

Codifica di Testi - Introduzione XML Markup a.a. 2018-2019

Angelo Mario Del Grosso

`angelo.delgrosso@ilc.cnr.it`

CNR-ILC-LicoLab

Istituto di Linguistica Computazionale “A. Zampolli”,
5th October 2018

Contenuto della lezione

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

1 I linguaggi di codifica

2 Fondamenti del linguaggio XML

3 Validare XML

- DTD

- XSD

- RELAXGN

4 Conclusioni

Progress status

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

1 I linguaggi di codifica

2 Fondamenti del linguaggio XML

3 Validare XML

4 Conclusioni

I linguaggi di codifica

introduzione

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML
DTD
XSD
RELAXGN

Conclusioni

Definizione di codifica digitale del testo

Per **codifica** digitale dei testi intendiamo la *rappresentazione formale* di un **testo** ad un qualche livello descrittivo, su di un supporto digitale, in un formato utilizzabile da un elaboratore (*Machine Readable Form*) mediante un opportuno **linguaggio informatico** (F. Ciotti).

Impostazione teorico-pratica

- un testo è molto di più della sequenza di caratteri che lo compongono
- per mezzo della codifica vogliamo rendere esplicite le caratteristiche che vogliamo analizzare
- solo quello che è esplicito può essere interpretato ed elaborato dal computer
- vogliamo codificare il testo per quello che è, non per quello che sembra
- codifica da effettuare mediante linguaggio di markup

I linguaggi di codifica

Linguaggi di marcatura

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

Il markup

Il termine markup è stato utilizzato in passato per denotare i segni grafici che accompagnavano un testo apposti sul documento per indicare correzioni o modalità grafiche di stampa.

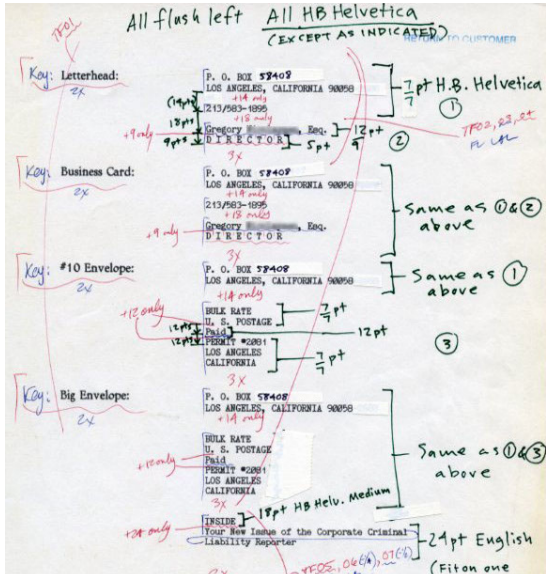
Linguaggi di marcatura

a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di codifica

Validare XML



I linguaggi di codifica

Linguaggi di marcatura

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML




















































Validare XML

DTD

XSD

RELAXGN

Conclusioni

		Delete		Flush Left		Set in Bold Face Italic
		Insert		Flush Right		Set in Light Face
		Join		Center Horizontally		Wrong Font
		Move closer		Center Vertically		Hyphen
		Space		Move to the next line		En Dash
		Add Space		Move to the preceding line		Em Dash
		Delete Space		Indent 1 em		Superscript
		Transpose Word		Indent 2 ems		Subscript
		Transpose Letters		Paragraph		Comma
		To separate two or more marks		All Caps		Apostrophe
		Let it Stand (ignore correction)		Small Caps		Period
		Move Left		Caps & Small Caps		Semicolon
		Move Right		Capital Letter		Colon
		Move Up		Lower Case		Quotation Marks
		Move Down		Set in Roman		Parentheses
		Align Vertically		Set in Italic		Brackets
		Align Horizontally		Set in Bold Face		

I linguaggi di codifica

Linguaggi di marcatura

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML
DTD
XSD
RELAXGN

Conclusioni

Il markup

La codifica con linguaggi di marcatura (markup) è in definitiva un insieme di convenzioni, rese attraverso specifiche sequenze di caratteri, etichette, codici, (detti tags) intercalati nel testo per permettere agli elaboratori elettronici di distinguere le varie parti di un documento.

Il markup formale

Un linguaggio di markup è un sistema formale per scambiare e pubblicare informazioni in formato testo in modo strutturato.

