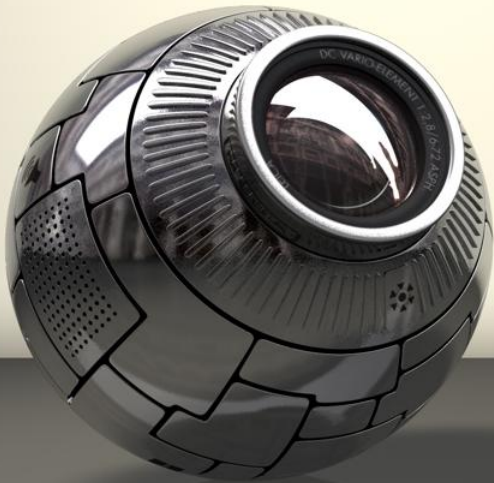
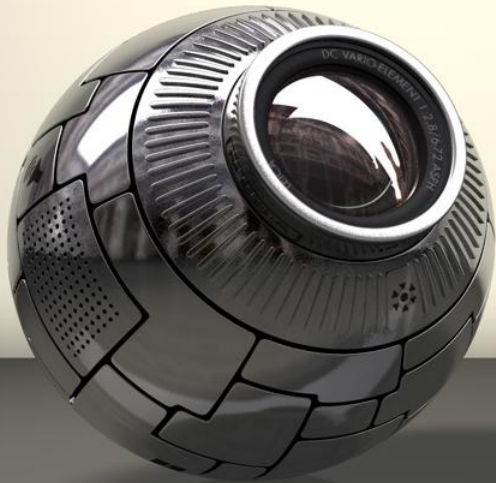



# 안드로이드 Dom 파서 활용

발표자 윤승환



# Index



 XML

 DOM

 DOM의 구조

 예제 및 참고자료



## Index

[XML](#)[DOM](#)[DOM 구조](#)[예제](#)[참고자료](#)

### XML

- XML 개요
- XML 문법
- XML 파서의 종류(DOM, SAX)

### DOM

- DOM 개요
- DOM 사용목적
- DOM 장점과 단점

### DOM의 구조

- 트리 구조도
- DOM인터페이스

### 예제 및 참고자료

- XML파일 DOM파싱
- DOM 인터페이스 추가내용
- 웹서버에 XML파일 DOM파싱



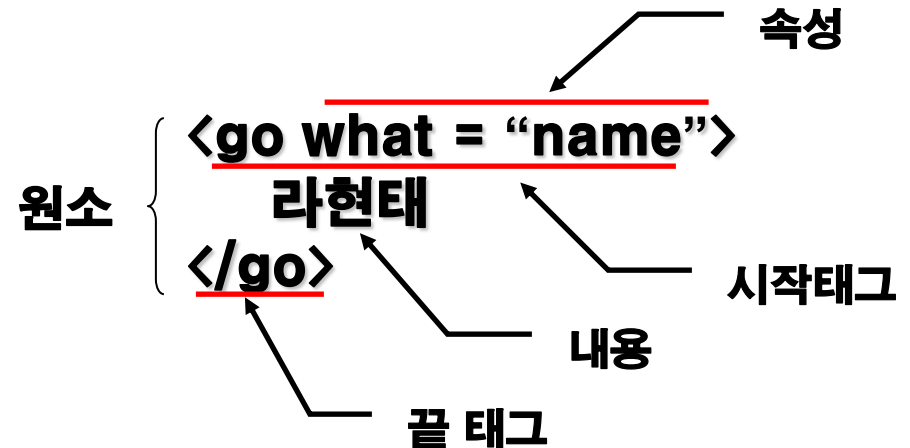
## XML의 개요

- eXtensible Markup Language(확장가능한 마크업 언어)의 약자
  - 확장가능: 기존에 없던 것을 새롭게 만듦
  - 마크업:문서의 논리적인 구조와 내용기술
- 데이터구조가 정의된 텍스트의 집합
- 문서의 구조를 임의의 DTD를 선언하고 Tag를 정의함으로써 문서를 표현가능
- XML tag들은 문서의 구조 혹은 데이터의 의미를 표현



## XML 문법

- 원소(Element)  
원소(Elements)는 XML 문서에서 핵심 구성 요소
- 태그(Tag)  
태그는 원소를 구성하기 위해 사용  
<원소이름>과 같은 시작 태그가 원소의 시작, </원소이름> 같은 마침 태그가 원소의 마지막을 표시
- 속성(Attribute)  
속성은 원소(element)에 대한 추가적인 정보를 제공  
속성은 원소의 시작 태그 안에 위치  
속성은 name/value 쌍으로 나타난다.





## XML 문법

- PCDATA(parsed character data)

PCDATA는 파서(해석기, parser)에 의해 해석이 될 텍스트  
XML 요소의 시작 태그와 마침 태그 사이에 위치한 텍스트

- CDATA(character data)

CDATA는 파서(해석기, parser)에 의해 해석이 되지 않을 텍스트  
<![CDATA[로 시작하여, ]]>로 끝나는 영역은 그대로 출력  
'<', '>', '&' 같은 특수문자를 사용할 때 CDATA로 감싸면 됨

- 엔터티(Entities)

XML 에서 엔터티는 변수와 비슷한 개념으로 생각할 수 있음  
엔터티는 문서 내에서 참조 될 수 있는 문자 집합의 단위  
HTML 문서에서 " "가 엔터티 참조의 한 예

[예제\)PCDATA와 CDATA의 차이점](#)



## XML 파서의 종류

### DOM (Document Object Model)

- 가장 대표적인 트리 기반 API
- 다양한 분야의 응용프로그램에 사용 가능
- XML 문서가 큰 경우 많은 시스템 자원을 사용

### SAX (Simple API for XML)

- 가장 대표적인 이벤트 기반 API
- DOM에 비해 간단한 구조
- 시스템 자원을 적게 사용
- 기능면에서 DOM 만큼 강력하지 못함

## DOM의 개요

### DOM(Document Object Model) 정의

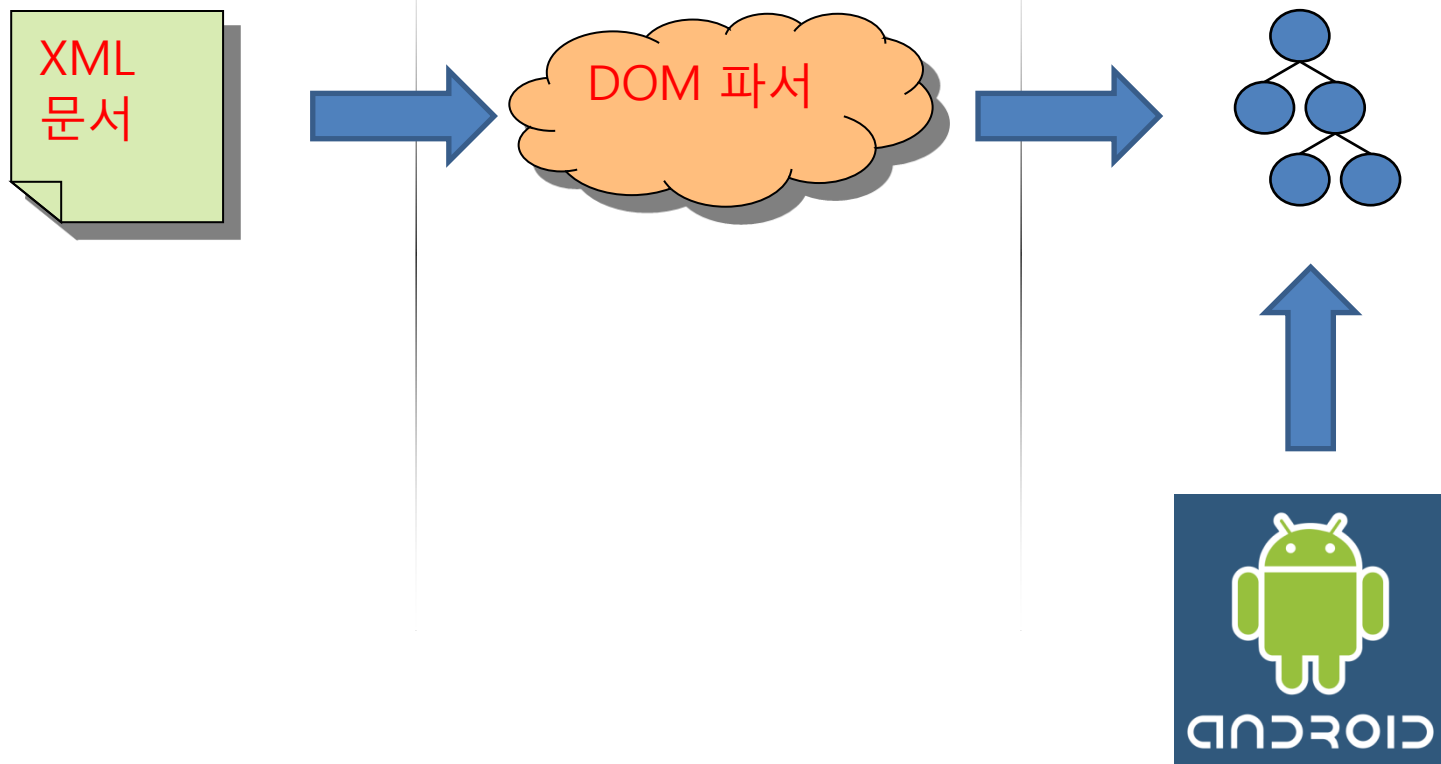
- 객체지향모델로 구조화된 문서 표현
- 플랫폼/ 언어 중립적 사용
- W3C에 의해 표준화

