Práctica 2
Programación XML

DIIC

Eduardo Martínez Graciá
Humberto Martínez Barberá

Procesamiento de XML

- Una aplicación que procese documentos XML debe analizarlos. El análisis se realiza usando librerías que implementan los analizadores.
- Existen dos tipos de analizadores XML:
  - Analizadores simples, comprueban que los documentos XML son documentos bien formados.
  - Analizadores de validación, comprueban que los documentos son bien formados y válidos XML de acuerdo a DTDs o esquemas.
- Otra clasificación de los analizadores es la siguiente:
  - Analizadores que tratan el documento como estructura plana (secuencialmente): el programador escribe los procedimientos a ejecutar durante el análisis: etiqueta de apertura, etiqueta de cierre, etc.
  - Analizadores que general un árbol de análisis. El programador recorre el árbol y consulta las propiedades de los nodos, que representan elementos XML.
- Segundo tipo más práctico, pero más lento. Además, los documentos grandes pueden requerir excesiva memoria para generar el árbol.

Procesamiento de XML

- Características de analizadores
  - Validación de los documentos XML
  - Interfaz (API) para manipular documentos XML
- Dos tipos de parsers
  - **DOM**, script's (java, vb, c++, c#,…) → árbol
  - **SAX**, JAVA → secuencial
    - JRE los tiene integrados

SAX: Simple API for XML

- SAX es una API estándar para el análisis de documentos XML. Apache Xerces incluye varias clases para emplearla.
- El análizador SAX trata de forma secuencial a los documentos XML, e invoca a código escrito por el programador cuando se producen eventos importantes a medida que recorre el documento.
- El programador registra *handlers* en el parser SAX. Un handler es un método escrito por el programador, cuya signatura está definida en un interfaz SAX, y que es invocado por el parser SAX al producirse cierto evento.
- Existen cuatro interfaces de handlers:
  - org.xml.sax.ContentHandler: eventos estándares relacionados con los datos.
  - org.xml.sax.ErrorHandler: eventos que notifican errores de análisis.
  - org.xml.sax.DTDHandler: eventos de análisis de DTDs.
  - org.xml.sax.EntityResolver: eventos para resolver entidades externas en documentos XML.
- El parser SAX ofrece métodos setContentHandler(), setErrorHandler()... para registrar clases que implementan los interfaces anteriores.

Creación de parser SAX (1)

```
import java.io.IOException;
import org.xml.sax.SAXException;
import org.xml.sax.Attributes;
import org.xml.sax.Locator;
import org.xml.sax.XMLReader;
import org.xml.sax.ContentHandler;
import org.xml.sax.ErrorHandler;
import org.apache.xerces.parsers.SAXParser;
...
public void performDemo(String uri) {
        System.out.println("Parsing XML File:" + uri );
        ContentHandler contentHandler = new MyContentHandler();
        ErrorHandler errorHandler = new MyErrorHandler();
        try {
        XMLReader parser = new SAXParser();
        parser.setContentHandler(contentHandler);
        parser.setErrorHandler(errorHandler);
                parser.parse(uri);
        } catch (IOException e) {
                System.out.println("Error reading URI:" + e.getMessage());
        } catch (SAXException e) {
                System.out.println("Error in parsing: " + e.getMessage());
        }}
```

Creación de parser SAX (2)

```
class MyContentHandler implements ContentHandler {

  private Locator locator;

  public void setDocumentLocator(Locator locator) {
        // Locator tiene los métodos
// getLineNumber() y getColumnNumber()
    this.locator = locator;
  }

  public void startDocument() throws SAXException {
    System.out.println("Parsing begins...");
  }

  public void endDocument() throws SAXException {
    System.out.println("....Parsing ends.");
  }

  public void processingInstruction(String target,
    String data) throws SAXException {
    System.out.println("PI: Target:" + target +
                       " and Data: " + data);
  }
```

Creación de parser SAX (3)

```
public void startPrefixMapping(String prefix,
   String uri) {
   System.out.println("Mapping starts for prefix " +prefix +
                      " mapped to URI " + uri);
 }

 public void endPrefixMapping(String prefix) {
   System.out.println("Mapping ends for prefix " +prefix);
 }

 public void startElement(String namespaceURI,
   String localName, String rawName, Attributes atts) throws SAXException {
   System.out.print("startElement: " + localName);
   if (!namespaceURI.equals("")) {
     System.out.println(" in namespace "+ namespaceURI +
                        " (" + rawName + ")");
   } else {
     System.out.println(" has no associated namespace"); }
   for (int i = 0; i < atts.getLength(); i++)
     System.out.println("  Attribute: " + atts.getLocalName(i)+
                        "=" + atts.getValue(i)); }
```

Creación de parser SAX (4)

```
  public void endElement(String namespaceURI,
     String localName, String rawName) throws SAXException {
       System.out.println("endElement: " + localName + "\n");
  }

  public void characters(char[] ch, int start, int end)
     throws SAXException {
       String s = new String(ch, start, end);
       System.out.println("characters: " + s);
  }

  public void ignorableWhitespace(char[] ch, int start, int end)
     throws SAXException {
  }

  public void skippedEntity(String name)
     throws SAXException {
  }
}
```

