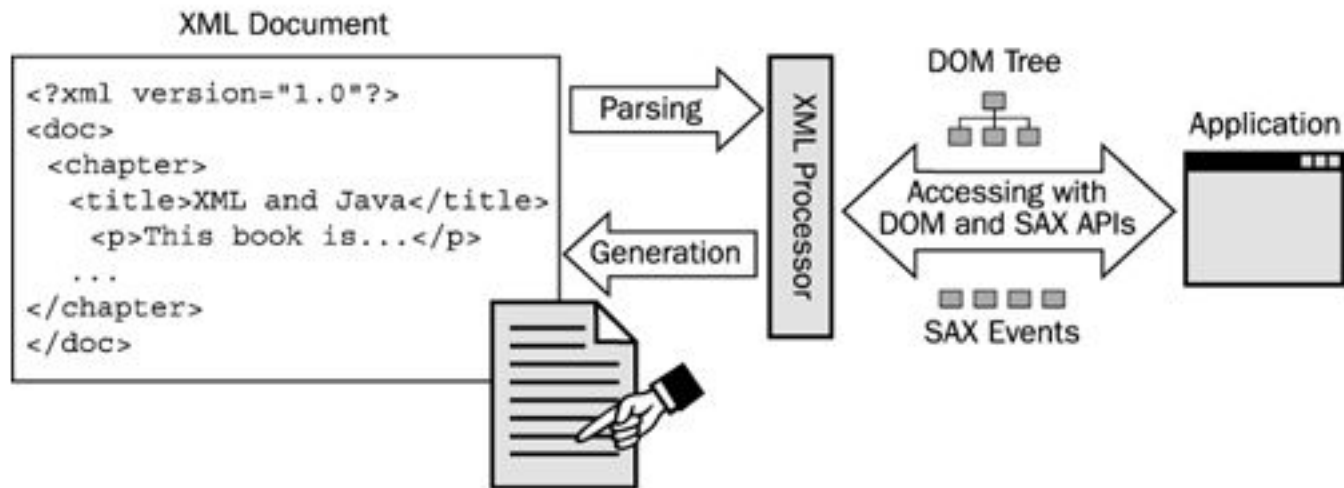




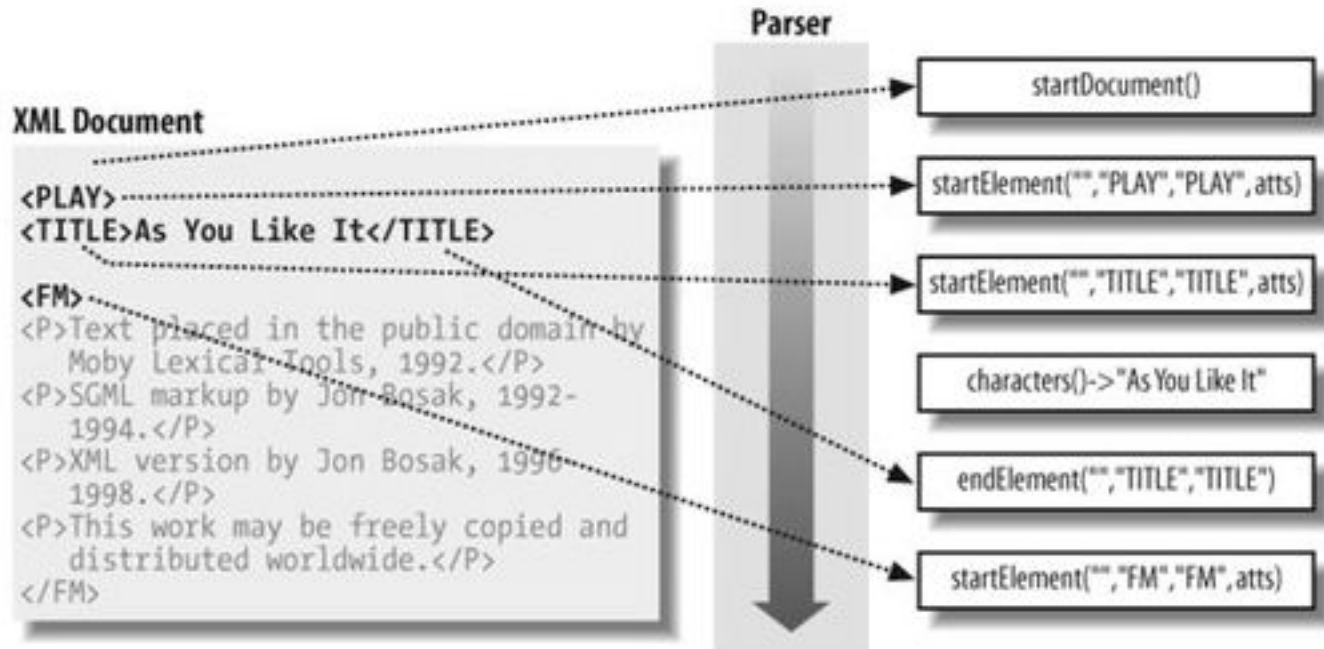
**Delivering Excellence in Software Engineering**



