



# Elementi di modellazione e programmazione di contenuti digitali

**Corso di Laurea Magistrale in Metodologie  
informatiche per le discipline umanistiche**

*Ing. Michela Paolucci*

Department of Systems and Informatics

University of Florence

Via S. Marta 3, 50139, Firenze, Italy

tel: +39-055-4796523, fax: +39-055-4796363

**Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet**

<http://www.disit.dsi.unifi.it/>

[paolucci@dsi.unifi.it](mailto:paolucci@dsi.unifi.it)

<http://www.dsi.unifi.it/~nesi>, <http://www.dsi.unifi.it/~paolucci>,

<http://www.axmedis.org> <http://mobmed.axmedis.org/>





# Elementi di modellazione e programmazione di contenuti digitali

**Corso di Laurea Magistrale in Metodologie  
informatiche per le discipline umanistiche**

- **Parte I: xml-schema**
- **Parte II: Esercitazione**





# Storia XML Schema

- | E' una delle attività del working group su XML. E` diventata recommendation nel 2001.
- | E' suddiviso nelle seguenti tre parti:
  - ♣ XML Schema Part 0: Primer (un'introduzione)
    - <http://www.w3.org/TR/xmlschema-0/>
  - ♣ XML Schema Part 1: Structures (struttura del documento XML Schema)
    - <http://www.w3.org/TR/xmlschema-1/>
  - ♣ XML Schema Part 2: Datatypes (modello dei dati e meccanismi di estensione dei tipi)
    - <http://www.w3.org/TR/xmlschema-2/>





# Come definire un xml-schema (1)

- | **Analogamente ad un Dtd**, un XML Schema è una descrizione formale di una grammatica (tipo di documento) per un linguaggio di mark-up basato su XML
- | Tuttavia, se si ha bisogno di un maggiore controllo sugli elementi che possono trovarsi all'interno di uno specifico tipo di documenti XML, i Dtd non risultano più sufficienti
- | un XML-Schema, a differenza di un Dtd che utilizza una propria sintassi specifica, usa la **stessa** sintassi XML per definire la grammatica di un linguaggio di mark-up
- | Un XML-Schema è un documento XML che descrive la grammatica di un linguaggio XML utilizzando un linguaggio di mark-up specifico





## Come definire un xml-schema (2)

- | In quanto documento XML, un XML Schema ha un root element che contiene tutte le regole di definizione della grammatica
- | La **struttura generale** di uno schema XML è la seguente:

♣ <?xml version="1.0"?>

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

... Definizione della grammatica ...

```
</xs:schema>
```

- | L'elemento root del documento è rappresentato dal tag **<xs:schema>** che indica al parser che in questo documento saranno utilizzati dei tag definiti dal namespace standard del W3C
- | I namespace rappresentano un meccanismo per identificare tag appartenenti ad una specifica grammatica. Nel nostro caso questi **tag speciali** sono caratterizzati dal prefisso **xs**:





## Come definire un xml-schema (3)

- | XML-Schema prevede il tag **<xs:element>** per la definizione degli elementi utilizzabili in un documento XML, specificando nell'attributo name il nome del relativo tag.
- | All'interno di ciascun tag **<xs:element>** si può indicare il tipo di dato dell'elemento e definire gli eventuali attributi
- | Ad esempio, la seguente definizione specifica l'elemento testo che può contenere soltanto stringhe:
  - ♣ `<xs:element name="testo" type="xs:string"/>`





# Formato di un XML-schema

- I Un documento di XML Schema è racchiuso in un elemento `<xs:schema>`, e può contenere, in varia forma ed ordine, i seguenti elementi:
- ♣ `<xs:import>` ed `<xs:include>` per inserire, in varia forma, altri frammenti di schema da altri documenti
  - ♣ `<xs:simpleType>` e `<xs:complexType>` per la definizione di tipi denominati usabili in seguito
  - ♣ `<xs:element>` ed `<xs:attribute>` per la definizione di elementi ed attributi **globali** del documento.
  - ♣ `<xs:attributeGroup>` e `<xs:group>` per definire serie di attributi e gruppi di content model complessi e denominati.
  - ♣ `<xs:notation>` per definire notazioni non XML all'interno di un documento XML
  - ♣ `<xs:annotation>` per esprimere commenti per esseri umani o per applicazioni diverse dal parser di XML Schema.





# Tipi di dato semplice e complesso

- | Per comprendere meglio ed apprezzare la potenza degli XML-Schema occorre analizzare nel dettaglio il concetto di tipo di dato: esistono due **categorie di tipi di dato: semplici e complessi**
- | Un tipo **semplice** è un tipo di dati che non può contenere mark-up e non può avere attributi. In pratica è una sequenza di caratteri
- | Un tipo **complesso** è un tipo di dati che può contenere mark-up e avere attributi. E' l'equivalente di un tipo strutturato o misto
- | Gli attributi sono sempre di tipo semplice
- | I tipi complessi e alcuni tipi semplici del **documento istanza** sono definiti nello XML-Schema
- | Gli altri tipi semplici fanno parte del repertorio dei tipi semplici *built-in* di XML-Schema





# XML-Schema: tipo di dato semplice

- | XML Schema introduce il concetto di **tipo di dato semplice** per definire gli elementi che non possono contenere altri elementi e non prevedono attributi
- | Si possono usare tipi di dato semplici predefiniti oppure è possibile personalizzarli
- | Alcuni **tipi di dato predefiniti** sono riportati nella tabella a fianco
- | Esempio:

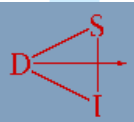
<b>xs:string</b>	Stringa di caratteri
<b>xs:integer</b>	Numero intero
<b>xs:decimal</b>	Numero decimale
<b>xs:boolean</b>	Valore booleano
<b>xs:date</b>	Data
<b>xs:time</b>	Ora
<b>xs:uriReference</b>	URL

♣ `<xs:element name="quantita" type="xs:integer" />`

Questa dichiarazione permette l'uso dell'elemento "quantita" in un documento XML consentendo soltanto un contenuto di tipo intero.

Ovvero: sarà considerato valido l'elemento `<quantita>123</quantita>` mentre non lo sarà l'elemento `<quantita>uno</quantita>`

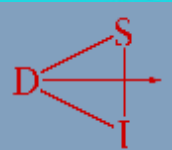
♣ **Riferimento:** <http://www.w3.org/TR/xmlschema-0/#CreatDt>





## XML-Schema: tipo di dato semplice (2)

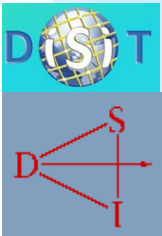
- ♣ **string**: una stringa di caratteri
- ♣ **boolean**: i valori 'true ' e 'false' (1,0)
- ♣ **integer**: interi con segno: ... -1, 0, 1, ...
- ♣ **decimal**: una stringa di numeri (con segno e punto): '-1.23, 0, 123.4, 1000.00'
- ♣ **float**: un reale in notazione scientifica: '12.78E-12 '
- ♣ **duration** : una stringa per una durata temporale nel formato PnYnMnDTnHnMnS. Ad esempio 'P1Y2M3DT10H30M12.3S' corrisponde a : '1 anno, 2 mesi, 3 giorni, 10 ore, e 30 minuti'
- ♣ **date**: una data nel formato anno-mese-giorno: '2009-11-18'
- ♣ **time**: un valore di orario nel formato hh:mm:ss con una appendice opzionale per l'indicazione del fuso orario. Es.: '13:20:00+01:00 ' significa 1:20 PM in Middle European Time (+01:00).
- ♣ **anyURI**: la stringa di un URI, come "http://www.w3.org/"  
Accetta sia URI relativi che assoluti





# XML-Schema: tipo di dato semplice (3)

- | Ogni tipo semplice è caratterizzato da alcune proprietà, dette **facets**, che ne descrivono formati (permessi ed obblighi) e vincoli (Si faccia riferimento a <http://www.w3.org/TR/xmlschema-0/#SimpleTypeFacets>)
- | Alcuni esempi:
  - ♣ **minExclusive, minInclusive, maxInclusive, maxExclusive**
  - ♣ **enumeration, lenght, maxLenght, minLenght, pattern, whitespace**
- | Nuovi tipi semplici (tipi semplici derivati) sono ottenuti per derivazione dai tipi semplici built-in (tipi di dati incorporati)
- | Un tipo semplice non può contenere elementi o attributi
- | I tipi semplici possono essere definiti in uno dei seguenti modi:
  - ♣ **restriction**: limita i valori disponibili per il tipo semplice a un sottoinsieme di quelli del tipo semplice ereditato
  - ♣ **list**: definisce un tipo semplice contenente un elenco di valori, separati da uno spazio vuoto di un tipo semplice ereditato
  - ♣ **union** Definisce un tipo semplice contenente un'unione di valori di due o più tipi semplici ereditati.
- | Dopo aver definito un tipo semplice, questo può essere utilizzato in un attributo, in una dichiarazione dell'elemento oppure in una definizione **complexType** (tipo di dato complesso)





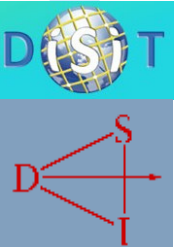
# XML-Schema: tipi semplici derivati per restrizione (restriction) (1)

- | È possibile restringere un tipo semplice built-in per ottenere un nuovo tipo semplice
- | Si utilizza l'elemento **simpleType** per dichiarare un nuovo tipo semplice, l'elemento **restriction** per indicare il tipo base e le possibili **facets** per stabilire il range di valori
- | Se, ad esempio, si ha bisogno di **limitare** il valore che può essere assegnato all'elemento <quantita>, è possibile definirlo nel seguente modo:

```
<xs:element name="quantita">  
  <xs:simpleType>  
    <xs:restriction base="xs:integer">  
      <xs:minInclusive value="1" />  
      <xs:maxInclusive value="100" />  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

```
documento.xml:  
[...]  
<quantita>  
  99  
</quantita>  
[...]
```

- | In questo caso, la dichiarazione indica che l'elemento <quantita>:
  - ♣ è di tipo **semplice**
  - ♣ prevede una **restrizione** sul tipo di dato intero predefinito accettando valori compresi tra **1** e **100**





# XML-Schema: tipi semplici derivati per restrizione (restriction) (2)

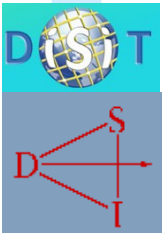
| Altro esempio (Facet enumeration):

```
<xs:simpleType name="shape">  
  <xs:restriction base="xsd:string">  
    <xs:enumeration value="circle"/>  
    <xs:enumeration value="triangle"/>  
    <xs:enumeration value="square"/>  
  </xs:restriction>  
</xs:simpleType>
```

```
<shape>triangle</shape>
```

```
<shape>circle</shape>
```

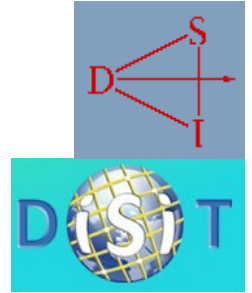
```
<shape>square</shape>
```





# XML-Schema: tipi s. derivati per unione (union)

Nell'esempio seguente viene illustrato un tipo semplice (allframesize) che rappresenta l'union di altri due tipi semplici, i quali definiscono, a loro volta, gli insiemi di valori enumerati. Il primo fornisce, come insieme di valori basati su integer, le dimensioni delle biciclette da strada, mentre il secondo enumera valori di stringa per le dimensioni delle mountain bike ('large', 'medium', 'small').