Creación de parser SAX (5)

```
class MyErrorHandler implements ErrorHandler {
  public void warning (SAXParseException exception) throws SAXException {
    System.out.println("**Parsing Warning**\n" +
                       "Line: " +
                                        exception.getLineNumber() + "\n" +
                                       "URI: " +
                                        exception.getSystemId() + "\n" +
                                       "Message: " +
                                        exception.getMessage());
    throw new SAXException("Warning encountered"); }

  public void error (SAXParseException exception) throws SAXException {
    // Errores de validación
    System.out.println("**Parsing Error**\n" +
                       "Line: " +
                                        exception.getLineNumber() + "\n" +
                                       "URI: " +
                                        exception.getSystemId() + "\n" +
                                       "Message: " +
                                        exception.getMessage());
    throw new SAXException("Error encountered"); }
```

Creación de parser SAX (6)

```
 public void fatalError (SAXParseException exception)
    throws SAXException {
    // Errores de XML mal formado
    System.out.println("**Parsing Fatal Error**\n" +
                       "Line: " +
                                        exception.getLineNumber() + "\n" +
                                       "URI: " +
                                        exception.getSystemId() + "\n" +
                                       "Message: " +
                                        exception.getMessage());
    throw new SAXException("Fatal Error encountered");
  }
}
```

Configuración del parser

- SAX 2.0 incorpora un mecanismo estándar y extensible para la configuración de características de un parser XML (dada la continua evolución de XML)
- La configuración se realiza con el método *setFeature("[URI]", true)* del interfaz XMLReader.
- URI es el identificador que representa la característica que se está configurando. SAX 2.0 define las siguientes:
  - Namespace Processing: *http://xml.org/sax/features/namespaces.* Activa o desactiva el procesamiento de namespaces (startPrefixMapping, endPrefixMapping, y ciertos parámetros de startElement y endElement).
  - Validación: *http://xml.org/sax/features/validation*. Activa o desactiva la validación (y por tanto las llamadas al interfaz ErrorHandler)
  - Procesamiento de entidades externas generales:
    - *http://xml.org/sax/features/external-general-entities*
  - Procesamiento de entidades externas parámetro
    - *http://xml.org/sax/features/external-parameter-entities*
- Los parsers pueden definir sus propios URIs de configuración:
- Validación con esquemas: *http://xml.apache.org/xerces2-j/features.html*

DOM: Document Object Model

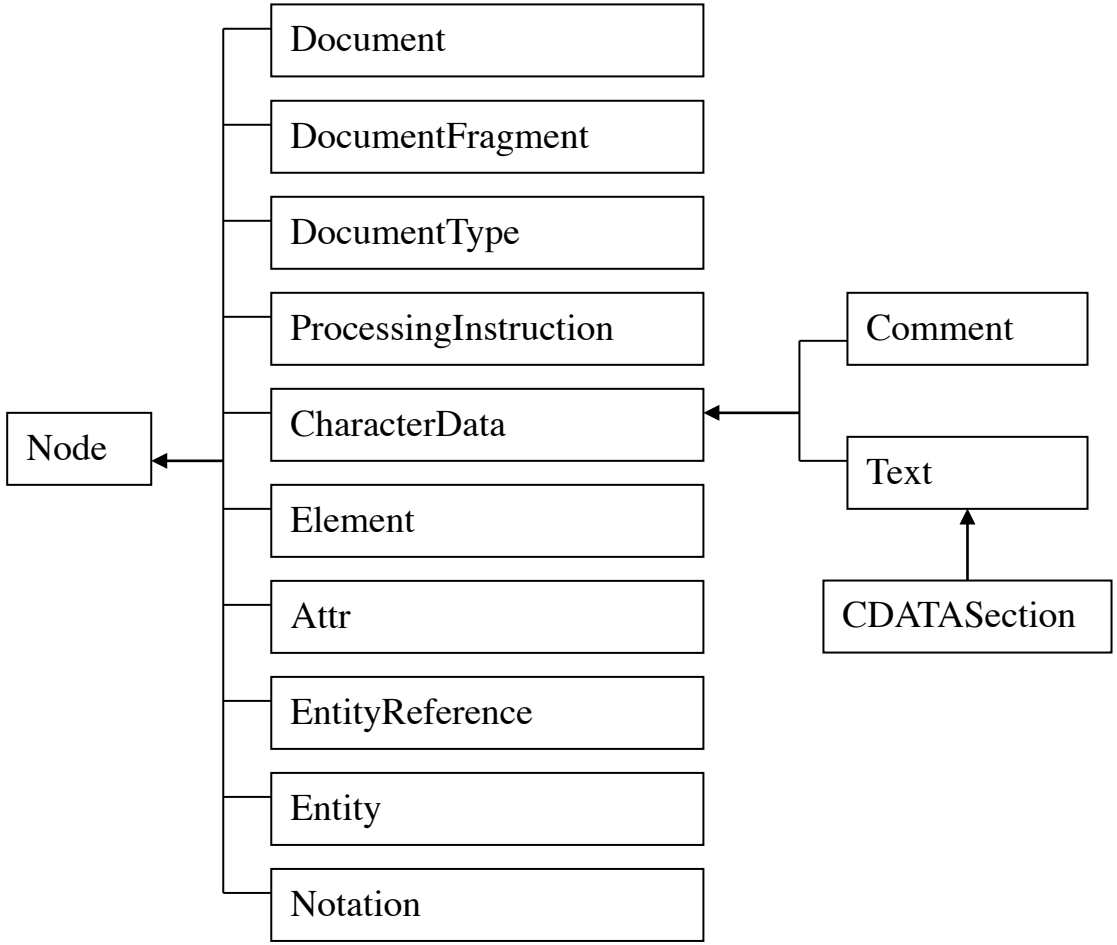
- DOM es una especificación del W3C (a diferencia de SAX) independiente de la plataforma que define una serie de interfaces basados en objetos que permiten crear programas y scripts que accedan y actualicen al contenido, la estructura y la presentación de los documentos.
- Existen "bindings" de DOM a diversos lenguajes de programación, entre ellos Java. En Java, los programadores emplean librerías que implementan el paquete *org.w3c.dom*
- El resultado del análisis es un *org.w3c.dom.Document*, un objeto que representa un árbol de análisis del documento. Por tanto, hasta que no finaliza el análisis no puede comenzar el procesamiento:

```
public void performDemo(String uri) {
        DOMParser parser = new DOMParser();
        try {
                parser.parse(uri);
                org.w3c.dom.Document doc = parser.getDocument();
                // Procesamiento

        } catch  (Exception e) { ... } }
```

