Programmazione Orientata agli Oggetti in Linguaggio Java

Programmazione XML: JDOM

versione 2.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Introduzione
- Operazioni Principali con JDOM
  - ⇨Caricamento del DOM
  - ⇨Visita del DOM
  - ⇨Creazione del DOM
  - ⇨Salvataggio del DOM
- Le Stesse Operazioni con JAXP
- Operazioni Avanzate con JDOM

Introduzione

- JDOM
  - ⇨una libreria alternativa per XML
- Rispetto a JAXP
  - ⇨due differenze fondamentali
  - ⇨I differenza: è pensata esplicitamente per Java e per risultare naturale allo sviluppatore Java
  - ⇨II differenza: è pensata per essere semplice

Introduzione

- JDOM come standard
  - ⇨JDOM è stata proposta per la standardizzazione al Java Community Process sotto forma di JSR 102
  - ⇨ma la JSR (Java Specification Request) si è bloccata prima di essere approvata
  - ⇨è possibile che in un prossimo futuro JDOM diventi una estensione standard della piattaforma

Introduzione

- L'approccio di JDOM
  - ⇨JDOM è basato su JAXP
  - ⇨incapsula i parser configurati per JAXP e consente di utilizzarli in modo più semplice
  - ⇨in particolare, JDOM cerca di utilizzare il parser ed il motore di trasformazione di JAXP
  - ⇨se non ci riesce, utilizza i suoi strumenti standard, ovvero xerces e xalan

Introduzione

- L'approccio di JDOM (continua)
  - ⇨JDOM introduce varie semplificazioni rispetto a DOM
  - ⇨incapsula le interfacce definite da DOM trasformandole in interfacce più semplici
  - ⇨utilizza strumenti nativi di Java; es: nasconde NodeList e trasforma le collezioni di DOM in collezioni standard di Java

Introduzione

- I package fondamentali di JDOM
  - ⇨org.jdom – contiene le classi corrispondenti agli elementi dell'InfoSet
  - ⇨org.jdom.input – per il parsing di documenti
  - ⇨org.jdom.output – per la scrittura di doc.
  - ⇨org.jdom.transform – per XSLT
  - ⇨org.jdom.xpath – per XPath

Operazioni Fondamentali con JDOM

- Nel seguito
  - ⇨le operazioni fondamentali su documenti XML basate su JDOM
- In particolare
  - ⇨generazione del DOM di un documento XML
  - ⇨visita del DOM generato in memoria
  - ⇨creazione e manipolazione di un DOM
  - ⇨salvataggio del documento XML dal DOM

Operazioni Fondamentali

ATTENZIONE
allo studio delle librerie

- Una nota di metodo
  - ⇨come affrontare lo studio di una libreria
- Imparare ad usare una libreria
  - ⇨conoscerne i principi di funzionamento
  - ⇨conoscerne i componenti (package e classi)
  - ⇨conoscerne i metodi
- Nel caso di JDOM
  - ⇨varie centinaia di metodi

Operazioni Fondamentali con JDOM

- Impostazione della lezione >> JavaDoc di JDOM
  - ⇨verranno per cominciare illustrati i principi di funzionamento
  - ⇨verranno elencati i principali componenti
  - ⇨verranno illustrati i principali metodi
- Ma è impossibile descrivere tutti i metodi
  - ⇨è necessario completare lo studio consultando di volta in volta la documentazione in formato JavaDoc

Caricamento del DOM

- Caricamento del DOM di un documento
  - ⇨basato sulla classe SAXBuilder
  - ⇨viene utilizzato il parser SAX, selezionato secondo le regole illustrate prima
  - ⇨SAXBuilder usa però il parser SAX per costruire l'intero albero di oggetti DOM (quindi il funzionamento di SAX è "nascosto")
  - ⇨viene restituita la radice dell'albero come riferimento di tipo org.jdom.Document


```
private static org.jdom.Document costruisciDOM(String nomeFile)
                                                    throws DAOException {
    org.jdom.input.SAXBuilder builder =
                        new org.jdom.input.SAXBuilder();
    builder.setValidation(true);
    org.jdom.Document document = null;
    try {
      document = builder.build(nomeFile);
      return document;
    } catch (org.jdom.JDOMException jde) {
        System.err.println(jde);
        throw new DAOException(jde);
    } catch (java.io.IOException ioe) {
        System.err.println(ioe);
        throw new DAOException(ioe);
    }
  }
```

creazione del parser SAX

richiede che il parser effettui la convalida rispetto ad un DTD

il metodo build costruisce l'albero di oggetti DOM e restituisce il riferimento alla radice

le possibili eccezioni sono org.jdom.JDOMException e java.io.IOException in questo metodo vengono trasformate e rilanciate (>>)

Visita del DOM

- Visita del DOM generato in memoria
  - a questo punto posso visitare l'albero
  - oltre a org.jdom.Document, ho varie classi corrispondenti ai vari tipi di nodo
  - org.jdom.Element
  - org.jdom.Attribute
  - org.jdom.Text
  - org.jdom.DocType
  - ...