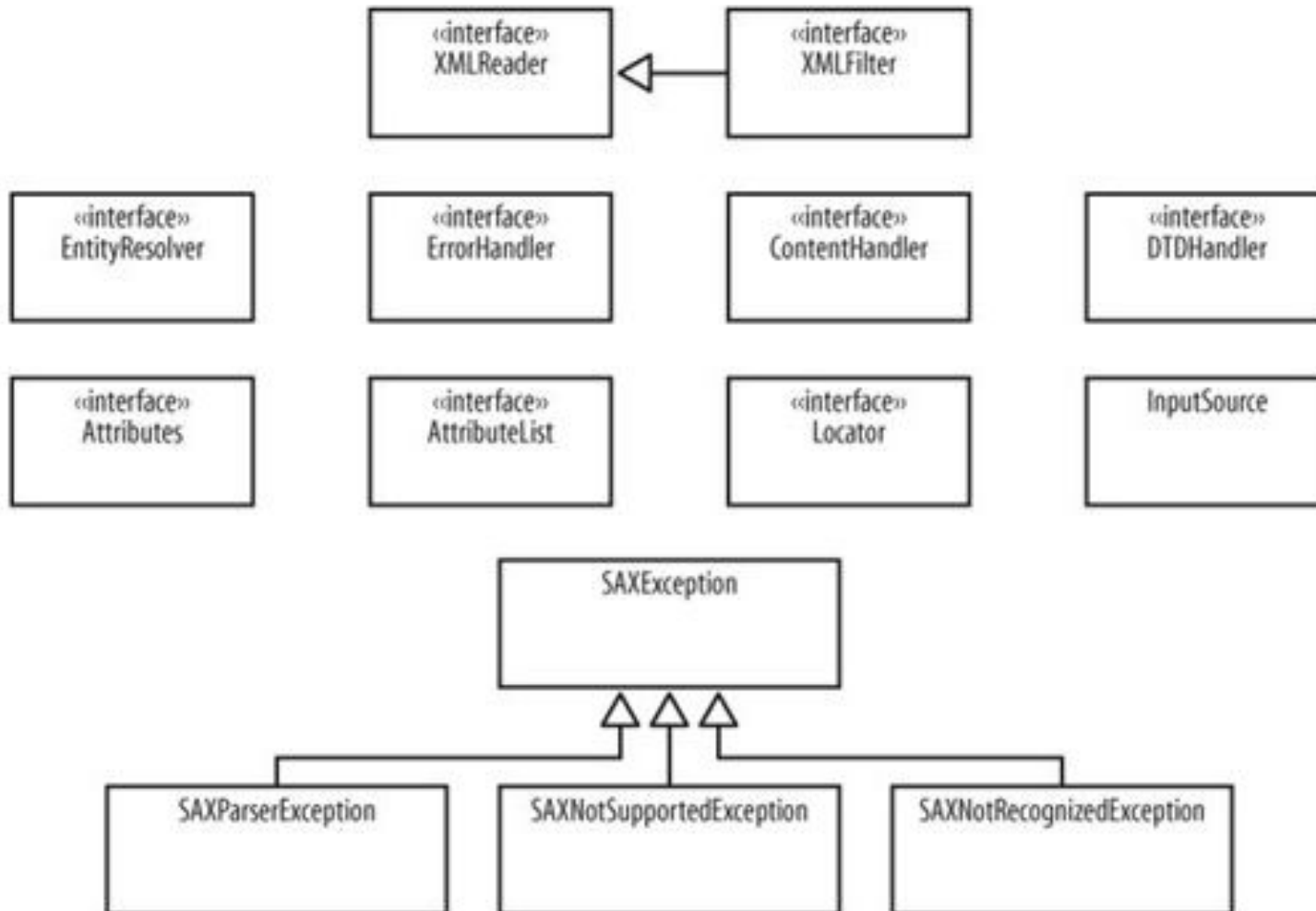
# XML парсеры



# SAX парсер (Simple API for XML)



# SAX API



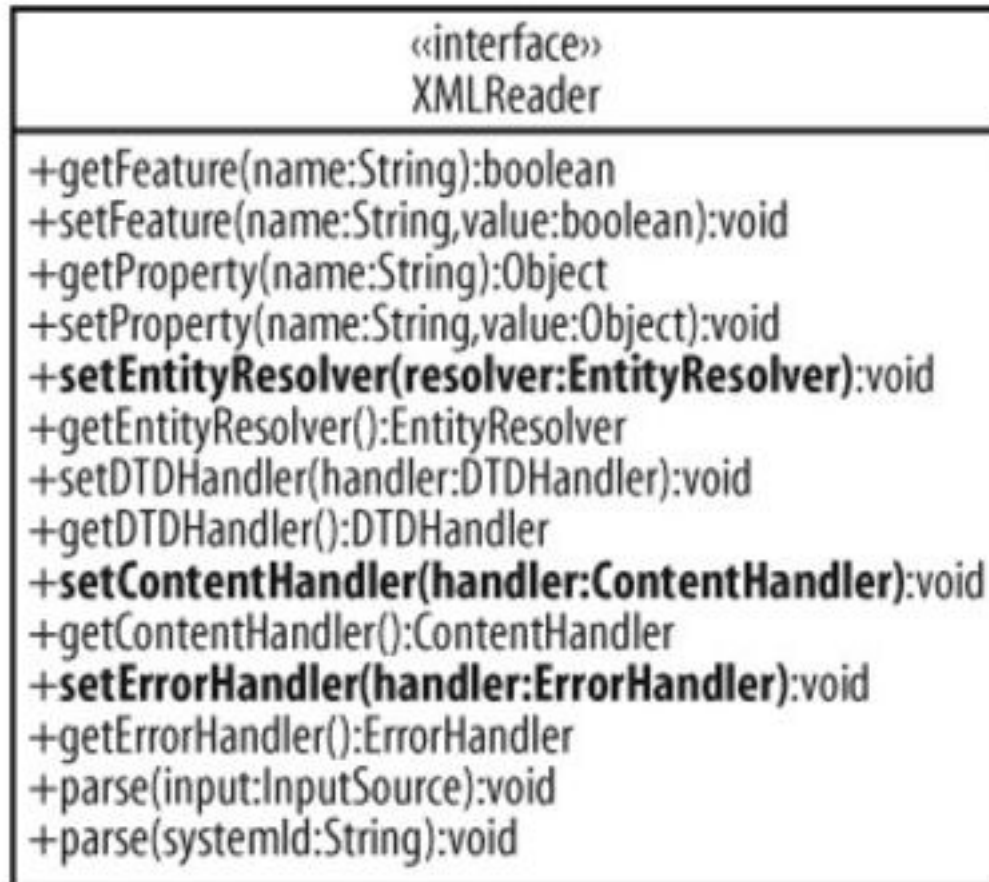
```
// Инстанцирование Reader
XMLReader reader = new org.apache.xerces.parsers.SAXParser( );

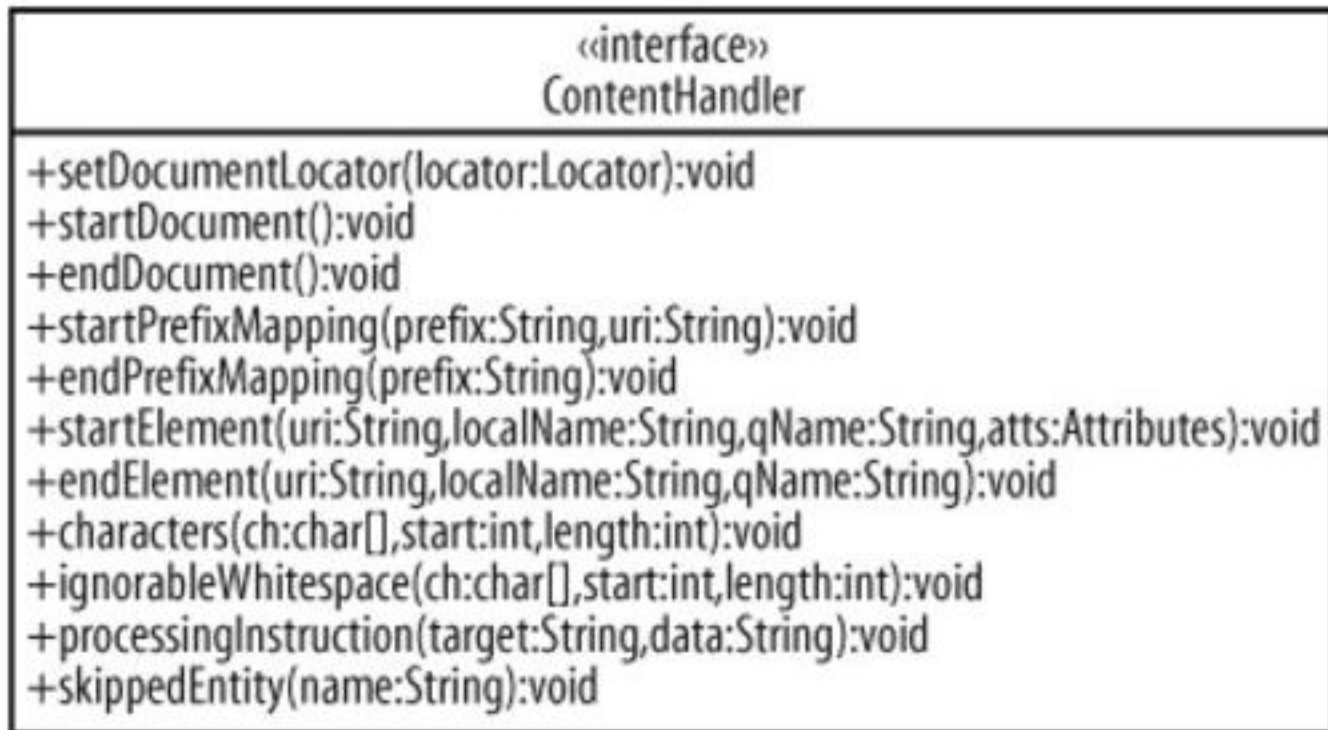
// Старт парсинг
reader.parse(uri);
```

## **Выбор другого вендора**

```
java -Dorg.xml.sax.driver=org.apache.xerces.parsers.SAXParser

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





```
public void setDocumentLocator(Locator locator) {  
    // Save this for later use  
    this.locator = locator;  
}
```

```
public void startDocument( ) throws SAXException {  
    // No visual events occur here  
}
```

```
public void endDocument( ) throws SAXException {  
    // No visual events occur here  
}
```



```
<catalog>
  <books>
    <book title="XML" xmlns:xlink="http://www.w3.org/1999/xlink">
      <cover xlink:type="simple" xlink:show="onLoad"
xlink:href="xmlnutCover.jpg" ALT="XML " width="125" height="350" />
    </book>
  </books>
</catalog>
```

