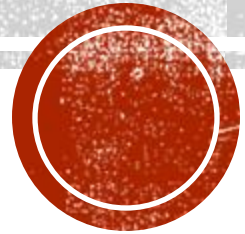
XML PT.1

Tecnologie Web e Laboratorio A.A. 2020-21 - Università degli Studi di Udine



Andrea Urgolo - andrea.urgolo@uniud.it

INDICE

EDITOR

Usate l'editor che preferite:

- **Brackets** http://brackets.io
- **Visual Studio Code** https://code.visualstudio.com
- **Atom** https://atom.io

Editor online:

- https://jsbin.com
- https://jsfiddle.net
- https://codesandbox.io
- https://codeanywhere.com
- ecc.

MATERIALE

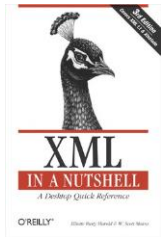
Documentazione ufficiale

- Raccomandazione del W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition) https://www.w3.org/TR/xml

Siti con materiale didattico:

- https://w3schools.com/xml
- http://users.dimi.uniud.it/~massimo.franceschet/caffe-xml

Libro di testo suggerito:



XML in a Nutshell, 3rd Edition by Elliotte Rusty Harold, W. Scott Means, Pub.:O'Reilly Media, Inc. ISBN: 9780596007645

INTRODUZIONE

XML – Tecnologie Web
e Laboratorio

DIVERSI LINGUAGGI

PER DEFINIRE PROCEDURE/ALGORITMI	PER DEFINIRE E STRUTTURARE I DATI	PER VISUALIZZZARE I DATI
Python	XML	CSS
Java	HTML	XSL
PHP		XSLT
Javascript		
C#		
ecc.		

COSA SI INTENDE PER MARKUP

- L'uso del markup permette di strutturare un un file in formato testo (documento) in componenti logiche, dette **elementi**, e di "etichettarle" opportunamente
- Le etichette (i *nomi* degli elementi) specificano il tipo di dato che una certa componente logica rappresenta
- Le etichette vengono inserite nel documento stesso come speciali sequenze di caratteri, dette *markup tag* o semplicemente *tag*

ESEMPIO 1: DOCUMENTO SENZA MARKUP

Jack Smith
513-555-3465
jsmith@email.com
John Doe
34 Fountain Square Plaza
Cincinnati, OH 45202
US
513-555-8889 (preferred)
513-555-7098
jdoe@email.com

ESEMPIO 1: DOCUMENTO CON MARKUP

```
<address-book>
  <contact>
    <name>Jack Smith</name>
    <tel>513-555-3465</tel>
    <email address='jsmith@email.com'/>
  </contact>

  <contact>
    <name>John Doe</name>
    <address>34 Fountain Square Plaza Cincinnati, OH
             45202 US</address>
    <tel>513-555-8889</tel>
    <tel>513-555-7098</tel>
    <email address='jdoe@email.com' />
  </contact>
</address-book>
```

ESEMPIO 2: DOCUMENTO SENZA MARKUP

Internet

1. il World Wide Web

Il World Wide Web (cui spesso ci si riferisce semplicemente con Web o con l'acronimo WWW) è stata cronologicamente l'ultima funzionalità di Internet ad essere sviluppata.

...

2. La posta elettronica

...

ESEMPIO 2: DOCUMENTO CON MARKUP

```
<libro>
 <capitolo><titolo>Internet</titolo>
 <sezione><titolo> 1. il World Wide Web </titolo>
 <paragrafo>Il World Wide Web (cui spesso ci si riferisce
 semplicemente con Web o con l'acronimo WWW) è stata
 cronologicamente l'ultima funzionalità di Internet ad essere
 sviluppata. </paragrafo>
 ...
 </sezione>
 <sezione><titolo> 2. La posta elettronica</titolo>
 ...
 </sezione>
 ...
 </capitolo>
</libro>
```

VANTAGGI DEI LINGUAGGI DI MARKUP

- Un programma può facilmente distinguere tra le componenti logiche di un documento con markup

  es.
  - per scopi di visualizzazione: elementi diversi possono essere visualizzati a schermo o stampati usando stili tipografici diversi
  - elaborazione e interrogazione dei dati rappresentati nel documento

