Programmazione Orientata agli Oggetti in Linguaggio Java

Programmazione su XML: Librerie per XML

versione 2.0

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

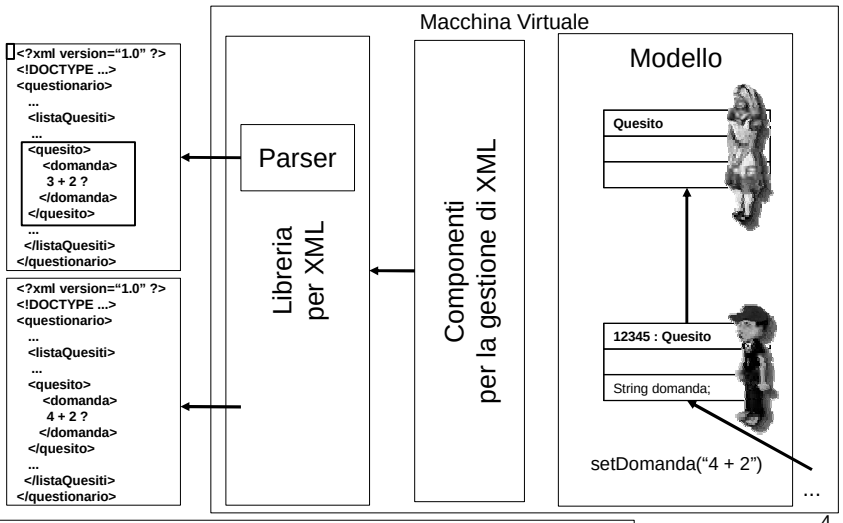
Sommario

- Introduzione
- Document Object Model (DOM)
- Simple API for XML (SAX)
- Differenze tra DOM e SAX
- Librerie per XML
- JAXP

Introduzione

- Programmare con XML
  - ⇨“caricare” i dati contenuti in uno o più documenti XML (es: questionario.xml)
  - ⇨rappresentarli attraverso oggetti del modello dell’applicazione (es: Questionario, Quesito)
  - ⇨effettuare le elaborazioni
  - ⇨eventualmente salvare i risultati modificati in nuovi documenti XML

Introduzione



Introduzione

- Un componente fondamentale
  - ⇨il parser
  - ⇨effettua l'analisi sintattica del file, riconosce gli elementi sintattici (tag, attributi, valori...)
  - ⇨e consente di estrarre i dati
- Tecnologie per il parsing di XML
  - ⇨DOM
  - ⇨SAX

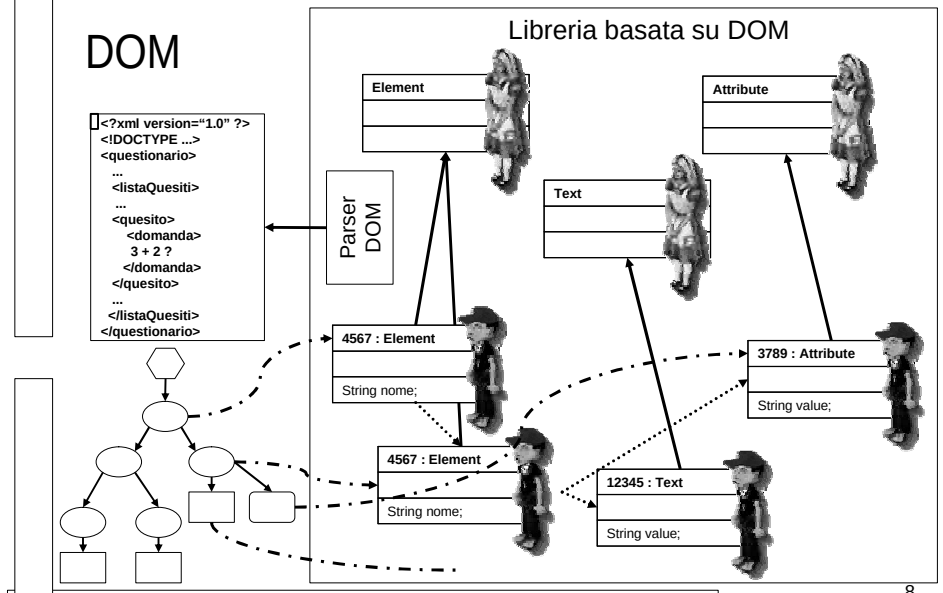
DOM

- Document Object Model (DOM)
  - ⇨standard del consorzio W3C
  - ⇨si tratta di una API, ovvero di un insieme di interfacce per la manipolazione di InfoSet
- Idea alla base di DOM
  - ⇨viene esaminato il documento XML e viene costruito in memoria centrale una rappresentazione del suo InfoSet
  - ⇨sotto forma di oggetti collegati

