

INTEGRATING XML WITH JAVA

SAX – JAXP – JDOM

1

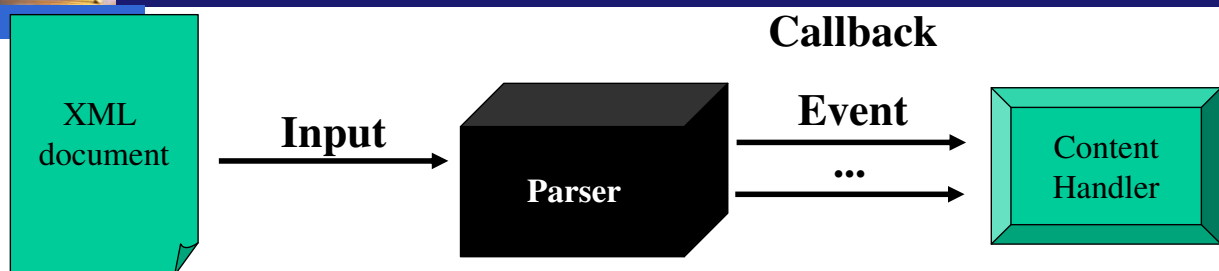
Objectives

- SAX
- DOM
- JAXP
- JDOM
- Chuyển đổi giữa SAX, DOM, JAXP và JDOM

SIMPLE API for XML – SAX

- ❑ Cung cấp các phương thức API để parse tài liệu XML
- ❑ Được sử dụng phía server hỗ trợ việc kết nối/ communication với client.
- ❑ Dùng để parse XML theo cơ chế đọc từng ký tự một tuần tự (từ trên xuống dưới, từ trái qua phải)
- ❑ Kích hoạt các biến cố (event) tương ứng với các thành phần được parsing, như
 - Element open/close tags
 - #PCDATA hay CDATA
 - Comments, entity, ...
- ❑ SAX 2.0 gồm 02 gói chuẩn và 01 gói mở rộng
 - **org.xml.sax** (cung cấp lõi của API SAX)
 - **org.xml.sax.helpers** (hỗ trợ các SAX driver)
 - **org.xml.sax.ext** (gói mở rộng hỗ trợ cơ chế bắt lỗi trong SAX)

SAX (tt)



- ❑ Ưu điểm
 - Không sử dụng nhiều bộ nhớ.
 - Xử lý nhanh
- ❑ Nhược điểm
 - Từng Element được xử lý trong cùng biến cố
 - Dữ liệu lưu tạm trong quá trình xử lý
 - Chỉ duyệt được tài liệu một lần
 - Không hỗ trợ DTD
 - Không cho phép truy cập ngẫu nhiên
 - Thông tin từ vệt không có sẵn



CONTENT HANDLER

- ❑ Interface cung cấp các biến cố parsing cho SAX (thư viện org.xml.sax.ContentHandler)
- ❑ Ứng dụng sử dụng parse XML phải implement ContentHandler Interface và đăng ký instance lưu trữ các thông tin xử lý
- ❑ Ứng dụng dùng instance để thể hiện kết quả xử lý.
- ❑ Các phương thức bắt buộc implement trong Content Handler

Method	Mô tả
void startDocument()	Kích hoạt khi đọc đầu văn bản
void endDocument()	Kích hoạt khi kết thúc văn bản
void startElement(String uri, String name, String qname, Attributes attr)	Kích hoạt khi bắt đầu đọc element
void endElement(String uri, String name, String qname)	Kích hoạt khi kết thúc element



CONTENT HANDLER (tt)

Method	Mô tả
characters(char ch[], int start, int length)	Kích hoạt khi đọc tuần tự từng ký tự
void ignorableWhitespace(char ch[], int start, int length)	Kích hoạt khi động ký tự khoảng trắng
void processingInstruction(String target, String data)	Kích hoạt khi xử lý các lệnh
void setDocumentLocator(Locator locator)	Kích hoạt xử lý có liên quan đến các đối tượng bên trong XML
void startPrefixMapping(String prefix, String uri)	Kích hoạt xử lý prefix của Namespace
void endPrefixMapping(String prefix)	Kích hoạt xử lý kết thúc Namespace
void skippedEntity(String name)	Kích hoạt loại bỏ Entity



CONTENT HANDLER (tt)

☐ Cơ chế thực hiện

- XMLReader (parser) đọc XML, parser yêu cầu phương thức startdocument()
- Parser đọc element start tag, parser yêu cầu phương thức startelement()
- Parser đọc các dữ liệu (text content), parser yêu cầu phương thức characters()
- Parser đọc element end tag, parser yêu cầu phương thức endelement()
- ...
- Parser đọc hết XML, parser yêu cầu phương thức enddocument()

