



Introduzione a XML

1

Che cos'è XML?

- XML: **Extensible Markup Language**:
 - è un **linguaggio** che consente la rappresentazione di documenti e dati strutturati su supporto digitale
 - è uno strumento potente e versatile per la creazione, memorizzazione e distribuzione di documenti digitali
 - la sua sintassi rigorosa e al contempo flessibile consente di utilizzarlo nella rappresentazione di dati strutturati anche molto complessi

2

Le origini

- XML è stato sviluppato dal World Wide Web Consortium
- Nel 1996 è stato formato un gruppo di lavoro con l'incarico di definire un linguaggio a markup estensibile di uso generale
- Le specifiche sono state rilasciate come W3C Recommendation nel 1998 e aggiornate nel 2004
- XML deriva da SGML, un linguaggio di mark-up dichiarativo sviluppato dalla International Standardization Organization (ISO), e pubblicato ufficialmente nel 1986 con la sigla ISO 8879
- XML nasce come un sottoinsieme semplificato di SGML orientato all' utilizzo su World Wide Web
- Ha assunto ormai un ruolo autonomo e una diffusione ben maggiore del suo progenitore

3

SGML

- **SGML** (Standard Generalized Markup Language) è il padre sia di HTML che di XML.
- **Pregi**
 - Potente e flessibile (standard ISO, espandibile, fortemente strutturato, non proprietario, indipendente dalla piattaforma)
- **Difetti**
 - Struttura pesante
 - Sono obbligatori un DTD e uno StyleSheet
 - Obbligatoria la validazione del documento
 - Le istanze SGML sono troppo pesanti e non robuste per applicazioni WEB

4

XML come linguaggio di markup

- Come SGML XML è un linguaggio a marcatori (markup)
- Un **linguaggio di markup** è composto di istruzioni, definite **tag** o **marcatori**, che descrivono la struttura e la forma di un documento
- Ogni marcatore (o coppia di marcatori) identifica un **elemento** o componente del documento
- I marcatori vengono inseriti all'interno del documento
- Sia il testo, sia i marcatori sono memorizzati in formato ASCII
- XML usa la codifica dei caratteri **UNICODE**
- Un documento XML è “leggibile” da un utente umano senza la mediazione di software specifico

5

Esempio

- Un documento XML è leggibile chiaro, intuibile:

`<documento>`

Marcatore di
inizio

`<corpo>`

`Testo del mio primo documento`

`</corpo >`

`</documento>`

Marcatore di fine

- **Attenzione:** XML è case sensitive, nei nomi dei tag distingue fra maiuscole e minuscole

6

Altro esempio...

```
<prenotazione>
  <idVolo>PA321</idVolo>
  <idCliente>PP2305</idCliente>
  <data>22-10-2001</data>
  <prezzo valuta="Euro">245</prezzo>
</prenotazione>
```

7

XML: caratteristiche

- XML è indipendente dal tipo di piattaforma hardware e software su cui viene utilizzato
- Permette la rappresentazione di qualsiasi tipo di documento (e di struttura) indipendentemente dalle finalità applicative
- E' indipendente dai dispositivi di archiviazione e visualizzazione
- Un documento XML:
 - può essere archiviato su qualsiasi tipo di supporto digitale
 - può essere visualizzato su qualsiasi dispositivo di output
 - può essere facilmente trasmesso via Internet tramite i protocolli HTTP, SMTP, FTP

8

XML: caratteristiche

- XML è uno standard di pubblico dominio
- ogni software “conforme XML” è in grado di gestire dati in formato XML
- sono disponibili numerose applicazioni e librerie open source per la manipolazione di dati in formato XML basate su diversi linguaggi di programmazione (Java, C, C#, Python, Perl, PHP...)
- una applicazione in grado di elaborare dati in formato XML viene definita **elaboratore XML**