DOM

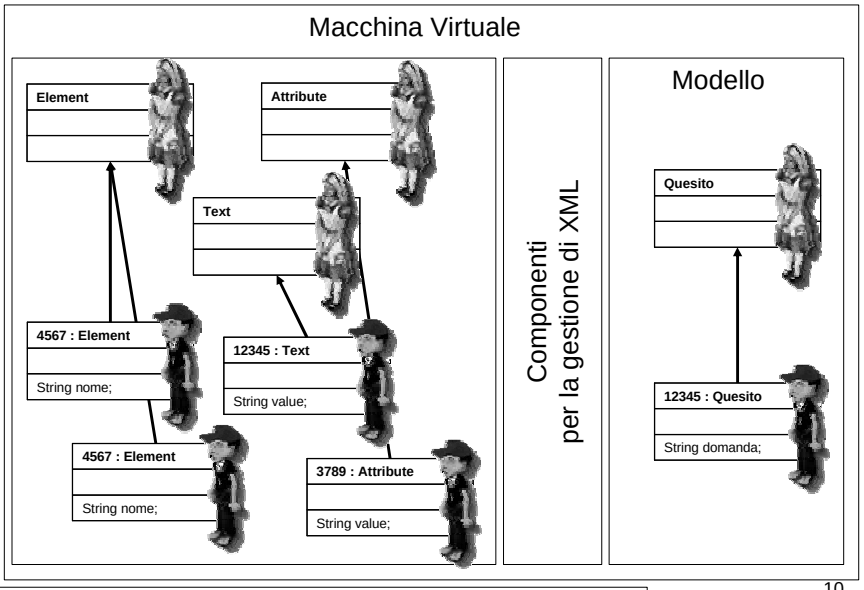
- Nota
  - ⇨gli oggetti costruiti dal parser DOM sono oggetti di tipo generico
  - ⇨Element, Attribute, Node ecc.
  - ⇨non sono quelli della mia logica applicativa, nè potrebbero esserlo
  - ⇨di conseguenza richiederanno per l'utilizzo un ulteriore passo di trasformazione

DOM



DOM

- Dopo aver costruito l'InfoSet
  - ⇨il programmatore ha accesso all'oggetto che rappresenta la radice dell'albero
  - ⇨può visitare l'albero e manipolarlo
  - ⇨tipicamente visita l'albero per costruire, a partire dai nodi, gli oggetti del suo modello
  - ⇨queste operazioni sono normalmente responsabilità dello strato di persistenza



DOM

- Attenzione
  - ⇨molti confondono InfoSet e albero DOM
- Viceversa si tratta di due concetti diversi
  - ⇨InfoSet: modello logico corrispondente all'albero di un documento XML
  - ⇨albero DOM: rappresentazione di un InfoSet attraverso classi ed oggetti di un linguaggio di programmazione

DOM

- Origine di DOM
  - ⇨introdotto originariamente nel browser Netscape 2.0 come API per il linguaggio JavaScript per manipolare pagine HTML
  - ⇨cosiddetto "DOM level 0" – 1995-1996
  - ⇨presto viene introdotta una versione incompatibile nel browser Internet Explorer 3.0

DOM

- La standardizzazione
  - ⇨operata dal consorzio W3C per mettere fine alla diffusione di tecnologie incompatibili
  - ⇨DOM level 1 – standard del 1998
  - ⇨DOM level 2 – standard del 2000
- Caratteristica dello standard
  - ⇨indipendente dal linguaggio
  - ⇨utilizzabile in qualsiasi linguaggio basato su oggetti