I linguaggi di codifica

Linguaggi di marcatura

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

Il markup formale

Markup formale: costituito da un sistema ben preciso di istruzioni, ognuna delle quali è dotata di una specifica semantica e sintassi.

I linguaggi di codifica

Linguaggi di marcatura

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

Diversi tipi di markup

Esistono diversi linguaggi di markup, per rappresentare diversi tipi di documenti.

- Linguaggi procedurali (specific markup languages)
- Linguaggi dichiarativi (generic markup languages)

I linguaggi di codifica

Linguaggi di marcatura procedurale

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML
DTD
XSD
RELAXGN

Conclusioni

Linguaggi procedurali

- Orientati al documento, indicano come deve essere elaborato e disposto il testo
- Istruzioni da inserire nel testo per specificarne specifiche caratteristiche
- Font, dimensione, spaziatura del carattere, posizionamento nella pagina, colore, etc.

Esempi: TeX e LaTeX, RTF

I linguaggi di codifica

Linguaggi di marcatura procedurale

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

Esempio RTF

```
{\rtf1\ansi\deff0\adeflang1025
{\fonttbl{\f0\froman\fprq2\fcharset0 Times New Roman;}
{\f1\froman\fprq2\fcharset0 Times New Roman;}
{\f2\fnil\fprq2\fcharset0 Lucida Sans Unicode;}
{\colortbl;\red0\green0\blue0;\red128\green128\blue128;}
{\stylesheet{\s1\cf0{\*\hyphen2\hyphlead2\hyphtrail2\hyphmax0}
\rtlch\af5\afs24\lang255\ltrch\dbch\af2\afs24\langfe255
\loch\f0\fs24\lang1040\snext1 Standard;}}
```

I linguaggi di codifica

Linguaggi di marcatura procedurale

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

Esempio LaTeX

```
documentclass a4paper , 10pt article
```

I linguaggi di codifica

Linguaggi di marcatura

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

Il markup procedurale

L'unico utilizzo di un testo codificato tramite un linguaggio procedurale è la creazione di un output orientato alla visualizzazione.

I linguaggi di codifica

Linguaggi di marcatura procedurale

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

Scrivere la tesi di laurea con $\text{\LaTeX} 2_{\epsilon}$

Dipartimento di Ingegneria Meccanica,
Nucleare e della Produzione
Università di Pisa
56126 Pisa PI

Sommario

Lo scopo del presente articolo è fornire gli strumenti per scrivere una tesi di laurea utilizzando $\text{\LaTeX} 2_{\epsilon}$. Tale obiettivo è conseguito analizzando i problemi tipici incontrati durante la stesura della tesi e le possibili soluzioni; si pone particolare attenzione ai pacchetti da usare nelle varie circostanze. I singoli argomenti non vengono approfonditi nei dettagli ma si rimanda alla letteratura specifica o ad i manuali dei pacchetti suggeriti, ove necessario.

*Ringrazio in primo luogo Fabiano Busdraghi che ha collaborato alla scrittura delle sezioni riguardanti le figure e gli oggetti flottanti. Ringrazio inoltre tutti coloro che mi hanno consigliato durante la stesura e la revisione di questo documento ed in particolare Claudio Beccari, Gustavo Cevolani, Massimo Guiggiani, Maurizio Himmelmann, Lorenzo Pantieri e Emiliano Vavassori.

Linguaggi dichiarativi

Orientati al testo, annotano la struttura, la funzione ed il significato degli elementi costitutivi del testo, tralasciandone l'aspetto.

- La posizione che il brano in questione occupa all'interno del documento (markup strutturale)
- Peculiarità del testo stesso (markup semantico)
- I fogli di stile definiscono la formattazione dell'output
- Molteplici usi del medesimo testo

Esempio: famiglia SGML, XML

I linguaggi di codifica

Linguaggi di marcatura dichiarativi

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

Markup dichiarativi: contenuto e presentazione

La separazione tra contenuto e presentazione non solo è intenzionale, ma è la caratteristica principale di questi sistemi di marcatura: essa permette di concentrarsi sull'annotazione logica-semantica per funzioni di ricerca e di analisi, lasciando ad altro (ai fogli di stile) la resa grafica.

Unico testo più usi

In questo modo si ha inoltre la possibilità di utilizzare uno stesso testo codificato con finalità o formattazioni differenti, a seconda delle varie esigenze.