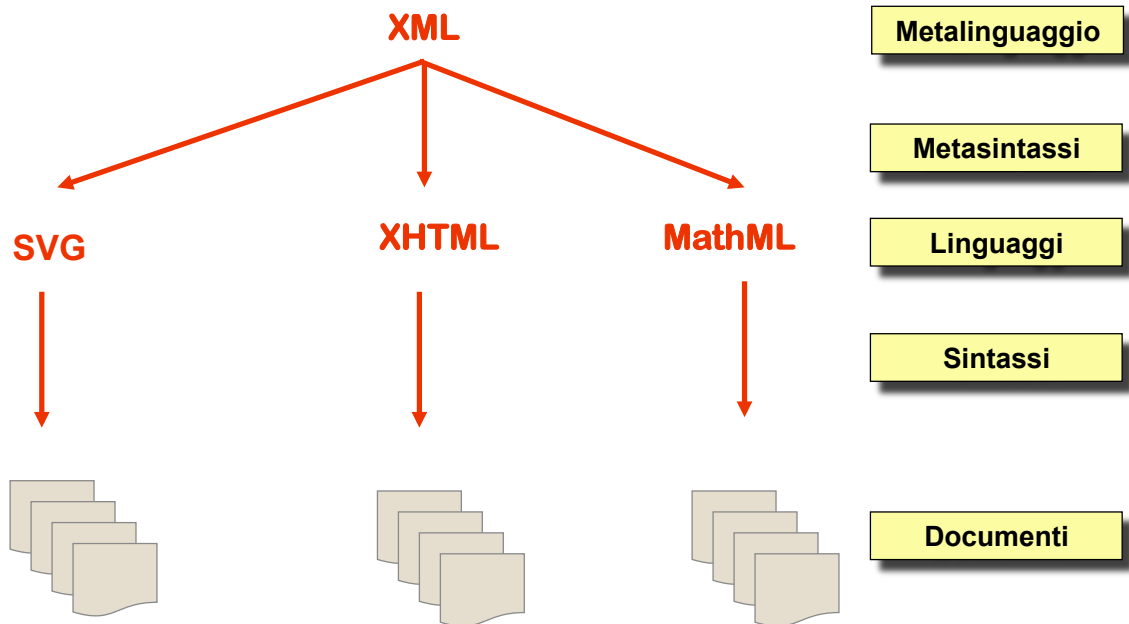
9

XML come metalinguaggio

- XML è un **metalinguaggio**
- Definisce un insieme regole (meta)sintattiche, attraverso le quali è possibile descrivere formalmente un linguaggio di markup, detto **applicazione XML**
- Ogni applicazione XML:
 - eredita un insieme di caratteristiche sintattiche comuni
 - definisce una sua sintassi formale particolare
 - è dotata di una semantica

10

Metalinguaggio e linguaggi



11

Linguaggi e grammatiche

- Per definire un linguaggio è necessario un meccanismo che vincoli l'utilizzo dei tag all'interno dei documenti.
- Si deve poter stabilire quali tag possono essere utilizzati e come, secondo una precisa struttura logica.
- Abbiamo cioè bisogno di definire una **grammatica**
- Una grammatica è un insieme di regole che indica quali vocaboli (elementi) possono essere utilizzati e con che struttura è possibile comporre frasi (documenti).
- Se un documento XML rispetta le regole definite da una grammatica è detto **valido** per un particolare linguaggio di markup

12

Documenti ben formati e documenti validi

- In XML ci sono **regole sintattiche** (o meglio metasintattiche)
 - **Come** dobbiamo scrivere le informazioni all'interno dei documenti
- Ci possono essere (ma non è obbligatorio) **regole semantiche**
 - **Cosa** possiamo scrivere in un documento XML.
- Un documento XML che rispetta le **regole sintattiche** si dice **ben formato (well-formed)**
- Un documento XML che rispetta le **regole sintattiche** e le **regole semantiche** si dice **valido**
- Un documento ben formato può non essere valido rispetto ad una grammatica, mentre un documento valido è necessariamente ben formato.