Uso de un árbol DOM

- Los nodos de un árbol DOM pueden ser de diversos tipos (org.w3c.dom)




```
Determinar el tipo de un nodo:

Node node;
...
switch (node.getNodeType()) {
case Node.DOCUMENT_NODE:
// Objeto Document
Document doc =(Document)node;
break;
case Node.ELEMENT_NODE:
// Objeto Element
Element e = (Element)node;
break;
case Node.TEXT_NODE:
// Objeto Text
Text t = (Text)node;
break;
... }
```

Impresión del árbol de análisis (1)

```
public void performDemo(String uri) {
        System.out.println("Parsing XML File: " + uri);
        DomParser parser = new DOMParser();
        try {  parser.parse(uri);
                Document doc = parser.getDocument();
                printNode(doc, "");
        } catch (IOException e) {
                System.out.println("Error reading URI: " + e.getMessage());
        } catch (SAXException e) {
                System.out.println("Error in parsing: " + e.getMessage());
        } }
public void printNode(Node node, String indent) {
  switch(node.getNodeType()) {
    case Node.DOCUMENT_NODE:
          System.out.println("<? xml version=\"1.0\" ?>\n");
          Document doc = (Document)node;
          DocumentType docType = doc.getDoctype();
          System.out.print("<!DOCTYPE " + docType.getName());
          System.out.print(" SYSTEM \"" + docType.getSystemId());
          System.out.println("\">");
          printNode(doc.getDocumentElement(), "");
          break;
```

Impresión del árbol de análisis (2)

```
  case Node.ELEMENT_NODE:
    String name = node.getNodeName();
    System.out.print(ident + "<" + name);
    NamedNodeMap attributes = node.getAttributes();
    for (int i = 0; i < attributes.getLength(); i++) {
      Node current = attributes.item(i);
          System.out.print(" " + current.getNodeName() +
                  " = \"" + current.getNodeValue() + "\"");
    }
    System.out.println(">");
    NodeList children = node.getChildNodes();

// Recursividad por cada hijo
    if (children != null) {
      for (int i = 0; i < children.getLength(); i++) {
           printNode(children.item(i), ident + "  "); } }
    System.out.println(ident + "</" + name + ">");
    break;
  case Node.TEXT_NODE:
  case Node.CDATA_SECTION_NODE:
    System.out.print(node.getNodeValue());
    break;  } }
```

Generación de documentos XML

```
import org.apache.xerces.dom.DOMImplementationImpl;
import org.w3c.dom.*;

// construyo el documento
DOMImplementation domImpl = new DOMImplementationImpl();

Document doc = domImpl.createDocument(uri_nombres, "respuestaPerfil", null);

// Raiz
Element _root = doc.getDocumentElement();

_root.setAttributeNS("http://www.w3.org/2001/XMLSchema-instance",
                     "xsi:schemaLocation", uri_nombres + " " + url_esquema);

Element _id = doc.createElementNS(uri_nombres, "id");
_id.appendChild(doc.createTextNode(cod_propio));
_root.appendChild(_id);

Element _nombre = doc.createElementNS(uri_nombres, "nombre");
_nombre.appendChild(doc.createTextNode(nombre));
_root.appendChild(_nombre);
```

Bibliografía

- M. Morrison (2000) *XML al descubierto*. Prentice-Hall
- B. McLaughlin (2001) *Java & XML*. O'Reilly & Associates
- Extensible Markup Language (XML). http://www.w3.org/XML/
- XML at The Apache Foundation. http://xml.apache.org/
- XML Schema. http://www.w3.org/XML/Schema