I linguaggi di codifica

Markup dichiarativi: esempio SGML

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

Standard Generalized Markup Language

```
1 <!DOCTYPE testo [  
2 <!ELEMENT testo (titolo?, paragrafo+)>  
3 <!ELEMENT titolo (#PCDATA)>  
4 <!ELEMENT paragrafo (#PCDATA)>  
5 ]>  
6 <testo>  
7   <titolo> Questo è il titolo del documento</titolo>  
8   <paragrafo> Questo è un paragrafo </paragrafo>  
9 </testo>  
10
```

I linguaggi di codifica

Markup dichiarativi vs Markup procedurali

resa a video della frase

Le *Guidelines for Electronic Text Encoding and Interchange* sono *molto* complete e descrivono uno standard di *markup* del testo basato su XML.

Le `\textit{Guidelines for Electronic Text Encoding and Interchange}` sono `\textit{molto}` complete e descrivono uno standard di `\textit{markup}` del testo basato su XML.

`<titolo>`Le Guidelines for Electronic Text Encoding and Interchange`</titolo>` sono `<enfasi>`molto`</enfasi>` complete e descrivono uno standard di `<linguastraniera>` markup`</linguastraniera>` del testo basato su XML.

LaTeX vs SGML

I linguaggi di codifica

Linguaggi di marcatura

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

linguaggi semi-dichiarativi e/o semi-procedurali

Esistono anche linguaggi che possono essere definiti semi-procedurali, o semi-dichiarativi, che come si intuisce utilizzano le istruzioni sia per una codifica di tipo procedurale, sia per una codifica di tipo descrittivo o dichiarativo.

HTML

HTML ha tra le sue etichette istruzioni di tipo procedurale per indicare come devono essere rese determinate porzioni di testo, e istruzioni di tipo dichiarativo che hanno una base semantica.

Progress status

Codifica di
Testi -
Introduzione
XML Markup

a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

1 I linguaggi di codifica

2 Fondamenti del linguaggio XML

3 Validare XML

4 Conclusioni

Fondamenti XML

eXtensible Markup Language

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

XXXXXXXXXXXXX

XML has its roots in Standard Generalized Markup Language (SGML), a language introduced in the 1980s that describes the structure and content of any machine- readable information.

XXXXXXXXXXXXXXXXXXx

XML can be thought of as a lightweight version of SGML. Like SGML, XML is a language used to create vocabularies for other markup languages

Fondamenti XML

eXtensible Markup Language

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML
DTD
XSD
RELAXGN

Conclusioni

XXXXXXXXXXXXXXXXXXXX

It is a set of rules for defining custom-built markup languages

XXXXXXXXXXXXXXXXXXXX

XML was originally created to structure, store, and transport information. Like SGML, XML can be used to create XML applications or vocabularies, which are markup languages tailored to contain specific pieces of information.

Fondamenti XML

eXtensible Markup Language

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

XXXXXXXXXXXXXXXXXX

XML is a markup language that is extensible, so it can be modified to match the needs of the document author and the data being recorded

XXXXXXXXXXXXXXXXXX

developed and maintained by the World Wide Web Consortium (W3C), an organization created in 1994 to develop common protocols and standards for sharing information on the World Wide Web

Fondamenti XML

eXtensible Markup Language

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML
DTD
XSD
RELAXGN

Conclusioni

XXXXXXXXXXXXXXXXXX

XML, or eXtensible Markup Language, is a specification for storing information. It is also a specification for describing the structure of that information. And while XML is a markup language (just like HTML), XML has no tags of its own