DOM

- Contenuto dello standard
  - ⇨la definizione di una serie di interfacce
  - ⇨in un linguaggio astratto per la definizione di interfacce (IDL: interface definition language)
  - ⇨lo standard non fornisce nessuna implementazione
  - ⇨è possibile implementare le interfacce in linguaggi diversi (“binding” per il linguaggio)

DOM

- Le principali interfacce
  - ⇨Document: rappresenta la radice del doc.
  - ⇨Node: antenato della gerarchia dei nodi
  - ⇨NodeList: collezione di nodi (NOTA)
  - ⇨Element: elemento (estende Node)
  - ⇨Attr: attributo (estende Node)
  - ⇨Text: valore (estende Node)

DOM

>> dom.idl

- Per ciascuna interfaccia
  - ⇨sono definite una serie di proprietà e metodi

```
interface Element : Node {
  readonly attribute DOMString tagName;
  DOMString getAttribute(in DOMString name);
  void setAttribute(in DOMString name, in DOMString value) raises(DOMException);
  void removeAttribute(in DOMString name) raises(DOMException);
  Attr getAttributeNode(in DOMString name);
  Attr setAttributeNode(in Attr newAttr) raises(DOMException);
  Attr removeAttributeNode(in Attr oldAttr) raises(DOMException);
  NodeList getElementsByTagName(in DOMString name);
  void normalize();
};
```

DOM

- Altre interfacce
  - ⇨Comment: commento
  - ⇨ProcessingInstruction
  - ⇨CDATASection (estende Text)
- Lo spazio di nomi
  - ⇨org.w3c.dom
- Parser che implementano DOM in Java
  - ⇨ne esistono vari

DOM

- Principali parser basati su DOM per Java
  - ⇨Crimson: basato su X Parser di Sun, sviluppato da Apache Software Foundation
  - ⇨Xerces: successiva implementazione di DOM della Apache Software Foundation
  - ⇨Oracle XML Parser per Java
- Utilizzando questi parser
  - ⇨è possibile costruire un albero di oggetti DOM in Java a partire da un documento XML

SAX

- Simple API for XML (SAX)
  - ⇨rappresenta l'alternativa a DOM
  - ⇨nato con specifico riferimento a Java
  - ⇨http://www.saxproject.org/
  - ⇨è uno standard di fatto, ma non esistono documenti ufficiali che lo standardizzano
  - ⇨di fatto non c'è nemmeno una reale forma di licenza – si tratta di software "public domain"

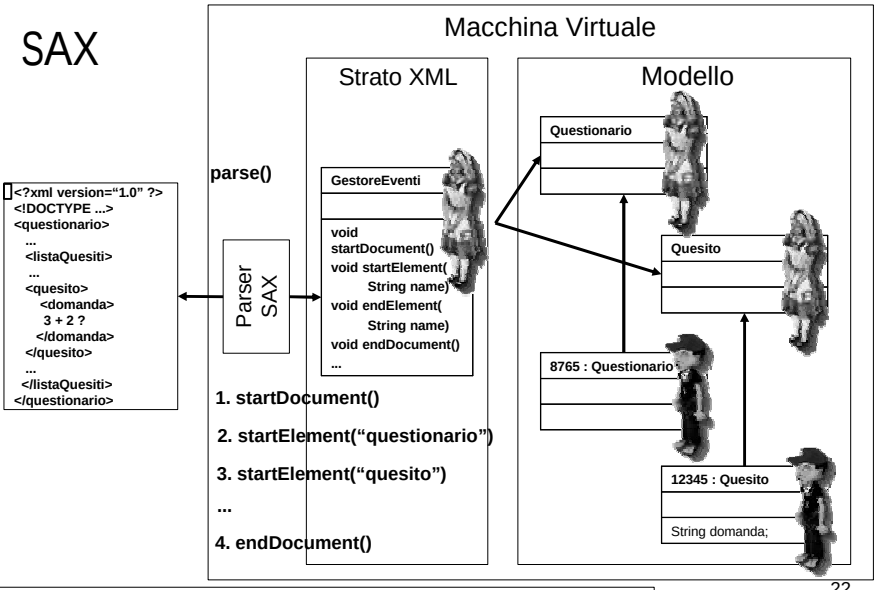
SAX

- La storia di SAX
  - ⇨prima versione SAX1 – 1997-1998
  - ⇨attualmente SAX2 – 1999-2000, "the most complete XML API available anywhere"
- Principale differenza tra SAX e DOM
  - ⇨SAX è una API per il parsing di documenti XML basata sulla programmazione ad eventi

