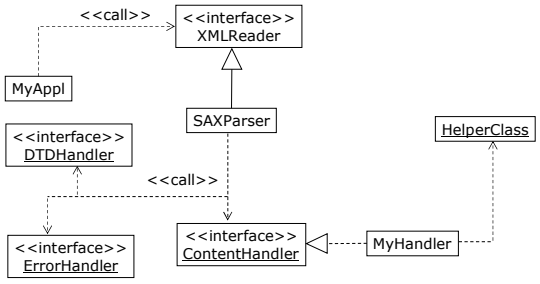
Laboratorio di sistemi interattivi

Lezione 12: Java & XML

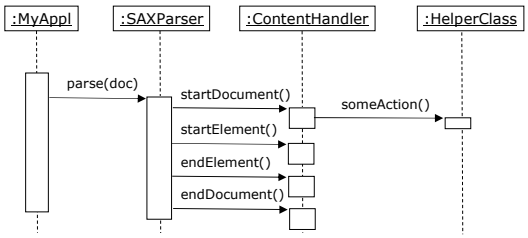
Struttura di un'applicazione Java-XML

- Parser analizza il contenuto di un file XML
- Produce eventi in corrispondenza delle diverse componenti
- Eventi provocano callback in gestori appropriati
- Costruita sfruttando SAX (a Simple API for XML)

Struttura di un'applicazione Java-XML



Protocollo SAX



Processo di parsing

- Parser implementa XMLReader
- I parser sono generalmente forniti da organizzazioni esterne.
  Es. org.apache.xerces.parsers.SAXParser
- Documento XML letto sequenzialmente
- Il metodo parse incontra i tag del documento XML e lancia degli eventi verso il ContentHandler.
- Parser legge un documento alla volta

Definizione dei Parser

```
public interface XMLReader {
  void parse(String uri);
  void parse(org.xml.sax.InputSource is);
  void setContentHandler(ContentHandler ch);
  void setErrorHandler(ErrorHandler eh);
…
}
```

Gestori del contenuto

- Specificano callback attivati in corrispondenza di eventi durante il processo di parsing.

Definizione gestori contenuto I

```
public interface ContentHandler {
  void setDocumentLocator(Locator locator);
  void startDocument();
  void endDocument();
  void processingInstruction(String target,
      String data);
  void startPrefixMapping(String prefix,
      String uri);
  void endPrefixMapping(String prefix);
```


Definizione gestori contenuto II

```
  void startElement(String namespaceURI,
      String localName, String rawName,
      Attributes atts)
  void endElement(String namespaceURI,
      String localName, String rawName);
  void characters(char[] ch, int start, int end);
  void ignorableWhitespace(char[] ch, int
      start, int end);
}
```

Gestione del contenuto I

- Locator indica la posizione di ogni evento nel documento. È associato al ContentHandler
- startDocument chiamato prima di ogni altro callback
- endDocument chiamato dopo ogni altro metodo (incluso il caso di errori)
- processingInstruction chiamata per codice del tipo <?target instruction?>

Gestione del contenuto II

- prefixMapping è un elemento che usa l'attributo `xmlns` per dichiarare uno spazio di nomi
- startPrefixMapping è lanciato prima del callback per l'elemento che dichiara lo spazio
- endPrefixMapping lanciato dopo il tag di chiusura per l'elemento (chiude il namespace)

Gestione del contenuto III

- startElement gestisce il nome dell'elemento XML e i suoi attributi
- Attributi non sono presentati nell'ordine in cui sono scritti.
- endElement gestisce la chiusura, caratteri successivi appartengono ad altri elementi
- characters gestisce i dati dell'elemento (attenzione! #PCDATA! I figli sono gestiti da coppie startElement/endElement annidate)

Definizione gestori errori

```
public interface ErrorHandler {
  void warning(SAXParseException exception);
  void error(SAXParseException exception);
  void fatalError(SAXParseException exception);
}
```


Gestione degli errori

- warning: problemi che non mettono in questione la gestione del documento
- error: (non fatale) violazione di qualche parte della specifica, recuperabile (es. versione successiva a quella gestita dal parser)
- fatalError: non permettono di proseguire con il processo di parsing

Validazione dei documenti

```
parser.setFeature(
   http://xml.org/sax/features/validation, true);
```

Permette al parser di gestire la corrispondenza con il DTD

```
public interface DTDHandler { //tratta la parte DTD
   unparsedEntityDecl();
// usata con entità dati che non vanno analizzati (es. immagine)
   notationDecl();
// usata per mettere in corrispondenza tipi di dati definiti
   esternamente con la loro definizione (es. formato immagine)
}
```

Features e Properties

- Feature:
  - Insieme di caratteristiche che possono essere assunte dal Parser (XMLReader)
- Esempi:
  - Validation; Namespace Processing
- Property:
  - Specifica di oggetti che possono essere usati nel processo di parsing
- Esempi:
  - LexicalHandler; DeclarationHandler; DOMNode; XMLString

Limitazioni nell'uso di SAX

- Lettura sequenziale:
  - non si può accedere casualmente agli elementi
  - non si mantiene informazione sul livello della gerarchia a cui ci si trova
    - Non si conoscono fratelli
    - Non si conoscono elementi annidati o parent

Document Object Model (DOM)

- DOMParser produce Document (albero di elementi dal documento XML)
- Interfaccia Node specializzata in:
  - Document, DocumentFragment, DocumentType, ProcessingInstruction, CharacterData, Element, Attr, EntityReference, Entity, Notation
  - CharacterData specializzata in:
    - Comment, Text (specializzata in CDATASection)

Uso del DOM

```
myMethod(Document doc) {
    doSomething(doc);
}

doSomething(Node node) {
   switch(node.getNodeType() {
        // azioni specifiche secondo nCl
   }
   // ciclo ricorsivo su elementi in getChildNodes()
}
```


DOM e XSLT

- Processori XSLT possono creare un DOM per il documento di output
- Processori XSLT possono copiare, modificare, aggiungere ed eliminare nodi dal DOM del documento di input e creare un albero di output
- Necessità di caricare in memoria l'intero albero

Trasformazioni di XML in Java

Xalan-Java è un XSLT processor per trasformare un documento XML in uno HTML, testo o un altro XML

Si può richiamare da linea di comando con:

java org.apache.xalan.xslt.Process

-IN [XML Document]
-XSL [XSL Stylesheet]
-OUT [Output Filename]

http://xml.apache.org/xalan-j/

Translet

Compilatore XSLTC:

- Ingresso: uno stylesheet
- Uscita: file Translet.class, programma compilato che trasforma XML secondo lo stylesheet

- Effetto: non è necessario leggere ogni volta lo stylesheet per trasformare un documento.