### DOM의 특징

- 객체 모델 기반 API
- 노드(node) 단위로 데이터 처리
- 문서의 구조와 내용을 객체로 이용
- 메모리 적재 방식



## DOM의 처리 과정



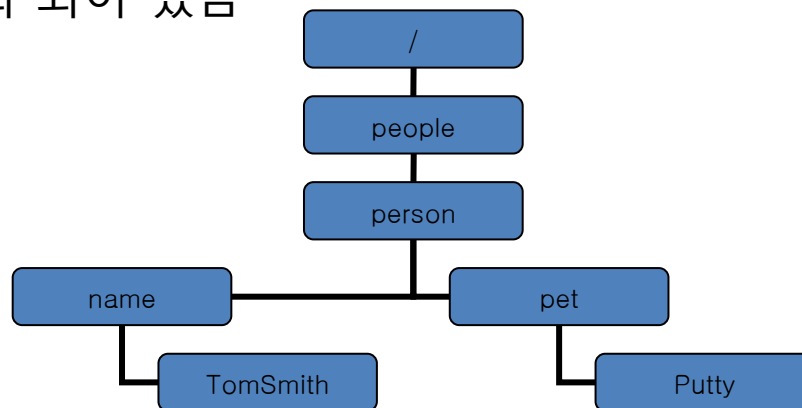
## DOM의 특징

DOM은 문서를 객체로 표현하고 처리할 수 있는 방법을 제공

- 문서에서 표현되는 데이터를 노드로 표현
- 새로운 내용을 삽입, 변경, 삭제 작업 수월
- XML문서의 element, attribute, entity, comment, processing instruction, text 등이 DOM에서는 객체로 정의 되어 있음

```
<people>  
  <person>  
    <name>TomSmith</name>  
    <pet>Putty</pet>  
  </person>  
</people>
```

  
DOM  
파싱



**목적 : 어떠한 환경이나 애플리케이션에서도 사용 할 수 있는  
표준 프로그래밍 인터페이스를 제공.**

## DOM의 장점, 단점

### DOM 장점

- XML 문서 조작을 위한 인터페이스
- XML 모델 처리시 범 프로그램적으로 사용 될 수 있는 공통적인 수단 제공
- 공통 인터페이스 공유를 통해 프로그래머 생산성 향상

### DOM 단점

- 큰 규모의 XML 문서를 다루는 데는 비효율적
- 대안으로 SAX (Simple API for XML)



## DOM 트리 구조도

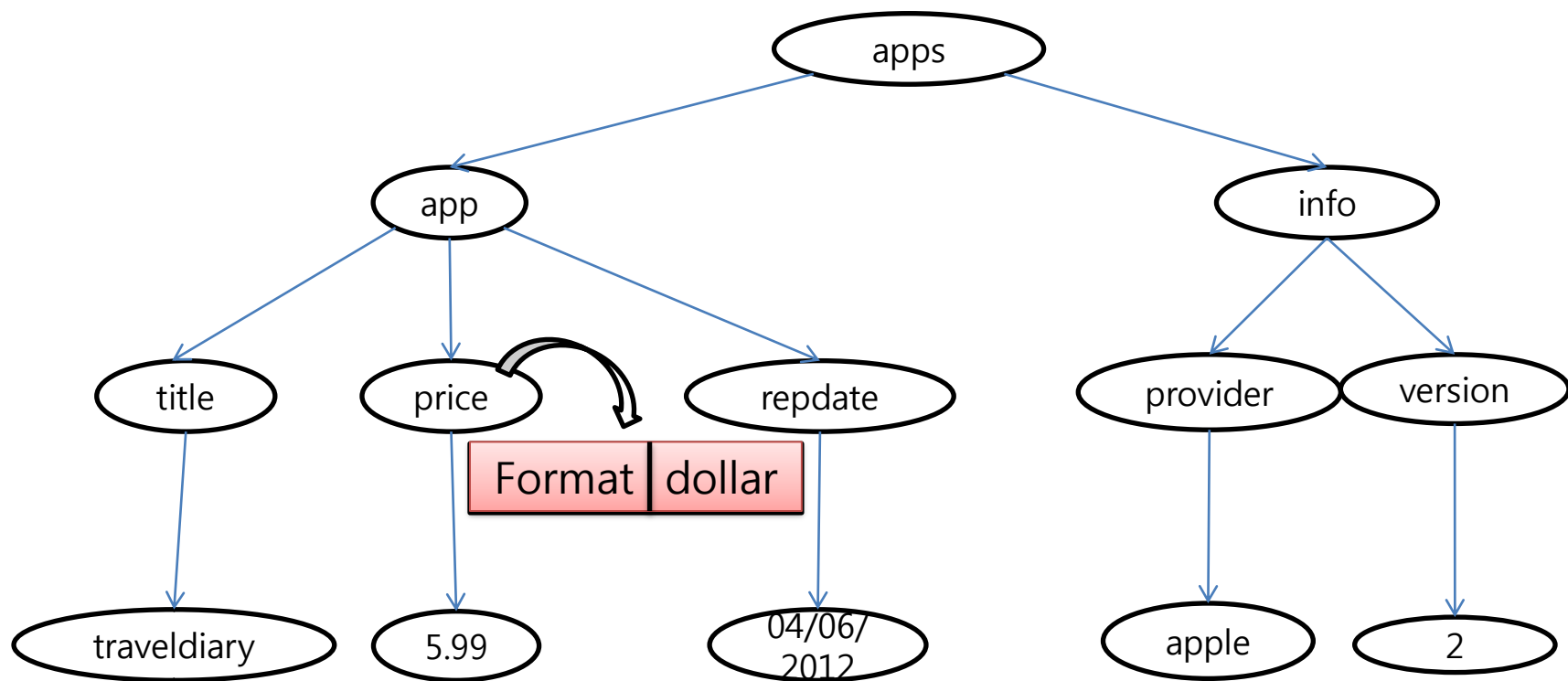
예제 XML파일

```
<?xml version="1.0" encoding="utf-8" ?>
<apps>
  <app>
    <title>traveldiary</title>
    <price format="dollar">5.99</price>
    <reupdate>04/06/2012</reupdate>
  </app>
  <info>
    <provider>apple</provider>
    <version>2</version>
  </info>
</apps>
```



## DOM 트리 구조도

예제 XML데이터를 DOM구조로 읽어왔을때 구조





## DOM 인터페이스

### DOM파서 속성(Attribute)

childNodes : 현재 요소의 자식을 배열로 표현

firstChild : 현재 요소의 첫번째 자식

lastChild : 현재 요소의 마지막 자식

previousSibling : 현재 요소의 바로 이전요소를 의미

nextSibling : 현재 요소의 바로 다음의 요소를 의미

nodeValue : 해당 요소의 값을 읽고 쓸 수 있는 속성정의(=data)

parentNode : 해당 요소의 부모노드



## DOM 인터페이스

### DOM파서 메소드 (문서)

`getElementById(id)`

-도큐먼트에서 특정한 id 속성값을 가지고 있는 요소를 리턴

`getElementsByTagName(name)`

-특정한 태그 이름을 가지고 있는 자식 요소로 구성된 배열을 리턴

`hasChildNodes()`

-해당 요소가 자식 요소를 포함유무 판단 (Boolean 값 리턴)

`getAttribute(name)`

-특정한 name 에 해당하는 요소의 속성값을 리턴

`document.createElement(tagName)`

-tagName 으로된 엘리먼트를 생성

`document.createTextNode(text)`

-정적 텍스트를 담고 있는 노드를 생성



## DOM 인터페이스

### DOM파서 메소드 (element)

`<element>.appendChild(childNode)`

-특정 노드를 현재 엘리먼트의 자식 노드 에 추가

`<element>.getAttribute(name)`

-속성명이 name 인 속성값을 반환

`<element>.setAttribute(name, value)`

-속성값 value 를 속성명이 name 인 곳에 저장

`<element>.insertBefore(newNode, tartgetNode)`

-newNode 를 tartgetNode 전에 삽입





## DOM 인터페이스

### DOM파서 메소드 (element)

`<element>.removeAttribute(name)`

-엘리먼트에서 name 속성을 제거

`<element>.removeChild(childNode)`

-자식 엘리먼트를 제거

`<element>.replaceChild(newNode, oldNode)`

-oldNode 를 newNode 로 치환

`<element>.hasChildNodes()`

-자식 노드가 존재하는지 여부를 판단(리턴 Boolean)



## 예 제

[XML](#)

[DOM](#)

[DOM 구조](#)

[예 제](#)

[참고자료](#)

- 예제
  - Test.xml이라는 xml파일을 domparsing하여 화면에 필요한 구성을 출력한다.
  - 파일 구성
    - **DOMEx.java**
    - **test.xml**



## 예제

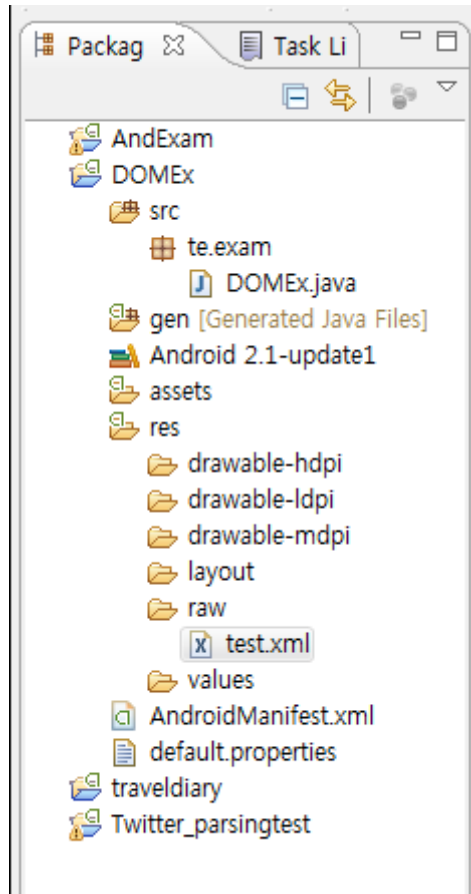
[XML](#)

[DOM](#)

[DOM 구조](#)

[예제](#)

[참고자료](#)