Fondamenti XML

eXtensible Markup Language

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

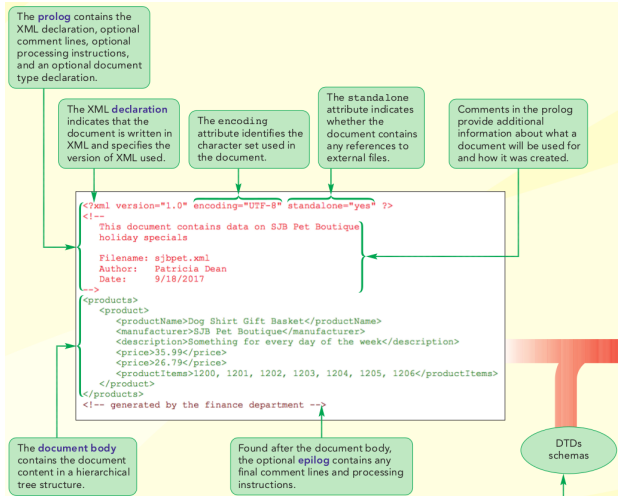


immagine dal libro *New Perspectives on XML, 3rd Edition*

Fondamenti XML

eXtensible Markup Language: Syntax Rules

- Every XML element must have a closing tag.
- XML tags are case sensitive.
- XML elements must be properly nested.
- Every XML document must have a root element.
- XML elements can have attributes in name-value pairs.
- Some characters have a special meaning in XML.
- Comments cannot occur prior to the XML Declaration.
Comments cannot be nested.

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

XML vista ad albero

XML is based on hierarchical trees in which order is significant. In XML, hierarchy and sequence are the main methods used to represent information.

Fondamenti XML

eXtensible Markup Language

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

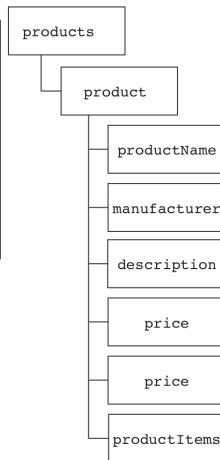
Conclusioni

XML document

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!--
  This document contains data on SJB Pet Boutique
  holiday specials

  Filename: sjbpet.xml
  Author:   Patricia Dean
  Date:    9/18/2017
-->
<products>
  <product>
    <productName>Dog Shirt Gift Basket</productName>
    <manufacturer>SJB Pet Boutique</manufacturer>
    <description>Something for every day of the week</description>
    <price>35.99</price>
    <price>26.79</price>
    <productItems>1200, 1201, 1202, 1203, 1204, 1205, 1206</productItems>
  </product>
</products>
<!-- generated by the finance department -->
```

Hierarchy tree structure



TEI-XML vocabulary

To meet the need of textual scholars, an XML vocabulary called Text Encoding Initiative (TEI) was developed, which codes text information.

XML Vocabularies

XML vocabularies: set of XML tags for a particular business function.

Fondamenti XML

eXtensible Markup Language: Esempio TEI

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

```
<div type="narrative" n="6">
  <head>Sixth Narrative</head>
  <head>contributed by Sergeant Cuff</head>
  <div type="fragment" n="6.1">
    <opener>
      <dateline>
        <name type="place">Dorking, Surrey,</name>
        <date>July 30th, 1849</date>
      </dateline>
      <salute>To <name>Franklin Blake, Esq.</name> Sir, -</salute>
    </opener>
    <p>I beg to apologize for the delay that has occurred in the
      production of the Report, with which I engaged to furnish you.
      I have waited to make it a complete Report ...</p>
    <closer>
      <salute>I have the honour to remain, dear sir, your
        obedient servant </salute>
      <signed>
        <name>RICHARD CUFF</name> (late sergeant in the
          Detective Force, Scotland Yard, London). </signed>
      </closer>
    </div>
  </div>
```

immagine dal sito TEI Guide Lines

Manutenibilità

Because XML focuses on communicating the data, the overall structure is simple and easy to design and maintain.

Fondamenti XML

eXtensible Markup Language

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML
DTD
XSD
RELAXGN

Conclusioni

Documento ben formato (well-formed)

A well-formed document contains no syntax errors and satisfies the general specifications for XML code as laid out by the W3C. At a minimum, an XML document must be well formed or it will not be readable by programs that process XML code.

Parti principali di un documento XML

An XML document consists of three parts—the prolog, the document body, and the epilog.

Fondamenti XML

eXtensible Markup Language: Esempio TEI

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/css" href="customStyle.css"?>
<!--The following document is made online by the Perseus Project -->
<!--Added the TEI-lite DTD and a processing instruction -->
<!DOCTYPE TEI.2 SYSTEM "teixbaby.dtd">

<TEI.2>
  <text lang="en">
    <body>
      <div1 type="book" n="1" org="uniform" sample="complete">
        <div2 type="section" n="327A" org="uniform" sample="complete">
          <p>
            327A - 328B Socrates describes how he visited the Piraeus in company with Glauco, and
            was induced by Polemarchus and others to defer his return to Athens.
          </p>
          <p>
            <lemma lang="greek" targOrder="U" from="ROOT" to="DITTO">κατέβην κτλ.</lemma>
            Dionys. Hal.
            <title lang="la">de comp. verb.</title>
            p. 208 (Reiske)
            <foreign lang="greek">
              ὁ δὲ Πλάτων, τοὺς
              ἑαυτοῦ διαλόγους κτενίζων καὶ βοστρυχίζων, καὶ πάντα τρόπον ἀναπλέκων, οὐ
              διέλιπεν ὀγδοήκοντα γεγονῶς ἔτη. πᾶσι γὰρ δὴ πού τοις φιλολόγοις γνῶριμα
              τὰ περὶ τῆς φιλοπονίας τάνδρὸς ἱστορούμενα, τὰ τ' ἄλλα, καὶ δὴ καὶ τὰ
              περὶ τὴν δέλτον ἦν τελευτήσαντος αὐτοῦ λέγουσιν εὐρεθῆναι ποικίλως
              μετακειμένην τὴν ἀρχὴν τῆς πολιτείας ἔχουσαν τήνδε "κατέβην χθές
              εἰς Πειραιᾶ μετὰ Γλαῦκωνος τοῦ Ἀριστῶνος
            </foreign>
            ."
          </p>
        </div2>
      </div1>
    </body>
  </text>
</TEI.2>

<!-- This document is not completed and was cut without a special meaning -->
```