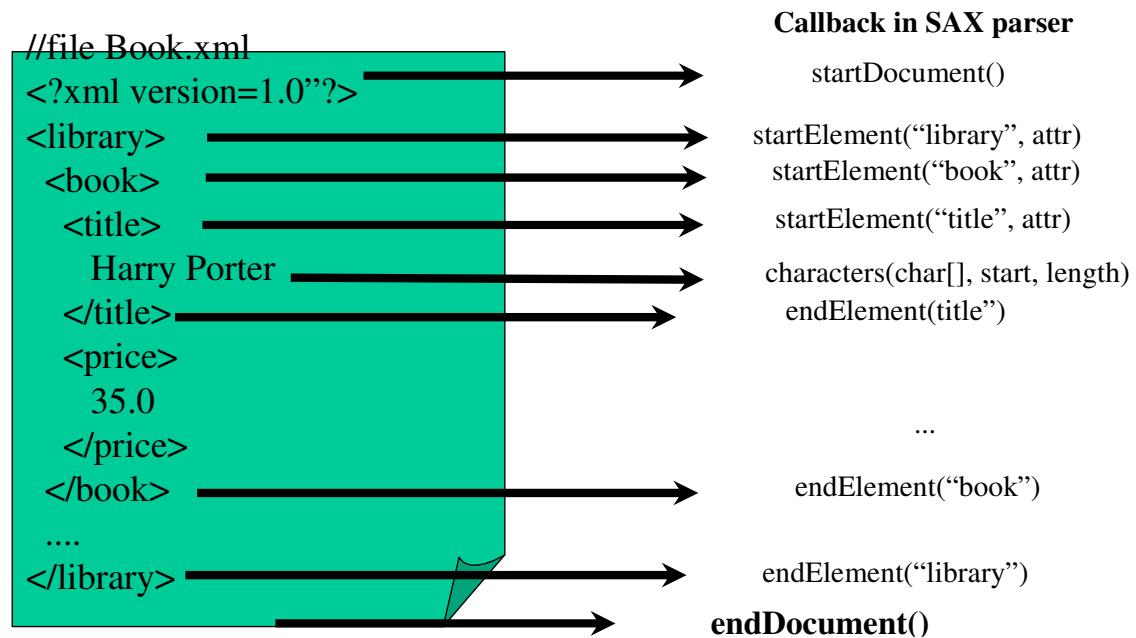


XML READER

- ☐ Thiết lập và truy vấn các thuộc tính của parser để đăng ký biến cố trong quá trình parsing và process XML
- ☐ Người sử dụng tạo instance parser qua factory, sau đó đưa tài liệu XML vào đối tượng InputSource để gọi phương parse() tài liệu XML
- ☐ Kết quả parse sẽ được đưa cho client
- ☐ Các Exception trong quá trình parse thông qua gói SAXException
- ☐ Đối tượng chính của SAX để đón nhận XML
- ☐ Một XML Reader gắn liền với một ContentHandler

CALLBACK INTERFACE

❑ SAX sử dụng XMLReader như là Subject và org.xml.sax.ContentHandler như là Observer (tương tự cách xác định các biến cố Listener cho subject – button trong AWT)



CÁC BƯỚC THỰC HIỆN

❑ **Step 1:** Tạo content handler (Default handler) để xử lý các element trong XML

- Tạo class (public) extends DefaultHandler hay implements ContentHandler
- Implement các phương thức callback điển hình
 - startDocument, endDocument
 - startElement(String uriNamespace, String local, String qualifiedname, Attributes attribute), endElement
 - characters (char[] chars, int startIndex, int length)
 - ignoreableWhitespace(optional)

Ví dụ

```
<exp:book xmlns:cwp="http://www.abc.com/xml">
  <exp:title id="123">Harry Porter</exp:title>
  ...
</exp:book>
```

Annotations: Local, URI namespace, Qualified Name, Attribute

Lưu ý: đối với ContentHandler cần vận dụng đủ 11 phương thức callback



CÁC BƯỚC THỰC HIỆN (tt)

❑ **Step 2:** Thực hiện parser XML và content handler đã được implement

- **Cách 1:** Sử dụng parser factory để tạo instance

```
SAXParserFactory factory = SAXParserFactory.newSAXParser();
```

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

- **Cách 2:** Sử dụng XMLReader để tạo instance của parser factory từ XMLReaderFactory và thiết lập ContentHandler cho parser

```
XMLReader parser = XMLReaderFactory.createXMLReader(  
    "org.apache.xerces.parsers.SAXParser");
```

```
parser.setContentHandler(contentHandler);
```

- Gọi phương thức parse với file input và handler

```
parser.parse(filename, handler);
```

- Bắt các lỗi ParserConfigurationException và FactoryConfigurationException



VÍ DỤ – CONTENTHANDLER

```
1. package saxparser;  
2. import org.xml.sax.*;  
3. import org.xml.sax.helpers.*;  
4. public class CountElement implements ContentHandler {  
5.     int count = 0;  
6.     public static void main(String[] args) {  
7.         System.out.println("main");  
8.         try{  
9.             if (args.length != 1) {  
10.                 System.err.println("Usage: java CountElement [filename]");  
11.                 System.exit(1);  
12.             }  
13.             XMLReader parser = XMLReaderFactory.createXMLReader(  
                "org.apache.xerces.parsers.SAXParser");
```



VÍ DỤ – CONTENTHANDLER (tt)

```
14.     CountElement cdC = new CountElement();
15.     parser.setContentHandler(cdC);
16.     parser.parse(args[0]);
17.     } catch(Exception e){
18.         e.printStackTrace();
19.     }
20.
21. }
22. public void setDocumentLocator(Locator locator) {}
23. public void startDocument() throws SAXException {}
24. public void endDocument() throws SAXException {
25.     System.out.println("Total number of elements including their multiple
                           occurrences are :" + count);
26. }
27.
28. public void startPrefixMapping(String string, String string0) throws SAXException {
29. }
30.
31. public void endPrefixMapping(String string) throws SAXException {}
```



VÍ DỤ – CONTENTHANDLER (tt)

```
32.     public void startElement(String uri, String localName, String qName
                                , Attributes attributes) throws SAXException {
33.         System.out.println("Element name is " + qName);
34.         count++;
35.     }
36.
37.     public void endElement(String string, String string0, String string1)
                                throws SAXException {
38.     }
39.
40.     public void characters(char[] c, int i, int i0) throws SAXException { }
41.
42.     public void ignorableWhitespace(char[] c, int i, int i0) throws SAXException { }
43.
44.     public void processingInstruction(String string, String string0) throws
                                SAXException { }
45.
46.     public void skippedEntity(String string) throws SAXException { }
47. }
```



