Gestione dei file: XML

Rachele Sprugnoli

rachele.sprugnoli@unipr.it

UNIVERSITÀ
DI PARMA

XML

- eXtensible Markup Language
- TEI: applicazione dell'XML per la codifica di generi testuali di interesse umanistico, es. https://tei-c.org/release/doc/tei-p5-doc/it/html/ref-persName.html

Capitolo 29

Qui, tra i poveri spaventati troviamo persone di nostra conoscenza.

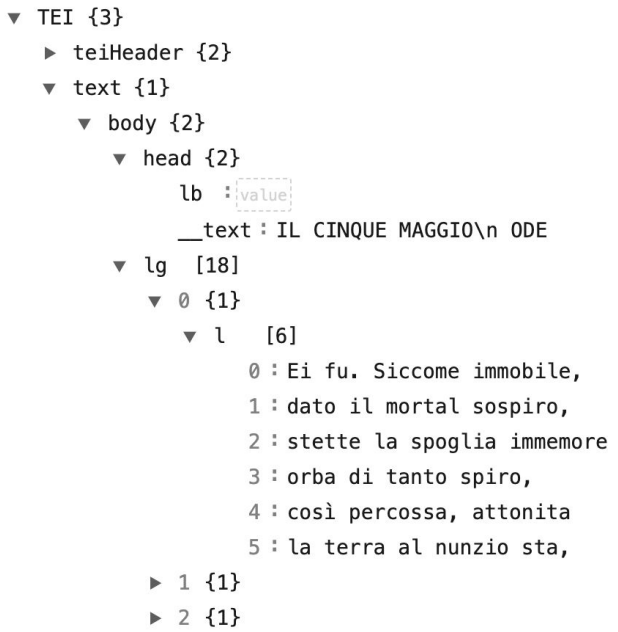
Chi non ha visto don Abbondio, il giorno che si sparsero tutte in una volta le notizie della calata dell'esercito, del suo avvicinarsi, e de' suoi portamenti, non sa bene cosa sia impiccio e spavento.

```
<head>Capitolo 29</head>
<p n="1">Qui, tra i poveri spaventati troviamo persone di nostra conoscenza.</p>
<p n="2">Chi non ha visto <persName>don Abbondio</persName>, il giorno che si sparsero tutte in una volta le notizie della calata dell'esercito, del suo avvicinarsi, e de' suoi portamenti, non sa bene cosa sia impiccio e spavento.</p>
```

Parsing

- Atto di scomporre il file XML nelle sue parti componenti
- Leggere un file/stringa e ottenerne il contenuto in base alla sua struttura (provare online: https://codebeautify.org/xml-parser-online)

```
</teiHeader>
<text>
   <body>
      <head>IL CINQUE MAGGIO<lb/> ODE</head>
         <lg>
            <l>Ei fu. Siccome immobile,</l>
            <l>dato il mortal sospiro,</l>
            <l>stette la spoglia immemore</l>
            <l>orba di tanto spiro,</l>
            <l>così percossa, attonita</l>
            <l>la terra al nunzio sta,</l>
         </lg>
```



Struttura ad albero

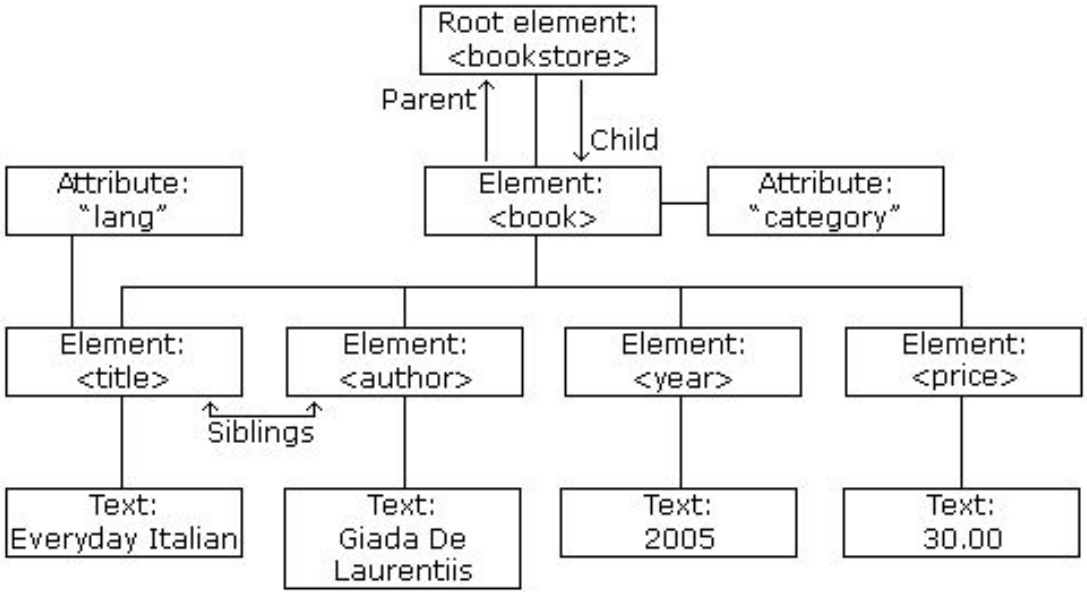


Immagine da: https://www.w3schools.com/xml/xml_tree.asp

Struttura ad albero

```
<body>
        <div type="sezione" n="1" xml:id="sezione1">
          <head>Traccia del Discorso sulla Moralità delle Opere Drammatiche</head>
          <p><milestone unit="comma" n="1"/> V'ha due modi di considerare le quistioni morali:</p>
          <p>Prescindendo dal Vangelo.</p>
          <p>Ponendolo per fondamento.</p>
…
</div>
</body>
```