```
<xs:attribute name="allframesize" />
```

```
<xs:simpleType>
```

```
<xs:union>
```

```
<xs:simpleType>
```

```
<xs:restriction
```

```
base="roadbikesize"/>
```

```
</xs:simpleType>
```

```
<xs:simpleType>
```

```
<xs:restriction
```

```
base="mountainbikesize"/>
```

```
</xs:simpleType>
```

```
</xs:union>
```

```
<bikesize allframesize="large" /> oppure </bikesize allframesize="46" >
```

```
</xs:simpleType>
```

```
<xs:simpleType name="roadbikesize">
```

```
<xs:restriction base="xs:positiveInteger">
```

```
<xs:enumeration value="46"/>
```

```
<xs:enumeration value="52"/>
```

```
<xs:enumeration value="55"/>
```

```
</xs:restriction>
```

```
</xs:simpleType>
```

```
<xs:simpleType name="mountainbikesize">
```

```
<xs:restriction base="xs:string">
```

```
<xs:enumeration value="small"/>
```

```
<xs:enumeration value="medium"/>
```

```
<xs:enumeration value="large"/>
```

```
</xs:restriction>
```

```
</xs:simpleType>
```

```
</xs:schema>
```



# XML-Schema: tipi semplici derivati per lista (list)

- I Nell'esempio seguente viene illustrato un tipo semplice (listOfDates) che consente di ottenere un elenco di date (le voci devono essere separate l'una dall'altra con uno spazio vuoto) come contenuto:

```
<xs:simpleType name="listOfDates">  
  <xs:list itemType="xs:date"/>  
</xs:simpleType>
```

- I File.xml:

```
<listOfDates>  
  2009-11-18 2000-01-23  
</listOfDates>
```

- I Riferimento: <http://www.w3.org/TR/xmlschema-0/#CreatDt>





# XML-schema: tipo di dato complesso

- I **tipi di dato complesso** si riferiscono ad elementi che possono contenere altri elementi e possono avere attributi
- Definire un elemento di tipo complesso corrisponde a definire la relativa struttura
- Lo schema generale per la definizione di un **elemento di tipo complesso** è il seguente:

```
<xs:element name="NOME_ELEMENTO">  
  <xs:complexType>  
    ... Definizione del tipo complesso ...  
    ... Definizione degli attributi ...  
  </xs:complexType>  
</xs:element>
```





# XML-schema: def. tipo di dato complesso

- I tipi di dato complesso sono elementi che possono:
  - ♣ contenere altri elementi
  - ♣ avere degli attributi

È possibile definire la sequenza di elementi che possono stare al suo interno utilizzando uno dei seguenti **costruttori di tipi complessi**:

- ♣ **<xs:sequence>** Consente di definire una sequenza ordinata di sottoelementi
- ♣ **<xs:choice>** Consente di definire un elenco di sottoelementi alternativi
- ♣ **<xs:all>** Consente di definire una sequenza non ordinata di sottoelementi





# Tipo di dato complesso: sequenza (sequence)

- I Il costruttore di tipo complesso `<xs:sequence>` si utilizza per definire un elemento complesso ottenuto come una sequenza **ordinata** di sotto elementi:

```
<xs:element name="articolo">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="paragrafo"/>
      <xs:element name="testo"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
file.xml
[...]
<articolo>
  <paragrafo>
    Questo è il
    paragrafo
    1.. Introduzione...
  </paragrafo>
  <testo>
    contenuto di testo
  </testo>
</articolo>
[...]
```





# Tipo di dato complesso: scelta (choice)

- Il costruttore di tipo complesso `<xs:choice>` si utilizza per definire un elenco di sottoelementi alternativi:

```
<xs:complexType name="articolo">  
  <xs:choice>  
    <xs:element name="paragrafo"/>  
    <xs:element name="testo"/>  
  </xs:choice>  
</xs:complexType>
```

```
file1.xml  [...]  
<articolo>  
  <testo>  
    contenuto testuale  
  </testo>  
</articolo>  
[...]
```

```
file2.xml  [...]  
<articolo>  
  <paragrafo>  
    contenuto testuale  
  </paragrafo>  
</articolo> [...]
```





# Tipo di dato complesso: all

- I Il costruttore di tipo complesso `<xs:all>` si usa per definire un elemento complesso ottenuto come una sequenza **non ordinata** di sottoelementi:

```
<xs:element name="articolo">
  <xs:complexType>
    <xs:all>
      <xs:element name="paragrafo"/>
      <xs:element name="testo"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

```
file1.xml [...]
<articolo>
  <paragrafo> paragrafo
</paragrafo>
  <testo>
    effettivo contenuto di
  testo
  </testo>
</articolo> [...]
```

```
file2.xml [...]
<articolo>
  <testo> effettivo contenuto di testo
</testo>
  <paragrafo> paragrafo
</paragrafo>
</articolo> [...]
```





# Tipo di dato complesso: combinazioni

- I E' possibile utilizzare i costruttori in modo integrato

```
<xs:element name="mezzoLocomozione">
```

```
<xs:complexType>
```

```
<xsd:sequence>
```

```
<xsd:element name="proprietario" type="xsd:string"/>
```

```
<xsd:choice>
```

```
<xsd:element name="ruote" type="xsd:integer"/>
```

```
<xsd:element name="ali" type="xsd:string"/>
```

```
</xsd:choice>
```

```
</xsd:sequence>
```

```
</xs:complexType>
```

```
</xs:element>
```

```
file.xml [...]  
<mezzoLocomozione>  
  <proprietario>Mario Rossi</proprietario>  
  ...<ruote>2</ruote>  
</mezzoLocomozione> [...]
```

```
[...] <mezzoLocomozione>  
  <proprietario>Franco Bosi</proprietario>  
  ...<ali>numero serie xxxx</ali>  
</mezzoLocomozione> [...]
```





# Tipo complesso: minOccurs e maxOccurs

- Per ciascuno dei costruttori visti (**sequence**, **choice**, **all**) e per ciascun elemento è possibile definire il numero di occorrenze previste utilizzando gli attributi **minOccurs** e **maxOccurs**
- Esempio: se si vuole costruire un tipo complesso in cui l'elemento **testo** può essere presente una o infinite volte all'interno di un paragrafo, si può esprimere questa condizione nel seguente modo:

```
<xs:element name="paragrafo">  
  <xs:complexType>  
    <xs:element name="testo" minOccurs="1"  
      maxOccurs="unbounded"/>  
  </xs:complexType>  
</xs:element>
```

~~<paragrafo><paragrafo>~~

```
<paragrafo>  
<testo> testo vero</testo>  
</paragrafo>
```

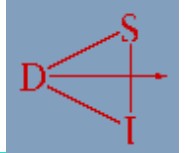
- In questo caso il valore **unbounded** indica che non è stabilito un massimo numero di elementi testo che possono stare all'interno di un paragrafo.

```
<paragrafo>  
<testo> primo testo </testo>  
<testo> secondo testo </testo>  
</paragrafo>
```





# Tipo di dato complesso: attributi



- I La definizione degli attributi è basata sull'uso del tag **<xs:attribute>**, come nel seguente esempio:
  - ♣ `<xs:attribute name="titolo" type="xs:string" use="required" />`
- I dove l'attributo **use** serve per specificare alcune caratteristiche come la presenza obbligatoria (**required**) o un valore predefinito (**default**) in combinazione con l'attributo **value**.
- I Si noti che: se non si specifica esplicitamente l'obbligatorietà dell'attributo, esso è considerato **opzionale**

```
<xs:element name="paragrafo">
  <xs:complexType>
    <xs:element name="testo"
      minOccurs="1" maxOccurs="unbounded"/>
  </xs:complexType>
  <xs:attribute name="titolo"
    type="xs:string" use="required" />
</xs:element>
```

```
<paragrafo titolo="Par. 1.0">
  <testo> primo testo </testo>
  <testo> secondo testo
  ...</testo>
</paragrafo>
```



# Tipi anonimi e tipi denominati

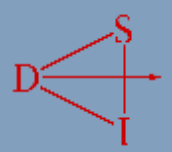
Si parla di tipi anonimi quando:

- ♣ Non è presente attributo type nella dichiarazione dell'elemento/attributo
- ♣ Non è assegnato un nome al complex/simpleType

ANONIMO:

```
<xs:complexType>  
  <xs:choice>  
    <xs:element name="paragrafo"/>  
    <xs:element name="testo"/>  
  </xs:choice>  
</xs:complexType>
```

```
<xs:complexType name="articolo" type="xsd:string">  
  <xs:choice>  
    <xs:element name="paragrafo"/>  
    <xs:element name="testo"/>  
  </xs:choice>  
</xs:complexType>
```



# XML-schema: esempio (poco leggibile)



```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="articolo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="paragrafo" maxOccurs="unbounded">
          <xs:complexType>
            <xs:all maxOccurs="unbounded">
              <xs:element name="immagine" minOccurs="0">
                <xs:complexType>
                  <xs:attribute name="file" use="required">
                    <xs:simpleType> <xs:restriction base="xs:string"/>
                  </xs:simpleType>
                </xs:attribute>
              </xs:complexType>
            </xs:element>
            <xs:element name="testo"/>
            <xs:element name="codice" minOccurs="0"/>
          </xs:all>
          <xs:attribute name="titolo" type="xs:string"
            use="optional"/>
          <xs:attribute name="tipo" use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="abstract"/>
                <xs:enumeration value="bibliografia"/>
                <xs:enumeration value="note"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute
      name="titolo"
      type="xs:string"
      use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```





# XML-schema: definizione modulare degli elementi (1)

- | XML Schema prevede la possibilità di rendere modulare la definizione della struttura di un documento XML tramite la dichiarazione di tipi e di elementi
- | Questo contribuisce a fornire una **struttura modulare** al XML-Schema, più ordinata, più comprensibile e semplice da modificare: un XML-Schema diventa una sequenza di dichiarazioni di tipi ed elementi:

```
♣ <xs:complexType name="nome_elemento">  
...  
</xs:complexType>
```

Il riferimento ad una dichiarazione di tipo viene fatta come se fosse un tipo predefinito, come mostrato nel seguente esempio:

```
<xs:element name="Nuovo">  
  <xs:complexType>  
    <xs:element type="nome_elemento"/>  
  </xs:complexType>  
</xs:element>
```





# XML-schema: definizione modulare degli elementi (2)

- | La possibilità di dichiarare elementi e tipi di dato implica l'esistenza di un **ambito di visibilità**:
  - ♣ Una definizione si dice **globale** se è posta all'interno del tag `<schema>`. In questo caso l'elemento o l'attributo può essere riutilizzato in ogni punto del documento
  - ♣ Una definizione si dice **locale** se è inserita all'interno di un tag `<complexType>`. In questo caso l'elemento o l'attributo esiste solo se esiste un'istanza di quel tipo, e non può essere riutilizzato fuori della dichiarazione del tipo complesso





# XML-schema: definizione modulare degli elementi (3) – [Tipo complesso]

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/... ">
  <xs:complexType name="paragrafoType">
    [...]
  </xs:complexType>

  <xs:element name="articolo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="paragrafo"
          type="paragrafoType"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="titolo" type="xs:string"
        use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

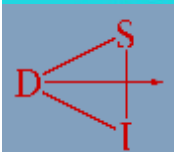
- I componenti di uno schema dichiarati al livello massimo, cioè come sotto elementi di root, sono dichiarati a livello globale e possono essere utilizzati nel resto dello schema
- Es: paragrafoType



# XML-schema: definizione modulare degli elementi (esempio) – [Tipo complesso]



```
<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="articolo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="paragrafo"
          type="paragrafoType"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="titolo" type="xs:string"
        use="required"/>
    </xs:complexType>
  </xs:element>
```

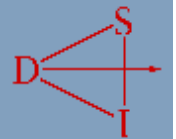


[...] si veda la slide successiva



```
<xs:complexType name="paragrafoType">
  <xs:all maxOccurs="unbounded">
    <xs:element name="immagine" type="immagineType"
      minOccurs="0"/>
    <xs:element name="testo"/>
    <xs:element name="codice" minOccurs="0"/>
  </xs:all>
  <xs:attribute name="titolo" type="xs:string"
    use="optional"/>
  <xs:attribute name="tipo" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="abstract"/>
        <xs:enumeration value="bibliografia"/>
        <xs:enumeration value="note"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:complexType name="immagineType">
  <xs:attribute name="file" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string"/>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
</xs:schema>
```

**(esempio completo)**





# Annotazioni

| In XML Schema, esiste un tag specifico da usare in caso si voglia mettere commenti, note ed istruzioni: l'elemento `<annotation>`

| L'elemento `<annotation>` può contenere elementi:

- ♣ `<documentation>`, creati per essere letti da esseri umani
- ♣ `<appInfo>`, pensati per essere digeriti da applicazioni specifiche

| Sintassi:

| `<xs:annotation>`

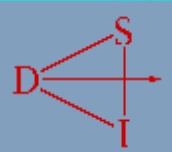
`<xs:appInfo>Appxxx Note</xs:appInfo>`

`<xs:documentation xml:lang="en">`

        This Schema defines a Note...

`</xs:documentation>`

`</xs:annotation>`





# Collegare un xml-schema ad un file xml

- | A partire da una grammatica definita tramite un XML-Schema, è possibile sfruttare un parser XML validante per **verificare la validità** di un documento XML
- | Il parser avrà bisogno:

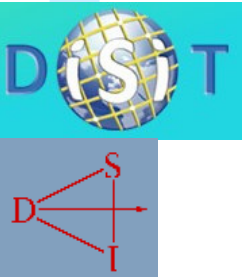
- ♣ del documento XML da validare
- ♣ dello schema XML rispetto a cui effettuare la validazione

- | Ci sono diversi modi per fornire al parser informazioni sullo schema da usare per la validazione. Uno di questi è il seguente:

- ♣ inserire nel documento XML un riferimento allo schema da usare associato all'elemento root, come nel seguente esempio:

```
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="articolo.xsd" titolo="documento XML" >
```

- **xmlns:xsi** indica un URL che specifica la modalità con cui si indicherà il riferimento allo schema XML
- **xsi:noNamespaceSchemaLocation** indica il nome e l'eventuale percorso del file contenente lo schema XML di riferimento.





# XML-schema: namespace (1)

- | Una delle caratteristiche auspicabili nella creazione di un nuovo linguaggio è la possibilità di integrare elementi derivanti da grammatiche diverse (definite in xml-schema differenti) in modo da **riutilizzare parti di grammatiche** già definite
- | Tuttavia la **composizione di linguaggi** pone almeno due tipi di problemi:
  - ♣ un documento che usa due grammatiche presenta il **problema della validazione**: è necessario capire in modo univoco a quale schema si deve fare riferimento per validare un documento XML "ibrido"
  - ♣ due linguaggi potrebbero avere tag ed attributi con lo stesso nome, anche se utilizzabili in contesti diversi
- | Per risolvere questa ambiguità si ricorre all'uso dei **namespace**: un namespace è un insieme di nomi di elementi e nomi di attributi identificati univocamente da un identificatore.





## XML-schema: namespace (2)

- | **L'identificatore univoco** individua l'insieme dei nomi distinguendoli da eventuali omonimie in altri namespace
- | Il concetto non è nuovo nell'informatica. Esempio: **definizione** dei nomi dei campi in una tabella di un database:
  - ♣ Non è possibile avere campi con lo stesso nome all'interno di una tabella, ma è possibile avere gli stessi nomi in tabelle diverse. In questo modo si risolve l'ambiguità tra due campi omonimi facendoli precedere dal nome della tabella (il namespace)
- | Se in un documento XML si utilizzano **elementi definiti in xml schema diversi** abbiamo bisogno di un meccanismo che permetta di identificare ciascun namespace e il relativo XML Schema che lo definisce

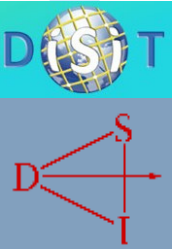




# XML-schema: sintassi dei namespace (1)

- | In un documento XML si fa riferimento ad un namespace utilizzando un attributo speciale (**xmlns**) associato al root element:
  - ♣ `<articolo xmlns="http://www.dominio.it/xml/articolo">`
- | Questo indica che l'elemento articolo ed i suoi sottoelementi usano i nomi definiti nel namespace identificato dall'identificatore `http://www.dominio.it/xml/articolo`
- | L'identificatore di un namespace può essere rappresentato da una qualsiasi stringa **univoca**. Solitamente si usa **un URI** (Uniform Resource Identifier, RFC 3986)

NOTA: **RFC (Request for Comments)** è un documento che riporta informazioni o specifiche riguardanti nuove ricerche, innovazioni e metodologie prevalentemente in ambito informatico o legate alle comunicazioni in rete. Una volta vagliati dall'**IETF (Internet Engineering Task Force, RFC 3935)** possono diventare degli *standard Internet*.





## XML-schema: sintassi dei namespace (2)

- I Per mettere in relazione un namespace con il relativo XML Schema occorre dichiararlo nel root element:

```
<articolo
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.dominio.it/xml/articolo"
  xmlns="http://www.dominio.it/xml/bibliografia"
  xsi:schemaLocation="http://www.dominio.it/xml/articolo
  articolo.xsd"
  xsi:schemaLocation="http://www.dominio.it/xml/bibliografia
  bibliografia.xsd">
```

dove:

- **xmlns:xsi** specifica la modalità con cui viene indicato il riferimento allo schema
- **xsi:schemaLocation** indica il namespace ed il file in cui è definito il relativo XML Schema separati da uno spazio
- I E' possibile **combinare più namespace** facendo in modo che ciascun elemento utilizzato faccia riferimento al proprio namespace
- I Si noti che quando si fa **riferimento ad un namespace**, questo riferimento vale per l'elemento corrente e per tutti gli elementi contenuti, a meno che non venga specificato un diverso namespace





# XML-schema: namespace esempio

```
| <articolo xmlns="http://www.dominio.it/xml/articolo"  
titolo="documento XML">  
  <paragrafo titolo="Introduzione">  
    <testo>  
      bla bla bla  
    </testo>  
  </paragrafo>  
  <paragrafo titolo="Bibliografia">  
    <bibliografia  
      xmlns="http://www.dominio.it/xml/bibliografia">  
      <autore>  
        Tizio  
      </autore>  
      <titolo>  
        Opera citata  
      </titolo>  
      <anno>  
        1999  
      </anno>  
    </bibliografia>  
  </paragrafo>  
</articolo>
```

*In questo esempio si usano elementi tratti da due diversi namespace:*

- uno relativo alla grammatica della struttura di un articolo*
- l'altro relativo alla grammatica di bibliografia*





# Presentazione di XML con CSS

- | A differenza di HTML, che è un linguaggio specifico di strutturazione e presentazione di documenti, XML è più generale e non ha una **semantica di presentazione**. Non è previsto alcun meccanismo predefinito per visualizzare i vari elementi di un documento.
- | Ad esempio, un documento **XML visualizzato in un browser** appare generalmente così com'è, al massimo con una indentazione e una colorazione dei tag impostata dal browser.
- | Un metodo per gestire la **presentazione del contenuto** di un documento XML consiste nell'utilizzare i Cascading Style Sheets (CSS).
- | È possibile **utilizzare i CSS** in modo analogo a come si utilizzano con HTML. Per ciascun elemento del documento XML che vogliamo formattare occorre definire una regola secondo lo schema:
  - | **Selettore { proprietà: valore; proprietà: valore; ... }**





# Presentazione di XML con CSS

- | Il **selettore** specifica a quale elemento la regola deve essere applicata, mentre la parte racchiusa tra parentesi graffe elenca le caratteristiche da impostare e il relativo valore.
- | È opportuno evidenziare una **importante differenza** tra l'utilizzo dei CSS per formattare documenti HTML e il loro uso per i documenti XML. In HTML la maggior parte dei tag ha una formattazione predefinita e pertanto un foglio di stile CSS consente di ridefinire tali impostazioni.
- | In XML i tag non hanno alcun significato di formattazione, pertanto è necessario **specificare tutto**. Ad esempio, senza l'opportuna indicazione il testo contenuto nei diversi elementi di un documento XML verrebbe visualizzato come un'unica stringa.





# Presentazione di XML con CSS

- | Per **strutturare visivamente il documento** dobbiamo indicare la modalità di visualizzazione di ciascun elemento tramite la proprietà `display` di CSS. Ad esempio, per formattare l'elemento paragrafo di un articolo possiamo definire una regola come la seguente:
  - ♣ `paragrafo {display: block; font-size: 12pt; text-align: left}`
- | Nel documento XML possiamo quindi **inserire un riferimento** ad esso mediante un'apposita direttiva di elaborazione, come nel seguente esempio:
  - | `<?xml-stylesheet type="text/css" href="stile.css" ?>`





# Presentazione di XML con CSS: esempio

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="stile.css" ?>
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
titolo="Titolo dell'articolo">
  <paragrafo titolo="Titolo del primo paragrafo">
    <testo> Blocco di testo del primo paragrafo</testo>
    <immagine file="immagine1.jpg"/>
  </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>
```