13

Struttura logica di un documento XML

- Un **documento XML**
 - E' strutturato i modo gerarchico
 - E' composto da **elementi**.
- Un **elemento**
 - rappresenta un componente logico del documento
 - può contenere un frammento di testo oppure altri elementi (**sottoelementi**).
- Ad un elemento possono essere associate informazioni descrittive chiamate **attributi**.
- Gli elementi sono organizzati in un albero con una radice singola (**root**)
- Ogni documento XML può essere rappresentato come un albero che prende il nome di **document tree**

14

Struttura fisica di un documento XML

- Un documento XML è un semplice file di testo
 - La **struttura del documento** viene rappresentata mediante marcatori (markup)
 - Gli **elementi** sono rappresentati mediante tag: coppie di marcatori che racchiudono il contenuto dell'elemento
 - I **sottoelementi** sono tag contenuti all'interno di un altro tag
 - Gli **attributi** vengono rappresentati sotto forma di coppie nome-valore all'interno dei tag
 - La **radice** è un tag che racchiude tutto il resto del documento (e quindi tutti gli altri tag)
 - Un documento può inoltre contenere spazi bianchi, a capo e **commenti**
-

15

Aspetti di sintassi generale

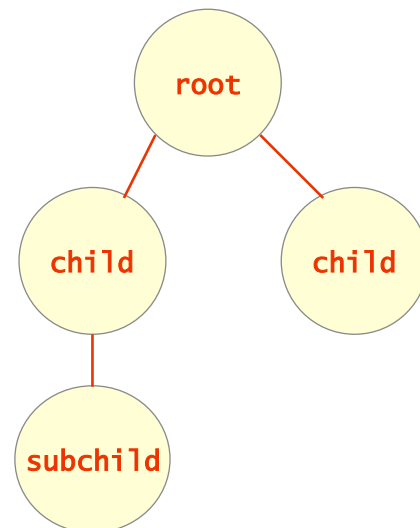
- Un documento XML è una stringa di caratteri ASCII o UNICODE
 - I nomi di elementi, attributi e entità sono sensibili alla differenza tra maiuscolo e minuscolo
 - Il mark-up è separato dal contenuto testuale mediante caratteri speciali:
 - **< > &**
 - I caratteri speciali non possono comparire come contenuto testuale e devono essere eventualmente sostituiti mediante i riferimenti a entità
 - **< > &**
-

16

Struttura logica e fisica

- Esiste una corrispondenza diretta fra struttura fisica e struttura logica (tree)

```
<root>
  <child>
    <subchild>
      ...
    </subchild>
  </child>
  <child>
    ...
  </child>
</root>
```



17

Struttura formale di un documento XML

- Un documento è costituito da due parti
 - Prologo**: contiene una dichiarazione XML ed il riferimento (opzionale) ad altri documenti che ne definiscono la struttura o direttive di elaborazione
 - Corpo**: è il documento XML vero e proprio