Visita del DOM

- I metodi per la visita dell'albero
  - metodi che consentono di accedere ai figli di un nodo
- Esempio: la classe Element
  - Element getChild(String name): restituisce il primo tra gli elementi figli con il nome specificato
  - List getChildren(): restituisce tutti gli elementi figli
  - List getChildren(String name): restituisce tutti gli elementi figli con il nome specificato

Visita del DOM

- Esempio: la classe Element (continua)
  - ⇨String getAttributeValue(String name): restituisce il valore dell'attributo con il nome specificato
  - ⇨String getText(): restituisce il contenuto dei nodi testo figli dell'elemento
- Nota
  - ⇨i metodi sono scritti in modo da distinguere la categoria di figli restituiti
  - ⇨in questo modo si aggirano molti dei problemi dovuti ai nodi di spazio bianco

Visita del DOM

- Esempio
  - ⇨estrazione della lista degli argomenti
  - ⇨visito l'albero a partire dalla radice alla ricerca dei nodi rilevanti
  - ⇨analizzo i nodi che rappresentano argomenti, estraggo valori dai nodi incontrati e genero oggetti del mio modello (Argomento)
  - ⇨li aggiungo al Questionario

```
public static Questionario caricaQuestionario(String nomeFile)
                                                    throws DAOException {
    org.jdom.Document document = costruisciDOM(nomeFile);
    Questionario questionario = new Questionario();
    analizzaRadice(document, questionario);
    riempiArgomenti(document, questionario);
    riempiDifficolta(document, questionario);
    riempiQuesiti(document, questionario);
    return questionario;
}

private static void analizzaRadice (org.jdom.Document document,
                                    Questionario questionario) {
    Element elementoRadice = document.getRootElement();
    String id = elementoRadice.getAttributeValue("id");
    String disciplina = elementoRadice.getAttributeValue("disciplina");
    questionario.setId(id);
    questionario.setDisciplina(disciplina);
}
```

analizzo i nodi dell'albero ed estraggo valori per gli oggetti del modello

```
private static void riempiArgomenti (org.jdom.Document document,
                                     Questionario questionario) throws DAOException {
    org.jdom.Element elementoRadice = document.getRootElement();
    org.jdom.Element elementoListaArg = elementoRadice.getChild("listaArgomenti");
    java.util.List listaArg = elementoListaArg.getChildren();
    for (int i = 0; i < listaArg.size(); i++ ) {
        org.jdom.Element elementoArgomento = (org.jdom.Element)listaArg.get(i);
        Argomento argomento = costruisciArgomento(elementoArgomento);
        questionario.addArgomento(argomento);
    }
}

private static Argomento costruisciArgomento(org.jdom.Element elemArgomento) {
    Argomento argomento = new Argomento();
    String id = elemArgomento.getAttributeValue("id");
    String nome = elemArgomento.getAttributeValue("nome");
    argomento.setId(id);
    argomento.setNome(nome);
    return argomento;
}
```

Vista del DOM

>> DAOQuestionario.java
>> DAORisposte.java

- Analogamente
  - ⇨nel caso in cui è necessario ricostruire un quesito
  - ⇨a partire dal nodo elemento che rappresenta il quesito viene visitato il sottoalbero
  - ⇨vengono estratti i valori e costruito l'oggetto di tipo Quesito corrispondente
- Lo stesso processo
  - ⇨viene utilizzato per caricare le risposte

Creazione del DOM

- Creazione e manipolazione di un DOM
  - ⇨in questo caso è necessario partire dagli oggetti del modello
  - ⇨e a partire da quelli creare i nodi dell'albero (nodo documento, eventuale dichiarazione di DTD, elementi, attributi, testo)
  - ⇨connettendoli secondo le strutture richieste

Creazione del DOM

- Esempio
  - ⇨creazione degli oggetti DOM corrispondenti alle risposte fornite dall'utente
  - ⇨parto da un oggetto di tipo Risposte
  - ⇨creo un albero DOM che rappresenta l'InfoSet di un documento conforme al DTD richiesto

