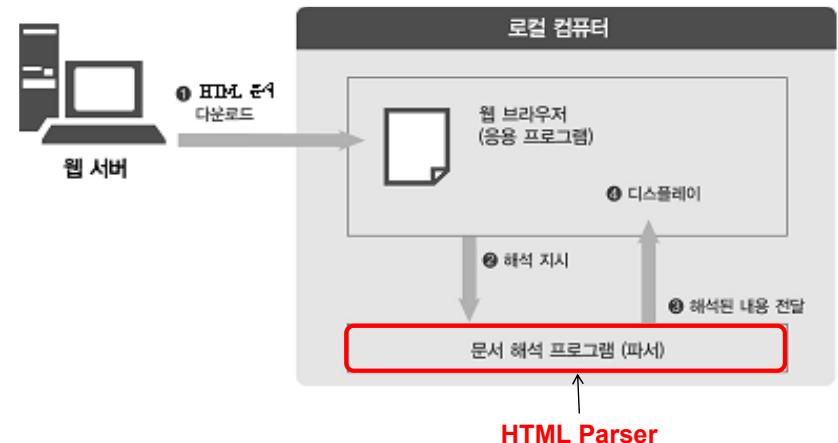


XML 문서 처리 이해

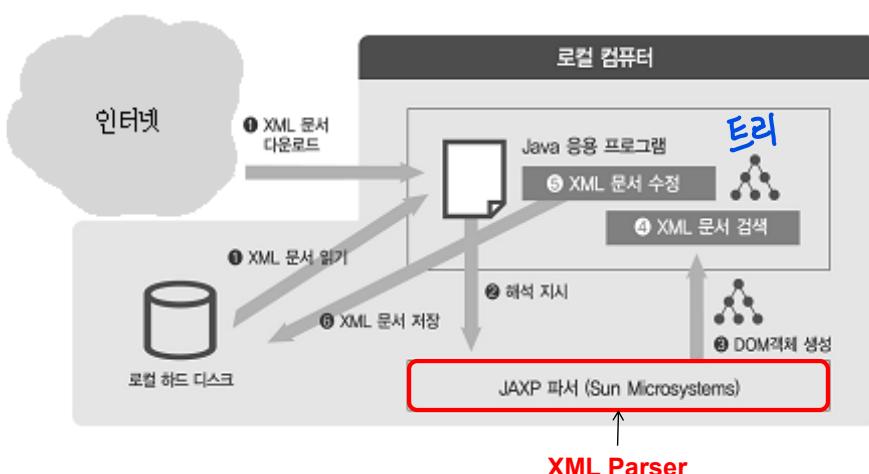
- HTML 문서 처리 과정 (웹 브라우저)



1

XML 문서 처리 이해

- XML 문서 처리 과정 (XML 응용 프로그램)



2

XML 문서 처리 이해

- XML Parser

- XML 문서에 대한 **parsing** 수행
 - 특정 언어의 문장을 분석하여 구성요소들의 문법적 관계를 **해석**
 - ✓ XML 문서를 읽고 해석하여 구성요소들(element name, content, attribute name, attribute value 등)을 식별하고 그것들 간의 문법적인 관계를 해석함
 - 해석 결과를 응용 프로그램이 이용할 수 있도록 메모리 내 구조 생성
- XML 응용 프로그램은 XML Parser의 parsing 결과를 이용
- XML Parser의 예
 - Xerces(Apache), MSXML(Microsoft), XML4J(IBM)

3

XML 문서 처리 이해

XML Parser의 종류

DOM(Document Object Model) Parser

- XML 문서를 해석한 후 메모리에 트리 구조 생성
- 생성된 트리를 이용하여 데이터 검색, 수정, 삭제 실행

SAX(Simple API for XML) Parser

- XML 문서를 해석하는 과정에서 다양한 이벤트(event)를 발생시킴
- 이벤트는 응용 프로그램에서 구현한 이벤트 처리기(handler)에 의해 처리됨

4

DOM 개요

Document Object Model (DOM)

XML이나 HTML 문서를 접근하고 조작하기 위한 표준 방법을 정의

- 프로그램과 스크립트가 동적으로 문서의 내용, 구조, 형식에 접근하고 수정할 수 있게 해 주는 플랫폼 중립적이고 언어 중립적인 인터페이스

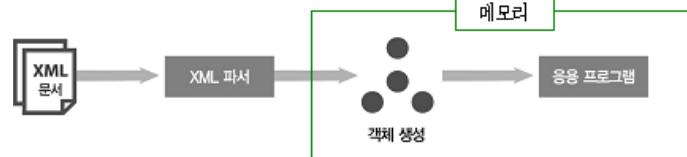
(a *platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document*)

5

DOM 개요

문서의 구성요소들에 대한 객체(object)와 속성(property)을 정의하고, 그들을 접근하는 수단(interface)을 정의

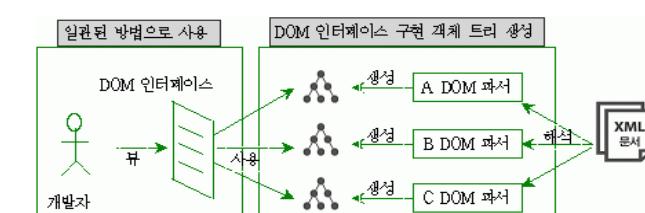
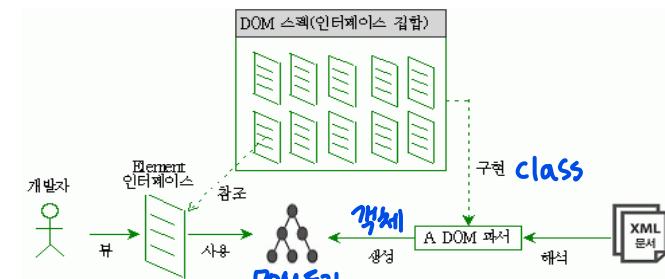
- XML Parser가 XML 문서에 대해 생성해야 하는 객체들의 타입을 정의
- 문서의 구성요소들을 접근하고 조작하기 위해 사용할 수 있는 표준적인 응용 프로그래밍 인터페이스(API) 정의
 - ✓ API를 이용하여 문서(DOM tree)에 속한 엘리먼트와 속성, 텍스트 등을 검색, 수정, 삭제, 생성 가능



6

DOM 개요

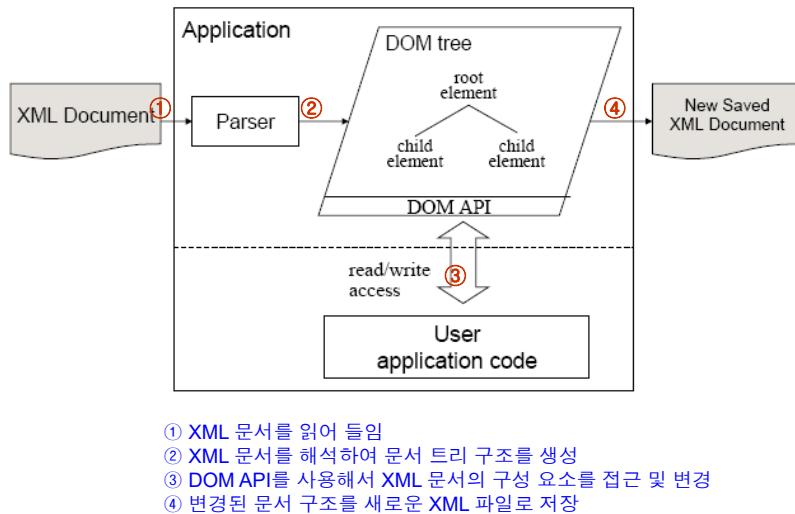
DOM을 이용하여 XML Parser에 독립적인 응용 프로그램 개발 가능



7

DOM 개요

- DOM을 이용한 XML 문서의 접근 및 처리



8

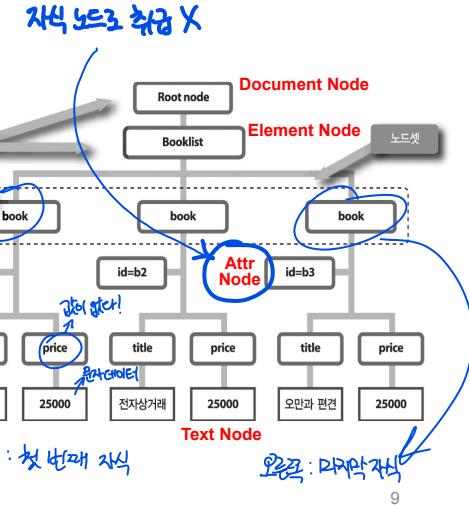
DOM 개요

- DOM Tree

XML 문서를 트리 구조로 표현 가능

XML 문서

```
<?xml version="1.0" encoding="euc-kr"?>
<booklist>
  <book id="b1">
    <title>XML</title>
    <price>25000</price>
  </book>
  <book id="b2">
    <title>전자상거래</title>
    <price>25000</price>
  </book>
  <book id="b3">
    <title>오만과 편견</title>
    <price>25000</price>
  </book>
</booklist>
```