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="gree.css"?>

<root>
  <!-- Questo è un commento -->
  <child>
    ...
  </child>
  <child>
    ...
  </child>
</root>
```

Prologo

Corpo

18

Prologo: XML Declaration

- Ogni documento XML inizia con un prologo che contiene una **XML declaration**
- Forme di XML declaration:
 - `<?xml version="1.0"?>`
 - `<?xml version="1.0" encoding="UTF-8"?>`
- Contiene informazioni su:
 - Versione: per ora solo 1.0
 - Set di caratteri (opzionale):
- Esempi di set di caratteri:
 - ASCII: set di caratteri a 7 bit
 - UTF-8: unicode a 8 bit
 - UTF-16: unicode a 16 bit
 - ISO-8859-1: set Latin 1 con lettere accentate

19

Prologo: riferimenti a documenti esterni

- Il prologo può contenere riferimenti a documenti esterni utili per il trattamento del documento
- **Processing instructions:** istruzioni di elaborazione
 - Esempio 1: rappresentazione mediante CSS:
`<?xml-stylesheet type="text/css" href="gree.css"?>`
 - Esempio 2: trasformazione XSLT
`<?xml-stylesheet type="text/xsl" href="go.xslt"?>`
- **Doctype declaration:** grammatica da utilizzare per la validazione del documento
 - Grammatica contenuta in un file locale
`<!DOCTYPE book SYSTEM "book.dtd">`
 - Grammatica accessibile ad un URL pubblico:
`<!DOCTYPE book PUBLIC "http://www.books.org/book.dtd">`

20

Commenti

- I commenti possono apparire ovunque in un documento XML (sia nel prologo che nel corpo)
- I commenti sono utili per
 - Spiegare la struttura del documento XML
 - Commentare parti del documento durante le fasi di sviluppo e di test del nostro software
- I commenti non vengono mostrati dai browser ma sono visibili da parte di chi guarda il codice sorgente del documento XML

```
<!-- Questo è un commento -->
```

21

Elementi e Tag

- Un **elemento** è un frammento di testo racchiuso fra uno start tag e un end tag
- Uno **start tag** è costituito da un nome più eventuali attributi (v. slide succ.) racchiusi dai simboli '<','>'

```
<TagName attribute-list>
```

- Un **end tag** è costituito da un nome (lo stesso dello start tag) racchiuso da '</','>':

```
</TagName>
```

- Un tag vuoto è rappresentabile come:

```
<TagName attribute-list />
```

- Equivale a

```
<TagName attribute-list></TagName>
```

💣 **Attenzione:** I tag non possono avere nomi che iniziano per XML, XMI, Xml, xml...

22

Attributi

- A ogni elemento possono essere associati uno o più **attributi** che ne specificano ulteriori caratteristiche o proprietà non strutturali.
- Ad esempio:
 - la lingua del suo contenuto testuale
 - un identificatore univoco
 - un numero di ordine
 - etc.
- Gli attributi XML sono caratterizzati da
 - un nome che li identifica
 - un valore

23

Esempio di documento con attributi

```
<?xml version="1.0" ?>
<articolo titolo="Titolo dell'articolo">
  <paragrafo titolo="Titolo del primo paragrafo">
    <testo>Blocco di testo del primo paragrafo</
testo>
    <immagine file="immagine1.jpg"></immagine>
  </paragrafo>
  <paragrafo titolo="Titolo del secondo paragrafo">
    <testo>Blocco di testo del secondo paragrafo</
testo>
    <codice>Esempio di codice</codice>
    <testo>Altro blocco di testo</testo>
  </paragrafo>
  <paragrafo tipo="bibliografia">
    <testo>Riferimento ad un articolo</testo>
  </paragrafo>
</articolo>
```

24

Elementi o attributi?

- Qualche regola per decidere:
 - Un elemento è estendibile in termini di contenuto (con elementi figli) e di attributi
 - Un attributo non è estendibile: può solo modellare una proprietà di un elemento in termini di valore
 - Un elemento è un'entità a se stante (un oggetto?)
 - Un attributo è strettamente legato ad un elemento
 - Un attributo può solamente contenere un valore "atomico"
- In pratica non c'è una regola valida in assoluto
- La scelta dipende da diversi fattori: leggibilità, semantica, tipo di applicazione, efficienza...

25

Elementi o attributi: esempio

- Vediamo tre varianti dello stesso pezzo di documento che usano in modo diverso elementi e attributi

