML PullParser Interface에 정의된 4개의 상수(Event로 작동)를 사용



XML PullParser

- XML PullParser 분석 진행 방법(속성)
  - getEventType() Method로 현재 속성을 가져옴
  - next() Method로 다음 속성으로 이동

속성	의미
START_DOCUMENT	✓ XML File의 시작에 도달했을 때 반환
START_TAG	✓ Element의 시작 Tag를 만났을 때 반환
TEXT	✓ Element의 Text를 만났을 때 반환 (<tag>TEXT</tag>에서 TEXT) ✓ getText() 메소드를 사용하여 Text Contents를 검색할 수 있음 ✓ 속성을 얻는 방법은 getAttributeValue()를 호출
END_TAG	✓ Element의 종료 Tag를 만났을 때 반환(</tag>)
END_DOCUMENT	✓ XML File의 끝에 도달했을 때 반환

XML PullParser

- XML PullParser Code 작성 방법
  - XML 문서에서 Element 단위로 읽기 위한 준비
    - XML File을 읽기 위해 입력 Stream을 사용

```
InputStream stream = getResources().
                    openRawResource(R.raw.xml_test);
```

    - XML PullParser Interface의 XML Parsing 기능이 구현된 Class는 크게 2가지
      - XmlPullParserFactory.newPullParser() Method를 통해 생성 가능한 KXmlParser
      - Xml.newPullParser() Method를 통해 생성할 수 있는 ExpatPullParser
    - XML 문서를 Element 단위로 읽어 들이는 방법

```
XmlPullParser xmlParser = Xml.newPullParser();
```

XML PullParser

- XML PullParser Code 작성 방법
  - XML PullParser Class가 제공하는 setInput() Method를 사용해 Parser를 처리하기 위해 입력 Stream으로 설정

```
xmlParser.setInput(in,"utf-8");
```

메소드	설명
void setInput(Reader in)	reader를 Parser의 입력 Source로 지정하고 Parser 초기화
void setInput(InputStream inputStream, String inputEncoding)	inputStream을 Parser의 입력 Source로 지정

XML PullParser

- XML PullParser Code 작성 방법
  - Builder 생성 후 Parser Object 생성
    - newInstance()를 통해 Builder Factory 생성 후 newPullParser() Method를 통해 Parser Object를 얻음

```
XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
XmlPullParser parser = factory.newPullParser();
```

  - 분석하고 싶은 XML File을 설정
    - XmlPullParser Class의 setInput(InputStream, String)을 통해 분석할 XML과 Encoding 방식을 설정

```
parser.setInput(new InputStreamReader(inputStream, "UTF-8"));
```

XML PullParser

- XML PullParser Code 작성 방법
  - XML 문서에서 Element의 내용 얻어내기
    - XmlPullParser 클래스의 getEventType() Method를 사용해 Tag(Element)가 START_TAG인지, END_TAG인지 등의 정보를 얻어냄

```
int eventType = xmlParser.getEventType();
```

    - while 문을 사용해 문서의 끝까지 Data를 추출하는 구문을 사용

```
while (eventType !=  XmlPullParser.END_DOCUMENT) {

}
```

    - XmlPullParser 클래스의 next() Method는 다음의 Event Element 얻어냄

```
eventType = xmlParser.next();
```

XML PullParser

- XML PullParser Code 작성 방법
  - XML 문서에서 Element의 내용 얻어내기
    - TEXT의 값을 얻기 위해서는 getText()
    - 속성을 얻는 방법은 getAttributeValue()를 호출하면 얻을 수 있음

XML PullParser

- XML PullParser Code 작성 방법
  - 문서를 순차적으로 읽으면서 Event를 진행
    - 현재 Event를 분석하고 next() Method로 다음 Event로 이동

```
int eventType = parser.getEventType();
while (eventType != XmlPullParser.END_DOCUMENT) {
   switch (eventType) {
        case XmlPullParser.START_DOCUMENT:
            break;
        case XmlPullParser.START_TAG:
            break;
        case XmlPullParser.TEXT:
            break;
        case XmlPullParser.END_TAG:
            break;
      }
      eventType = parser.next();
}
```

XML PullParser

- PullParser 사용 시 주의사항
  - 정확한 XML 형식
    - XML 문서가 형식적으로 올바르지 않으면 Parsing 중 예외가 발생함
  - 대용량 XML 처리
    - 대량의 Data를 처리하는 경우 중간 Data를 정리하거나 File의 특정 부분만 처리하도록 설계해야 함
  - Event 처리 순서
    - START_TAG와 END_TAG Event를 정확히 처리하지 않으면 Data 누락이 발생할 수 있음