VÍ DỤ – DEFAULTHANDLER & XMLREADER

```
1. package saxparser;
2. import org.xml.sax.*;
3. import org.xml.sax.helpers.*;
4. public class Notification extends DefaultHandler {
5.     public void startElement(String uri, String localName, String qName,
                               Attributes attributes) {
6.         System.out.println("Start Element: " + localName);
7.         for(int ctr = 0; ctr < attributes.getLength(); ctr++) {
8.             String attname = attributes.getQName(ctr);
9.             String type = attributes.getType(ctr);
10.            String value = attributes.getValue(ctr);
11.            System.out.println("attribute name = " + attname + " type = " + type
                               + " value = " + value);
12.        }
13.    }
14.    public void characters(char[] ch, int start, int length) {
15.        System.out.println(new String(ch, start, length));
16.    }
17.    public void endElement(String uri, String localName, String qName) {
18.        System.out.println("End element: " + localName);
19.    }
```



VÍ DỤ – DEFAULTHANDLER & XMLREADER (tt)

```
20. public static void main(String[] args) {
21.     try{
22.         XMLReader parser = XMLReaderFactory.createXMLReader(
                               "org.apache.xerces.parsers.SAXParser");
23.         Notification notify = new Notification();
24.         parser.setContentHandler(notify);
25.         parser.parse(args[0]);
26.     } catch(Exception e) {
27.         e.printStackTrace();
28.     }
29. }
30. }
```




VÍ DỤ – DEFAULTHANDLER & FACTORY

```
1. package saxparser;

2. import java.io.*;
3. import org.xml.sax.*;
4. import javax.xml.parsers.*;
5. import org.xml.sax.helpers.DefaultHandler;
6. public class CountElementFactory extends DefaultHandler {
7.     int count = 0;
8.     public void startElement(String uri, String localName, String qName
9.                             , Attributes attributes) throws SAXException {
10.         System.out.println("Element name is " + qName);
11.         count++;
12.     }
13.     public void endElement(String string, String string0, String string1
14.                           throws SAXException {
15.         System.out.println("Total number of elements including their multiple
16.                           occurrences are " + count);
17.     }
```



VÍ DỤ – DEFAULTHANDLER & FACTORY (tt)

```
15. public static void main(String[] args) {
16.     SAXParserFactory factory = SAXParserFactory.newInstance();
17.     SAXParser parser;
18.     try {
19.         parser = factory.newSAXParser();
20.         DefaultHandler handler = new CountElementFactory();
21.         parser.parse(new File(args[0]), handler);
22.     }
23.     catch (ParserConfigurationException ex) {
24.         ex.printStackTrace();
25.     }
26.     catch (SAXException ex) {
27.         ex.printStackTrace();
28.     }
29.     catch (IOException ex) {
30.         ex.printStackTrace();
31.     }
```



ERROR HANDLER

- ❑ **Hỗ trợ tìm kiếm các lỗi trong quá trình parsing XML**
- ❑ extends ErrorHandler (gói org.apache.xml.utils của thư viện xalan-2.4.1.jar)
- ❑ Các phương thức callback cần vận dụng là error(SAXParseException e) để kiểm tra các lỗi
 - Lỗi không nghiêm trọng, không đúng định dạng
 - Lỗi phát sinh khi validity XML
- ❑ **Các phương thức hỗ trợ lấy lỗi**
 - getColumnNumber
 - getLineNumber
 - getSystemID để xác định uri của file
 - getMessage xác định lỗi
- ❑ **Thiết lập các error cho parser**
 - Sử dụng phương thức setErrorHandler
 - Bật chế độ validation = true với phương thức
 - Đối với XML Reader
 - parser.setFeature("http://xml.org/sax/features/validation", true)**
 - Đối với SAX factory
 - Factory.validation(true)**

<http://xml.apache.org/xalan-j/downloads.html>



VÍ DỤ

book.xml

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

<?xml-stylesheet type="text/xsl" href="Books_format.xsl" ?>
<!DOCTYPE library[
<!ELEMENT library (book)*>
<!ELEMENT book (booktitle, author, price)+>
<!ELEMENT booktitle (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
]>
<library>
  <book><booktitle>The Firm</booktitle><author>John Grisham</author>
    <price>99</price></book>
  <book><author>Robin Cook</author><booktitle>Coma</booktitle>
    <price>99.5</price> </book>
</library>
```




SAXError.java

```
1. package saxparser;

2. import org.apache.xml.utils.DefaultErrorHandler;
3. import org.xml.sax.*;
4. import org.xml.sax.helpers.*;

