XML

Caratteristiche

- XML è un **metalinguaggio**, ovvero uno strumento che aiuta a definire un linguaggio, facendogli da base sintattica. Su XML si possono definire N linguaggi diverse
- È **indipendente dai dispositivi** e questo lo rende standard de facto per lo ***scambio di dati*** in applicazioni middleware.
- Questo linguaggio permette la **rappresentazione di qualsiasi tipo di documento** e porta con se numerose ***librerie open source*** per la manipolazione dei dati in formato XML

I Vantaggi

Permette agli sviluppatori di **creare strutture ad-hoc** per contenere le informazioni ed inoltre il fatto che sia un linguaggio strutturato è possibile **applicarci sopra delle validazioni**. Essendo text-based è anche **Human Readable** e supporta unicode.
Altro vantaggio da non sottovalutare è quello di poter **trasmettere dati** usando il protocollo HTTP.

Gli Svantaggi

Proprio a causa della sua struttura e leggibilità, XML **provoca ridondanza**, il che porta i documenti XML ad essere più ingombranti rispetto a quelli in formato binario. Inoltre i parser XML non sono veloci come quelli che vengono appositamente scritti per formati specifici, il che ne va ad intaccare le prestazioni.

I file XML

XML definisce un'insieme di regole sintattiche che permettono di descrivere formalmente un ***linguaggio di markup***. È possibile anche **andare a creare i nostri tag**. Non vengono fissate regole particolari (che potremmo fissare noi) ma piuttosto **regole comuni** per eseguire correttamente il parsing del file.

1. Il nome di un elemento XML deve iniziare con una lettera o un carattere di sottolineatura
2. I caratteri successivi possono essere numeri, trattini, punti o altri caratteri di sottolineatura
3. Non c'è limite al numero di caratteri per elemento
4. È ammesso specificare i due punti solo per specificare i namespace
5. Gli spazi vuoti e i simboli non sono ammessi nei nomi degli elementi

6. Gli elementi non vuoti devono avere un tag iniziale ed uno finale
7. I tag XML devono essere annidati correttamente

Bisogna ricordarsi che è **linguaggio case sensitive**.

La struttura del file XML rimanda ***alla struttura di un albero***, ed è quindi possibile definire **'rapporti di parentela' tra gli elementi**, dove più ci si addentra all'interno dei nodi, più si vanno a visitare i discendenti. Questo aspetto è fondamentale nella ricerca degli elementi in un file XML, che vengono cercati esattamente come in un albero.
È quindi possibile individuare **Radice, Figli, Padre, fratelli, discendenti e predecessori** ed anche rappresentare la vera e propria struttura come un albero, il che ci permette di visitare i file proprio come se fossero degli alberi.

Gli Attributi

Sui tag XML è anche possibile andare a definire degli attributi, che forniscono informazioni aggiuntive su un elemento, composti da **un numero ed unvalore** racchiuso tra virgolette proprio in questa maniera:

```
<!-- Questo è un commento in XML -->
<tag> <!-- Apertura Tag -->
    <automobile marca="Fiat" modello="Punto">
    </automobile>
    <!-- Uso degli attributi -->
    <automobile marca="Fiat" modello="Punto">
    </automobile>
</tag> <!-- Chiusura Tag -->
```

Ci sono anche dei **caratteri speciali** che vengono trattati non come caratteri, ma come dei "codici" che sono:

- & che si traduce in &
- < che si traduce in <
- > che si traduce in >
- " che si traduce in "
  Questo serve in caso ad esempio di vogliano utilizzare i simboli di maggiore all'interno di un tag senza andare in conflitto con la sintassi di quest ultimo. In questo caso allora utilizzi >

Un tag da tenere a mente è quello della **sezione CDATA**, ovvero un blocco di testo che viene considerato sempre come tale, anche se dovesse contenere codice XML. Per indicare la sezione CDATA basta fare così:

```
<codice>
    <![CDATA[
        <libro>
            <capitolo>
            </capitolo>
        </libro>
    ]]>
</codice>
```

Il **prologo** dell'XML indica che si tratta di un file XML, è **obbligatorio** è composto così:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

con l'attributo **version obbligatorio** che indica la versione di XML utilizzata, **encoding opzionale** ed indica la codifica dei caratteri usata (di default è UTF-8 o 16) e **standalone opzionale** che se impostato a `yes` indica che il file non fa riferimento ad altri file esterni (di default no).

Namespace

Immaginate che si hanno due applicazioni che devono essere messere in comunicazione tra loro e vanno a scrivere sullo stesso file XML lo stesso elemento, rendendo **impossibile andare a distinguere quale software ha scritto cosa**.
Usiamo quindi i **Namespace**, ovvero nomi che possono essere messi prima di ogni tag ad indicare chi ha specificato quel tag. L'utlizzo all'interno del tag è il seguente: `<f:nometag>`.

Dobbiamo però definirlo nel file:

```
<libro xmlns="http://esempio.com/libri"
       xmlns:f="http://esempio.com/metadata">
```

Possono anche **essere legati agli attributi**. La buona prassi sarebbe quella di andare a inserire all'interno del link del namespace lo schema del mio file XML.

È anche possibile **rappresentare immagini con il tag svg**, **rappresentare processi con bpmn**.

Controllo del documento XML

Ci sono due diversi livelli di controllo del documento che danno due livelli di correttezza:

- XML ben formato, ovvero che aderisce allo standard
- XML valido, che usa uno schema considerato valido

Librerie per lavorare con XML

- JDOM
- Oppure se si vuole tenere il dtd con le librerie ***dom e sax***, dobbiamo dichiararlo come file esterno.

---

xPath

Una delle tecnologie più utilizzate tutt'oggi e permette di **ricercare parti di un documento XML**, andando ad operare sulla ***struttura logica del documento***, utilizzando una sintassi non XML accettabile all'interno di URI e dispone di primitive per **eseguire semplice operazioni** su stringhe, numeri e valori booleani in maniera molto snella.

- Il documento XMK viene trattato come un albero ed ogni elemento, commento, attributo o stringa viene considerata **nodo dell'albero**.
- Si lavora sui concetti di:
  - **Location Path**: specifica come ***spostarsi tra i nodi dell'albero***.
  - **Location Step**: ovvero la ***sequenza di passi*** di cui si compone la Location Path, separati da `/` letti da sinistra a destra.
    -Il location step ha 3 parti: asse, nodo di test e da 0 o più predicati.

L'asse - axis:test[pred1][pred2]...[predN]

Indica la location path e permette di scegliere tra **figli del nodo corrente**, ma anche tra una serie di altri insiemi, gli assi sono: `self, parent, child, precedent-sibling, following-sibling, ancestor, descendant, preceding, following`.