**stile.css**

**paragrafo** {display: block; font-size: 12pt; text-align: left}

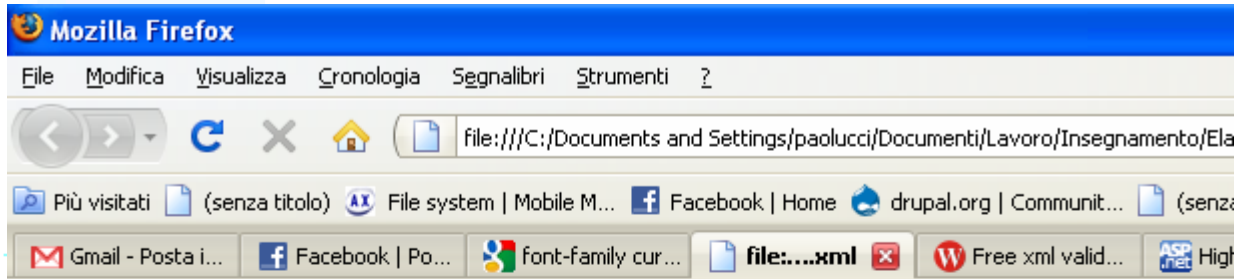
**testo** {display: block; font-size: 12pt; text-align: left; color: green}

**codice** {display: block; font-size: 20pt; font-family: Courier; text-align: left; color: blue}





# Presentazione di XML con CSS: visualizzazione esempio



Blocco di testo del primo paragrafo

Blocco di testo del secondo paragrafo

Esempio di codice

Altro blocco di testo

Riferimento ad un articolo

**stile.css**

```
paragrafo {display: block; font-size:  
12pt; text-align: left}
```

```
testo {display: block; font-size:  
12pt; text-align: left; color: green}
```

```
codice {display: block; font-size:  
20pt; font-family: Courier; text-align:  
left; color: blue}
```



# Link utili

- | <http://www.w3.org/XML>
- | <http://www.w3.org/TR/xmlschema-0/>
- | <http://www.w3.org/2004/11/uri-iri-pressrelease.html.en>
- | <http://www.ietf.org/rfc/rfc3986.txt>
- | <http://www.w3.org/TR/xmlschema-1/>
- | <http://www.altova.com/xml-editor/>
- | <http://www.w3.org/TR/html4/sgml/entities.html>
- | <http://www.html.it/>



# XML Parte III: Esercitazione





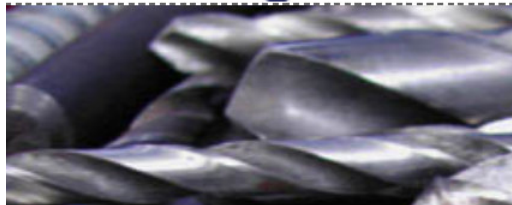
# Introduzione

- | Per controllare che un file xml sia ben formato:
  - ♣ scaricare il plugin per firefox: OpenXMLViewer
  - ♣ Usare direttamente Internet Explorer8

---

- | Per la validazione:
  - ♣ xml-spy (<http://www.altova.com/xml-editor/>) è proprietario ma esiste una versione free per 30 gg
  - ♣ Online schema validator:  
<http://tools.decisionsoft.com/schemaValidate/>





### Terms of use for CoreFiling online tools

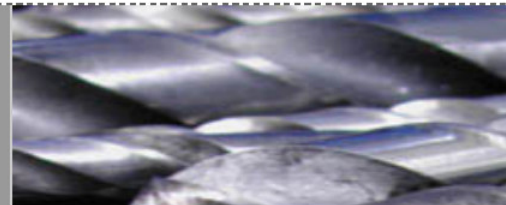
© Copyright 1998-2005 CoreFiling Limited

Use of this site shall be governed by the following terms and conditions.

This software is being licensed to you by CoreFiling Limited in A FREE DEMONSTRATION VERSION ONLY and is provided for evaluation or personal non-profit use only. Commercial usage of this product is strictly prohibited without prior permission from CoreFiling Limited. Please **contact us** to discuss possible commercial applications for these tools.

You are welcome to link to these pages but if you wish to provide your own front-end then we require you to **contact us** first for permission.

COREFILING LIMITED MAKES NO REPRESENTATIONS ABOUT THE ACCURACY OR SUITABILITY OF THIS MATERIAL FOR ANY PURPOSE. IT IS PROVIDED 'AS IS', WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES.



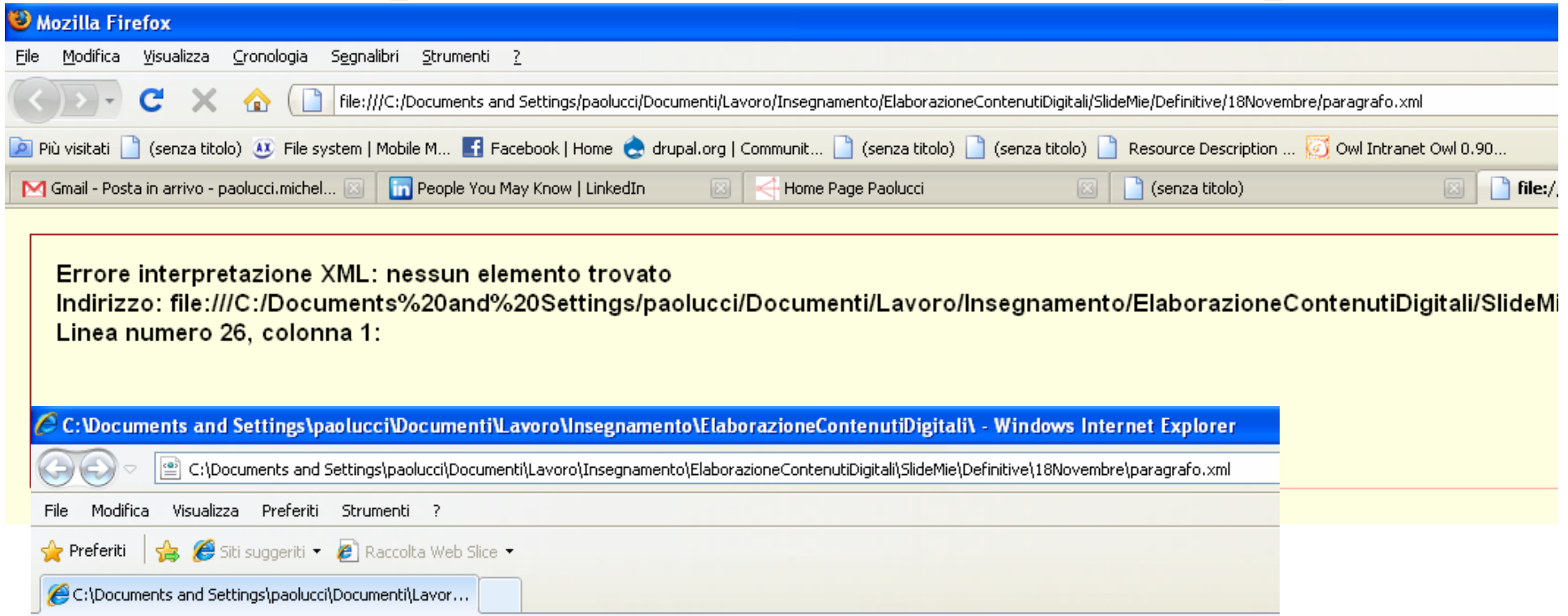
#### TOOLS

- Home
- xmlpp
- xmldiff
- Schema Validation

#### COREFILING

- Corporate
- Open Source
- Contact
- Terms of Use

# Firefox+OpenXMLViewer / Internet Explorer8



## Impossibile visualizzare la pagina XML

Impossibile visualizzare l'input XML tramite il foglio di stile XSL.  
Correggere l'errore, quindi fare clic su [Aggiorna](#), oppure riprovare in un momento successivo.