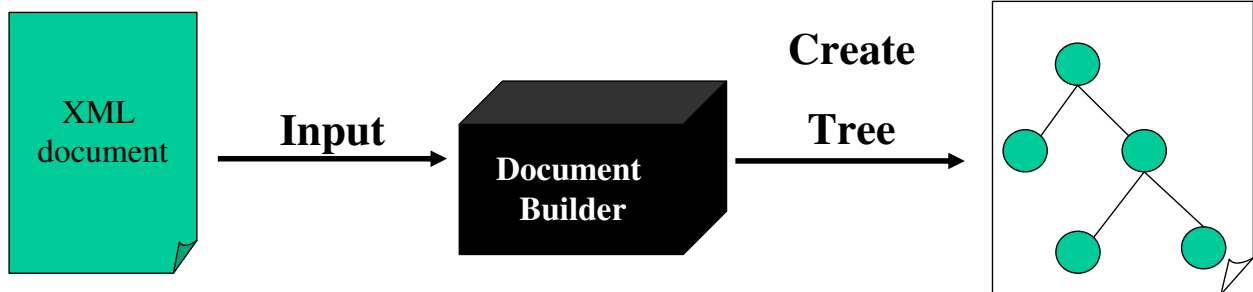
5. public class SAXError extends DefaultErrorHandler {
6.
7.     public void error(SAXParseException exception) throws SAXException {
8.         System.out.println("Parsing error: \nLine: " + exception.getLineNumber()
9.         + "\nColumn: " + exception.getColumnNumber()
10.        + "\nURI: " + exception.getSystemId()
11.        + "\n Message: " + exception.getMessage());
12.    }
```



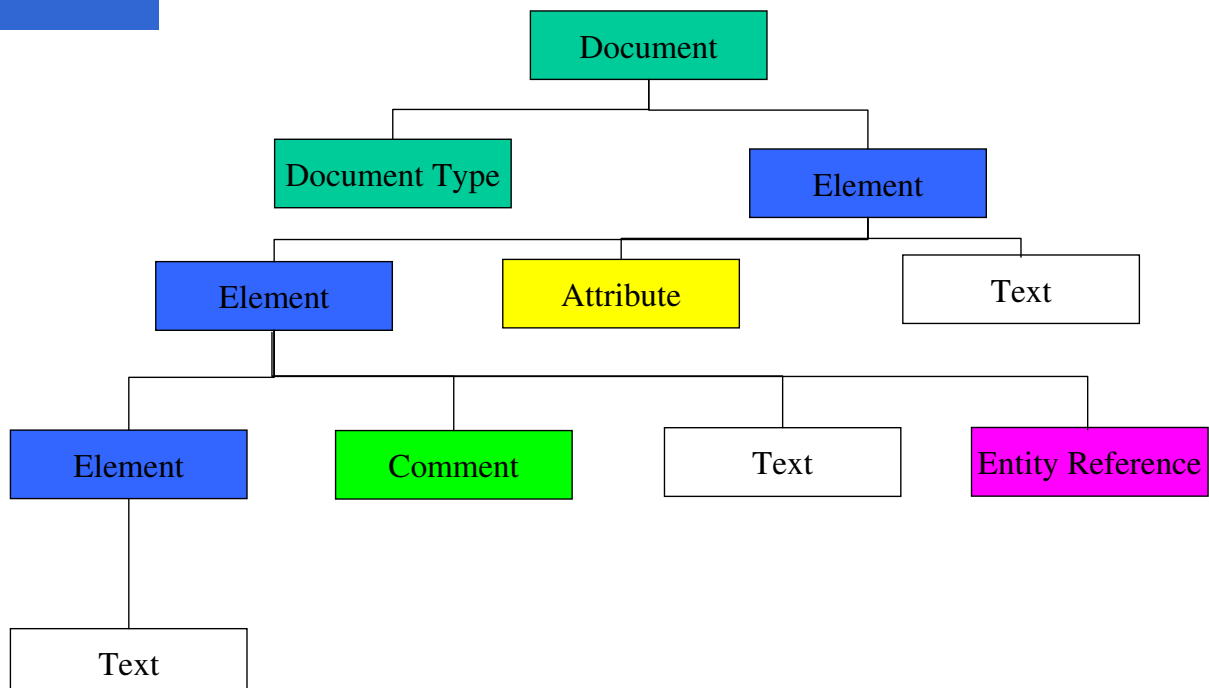
```
13. public static void main(String[] args) {
14.     DefaultHandler contentHandler = new DefaultHandler();
15.     ErrorHandler errorHandler = new SAXError();
16.     try {
17.         XMLReader parser = XMLReaderFactory.createXMLReader(
18.             "org.apache.xerces.parsers.SAXParser");
19.         parser.setContentHandler(contentHandler);
20.         parser.setErrorHandler(errorHandler);
21.         parser.setFeature("http://xml.org/sax/features/validation", true);
22.         parser.parse(args[0]);
23.     } catch (Exception ex) {
24.         ex.printStackTrace();
25.     }
26. }
```

DOCUMENT OBJECT MODEL – DOM

- ☐ Parse XML sử dụng cấu trúc cây với các node element và node text
- ☐ Có thể duyệt cây theo 02 hướng
- ☐ Đòi hỏi bộ nhớ lớn
- ☐ Gói hỗ trợ org.w3c.dom.*
- ☐ Cơ chế thực hiện:
 - Ứng dụng cung cấp XML để DOM builder parsing
 - Builder trả về DOM Document



DOM TREE





CÁC THÀNH PHẦN CỦA DOM

- ☐ DocumentBuilderFactory: tạo instance builder factory
- ☐ DocumentBuilder: xác định instance của XML parser
- ☐ Phương thức parse chuyển thành DOM document
- ☐ Node: thành phần của DOM tree với các phương thức
 - **getFirstChild**: lấy node con đầu tiên
 - **getNextSibling**: lấy node con ngang cấp
 - **getNodeName**: lấy tên node



DOM (tt)

- ☐ Khuyết điểm của DOM
 - Chiếm bộ nhớ vô cùng lớn
 - Độ phức tạp khá lớn
 - DOM chỉ được tạo khi duyệt hết văn bản
- ☐ Cách thức thực hiện
 - Đưa tài liệu qua DOM Parser
 - Đón nhận DOM Document trả về
 - Thực hiện duyệt và in node ra màn hình
- ☐ Ví dụ: in ra màn hình cây DOM Tree.



VÍ DỤ PrintDOMTree