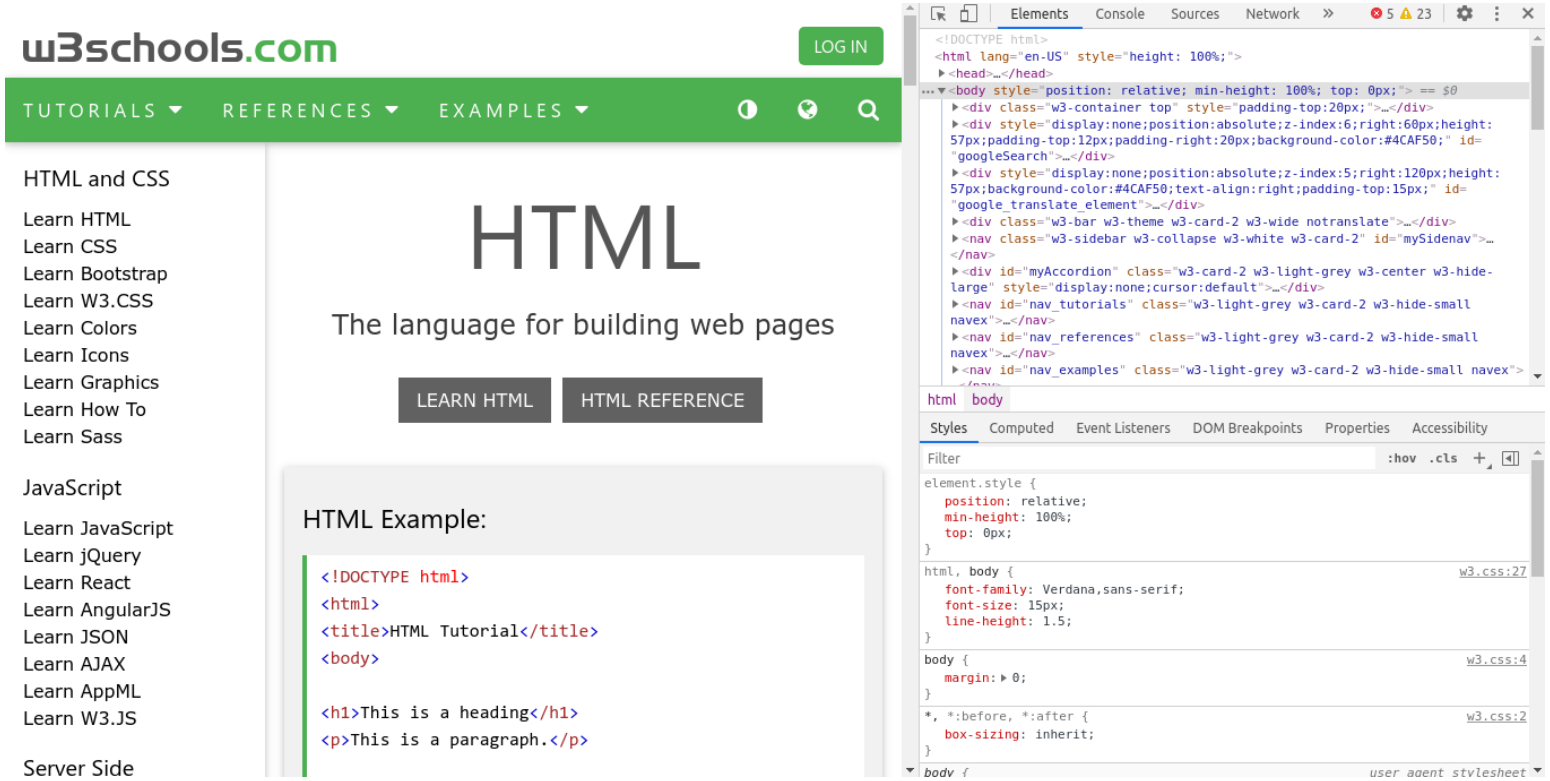
- Costituiscono un formato semplice per lo scambio di dati (in formato testo)

- Sono comprensibili anche agli umani

UTILIZZI DEI LINGUAGGI DI MARKUP

- Dagli anni '70 fino agli anni '90: principalmente, nel trattamento di testi e nell'editoria elettronica
  - es TeX, RTF (Rich Text Format)
- Negli anni '90: pagine Web (HTML)
- Dalla fine degli anni '90: pagine Web, news, MMS, transazioni finanziarie, scambio di dati, disegni tecnici, dati geografici, fogli elettronici, documenti multimediali, oggetti e ambienti tridimensionali, …