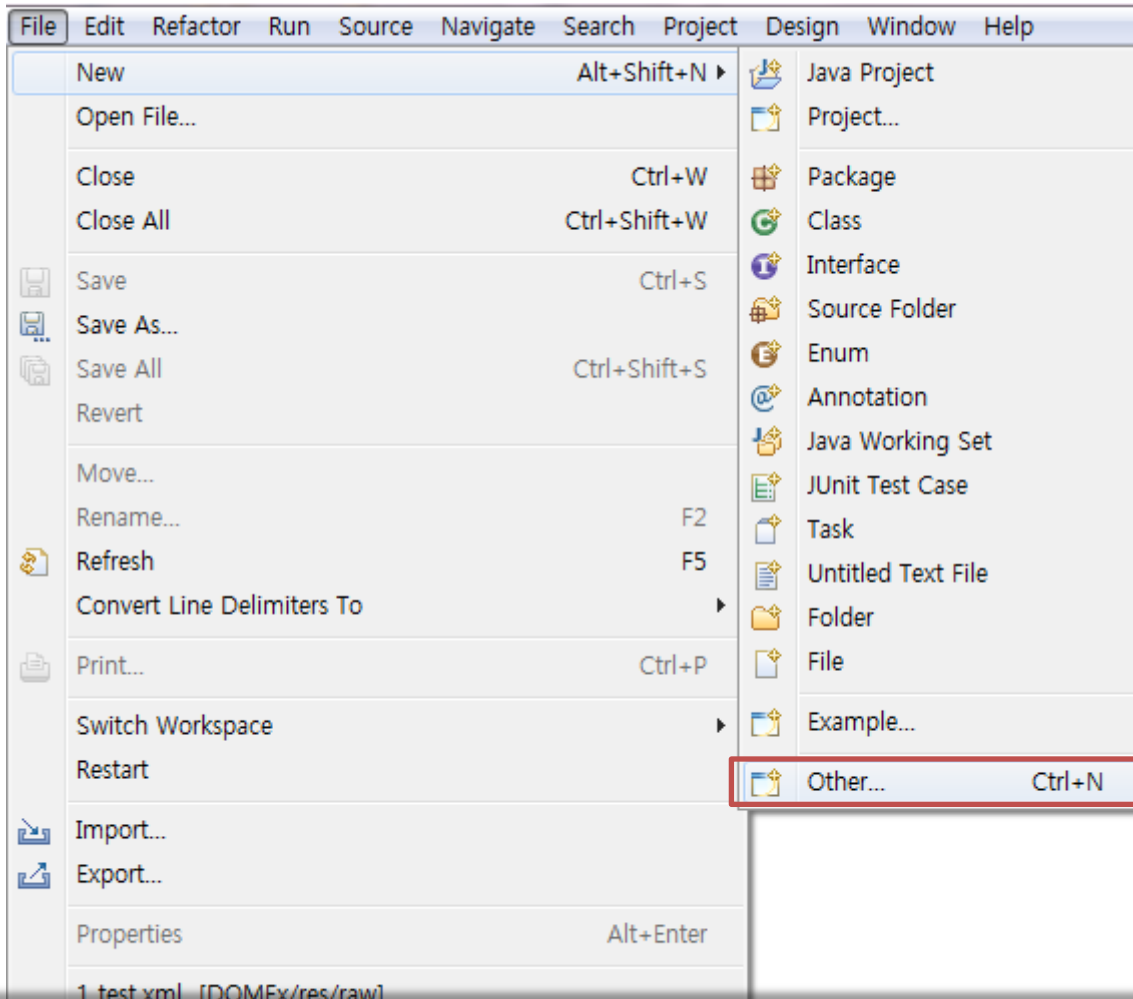


test.xml파일 생성하여 XML파일을 DOM파싱

1. DOMEx 이라는 프로젝트를 생성
2. DOMEx/res/raw폴더에 test.xml파일을 생성
3. DOMEx/src/te.exam/DOMEX.java파일에 DOM파싱소스추가
4. AVD manager를 통해 출력이 제대로 되는지 확인



## 1.DOMEx 이라는 프로젝트를 생성합니다.





예제

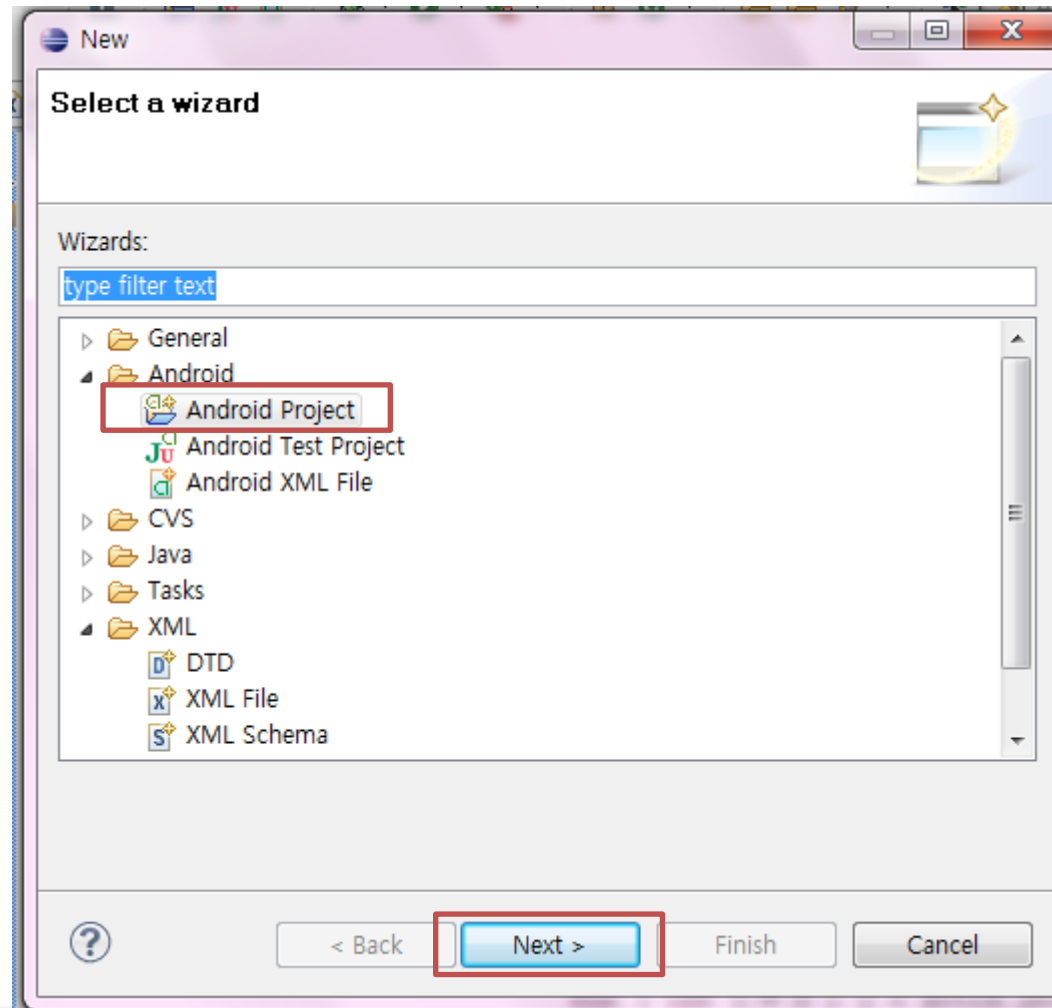
[XML](#)

[DOM](#)

[DOM 구조](#)

[예제](#)

[참고자료](#)





Project name:

Contents

☒ Create new project in workspace  
☐ Create project from existing source  
☒ Use default location

Location:

☐ Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	API ...
<input type="checkbox"/> Android 1.5	Android Open Source Project	1.5	3
<input type="checkbox"/> Google APIs	Google Inc.	1.5	3
<input type="checkbox"/> Android 1.6	Android Open Source Project	1.6	4
<input type="checkbox"/> Google APIs	Google Inc.	1.6	4
<input type="checkbox"/> Android 2.1-upd...	Android Open Source Project	2.1-upd...	7
<input type="checkbox"/> Google APIs	Google Inc.	2.1-upd...	7
<input type="checkbox"/> Android 2.2	Android Open Source Project	2.2	8
<input type="checkbox"/> Google APIs	Google Inc.	2.2	8



## 예제

[XML](#)[DOM](#)[DOM 구조](#)[예제](#)[참고자료](#)

<input type="checkbox"/>	Google APIs	Google Inc.	2.1-upd...	7
<input type="checkbox"/>	Android 2.2	Android Open Source Project	2.2	8
<input checked="" type="checkbox"/>	Google APIs	Google Inc.	2.2	8
<input type="checkbox"/>	GALAXY Tab Ad...	Samsung Electronics Co., Ltd.	2.2	8
<input type="checkbox"/>	Android 2.3.1	Android Open Source Project	2.3.1	9
<input type="checkbox"/>	Google APIs	Google Inc.	2.3.1	9
<input type="checkbox"/>	Android 2.3.3	Android Open Source Project	2.3.3	10
<input type="checkbox"/>	Google APIs	Google Inc.	2.3.3	10
<input type="checkbox"/>	Android 3.0	Android Open Source Project	3.0	11
<input type="checkbox"/>	Google APIs	Google Inc.	3.0	11