```
1. package domparser;

2. import org.apache.xerces.parsers.DOMParser;
3. import org.w3c.dom.Document;
4. import org.w3c.dom.Node;
5. import org.xml.sax.InputSource;

6. public class PrintDOMTree {
7.     public static void main(String[] args) {
8.         try{
9.             InputSource inputSource = new InputSource(args[0]);
10.            DOMParser parser = new DOMParser();
11.            parser.parse(inputSource);
12.            Document doc = parser.getDocument();
13.            printNode(doc, "");
14.        } catch(Exception e) {
15.            e.printStackTrace();
16.        }
17.    }
```



VÍ DỤ PrintDOMTree (tt)

```
18. private static void printNode(Node node, String indent) {
19.     System.out.println(indent + "<" + node.getNodeName() + ">");
20.     Node child = node.getFirstChild();
21.     while (child!=null) {
22.         printNode(child, indent + "  ");
23.         child = child.getNextSibling();
24.     }
25.     System.out.println(indent + "<" + node.getNodeName() + ">");
26. }
27. }
```

PrintDOMTree - output

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <library>
3.     <book><booktitle>The Firm</booktitle>
4.         <author>John Grisham</author>
5.         <price>99</price></book>
6.     <book>
7.         <author>Robin Cook</author>
8.         <booktitle>Coma</booktitle>
9.         <price>99.5</price> </book>
10. </library>
```

```
1. <#document>
2. <library>
3. <#text>
4. <#text>
5. <book>
6.     <booktitle>
7.         <#text>
8.         <#text>
9.     <booktitle>
10.    <#text>
11.    <#text>
12.    <author>
13.        <#text>
14.        <#text>
15.    <author>
16.        <#text>
17.        <#text>
18.    <price>
19.        <#text>
20.        <#text>
21.    <price>
22. <book>
```

JAVA API for XML – JAXP

- ☐ JAXP khắc phục các khuyết điểm của SAX và DOM bằng cách cho phép ứng dụng parse và transform XML độc lập với việc xử lý.
- ☐ Ngoài ra, JAXP tích hợp SAX, DOM và XSLT engine hỗ trợ chuyển đổi.
- ☐ Sử dụng các hàm API để chuyển đổi giữa các parser.

User Application

JAXP API

javax.xml.parsers.*

org.xml.sax.*

javax.xml.transform.*

org.w3c.dom.*

SỬ DỤNG JAXP

- ❑ Sử dụng với SAX: sử dụng cách 1 của SAX kết hợp với instance của SAXParser (phương thức newSAXParser).
- ❑ Sử dụng với DOM: thay thế SAX Parser thành Document Builder (kết hợp giữa java.xml.parsers với org.w3c.dom)

DocumentBuilderFactory

DBFactory=DocumentBuilderFactory.newInstance();

DocumentBuilder DBbuilder=DBFactory.newDocumentBuilder();

Document doc=DBbuilder.parse(tên file);

- ❑ Lựa chọn giữa SAX và DOM

Tài liệu lớn	Tài liệu nhỏ
Chỉ duyệt tài liệu 1 lần	Có thể duyệt tài liệu nhiều lần và có khả năng modify
Event Driven – chức năng giới hạn	Cần nhiều bộ nhớ để load toàn bộ XML

VÍ DỤ – JAXP với SAX

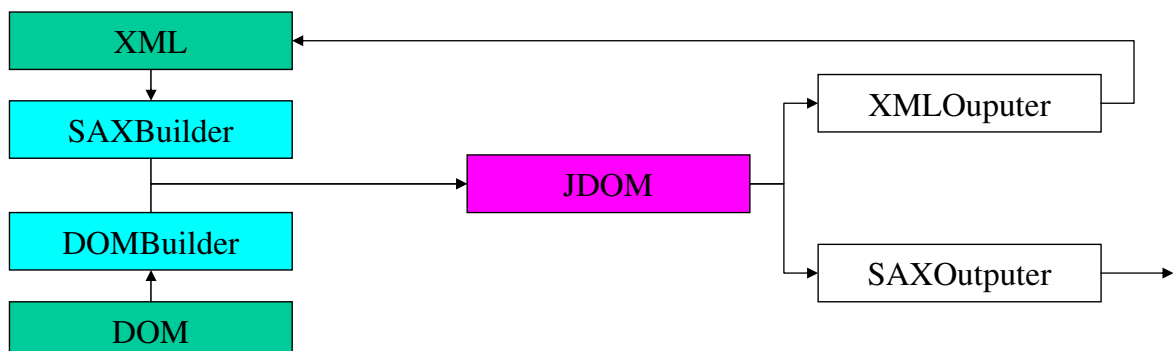
```
1. package domparser;
2. import java.io.File;
3. import javax.xml.parsers.*;
4. import org.xml.sax.helpers.DefaultHandler;
5. public class JAXPDocumentParsing {
6.     public static void main(String[] args) {
7.         try{
8.             SAXParserFactory factory = SAXParserFactory.newInstance();
9.             SAXParser parser = factory.newSAXParser();
10.            DefaultHandler handler = new MyHandler();
11.            parser.parse(new File(args[0]), handler);
12.        } catch(ParserConfigurationException e){
13.            System.out.println("The underlying parser does not support the requested features.");
14.        } catch(FactoryConfigurationError e){
15.            System.out.println("Error occurred obtaining SAX Parser Factory.");
16.        } catch(Exception e){e.printStackTrace();
17.        }
18.    }
19. }
20. class MyHandler extends DefaultHandler {
21. }
```