ALCUNI ESEMPI - 1



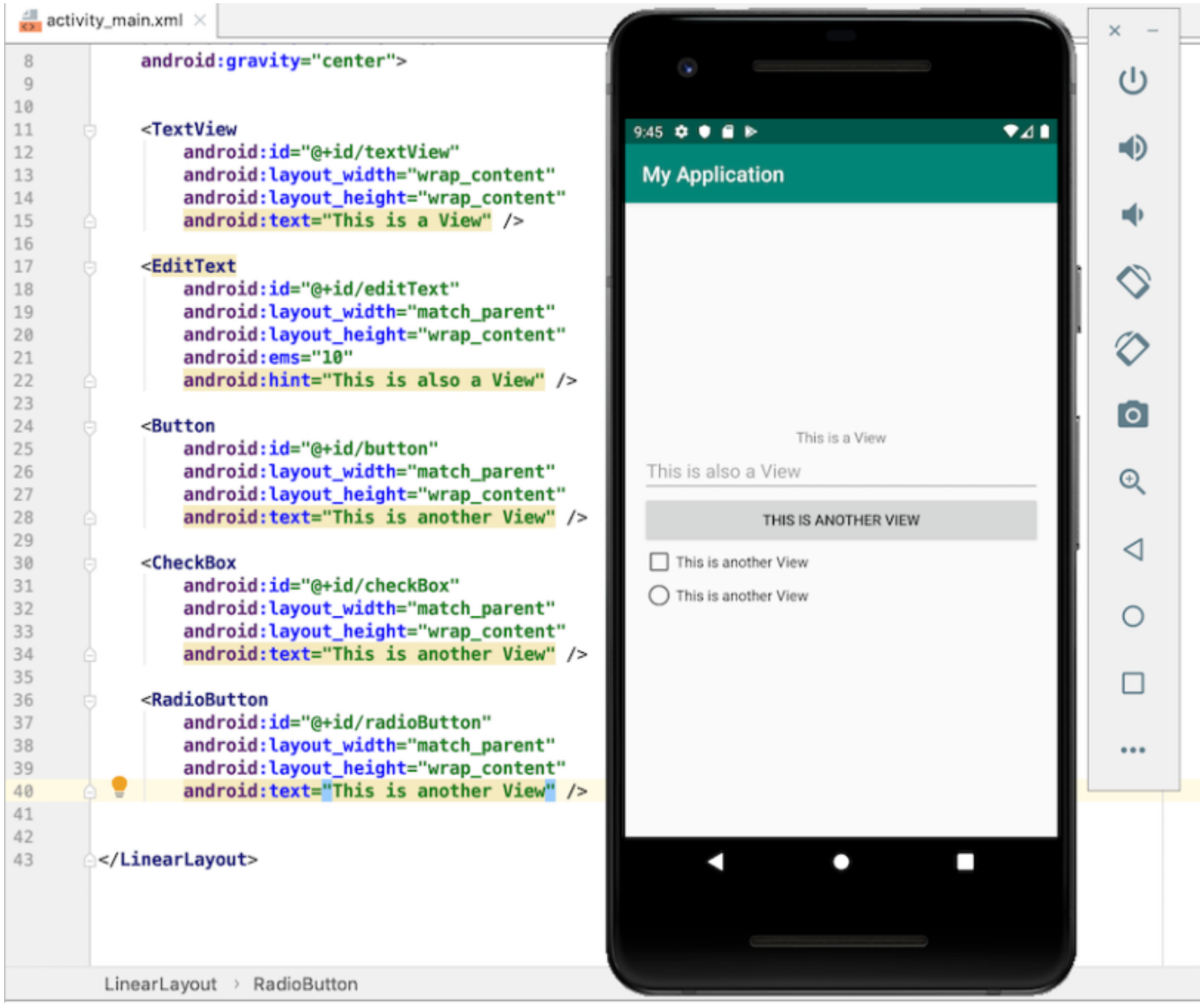
- *HyperText Markup Language* (**HTML**): linguaggio di markup per la definizione di ipertesti
  - nasce nel 1993, e' una delle basi del WWW
  - nelle prime versioni markup tag impiegati anche per controllare la visualizzazione del documento
  - HTML5: tag strutturali (presentazione con fogli di stile CSS)

ALCUNI ESEMPI - 2

- Scalable Vector Graphics (**SVG**): linguaggio di markup per la definizione di grafica bidimensionale
  - www.w3.org/Graphics/SVG/
- Synchronized Multimedia Integration Language (**SMIL**): linguaggio di markup per la definizione di presentazioni multimediali
  - www.w3.org/AudioVideo/
- Extensible 3D (**X3D**): linguaggio di markup per la definizione di oggetti e ambienti tridimensionali
  - www.web3d.org/

ALCUNI ESEMPI - 3

Definizione GUI (app Android)



COS'È XML

XML è l'acronimo di **eXtensible Markup Language**

- **linguaggio formale** definito tramite un insieme di regole formali (una grammatica) che permettono di specificare come comporre un documento
- informazioni sul significato del contenuto presente nel documento vengono fornite tramite annotazioni realizzate con codice di **markup** sintatticamente distinguibile
- l'insieme dei tag di marcatura può essere **esteso** per soddisfare diverse esigenze e adattarsi a diversi domini applicativi

XML è un **meta-linguaggio** di markup, ovvero un linguaggio per la costruzione di linguaggi di markup più specifici (ad es. XHTML)

COSA NON È XML

XML **non** è un

- **linguaggio di presentazione** come SMIL, SVG, HTML (prime versioni): il codice di markup utilizzato nei documenti XML definisce il significato del contenuto, non dice nulla sullo stile con cui va presentato
- **linguaggio di programmazione** come C/C++, C#, Java, Python**:** un documento XML non effettua nessun tipo di computazione
- **protocollo di trasmissione** come HTTP: XML non dà indicazioni su come trasferire informazioni attraverso la rete
- **database management system** come MySQL, PostgreSQL, Oracle, MS SQL Server: XML non fornisce funzionalità per l'archiviazione e il recupero di dati

STORIA

- XML è un discendente dello Standard Generalized Markup Language (SGML) inventato da Charles F. Goldfarb, Ed Mosher e Ray Lorie dell'IBM negli anni '70 e diventato standard (ISO 8879) nel 1986.
- SGML è un linguaggio di markup semantico e strutturale per documenti testuali estremamente complesso ed espressivo. Impiegato con successo dalle forze armate e dal governo degli Stati Uniti, nel settore aerospaziale e in altri domini che richiedevano metodi per gestire in modo efficiente documenti tecnici lunghi decine di migliaia di pagine.
- In 1996, Jon Bosak, Tim Bray, C. M. Sperberg-McQueen, James Clark, e molti altri iniziarono a lavorare su una versione "lite" di SGML che manteneva la maggior parte del potere espressivo di SGML, tagliando quelle caratteristiche che si erano dimostrate ridondanti, troppo complicate da implementare, confuse per gli utenti finali o semplicemente non utili.
- Come risultato, nel febbraio 1998, le specifiche di XML 1.0 divennero una raccomandazione ufficiale del W3C.
- Ben presto ci si accorse che XML non era limitato al solo contesto Web ma era qualcosa di più: uno strumento che poteva essere impiegato in vari contesti, dalla definizione della struttura di documenti, allo scambio delle informazioni tra sistemi diversi, dalla rappresentazione di immagini alla definizione di formati di dati.