Parti principali di un documento XML

The prolog includes the following parts:

- XML declaration: indicates that the document is written in the XML language
- Processing instructions (optional): provide additional instructions to be run by programs that read the XML document
- Comment lines (optional): provide additional information about the document contents
- Document type declaration (DTD) (optional): provides information about the rules used in the XML document's vocabulary

Parti principali di un documento XML

The document body, found immediately after the prolog, contains the document's content in a hierarchical tree structure. Following the document body is an optional epilog, which contains any final comments or processing instructions.

Fondamenti XML

eXtensible Markup Language: Prologo

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

XML declaration

```
<?xml version="version number" encoding="encoding  
type" standalone="yes|no" ?>
```

Creating an XML Declaration • To create an XML declaration, enter the code `<?xml ?>` in the first line of an XML document.

- To specify a version of XML to use, enter the code `version="version number"` after the opening `<?xml` tag, where version number is either 1.0 or 1.1.
- To specify a character encoding, enter the code `encoding="encoding type"` after the version attribute-value pair, where encoding type identifies the character set used in the document.
- To indicate whether the document is a standalone document, enter the code `standalone="yes—no"` after the encoding attribute-value pair, where the value yes or no indicates whether access to external files will be needed when processing the document.

ERRORI: `<?XML VERSION="1.0" ENCODING="ISO-8859-1" STANDALONE="YES" ?>` `<?xml version=1.0 encoding=ISO-8859-1 standalone=yes ?>` `<?xml version="1.0"`

standalone="yes" encoding="ISO-8859-1" ? ζ

Generally speaking, comments are ignored by programs reading the document and do not affect the document's contents or structure. To insert a comment in an XML document, enter ζ !–comment – ζ

If you have a comment that will occupy more than one line, you can continue the comment on as many lines as you need

ESERCIZIO

ζ !– This document contains data on SJB Pet Boutique holiday specials
Filename: project.xml Author: your name Date: today's date – ζ

A program that reads and interprets an XML document is

called an XML processor or XML parser, or simply a processor or parser.

First well-formed A second function of a parser is to interpret PCDATA in a document and resolve any character or entity references found within the document finally, an XML document might contain processing instructions that tell a parser exactly how the document should be read and interpreted

The current versions of all major web browsers include an XML parser of some type.

To test for well-formedness, you'll use an XML parser to compare the XML document against the rules established by the W3C. INTRODURRE XMILLINT

The document body in an XML document is made up of elements that contain data

to be stored in the document. Elements are the basic building blocks of XML files.

An element can have text content and child element content.

`<element>content</element>` The opening tag is `<element>` , and `</element>` is the closing tag.

- Gli elementi XML possono avere diversi tipi di contenuto: – contenuto strutturale: solo altri elementi, non testo – contenuto misto: testo e anche altri elementi – contenuto testuale: solo testo, non altri elementi

Element names might be established already if an author is

using a particular XML vocabulary, such as TEI-XML

There are a few important points to remember about XML elements:

- Element names are case sensitive, which means that, for example, itemnumber, itemNumber, and ItemNumber are unique elements.
- Element names must begin with a letter or the underscore character (_) and may not contain blank spaces. Thus, you cannot name an element Item Number , but you can name it Item_Number .
- Element names cannot begin with the string xml because that group of characters is reserved for special XML commands.
- The name in an element's closing tag must exactly match the name in the opening tag.
- Element names can be used more than once, so the element names can mean different things at different points in the hierarchy of an XML document.