Android + Google APIs

Properties

Application name: domex

Package name: ex.doma

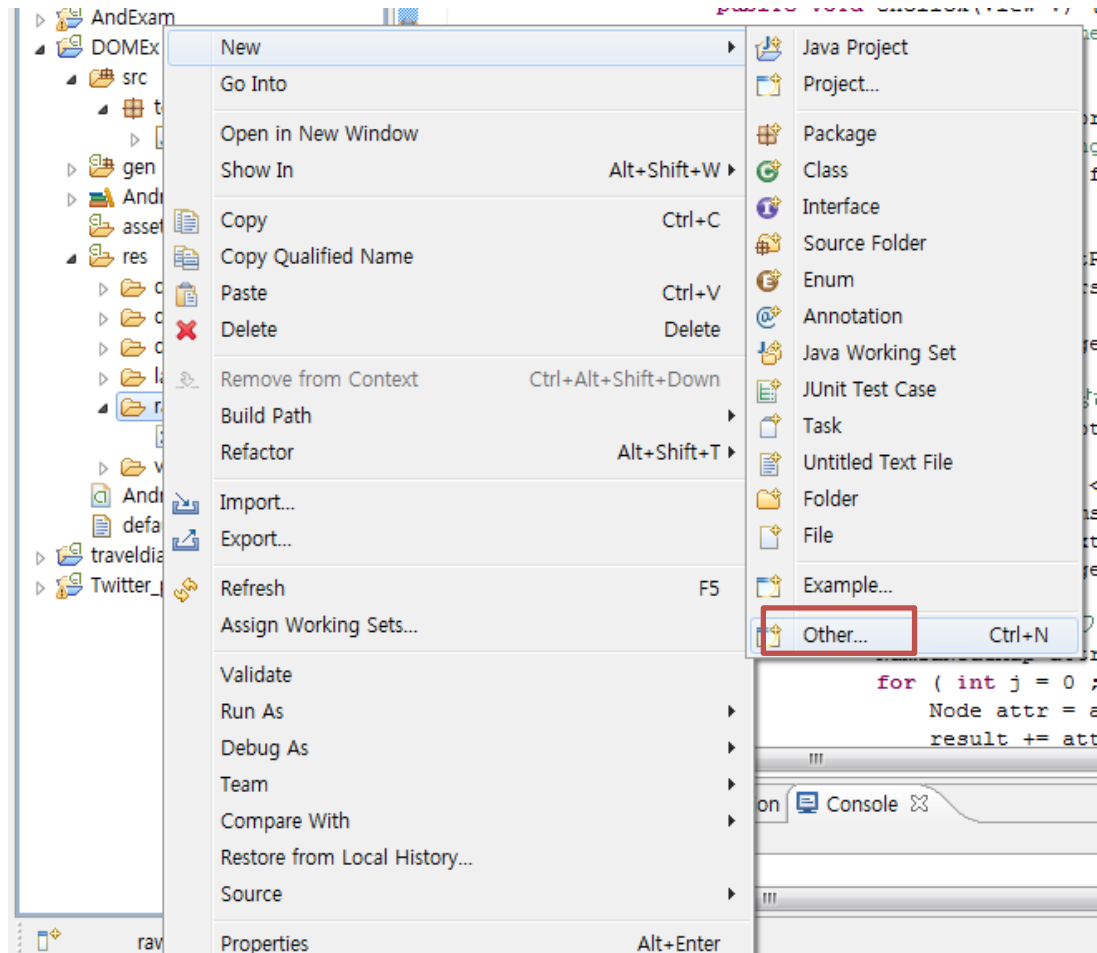
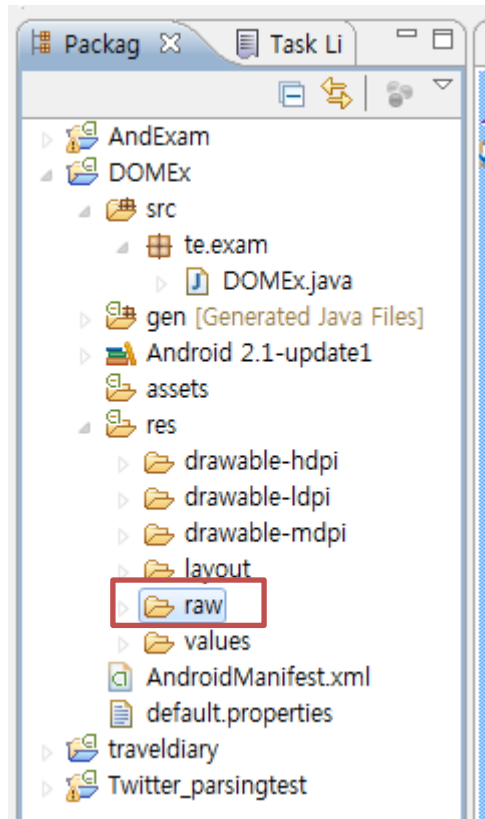
☒ Create Activity: dom

Min SDK Version:

< Back Next > Finish Cancel



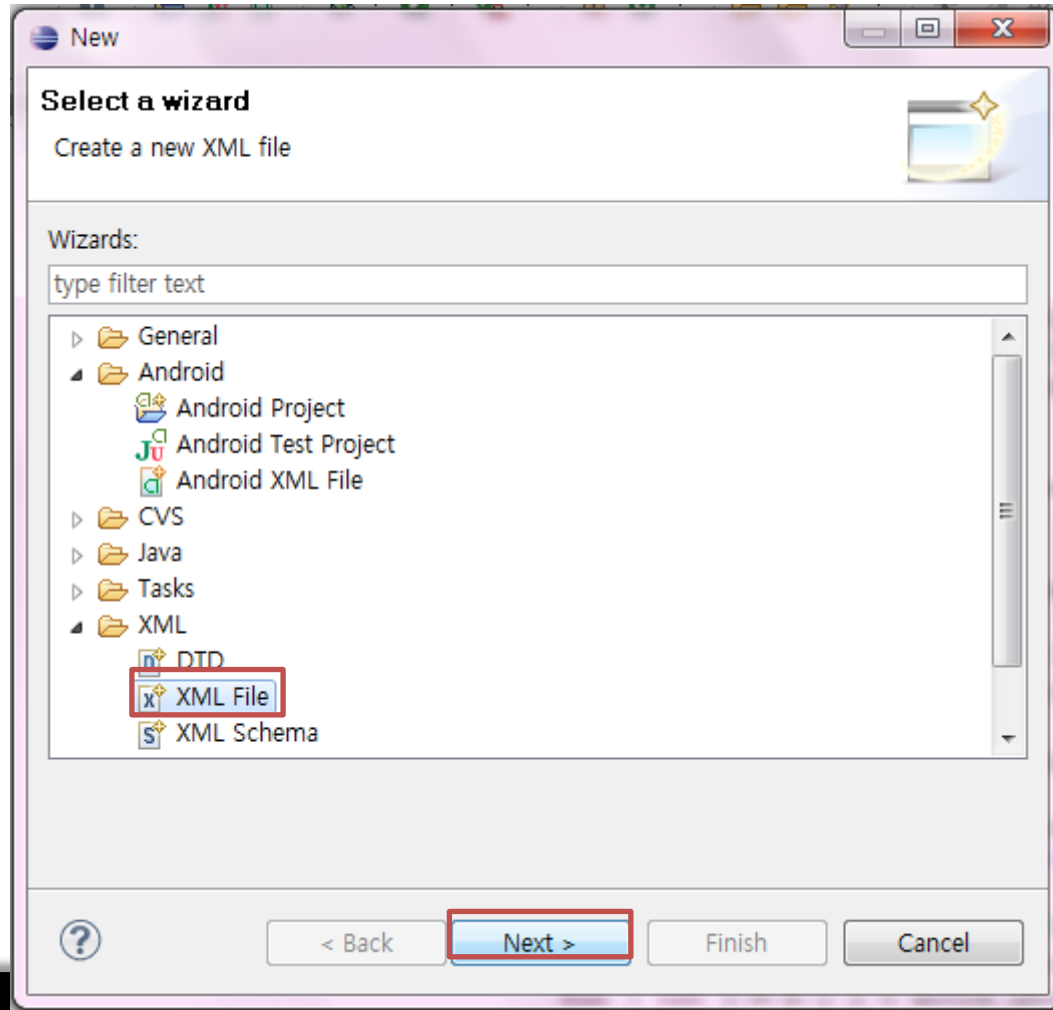
2. DOMEx/res/raw폴더에 test.xml파일을 생성합니다.







2. DOMEx/res/raw폴더에 test.xml파일을 생성합니다.





## 예제

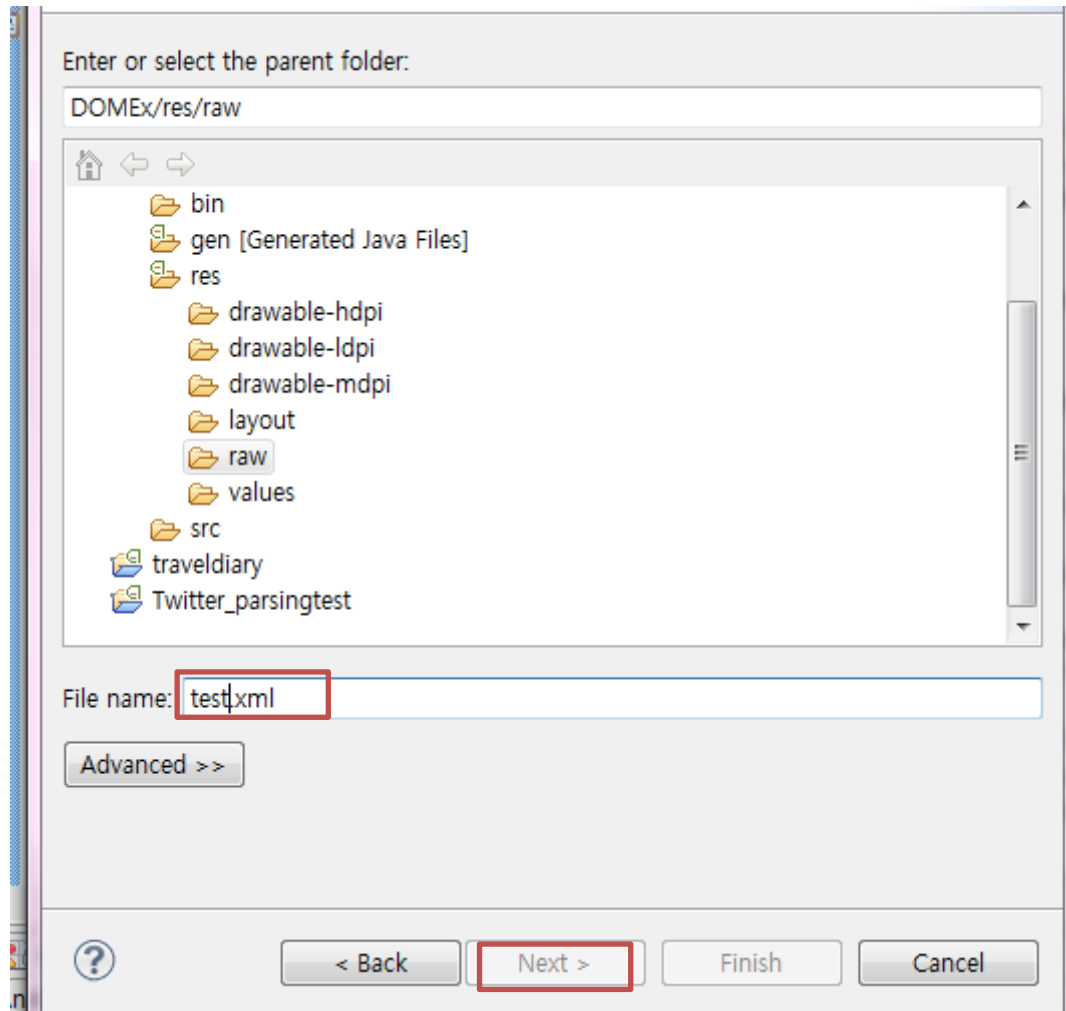
[XML](#)

[DOM](#)

[DOM 구조](#)

[예제](#)

[참고자료](#)





## 예제

[XML](#)[DOM](#)[DOM 구조](#)[예제](#)[참고자료](#)

IDE Screenshot showing the XML file 'test.xml' and the Java file 'DOMEx.java'.