JAXPDOMParsing.java

```
package domparser;
import java.io.File;
import javax.xml.parsers.*;
import org.w3c.dom.*;
public class JAXPDOMParsing {
    public static void main(String[] args) {
        try{
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.parse(new File(args[0]));
            printNode(doc, "");
        } catch(ParserConfigurationException e){
            System.out.println("The underlying parser does not support the requested feature.");
        } catch(FactoryConfigurationError e){
            System.out.println("Error occurred obtaining Document Builder Factory.");
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
    private static void printNode(Node node, String indent){
    }
}
```

JDOM

- ❑ Nguồn mở, Java API (tree) dùng để parsing, tạo, xử lý và lưu trữ cho XML nhằm khắc phục các nhược điểm của DOM (rõ ràng và đơn giản hơn)
- ❑ JDOM là sự kết hợp và khắc phục các nhược điểm của SAX cùng DOM
 - Truy cập node trên cây bất kỳ lúc nào(sử dụng phương thức API)
 - Các node được thể hiện thông qua class (không qua interface)
 - Dữ liệu nhập không nhất thiết là XML mà có thể là CSDL
 - Có khả năng output các tài liệu
- ❑ Gói hỗ trợ:org.jdom.* (thư viện: jdom.jar)





TẠO XML VỚI JDOM

- ❑ **Sử dụng gói org.jdom.output.XMLOutputter**
 - Hỗ trợ phương thức serialize và output (xuất file xml)
 - Hỗ trợ lưu trữ element, attribule, CDATA ...(các thành phần XML)
- ❑ **Chuyển đổi trực tiếp từ tập tin XML sẵn có**
 - Dùng SAXBuilder parsing tài liệu XML
 - Sử dụng XMLOutputter để chuyển đổi thành tài liệu XML
- ❑ **Tạo XML trong quá trình thực thi**
 - DataType: Element – tạo ra các element trong XML
 - Method
 - setAttribute(“thuộc tính”, giá trị): tạo thuộc tính
 - setText(“text”): tạo PCDATA
 - addContent(Element e): gắn element vào cấu trúc XML (doc)
 - Doctype.setInstanceSubset(String): các thành phần lọc của.dtd
 - ...
 - Sử dụng XMLOutput để chuyển đổi thành XML



VÍ DỤ – TRỰC TIẾP

```
1. package jdomparser;
2. import java.io.File; import java.io.FileWriter; import org.jdom.Document;
3. import org.jdom.input.SAXBuilder; import org.jdom.output.Format;
4. import org.jdom.output.XMLOutputter;
5. public class CreateXMLDoc {
6.     public static void main(String[] args) throws Exception {
7.         if(args.length < 2){
8.             System.out.println("Usage: java CreateXMLDOc file1 file2 ...");
9.             System.exit(1);
10.        }
11.        SAXBuilder builder = new SAXBuilder();
12.        Document doc = builder.build(new File(args[0]));
13.        Format format = Format.getPrettyFormat();
14.        XMLOutputter outputter = new XMLOutputter(format);
15.        outputter.output(doc, new FileWriter(args[1]));
16.        System.out.println("File created ..... " + args[1]);
17.    }
18. }
```



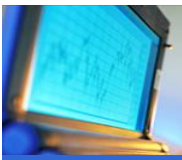
SỬ DỤNG JDOM ĐỂ XỬ LÝ

- ❑ DOMBuilder hỗ trợ JDOM tạo tree để JDOM có thể truy xuất và định hướng xuất
 - Ví dụ: đếm số từ có trong XML
- ❑ Kết hợp JDOM và SAX trong việc validation XML (tương tự parser của SAX) qua phương thức
 - setValidation(true) của SAXBuilder
 - setFeature(uri, true)



VÍ DỤ – XỬ LÝ KẾT QUẢ WordCounter

```
1. package jdomparser;
2. import java.io.*;    import java.util.*; import org.jdom.*;
3. import org.jdom.input.SAXBuilder;
4. public class WordCounter {
5.     public static void main(String[] args) {
6.         if (args.length == 0) {
7.             System.out.println("Usage: java CountWords file1 file2 ...");
8.         }
9.         SAXBuilder builder = new SAXBuilder();
10.        for (int ctr=0; ctr<args.length; ctr++){
11.            try {
12.                Document doc = builder.build(new File(args[ctr]));
13.                Element root = doc.getRootElement();
14.                int numWords = countWordsInElement(root);
15.                System.out.println(numWords + " words present in this XML document");
16.            } catch (JDOMException jex) {
17.                System.out.println(args[ctr] + " is not well formed.");
18.                System.out.println(jex.getMessage());
19.            } catch (IOException iex) {
20.                System.out.println(iex.getMessage());
21.            }
22.        } }
```



WordCounter (tt)

```
23. private static int countWordsInString(String str){
24.     if (str == null) {
25.         return 0;
26.     }
27.     str = str.trim();
28.     if (str.length() == 0) {
29.         return 0;
30.     }
31.     StringTokenizer st = new StringTokenizer(str);
32.
33.     return st.countTokens();
34. }
```



WordCounter (tt)

```
35. private static int countWordsInElement(Element element) {
36.     int numWords = 0, numWords1 = 0;
37.     List children = element.getContent();
38.     for(Iterator iterator = children.iterator(); iterator.hasNext();) {
39.         Object obj=iterator.next();
40.         if(obj instanceof String) {
41.             numWords += countWordsInString((String)obj);
42.             System.out.println("num words in String " + numWords);
43.         } else if(obj instanceof Element) {
44.             Element e = (Element)obj;
45.             numWords += countWordsInElement(e);
46.             numWords1 += countWordsInString(e.getText());
47.         }
48.     }
49.     return numWords+numWords1;
50. }
51. }
```