9

DOM 개요

- DOM tree는 Node 타입의 객체들로 구성됨

- XML 문서와 그것에 포함된 모든 구성요소들에 대해 노드 객체 생성
- 노드의 종류 (하위 타입)
 - 문서 노드(Document node): 전체 XML 문서를 나타냄
 - 엘리먼트 노드(Element node): XML 엘리먼트를 나타냄
 - 속성 노드(Attribute node): XML 엘리먼트에 속한 속성을 나타냄
 - 텍스트 노드(Text node): XML 엘리먼트의 텍스트를 나타냄
 - 주석 노드(Comment node): XML 문서 내의 주석을 나타냄
 - 기타

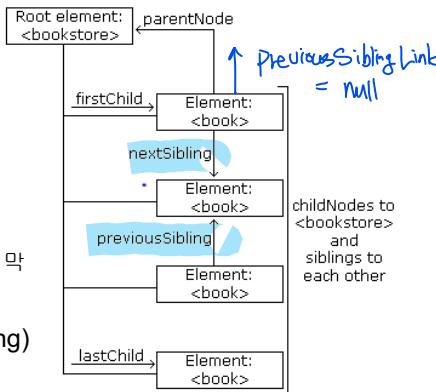
10

DOM 개요

- 노드들 간의 관계

- 부모-자식 관계, 형제 관계
- 최상위 노드를 루트(root)라 함
- 루트를 제외한 모든 노드는 하나의 부모(parent) 노드를 가짐
- 노드는 하나 이상의 자식(child) 노드들을 가질 수 있음
 - 모든 자식노드 집합, 첫번째 자식 노드, 마지막 자식 노드를 조회 가능
- 같은 부모 노드를 갖는 노드들을 형제(sibling) 노드라 함

다음 형제 노드, 이전 형제 노드를 접근 가능
DOM에서는 메모드를 통해서!
직접 link(X)



11

DOM 개요

- 메모리에 대한 고려
 - DOM은 XML 문서에 대한 메모리 내 표현(in-memory representation)으로 객체들의 트리를 생성
 - XML 문서 파일 자체의 크기보다 훨씬 더 많은 메모리 공간이 필요할 수 있음
 - 크기가 큰 XML 문서나 많은 XML 문서들을 동시에 처리하는 경우 시스템의 부하가 크게 증가함
 - DOM에 대한 대안 → SAX(Simple API for XML)
 - 6
메모리로 계속 input을 넣기 어려움

12

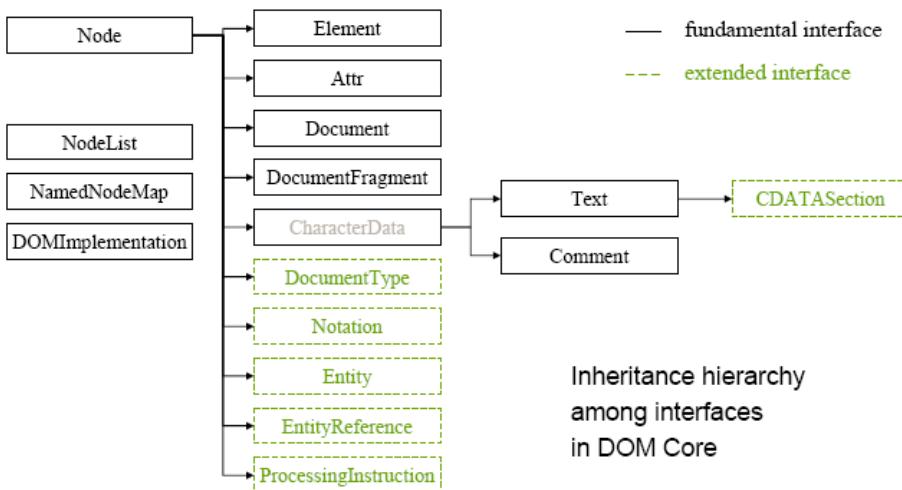
DOM Core Interfaces

- DOM Core Interfaces
 - DOM은 노드들을 접근하기 위한 interface 집합을 정의
 - 최상위 interface는 Node
 - DOM 구현은 XML 문서를 Node 객체들의 계층(hierarchy)으로 표현되는 트리 구조 생성
- 종류 *SAX*
 1. Fundamental interfaces (기본 인터페이스) 핵심
 - Element, Attribute, Document, DocumentFragment, Text, Comment
 2. Extended interfaces (확장 인터페이스)
 - DocumentType, Notation, Entity, EntityReference, ProcessingInstruction, CDATASection

13

DOM Core Interfaces

□ Interfaces 계층도



14

DOM Core Interfaces

□ Fundamental Interfaces

종류	용도
Node	모든 인터페이스들의 부모 인터페이스로, 모든 인터페이스가 공통적으로 갖는 메소드(method)들이 정의되어 있음
NodeList	순서가 있는 노드들의 리스트를 나타내며, 위치(인덱스)를 통해 각 노드들을 접근할 수 있는 인터페이스
NamedNodeMap	순서가 없는 노드들의 집합을 나타내며, 이름을 통해 각 노드를 접근할 수 있는 인터페이스. 주로 속성 노드들의 집합을 나타내기 위해 사용됨
DOMImplementation	Document 객체를 생성하는 메소드를 포함하는 인터페이스
Document	DOM 트리의 최상위 노드로, XML 문서 자체를 나타내는 인터페이스. 동적으로 다른 타입의 노드들을 생성하는 메소드들을 포함
DocumentFragment	DOM 트리의 일부로 사용될 서브 트리를 나타내는 인터페이스
Element	XML 문서에서 엘리먼트를 나타내는 인터페이스
Attr	XML 문서에서 속성을 나타내는 인터페이스
CharacterData	XML 문서에서 문자 데이터를 나타내는 인터페이스
Text	XML 문서에서 엘리먼트의 text content를 나타내는 인터페이스

DOM Core Interfaces

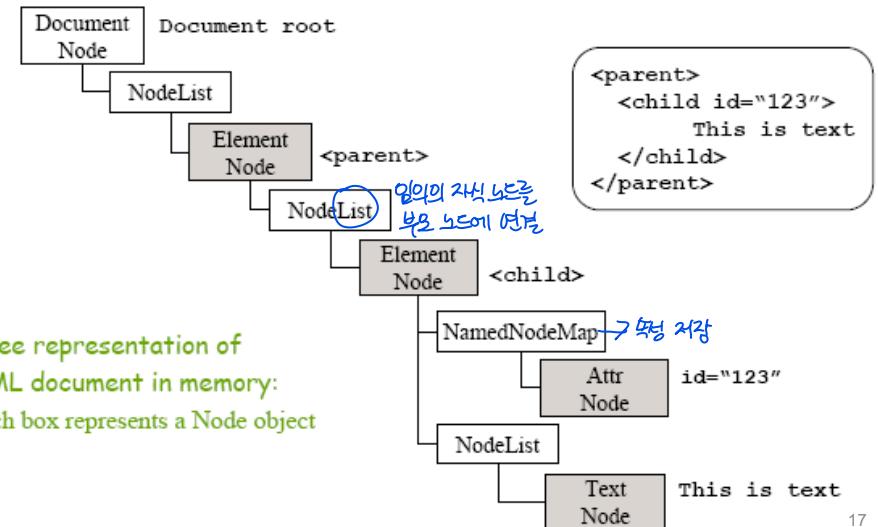
□ Extended Interfaces

종류	용도
CDATASection	XML 문서에서 CDATA 섹션에 해당되는 인터페이스
DocumentType	XML 문서에서 DTD 선언에 해당하는 인터페이스
Notation	DTD에서 Notation 선언에 해당하는 인터페이스
Entity	DTD에서 Entity 선언에 해당하는 인터페이스
EntityReference	XML 문서 또는 DTD에서 Entity 참조에 해당하는 인터페이스
ProcessingInstruction	XML 문서에서 프로세싱 지시자에 해당하는 인터페이스