**test.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<order>
  <item studentid="20081579" name="문승환">컴퓨터공학과</item>
  <item studentid="20091921" name="라현태">컴퓨터공학과</item>
  <item studentid="20081611" name="제갈민재">컴퓨터공학과</item>
  <item studentid="20091511" name="배동혁">컴퓨터공학과</item>
</order>
```

**DOMEx.java**



## 예 제

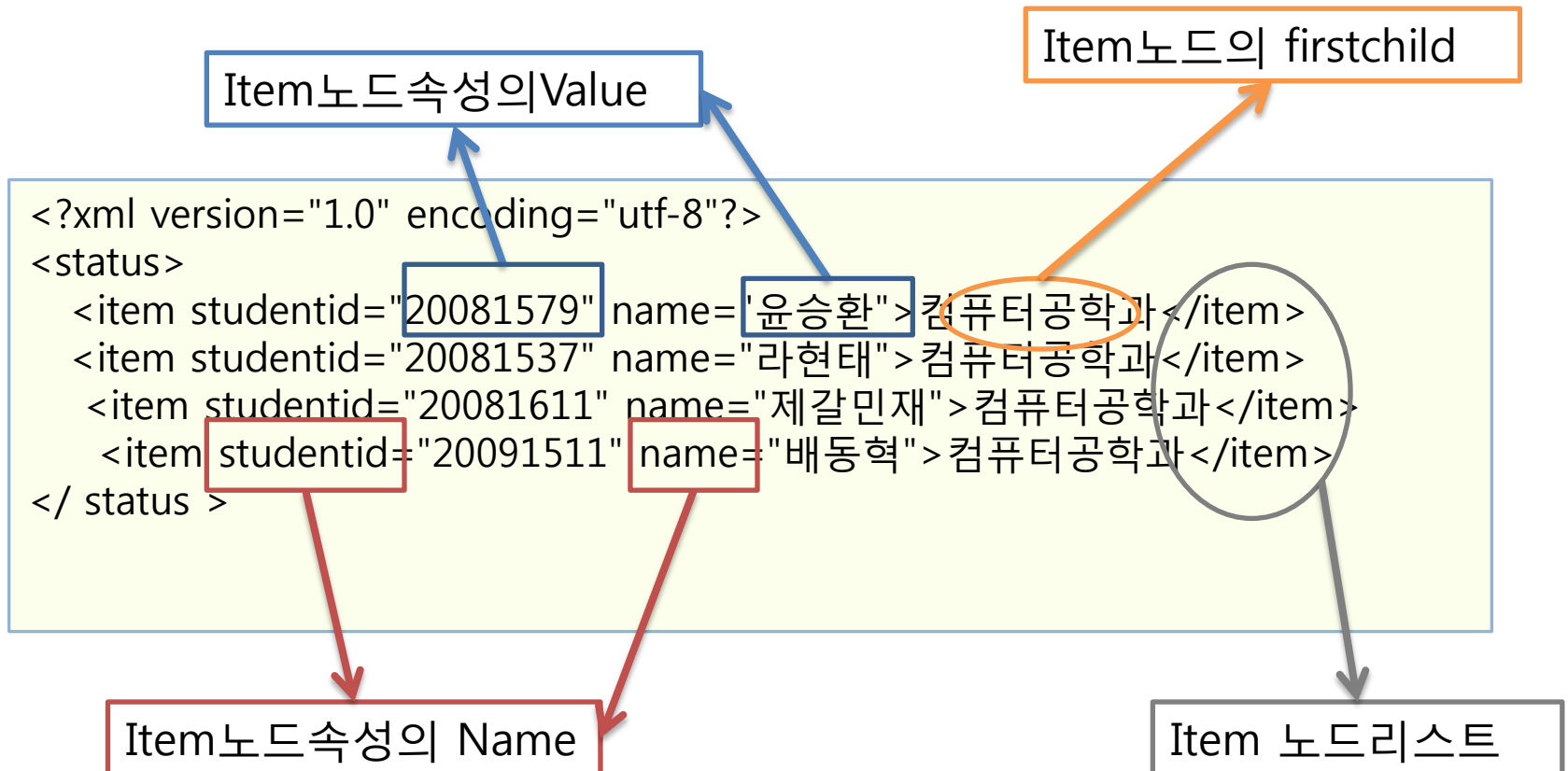
XML

DOM

DOM 구조

예 제

참고자료





### 3. DOMEx/src/te.exam/DOMEx.java파일에 DOM파싱소스를 추가합니다.

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes:

- AndExam
- DOMEx
  - src
    - te.exam
      - DOMEx.java** (highlighted with a red box)
  - gen [Generated Java Files]
  - Android 2.1-update1
  - assets
  - res
    - drawable-hdpi
    - drawable-ldpi
    - drawable-mdpi
    - layout
    - raw
  - test.xml

The code editor shows the following Java code in DOMEx.java:

```
try {
    // DOM 파서 생성
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    //factory.setIgnoringElementContentWhitespace(true);
    DocumentBuilder b = factory.newDocumentBuilder();

    // 리소스 파일 가져오기
    InputStream is = getResources().openRawResource(R.raw.test);
    Document doc = b.parse(is);

    Element root = doc.getDocumentElement();

    // 동일한 태그이름에 해당하는 Node 가져오기
    NodeList items = root.getElementsByTagName("item");
    String result = "";
    for ( int i = 0 ; i < items.getLength() ; i++ ) {
        Node item = items.item(i);
        Text data = (Text) item.getFirstChild();
        result += data.getData() + "\n";

        // 각 item 노드가 가지고 있는 속성 가져오기
        NamedNodeMap attrs = item.getAttributes();
        for ( int j = 0 ; j < attrs.getLength(); j++ ) {
            Node attr = attrs.item(j);
            result += attr.getNodeName();
            result += "=";
            result += attr.getNodeValue() + "\n";
        }
        result += "\n";
    }
    et.setText(result);
} catch (Exception e) {
    Log.d("TAG", e+"dom fail");
}
```



```
import java.io.InputStream;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.Text;
```

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
```



Dom 파서 생성

```
try {  
    DocumentBuilderFactory factory =  
        DocumentBuilderFactory.newInstance();  
    DocumentBuilder b = factory.newDocumentBuilder();
```

```
    InputStream is = getResources().openRawResource(R.raw.test);  
    Document doc = b.parse(is);
```

```
    Element root = doc.getDocumentElement();
```

리소스 파일 가져오기



```
NodeList items = root.getElementsByTagName("item");
```

```
String result = "";
```

```
for ( int i = 0 ; i < items.getLength() ; i++ ) {
```

```
Node item = items.item(i);
```

```
Text data = (Text) item.getFirstChild();
```

```
result += data.getData() + "\n";
```

동일한 태그이름에  
해당하는 Node 가져오기

```
<?xml version="1.0" encoding="utf-8"?>
<order>
  <item studentid="20081579" name="윤승환">컴퓨터공학과</item>
  <item studentid="20091921" name="라현태">컴퓨터공학과</item>
  <item studentid="20081611" name="제갈민재">컴퓨터공학과</item>
  <item studentid="20091511" name="배동혁">컴퓨터공학과</item>
</order>
```





```
NamedNodeMap attrs = item.getAttributes();
```

```
for ( int j = 0 ; j < attrs.getLength(); j++ ) {  
    Node attr = attrs.item(j);  
    result += attr.getNodeName();  
    result += "=";  
    result += attr.getNodeValue() + "\n";  
}
```

각 item 노드가 가지고 있는 속성 가져오기

```
result += "\n";  
}
```

```
<?xml version="1.0" encoding="utf-8"?>  
<order>  
  <item studentid="20081579" name="문승환">컴퓨터공학과</item>  
  <item studentid="20091921" name="관현태">컴퓨터공학과</item>  
  <item studentid="20081611" name="재갈민재">컴퓨터공학과</item>  
  <item studentid="2009151" name="배동호">컴퓨터공학과</item>  
</order>
```



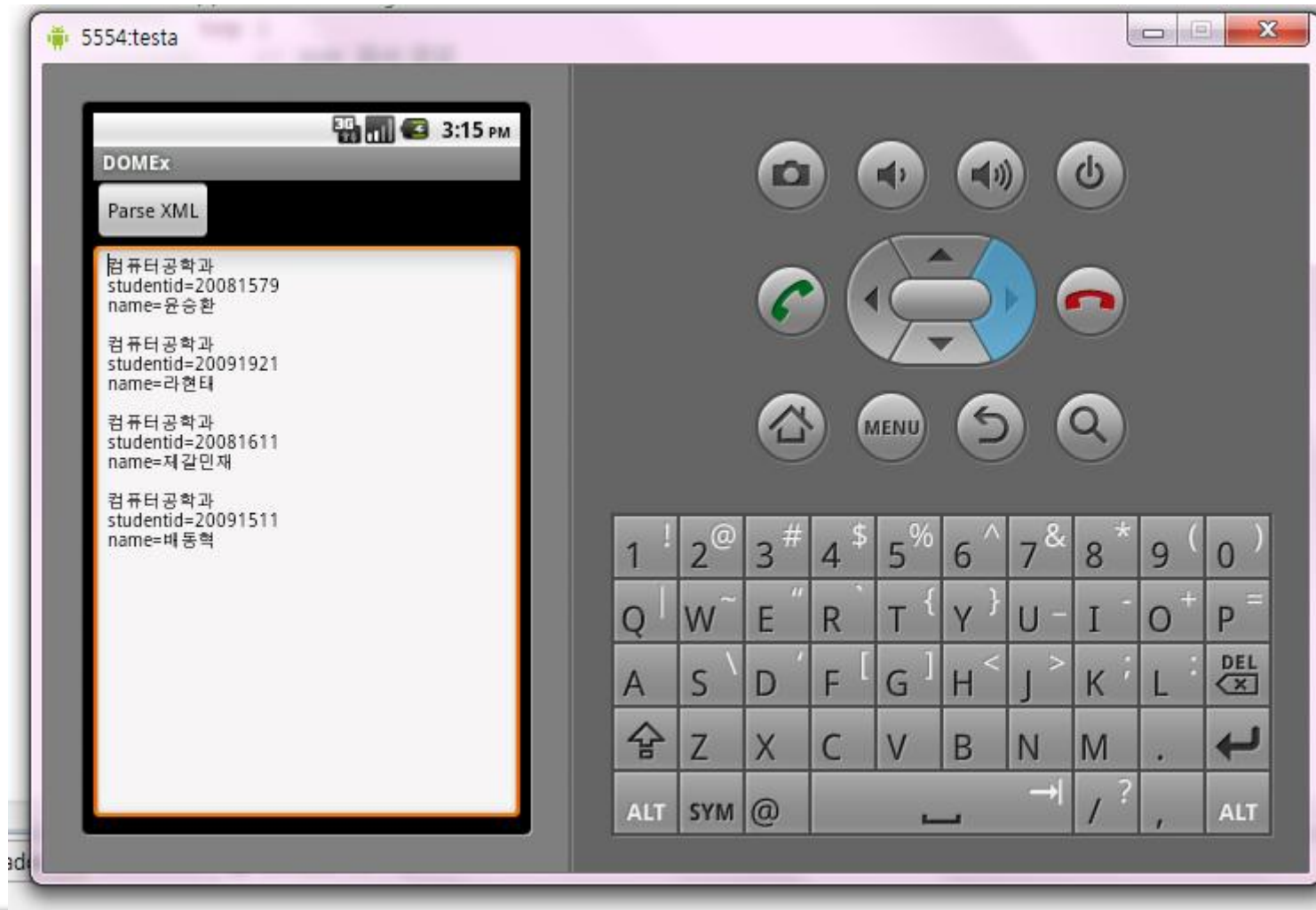
```
et.setText(result);
```

```
} catch (Exception e) {  
    Log.d("TAG", e + "dom fail");  
}  
  
}  
});  
}  
}
```