```
<libro isbn="1324AX" titolo="On the road" />
```

```
<libro isbn="1324AX">  
  <titolo>On the road</titolo>  
</libro>
```

```
<libro>  
  <isbn>1324AX</isbn>  
  <titolo>On the road</titolo>  
</libro>
```

26

Riferimenti ad entità

- I riferimenti ad entità servono per rappresentare caratteri riservati (per esempio < > o &)

Nome entità	Riferimento	Carattere
lt	<	<
gt	>	>
amp	&	&
apos	'	'
quot	"	"

- Oppure per rappresentare caratteri UNICODE mediante la notazione &#XXXX:
 - ½ → ½
 - è → è

27

Sezione CDATA

- Per poter inserire brani di testo senza preoccuparsi di sostituire i caratteri speciali si possono utilizzare le sezioni **CDATA** (Character Data)
- Il testo contenuto in una sezione CDATA NON viene analizzato dal parser
- Una sezione CDATA può contenere caratteri “normalmente” proibiti
- Si utilizza la seguente sintassi:

<![CDATA[*Contenuto della sezione*]]>

- L'unica sequenza non ammessa è]] (chiusura)
- Esempi:

```
<E1> <![CDATA[ <<"'! ]& ]]> </E1>  
<E> <![CDATA[<Elemento/><A>Ciao</A>]]> </E>
```

28

Conflitti sui nomi

- Capita abbastanza comunemente, soprattutto in documenti complessi, la necessità di dare nomi uguali ed elementi (o attributi) con significati diversi
- Ad esempio:

```
<libro>
  <autore>
    <titolo>Sir</titolo>
    <nome>William Shakespeare</nome>
  </autore>
  <titolo>Romeo and Juliet</titolo>
</libro>
```

29

Namespace

- Per risolvere il problema si ricorre al concetto di “spazio dei nomi” (**namespace**)
- Si usano prefissi che identificano il vocabolario di appartenenza di elementi ed attributi
- Ogni prefisso è associato ad un URI (Uniform Resource Identifier) ed è un alias per l’URI stesso
- L’URI in questione è normalmente un URL: si ha quindi la certezza di univocità
- E’ un meccanismo simile ai nomi lunghi delle classi in Java (i package definiscono un sistema di namespace):
 - Nome breve: **JButton**
 - Nome lungo: **javax.swing.JButton**

30

Esempio di uso di namespace

- **Riprendiamo l'esempio del libro e usiamo i namespace:**

The diagram illustrates the structure of an XML document with three annotations pointing to specific parts of the code:

- Prefisso** (Prefix) points to `lb:` in `<lb:libro>`.
- Dichiarazione del prefisso e associazione all'URI** (Declaration of the prefix and association to the URI) points to `xmlns:lb="mysite.com/libri"`.
- URI** (URI) points to `mysite.com/libri` in the namespace declaration.

```
<lb:libro xmlns:lb="mysite.com/libri">
  <au:autore xmlns:au="mysite.com/autori">
    <au:titolo>Sir</au:titolo>
    <au:nome>William Shakespeare</au:nome>
  </au:autore>
  <lb:titolo>Romeo and Juliet</lb:titolo>
</lb:libro>
```

31

Definizione di namespace

- Per definire un namespace si usa la seguente sintassi:
xmlns:NamespacePrefix="NamespaceURI"
- La definizione è un attributo di un elemento e può essere messa ovunque all'interno del documento
- Lo scope del namespace è l'elemento all'interno del quale è stato dichiarato
 - Si estende a tutti i sottoelementi
 - Se si dichiara un namespace nell'elemento radice, il suo scope è l'intero documento
- L'URI può essere qualsiasi (il parser non ne controlla l'univocità) ma dovrebbe essere scelto in modo da essere effettivamente univoco

32

Esempio