ELEMENTI

XML – Tecnologie Web
e Laboratorio

ELEMENTI

- Sono "contenitori" che incapsulano altri elementi o dati intesi come stringhe di caratteri (character data)
  - **es.:** `<address><street>... </street></address>`
  - **es.:** `<tel>513-555-7098</tel>`
- Un elemento ha un **nome** e un **contenuto**
  - nel secondo es.: nome: tel, contenuto: 513-555-7098
- Il contenuto è delimitato da un **tag di inizio** e da un **tag di fine**
- Il tag di inizio contiene il nome ircondato da parentesi angolari (< >)
- il tag di fine contiene a sua volta il nome tra parentesi angolari, ma preceduto da uno "slash" (/)

NOMI VALIDI PER GLI ELEMENTI

- Devono iniziare con una lettera o con _ (underscore)
- Gli altri caratteri del nome possono essere lettere, cifre, _ , . e - (gli spazi non sono ammessi)
- Non possono iniziare con la stringa "xml"
- Sono case sensitive, cioe' scrivere in maiuscolo o minuscolo fa differenza
  - es.: <address> e <ADDRESS> sono due elementi diversi

ATTRIBUTI

XML – Tecnologie Web e Laboratorio

COSA SONO GLI ATTRIBUTI

- Informazioni aggiuntive che vengono associate agli elementi
- Hanno un **nome** e un **valore** (per il nome valgono le regole viste nel caso degli elementi)
- Vengono inseriti all'interno del tag di inizio, dopo il nome dell'elemento
  - es.: <tel preferred="true">513-555-8889</tel>
  - nome dell'attributo: preferred
  - valore dell'attributo: true

COSA SONO GLI ATTRIBUTI - 2

- Per ogni elemento si possono inserire uno o piu' attributi (separati da spazi)
  - es.: <tel preferred="true" location="home">513-555-8889</tel>
- Il valore puo' essere indicato tra due doppi apici ("...")
  - es.: <tel preferred="true">
- oppure tra due apici singoli ('...')
  - es.: <tel preferred='true'>

ESERCIZIO

Scrivere un documento XML che rappresenti una playlist musicale

ESERCIZIO (POSSIBILE SOLUZIONE)

```
<playlist>
<track id="t01" genere="rock">
  <title>Time</title>
  <artist>Pink Floyd</artist>
  <album>The Dark Side of the Moon</album>
  <file src="01-time.mp3">
    <encoding>MP3</encoding><bitrate>160kbps</bitrate>
  </file>
</track>

<track id="t02">
  ...
</track>
...
</playlist>
```

ENTITÀ

XML – Tecnologie Web e Laboratorio

COSA SONO LE ENTITÀ

- Un'**entità** è un **riferimento ad una sequenza di caratteri**
- Ha un **nome** e un **contenuto** (una sequenza di caratteri)
- Per inserire un'entità si utilizza il codice **&nome-entità;** (entity reference), cioè il nome dell'entità tra '**&**' e '**;**'
- Si può immettere **solo** nel contenuto degli elementi e nel valore degli attributi
- È equivalente ad inserire nella medesima posizione il contenuto riferito dall'entità
  - es.: assumiamo di aver definito l'entità **us** con contentuto **United States**. Allora sono equivalenti
    - `<state>&us;</state>`
    - `<state>United States</state>`

ENTITÀ PREDEFINITE

Caratteri impiegati nel codice di markup (in questo caso è obbligatorio usare l'entità)

Riferimento all'entità	Contenuto dell'entità
&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

ENTITÀ PREDEFINITE - 2

Riferimenti generici a caratteri:

- ogni carattere può essere sostituito dal suo codice UNICODE (sia in formato decimale che esadecimale)
- i riferimenti che iniziano con &#x usano il formato esadecimale, i riferimenti che iniziano con &# usano il formato decimale
  - es.:

```
<scuola>Universit&#x00E0;</scuola>
```

è equivalente a

```
<scuola>Università</scuola>
```

Lista codici UNICODE in formato esadecimale: http://www.unicode.org/charts

ALTRE COMPONENTI

XML – Tecnologie Web e Laboratorio

DICHIARAZIONE XML

```
<?xml version="1.0"?>
```

La dichiarazione XML identifica il documento come un documento XML
La forma più semplice di dichiarazione XML è **<?xml version="1.0"?>**

Non è obbligatoria, ma se presente deve iniziare nel **primo carattere della prima riga del documento**

È consigliato inserirla sempre

Attributi opzionali

- **encoding:** specifica la codifica dei caratteri adottata (default: UTF-8)
- **standalone** (*yes* | *no*)*:* se *yes*, il documento è auto-contenuto, il che significa che tutti i suoi valori sono presenti in esso (dichiarazioni entità e valori di default); se *no* (default), possono essere specificati valori del documento in un DTD esterno.