16

DOM Core Interfaces

□ DOM 트리 구조의 예



17

DOM Core Interfaces

□ Inheritance vs. Flattened views

- Inheritance
 - 상속을 통해 인터페이스들을 계층적으로 정의
 - ✓ 객체의 종류에 따라 해당 인터페이스 이용
- Flattened (simplified) view
 - DOM 트리에 포함된 모든 객체는 Node 객체로 간주될 수 있음
 - ✓ 형 변환(type casting) 없이 Node 인터페이스를 통해 모든 종류의 객체에 대한 접근 및 조작 가능

- API의 중복성
 - 예: 엘리먼트 노드에 대해 다음 두 속성은 같은 값을 나타냄
 - ✓ Node interface에 속한 nodeName 속성) 영향 받을 가능
 - ✓ Element interface에 속한 tagName 속성

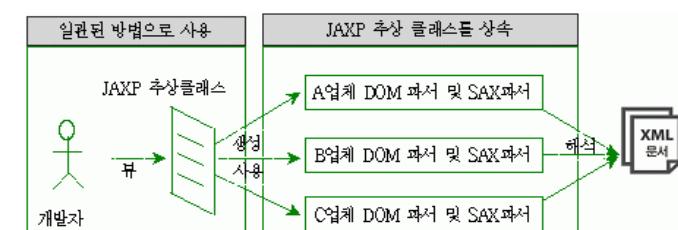
더 이상 Node Type (X) → Element Type

18

JAXP

□ JAXP(Java API for XML Processing)

- Sun에서 개발한 XML 문서 처리를 위한 표준 Class Library
 - javax.xml.* package: XML parsing, XSLT, XPath, StAX 등에 관한 기능 제공
 - org.w3c.dom.* package: DOM spec을 구현
 - org.xml.sax.* package: SAX spec을 구현
- 대부분의 Java 기반 DOM parser들은 JAXP Library를 구현
 - 예: Apache Xerces
- Java SE(JRE)에 포함되어 있음



19

JAXP

- DOM parser 관련 추상 클래스 (`javax.xml.parsers.*`)

추상 클래스 명	설명
DocumentBuilderFactory	DocumentBuilder(DOM parser) 객체를 생성하는 factory class
DocumentBuilder	DOM parser 객체를 나타내는 클래스
SAXParserFactory	SAXParser 객체를 생성하는 factory class
SAXParser	SAX parser 객체의 나타내는 클래스

DOM
SAXP

20

JAXP

- DOM parser 생성

- `DocumentBuilder`를 확장한 클래스의 객체 생성
 - `DocumentBuilderFactory#newDocumentBuilder()` 메소드를 이용하여 생성 (Factory pattern)

```
import javax.xml.parsers.*;
public class CreateDomParser {
    public static void main(String[] args) throws Exception {
        // DOM parser factory 생성
        DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();           // static method
        // DOM parser 생성 (default: JavaSE에 포함된 Apache Xerces Parser 사용)
        DocumentBuilder parser = factory.newDocumentBuilder();
        ...
    }
}
```

21

JAXP

- DOM parser의 기능 설정

- `DocumentBuilderFactory` 객체의 feature 값을 설정

```
public class CreateDomParser {
    public static void main(String[] args) throws Exception {
        // DOM parser factory 생성
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

        // ① WhiteSpace를 Text 객체로 생성하지 않도록 함 (default: false)
        factory.setIgnoringElementContentWhitespace(true); // 공백문자 무시
        // ② Validation 검사를 실시하도록 함 (default: false)
        factory.setValidation(true);
        // ③ Namespace를 인식하도록 함 (default: false)
        factory.setNamespaceAware(true);

        // DOM parser 생성
        DocumentBuilder parser = factory.newDocumentBuilder();
        ...
    }
}
```

22

JAXP

- XML 문서의 parsing

- `Document` 객체를 반환하는 `DocumentBuilder#parse()` 메소드 이용

```
// 로컬 디스크에 있는 XML 문서(파일)에 대한 parsing
public Document parse(String uri);
public Document parse(File f);

// 네트워크를 통해 전송되는 XML 문서(데이터)에 대한 parsing
public Document parse(InputStream is);
public Document parse(InputStream is, String systemId);
public abstract Document parse(InputSource is);
```

- 예

```
// 로컬 디스크에 있는 XML 문서 parsing
Document xmlDoc = parser.parse("C:/Temp/bml.xml");

// 웹 서버에 있는 XML 문서에 대한 parsing
URL url = new URL("http://www.example.com/bml.xml");
InputStream is = url.openStream();
Document xmlDoc = parser.parse(is);
```

23

- XML 문서 parsing 후 루트 엘리먼트 객체 참조 예

```

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

public class GetRootElement {
    public static void main(String[] args) throws Exception {
        // DOM parser 생성
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();

        // XML 문서 parsing하기
        Document document = builder.parse("ch9/bmi.xml");

        // 루트 엘리먼트의 참조 구하기
        Element eRoot = document.getDocumentElement();

        // 루트 엘리먼트 이름 출력
        System.out.println(eRoot.getTagName());
    }
}

```

24

노드로 바꿔도 가능! but, 이름 출력 불가!

getTagName() 사용하면 같은 결과 얻을 수 있음!

Node interface

□ Node 인터페이스의 속성

속성 이름	속성 타입	설명
nodeName	String	노드 이름
nodeValue	Text	노드 값
nodeType	Number	노드 타입. 0부터 시작하는 정수로 표현
parentNode	Node	노드의 부모 노드
childNodes	NodeList	노드의 자식 노드
firstChild, lastChild	Node	첫 번째 자식 노드, 마지막 자식 노드
previousSibling, nextSibling	Node	노드의 형제 노드 중 바로 앞 노드, 바로 다음 노드
attributes	NameNodeMap	노드의 속성들
ownerDocument	Document	노드를 포함하는 문서
namespaceURI, prefix	String	노드의 namespace URI, prefix

Node interface

□ Node 인터페이스

- DOM에서 가장 기본이 되는 인터페이스
- 기능
 - 특정 노드에 대한 정보를 얻음
 - DOM 트리에 노드를 추가, 삭제, 변경 가능
 - DOM 트리의 노드들을 순회(traversal)할 수 있음

Node interface

- 하위 인터페이스에서의 nodeName, nodeValue, attributes 값

하위 인터페이스	nodeName	nodeValue	attributes
Element	태그 이름	null	NamedNodeMap
Attr	속성의 이름	속성의 값	null
Text	#text	텍스트 노드의 내용	null
CDATASection	#cdata-section	CDATA Section의 내용	null
Comment	#comment	주석 내용	null
Document	#document	null	null
DocumentFragment	#document-fragment	null	null
DocumentType	문서 타입 이름	null	null
Entity	개체 이름	null	null
EntityReference	개체 참조 이름	null	null
Notation	notation 이름	null	null
ProcessingInstruction	target	target을 제외한 내용	null

Node interface

- Node type을 표현하기 위한 정적 멤버 필드
 - Node 인터페이스의 `nodeType` 속성의 값으로 사용됨

멤버 필드	상수값	멤버 필드	상수값
ELEMENT_NODE	1	PROCESSING_INSTRUCTION_NODE	7
ATTRIBUTE_NODE	2	COMMENT_NODE	8
TEXT_NODE	3	DOCUMENT_NODE	9
CDATA_SECTION_NODE	4	DOCUMENT_TYPE_NODE	10
ENTITY_REFERENCE_NODE	5	DOCUMENT_FRAGMENT_NODE	11
ENTITY_NODE	6	NOTATION_NODE	12

28

Node interface

- Node 인터페이스의 메소드
 - 노드 정보를 구함

메소드	기능
<code>String getNodeName()</code>	노드 이름을 반환
<code>String getNodeValue()</code>	노드 값을 반환
<code>void setNodeValue(String nodeValue)</code>	노드 값을 설정
<code>short getNodeType()</code>	노드 타입을 반환
<code>NamedNodeMap getAttributes()</code>	속성을 반환
<code>Document getOwnerDocument()</code>	현재 노드가 속한 문서를 반환
<code>String getTextContent()</code>	노드와 그 자손들의 text content를 반환
<code>void setTextContent(string textContent)</code>	새로운 Text 타입 자식 노드를 생성