SAX

- Programmazione ad eventi su XML
  - ⇨un parser SAX analizza il documento XML in maniera incrementale
  - ⇨una volta avviata l'analisi sintattica, ogni evento riscontrato durante l'analisi del documento viene segnalato all'applicazione
  - ⇨es: inizio del documento, tag di apertura dell'elemento di nome X, tag di chiusura dell'elemento di nome X, fine del documento

SAX



SAX

- Come DOM
  - ⇨SAX è solo la definizione di un'interfaccia
- Esistono vari parser che la implementano
  - ⇨in effetti tutti i parser che implementano DOM implementano anche SAX
  - ⇨Crimson
  - ⇨Xerces

Differenze tra DOM e SAX

- Principali differenze tra le due API
  - ⇨semplicità di utilizzo
  - ⇨efficienza
- Semplicità di utilizzo
  - ⇨i parser DOM sono molto più semplici da utilizzare perchè costruiscono un albero di oggetti da visitare
  - ⇨i parser SAX sono molto più complessi da usare perchè richiedono la gestione di eventi

Differenze tra DOM e SAX

- Efficienza
  - ⇨d'altro canto DOM costruisce una rappresentazione in memoria centrale di tutto l'InfoSet
  - ⇨viceversa il parsing con SAX è incrementale e non richiede di costruire l'intera rappresentazione dell'InfoSet
  - ⇨di conseguenza per documenti di grandi dimensioni usare SAX è indispensabile

Librerie per XML

- A questo punto
  - ⇨possiamo riassumere il concetto di libreria per XML
- La libreria principali per Java
  - ⇨Java API for XML Programming (JAXP) – fornita a corredo di J2SE
  - ⇨JDOM – strumento open source sviluppato indipendentemente e distribuito su http://www.jdom.org

JAXP

- Java API for XML Processing (JAXP)
  - ⇨libreria fornita a corredo di J2SE
  - ⇨ultima versione: 1.3 in J2SDK 5.0
- Principali funzionalità
  - ⇨consente l'utilizzo di un parser DOM e SAX
  - ⇨supporta DTD e XMLSchema
  - ⇨supporta XPath
  - ⇨supporta XSLT

JAXP

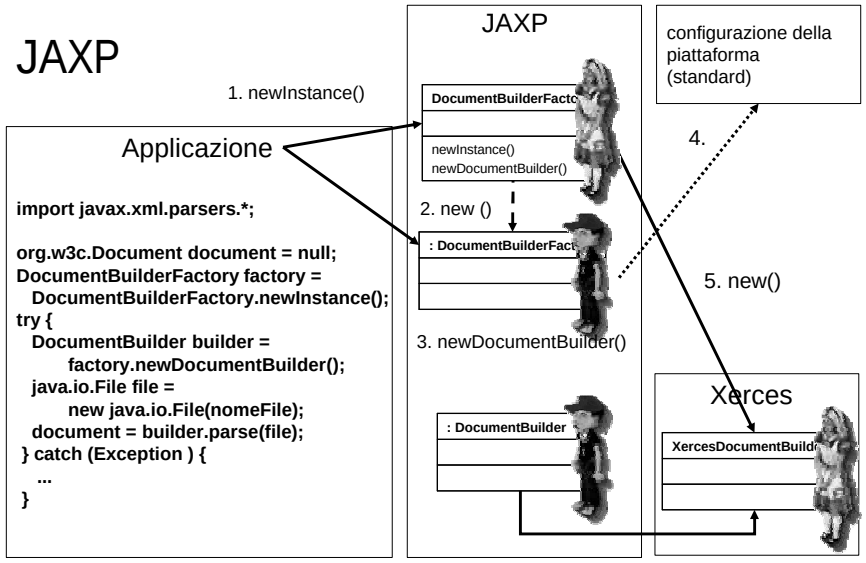
ATTENZIONE
all'approccio di JAXP
rispetto agli strumenti