Es.: `<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>`

SEZIONI CDATA

- Le sezioni **CDATA** sono parti di documento XML delimitate da **<![CDATA[** e **]]>**
- Una sezione CDATA in un documento XML fa ignorare tutti i caratteri di markup ad eccezione di **]]>** (delimitatore di chiusura della sezione)
- Utile, ad esempio, per inserire un documento XML in un documento XML oppure se il contenuto presenta troppi caratteri di markup da sostituire con riferimenti a entità
- Problema: come inserisco **]]>** in una sezione CDATA? (sol.: uso riferimento a entità >)

ESEMPIO: CONTENUTO CON MARKUP

```
<?xml version="1.0"?>
<example>
  <?xml version="1.0"?>
  <entry>
    <name>John Doe</name>
    <email href="mailto:jdoe@email.com" />
  </entry>
</example>
```

Come inserire il contenuto in grasseto?

ESEMPIO: SEZIONE CDATA

```
<?xml version="1.0"?>
<example>
  <![CDATA[
  <?xml version="1.0"?>
  <entry>
    <name>John Doe</name>
    <email href="mailto:jdoe@email.com" />
  </entry>
  ]]>
</example>
```

COMMENTI

- È possibile inserire dei commenti in un documento XML (ovvero, del **testo che NON fa parte del documento**, ma è utile a chi lo ha scritto o lo deve modificare)

- La sintassi dei commenti è:

```
<!-- Questo è un commento -->
```

- Attenzione: i commenti non possono essere inseriti all'interno del markup!

BEN FORMATEZZA

XML – Tecnologie Web e Laboratorio

DOCUMENTO XML BEN FORMATO

Un documento XML si dice **ben formato** se:

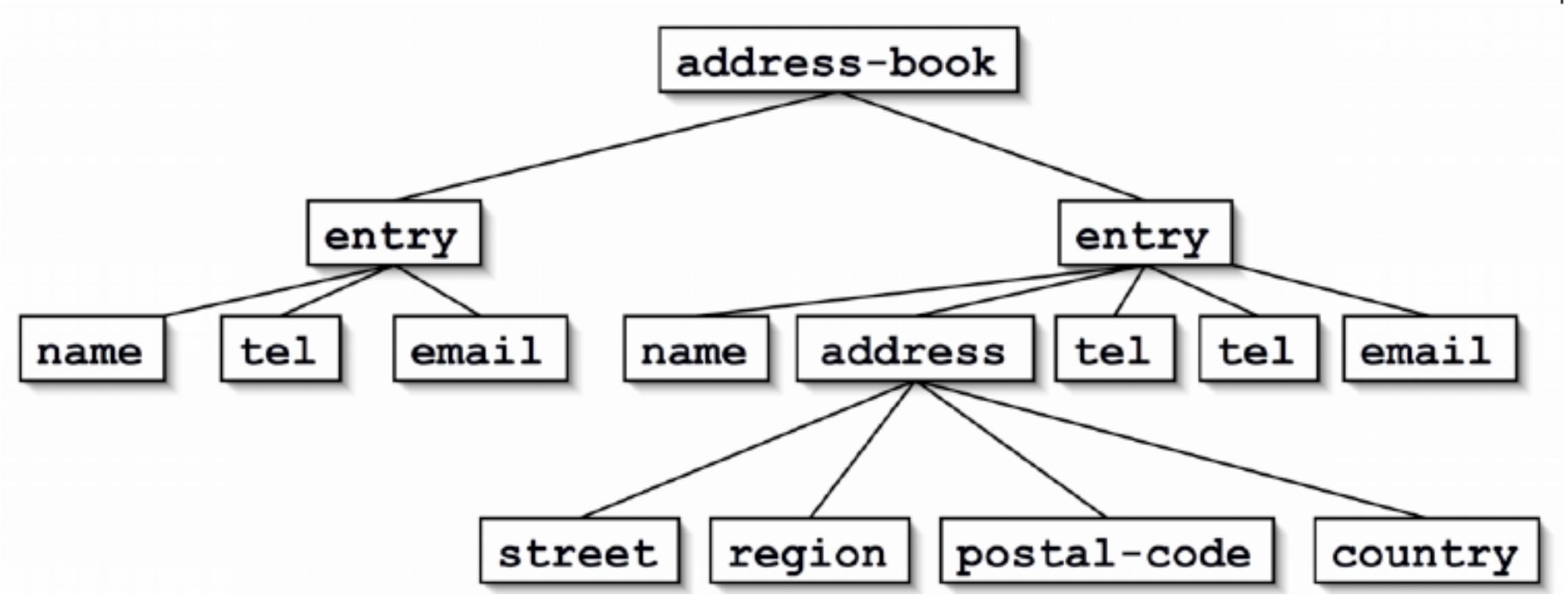
- ogni tag di inizio ha un corrispettivo tag di fine
- i valori degli attributi sono delimitati da apici o doppi apici
- un elemento non può avere due attributi con lo stesso nome
- gli elementi sono correttamente annidati
- c'è esattamente **un** "elemento radice"
- vale quanto detto in precedenza: validità nomi, escaping caratteri di markup, commenti non all'interno del markup, ...