어떤 노드든 모두 호출 가능

29

Node interface

- 문서(DOM tree) 조작

메소드	기능
<code>Node insertBefore(Node newChild, Node refChild)</code>	refChild 노드 앞에 newChild 노드를 삽입 refChild 노드가 null이면 newChild 노드는 맨 마지막 자식 노드로 삽입
<code>Node replaceChild(Node newChild, Node oldChild)</code>	oldChild 노드를 newChild 노드로 대체
<code>Node removeChild(Node oldChild)</code>	트리에서 oldChild 자식 노드를 제거한 후 그 노드 객체를 반환
<code>Node appendChild(Node newChild)</code>	트리에 노드를 추가 추가하려는 노드 위치에 다른 형제 노드가 있으면 마지막에 노드를 추가함
<code>Node cloneNode(boolean deep)</code>	노드의 복사본을 만드는 메소드 deep 값이 true이면 deep clone, false면 shallow clone 수행 - deep clone: 노드 밑의 서브트리를 모두 복사함 - shallow clone: 해당 노드만 복사 (속성 등은 함께 복사됨)

30

Node interface

- 부모, 자식, 형제 노드 관련 정보를 구함

메소드	기능
<code>Node getParentNode()</code>	부모 노드를 반환
<code>NodeList getChildNodes()</code>	자식 노드를 NodeList 타입으로 반환
<code>Node getFirstChild()</code>	첫 번째 자식 노드를 반환
<code>Node getLastChild()</code>	마지막 자식 노드를 반환
<code>Node getPreviousSibling()</code>	현재 노드의 바로 앞에 있는 형제 노드를 반환
<code>Node getNextSibling()</code>	현재 노드의 바로 뒤에 있는 형제 노드를 반환
<code>boolean hasChildNodes()</code>	단순히 노드가 자식을 가지고 있는지 없는지 확인하는 메소드로, 자식의 유무를 boolean 값으로 반환. 자식이 Text node라도 true를 반환함

31

Node interface

파악 길이 → 노드 인터페이스

□ Node 인터페이스 사용 예

```
import org.w3c.dom.*;
public class GetRootElement {
    public static void main(String[] args) throws Exception {
        ... // DOM parser 생성 및 XML 문서 parsing
        Element root = document.getDocumentElement(); // 루트 엘리먼트의 참조 구하기
        System.out.println( root.getNodeName() ); // 루트 엘리먼트 이름 출력
        // 첫번째 자식 엘리먼트 참조 구하기
        Element firstChildNode = (Element) root.getFirstChild();

        // 첫번째 자식 엘리먼트를 복사하여 마지막 자식으로 추가
        Element cloneNode = firstChildNode.cloneNode(false);
        root.appendChild(cloneNode);
        // 첫번째 자식 엘리먼트를 복사하여 그 앞에 추가 (즉, 맨 앞에 추가)
        cloneNode= firstChildNode.cloneNode(false);
        root.insertBefore(cloneNode, firstChildNode);
        // 첫번째 자식 엘리먼트를 복사하여 두번째 자식 엘리먼트를 대체
        cloneNode= firstChildNode.cloneNode(false);
        root.replaceChild(cloneNode, firstChildNode.getNextSibling());
    }
}
```

32

Node interface

□ Node 인터페이스를 이용한 DOM 트리 구조 접근

– Node 인터페이스의 관련 메소드 이용

- ✓ Node getParentNode()
- ✓ NodeList getChildNodes()
- ✓ Node getFirstChild()
- ✓ Node getLastChild()
- ✓ Node getPreviousSibling()
- ✓ Node getNextSibling()
- ✓ boolean hasChildNodes()
- node.getFirstChild() != null
- node.getLastChild() != null
- node.getChildNodes().getLength() > 0

동일한 결과

length=0 이면, 자식이 0

33

Node interface

– 자식 노드들을 문서 순서(document order)대로 처리하는 방법

```
for (Node child = node.getFirstChild();
     child != null;
     child = child.getNextSibling()) {
    process child ...
}
```

또는

```
NodeList childNodes = node.getChildNodes();
for (int i = 0; i < childNodes.getLength(); i++) {
    Node child = childNodes.item(i);
    process child ...
}
```

DOM에서 정의된 NodeList의 차이
Size()가 아니라
getLength() 사용!

▪ Reverse order로 처리하는 방법은?

`getPreviousSibling()` or `i = length - 1`에서 시작!

34

Node interface

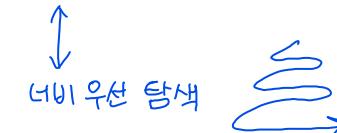
– 트리 탐색 (Tree traversal) 전위 후위

▪ 방법 1

가장 간결한 코드 → 재귀 사용 ... 또 다른 방법 : stack 사용

```
Public void processNodeRecursively(Node node) {
    process the current node ...
    for (Node child = node.getFirstChild();
         child != null;
         child = child.getNextSibling()) {
        processNodeRecursively(child); // process child
    }
}
```

전위
⇒ "pre-order depth-first traversal" = 깊이 우선 탐색



35

Node interface

- 트리 탐색 (Tree traversal)
 - 방법 2: 반복자(iterator) 인터페이스 이용
 - ✓ [org.w3c.dom.traversal](#) API 참조

```
import org.w3c.dom.traversal.DocumentTraversal;
import org.w3c.dom.traversal.NodeIterator;
import org.w3c.dom.traversal.NodeFilter;

DocumentTraversal trav = (DocumentTraversal) node.getOwnerDocument();
NodeIterator iterator = trav.createNodeIterator(node, NodeFilter.SHOW_ALL, null, false);
while ((Node child = iterator.nextNode() ) != null) {
    // process child ...
}
iterator.detach();
```

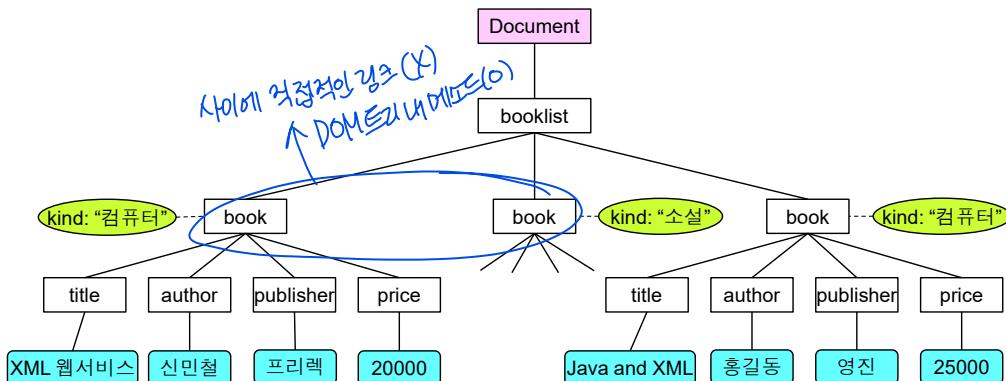
✓ 장점

- 탐색 시작 전에 탐색에 포함될 노드 타입들을 지정 가능
 - 예: NodeFilter.{SHOW_ALL, SHOW_ELEMENT, SHOW_TEXT, ... }
- 탐색 방향을 변경 가능
 - 예: NodeIterator#nextNode(), NodeIterator#previousNode()

앞의 원래 탐색 ↓
 ↓
 반대! 방향

36

Node interface



38

Node interface

Examples

- bml.xml

```
<booklist>
    <book kind="컴퓨터">
        <title>XML 웹 서비스</title>
        <author>신민철</author>
        <publisher>프리렉</publisher>
        <price>20000</price>
    </book>
    <book kind="소설">
        <title>금붕어 메세지</title>
        <author>이등문</author>
        <publisher>무한(구)생각과기억</publisher>
        <price>9000</price>
    </book>
    <book kind="컴퓨터">
        <title>Java And XML</title>
        <author>홍길동</author>
        <publisher>영진 출판사</publisher>
        <price>25000</price>
    </book>
</booklist>
```