```
public void startPrefixMapping(String prefix, String uri) {
    // No visual events occur here.
    namespaceMappings.put(uri, prefix);
}
public void endPrefixMapping(String prefix) {
    // No visual events occur here.
    for (Iterator i = namespaceMappings.keySet( ).iterator( );
        i.hasNext( ); ) {
        String uri = (String)i.next( );
        String thisPrefix = (String)namespaceMappings.get(uri);
        if (prefix.equals(thisPrefix)) {
            namespaceMappings.remove(uri); break;
        }
    }
}
```

**public void startElement(String namespaceURI, String  
localName, String qName, Attributes atts) throws  
SAXException**

**public void endElement(String namespaceURI, String  
localName, String qName) throws SAXException**

```
public void characters(char[] ch, int start, int  
length) throws SAXException{
```

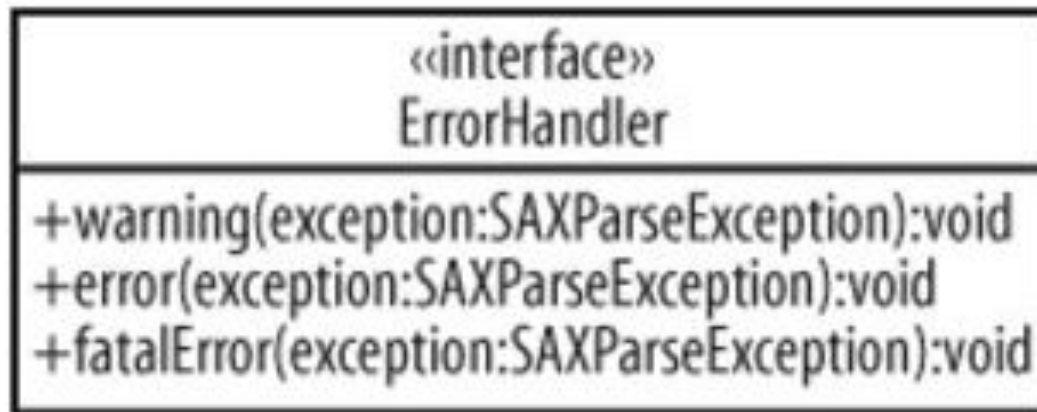
```
// Не правильно
```

```
    for (int i=0; i<ch.length; i++) {  
System.out.println(ch[i]);  
    }
```

```
// Правильно
```

```
    String data = new String(ch, start, length);  
}
```

```
// Создаем экземпляр для парсинга  
XMLReader reader = XMLReaderFactory.createXMLReader( );  
  
//Создаем ContentHandler  
ContentHandler myHandler = new MyHandler();  
  
//Регистрируем content handler  
reader.setContentHandler(myHandler);  
  
// Разбираем InputSource  
inputSource = new InputSource(xmlURI);  
reader.parse(inputSource);
```



class MyHandler implements ContentHandler, **ErrorHandler**

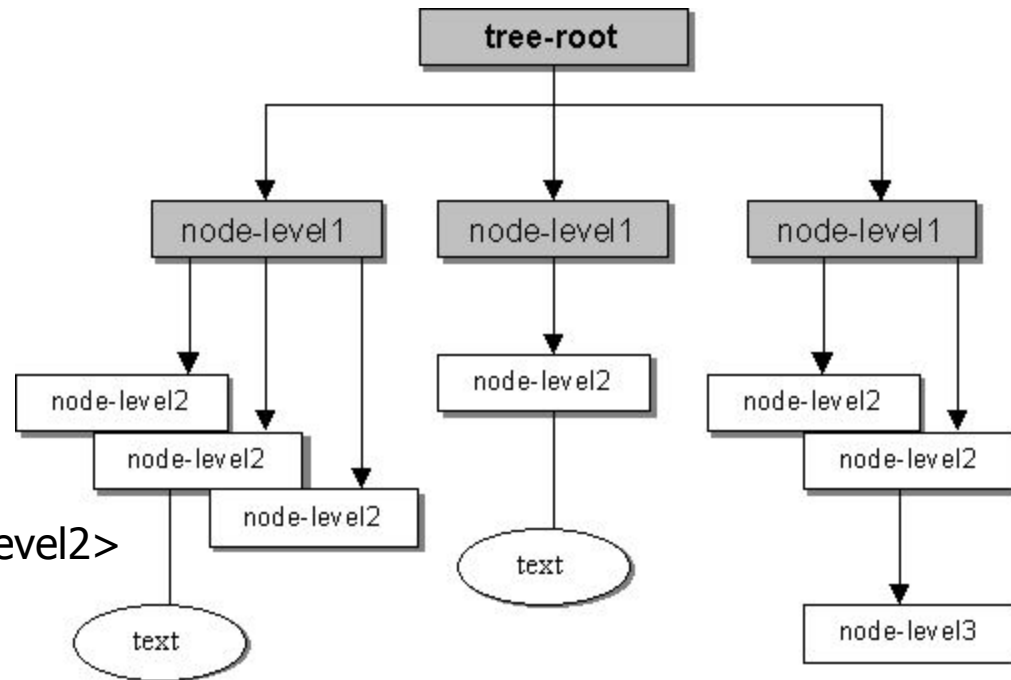
```
public void warning(SAXParseException exception) throws SAXException
{
    try {
        FileWriter fw = new FileWriter("error.log");
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("Warning: " + exception.getMessage( ) + "\n");
        bw.flush( );
        bw.close( );
        fw.close( );
    } catch (IOException e) {
        throw new SAXException("Could not write to log file", e);
    }
}
```