**I seguenti tag non sono stati chiusi: articolo. Errore durante l'elaborazione della risorsa "file:///C:/Documents and Setti..."**

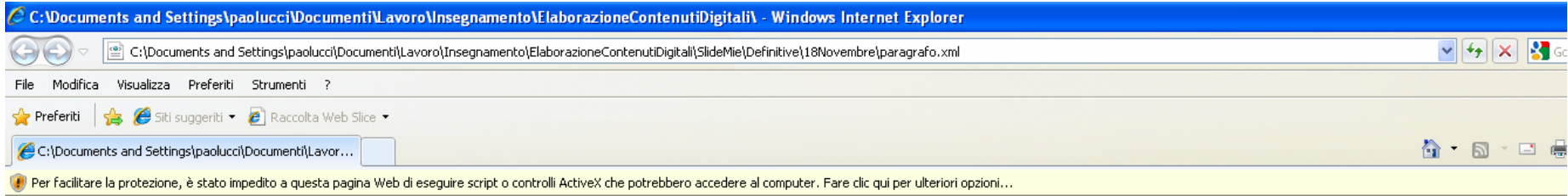
Browser:  
Segnalazione  
di errore



- `<articolo titolo="Come definire un documento xml">`
- `<paragrafo titolo="Introduzione">`
- `<testo>`  
Questo documento è di prova per la definizione di file xml ...  
`</testo>`
- `</paragrafo>`
- `<paragrafo titolo="Descrizione di un documento xml">`
- `<testo>`  
In questo paragrafo oltre al testo metto un'immagine:  
`<testo>`  
`<immagine titolo="immagine" formato="jpg"/>`  
`<testo>` Adesso commento l'immagine `</testo>`
- `</paragrafo>`
- `<paragrafo titolo="Ringraziamenti">`  
`<testo>` Ringrazio ..... `</testo>`
- `</paragrafo>`
- `</articolo>`

*Verificare che il file xml sia ben formato aprendolo sul browser*

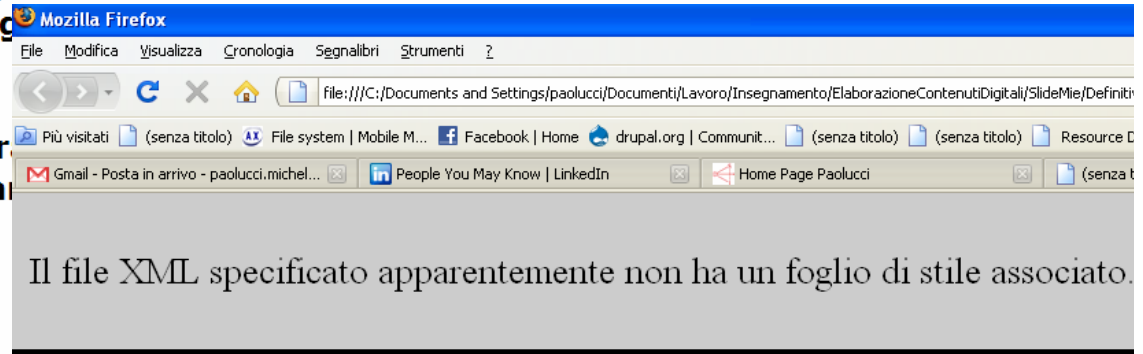




```

<?xml version="1.0" encoding="UTF-8" ?>
- <articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" titolo="Titolo dell'articolo">
- <paragrafo titolo="Titolo del primo paragrafo">
  <testo>Blocco di testo del primo paragrafo</testo>
  <immagine file="immagine1.jpg" />
</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>

```



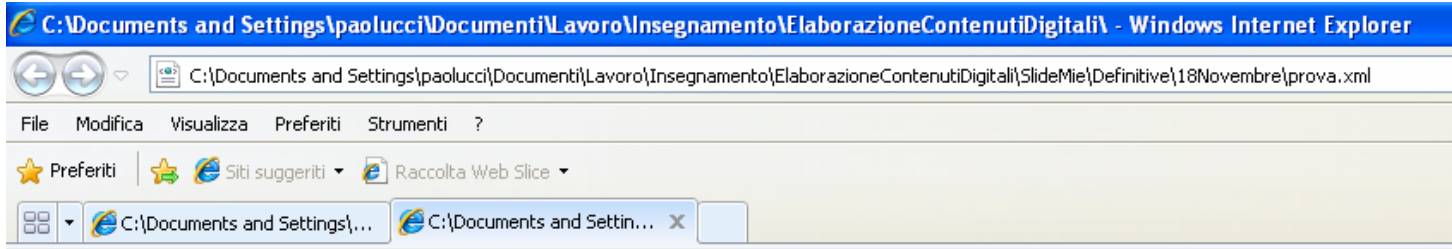
```

- <articolo titolo="Titolo dell'articolo">
- <paragrafo titolo="Titolo del primo paragrafo">
  <testo> Blocco di testo del primo paragrafo </testo>
  <immagine file="immagine1.jpg"/>
</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>

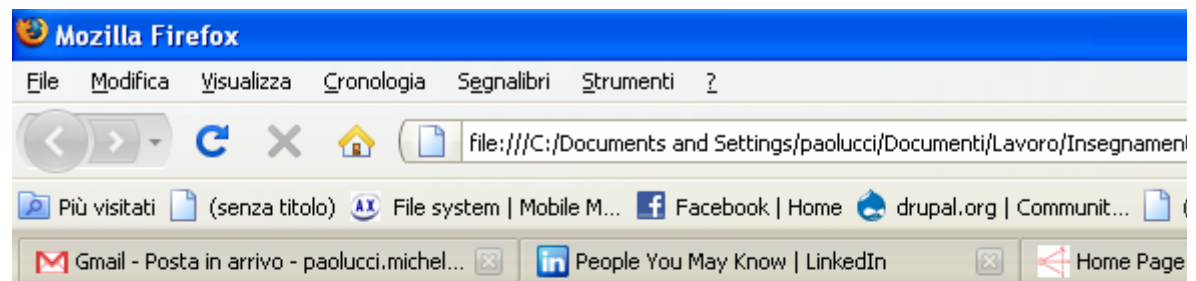
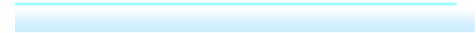
```

Browser:  
Visualizzazione di  
Un documento xml  
Ben formato





Blocco di testo del primo paragrafo  
 Blocco di testo del secondo paragrafo  
**Esempio di codice**  
 Altro blocco di testo  
 Riferimento ad un articolo



Browser:  
 Visualizzazione di  
 Un documento xml  
 Ben formato, con un  
 file ccs associato

Blocco di testo del primo paragrafo  
 Blocco di testo del secondo paragrafo  
**Esempio di codice**  
 Altro blocco di testo  
 Riferimento ad un articolo





# Come definire un documento xml

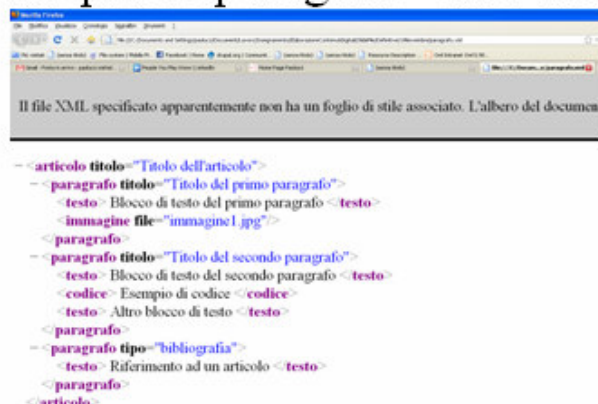
## Introduzione

Questo documento è di prova per la definizione di file xml...



## Descrizione di un documento xml

In questo paragrafo oltre al testo metto un'immagine:



Entità	valore
.	&#8901;
...	&#8230;
è	&#232;

Adesso commento l'immagine

## Ringraziamenti

Ringrazio .....



```
<?xml version="1.0"
encoding="UTF-8"?>
```

```
<articolo
xmlns:xsi="http://www.w3.org/20
01/
XMLSchema-instance"
titolo="Titolo dell'articolo">
[...]TODO
</articolo>
```

Per descrivere l'articolo in esame usare i seguenti tipi di elementi, con i relativi attributi associati:

- ♣ articolo (titolo), paragrafo (titolo), testo, immagine (titolo, formato)

# Come definire un documento xml

## Introduzione

Questo documento è di prova per la definizione di file xml.

## Descrizione di un documento xml

In questo paragrafo oltre al testo metto un'immagine:



```
<articolo titolo="Titolo dell'articolo">
  <paragrafo titolo="Titolo del primo paragrafo">
    <testo> Blocco di testo del primo paragrafo </testo>
    <immagine file="immagine1.jpg">
  </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>
```

Adesso commento l'immagine

## Ringraziamenti

Ringrazio .....

# E

NOTA:

usare &#8230;

e &#232;

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<articolo
```

```
  xmlns:xsi="http://www.w3.org/2001/
```

```
  XMLSchema-instance"
```

```
  titolo="Titolo dell'articolo"> [...] TODO
```

```
</articolo>
```

<articolo> ha un attributo <titolo> (con valore 'Come definire...') e contiene:

- Tre paragrafi

Il primo paragrafo ha l'attributo <titolo> (valore "Introduzione") e contiene:

- Un campo di <testo> (con valore 'Questo documento...')

Il secondo <paragrafo> ha un attributo <titolo> (con valore 'Descrizione di...') e contiene:

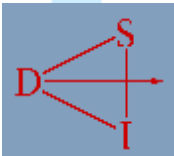
- Un campo di <testo> (con valore 'In questo paragrafo ...')

- Un' <immagine> (con attributi <titolo> e <formato> non specificati nel testo)

- Un altro campo di <testo> (con valore "Adesso...")

Il terzo paragrafo ha sempre l'attributo <titolo> (con valore "Ringraziamenti") e contiene:

- Un campo di <testo> (con valore "Ringrazio...")