37

Node interface

Examples

- GetFirstBookTitle.java

```
public static void main(String[] args) throws Exception {
    // DOM parser 생성
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    factory.setIgnoringElementContentWhitespace(true);
    DocumentBuilder builder = factory.newDocumentBuilder();

    // XML 문서 parsing하기
    Document document = builder.parse("ch9/bml.xml");

    // 루트 엘리먼트 참조 구하기
    Element eRoot = document.getDocumentElement();

    // 첫번째 book 엘리먼트 정보 구하기
    Node eBook = eRoot.getFirstChild();
    Node eTitle = eBook.getFirstChild();
    Text tTitle = (Text) eTitle.getFirstChild();
    String strTitle = tTitle.getData();

    System.out.println(strTitle);
}
```

why?
인덴테이션,
줄바꿈
무시

39

Node interface

Examples

- GetLastBookPrice.java

```
public static void main(String[] args) throws Exception {
    // DOM parser 생성
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    factory.setIgnoringElementContentWhitespace(true);
    DocumentBuilder builder = factory.newDocumentBuilder();

    // XML 문서 parsing하기
    Document document = builder.parse("ch9/bm1.xml");

    // 루트 엘리먼트 참조 구하기
    Element eRoot = document.getDocumentElement();

    // 마지막 book 엘리먼트 정보 구하기
    Element eBook = (Element) eRoot.getLastChild();
    Element ePrice = (Element) eBook.getLastChild();
    Text tPrice = (Text) ePrice.getFirstChild();
    String strPrice = tPrice.getData();

    System.out.println(strPrice);
}
```

40

Node interface

Examples

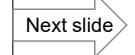
- GetFirstBookInfo.java

```
// 첫번째 book 엘리먼트 정보 구하기
Element eBook = (Element) eRoot.getFirstChild();
Element eTitle = (Element) eBook.getFirstChild();
String strTitle = ((Text)eTitle.getFirstChild()).getData();
System.out.println("제목: " + strTitle);

Element eAuthor = (Element) eTitle.getNextSibling();
Text tAuthor = (Text) eAuthor.getFirstChild();
String strAuthor = tAuthor.getData();
System.out.println("저자: " + strAuthor);

Element ePublisher = (Element) eAuthor.getNextSibling();
Text tPublisher = (Text) ePublisher.getFirstChild();
String strPublisher = tPublisher.getData();
System.out.println("출판사: " + strPublisher);

Element ePrice = (Element) ePublisher.getNextSibling();
Text tPrice = (Text) ePrice.getFirstChild();
String strPrice = tPrice.getData();
System.out.println("가격: " + strPrice);
```



41

Node interface

Examples

switch문으로 구현 가능

- GetFirstBookInfo2.java

```
// 첫번째 book 엘리먼트 정보 구하기
Element eBook = (Element) eRoot.getFirstChild();

for(Node ch = eBook.getFirstChild(); ch != null; ch = ch.getNextSibling())
{
    if (ch.getNodeName().equals("title"))
        System.out.print("제목: ");
    else if (ch.getNodeName().equals("author"))
        System.out.print("저자: ");
    else if (ch.getNodeName().equals("publisher"))
        System.out.print("출판사: ");
    else if (ch.getNodeName().equals("price"))
        System.out.print("가격: ");
    System.out.println(ch.getFirstChild().getNodeValue());
    // == System.out.println(((Text)ch.getFirstChild()).getData()); 와 동일
}
```

Text는 각자 사용 가능

42

Document interface

Document 인터페이스

문서 전체에 대한 접근을 제어하는 인터페이스

Node 인터페이스에서 확장된 인터페이스

- Node 인터페이스의 메소드와 속성을 Document 객체 내에서 사용 가능

– 가능한 작업

- 노드 트리 검색 및 노드 목록 반환
- 새로운 노드나 속성을 생성

– 제공되는 메소드

- 문서 관련 정보를 얻는 메소드
- 문서 트리를 순회하기 위한 메소드
- 문서를 작성하기 위한 메소드

43

Document interface

- Document 인터페이스의 메소드
 - 문서 관련 정보를 얻는 메소드

메소드	내용
DocumentType getDoctype()	DocumentType 객체를 반환 DocumentType 인터페이스는 문서에 정의된 개체에 접근하기 위한 메소드들을 제공함
DOMImplementation getImplementation()	DOMImplementation 객체를 반환 DOMImplementation은 parser의 특성을 파악하기 위한 메소드를 가지고 있다.

44

Document interface

- 문서 트리를 순회하기 위한 메소드

메소드	내용
Element getDocumentElement()	문서의 Root Element를 반환
Element getElementByID(String elementID)	elementID에 지정된 ID를 가진 엘리먼트를 반환하고 지정된 ID를 가진 엘리먼트가 없으면 null을 반환
NodeList getElementsByTagNameNS(String namespaceURI, String localName)	namespaceURI로 지정된 네임스페이스와 localName에 지정된 로컬 이름을 가진, 문서 안에 있는 모든 엘리먼트의 NodeList를 반환
NodeList getElementsByTagName(String tagName)	문서 안에 tagName의 이름을 가진 모든 엘리먼트의 NodeList 객체를 반환

45

Document interface

- Factory Methods: 다른 노드를 생성

메소드	내용
Element createElement(String tagName)	tagName에 지정된 이름의 Element를 생성
Element createElementNS(String namespaceURI, String qualifiedName)	주어진 이름과 네임스페이스를 이용해서 Element를 생성
DocumentFragment createDocumentFragment()	비어있는 DocumentFragment 객체를 생성
Text createTextNode(String data)	data의 텍스트가 있는 Text 노드를 생성
Comment createComment(String data)	data의 텍스트가 있는 Comment 노드를 생성
CDATASection createCDATASection(String data)	data의 텍스트가 있는 CDATASection 노드를 생성

46

Document interface

메소드	내용
ProcessingInstruction createProcessingInstruction(String target, String data)	지정된 target과 data를 가진 ProcessingInstruction 노드를 생성. target은 ProcessingInstruction이 가리키는 애플리케이션을 지정하는 문자열, data는 target을 제외한 나머지 부분의 문자열을 지정
Attr createAttribute(String name)	지정된 name의 속성을 생성. 생성된 노드는 Node 인터페이스의 nodeName 속성에 접근 가능
Attr createAttributeNS(String namespaceURI, String qualifiedName)	주어진 이름과 네임스페이스를 이용해서 속성을 생성
EntityReference createEntityReference(String name)	지정된 name의 개체 참조를 생성 필요한 곳으로 노드를 복사할 수 있도록 함
Node importNode(Node importNode, boolean deep)	다른 문서에서 이 문서로 importNode 노드를 가져옴. 원래의 노드는 이전 문서에서 제거되지 않고 복사됨. deep속성은 deep 또는 shallow copy 여부를 지정

true false

47

Document interface

Examples

GetBookTitle2.java

```
public static void main(String[] args) throws Exception {
    // DOM parser 생성
    ...
    // XML 문서 parsing하기
    Document document = builder.parse("ch9/bml.xml");

    // 책 제목들만 뽑아오기
    NodeList nlTitles = document.getElementsByTagName("title");
    for (int i = 0; i < nlTitles.getLength(); i++) {
        Element eTitle = (Element) nlTitles.item(i);
        Text tTitle = (Text) eTitle.getFirstChild();
        String strTitle = tTitle.getData();
        System.out.println(strTitle);
    }
}
```

48

Document interface

Examples

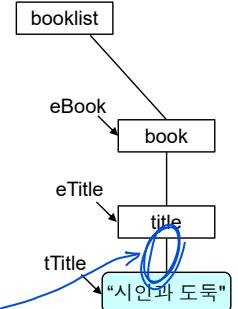
AppendNewBook.java

```
public static void main(String[] args) throws Exception {
    // DOM parser 생성
    ...
    // XML 문서 parsing하기
    Document document = builder.parse("ch9/bml.xml");

    //루트 엘리먼트 객체 참조 구하기
    Element eRoot = document.getDocumentElement();

    //book 엘리먼트 객체 생성
    Element eBook = document.createElement("book");

    //title 엘리먼트 객체 생성
    Element eTitle = document.createElement("title");
    Text tTitle = document.createTextNode("시인과도둑");
    eTitle.appendChild(tTitle);
    ...
    eBook.appendChild(eTitle);
    ...
    eRoot.appendChild(eBook);
}
```

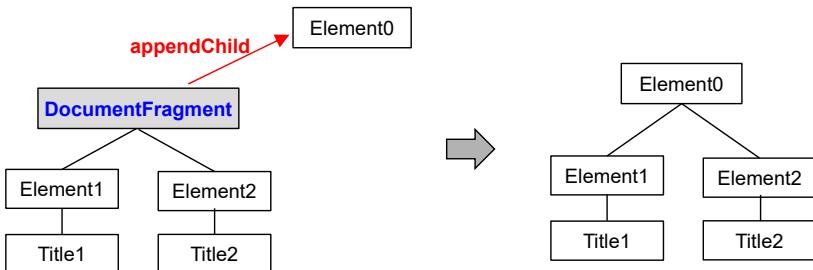


49

DocumentFragment interface

DocumentFragment 인터페이스

- XML 문서의 일부분을 임시로 보관하기 위해 사용
- 객체들의 서브 트리에 대한 연산에 이용
- DocumentFragment 노드가 DOM 트리에 삽입될 때 그것의 모든 자식 노드들이 트리에 삽입됨
- Node 인터페이스에 정의된 모든 메소드 사용 가능