DOM to SAX

Sử dụng qua đối tượng trung gian

- Outputstream
- Transforms của JAXP (hỗ trợ cả DOM và SAX) với các thành phần chính là DOMSource, SAXResult
- Thư viện
 - import javax.xml.transform.*;
 - import javax.xml.transform.dom.DOMSource;
 - import javax.xml.transform.stream.StreamResult;
 - import javax.xml.transform.sax.SAXResult;
 - import javax.xml.transform.sax.SAXSource;
 - import javax.xml.parsers.*;
- Dùng phương thức transform để thực hiện chuyển đổi kết quả nhập và xuất



CÁC BƯỚC THỰC HIỆN

- **Step 1:** Tạo DefaultHandler cho SAX
- **Step 2:** Tạo DOM từ XML nhập

```
InputStream Isource=new InputStream(args[0]);
DOMParser Dparser=new DOMParser();
Dparser.parse(Isource);
Document doc=Dparser.getDocument();
```
- **Step 3:** Thực hiện chuyển đổi sử dụng SAXResult và Transformer

```
Source input=new DOMSource(doc);
ContentHandler contentHandler=new Notification();
Result output=new SAXResult(contentHandler);
TransformerFactory formFactory=TransformerFactory.newInstance();
Transformer idTransform = formFactory.newTransformer();
idTransform.transform(input, output);
```



VÍ DỤ DOMSAX

```
1. package dom2sax;

2. import javax.xml.transform.*;
3. import javax.xml.transform.dom.DOMSource;
4. import javax.xml.transform.sax.SAXResult;
5. import org.apache.xerces.parsers.DOMParser;
6. import org.w3c.dom.Document;
7. import org.xml.sax.ContentHandler;
8. import org.xml.sax.InputSource;
9. import org.xml.sax.helpers.DefaultHandler;

10. class Notification extends DefaultHandler {
11. }
```



DOMSAX (tt)

```
12. public class DOMSAX {
13.     public static void main(String[] args) {
14.         try {
15.             InputSource lsource = new InputSource(args[0]);
16.             DOMParser Dparser = new DOMParser();
17.             Dparser.parse(lsource);
18.             Document doc = Dparser.getDocument();
19.             Source input = new DOMSource(doc);
20.             ContentHandler contentHandler = new Notification();
21.             Result output = new SAXResult(contentHandler);
22.             TransformerFactory formFactory = TransformerFactory.newInstance();
23.             Transformer idTransform = formFactory.newTransformer();
24.             idTransform.transform(input, output);
25.         } catch (Exception e) {
26.             e.printStackTrace();
27.         }
28.     }
29. }
```



DOM to JDOM

- ❑ Chuyển đổi từ DOM tree sang JDOM tree sử dụng gói org.jdom.input DOMBuilder của jdom
 - Xuất XML thành DOM và chuyển đổi thành JDOM

```
DOMBuilder builder = new DOBBuilder()
org.jdom.Document jdomDoc=builder.build(domDOC)
```
 - Sử dụng XMLOutputter và phương thức output để duyệt JDOM tree

```
XMLOutputter outputter = XMLOutputter();
outputter.output(jdomDoc, System.out);
```



DOM2JDOM

1. package dom2jdom;
2. import java.io.IOException;
3. import org.apache.xerces.parsers.DOMParser;
4. import org.jdom.JDOMException;
5. import org.jdom.input.DOMBuilder;
6. import org.jdom.output.XMLOutputter;
7. import org.w3c.dom.Document;
8. import org.xml.sax.InputSource;
9. public class DOM2JDOM {
10. Document domDoc;
11. public DOM2JDOM(String systemID)throws Exception{
12. DOMParser parser = new DOMParser();
13. parser.parse(new InputSource(systemID));
14. domDoc = parser.getDocument();
15. }



DOM2JDOM (tt)

```
16. public org.jdom.Document convert() throws JDOMException
    , IOException {
17.     DOMBuilder builder = new DOMBuilder();
18.     org.jdom.Document jdomDoc = builder.build(domDoc);
19.     return jdomDoc;
20. }
21. public static void main(String[] args) {
22.     try {
23.         DOM2JDOM tester = new DOM2JDOM(args[0]);
24.         org.jdom.Document jdomDoc = tester.convert();
25.         XMLOutputter outputter = new XMLOutputter();
26.         outputter.output(jdomDoc, System.out);
27.     } catch (Exception e) {
28.         e.printStackTrace();
29.     }
30. }
31. }
```



JDOM to SAX

- ❑ Chuyển đổi căn bản giữa JDOM document thành SAX ContentHandler sử dụng
 - [org.jdom.output.SAXOutputter](#)
 - phương thức output để duyệt các thành phần của XML
 - ❑ Cách thức thực hiện
 - Vận dụng SAX với DefaultHandle để parsing
 - Thực hiện chuyển đổi từ jdom sang SAX qua SAXOutputter
 - Duyệt thành phần của SAX sử dụng phương thức output
- ```
SAXBuilder buider=new SAXBuilder();
Document jdomDoc=buider.build(args[0]);
ContentHandler contentHandler=new Notification();
SAXOutputter converter=new SAXOutputter();
converter.setContentHandler(contentHandler);
converter.output(jdomDoc);
```





```
1. package jdom2sax;

2. import java.io.IOException;
3. import org.jdom.Document;
4. import org.jdom.JDOMException;
5. import org.jdom.input.SAXBuilder;
6. import org.jdom.output.SAXOutputter;
7. import org.xml.sax.ContentHandler;
8. import org.xml.sax.helpers.DefaultHandler;

9. class Notification extends DefaultHandler {
10. }
```