STRUTTURA LOGICA

Un documento XML ben formato è un **albero** di elementi in cui:

- la radice dell'albero è il primo elemento del documento
- se l'elemento x è contenuto nell'elemento y, allora x è figlio di y e y è il genitore di x
- le foglie sono quindi gli elementi che non contengono altri elementi

STRUTTURA LOGICA: ESEMPIO



STRUTTURA LOGICA: ESEMPIO (XML)

```
<address-book>
  <entry>
    <name>Jack Smith</name>
    <tel>513-555-3465</tel>
    <email address='jsmith@email.com'></email>
  </entry>
  <entry>
    <name>John Doe</name>
    <address>
      <street>34 Fountain Square Plaza</street>
      <region>Cincinnati, OH</region>
      <postal-code>45202</postal-code> <country>US</country>
    </address>
    <tel>513-555-8889</tel>
    <tel>513-555-7098</tel>
    <email address='jdoe@email.com'></email>
  </entry>
</address-book>
```

ANNIDAMENTO CORRETTO DI ELEMENTI

I figli devono essere **completamente contenuti** nei genitori, in altre parole:

- **non è possibile** che il tag di fine di un figlio appaia dopo il tag di fine di un genitore
  - **es., è errato:**

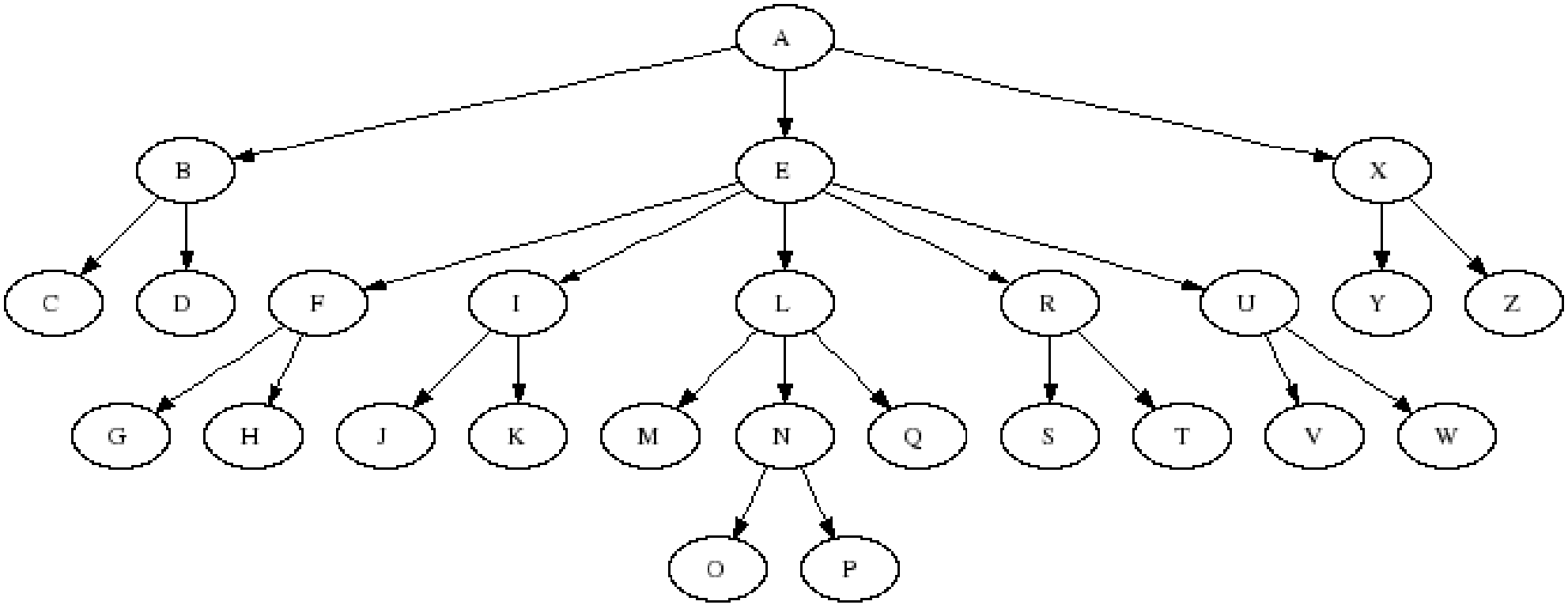
```
<name><fname>Jack</fname><lname>
Smith</name></lname>
```

UNICITÀ DELL'ELEMENTO RADICE

- Solo **un** elemento del documento XML può essere la radice dell'albero, e tutti gli altri elementi devono essere suoi discendenti
  - es., è errato:

```
<?xml version=”1.0”?>
<entry><name>John Doe</name></entry>
<entry><name>Jack Smith</name></entry>
```