50

DocumentFragment interface

DocumentFragment 인터페이스 사용 예

```
public static void main(String[] args) throws Exception {
    // DOM parser 생성
    ...
    // XML 문서 parsing하기
    Document document = builder.parse("ch9/bml.xml");

    // DocumentFragment 생성
    DocumentFragment docFrag = document.createDocumentFragment();
    Element e = document.createElement("출판일")
    Text t = document.createTextNode("2013년 5월 1일")
    e.appendChild(t);
    docFrag.appendChild(e);

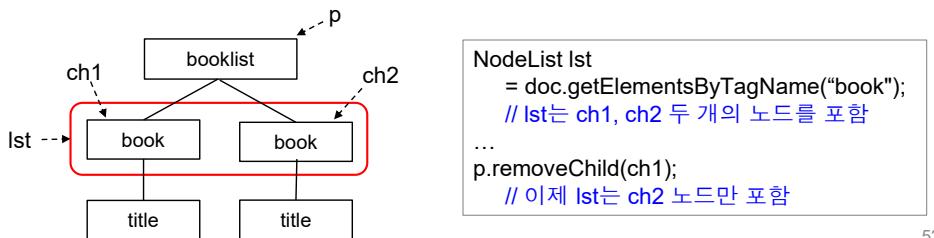
    // DocumentFragment를 첫번째 book의 마지막 child로 삽입
    document.getDocumentElement().getFirstChild().appendChild(docFrag);
}
```

51

NodeList interface

❑ NodeList 인터페이스

- 하나 이상의 노드들의 집합(node set)을 반환할 때 이용됨
 - 정렬된 집합체 형태
- NodeList에 포함된 노드들은 인덱스 값을 이용해서 접근할 수 있음
 - 인덱스는 0부터 시작
- 항상 live 상태를 유지
 - DOM tree에 노드를 삽입하거나 삭제할 경우 항상 그 결과를 반영함



NodeList interface

❑ NodeList 인터페이스에서 제공하는 메소드

메소드	내용
<code>int getLength()</code>	NodeList 안에 포함된 노드 수를 반환
<code>Node item(int index)</code>	NodeList 목록 중에서 지정된 곳의 Node를 반환 index는 0~length-1 index== length일 경우 null을 반환

❑ NodeList 인터페이스 사용 예

- Document 인터페이스의 사용 예 (GetBookTitle2.java) 참조

NamedNodeMap interface

❑ NamedNodeMap 인터페이스

- NodeList와 유사하나 정렬되지 않은 집합체를 표현
- 이름을 이용해서 노드에 접근하고자 할 때 사용 가능
 - 주로 엘리먼트에 포함된 속성들을 추출하기 위해 사용
- 노드는 이름 또는 인덱스 값을 이용해서 접근할 수 있음
 - 인덱스는 0부터 시작

NamedNodeMap interface

❑ NamedNodeMap 인터페이스의 메소드

메소드	내용
<code>Node getNamedItem(String name)</code>	노드 이름이 name으로 지정한 것과 같은 Node를 찾아서 반환함. 해당 노드가 없으면 null을 반환
<code>Node setNamedItem(Node arg)</code>	새로운 노드를 추가. → 추가 같은 이름의 노드가 존재하면 기존 노드를 대체하고 대체된 Node를 반환함. 그렇지 않으면 null을 반환
<code>Node removeNamedItem(String name)</code>	name으로 지정한 Node를 제거하고 해당 Node를 반환
<code>Node item(int index)</code>	index로 지정한 위치의 Node를 반환. index가 length보다 크거나 같으면 null을 반환
<code>int getLength()</code>	NamedNodeMap에 포함된 노드 수를 반환

NamedNodeMap interface

NamedNodeMap 인터페이스 사용 예

```
public static void main(String[] args) throws Exception {  
    // DOM parser 생성  
    ...  
    // XML 문서 parsing하기  
    Document document = builder.parse("ch9/bml.xml");  
    //루트 엘리먼트 객체참조 구하기  
    Element eRoot = document.getDocumentElement();  
  
    // 첫번째 book 엘리먼트의 속성들을 추출  
    NamedNodeMap map = eRoot.getFirstChild().getAttributes();  
  
    // kind 속성 값 출력  
    String str = map.getNamedItem("kind").getNodeValue();  
    System.out.println(str);  
  
    // kind 속성을 삭제한 후 값을 재설정하여 삽입  
    Node n = map.removeNamedItem("kind");  
    n.setNodeValue("computer");  
    map.setNamedItem(n);  
}
```

트리에도 추가됨!

56

Element interface

Element 인터페이스

- XML 문서의 각 엘리먼트를 표현
- Element로 바로 접근해서 작업할 수 있도록 하는 인터페이스
- 유일한 속성은 *tagName* 속성
 - 엘리먼트 이름을 반환하며, 읽기 전용 속성으로 사용
- 메소드
 - Element 객체의 태그 및 속성에 접근하기 위한 메소드

57

Element interface

Element 인터페이스의 메소드

메소드	내용
<code>String getAttribute(String name)</code>	지정된 name 속성값을 반환. 속성에 지정된 값이 없거나, 기본값을 가지고 있지 않으면 빈 문자열을 반환
<code>Attr setAttribute(String name, String value)</code>	지정된 name을 갖는 속성 값을 value 값으로 변경. 해당 속성이 존재하지 않으면 새로운 속성을 추가
<code>void removeAttribute(String name)</code>	지정된 name을 갖는 속성을 제거. 기본값이 정의된 속성인 경우 새로운 속성으로 대체
<code>Attr getAttributeNode(String name)</code>	지정된 name의 속성을 담고 있는 Attr 노드를 반환. 해당 노드가 없으면 null을 반환
<code>Attr setAttributeNode(Attr newAttr)</code>	현재 엘리먼트에 새로운 Attr 노드를 추가. 같은 이름의 속성이 이미 존재하면 대체함. Attr이 대체되면 대체된 값이 반환되고, 그렇지 않으면 null을 반환
<code>Attr removeAttributeNode(Attr oldAttr)</code>	지정된 Attr 노드를 제거하고 반환. 기본값이 정의된 속성인 경우 새로운 Attr 속성으로 대체

P.45
앞서
언급해
았음

Element interface

Element 인터페이스의 메소드

메소드	내용
<code>String getTagName()</code>	태그 이름을 반환
<code>NodeList getElementsByTagName(String name)</code>	name의 태그 이름을 갖는 모든 자손 노드들의 NodeList를 반환 (in document order)
<code>NodeList getElementsByTagNameNS(String namespaceURI, String localName)</code>	주어진 이름과 네임스페이스를 갖는 모든 엘리먼트들의 리스트를 반환
<code>boolean hasAttribute(String name)</code>	주어진 이름의 속성이 있는 경우, true를 반환
<code>boolean hasAttributeNS(String namespaceURI, String localName)</code>	주어진 이름과 네임스페이스를 갖는 속성이 있는 경우, true를 반환

59

Element interface

Example

- GetFirstBookKind2.java + ModifyFirstbook.java

```
public static void main(String[] args) throws Exception {
    // DOM parser 생성
    ...
    // XML 문서 parsing하기
    ...
    // 루트 엘리먼트 참조 구하기
    ...
    // 첫번째 book 엘리먼트의 kind 속성값 변경
    Element eBook = (Element) eRoot.getFirstChild();
    String strKind = eBook.getAttribute("kind");
    System.out.println(strKind);
    eBook.setAttribute("kind", "computer");
    // == eBook.getAttributeNode("kind").setNodeValue("computer");
    System.out.println(eBook.getAttribute("kind"));
    // 첫번째 book 엘리먼트에 새로운 속성 추가
    eBook.setAttribute("publishDate", "20130501");
    System.out.println(eBook.getAttribute("publishDate"));
}
```

60

Node 라이브 (Element)³
타입 캐스팅!

기존에 없음 → 새로 추가됨!