Creating XML Elements XML 24 • To create an XML element, use the syntax `<element>content</element>` where `element` is the name given to the element, `content` represents the text content of the element, `<element>` is the opening tag, and `</element>` is the closing tag. • To create an empty XML element with a single tag, use the following syntax: `<element />` • To create an empty XML element with a pair of tags, use the syntax `<element></element>`

An open element or empty element is an element with no content.

`<element />` where `element` is the name of the empty element

`<element></element>` a two-sided tag with no content

Empty elements can also contain attributes that can be used to

store information

Nesting Elements In addition to text content, elements also can contain other elements. An element contained within another element is said to be a nested element.

ESEMPIO `<product>
 <productName>Dog Shirt Gift
 Basket</productName>
 <manufacturer>SJB Pet
 Boutique</manufacturer>
 <description>Something for every day
 of the week</description>
 <price>35.99</price>
 <price>26.79</price>
 <productItems>1200, 1201, 1202, 1203,
 1204, 1205, 1206 </productItems>
</product>`

hierarchical relationships between elements A nested element is a child element of the element in which it is nested— its parent element Elements that are side-by-side in a document's hierarchy are sibling elements

syntax error in creating an XML document is improperly nesting

XML does not allow the opening and closing tags of parent and child elements to overlap

Esercizio: scrivere e fare il check di un xml non opportunamente annidato

The Element Hierarchy All elements in the body are children of a single element called the root element or document element.

hierarchy represented in a tree diagram

Note that the XML declaration and comments are not included in the tree structure of the document body

An XML document must include a root element to be considered well formed

However, by indenting the code and placing siblings on their own lines, you can visually reveal the hierarchy relationships and add a dimension of visual communication to your code.

A quick way to view the overall structure of a document body is to chart the elements in a tree structure

It would be useful to have a general tree diagram that indicates whether a particular child element can occur zero times, once, or several times within a parent

The symbols `?`, `*`, and `+` are part of the code used in creating DTDs to validate XML documents.

MIXED CONTENT Mixed content elements are ideal for descriptive, text-based chunks of information. They are not very common in database-type applications.

In some cases, you may want an element to contain both content and child elements. This is referred to as mixed content

ESEMPIO `<wonder> Temple of Artemis at Ephesus
<city>Ephesus</city> <country>Turkey</country> </wonder>`

Esercizio: aprire il file XML non ben formato nella cartella xml
e: - correggerlo (mettendo come commenti la correzione fatta e
breve spiegatura) – nested, case sensitive - aggiungere un
figlio (child) all'elemento XYZ1 - aggiungere un fratello
(sibling) all'elemento XYZ2

Working with Attributes Every element in an XML document
can contain one or more attributes An attribute describes a
feature or characteristic of an element

`<element attribute="value"> ... </element><element`

attribute="value" /i

Attribute values are text strings. Therefore, an attribute value always must be enclosed within either single or double quotes.

Esempio: (preso dalla TEI)

name for an attribute as long as it meets the following rules: • An attribute name must begin with a letter or an underscore (_). • Spaces are not allowed in attribute names. • Like an element name, an attribute name should not begin with the text string xml.

An attribute name can appear only once within an element. Like all other aspects of XML, attribute names are case sensitive, and incorrect case is a common syntax error found in attributes. The order of attributes is not significant and there

is no way to control the order of attributes

Adding an Attribute to an Element • To add an attribute to an element, use the syntax `<element attribute="value"> ... </element>` where `element` is the name given to the element, `attribute` is the attribute's name, and `value` is the attribute's value. • To add an attribute to a single-sided tag, use the syntax `<element attribute="value" />` • To specify multiple attributes for a single element, use the syntax `<element attribute1="value1" attribute2="value2" ...> ... </element>` where `attribute1` is the first attribute's name, `value1` is the first attribute's value, `attribute2` is the second attribute's name, `value2` is the second attribute's value, and so on. Each attribute is separated by a space.