ESERCIZIO: SCRIVERE L'XML EQUIVALENTE



ESERCIZIO: SOLUZIONE

```
<A>
  <B>
      <C></C>
      <D></D>
  </B>
  <E>
      <F>
        <G></G>
          <H></H>
       </F>
       <I>
           <J></J>
           <K></K>
       </I>
       <L>
          <M></M>
            ...
                <N>
                   <O></O>
                   <P></P>
                </N>
                <Q></Q>
            </L>
           <R>
             <S></S>
             <T></T>
           </R>
           <U>
             <V></V>
             <W></W>
           </U>
      </E>
      <X>
          <Y></Y>
          <Z></Z>
      </X>
</A>
```

DOCUMENT TYPE DEFINITION

XML – Tecnologie Web
e Laboratorio

DOCUMENT TYPE DEFINITION (DTD)

- Il Document Type Definition (DTD) dichiara come un certo "tipo" di documento XML deve essere scritto
  - es. che elementi e attributi si possono usare, che contenuto possono avere i diversi elementi, …
- Se un documento XML soddisfa le regole di un certo DTD si dice valido rispetto a quel DTD

COMPOSIZIONE DI UN DTD

- Un DTD è un documento in formato testuale composto da una lista di dichiarazioni
  - ogni dichiarazione è relativa a un "oggetto" che può essere contenuto nel tipo di documento XML considerato
- Ogni "oggetto" (elementi, attributi, ...) usabile in quel tipo di documento va definito!
- Il DTD segue una sintassi formale ben definita

DICHIARAZIONE DI ELEMENTI - 1

<!ELEMENT *nome_el contenuto_el*>

- ***nome_el*** è il nome dell'elemento
- ***contenuto_el*** specifica il contenuto dell'elemento, es.
  - vuoto
  - testo
  - altri elementi

DICHIARAZIONE DI ELEMENTI - 2

<!ELEMENT *nome_el* EMPTY>

- Dichiara un elemento vuoto
- **Es.** `<!ELEMENT email EMPTY>`
  - **nel documento XML, è corretto:**

    `<email />` **o** `<email></email>`
  - **non è corretto:**

    `<email>john@email.com</email>` **oppure**
    `<email><home>john@email.com</home></email>`

DICHIARAZIONE DI ELEMENTI - 3

<!ELEMENT *nome_el* (#PCDATA)>

- Dichiara un elemento che può contenere solo testo (di solito si usa per elementi foglia)

- **Es.** `<!ELEMENT name (#PCDATA)>`
  - **nel documento XML, è corretto:**

```
<name>John Doe</name>
```

  - **non è corretto:**

```
<name><fname>John</fname></name>
```

DICHIARAZIONE DI ELEMENTI - 4

<!ELEMENT *nome_el* ANY>

- Dichiara un elemento che può contenere qualunque altro elemento dichiarato nel DTD (si usa raramente)

- **Es.** `<!ELEMENT entry ANY>`

  - nel documento XML, è corretto mettere nel contenuto di entry qualunque altro elemento dichiarato nel DTD

  - è scorretta qualsiasi altra cosa

DICHIARAZIONE DI ELEMENTI - 5

<!ELEMENT *nome_el* (*expr*)>

- Dichiara un elemento che deve contenere altri elementi, in un certo ordine e numero definiti dall'espressione regolare *expr*

- **Es.** `<!ELEMENT address-book (entry)>`
  - **nel documento XML è necessario inserire l' elemento entry dentro ogni elemento address-book, es.** `<address-book><entry>...</entry></address-book>`
  - **non corretto:** `<address-book><a></a></address-book>`

INDICATORI DI OCCORRENZA

- Come si definisce la numerosità degli elementi contenuti nell'elemento dichiarato?
- In *expr* si utilizzano i simboli **+**, ***** e **?**, i quali sono indicatori di occorrenza degli elementi
  - se il nome di un elemento e' senza alcun simbolo di occorrenza, deve apparire una e una sola volta

    es.: `<!ELEMENT address-book (entry)>`
  - un elemento seguito da **+** deve apparire una o più volte

    es.: `<!ELEMENT address-book (entry+)>`
  - un elemento seguito da ***** deve apparire zero o più volte

    es.: `<!ELEMENT address-book (entry*)>`
  - un elemento seguito da **?** deve apparire zero o una volta

    es.: `<!ELEMENT address-book (entry?)>`

INDICATORI DI OCCORRENZA - ESEMPI

```
<!ELEMENT address-book (entry?)>
```

- Ogni elemento address-book del documento deve contenere zero oppure un elemento entry

```
<!ELEMENT address-book (entry+)>
```

- Ogni elemento address-book del documento deve contenere uno o più elementi entry
  - es.:

```
<address-book>
   <entry>…</entry><entry>…</entry>…
</address-book>
```

INDICATORI DI ORDINE

- Come si definisce l'ordine degli elementi contenuti nell'elemento dichiarato?

- In *expr* si utilizzano i simboli , e |

  - , permette di definire una *sequenza* di elementi

    es.: `<!ELEMENT address (street, city, state)>`

    - **corretto:** `<address><street>…</street><city>…</city><state>…</state></address>`
    - **non corretto:** `<address><state>…</state><city>…</city><street>…</street></address>`
    - **non corretto:** `<address><street>…</street><city>…</city></address>`

INDICATORI DI ORDINE - 2

- Come si definisce l'ordine degli elementi contenuti nell'elemento dichiarato?