Attr interface

Attr 인터페이스

- 엘리먼트의 속성에 대한 인터페이스 제공
- Node 인터페이스를 확장한 인터페이스
- 속성은 XML 트리 구조의 일부가 아님
 - 속성은 관련된 엘리먼트와 자식 및 부모 관계가 아님
 - 단지 엘리먼트의 속성을 표현
 - Node의 속성인 `parentNode`, `previousSibling`, `nextSibling`은 null 값을 갖음
- Attr의 속성
 - `name`: 이름을 반환하는 읽기전용 속성
 - `value`: 값을 나타내는 속성
- ▶ `ownerElement`: 이 속성을 포함하는 엘리먼트를 나타냄

61

Attr interface

Attr 인터페이스의 메소드

메소드	내용
<code>String getName()</code>	속성 이름을 반환
<code>boolean getSpecified()</code>	원 문서에서 속성을 명시적인 값으로 준 경우 true를 반환하고, 그렇지 않은 경우에는 false를 반환
<code>String getValue()</code>	속성 값을 반환
<code>void setValue(String value)</code>	속성 값을 설정
<code>Element getOwnerElement()</code>	해당 속성이 속한 Element 노드를 반환



기타

CharacterData, Text 인터페이스

- XML 문서 내의 문자데이터 및 텍스트를 표현
- Text는 CharacterData의 확장
 - CharacterData 자체는 DOM 객체로 사용되지 않음
 - 자식 노드를 포함할 수 없음

Comment 인터페이스

- 주석문을 표현하기 위한 인터페이스
- 정의된 method나 property는 없음

62

63

기타

CharacterData 인터페이스의 메소드

메소드	내용
void appendData(String arg)	기존 문자 데이터의 끝에 arg 문자열을 추가
void deleteData(int offset, int count)	offset에서 시작하는 문자열의 일부를 삭제. count로 지정된 길이만큼 삭제하거나, 그보다 길이가 짧다면 문자열의 끝까지 삭제
String getData()	CharacterData 노드의 문자열을 반환
int getLength()	CharacterData 노드의 문자열의 길이를 반환
void insertData(int offset, String arg)	offset으로 지정된 위치에 arg 문자열을 삽입
void replaceData(int offset, int count, String arg)	offset에서 시작하고 count 값의 길이를 갖는 문자열의 일부를 arg 문자열로 대체. 남은 문자열의 길이가 count보다 짧다면 문자열의 끝까지 대체함
void setData(String data)	CharacterData 노드의 문자열을 설정 → 기존 값 변경 가능
String substringData(int offset, int count)	offset에서 시작해서 count로 값의 길이에 해당하는 부분 문자열을 반환

64

기타

Text 인터페이스의 메소드

메소드	내용
Text splitText(int offset)	offset을 기준으로 텍스트를 두 개로 분리

DOM을 활용한 XML 문서 조작

Examples

bml.xml

```
<booklist>
    <book kind="컴퓨터">
        <title>XML 웹 서비스</title>
        <author>신민철</author>
        <publisher>프리렉</publisher>
        <price>20000</price>
    </book>
    <book kind="소설">
        <title>금붕어메세지</title>
        <author>이능문</author>
        <publisher>무한(구)생각과기억</publisher>
        <price>9000</price>
    </book>
    <book kind="컴퓨터">
        <title>Java And XML</title>
        <author>홍길동</author>
        <publisher>영진 출판사</publisher>
        <price>25000</price>
    </book>
</booklist>
```

66

DOM을 활용한 XML 문서 조작

데이터 추가

기존의 XML 문서에 새로운 노드를 추가

관련 메소드

Document 인터페이스

- ✓ createXXX(...)
- 예: Element createElement(String tagName)
Text createTextNode(String data)

Node 인터페이스

- ✓ Node insertBefore(Node newChild, Node refChild)
- ✓ Node appendChild(Node newChild)
- ✓ Node replaceChild(Node newChild, Node refChild)
- ✓ Document getOwnerDocument()
- ✓ Node cloneNode(boolean deep)

67

DOM을 활용한 XML 문서 조작

- 엘리먼트 및 텍스트 노드 추가 예

- 문서에 포함된 모든 book 엘리먼트를 찾아 publishDate 자식 엘리먼트 추가

```
public static void addPublishDate(Node n) { // n은 root node부터 시작
    if (n.getNodeType() == Node.ELEMENT_NODE) { // 엘리먼트를 처리함
        if (n.getnodeName().equals("book")) { // book 엘리먼트 발견
            Document doc = n.getOwnerDocument(); // 문서 객체를 찾음
            Element date = doc.createElement("publishDate"); // 엘리먼트 노드 생성
            Text txt = doc.createTextNode("20130501"); // text node 생성
            date.appendChild(txt); // <publishDate>20130501</publishDate>
            n.appendChild(date); // 현재 엘리먼트의 마지막 자식 엘리먼트로 추가
        }
        for (Node ch = n.getFirstChild(); ch != null; ch = ch.getNextSibling()) {
            addPublishDate(ch); // 각 자식 엘리먼트에 대해 recursion 수행
        }
    }
}
```

깊이우선 탐색

Document가 넘어오지 않는데
createElement(), createTextNode()를
사용해야 한다

기본 노드에 대해서는
removeChild()를 사용해야 한다

68

DOM을 활용한 XML 문서 조작

□ 데이터 삭제

- 기존 XML 문서에서 노드 삭제

- 관련 메소드

▪ Node 인터페이스

- ✓ Node getParentNode()

- ✓ Node removeChild(Node Child)

부모 노드에 대해서는
removeChild()를 사용해야 한다

기본 노드에 대해서는
removeChild()를 사용해야 한다

69

DOM을 활용한 XML 문서 조작

- 엘리먼트 삭제 예

- 문서에 포함된 모든 price 엘리먼트(및 그 sub-tree)를 찾아 삭제함

```
public static void delPrice(Node n) { // n을 root로 하는 sub-tree 내에서 price 삭제
    for (Node ch = n.getFirstChild(); ch != null; ch = ch.getNextSibling()) {
        if (ch.getNodeType() == Node.ELEMENT_NODE) {
            if (ch.getnodeName().equals("price")) {
                nextChild = ch.getNextSibling();
                n.removeChild(ch); // 부모 노드에서 price 엘리먼트 삭제
                ch = nextChild; // 다음 형제 노드로 이동
            } else {
                delPrice(ch); // 각 자식 엘리먼트에 대해 recursion 수행
                ch = ch.getNextSibling();
            }
        }
    }
}
```

ch = ch.getNextSibling()가
앞에서
앞의 코드가
들어감

70

DOM을 활용한 XML 문서 조작

- 참고: 특정 노드의 모든 자식 노드들의 삭제 방법

- NodeList나 NamedNodeMap에 속한 노드들은 “live” 데이터

✓ DOM 트리 구조나 특정 노드 변경 시 변화된 상태가 반영됨

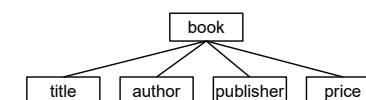
▪ 잘못된 방법

```
for (Node ch = n.getFirstChild(); ch != null;
     ch = ch.getNextSibling()) {
    n.removeChild(ch);
}
```

결과? X
removeChild()하면
getNextSibling() 불가!

```
NodeList nodeList = node.getChildNodes();
for (int i = 0; i < nodeList.getLength(); i++) {
    node.removeChild(nodeList.item(i));
}
```

결과?