```
public void error(SAXParseException exception) throws  
SAXException
```

```
public void fatalError(SAXParseException exception) throws  
SAXException
```

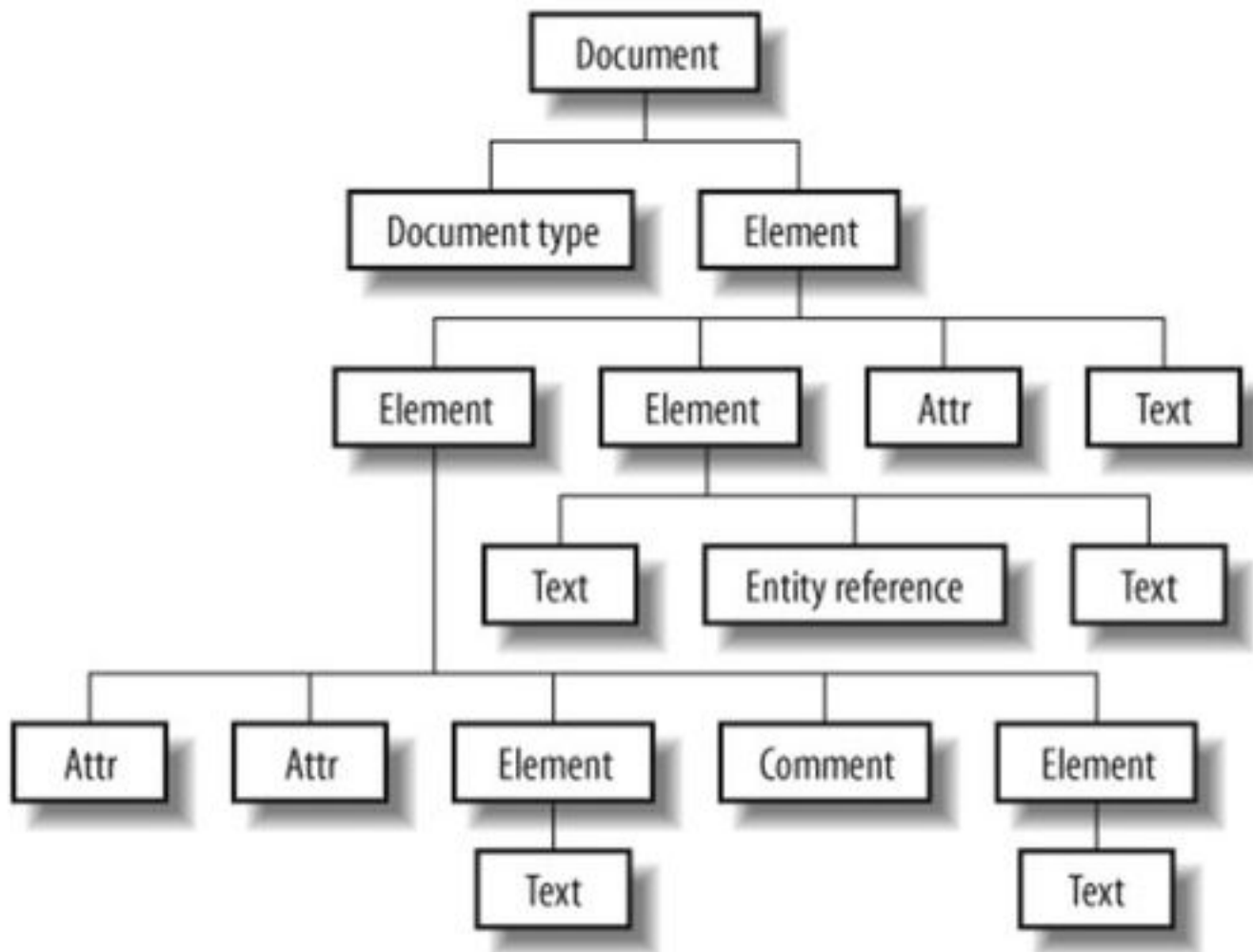
# DOM модель

```
<tree-node>  
  <node-level1>  
    <node-level2/>  
    <node-level2>text</node-level2>  
    <node-level2/>  
  </node-level1>  
<node-level1>  
  <node-level2>text</node-level2>  
<node-level1>  
  <node-level2/>  
  <node-level2><node-level3/></node-level2>  
</node-level1>  
</tree-node>
```





# Структура DOM



```
import org.apache.xerces.parsers.DOMParser;
...
public void test(OutputStream outputStream) throws Exception
{
    DOMParser parser = new DOMParser( );

    // Get the DOM tree as a Document object
    FileInputStream fis = new FileInputStream(inputXML);
    parser.parse(new InputSource(fis));
    Document doc = parser.getDocument( );
}
```