```
<DC:Docenti xmlns:DC="www.unibo.it/docenti">
  <DC:Docente DC:codAteneo="112233">
    <DC:Nome>Enrico</DC:Nome>
    <DC:Cognome>Lodolo</DC:Cognome>
    <CR:Corso id="123" xmlns:CR="www.unibo.it/corsi">
      <CR:Nome>Tecnologie Web L-A</CR:Nome>
    </CR:Corso>
    <CO:Corso id="124" xmlns:CO="www.unibo.it/corsi">
      <CO:Nome>Fondamenti di informatica L-B</CO:Nome>
    </CO:Corso>
  </DC:Docente>
</DC:Docenti>
```

1. CR e CO sono prefissi “collegati” allo stesso namespace
2. Nel secondo elemento Corso è necessario ripetere la dichiarazione di namespace poiché ricade fuori dallo scope della prima dichiarazione
 - Per evitare la seconda dichiarazione basta dichiarare il namespace in un elemento più in alto nella gerarchia

33

Namespace di default

- È possibile definire un namespace di default associato al prefisso nullo.
- Tutti gli elementi non qualificati da prefisso appartengono al namespace di default
- Attenzione: riduce la leggibilità di un documento

```
<Docenti xmlns="www.unibo.it/docenti">
  <Docente codAteneo="112233">
    <Nome>Enrico</Nome>
    <Cognome>Lodolo</Cognome>
    <CR:Corso id="123" xmlns:CR="www.unibo.it/corsi">
      <CR:Nome>Tecnologie Web L-A</CR:Nome>
    </CR:Corso>
  </Docente>
</Docenti>
```

34

Rdefinizione di prefissi

- Un prefisso di namespace (anche quello vuoto di default) può essere associato a diversi namespace all'interno di uno stesso documento.
- E' però preferibile evitare le ridefinizioni: riducono la leggibilità del documento.