← 예외처리



## 예제

[XML](#)[DOM](#)[DOM 구조](#)[예제](#)[참고자료](#)





1)DOM 인터페이스  
프로퍼티, 함수 설명

2)트위터 xml으로  
dom파싱하여 출력하  
기



## Property (프로퍼티)

- 객체의 정보에 접근하는 메소드
- 대괄호 [ ] 를 사용하지 않고 객체의 정보에 접근
- 요소의 정보, 속성에 대한 구체적인 값, 속성값

참고 <http://cafe.naver.com/appforge/1159>



## DOM 인터페이스 - Node

DOM 객체가 만들어지는 기본 인터페이스

프로퍼티	타입	설명
nodeName	DOMString(Read)	노드의 이름, 타입에 따라 다른 이름 반환
nodeValue	DOMString	노드의 값, 타입에 따라 다른 값을 반환
nodeType	unsigned short (Read)	노드의 타입
parentNode	Node(Read)	이 노드의 부모 노드
childNodes	NodeList(Read)	이 노드의 자식 노드를 포함하는 NodeList 자식노드가 없으면 빈 NodeList 반환
firstChild	Node(Read)	이 노드의 첫 번째 자식 노드
lastChild	Node(Read)	이 노드의 마지막 자식 노드
previousSibling	Node(Read)	이 노드의 바로 앞 노드
nextSibling	Node(Read)	이 노드의 바로 뒤 노드



## DOM 인터페이스 - Node

프로퍼티	타입	설명
attributes	NamedNodeMap (Read)	이 노드의 어트리뷰트를 포함하는 NamedNodeMap
ownerDocument	DOMString(Read)	이 노드가 속한 도큐먼트
namespaceURI	DOMString(Read)	이 노드의 네임스페이스 URI
prefix	DOMString	네임스페이스 접두어
localName	DOMString(Read)	노드의 QName 부분 이름을 반환

프로퍼티 참고





## DOM 인터페이스 - Node

함수	설명
Node insertBefore(Node newChild, Node refChild)	존재하는 refChild 앞에 newChild를 삽입 refChild가 NULL이면 리스트 끝에 삽입
Node replaceChild(Node newChild, Node oldChild)	oldChild를 newChild로 대체 oldChild를 반환
Node removeChild(Node oldChild)	리스트에서 oldChild를 제거하고 그것을 반환
Node appendChild(Node newChild)	리스트 끝에 newChild를 추가하고 그것을 반환
boolean hasChildNodes()	노드에 자식이 있으면 true, 없으면 false
Node cloneNode(boolean deep)	이 노드를 반환, 파라미터가 true이면 노드의 하위 트리까지 복사하고 그렇지 않으면 그 노드만 복사
void normalize()	복수의 근접한 Text 자식 노드가 있으면 그것을 합침
boolean supports(DOMString feature, DOMString version)	DOM 구현체가 넘겨받은 특징을 지원하는지 여부 검사, 지원하면 true, 아니면 false



## DOM 인터페이스 – Document

전체 XML 문서를 표현  
Node 인터페이스 상속

프로퍼티	타입	설명
DocType	DocumentType(Read)	이 문서와 연결된 문서 형식을 표현하는 Document Type 객체 반환 정의된 문서 형식이 없다면 NULL반환
Implementation	DOMImplementation (Read)	이 문서에 사용된 DOMImplementation 객체
DocumentElement	Element(Read)	이 문서의 루트 엘리먼트



## DOM 인터페이스 – Document

함수	설명
Element createElement(DOMString tagName)	지정된 이름으로 엘리먼트를 생성
DocumentFragment createDocumentFragment()	빈 DocumentFragment 객체를 생성
Text create TextNode(DOMString data)	data에 텍스트를 포함하는 Text 노드를 생성
Comment createComment(DOMString data)	data에 텍스트를 포함하는 Comment 노드를 생성
CDATASection createCDATASection(DOMString data)	data에 텍스트를 포함하는 CDATASection 노드를 생성
ProcessingInstruction createProcessingInstruction(DOMString target, DOMString data)	지정된 target과 data를 가지는 ProcessingInstruction 노드를 생성
Attr createAttribute(DOMString name)	지정된 name을 가지는 어트리뷰트를 생성
EntityReference createEntityReference(DOMString name)	지정된 name으로 엔티티 참조 생성



## DOM 인터페이스 – Document

함수	설명
NodeList getElementsByTagName(DOMStringtagname)	문서에서 이 tagname을 가진 모든 엘리먼트를 NodeList로 반환, 엘리먼트 문서에 있는 순서로 반환
Node importNode(Node importedNode, boolean deep)	다른 문서에서 importedNode를 가져옴. 이전 문서에서 원래 노드 지원하지 않고 복제
Element createElementNS(DOMStringnamespaceURI, DOMStringqualifiedName)	지정된 네임스페이스와 QName로 엘리먼트를 생성
Attr createAttributeNS(DOMStringnamespaceURI, DOMStringqualifiedName)	지정된 네임스페이스와 QName로 어트리뷰트를 생성
NodeList getElementsByTagNameNS(DOMString namespaceURI, DOMString localName)	지정된 부분 이름을 가지고 namespaceURI에 의해서 정해지는 네임스페이스 안에 있는 모든 엘리먼트 NodeList로 반환
Element getElementById(DOMStringelementID)	elementID를 가진 엘리먼트를 반환



## DOM 인터페이스 – DOMImplementation

특정 문서에 종속적이지 않은 어떤 문서라도 사용할 수 있는 함수 제공

함수	설명
<code>boolean hasFeature(DOMString feature, DOMString version)</code>	이 DOM 구현체가 요청 받은 feature를 지원하는지 여부를 반환 version은 테스트할 이 특징의 버전
<code>DocumentType createDocumentType(DOMString qualifiedName, DOMString publicID, DOMString systemID, DOMString internalSubset)</code>	지정된 어트리뷰트를 가지고 Document Type 객체를 만듦
<code>Document createDocument(DOMString namespaceURI, DOMString qualifiedName, DocumentType doctype)</code>	지정된 qualifiedName을 가지고 Document 객체를 만듦



## DOM 인터페이스 – NodeList

인덱스로 접근이 가능한 노드 그룹을 포함

프로퍼티	타입	설명
Length	unsigned long(Read)	리스트에 포함된 노드의 수. 유효 숫자는 0 부터 length-1까지

함수	설명
boolean hasFeature(DOMString feature, DOMString version)	이 DOM 구현체가 요청 받은 feature를 지원하는지 여부를 반환 version은 테스트할 이 특징의 버전
DocumentType createDocumentType(DOMString qualifiedName, DOMString publicID, DOMString systemID, DOMString internalSubset)	지정된 어트리뷰트를 가지고 Document Type 객체를 만듦
Document createDocument(DOMString namespaceURI, DOMString qualifiedName, DocumentType doctype)	지정된 qualifiedName을 가지고 Document 객체를 만듦



## DOM 인터페이스 - Element

- Element 관련된 함수 제공
- Node 인터페이스에서 상속

프로퍼티	타입	설명
tagName	DOMString(Read)	엘리먼트 이름

함수	설명
DOMString getAttribute(DOMString name)	지정된 name으로 어트리뷰트 값을 반환, 어트리뷰트가 값을 가지고 있지 않으면 빈 문자열 반환
void setAttribute(DOMString name, DOMString value)	특정 어트리뷰트에 새로운 값 할당. 해당 어트리뷰트가 존재하지 않으면 새롭게 생성
void removeAttribute(DOMString name)	어트리뷰트 제거, 어트리뷰트가 기본값을 가지고 있으면 이 기본 값을 가진 동일 어트리뷰트로 대체
void getAttributeNode(DOMString name)	name을 가진 Attr 노드를 반환, 어트리뷰트가 존재하지 않으면 NULL 반환



## DOM 인터페이스 - Element

함수	설명
Attr setAttributeNode (Attr newAttr)	새로운 어트리뷰트 노드 추가. 동일한 이름을 가진 것이 이미 있으면 치환
Attr removeAttributeNode(Attr oldAttr)	지정한 Attr노드를 제거하고 반환
NodeList getElementsByTagName(DOMString name)	주어진 노드 이름을 가진 모든 자손들을 NodeList로 반환
DOMString getAttributeNS(DOMString namespaceURI, DOMString localName)	지정된 어트리뷰트 값을 반환, 어트리뷰트가 지정되지 않거나 값이 없으면 빈 문자열을 반환
void setAttributeNS(DOMString namespaceURI, DOMString qualifiedName, DOMString value)	지정된 어트리뷰트에 새로운 값을 할당
void removeAttributeNS(DOMString namespaceURI, DOMString localName)	지정된 어트리뷰트 제거
Attr getAttributeNodeNS(DOMString namespaceURI, DOMString localName)	지정된 어트리뷰트를 가진 Attr 노드를 반환
Attr setAttributeNodeNS(Attr newAttr)	리스트에 Attr 노드를 추가





## DOM 인터페이스 - NamedNodeMap

- 노드의 집합을 순서 없이 제공
- 노드들은 이름으로 추출 가능

프로퍼티	타입	설명
length	unsigned long(Read)	이 맵에 있는 노드의 개수

함수	설명
Node getItem(DOMString name)	지정된 name을 nodeName으로 가진 Node를 반환
Node setNameItem(Node arg)	arg 파라미터는 추가될 Node 객체. nodeName 프로퍼티가 이름을 위해 사용
Node removeNamedItem(DOMString name)	지정된 이름을 가진 Node가 제거되고 그것을 반환
Node item(unsigned long index)	지정된 인덱스를 가진 노드 반환
Node setNamedItem(Node arg)	arg 파라미터는 추가될 Node 객체