## JDOM2SAX

```
11. public class JDOM2SAX {
12. public static void main(String[] args) {
13. try{
14. SAXBuilder buider= new SAXBuilder();
15. Document jdomDoc = buider.build(args[0]);
16. ContentHandler contentHandler = new Notification();
17. SAXOutputter converter = new SAXOutputter();
18. converter.setContentHandler(contentHandler);
19. converter.output(jdomDoc);
20. } catch (JDOMException jex) {
21. System.out.println(args[0] + " not well form! " + jex.getMessage());
22. } catch (IOException ioex) {
23. System.out.println("IO Error: " + ioex.getMessage());
24. }
25. }
26. }
```



## JDOM to DOM

- ❑ Chuyển đổi căn bản giữa JDOM document thành DOM sử dụng

- `org.jdom.output.DOMOutputter`
- Phương thức `output` để chuyển đổi

- ❑ Các thực hiện

- Implement phương thức duyệt Node
- Thực hiện chuyển đổi từ JDOM sang DOM thông qua `SAXBuilder`

```
SAXBuilder builder=new SAXBuilder();
jdomDoc=builder.build(new FileInputStream(args[0]));
DOMOutputter converter=new DOMOutputter();
domDoc=converter.output(jdomDoc);
printNode(domDoc, "");
```



## JDOM2DOM

```
1. package jdom2dom;
2. import java.io.FileInputStream; import org.jdom.input.SAXBuilder;
3. import org.jdom.output.DOMOutputter; import org.w3c.dom.Node;
4. public class JDOM2DOM {
5. private static void printNode(Node Node, String indent) {
6. }
7. public static void main(String[] args) {
8. org.w3c.dom.Document domDoc;
9. org.jdom.Document jdomDoc;
10. try {
11. SAXBuilder builder = new SAXBuilder();
12. jdomDoc = builder.build(new FileInputStream(args[0]));
13. DOMOutputter converter = new DOMOutputter();
14. domDoc = converter.output(jdomDoc);
15. printNode(domDoc, "");
16. } catch(Exception e) {
17. e.printStackTrace();
18. }
19. }
20. }
```

# SAX2DOM

- ❑ Áp dụng cách ngược lại của DOM to SAX với DOMResult và SAXResult kết hợp XMLReader để tạo SAX và Transform để chuyển đổi

- **Xây dựng phương thức duyệt Node của DOM**

- **Tạo SAX với XML Reader**

```
XMLReader saxParser=XMLReaderFactory.createXMLReader();
```

```
Source input=new SAXSource(saxParser, new InputSource(args[0]));
```

- **Thực hiện chuyển đổi**

```
Result output=new DOMResult();
```

```
TransformerFactory formFactory=TransformerFactory.newInstance();
```

```
Transformer transformer=formFactory.newTransformer();
```

```
transformer.transform(input, output);
```

- **Thực hiện duyệt cây**

```
DOMResult rs=(DOMResult)output;
```

```
Node node=rs.getNode();
```

```
printNode(node, "");
```

# SAX2DOM

```
1. package sax2dom;
2. import javax.xml.transform.*; import javax.xml.transform.dom.DOMResult;
3. import javax.xml.transform.sax.SAXSource; import org.w3c.dom.Node;
4. import org.xml.sax.*; import org.xml.sax.helpers.XMLReaderFactory;
5. public class SAX2DOM {
6. private static void printNode(Node node, String indent) {}
7. public static void main(String[] args) {
8. try {
9. XMLReader saxParser = XMLReaderFactory.createXMLReader();
10. Source input = new SAXSource(saxParser, new InputSource(args[0]));
11. Result output = new DOMResult();
12. TransformerFactory formFactory = TransformerFactory.newInstance();
13. Transformer transformer = formFactory.newTransformer();
14. transformer.transform(input, output);
15. DOMResult rs = (DOMResult) output;
16. Node node = rs.getNode();
17. printNode(node, "");
18. } catch (Exception e) {
19. e.printStackTrace();
20. }
21. }
22. }
```



## SAX to JDOM

- ❑ JDOM được xây dựng trực tiếp từ SAX cho nên việc chuyển đổi từ SAX sang JDOM chỉ cần áp dụng phương thức build của SAXBuilder được hỗ trợ bởi gói org.jdom.input.SAXBuilder



## SAX2JDOM

```
1. package sax2jdom;
2. import java.io.*; import org.jdom.*;
3. import org.jdom.input.SAXBuilder; import org.jdom.output.XMLOutputter;
4. public class SAX2JDOM {
5. File file;
6. public SAX2JDOM(File file) { this.file=file; }
7. public Document convert() throws JDOMException, IOException {
8. SAXBuilder builder = new SAXBuilder();
9. Document doc = builder.build(new FileInputStream(file));
10. return doc;
11. }
12. public static void main(String[] args) {
13. try{
14. File file = new File(args[0]);
15. SAX2JDOM tester = new SAX2JDOM(file);
16. Document doc = tester.convert();
17. XMLOutputter outputter = new XMLOutputter();
18. outputter.output(doc, System.out);
19. } catch (Exception e) {
20. e.printStackTrace();
21. }
22. }
23. }
```



## Bài tập

- ☐ Tự thực hiện lại các bài demo
- ☐ Mở rộng các chức năng trong bài tập demo với các XML khác nhau