```
// Determine action based on node type
switch (node.getNodeType( ))
{
    case Node.DOCUMENT_NODE: break;
    case Node.ELEMENT_NODE: break;
    case Node.TEXT_NODE: break;
    case Node.CDATA_SECTION_NODE: break;
    case Node.COMMENT_NODE: break;
    case Node.PROCESSING_INSTRUCTION_NODE: break;
    case Node.ENTITY_REFERENCE_NODE: break;
    case Node.DOCUMENT_TYPE_NODE: break;
}
```

## Создание DOM дерева

```
DOMImplementation domImpl = new DOMImplementationImpl( );  
Document doc = domImpl.createDocument(null, "item", null);  
Element root = doc.getDocumentElement( );
```

## Добавление атрибута id

```
root.setAttribute("id", id);
```

## Создание нового элемента и текста в нем

```
Element nameElement = doc.createElement("name");  
Text nameText = doc.createTextNode(name);  
nameElement.appendChild(nameText);  
root.appendChild(nameElement);
```

## **Изменение содержания элемента**

```
NodeList nameElements = root.getElementsByTagName("name");  
Element nameElement = (Element)nameElements.item(0);  
Text nameText = (Text)nameElement.getFirstChild( );  
nameText.setData(name);
```

## **Получение *description* элемента**

```
NodeList descriptionElements =  
root.getElementsByTagName("description");  
Element descriptionElement = (Element)descriptionElements.item(0);
```

## **Удаление и создание другого *description* элемента**

```
root.removeChild(descriptionElement);  
descriptionElement = doc.createElement("description");  
Text descriptionText = doc.createTextNode(description);  
descriptionElement.appendChild(descriptionText);  
root.appendChild(descriptionElement);
```

## **Задаем формат DOM**

```
OutputFormat format = new OutputFormat(doc);
```

## **Создаем Writer и Serializer**

```
StringWriter stringOut = new StringWriter();
```

```
XMLSerializer serial = new XMLSerializer(stringOut, format);
```

## **получаем интерфейс DOMSerializer**

```
serial.asDOMSerializer();
```

## **Сериализуем XML и получаем строку**

```
serial.serialize(doc.getDocumentElement());
```

```
String result = stringOut.toString()
```

# Использование Xerces

## (1) Xerces: DOM parser

```
import org.w3c.dom.Document;  
import org.apache.xerces.parsers.DOMParser;  
import org.w3c.dom.Document;
```

```
String filename;
```

```
...
```

```
DOMParser parser = new DOMParser();  
parser.parse(filename);  
Document doc = parser.getDocument();
```

## (2) Xerces: SAX parser

```
import org.xml.sax.helpers.XMLReaderFactory;  
import org.xml.sax.XMLReader;  
import org.xml.sax.helpers.DefaultHandler;  
import org.w3c.dom.Document;  
DefaultHandler handler; String filename;
```

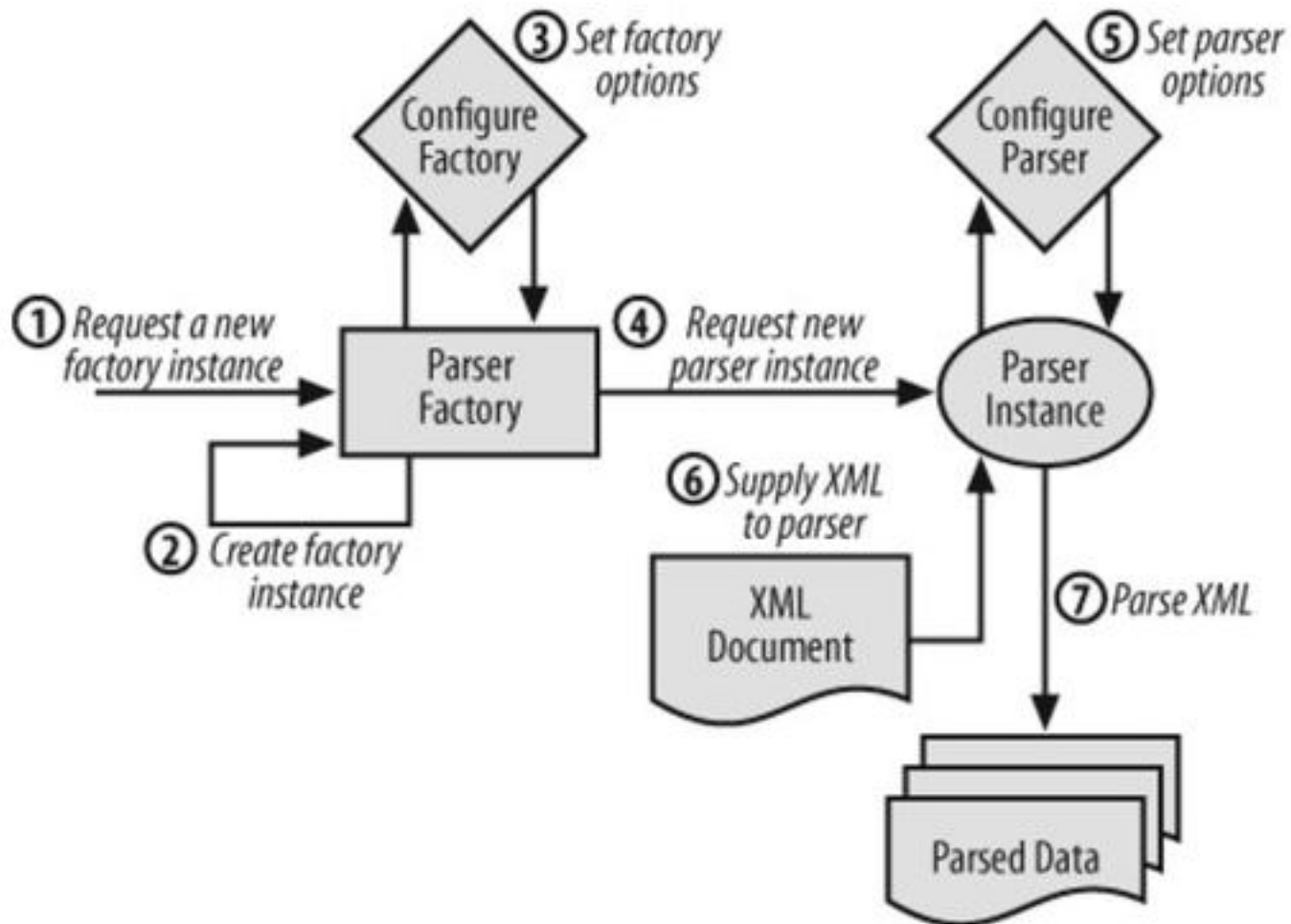
```
...
```