Creazione del DOM

- Il DTD delle risposte

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

<!ELEMENT risposte  (listaRisposte) >
<!ATTLIST risposte     idquestionario  CDATA #REQUIRED >
<!ELEMENT listaRisposte   (risposta)+ >
<!ELEMENT risposta  (#PCDATA) >
<!ATTLIST risposta     id     ID      #REQUIRED >
```

```
private static org.jdom.Document costruisciDocumento() {
        org.jdom.Document document = new org.jdom.Document();
        org.jdom.DocType docType =
                    new org.jdom.DocType("risposte", "risposte.dtd");
        document.setDocType(docType);
        org.jdom.Element radice = new org.jdom.Element("risposte");
        document.setRootElement(radice);
        return document;
 }

private static void completaRadice(Risposte risposte,
                                   org.jdom.Document document){
        org.jdom.Attribute idQuestionario =
                new org.jdom.Attribute("idquestionario",
                                       risposte.getIdQuestionario());
        org.jdom.Element radice = document.getRootElement();
        radice.setAttribute(idQuestionario);
}
```

l'oggetto DocType serve a specificare il DTD prevede di specificare il nome dell'elemento radice e il riferimento al file contenente il DTD


```
private static void aggiungiListaRisposte(Risposte risposte,
                                          org.jdom.Document document) {
        org.jdom.Element radice = document.getRootElement();
        org.jdom.Element elementoListaRisposte =
                            new org.jdom.Element("listaRisposte");
        radice.addContent(elementoListaRisposte);
        for (int i = 0; i < risposte.getNumeroRisposte(); i++) {
           aggiungiRisposta(risposte, i, elementoListaRisposte);
        }
}

private static void aggiungiRisposta(Risposte risposte, int i,
                                     org.jdom.Element elementoListaRisposte) {
        org.jdom.Element elementoRisposta = new org.jdom.Element("risposta");
        String risposta = risposte.getRisposta(i);
        elementoRisposta.setAttribute("id", "r" + i);
        elementoRisposta.setText(risposta);
        elementoListaRisposte.addContent(elementoRisposta);
    }
```

Salvataggio del DOM

- Salvataggio del DOM creato
  - una volta creato completamente l'albero di oggetti DOM, è necessario salvarlo in un documento XML
  - è necessario utilizzare un flusso
  - inoltre è opportuno specificare le opzioni di formato desiderate: indentazione, utilizzo di caratteri di fine riga ecc.

Salvataggio del DOM

>> JavaDoc di Format

- Per il salvataggio del DOM
  - è possibile utilizzare un oggetto di tipo org.jdom.output.XMLOutputter
- Per specificare il formato
  - è possibile utilizzare la classe org.jdom.output.Format
  - con varie opzioni di formato
  - RawFormat, PrettyFormat, CompactFormat
  - per ottenere ciascuno un opportuno metodo

```
private static void salvaDocumento(org.jdom.Document document, String nomeFile)
                                                    throws DAOException {
    org.jdom.output.XMLOutputter outputter = new org.jdom.output.XMLOutputter();
    org.jdom.output.Format format = org.jdom.output.Format.getPrettyFormat();
    format.setIndent("  ");
    format.setLineSeparator("\n");
    outputter.setFormat(format);
    java.io.FileWriter fileWriter = null;
    try {
       fileWriter = new java.io.FileWriter(nomeFile);
       outputter.output(document, fileWriter);
    } catch (java.io.IOException ioe) {
       throw new DAOException(ioe);
    } finally {
       if (fileWriter != null) {
          try {
             fileWriter.close();
          } catch (java.io.IOException ioe) {}
       }
    }
 }
```

l'oggetto di tipo Format
consente di specificare varie opzioni, tra cui
-gli spazi da usare nell'indentazione
- il carattere da utilzzare come fine riga

Le Stesse Operazioni con JAXP

- Nel complesso, con JDOM: 8 classi princ.
  - ⇨org.jdom.input.SAXBuilder
  - ⇨org.jdom.JDOMException
  - ⇨org.jdom.Document
  - ⇨org.jdom.Element
  - ⇨org.jdom.Attribute (evitabile)
  - ⇨org.jdom.DocType
  - ⇨org.jdom.output.XMLOutputter
  - ⇨org.jdom.output.Format

Le Stesse Operazioni con JAXP

>> DAOQuestionario.java
>> DAORisposte.java

- Utilizzando JAXP
  - ⇨questionari2, un'applicazione identica scritta senza usare JDOM
- Caratteristiche fondamentali
  - ⇨codice più verboso (circa un 25-30% in più)
  - ⇨più classi utilizzate di package diversi
  - ⇨è necessario sviluppare un ErrorHandler per gestire gli errori durante la convalida

Le Stesse Operazioni con JAXP

- Le classi utilizzate: 16 (!)

org.w3c.dom.Document
org.w3c.dom.Element
org.w3c.dom.Attr
org.w3c.dom.Text
org.w3c.dom.NodeList

javax.xml.parsers.DocumentBuilderFactory
javax.xml.parsers.DocumentBuilder
javax.xml.parsers.ParserConfigurationException

org.xml.sax.SAXException

javax.xml.transform.TransformerFactory
javax.xml.transform.Transformer
javax.xml.transform.dom.DOMSource
javax.xml.transform.stream.StreamResult
javax.xml.transform.TransformerConfigurationException
javax.xml.transform.TransformerException

Operazioni Avanzate con JDOM

- Oltre alle operazioni di base
  - ⇨JDOM supporta altre operazioni
- In particolare
  - ⇨l'utilizzo di XPath per specificare cammini nell'InfoSet
  - ⇨l'utilizzo di XMLSchema invece dei DTD per la convalida dei documenti

Operazioni Avanzate con JDOM

- Utilizzo di XPath
  - ⇨è possibile specificare cammini XPath
  - ⇨e ottenere una lista che l'insieme dei nodi risultato della valutazione del cammino
  - ⇨la classe relativa è la classe XPath
  - ⇨del package org.jdom.xpath
- Esempio
  - ⇨estrarre i nodi risposta dal documento delle risposte