- L'approccio di JAXP
  - ⇨tipico approccio delle librerie Java
  - ⇨indipendenza rispetto agli strumenti
- In questo caso
  - ⇨JAXP può utilizzare parser diversi
  - ⇨il parser DOM e SAX standard di JAXP 1.1 era Crimson
  - ⇨il parser standard di JAXP 1.3 è Xerces
  - ⇨l'utente può scegliere un parser diverso

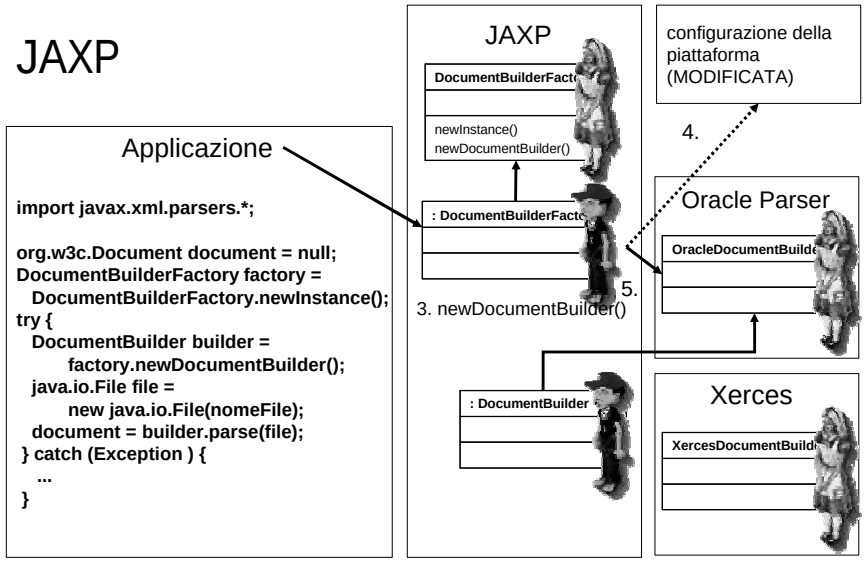
JAXP

- Per ottenere un riferimento al parser
  - ⇨lo sviluppatore utilizza la classe javax.xml.parsers.DocumentBuilderFactory
  - ⇨restituisce un riferimento ad un oggetto di tipo DocumentBuilder che rappresenta il parser
  - ⇨a seconda della configurazione specificata il DocumentBuilder restituito è diverso

JAXP



JAXP



JAXP

- Procedura per decidere il parser
  - ⇨per cominciare la macchina virtuale verifica se è stata specificata la proprietà di sistema javax.xml.parsers.DocumentBuilderFactory
  - ⇨altrimenti cerca e consulta il file jaxp.properties contenuto in %JRE_HOME%\lib
  - ⇨altrimenti utilizza il parser standard (org.apache.crimson oppure org.apache.xerces a seconda della versione)

JAXP

- Analogamente
  - per il motore di trasformazione per XSLT
  - il motore standard è org.apache.Xalan
  - l'utente può utilizzare motori diversi
- Attenzione
  - questo fatto provoca a volta problemi dovuti alla presenza nel classpath di parser diversi
  - molti pacchetti (es: IDE) usano parser per XML

JAXP

- La caratteristica fondamentale di JAXP
  - è una implementazione fedele di DOM
  - riproduce esattamente le interfacce di DOM
- Conseguenze
  - alcune operazioni risultano macchinose e poco naturali per un programmatore Java
  - es: chiedendo la lista dei figli di un elemento si ottiene un oggetto di tipo org.w3c.dom.NodeList e non una ArrayList

JAXP

- L'altra caratteristica di JAXP
  - ⇨è una collezione di package
  - ⇨molto complessa
- I package fondamentali di JAXP

javax.xml	javax.xml.xpath
javax.xml.parsers	javax.xml.validation
javax.xml.transform	javax.xml.namespace
org.w3c.dom	javax.xml.datatype
org.xml.sax	...

Riassumendo

- Introduzione
- Document Object Model (DOM)
- Simple API for XML (SAX)
- Differenze tra DOM e SAX
- Librerie per XML
- JAXP

Termini della Licenza