```
XMLReader parser = XMLReaderFactory.createXMLReader();  
parser.setContentHandler(handler);  
parser.setDTDHandler(handler);  
parser.setErrorHandler(handler);  
parser.parse(filename);
```

- **XML Parsing and Validation**
- **XSL Processing**
- **XPath**



# XML парсинг с помощью JAXP



## (1) JAXP: DOM parser

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document; String filename;
...
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.parse(filename);
```

## (2) JAXP: SAX parser

```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.helpers.DefaultHandler;
import org.w3c.dom.Document;
DefaultHandler handler;
String filename;
...
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();
parser.parse(filename, handler);
```

- **Работа с документом во время парсинга как в SAX.**
- **Приложение руководит порядком разбора**

```
StringReader stringReader = new StringReader(documentAsString);  
XMLInputFactory inputFactory = XMLInputFactory.newInstance( );  
XmlStreamReader reader =  
    inputFactory.createXMLStreamReader(stringReader);
```

# StAX - XmlStreamReader интерфейс

«interface» XMLStreamReader
+getProperty(name : String) : Object
+next() : int
+require(type : int, namespaceURI : String, localName : String) : void
+getElementText() : String
+nextTag() : int
+hasNext() : boolean
+close() : void
+getNamespaceURI(prefix : String) : String
+isStartElement() : boolean
+isEndElement() : boolean
+isCharacters() : boolean
+isWhiteSpace() : boolean
+getAttributeValue(namespaceURI : String, localName : String) : String
+getAttributeCount() : int
+getAttributeName(index : int) : QName
+getAttributeNamespace(index : int) : String
+getAttributeLocalName(index : int) : String
+getAttributePrefix(index : int) : String
+getAttributeType(index : int) : String
+getAttributeValue(index : int) : String
+isAttributeSpecified(index : int) : boolean
+getNamespaceCount() : int
+getNamespacePrefix(index : int) : String
+getNamespaceURI(index : int) : String
+getNamespaceContext() : NamespaceContext
+getEventType() : int
+getText() : String
+getTextCharacters() : char[]
+getTextCharacters(sourceStart : int, target : char[], targetStart : int, length : int) : int
+getTextStart() : int
+getTextLength() : int
+getEncoding() : String
+hasText() : boolean
+getLocation()
+getName() : QName
+getLocalName() : String
+hasName() : boolean
+getNamespaceURI() : String
+getPrefix() : String
+getVersion() : String
+isStandalone() : boolean
+standaloneSet() : boolean
+getCharacterEncodingScheme() : String

```
while (reader.hasNext( ))  
{  
    int type = reader.next( );  
    switch (type)  
    {  
        case XMLStreamConstants.START_DOCUMENT: ...  
        case XMLStreamConstants.END_DOCUMENT: ...  
        case XMLStreamConstants.START_ELEMENT: ...  
        case XMLStreamConstants.END_ELEMENT: ...  
        case XMLStreamConstants.CHARACTERS: ...  
        case XMLStreamConstants.ATTRIBUTE: ...  
        case XMLStreamConstants.CDATA: ...  
        case XMLStreamConstants.NAMESPACE: ...  
        case XMLStreamConstants.COMMENT: ...  
        case XMLStreamConstants.ENTITY_DECLARATION: ...  
        ...  
    }  
}
```