```
private static void estraiRisposte(org.jdom.Document document,
                                   Risposte risposte) {
    org.jdom.Element radice = document.getRootElement();
    String cammino = "/risposte/listaRisposte/risposta";
    java.util.List listaNodiRisultato =
            org.jdom.xpath.XPath.selectNodes(radice, cammino);
    for (int i = 0; i < listaNodiRisultato.size(); i++) {
        org.jdom.Element nodo =
            (org.jdom.Element)listaNodiRisultato.get(i);
        aggiungiRisposta(risposte, nodo);
    }
}
```

-viene specificato un cammino
-viene chiesto alla classe XPath di valutare il risultato del cammino a partire da un nodo di contesto (in questo caso la radice)

Operazioni Avanzate con JDOM

- L'implementazione di XPath
  - è possibile utilizzare varie implementazioni
  - attraverso il valore della proprietà di sistema org.jdom.xpath.class
  - o utilizzando il metodo setXPathClass()
  - l'implementazione standard è Jaxen, contenuto in jaxen-core.jar, jaxen-jdom.jar
  - devono essere raggiungibili attraverso il CLASSPATH

Operazioni Avanzate con JDOM

- Utilizzo di XMLSchema
  - ⇨non fa parte delle funzionalità incluse nel package JDOM
  - ⇨ma del package JDOM-Contrib, scaricabile da www.jdom.org
- In particolare
  - ⇨il package org.jdom.contrib.schema consente di effettuare convalide rispetto a vari tipi di schema

Operazioni Avanzate con JDOM

- Convalida rispetto ad uno schema
  - ⇨può essere effettuata utilizzando la classe org.jdom.contrib.schema.Schema
  - ⇨supporta XMLSchema
  - ⇨e la maggior parte degli schemi alternativi (Relax, Relax-NG, Trex, ecc.)
  - ⇨si basa sul package Iso-Relax Verifier, contenuto in iso-relax.jar, che deve essere raggiungibile attraverso il CLASSPATH

Operazioni Avanzate con JDOM

- Per convalidare rispetto a XMLSchema

```
import org.jdom.contrib.schema.Schema;

public void validate(org.jdom.Document document,
                     String URIDelloSchema) {
    Schema schema = Schema.parse(URIDelloSchema,
                                 Schema.W3C_XML_SCHEMA);
    java.util.List errori = schema.validate(document);
    if (errori != null) {
        stampaErrori(errori);
    } else {
        System.out.println("Documento valido rispetto allo schema");
    }
}
```

il riferimento allo schema viene fornito sotto forma di URI del tipo file:///d:/schemi/questionario.xsd oppure http://www.w3c.org/schema.xsd

Riassumendo

- Introduzione
- Operazioni Principali con JDOM
  - ⇨Caricamento del DOM
  - ⇨Visita del DOM
  - ⇨Creazione del DOM
  - ⇨Salvataggio del DOM
- Le Stesse Operazioni con JAXP
- Operazioni Avanzate con JDOM

Termini della Licenza