It's not always clear when to use an attribute value rather than

inserting a new element. A general rule of thumb is that if all of the XML tags and their attributes were removed from a document, the remaining text should comprise the document's content or information. Another rule of thumb is that attributes should be used to describe data, but should not contain data themselves. Different developers have different preferences, and there's no right answer.

Using Character and Entity References a numeric character reference, also known simply as a character reference. The syntax for a character reference is `&#nnn`; where `nnn` is a character number from the ISO/IEC character set

the character numbers for different symbols, some symbols also can be identified using a character entity reference—also known simply as an entity reference—using the syntax `&EntityName` Unicode

can be used in conjunction with entity references to display specific characters or symbols. &entity;

Immagine

Inserting Character and Entity References • To insert a character reference into an XML document, use &#nnn; where nnn is a character reference number from the ISO/IEC character set. • To insert an entity reference, use &entity; where entity is a recognized entity name.

text characters fall into three categories—parsed character data, character data, and white space.

Parsed Character Data Parsed character data (PCDATA) consists of all those characters that XML treats as parts of the code of an XML document. • the XML declaration • the

opening and closing tags of an element • empty element tags • character or entity references • comments

The presence of PCDATA can cause unexpected errors to occur within a document This means that symbols such as &, ¡, or ¿, which are all used in creating markup tags or entity references, are extracted and the appropriate content is used in your program.

Character Data Character data is not processed, but instead is treated as pure data content.

As an alternative to using character references, you can place text into a CDATA section. A CDATA section is a block of text that XML treats as character data only. The syntax for creating a CDATA section is ¡[CDATA [character data]]

In a CDATA section, these characters are interpreted by XML parsers or XML editors as text rather than as markup commands. A CDATA section • may be placed anywhere within a document. • cannot be nested within other CDATA sections. • cannot be empty.

The only sequence of symbols that may not occur within a CDATA section is]] because this is the marker ending a CDATA section.

Esempio: The following example shows an element named `htmlCode` containing a CDATA section used to store several HTML tags: `<htmlCode><h1>SJB Pet Boutique</h1><h2>Fashion for Pets and Their Humans</h2>]]</htmlCode></code></p></div><div data-bbox="156 915 913 958" data-label="Text"><p>You can use CDATA blocks when you want to include large</p></div><div data-bbox="930 930 988 953" data-label="Page-Footer"></div><div data-bbox="58 966 186 990" data-label="Page-Footer"><p>A.M. Del Grosso</p></div><div data-bbox="296 966 748 992" data-label="Page-Footer"><p>Codifica di Testi - Introduzione XML Markup a.a. 2018-2019</p></div><div data-bbox="921 966 987 992" data-label="Page-Footer"><p>38/ 100</p></div>`

blocks of special characters as character data

You cannot use XML comments in a CDATA section.

You cannot nest a CDATA section inside another CDATA section

White Space

White space refers to nonprintable characters such as spaces, new lines, tabs. Technically, no white space stripping occurs for element content, which means that the content of the XML element

When white space appears in places other than element content, XML treats it in the following manner: • White space is ignored when it is the only character data between element

tags; this allows XML authors to format their documents to be readable without affecting the content or structure. • White space is ignored within a document's prolog and epilog, and within any element tags. • White space within an attribute value is treated as part of the attribute value.

Esercizio: inserire attributi e CDATA section

Inserting a Processing Instruction A processing instruction is a command that tells an XML parser how to process the document. `<?target instruction ?;`

where target identifies the program (or object) to which the processing instruction is directed and instruction is information that the document passes on to the parser for processing.

Usually the instruction takes the form of attributes and

attribute values

Esempio: `<?xml-stylesheet type="text/css" href="main.css" media="all" ?>` Multiple processing instructions can exist within the same XML document for different media types

Working with Namespaces A namespace is a defined collection of element and attribute names. An XML vocabulary could define a single namespace.

involves two steps: 1. Declare the namespace. 2. Identify the elements and attributes within the document that belong to that namespace.

Add an attribute within the opening tag for the element using the syntax `<element xmlns:prefix="uri">...</element>`

(URI)—a text string that uniquely identifies a resource. The purpose of a URI is simply to provide a unique string of characters that identify a resource. One version of a URI is the Uniform Resource Locator (URL) URLs serve as a built-in mechanism on the web for generating unique addresses Note that although a URI doesn't actually need to point to a real site on the web, it's often helpful to place documentation at the site identified by a URI so users can go there to learn more about the XML vocabulary being referenced.