XML PullParser

- XML Parser 사용 시 주의 사항
  - Tag를 Parsing할 때 Tag의 이름을 가져올 때는 getName(), Text를 Parsing할 때 Text의 내용을 가져올 때는 getText()를 사용
  - 시작 Tag와 종료 Tag 사이에 빈 문자열만 있는 경우에도 TEXT Event 실행
    - Tag 사이에 빈 문자열 조차 존재하지 않는다면, TEXT Event는 아예 실행되지 않는다는 것을 주의해야 함
  - 시작 Tag와 종료 Tag의 바깥에 공백, 줄 바꿈에도 TEXT Event 실행
    - TEXT Event는 시작 Tag와 종료 Tag의 안쪽 문자열에만 해당되는 것이 아님
    - Tag 밖에 문자열이 존재하는 경우에도 TEXT Event는 실행

XML PullParser

- XML 문서 Parsing 방법

Parser	특징	장점	단점
DOM (Document Object Model) Parser	Element를 모두 Tree 구조로 Memory에 넣어두고 사용함	Memory에 Tree 구조로 정보가 들어있기 때문에 한번 Parsing해 두면 아무 때나 얻고 싶은 Element에 대한 정보를 얻을 수 있음	문서가 커지면 Memory를 지나치게 많이 소모함
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 Parsing함	Line 단위로 Parsing하기때문에 Parsing하는데 적은 Memory 소요	Parsing시 그냥 지나갔던 Element의 정보를 얻고 싶으면 다시 Parsing해야함
XML Pull Parser	SAX와 같이 Event 기반의 Parser이지만, SAX와 달리 문서에 대한 모든 Parsing을 하지 않고도 특정 부분까지의 Parsing 내용을 활용할 수 있음	원하는 부분을 Parsing할 수 있음	SAX의 단점을 가지며 SAX보다 약간 느림

XMLResourceParser

- XMLResourceParser는 Android의 XML Resource를 Parsing하기 위해 제공되는 Class
- 이 Class는 XmlPullParser의 Sub Class이며, XML File을 Parsing할 때 추가적으로 Android Resource System과의 통합을 지원함
  - XMLResourceParser는 XmlPullParser를 확장하여 Resource System에 맞춘 몇 가지 추가 기능을 제공
- 이는 Android Application에서 정의된 XML Resource File(예: Layout, Menu, Animation 등)을 효율적으로 처리하는 데 유용

XMLResourceParser

- XMLResourceParser의 주요 특징
  - XmlPullParser 상속
    - XMLResourceParser는 XmlPullParser와 동일한 Event 기반 Parsing Mechanism을 사용
  - Android Resource System 지원
    - Resource ID, 속성, Context와 같은 추가 정보를 처리할 수 있음
  - 효율적인 Resource Parsing
    - Android에서 XML Resource를 Parsing할 때 성능을 고려한 설계를 제공

XMLResourceParser

- 이를 통해 XML File을 읽고 Parsing하는 데 필요한 Event 기반 Parsing을 지원
- Event Type
  - XMLResourceParser는 XML PullParser와 유사하게 XML 문서를 순차적으로 읽어 들이며, 각각의 요소에 대한 Event를 생성
  - START_DOCUMENT: XML 문서의 시작
  - END_DOCUMENT: XML 문서의 끝
  - START_TAG: 시작 Tag
  - END_TAG: 끝 Tag
  - TEXT: Tag내의 Text

XMLResourceParser

- XMLResourceParser 사용 방법
  - XMLResourceParser는 보통 Android Framework 내부에서 사용되며, Resources Object를 통해 XML Resource를 Parsing하는 데 사용
  - XMLResourceParser 얻기
    - Resources.getXml(int id)
      - 주어진 XML Resource ID를 통해 XMLResourceParser Object를 반환
    - XMLResourceParser는 Layout(res/layout), Values(res/values), Menu(res/menu) 등 Android XML File을 다룰 때 사용

XMLResourceParser

- 주요 Method
  - int getEventType()
    - 현재 Event Type을 반환
  - int next()
    - 다음 Event로 이동하고 해당 Event Type을 반환
  - String getName()
    - 현재 Tag의 이름을 반환
  - String getAttributeValue(String namespace, String name)
    - 지정된 이름의 속성 값을 반환
  - String getPositionDescription()
    - 현재 XML Parsing 위치에 대한 설명을 반환
  - String nextText()
    - 현재 Tag 내의 Text를 반환

XMLResourceParser

- XMLResourceParser와 XmlPullParser 비교

특징	XmlPullParser	XMLResourceParser
사용 목적	일반적인 XML File Parsing	Android Resource XML File Parsing
Resource 통합	Resource System과 통합되지 않음	Android Resource System과 통합
속성 지원	속성 처리 시 추가 Logic 필요	속성 이름 및 값을 쉽게 처리 가능
추가 기능	없음	Resource ID, Context 처리 가능