Beautiful Soup

- Libreria Python per cercare, estrarre e modificare dati da file HTML e XML
- Supportato da Python >= 3.6
- Usa vari parser esterni:
  - `html.parser`, `lxml` e `html5lib` per HTML
  - `lxml-xml` e `xml` per XML
- Usiamo la versione 4: https://beautiful-soup-4.readthedocs.io/en/latest/#

Beautiful Soup: esempio

```
from bs4 import BeautifulSoup as bs
with open("nome_file.xml", 'r') as file:
    soup = bs(file, 'nome_parser')
print(soup)
---
from bs4 import BeautifulSoup as bs
with open("il-cinque-maggio-ode.xml", 'r') as tei:
  soup = bs(tei, 'lxml-xml')
print(soup)
```

Beautiful Soup: usi

- Cercare:
  - `soup.find("tag")` → trova la PRIMA occorrenza
  - `soup.tag` → trova la PRIMA occorrenza del tag
  - `soup.find_all("tag")` → trova TUTTE le occorrenze
  - `soup.tag.find_all()` → trova i discendenti del tag
  - `soup.tag.text` → trova il contenuto testuale del tag
  - `soup.tag.contents` → trova il contenuto testuale del tag
  - `soup.tag.get("att")` → trova il valore dell'attributo (att) del tag

Beautiful Soup: usi

- Cercare:
  - `soup.tag.parent` → trova il genitore del tag
  - `soup.tag.attrs` → trova gli attributi del tag, l'output è un dizionario, si possono usare i metodi dei dizionari:
    - `soup.tag.attrs.keys()` → estrae solo il nome dell'attributo
    - `soup.tag.attrs.values()` → estrae il valore dell'attributo

Beautiful Soup: usi

- Cercare:
  - `soup.tag.next_sibling` → trova il fratello subito successivo al tag
  - `soup.tag.previous_sibling` → trova il fratello subito precedente
  - `soup.tag.next_siblings` → trova tutti i fratelli successivi al tag
  - `soup.tag.previous_siblings` → trova tutti i fratelli precedenti

Beautiful Soup: usi

- Se voglio cercare su più elementi multipli uso l'iterazione:

```
for sibling in soup.tag.next_siblings
print(sibling)
```

```
for child in soup.tag.children
print(child)
```

Beautiful Soup: usi

- Modificare:
  - `soup.tag.name = "nuovo_nome"` → cambia il nome del tag
  - `soup.tag["attr"] = "nuovo_valore"` → cambia il valore dell'attributo attr, se attr non esiste viene creato
  - `soup.tag.string = "nuovo_contenuto"` → cambia il contenuto del tag

Beautiful Soup: usi

- Aggiungere:
  - `soup.new_tag("nome_tag", attr="valore")` → definisce un nuovo tag con un certo nome e un attributo con un certo valore
  - `soup.tag.append(new_tag)` → aggiunge il nuovo tag alla fine del tag genitore
  - `soup.tag.insert(1, new_tag)` → aggiunge il nuovo tag alla posizione 1
  - `soup.tag.insert_before(new_tag)` → aggiunge il nuovo tag prima del tag
  - `soup.tag.insert_after(new_tag)` → aggiunge il nuovo tag dopo il tag

Beautiful Soup: usi

- Rimuovere:
  - `del tag["att"]` → rimuove l'attributo att del tag
  - `soup.tag.decompose` → rimuove il tag e il suo contenuto

Rimuovere tutte le annotazioni

- Usiamo la libreria lxml: https://lxml.de/
  - funzione `tostring()`
  - argomento `method="text"`

- EXAMPLE:

```
etree.tostring(file, encoding="utf8", method="text")
```

Un po’ di pratica

- Lezione6.ipynb

  https://colab.research.google.com/drive/1WEKmMg0un20vfp29Qu94p40tHNrgnLL6?usp=sharing

Esercizio 1

- Nel file della-moralita-delle-opere-tragiche.xml:
  - Trovare tutti gli elementi <hi>
  - Contare gli elementi <hi>
  - Stampare solo il contenuto di ogni <hi>

Esercizio 2

- Nel file della-moralita-delle-opere-tragiche.xml:
  - Aggiungere il tag <distributor> con contenuto “Università di Bologna” alla fine dall’elemento <publicationStmt>
  - Salvare il tei modificato in un altro file

Soluzioni esercizi

- https://colab.research.google.com/drive/1ZGTZVigp7T-2ckw4AA6Vb6o1rHzGkuPF?usp=sharing