# StAX – создание документа

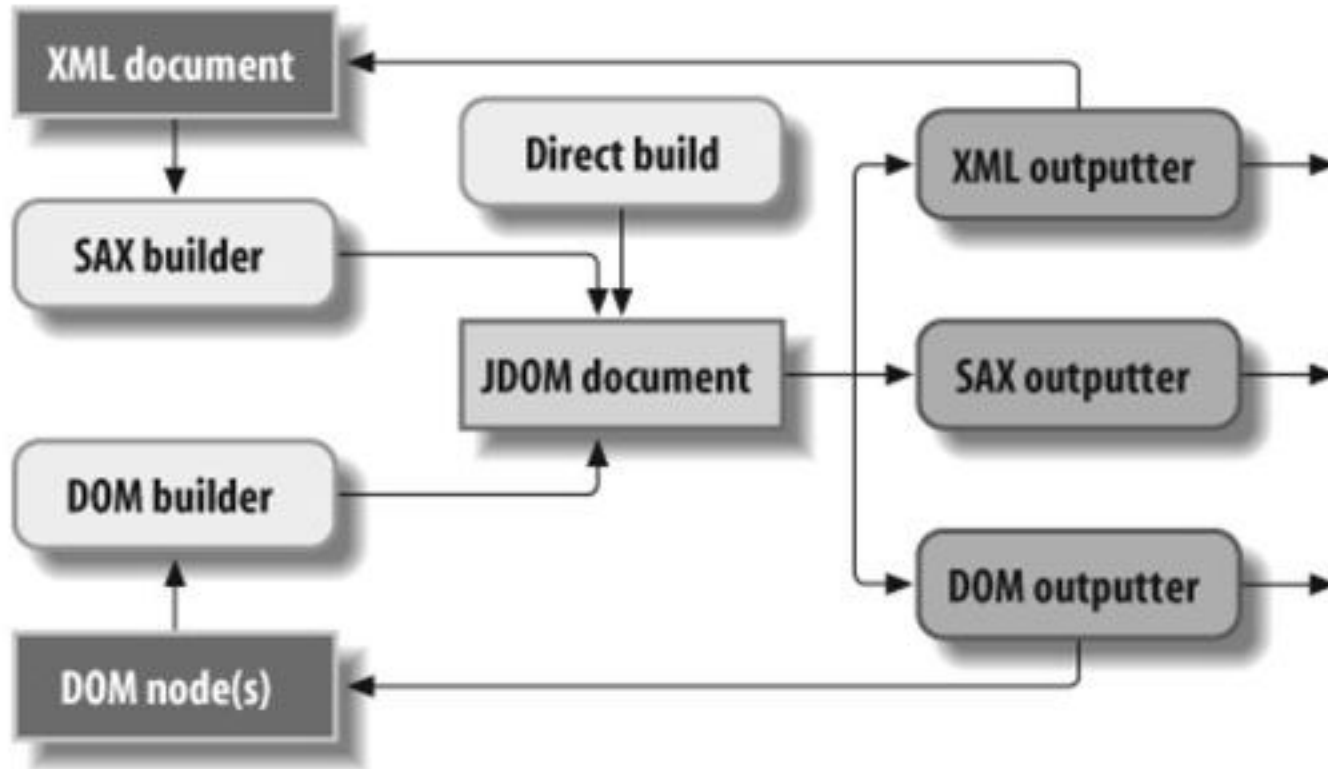
```
import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamWriter;

public class SimpleStreamOutput {
    public static void main(String[] args) throws Exception
    {
        XMLOutputFactory outputFactory = XMLOutputFactory.newInstance( );
        XMLStreamWriter writer = outputFactory.createXMLStreamWriter(System.out);

        writer.writeStartDocument("1.0");
        writer.writeStartElement("person");
        writer.writeStartElement("name");
        writer.writeStartElement("first_name");
        writer.writeCharacters("Alan");
        writer.writeEndElement( );
        writer.writeEndElement( );
        writer.writeEndElement( );
        writer.writeEndDocument( );
        writer.flush( );
    }
}
```

- **Java представление XML модели.**
- **Не является парсером.**
- **Основан на классах.**
- **Имеет поддержку XPath.**
- **Поддерживает XSLT трансформацию с помощью своего класса унаследованного от TrAX API Template класса.**





## Создание JDOM модели из SAX events и DOM модели

```
SAXBuilder builder = new SAXBuilder( );  
Document doc = builder.build(new FileInputStream("contents.xml"));
```

```
DOMBuilder builder = new DOMBuilder( );  
Document doc = builder.build(myDomDocumentObject);
```

## Преобразование JDOM в DOM и в SAX events

```
DOMOutputter outputter = new DOMOutputter( );  
org.w3c.dom.Document domDoc = outputter.output(myJDOMDocumentObject);
```

```
SAXOutputter outputter = new SAXOutputter( );  
outputter.setContentHandler(myContentHandler);  
outputter.setErrorHandler(myErrorHandler);  
outputter.output(myJDOMDocumentObject);
```

## Вывод JDOM

```
XMLOutputter outputter = new XMLOutputter( );  
outputter.output(jdomDocumentObject, new FileOutputStream("results.xml"));
```

- **Java представление XML модели**
- **Не является парсером**
- **Часть API похожа с JDOM**
- **Основан на интерфейсах**
- **Имеет поддержку XPath**
- **Интегрируется с JAXP для XSLT**

## Чтение документа

```
File file = new File(path);  
SAXReader reader = new SAXReader( );  
Document doc = reader.read(file);
```

## Создание документа

```
DocumentFactory factory = DocumentFactory.getInstance( );  
Document doc = factory.createDocument( );
```

**или**

```
Document doc = DocumentHelper.createDocument( );
```

## Добавление элемента

### долгий способ

```
Element myElement = factory.createElement("name");  
doc.add(myElement);
```

### быстрый способ

```
doc.addElement("name");
```



Delivering Excellence in Software Engineering

# Presentation Title

For more information, please contact:

**Name**

Title

EPAM Systems, Inc.

Address

City, State, Zip Code

Phone:

Fax:

Email:

<http://www.epam.com>