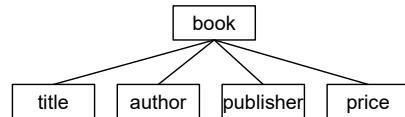
DOM을 활용한 XML 문서 조작

- 올바른 방법

```
while (node.hasChildNodes()) {
    node.removeChild(node.getFirstChild());
}
```

또는

```
NodeList nodeList = node.getChildNodes();
while (nodeList.getLength() > 0) {
    node.removeChild(nodeList.item(0));
}
```



72

DOM을 활용한 XML 문서 조작

- 특정 엘리먼트(<title>)들을 추출하여 새로운 XML 문서를 생성하는 예

```
... // DOM parser 생성

Document doc = builder.parse("ch9/bml.xml");           // XML 문서 parsing

Document newDoc = builder.newDocument();               // 새로운 문서 객체 생성
Element root = newDoc.createElement("bookTitles");    // 루트 엘리먼트 생성
newDoc.appendChild(root);                            // 추가

// 문서에 포함된 모든 title 엘리먼트들을 찾음
NodeList lst = doc.getElementsByTagName("title");        // > 이미 우선 탐색
for (int i = 0; i < lst.getLength(); i++) {
    Element title = newDoc.createElement("title");
    Node txtNode = lst.item(i).getFirstChild();
    Text txt = newDoc.createTextNode(txtNode.getNodeValue());
    title.appendChild(txt);
    root.appendChild(title);
}
...
```

Node title = newDoc.importNode(lst.item(i), true); 와 동일

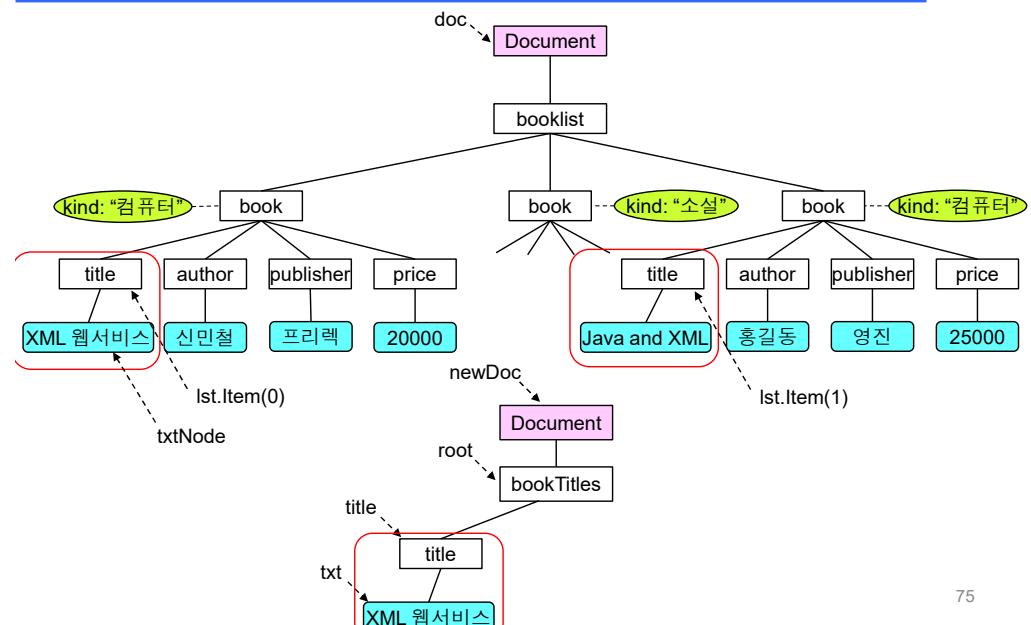
DOM을 활용한 XML 문서 조작

데이터 추출

- 기존의 XML 문서에서 특정 노드를 찾음
 - Element, 속성, Text 등
- Element 관련 메소드
 - Document 인터페이스
 - NodeList getElementsByTagName(String tagName)
 - Element 인터페이스
 - NodeList getElementsByTagName(String tagName)
 - Node 인터페이스
 - NodeList getChildNodes() → Element 인터페이스에 상속됨
 - NodeList 인터페이스
 - int getLength()
 - Node item(int index)

73

DOM을 활용한 XML 문서 조작



75

DOM을 활용한 XML 문서 조작

④ DOM 객체를 파일로 저장

- `javax.xml.transform.Transformer` 주상 클래스를 상속한 XSLT 변환기를 이용하여 XML 문서를 파일로 저장 (`SaveDOMToFile.java` 참조)

```
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import java.io.*;

// 변환기 생성
TransformerFactory tf = TransformerFactory.newInstance();
Transformer transformer = tf.newTransformer();

// XML 선언과 DTD 선언 설정
transformer.setOutputProperty(OutputKeys.ENCODING, "euc-kr");
transformer.setOutputProperty(OutputKeys.DOCTYPE_SYSTEM, "bml.dtd");
transformer.setOutputProperty(OutputKeys.INDENT, "yes");

DOMSource source = new DOMSource(document);           // DOMSource 객체 생성
StreamResult result = new StreamResult(new File("new.xml")); // StreamResult 객체 생성
transformer.transform(source, result);                // 파일로 저장
```

DOM을 활용한 XML 문서 조작

□ 엘리먼트의 속성 처리

- 속성 관련 메소드
 - `Node` 인터페이스
 - ✓ `NamedNodeMap getAttributes()` → `Element` 인터페이스에 상속됨
 - `NamedNodeMap` 인터페이스
 - ✓ `Node getNamedItem(String name)` → 특정 이름으로 노드를 찾음
 - ✓ `int getLength()`
 - ✓ `Node item(int index)` } 모든 속성들에 대한 처리 시 이용
 - `Element` 인터페이스
 - ✓ `String getAttribute(String name)`
 - ✓ `Attr getAttributeNode(String name)`
 - ✓ `void setAttribute(String name, String value)`
 - ✓ `Attr setAttributeNode(Attr newAttrNode)`
 - ✓ `void removeAttribute(String name);`
 - ✓ `Attr removeAttributeNode(Attr oldAttrNode)`

77

DOM을 활용한 XML 문서 조작

- 특정 이름("book")의 엘리먼트들에 속한 특정 이름("isbn")의 속성들을 찾아 새로운 문서에 엘리먼트(<isbn>)로 추가하는 예

```
... // DOM parser 생성...

NodeList lst = doc.getElementsByTagName("book");
for (int i = 0; i < lst.getLength(); i++) {
    Node n = lst.item(i);
    NamedNodeMap attrlist = n.getAttributes();
    Node attr = attrlist.getNamedItem("isbn");
    // 또는 Attr attr = ((Element)n.getAttributeNode("isbn"));

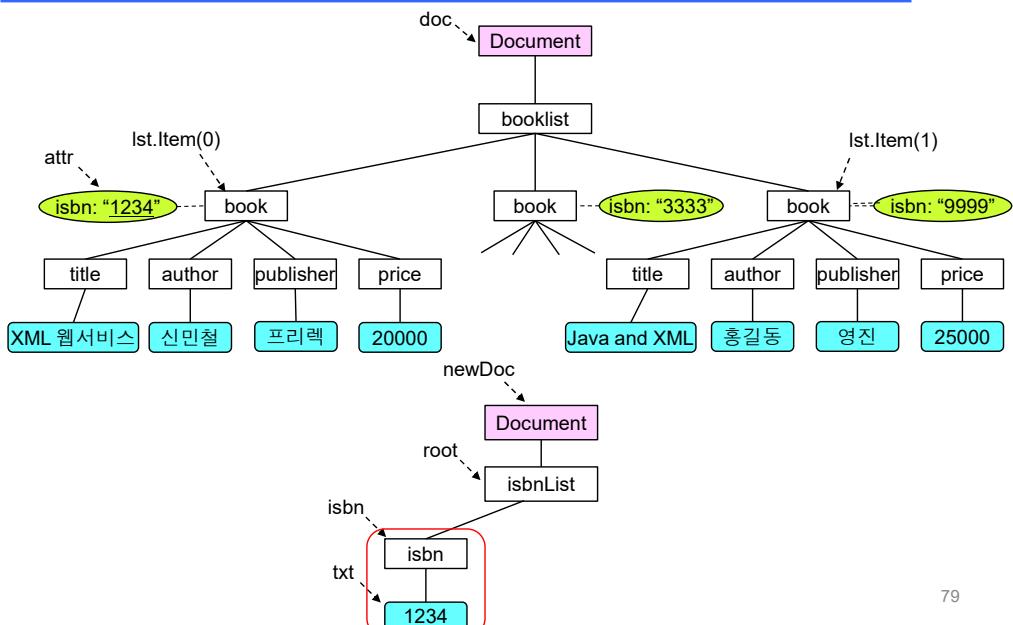
    String isbnValue = attr.getNodeValue();

    Element isbn = newDoc.createElement("isbn");
    Text txt = newDoc.createTextNode(isbnValue);
    isbn.appendChild(txt);
    root.appendChild(isbn);
}
```

...
78

`String isbnValue = ((Element)n.getAttribute("isbn"))
과 동일`

DOM을 활용한 XML 문서 조작



References

- ❑ W3C Document Object Model
 - <http://www.w3.org/DOM/>
- ❑ DOM Level 3 Core Specification
 - <http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/>
- ❑ Java API for XML Processing (JAXP)
 - http://en.wikipedia.org/wiki/Java_API_for_XML_Processing
 - <http://docs.oracle.com/javase/8/docs/technotes/guides/xml/jaxp/index.html>
 - <https://www.oracle.com/java/technologies/jaxp-introduction.html>
 - <http://docs.oracle.com/javase/8/docs/api/> (javadoc)