```
<PR:Docenti xmlns:PR="www.unibo.it/docenti">
  <PR:Docente codAteneo="112233">
    <PR:Nome>Enrico</PR:Nome>
    <PR:Cognome>Lodolo</PR:Cognome>
    <PR:Corso id="123" xmlns:PR="www.unibo.it/corsi">
      <PR:Nome>Tecnologie Web L-A</PR:Nome>
    </PR:Corso>
  </PR:Docente>
</PR:Docenti>
```

35

Vincoli di buona formazione

- Per avere un documento XML **ben formato**:
 - Deve contenere una dichiarazione corretta
 - Il corpo deve avere un unico elemento radice (esclusi eventuali commenti).
 - Ogni elemento deve avere un tag di apertura e uno di chiusura. Se l'elemento è vuoto si può utilizzare la forma abbreviata (<nometag/>)
 - Gli elementi devono essere opportunamente nidificati, cioè i tag di chiusura devono seguire l'ordine inverso dei rispettivi tag di apertura
 - I nomi dei tag di apertura e chiusura devono coincidere, anche in termini di maiuscole e minuscole
 - I valori degli attributi devono sempre essere racchiusi tra singoli o doppi apici

36

Documenti ben formati e documenti validi

- In XML ci sono **regole sintattiche**
 - **Come** dobbiamo scrivere le informazioni all'interno dei documenti
- Ci possono essere (ma non è obbligatorio) **regole semantiche** (definite da grammatiche)
 - **Cosa** possiamo scrivere in un documento XML.
- Un documento XML che rispetta le **regole sintattiche** si dice **ben formato (well-formed)**
- Un documento XML che rispetta le **regole sintattiche** e le **regole semantiche** si dice **valido**
- Un documento ben formato può non essere valido rispetto ad una grammatica, mentre un documento valido è necessariamente ben formato.

37

Validazione: Document Type Definition

- Un primo strumento per definire grammatiche è costituito dalla **Document Type Definition**
- Una DTD è costituita da un elenco di dichiarazioni (**markup declaration**) che descrivono la struttura del documento
- Le dichiarazioni di una DTD definiscono:
 - gli elementi strutturali (**element**) di un documento mediante un identificatore generico
 - il modello di contenuto di ogni elemento (**content model**) ovvero gli elementi che contiene ed i loro rapporti (un elemento può essere vuoto)
 - la lista degli **attributi** associati a ciascun elemento e il loro tipo

38

Limiti e problemi delle DTD

- Le DTD sono difficili da comprendere
- Sono in un linguaggio diverso da quello usato per descrivere le informazioni
- Soffrono di alcune limitazioni:
 - Non permettono di definire il tipo dei dati
 - Non consentono di specificare il numero minimo o massimo di occorrenze di un tag in un documento

39

Validazione: XML Schema (XSD)

- Dato che XML può descrivere tutto perchè non usarlo per descrivere anche lo schema di un documento?
- E' stato quindi definito lo standard XSD (**XML Schema Definition**)
- XSD nasce dall'idea di utilizzare XML per descrivere la struttura di XML:
 - Descrive le regole di validazione di un documento
 - Permette di tipizzare i dati (intero, stringa, ora, data ecc.)
 - E' estensibile ed aperto alla possibilità di supportare modifiche

40

Elementi di XSD

- Un documento XML Schema (XSD) comprende:
- **Namespace** di riferimento:
<http://www.w3.org/2001/XMLSchema>
- **Dichiarazione di:**
 - Elementi
 - Attributi
- **Definizione di tipi**
 - Semplici
 - Complessi
 - Estesi

41

HTML e XML: XHTML

- HTML è una grammatica XML?
- Quasi, però sono ammessi “pasticci” che XML non prevede:
 - Tag non chiusi: `<br>` (in XML `<br></br>` o `<br/>`)
 - Tag “incrociati” `<b><u>Ciao</b><u>`
- HTML è una grammatica SGML ma non XML
- E' stata definita una versione di HTML “corretto” in modo da rispettare la sintassi XML: **XHTML**
- Un documento XHTML è un documento XML ben formato che può essere validato su uno schema definito dal W3C
- Quindi **XHTML è una grammatica XML**

42

DOM

- Il DOM (Document Object Model) è un modello ad oggetti definito dal W3C per navigare e creare contenuti XML
- Rappresenta il contenuto di un documento XML tramite un albero in memoria
- Permette di navigare l'albero ragionando per gradi di parentela (nodi figli, nodo padre, ecc.)
- Esistono 3 interfacce base:
 - **Node** (è praticamente la base di tutto)
 - **NodeList** (collezione di nodi)
 - **NamedNodeMap** (collezione di attributi)
- Un parser DOM è un'applicazione in grado di leggere un file XML e creare un DOM e viceversa

43

Presentazione di documenti XML

- Un documento XML definisce il contenuto informativo e non come deve essere rappresentato tale contenuto
- La presentazione di un documento XML viene controllata da uno o più fogli di stile
- I linguaggi di stile utilizzabili con XML sono
 - Extensible Stylesheet Language (**XSL**)
 - Cascading Style Sheet (**CSS**) (usati con XHTML)

44

XSL

- **XSL** = e**X**tensible **S**tylesheet **L**anguage
- Si occupa della trasformazione e della impaginazione di contenuti XML.
- Si basa principalmente su:
 - **XSLT** (**XSL** for **T**ransformation): gestisce la trasformazioni e non la impaginazione dei contenuti
 - **XSL-FO** (**XSL** **F**ormatting **O**bjects): orientato alla visualizzazione ed impaginazione dei contenuti (per esempio in PDF)
 - **XPath** (**X**ML **P**ath Language): serve per costruire percorsi di ricerca di informazioni all'interno di documenti XML

45

XSLT

- XSLT è un linguaggio di programmazione a tutti gli effetti
- Permette di gestire variabili, parametri, cicli, condizioni, funzioni
- **È una grammatica XML**
- Lavora sulla **struttura** del documento
 - Costruisce l'albero del documento
 - Lo attraversa cercando le informazioni indicate
 - Produce un nuovo documento - di solito **XML** - applicando le regole definite

46

Un esempio tipico