- In *expr* si utilizzano i simboli , e |

  - | permette di definire elementi da inserire *in alternativa*

    es.: `<!ELEMENT address (street | city | state)>`

    - **corretto:** `<address><street>…</street></address>`
    - **corretto:** `<address><city>…</city></address>`
    - **corretto:** `<address><state>…</state></address>`
    - **non corretto:** `<address><state>…</state><city>…</city><street>…</street></address>`

RAGGRUPPAMENTO DI ELEMENTI

- E' possibile usare le parentesi tonde per raggruppare elementi in *expr*

  Es.:

```
<!ELEMENT name (lname, (fname | title))>
```

- E' possibile dichiarare un contenuto misto

  Es.:

```
<!ELEMENT name (#PCDATA | (fname , lastname))*>
```

EVITARE AMBIGUITÀ

- La dichiarazione di *expr* deve essere **non ambigua**, cioè deve essere possibile decidere univocamente quale parte dell'espressione regolare si applica ad un certo elemento nel documento XML
  - ad es. è ambiguo

```
<!ELEMENT cover ((title, author) | (title, subtitle))>
```

  - non è però ambiguo l'equivalente

```
<!ELEMENT cover (title, (author | subtitle))>
```

ESEMPIO: DTD ADDRESS-BOOK

```
<!ELEMENT address-book (entry+)>
<!ELEMENT entry (name,address?,tel+,email*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (street,
          region,postal-code,locality?,country)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT email EMPTY>
<!ELEMENT street (#PCDATA)>
<!ELEMENT region (#PCDATA)>
<!ELEMENT postal-code (#PCDATA)>
<!ELEMENT locality (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ATTLIST email address CDATA #REQUIRED>
```

DICHIARAZIONE DI ATTRIBUTI

```
<!ATTLIST nome_el nome_attr tipo_attr uso_attr>
```

- ***nome_el:*** è il nome dell'elemento in cui l'attributo si può usare
- ***nome_attr:*** è il nome dell'attributo
- ***tipo_attr:*** fornisce informazioni sul valore dell'attributo
- ***uso_attr:*** fornisce informazioni sull'utilizzo dell'attributo

USO ATTRIBUTO

<!ATTLIST *nome_el nome_attr tipo_attr* ***uso_attr>***

Valori possibili per ***uso_attr***

- **#REQUIRED:** il valore dell'attributo deve essere inserito nel documento
- **#IMPLIED:** l'attributo è opzionale; non va fornito un valore di default
- **#FIXED 'val':** il valore dell'attributo è sempre **val**
- **'val':** l'attributo, se omesso, ha il valore di default **val**

TIPO ATTRIBUTO

<!ATTLIST *nome_el nome_attr* ***tipo_attr*** *uso_attr***>**

Valori possibili per ***tipo_attr***

- **CDATA:** il valore dell'attributo è una sequenza di caratteri arbitraria
- **ID:** un identificatore univoco per l'attributo; il valore dell'attributo è un nome (e deve seguire le regole viste sui nomi XML) che deve essere unico nel documento (lo stesso valore non può essere assegnato a più elementi)
- **IDREF:** contiene il riferimento all'attributo ID di un elemento nel documento
- **IDREFS:** contiene riferimenti a più attributi ID di elementi nel documento

TIPO ATTRIBUTO - 2

Valori possibili per ***tipo_attr*** (cont.)

- **NMTOKEN**: *name token*, il valore dell'attributo è una parola senza spazi che può contenere lettere, numeri e i simboli '**-**', '_', '**.**' e '**:**'
- **NMTOKENS**: il valore dell'attributo è un elenco di *nmtoken* separati da spazi
- **(n1 | n2 | ... | nk):** il valore dell'attributo deve essere uno fra i *nmtoken* n1, n2, …, nk
- **ENTITY, ENTITIES, NOTATION:** riferimenti a entità, elenco di riferimenti a entità separati da spazi e informazioni su formati interpretabili da applicazioni esterne (poco usati)
  - es.: <!NOTATION gif SYSTEM "image/gif">
    <!NOTATION tiff SYSTEM "image/tiff">
    <!NOTATION jpeg SYSTEM "image/jpeg">
    <!NOTATION png SYSTEM "image/png">
    <!ATTLIST image type NOTATION (gif | tiff | jpeg | png) #REQUIRED>

ESEMPI: DICHIARAZIONE DI ATTRIBUTI

```
<!ATTLIST tel preferred (true | false) "false">
```

Il valore di preferred (attributo dell'elemento tel) può essere solo true oppure false, e se omesso vale false

```
<!ATTLIST email href CDATA #REQUIRED>
```

Il valore di href (attributo dell'elemento email) può essere qualsiasi sequenza di caratteri e deve essere specificato in ogni elemento email

PIÙ ATTRIBUTI PER LO STESSO ELEMENTO

È possibile definire più attributi per lo stesso elemento usando un'unica dichiarazione di tipo ATTLIST

<!ATTLIST nome_el nome_attr1 tipo_attr1 uso_attr1
nome_attr2 tipo_attr2 uso_attr1 … >

es.

```
<!ATTLIST tel prefix CDATA #REQUIRED
              preferred (true | false) "false">
```