Esempio TEI ns.

In addition, a namespace that has been declared within an element can be applied to any descendant of the element.

The number of namespace attributes that can be declared within an element is unlimited.

root element so that each namespace is available to all elements within the document

You can declare a default namespace by omitting the prefix in the namespace declaration. Any descendant element or attribute is then considered part of this namespace unless a different namespace is declared within one of the child elements.

```
<element xmlns="uri" > ... </element>
```

Esempio TEI

Declaring a Namespace • To declare a namespace for an element within an XML document, add the xmlns:prefix attribute to the opening tag of the element using the syntax `<element xmlns:prefix="uri" > ... </element>` where element is

the element in which the namespace is declared, prefix is the namespace prefix, and uri is the URI of the namespace. • To declare a default namespace, add the xmlns attribute without specifying a prefix, as follows: `<element xmlns="uri"> ...`

Progress status

i/elementi

- 1 I linguaggi di codifica
- 2 Fondamenti del linguaggio XML
- 3 Validare XML**
 - DTD
 - XSD
 - RELAXGN
- 4 Conclusioni

Elementi per la definizione degli schemi xml

principi

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

we need to make sure that the data that we receive follows a certain XML structure and should contain values which are coherent.

Your function needs to make sure that the caller passes correct XML data. You could make use of an XML Schema to perform this validation.

Performing such validations without the help of a SCHEMA will be extremely difficult most of the time.

Elementi per la definizione degli schemi xml

principi

Make sure that the XML document is structured exactly the way your function expects it to be.

We need an XML schema when we need to make sure that the XML document that we need to work with is in the expected format. Make sure that the values of elements and attributes are within the accepted range.

When data is managed and exchanged in XML format, there needs to be clear agreement about the structure of the XML document.

There needs to be a contract between the caller and the callee about the XML document being exchanged.

validate the XML document to make sure that it adheres to the format defined in the contract.

Elementi per la definizione degli schemi xml

principi

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

A Schema provides such the contract.
It defines the structure of the XML document.
It defines rules to validate the value of elements and attributes
as well as their formats.
Once a schema is defined, a Schema Validator can validate an
XML document against the rules defined in the Schema.

Elementi per la definizione degli schemi xml

Tipi di formalismi per definire schemi XML

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD
XSD
RELAXGN

Conclusioni

DTD, XDR, SOX, Schematron, DSD, DCD, DDML, RELAX
NG

Elementi per la definizione degli schemi xml

Principi Document Type Definition

Esempio DTD

An XML document may have a reference to an external DTD file or can have the DTD embedded as part of the XML file. The XML document given below has embedded DTD information.

```
<?xml version="1.0"?>
<!DOCTYPE Employee [
  <ELEMENT Name {
    {#PCDATA}
  }>
  <ELEMENT First {
    {#PCDATA}
  }>
  <ELEMENT Middle {
    {#PCDATA}
  }>
  <ELEMENT Last {
    {#PCDATA}
  }>
]>
<Employee>
  <Name>
    <First>Jacob</First>
    <Middle>Sebastian</Middle>
    <Last>
      <Name>
        <First>Jacob</First>
        <Middle>Sebastian</Middle>
        <Last>Sebastian</Last>
      </Name>
    </Last>
  </Name>
</Employee>
```

The example given below shows an XML document that refers to an external DTD file.

```
<?xml version="1.0"?>
<!DOCTYPE Employee SYSTEM "employee.dtd">
<Employee>
  <Name>
    <First>Jacob</First>
    <Middle>Sebastian</Middle>
    <Last>
      <Name>
        <First>Jacob</First>
        <Middle>Sebastian</Middle>
        <Last>Sebastian</Last>
      </Name>
    </Last>
  </Name>
</Employee>
```

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

Elementi per la definizione degli schemi xml

Principi Document Type Definition

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML
DTD
XSD
RELAXGN

Conclusioni

Document Type Definition (DTD)

A document type definition describes the rule for an xml document structure, declaring the elements, attributes and entities that are part of the xml document

Progress status

Codifica di
Testi -
Introduzione
XML Markup
a.a.
2018-2019

A.M. Del
Grosso

I linguaggi di
codifica

Fondamenti
del linguaggio
XML

Validare XML

DTD

XSD

RELAXGN

Conclusioni

1 I linguaggi di codifica

2 Fondamenti del linguaggio XML

3 Validare XML

4 Conclusioni