## DOM 인터페이스 - Attr

- Attribute를 다루는 데 필요한 프로퍼티 제공
- Node 인터페이스 상속

프로퍼티	타입	설명
Name	DOMString(Read)	어트리뷰트 이름
Specified	boolean(Read)	어트리뷰트가 지정되었는지 여부를 알려줌
Value	DOMString	어트리뷰트 값
ownerElement	Element(Read)	어트리뷰트가 속한 엘리먼트를 표현



## DOM 인터페이스 - CharacterData

- 문자 데이터를 다루는 데 필요한 프로퍼티, 함수 제공
- Node 인터페이스 상속

프로퍼티	타입	설명
Data	DOMString	CharacterData 노드의 텍스트
length	Unsigned long(Read)	이 노드에 있는 문자 수

함수	설명
DOMString subStringData(unsigned long offset, unsigned long count)	offset으로 시작하는 부분의 문자열을 반환, count만큼 문자열 반환
void appendData(DOMString arg)	문자열 뒤에 arg를 붙임
void insertData(unsigned long offset, unsigned long count)	문자열의 가운데 offset으로 시작하는 부분에 arg 문자열 삽입
void deleteData(unsigned long offset, unsigned long count)	offset부터 문자열 부분을 제거, count만큼 문자열 제거



## DOM 인터페이스 - Text

- 텍스트 노드를 다루는 데 필요한 함수 제공
- CharacterData 인터페이스를 상속

함수	설명
Text splitText(unsigned long offset)	하나의 Text노드를 인접한 두 개의 Text 노드로 나눔 offset까지 첫 번째 노드, 그 이후가 두 번째 노드



- 예제 2-1

- Yoonseunghwan 이라는 Twiter 사용자의 최근 게시물 3개의 내용을 출력한다.
- URL:[http://twitter.com/statuses/user\\_timeline.xml?screen\\_name=correct05&count=3](http://twitter.com/statuses/user_timeline.xml?screen_name=correct05&count=3)

- 파일 구성

- Twitter\_parsingtest.java
- main.xml



## 예제

XML

DOM

DOM 구조

예제

참고자료

```
<?xml version="1.0" encoding="UTF-8"?>
- <statuses type="array">
  + <status>
  + <status>
  + <status>
</statuses>
```

```
<?xml version="1.0" encoding="UTF-8"?>
- <statuses type="array">
  + <status>
  + <status>
  + <status>
</statuses>
```



```
<?xml version="1.0" encoding="UTF-8"?>
```

```
- <statuses type="array">
```

```
- <status>
```

```
  <created_at>Sun Apr 01 02:29:54 +0000 2012</created_at>
```

```
  <id>186279190138585090</id>
```

```
  <text>Life is a targedy when seen in close-up, but a comedy
```

```
  <source>web</source>
```

```
  <full_text>Life is a targedy when seen in close-up, but a comedy
```

```
  <text>Life is a targedy when seen in close-up, but a comedy in long-shot (Charlie Chaplin)</text>
```

```
  <source>web</source>
```

```
  <truncated>>false</truncated>
```

```
  <favorited>>false</favorited>
```

```
  <in_reply_to_status_id/>
```

```
  <in_reply_to_user_id/>
```

```
  <in_reply_to_screen_name/>
```

```
  <retweet_count>0</retweet_count>
```

```
  <retweeted>>false</retweeted>
```

```
+ <user>
```

```
  <geo/>
```

```
  <coordinates/>
```

```
  <place/>
```

```
  <contributors/>
```

```
</status>
```

```
+ <status>
```

```
+ <status>
```

```
</statuses>
```



```
- <statuses type="array">
  - <status>
    <created_at>Sun Apr 01 02:29:54 +0000 2012</created_at>
    <id>186279190138585090</id>
    <text>Life is a targedy when seen in close-up,</text>
    <source>web</source>
    <truncated>>false</truncated>
    <favorited>>false</favorited>
    <in_reply_to_status_id/>
    <in_reply_to_user_id/>
    <in_reply_to_screen_name/>
    <retweet_count>0</retweet_count>
    <retweeted>>false</retweeted>
  - <user>
    <id>223308229</id>
    <name>yoonseunghwan</name>
    <screen_name>correct05</screen_name>
```





## 예제

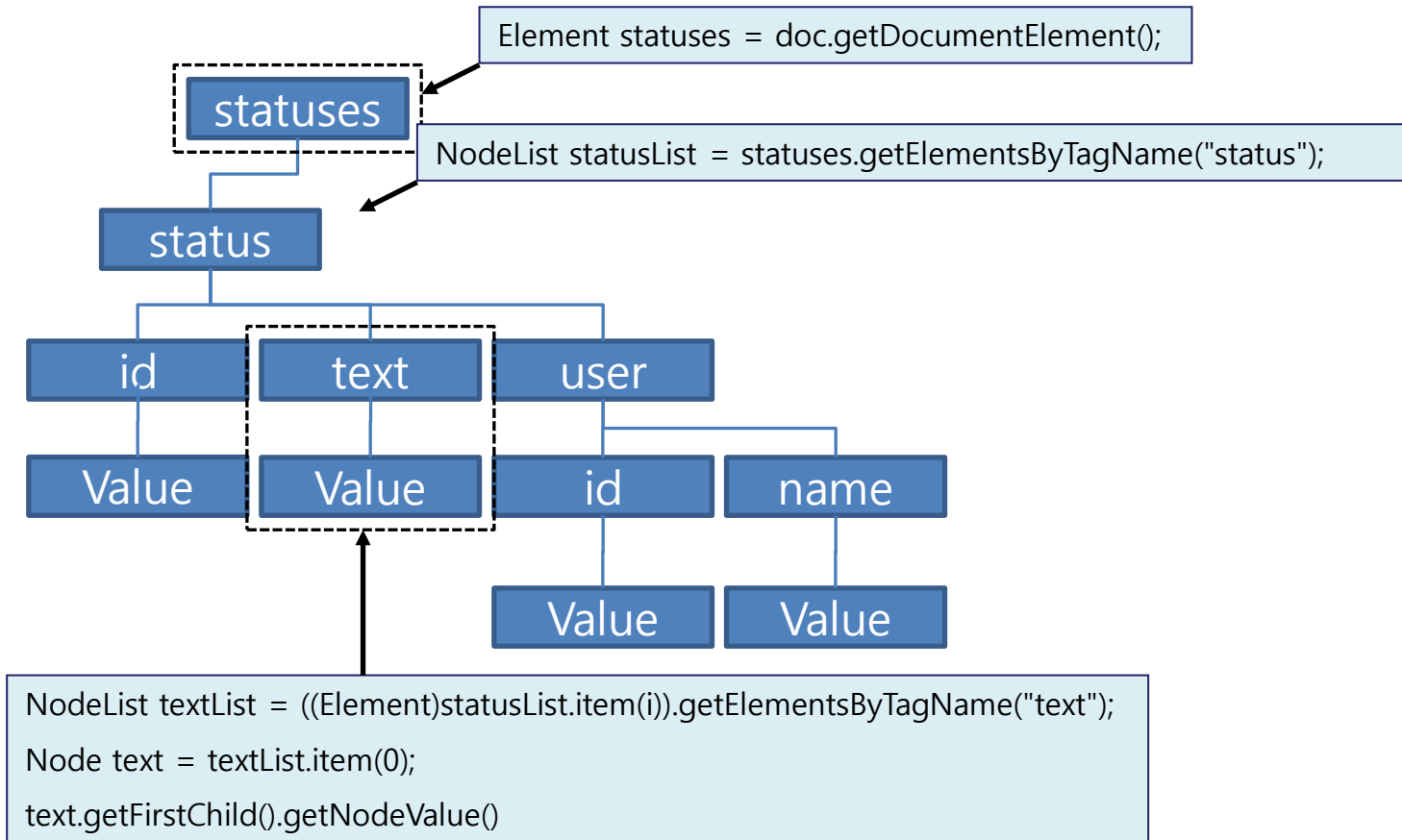
[XML](#)

[DOM](#)

[DOM 구조](#)

[예제](#)

[참고자료](#)



참고



## 예제

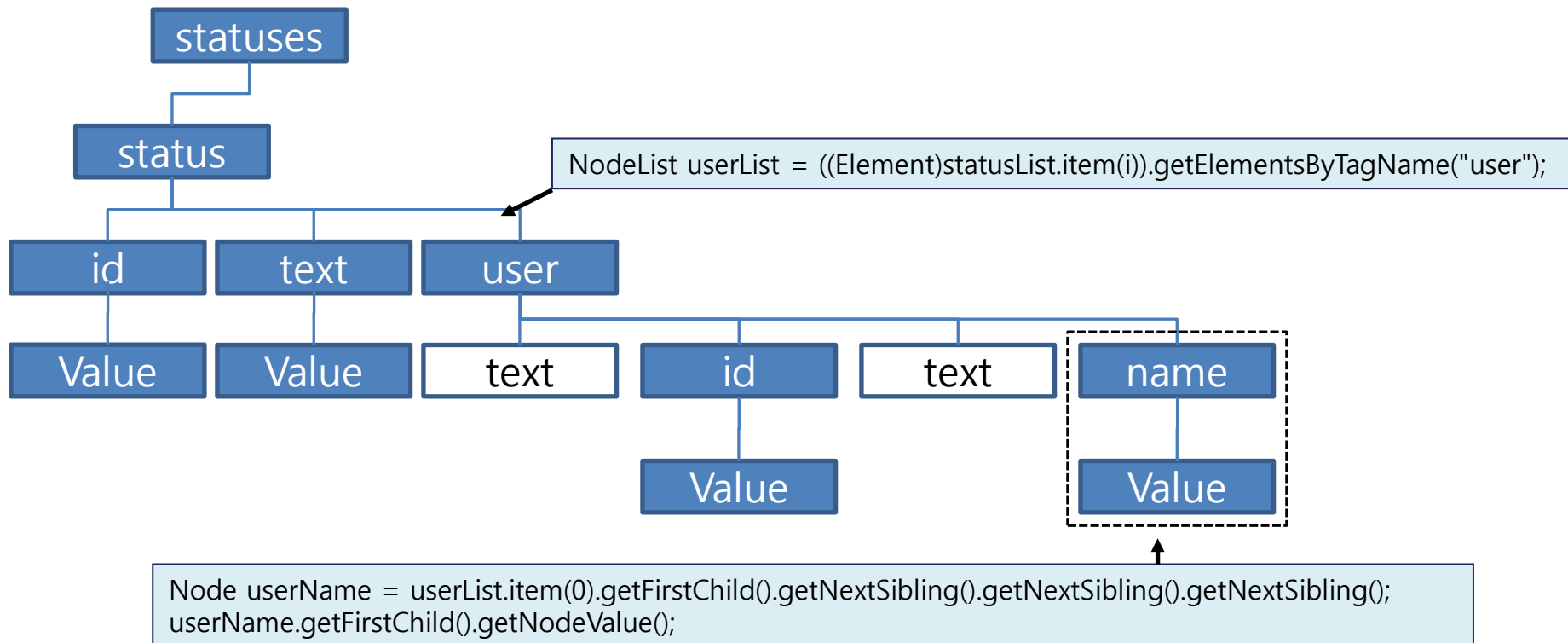
XML

DOM

DOM 구조

예제

참고자료





## twitter\_parsingtest.java

```
package exam.twitter;
import java.net.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import android.app.*;
import android.os.*;
import android.view.*;
import android.widget.*;

public class twitter_parsingtest extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.downxml);

        Button btn = (Button) findViewById(R.id.down);
        btn.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                String xml;
                String url;
                String name = "correct05";
                String output = "";
                int count = 3;
                url = "http://twitter.com/statuses/user_timeline.xml?screen_name="+ name + "&count=" + count;
                xml = DownloadHtml(url);
                EditText result = (EditText) findViewById(R.id.result);
            }
        });
    }
}
```