```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
titolo="Come definire un documento xml">
```

```
  <paragrafo titolo="Introduzione">
```

```
    <testo>Questo documento &#232; di prova per la definizione di file xml&  
    #8230;    </testo>
```

```
  </paragrafo>
```

```
  <paragrafo titolo="Descrizione di un documento xml">
```

```
    <testo>In questo paragrafo oltre al testo metto un'immagine:
```

```
    </testo>
```

```
    <immagine titolo="immagine" formato="jpg" >
```

```
    <testo>Adesso commento l'immagine
```

```
    </testo>
```

```
  </paragrafo>
```

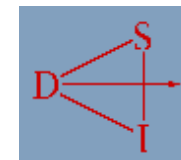
```
  <paragrafo titolo="Ringraziamenti">
```

```
    <testo>Ringrazio &#8230;&#8230;
```

```
    </testo>
```

```
  </paragrafo>
```

```
</articolo>
```



# Come definire un documento xml



## Introduzione

Questo documento è di prova per la definizione di file xml...



## Descrizione di un documento xml

In questo paragrafo oltre al testo metto un'immagine:



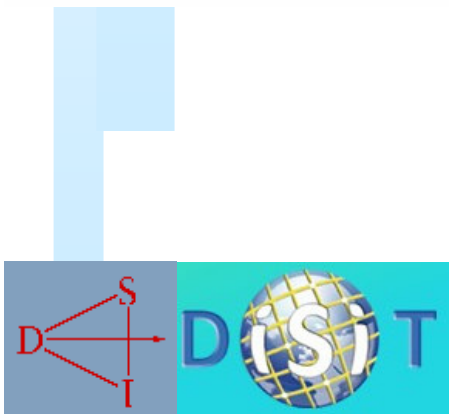
```
<?xml-stylesheet type="text/css" href="stile.css" ?>
<articolo titolo="Titolo dell'articolo">
  <paragrafo titolo="Titolo del primo paragrafo">
    <testo> Blocco di testo del primo paragrafo </testo>
    <immagine file="immagine1.jpg"/>
  </paragrafo>
  <paragrafo titolo="Titolo del secondo paragrafo">
    <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>
```

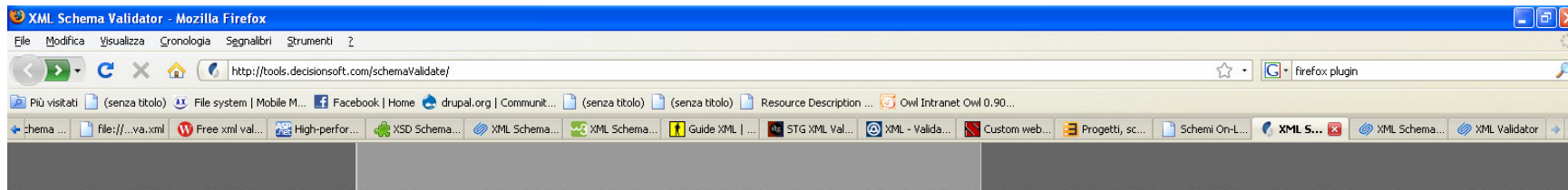
Adesso commento l'immagine

## Ringraziamenti

Ringrazio .....

- | Associare al file xml un css in modo che titoli e testo compaiano come in figura
- | Sintassi:  
`<?xml-stylesheet type="text/css" href="stile.css" ?>`





DecisionSoft

<http://tools.decisionsoft.com/schemaValidate/>

**XML Schema Validator**  
Version: 1.0.1.r120833

This will take a single schema plus an instance document and list any errors found whilst validating the document against the schema.

An attempt will be made to fetch any referenced schema from the Internet if the supplied schema is not sufficient for validation.

XML Schema:

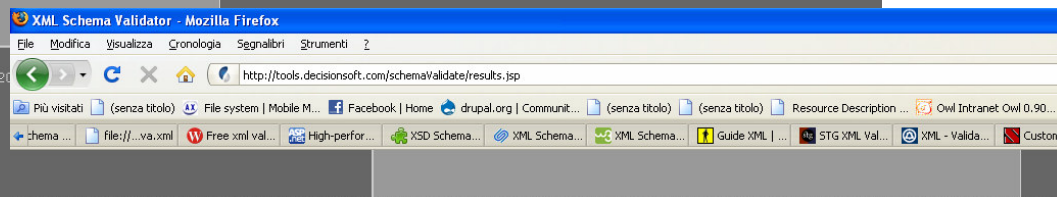
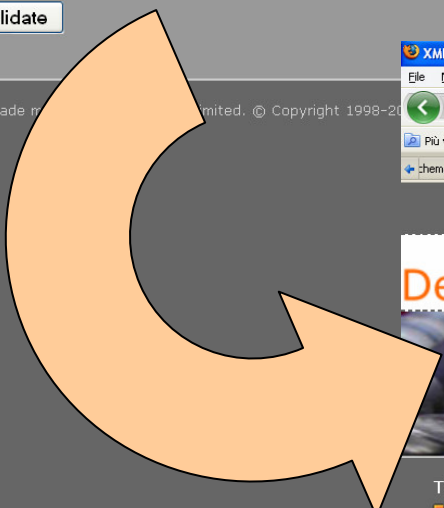
XML Instance:

**TOOLS**

- Home
- xmlpp
- xmldiff
- Schema Validation

**DECISIONSOFT**

- Corporate
- Open Source
- Contact
- Terms of Use



**DecisionSoft**

**XML Schema Validator**  
Version: 1.0.1.r120833

Well Formed: **VALID**  
Schema Validation: **VALID**

**TOOLS**

- Home
- xmlpp
- xmldiff
- Schema Validation

**DECISIONSOFT**

- Corporate
- Open Source
- Contact
- Terms of Use

XBRL Toolkit, X-Index, X-Meta, X-Tract and XML Script are trade marks of DecisionSoft Limited. © Copyright 1998-2009 DecisionSoft Limited

× Trova:      Maiuscole/minuscole

Completato





# Schema



## Passi:

- ♣ Scrivere uno schema (ben formato)
- ♣ Scrivere un documento xml ben formato e riferito allo schema
- ♣ Controllare che il documento xml sia anche valido

## Definire un xml-schema per una serie di **articoli** che abbiano:

- ♣ Un **titolo**
- ♣ Almeno un **paragrafo** contenente del testo

## Quindi:

- ♣ **articolo** è un tipo complesso (complextype) contenente:
  - Una sequenza (<xs:sequence>) di paragrafi (almeno un paragrafo)
  - Un attributo
  - Ogni paragrafo: è un tipo complesso (complextype) contenente: un elemento di tipo stringa (type="xs:string") che si chiama 'testo' (name="testo")
  - Ogni paragrafo ha un titolo (sotto forma di attributo)





## Si ricorda che:

# E

- | I **tipi di dato complesso** si riferiscono ad elementi che possono contenere altri elementi e possono avere attributi
- | Definire un elemento di tipo complesso corrisponde a definire la relativa struttura
- | Lo schema generale per la definizione di un **elemento di tipo complesso** è il seguente:

```
<xs:element name="NOME_ELEMENTO">  
  <xs:complexType>  
    ... Definizione del tipo complesso ...(<xs:all> oppure  
    <xs:sequence> oppure <xs:choice>)  
    ... Definizione degli attributi ... (<xs:attribute name="" type="">)  
  </xs:complexType>  
</xs:element>
```





Quindi:

♣ **articolo** è un tipo complesso (complextype) contenente:

→ Una sequenza di paragrafi (almeno un paragrafo)

→ Un attributo

→ Ogni paragrafo: è un tipo complesso (complextype) contenente:  
un elemento () che si chiama 'testo'

→ Ogni paragrafo ha un titolo (sotto forma di attributo)

**E**

```
| <?xml version="1.0" encoding="UTF-8"?>  
| <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
| elementFormDefault="qualified"> TODO [...]  
| </xs:schema> <?xml version="1.0" encoding="UTF-8"?>  
| <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
| elementFormDefault="qualified"> TODO [...]  
| </xs:schema>
```

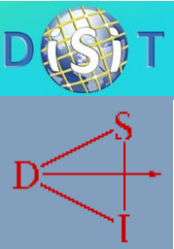
Usare:

♣ <xs:complexType>

♣ <xs:sequence>

→ <xs:element name="" minOccurs="" maxOccurs="" >

♣ <xs:attribute name="">





# E

```
<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
            elementFormDefault="qualified
```

```
<xs:element name="articolo">
```

```
  <xs:complexType >
```

```
    <xs:sequence>
```

```
      <xs:element name="paragrafo"
```

```
        minOccurs="1" maxOccurs="unbounded" >
```

```
        <xs:complexType>
```

```
          <xs:sequence>
```

```
            <xs:element name="testo" maxOccurs="unbounded"  
                        type="xs:string
```

```
          </xs:sequence>
```

```
            <xs:attribute name="titolo" />
```

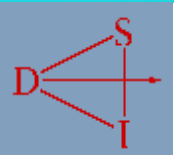
```
          </xs:complexType>
```

```
        </xs:element>
```

```
      </xs:sequence>
```

```
    <xs:attribute name="titolo" type="xs:string"/>
```

```
  </xs:complexType>
```

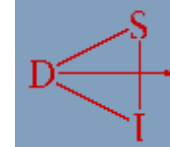




```
<?xml version="1.0" encoding="UTF-8"?>
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" titolo="Come definire un documento xml">
  <paragrafo titolo="primo">
    <testo> Introduzione </testo>
  </paragrafo>
  <paragrafo titolo="secondo">
    <testo> testo del secondo paragrafo </testo>
  </paragrafo>
</articolo>
```

**E**

*Documenti ben formati e validi secondo lo schema definito*



```
<?xml version="1.0" encoding="UTF-8"?>
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
titolo="Come definire un documento xml">
  <paragrafo titolo="primo">
    <testo> Introduzione </testo>
  </paragrafo>
</articolo>
```



## Documenti ben formati MA NON VALIDI

```
<?xml version="1.0" encoding="UTF-8"?><articolo  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
titolo="Come definire un documento xml">
```

```
  <paragrafo titolo="primo">
```

```
    testo
```

```
  </paragrafo>
```

```
</articolo>
```

*Manca l'elemento <testo>*

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
titolo="Come definire un documento xml">
```

```
</articolo>
```

*Non c'è neanche un paragrafo,  
Mentre nel xml-schema c'è indicato  
che deve essere presente almeno  
un paragrafo (<xs:element name="paragrafo"  
minOccurs="1" maxOccurs="unbounded" >)*

**E**





# Elementi di modellazione e programmazione di contenuti digitali

**Corso di Laurea Magistrale in Metodologie  
informatiche per le discipline umanistiche**

*Ing. Michela Paolucci*

Department of Systems and Informatics

University of Florence

Via S. Marta 3, 50139, Firenze, Italy

tel: +39-055-4796523, fax: +39-055-4796363

**Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet**

<http://www.disit.dsi.unifi.it/>

[paolucci@dsi.unifi.it](mailto:paolucci@dsi.unifi.it)

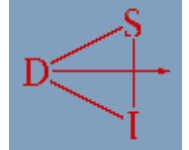
<http://www.dsi.unifi.it/~nesi>, <http://www.dsi.unifi.it/~paolucci>,

<http://www.axmedis.org> <http://mobmed.axmedis.org/>





# Esempio: Associare un file css ai tipi di documenti appena descritti



# E

Definire il file.css

Per collegare il css al file xml usare:

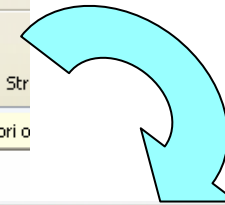
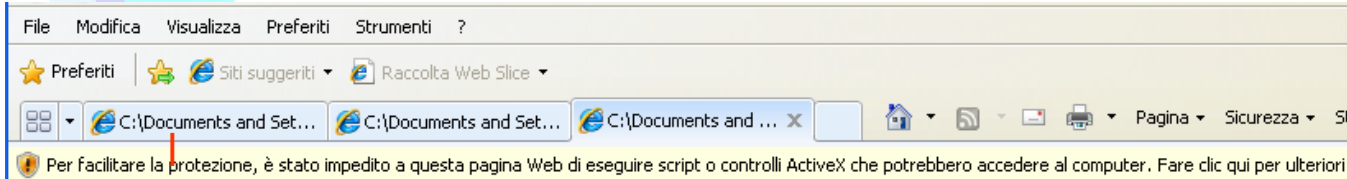
♣ <?xml-stylesheet type="text/css" href=".../file.css" ?>

♣ Sintassi:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?xml-stylesheet type="text/css" href=".../file.css" ?>
```

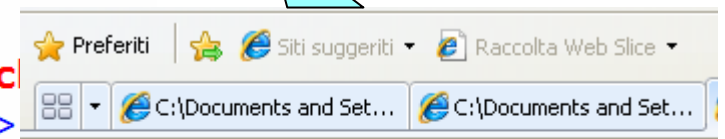
```
<articolo...>
```



```

<?xml version="1.0" encoding="UTF-8" ?>
- <articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" titolo="Come definire un documento xml">
- <paragrafo titolo="primo">
  <testo>Introduzione</testo>
</paragrafo>
- <paragrafo titolo="secondo">
  <testo>testo del secondo paragrafo</testo>
</paragrafo>
</articolo>

```



Introduzione

Con css

testo del secondo paragrafo



## Esempio2 (1)

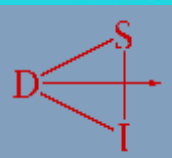
I Partire dallo schema precedente e aggiungere:

- ♣ Ogni paragrafo puo' contenere anche un'immagine
- ♣ Ogni immagine ha associati I seguenti attributi:
  - titolo (xs:string)
  - formato (xs:string) a scelta (definire come restrizione di un tipo semplice) tra jpg, png, gif
  - url (xs:string)
- ♣ Si ricorda che la sintassi per restringere un tipo semplice (in questo caso una stringa) è la seguente:

I

```
<xs:simpleType name=" " >  
  <xs:restriction base="xsd:string">  
    <xs:enumeration value=" " />  
    <xs:enumeration value=" " />  
    <xs:enumeration value=" " />  
  </xs:restriction>  
</xs:simpleType>
```

NOTA: Quindi attenzione a definire correttamente l'attributo 'formato'



```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="articolo">
<xs:complexType >
  <xs:sequence>
    <xs:element name="paragrafo" minOccurs="1" maxOccurs="unbounded" >
      <xs:complexType>
        <xs:sequence>
          <xs:element name="testo" maxOccurs="unbounded" type="xs:string"/>
          <xs:element name="immagine" minOccurs="0" maxOccurs="unbounded" >
            <xs:complexType>
              <xs:attribute name="titolo" type="xs:string" use="required"/>
              <xs:attribute name="formato" use="required">
                <xs:simpleType>
                  <xs:restriction base="xs:string" >
                    <xs:enumeration value="png"/>
                    <xs:enumeration value="jpg"/>
                    <xs:enumeration value="gif"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
              <xs:attribute name="url" type="xs:string" use="optional"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="titolo" type="xs:string"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="titolo" type="xs:string"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="titolo" type="xs:string"/>
</xs:complexType>
</xs:element >
</xs:schema>

```

## Esempio2 (2)

**E**





## **E** Esempio2 (3)

- | Verificare che lo schema sia well-formed
- | Scrivere il file.xml seguendo lo schema
- | Verificare che il file.xml sia well-formed
- | Verificare che il file.xml sia valido secondo lo schema definito
- | Associare un css al file xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    titolo="Come definire un documento xml">  
  <paragrafo titolo="primo">  
    <testo> Introduzione </testo>  
  </paragrafo>  
  <paragrafo titolo="secondo">  
    <testo> testo del secondo paragrafo </testo>  
    <immagine titolo="titolsdkjgh" formato="png" url="http://localhost/css.html"/>  
  </paragrafo>  
</articolo>
```





# Esercizi

- | 1) Fare in modo che sia possibile inserire in un paragrafo in sequenza:
  - ♣ testo
  - ♣ immagine
  - ♣ testo
- | 2) Usare `xs:choice` o `xs:all` al posto di `sequence`
- | 3) Definire altri tipi semplici derivati (anziché restringere una stringa, provare a restringere un intero)
- | 4) Scrivere uno schema usando la definizione modulare degli elementi
- | 5) Derivare tipi semplici per unione o per lista







# E

## Esercizio 1): file.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  titolo="Come definire un documento xml">
  <paragrafo titolo="primo">
    <testo> Introduzione </testo>
  </paragrafo>
  <paragrafo titolo="secondo">
    <testo> testo del secondo paragrafo </testo>
    <testo> introduco la figura </testo>
    <immagine titolo="titolsdkjgh" formato="png" url="http://localhost/css.html"/>
    <testo> spiego la figura del secondo paragrafo </testo>
  </paragrafo>
</articolo>
```





## **E** Esercizio 2)

- | Modificare lo schema precedentemente definito usando **xs:choice** in modo che ogni paragrafo possa contenere o un'immagine o un testo
- | Scrivere almeno un file xml
- | Validare il/i file.xml



## Esercizio 2): schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
```

```
  <xs:element name="articolo">
```

```
<xs:complexType >
```

```
  <xs:sequence>
```

```
    <xs:element name="paragrafo" minOccurs="1" maxOccurs="unbounded" >
```

```
      <xs:complexType>
```

```
        <xs:choice>
```

```
          <xs:element name="testo" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
```

```
          <xs:element name="immagine" minOccurs="0" maxOccurs="unbounded" >
```

```
            <xs:complexType>
```

```
              <xs:attribute name="titolo" type="xs:string" use="required"/>
```

```
              <xs:attribute name="formato" use="required">
```

```
                <xs:simpleType>
```

```
                  <xs:restriction base="xs:string" >
```

```
                    <xs:enumeration value="png"/>
```

```
                    <xs:enumeration value="jpg"/>
```

```
                    <xs:enumeration value="gif"/>
```

```
                  </xs:restriction>
```

```
                </xs:simpleType>
```

```
              </xs:attribute>
```

```
              <xs:attribute name="url" type="xs:string" use="optional"/>
```

```
            </xs:complexType>
```

```
          </xs:element>
```

```
        </xs:choice>
```

```
      <xs:attribute name="titolo" type="xs:string"/>
```

```
    </xs:complexType>
```

```
  </xs:element>
```

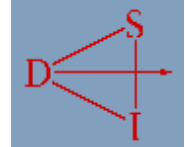
```
</xs:sequence>
```

```
<xs:attribute name="titolo" type="xs:string"/>
```

```
</xs:complexType>
```

```
</xs:element >
```

```
</xs:schema>
```



**E**



# E Esercizio 2): file.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  titolo="Come definire un documento xml">  
  <paragrafo titolo="primo">  
    <testo> Introduzione </testo>  
  </paragrafo>  
  <paragrafo titolo="secondo">  
    <immagine titolo="titolsdkjgh" formato="png" url="http://localhost/css.html"/>  
  </paragrafo>  
</articolo>
```

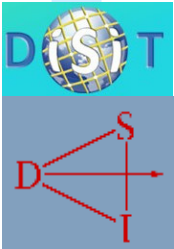
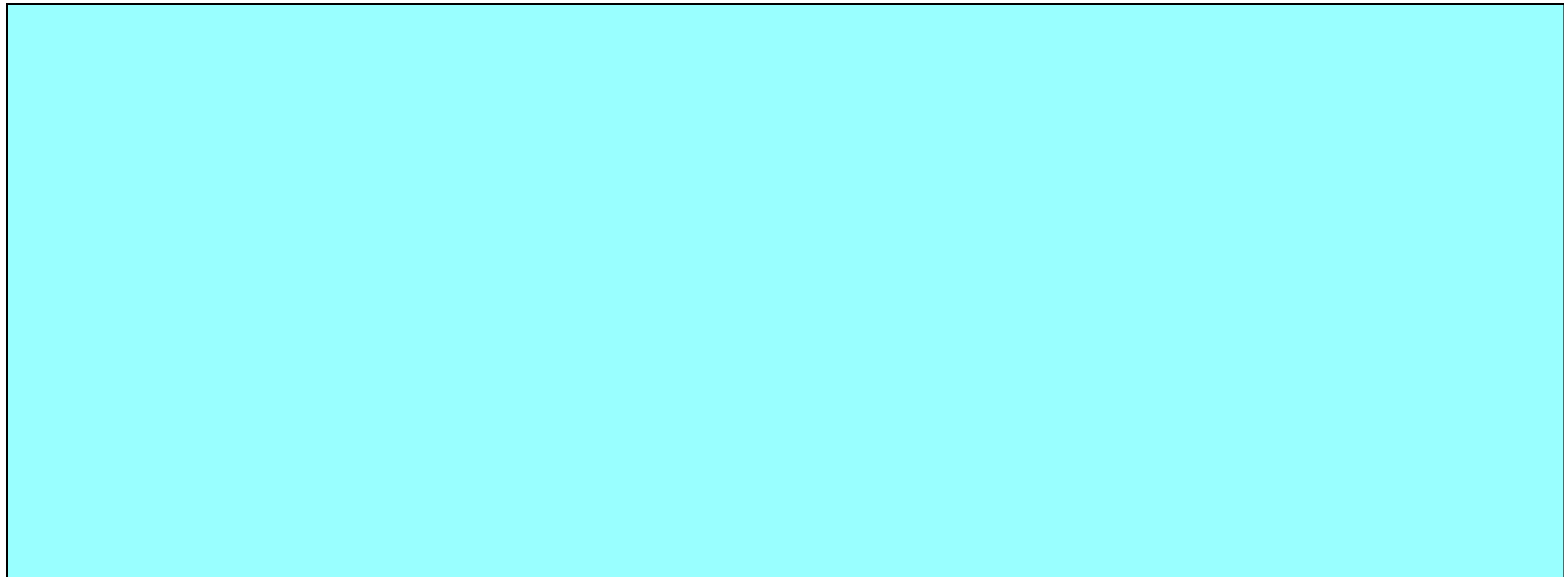




## **E** Esercizio 3)

- | 3) Definire altri tipi semplici derivati (anziché restringere una stringa, provare a restringere un intero)
- | In particolare: aggiungere all'elemento **paragrafo** dell'esercizio precedente l'attributo **numero** (per indicare la numerazione dei paragrafi) in modo tale che il valore assunto sia da 1 a 10 (compresi).
- | Il documento partirà quindi dal paragrafo 1 e potrà avere al massimo 10 paragrafi.

| NOTE:





# E Esercizio 3)

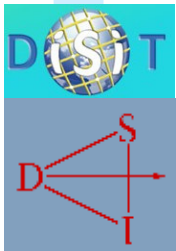
- | 3) Definire altri tipi semplici derivati (anziché restringere una stringa, provare a restringere un intero)
- | In particolare: aggiungere all'elemento **paragrafo** dell'esercizio precedente l'attributo **numero** (per indicare la numerazione dei paragrafi) in modo tale che il valore assunto sia da 1 a 10 (compresi).
- | Il documento partirà quindi dal paragrafo 1 e potrà avere al massimo 10 paragrafi.

## | NOTE:

- ♣ L'attributo **numero** si ottiene come restrizione del tipo semplice `xs:integer`, usando poi i Facet `minInclusive` e `maxInclusive` per determinare l'intervallo di validità

- ♣ Sintassi:

```
<xs:attribute name="    " >  
<xs:simpleType>  
  <xs:restriction base="xs:integer" >  
    <xs:minInclusive value="    "/>  
    <xs:maxInclusive value="    "/>  
  </xs:restriction>  
</xs:simpleType> </xs:attribute>
```





## **E** Esercizio 3)

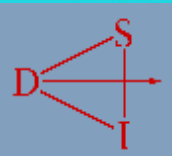
- | 3) Definire altri tipi semplici derivati (anziché restringere una stringa, provare a restringere un intero)
- | In particolare: aggiungere all'elemento **paragrafo** dell'esercizio precedente l'attributo **numero** (per indicare la numerazione dei paragrafi) in modo tale che il valore assunto sia da 1 a 10 (compresi).
- | Il documento partirà quindi dal paragrafo 1 e potrà avere al massimo 10 paragrafi.

### | NOTE:

- ♣ L'attributo **numero** si ottiene come restrizione del tipo semplice `xs:integer`, usando poi i Facet `minInclusive` e `maxInclusive` per determinare l'intervallo di validità

- ♣ Sintassi:

```
<xs:attribute name="    " >  
<xs:simpleType>  
  <xs:restriction base="xs:integer" >  
    <xs:minInclusive value="    "/>  
    <xs:maxInclusive value="    "/>  
  </xs:restriction>  
</xs:simpleType> </xs:attribute>
```



```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="articolo">
<xs:complexType >
  <xs:sequence>
    <xs:element name="paragrafo" minOccurs="1" maxOccurs="unbounded" >
      <xs:complexType>
        <xs:sequence>
          <xs:element name="testo" minOccurs="0" maxOccurs="unbounded" type="xs:string" />
          <xs:element name="immagine" minOccurs="0" maxOccurs="unbounded" >
            <xs:complexType>
              <xs:attribute name="titolo" type="xs:string" use="required" />
              <xs:attribute name="formato" use="required">
                <xs:simpleType>
                  <xs:restriction base="xs:string" >
                    <xs:enumeration value="png" />
                    <xs:enumeration value="jpg" />
                    <xs:enumeration value="gif" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
              <xs:attribute name="url" type="xs:string" use="optional" />
            </xs:complexType>
          </xs:element>
          <xs:element name="testo" minOccurs="0" maxOccurs="unbounded" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="titolo" type="xs:string" />
        <xs:attribute name="numero" >
          <xs:simpleType>
            <xs:restriction base="xs:integer" >
              <xs:minInclusive value="1" />
              <xs:maxInclusive value="10" />
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="titolo" type="xs:string" />
</xs:complexType>
</xs:element >
</xs:schema>

```

## Esercizio 3): schema.xsd



# E

Scrivere un file.xml  
in base a questo  
xml-schema e validarlo



# E

## Esercizio 3): file.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        titolo="Come definire un documento xml">
  <paragrafo titolo="primo" numero="2">
    <testo> Introduzione </testo>
  </paragrafo>
  <paragrafo titolo="secondo" numero="4">
    <testo> testo del secondo paragrafo </testo>
    <testo> introduco la figura </testo>
    <immagine titolo="titolsdkjgh" formato="png" url="http://localhost/css.html"/>
    <testo> spiego la figura del secondo paragrafo </testo>
  </paragrafo>
</articolo>
```





## Esercizio 5)



Partire dall'esercizio precedente e scrivere lo schema usando la definizione modulare degli elementi

Sintassi:

```
<xs:schema ...>
```

```
  <xs:complexType name="TipoComplesso">
```

```
    <xs:sequence> ....</xs:sequence>
```

```
  </xs:complexType>
```

```
  <xs:simpleType name="TipoSemplice">
```

```
    <xs:restriction> ....</xs:restriction>
```

```
  </xs:simpleType>
```

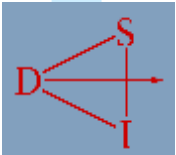
```
  <xs:element name="DocumentoEffettivo">
```

```
    <xs:element name="Complesso"  
type="TipoComplesso"/>
```

```
    <xs:attribute name="Semplice" type="TipoSemplice"/>
```

```
  </xs:element>
```

```
</xs:schema>
```



```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
```

```
<xs:simpleType name="numeroType">
  <xs:restriction base="xs:integer" >
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="10"/>
  </xs:restriction>
</xs:simpleType>

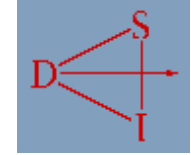
<xs:simpleType name="formatoType">
  <xs:restriction base="xs:string" >
    <xs:enumeration value="png" />
    <xs:enumeration value="jpg" />
    <xs:enumeration value="gif" />
  </xs:restriction>
</xs:simpleType>
```



# Esercizio 4): schema.xsd

*Passo 1: definizione modulare dei simpleType usati (numeroType e formatoType)*

```
<xs:element name="articolo">
  <xs:complexType >
    <xs:sequence>
      <xs:element name="paragrafo" minOccurs="1" maxOccurs="unbounded" >
        <xs:complexType>
          <xs:sequence>
            <xs:element name="testo" minOccurs="0" maxOccurs="unbounded" type="xs:string" />
            <xs:element name="immagine" minOccurs="0" maxOccurs="unbounded" >
              <xs:complexType>
                <xs:attribute name="titolo" type="xs:string" use="required" />
                <xs:attribute name="formato" type="formatoType" use="required" />
                <xs:attribute name="url" type="xs:string" use="optional" />
              </xs:complexType>
            </xs:element>
            <xs:element name="testo" minOccurs="0" maxOccurs="unbounded" type="xs:string" />
          </xs:sequence>
          <xs:attribute name="titolo" type="xs:string" />
          <xs:attribute name="numero" type="numeroType" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="titolo" type="xs:string" />
  </xs:complexType>
</xs:element>
</xs:schema>
```



# E

## Esercizio 4): schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:simpleType name="numeroType">
    <xs:restriction base="xs:integer" >
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="10"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="formatoType">
    <xs:restriction base="xs:string" >
      <xs:enumeration value="png"/>
      <xs:enumeration value="jpg"/>
      <xs:enumeration value="gif"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="immagineType">
    <xs:attribute name="titolo" type="xs:string" use="required"/>
    <xs:attribute name="formato" type="formatoType" use="required"/>
    <xs:attribute name="url" type="xs:string" use="optional"/>
  </xs:complexType>
  <xs:complexType name="ParagrafoType">
    <xs:sequence>
      <xs:element name="testo" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
      <xs:element name="immagine" type="immagineType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="testo" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="titolo" type="xs:string"/>
    <xs:attribute name="numero" type="numeroType"/>
  </xs:complexType>
  <xs:element name="articolo">
    <xs:complexType >
      <xs:sequence>
        <xs:element name="paragrafo" type="ParagrafoType" minOccurs="1" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="titoloArticolo" type="xs:string"/>
    </xs:complexType>
  </xs:element >
</xs:schema>
```

*Passo 2: definizione modulare dei complexType usati (immagineType e paragrafoType)*



Notare che la definizione del documento vero e proprio (articolo) risulta piu' chiara e leggibile



# E Esercizio 4): file.xml

- NOTA: verificare che il file.xml dell'esempio precedente (validato in base allo schema scritto in modo NON modulare) si valido secondo il nuovo schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<articolo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
titoloArticolo="Come definire un documento xml">
  <paragrafo titolo="primo" numero="2">
    <testo> Introduzione </testo>
  </paragrafo>
  <paragrafo titolo="secondo" numero="4">
    <testo> testo del secondo paragrafo </testo>
    <testo> introduco la figura </testo>
    <immagine titolo="titolsdkjgh" formato="png" url="http://localhost/css.html"/>
    <testo> spiego la figura del secondo paragrafo </testo>
  </paragrafo>
</articolo>
```





## **E** 5) Derivare tipi semplici per unione

Partendo dall'esercizio 4) derivare i tipi semplici con l'unione  
Si ricorda la sintassi:

```
<xs:schema ...>
```

```
  <xs:simpleType name="TipoSemplice1">
```

```
    <xs:restriction ...></xs:restriction>
```

```
  </xs:simpleType>
```

```
  <xs:simpleType>
```

```
    <xs:union>
```

```
      <xs:simpleType>
```

```
        <xs:restriction
```

```
          base="TipoSemplice1"/>
```

```
        </xs:simpleType>
```

```
      <xs:simpleType>
```

```
        <xs:restriction base="TipoSemplice2"/>
```

```
      </xs:simpleType>
```

```
    </xs:union>
```

```
  </xs:simpleType>
```





# Elementi di modellazione e programmazione di contenuti digitali

**Corso di Laurea Magistrale in Metodologie  
informatiche per le discipline umanistiche**

*Ing. Michela Paolucci*

Department of Systems and Informatics

University of Florence

Via S. Marta 3, 50139, Firenze, Italy

tel: +39-055-4796523, fax: +39-055-4796363

**Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet**

<http://www.disit.dsi.unifi.it/>

[paolucci@dsi.unifi.it](mailto:paolucci@dsi.unifi.it)

<http://www.dsi.unifi.it/~nesi>, <http://www.dsi.unifi.it/~paolucci>,

<http://www.axmedis.org> <http://mobmed.axmedis.org/>