HTML addr을 사용하여 URL 객체 생성





## twitter\_parsingtest.java

Dom 파서 생성

```
try {  
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();  
    DocumentBuilder builder = factory.newDocumentBuilder();
```

```
    InputStream istream = new ByteArrayInputStream(xml.getBytes("utf-8"));
```

```
    Document doc = builder.parse(istream);
```

Stream 형태로 입력을 받아 parsing

```
    Element statuses = doc.getDocumentElement();
```

root 엘리먼트

```
    NodeList statusList = statuses.getElementsByTagName("status");
```

statuses child 엘리먼트 중  
status의 List

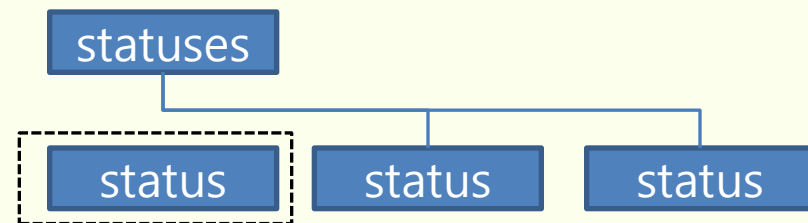
status

status

status



## twitter\_parsingtest.java

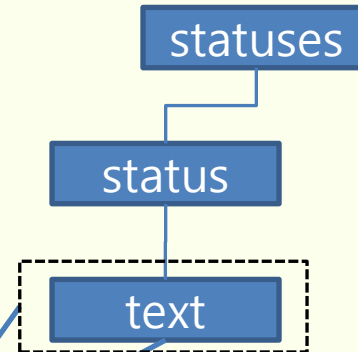


```
for (int i = 0; i < statusList.getLength(); i++) {
    NodeList textList = ((Element)statusList.item(i)).getElementsByTagName("text");
    NodeList userList = ((Element)statusList.item(i)).getElementsByTagName("user");
    Node userName = userList.item(0).getFirstChild().getNextSibling().getNextSibling().getNextSibling();
    Node text = textList.item(0);
    String uName = userName.getFirstChild().getNodeValue();
    output += uName + " : ";
    output += text.getFirstChild().getNodeValue() + "\n\n";
}
} catch (Exception e) { Toast.makeText(v.getContext(), e.getMessage(), 0).show(); }
```

```
result.setText(output);
}};
```



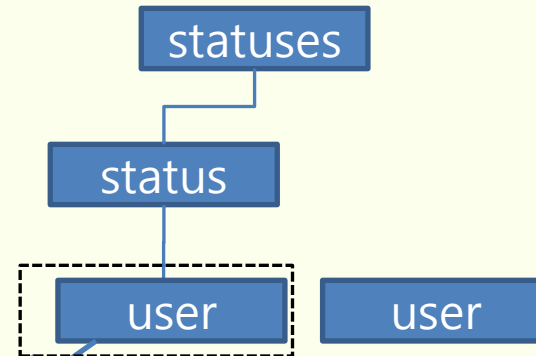
## twitter\_parsingtest.java



```
for (int i = 0; i < statusList.getLength(); i++) {  
    NodeList textList = ((Element)statusList.item(i)).getElementsByTagName("text");  
    NodeList userList = ((Element)statusList.item(i)).getElementsByTagName("user");  
    Node userName = userList.item(0).getFirstChild().getNextSibling().getNextSibling().getNextSibling();  
    Node text = textList.item(0);  
    String uName = userName.getFirstChild().getNodeValue();  
    output += uName + " : ";  
    output += text.getFirstChild().getNodeValue() + "WnWn";  
}  
} catch (Exception e) { Toast.makeText(v.getContext(), e.getMessage(), 0).show(); }  
  
result.setText(output);  
}});
```



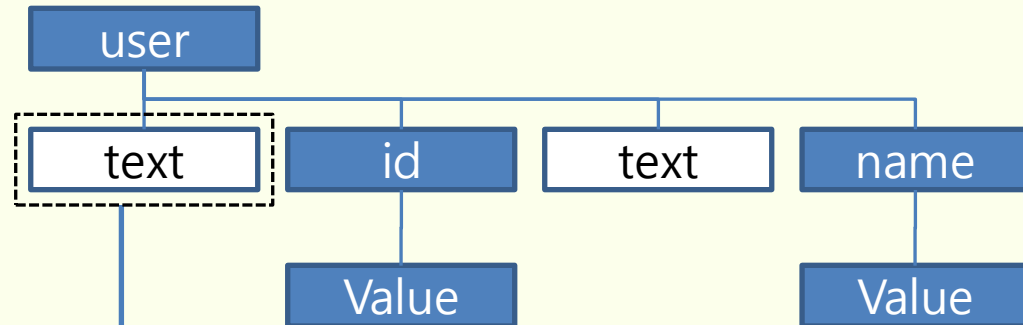
## twitter\_parsingtest.java



```
for (int i = 0; i < statusList.getLength(); i++) {  
    NodeList textList = ((Element)statusList.item(i)).getElementsByTagName("text");  
    NodeList userList = ((Element)statusList.item(i)).getElementsByTagName("user");  
    Node userName = userList.item(0).getFirstChild().getNextSibling().getNextSibling().getNextSibling();  
    Node text = textList.item(0);  
    String uName = userName.getFirstChild().getNodeValue();  
    output += uName + " : ";  
    output += text.getFirstChild().getNodeValue() + "\n\n";  
}  
} catch (Exception e) { Toast.makeText(v.getContext(), e.getMessage(), 0).show(); }  
  
result.setText(output);  
}};
```



## twitter\_parsingtest.java



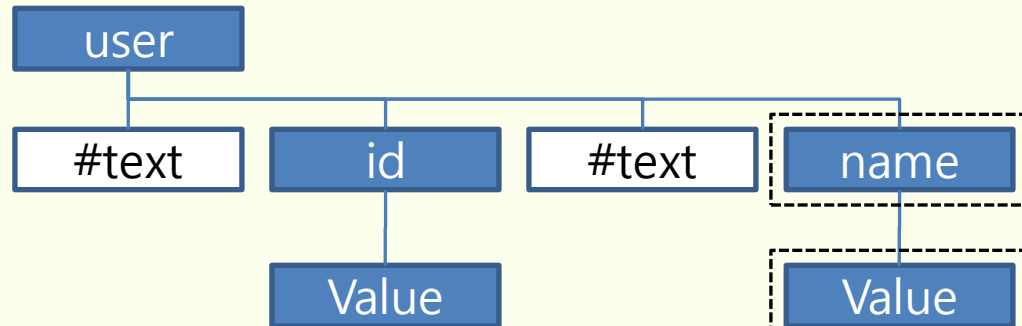
```
for (int i = 0; i < statusList.getLength(); i++) {
    NodeList textList = ((Element)statusList.item(i)).getElementsByTagName("text");
    NodeList userList = ((Element)statusList.item(i)).getElementsByTagName("user");
    Node userName = userList.item(0).getFirstChild().getNextSibling().getNextSibling().getNextSibling();
    Node text = textList.item(0);
    String uName = userName.getFirstChild().getNodeValue();
    output += uName + " : ";
    output += text.getFirstChild().getNodeValue() + "\n\n";
}
} catch (Exception e) { Toast.makeText(v.getContext(), e.getMessage(), 0).show(); }
```

```
result.setText(output);
}};
```





## twitter\_parsingtest.java



```
for (int i = 0; i < statusList.getLength(); i++) {
    NodeList textList = ((Element)statusList.item(i)).getElementsByTagName("text");
    NodeList userList = ((Element)statusList.item(i)).getElementsByTagName("user");
    Node userName = userList.item(0).getFirstChild().getNextSibling().getNextSibling().getNextSibling();
    Node text = textList.item(0);
    String uName = userName.getFirstChild().getNodeValue();
    output += uName + " : ";
    output += text.getFirstChild().getNodeValue() + "\n\n";
}
} catch (Exception e) { Toast.makeText(v.getContext(), e.getMessage(), 0).show(); }
```

```
result.setText(output);
}};
```



## twitter\_parsingtest.java

```
String DownloadHtml(String addr) {  
    StringBuilder html = new StringBuilder();  
    try {  
        URL url = new URL(addr);  
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
        if (conn != null) {  
            conn.setConnectTimeout(10000);  
            conn.setUseCaches(false);  
            if (conn.getResponseCode() == HttpURLConnection.HTTP_OK) {  
                BufferedReader br = new BufferedReader(  
                    new InputStreamReader(conn.getInputStream()));  
                for (;;) {  
                    String line = br.readLine();  
                    if (line == null)  
                        break;  
                    html.append(line + '\n');  
                }  
                br.close();  
            }  
            conn.disconnect();  
        }  
    } catch (Exception ex) {}  
    return html.toString();  
}
```

연결 속성 설정

getReponse 메소드로 요청을 보내고  
요청이 정상적으로 리턴되면 입력  
스트림으로 부터 HTML 문서를  
읽어들인다.



## • 결과

- Yoonseunghwan 이라는 Twiter 사용자의 최근 게시물 3개의 내용을 출력한다.
- 파일 구성
  - Twiter\_parsingtest.java
  - main.xml

