XML

- standard W3C
  - Versione 1.0 1998
  - Versione 1.1 2004
- meta-linguaggio di markup
- application profile di SGML

XML: obiettivi

Citando lo standard:

"The design goals for XML are:

- *XML shall be straightforwardly usable over the Internet.*
- *XML shall support a wide variety of applications.*
- *XML shall be compatible with SGML.*
- *It shall be easy to write programs which process XML documents.*
- *The number of optional features in XML is to be kept to the absolute minimum, ideally zero.*

XML: obiettivi

- *XML documents should be human-legible and reasonably clear.*
- *The XML design should be prepared quickly.*
- *The design of XML shall be formal and concise.*
- *XML documents shall be easy to create.*
- *Terseness in XML markup is of minimal importance."*

XML: vantaggi e svantaggi

- Vantaggi:
  - Human-legible
  - Semplice da utilizzare
  - Molto diffuso
  - Diverse tecnologie e standard collegati
  - Presenza numerosi strumenti
- Svantaggi:
  - Molto verboso

XML: esempi di utilizzo

- Pagine web
- Web service
- Scambio di dati tra DB
- File di configurazione
- ecc...

XML

- Documento formato da:
  - Markup
    - Elementi
    - Riferimenti a entità
    - Commenti
    - Sezioni CDATA
    - DTD (Document Type Declarations)
    - PI (Processing Instructions)
  - Contenuto

XML

- struttura gerarchica formata da elementi
- ogni elemento può contenere altri elementi
- gli elementi possono avere attributi
- ogni elemento è definito tramite tag
  - <nome_elemento></nome_elemento>
  - <nome_elemento/>
- case sensitive

XML: prolog

<?xml version="1.0" encoding="ISO-8859-1" >

document type declaration

<?target *instructions*?>

XML: prolog

```
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting>Hello, world!</greeting>


<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```


XML

- tutti i documenti XML devono essere ben formati (well formed)
  - deve contenere un unico elemento di massimo livello (root) che contenga tutti gli altri elementi del documento.
  - ogni elemento deve avere un tag di chiusura o, se vuoti, possono prevedere la forma abbreviata (/>)
  - gli elementi devono essere opportunamente innestati
  - i valori degli attributi devono sempre essere racchiusi tra singoli o doppi apici

XML: sezioni CDATA

- Si utilizzano per per potere inserire del testo contenente dei caratteri che potrebbero essere riconosciuti come markup

  inizia con ![CDATA[

  termina con]]>
- Esempio:
  - ![CDATA[<test><]]>

XML: commenti

- I commenti

  iniziano con <!--

  terminano con -->
- Esempio:
  - <!-- commento -->

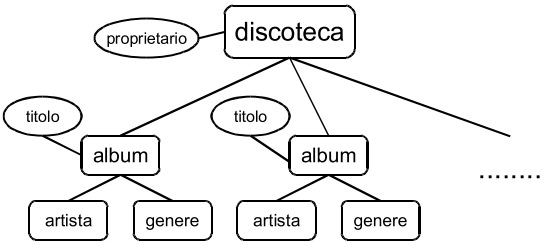
XML: esempio

- Definire il tipo di dato “discoteca”:
  - nome del proprietario
  - album
    - titolo
    - artista
    - genere

XML: esempio

```
<?xml version="1.0" ?>
<discoteca proprietario="Mario Rossi"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="file:discoteca.xsd">
   <album titolo="Symbolic">
      <artista>Death</artista>
      <genere>Death Metal</genere>
   </album>
   <album titolo="Heavy Weather">
      <artista>Weather Report</artista>
      <genere>Fusion</genere>
   </album>
</discoteca>
```


XML



XML: namespace

- la definizione degli elementi può provenire da più risorse
  - problema della collisione dei nomi
  - si legano gli elementi ad un namespace
- si fa riferimento ad un namespace utilizzando l' attributo
  - xmlns:*prefisso oppure*
  - *xmlns*
- lo scope del namespace comincia dal tag in cui è dichiarato al tag di chiusura del medesimo

XML: namespace

<discoteca
xmlns="http://example.com/discoteca">

deve essere un URI

indica che agli elementi (non gli attributi) senza prefisso si applica il namespace http://example.com/discoteca

<disco:discoteca xmlns:disco="
http://example.com/discoteca">

indica che agli elementi col prefisso disco (disco:discoteca, disco:album, ecc...) si applica il namespace http://example.com/discoteca

XML

- Una grammatica definisce uno specifico linguaggio di markup.
- Se un documento rispetta una grammatica si dice valido rispetto a quella grammatica.
- Un documento ben formato può non essere valido rispetto ad una grammatica
- Un documento valido è necessariamente ben formato

XML: grammatica

Come definire una grammatica:

- DTD - Document Type Definition
- XML Schema
- Relax NG
- ...

XML e XML schema: esempio

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
```


XML e XML schema: esempio

```
  <comment>Hurry, my lawn is going wild<!/comment>
  <items>
    <item partNum="872-AA">
      <productName>Lawnmower</productName>
      <quantity>1</quantity>
      <USPrice>148.95</USPrice>
      <comment>Confirm this is electric</comment>
    </item>
    <item partNum="926-AA">
      <productName>Baby Monitor</productName>
      <quantity>1</quantity>
      <USPrice>39.98</USPrice>
      <shipDate>1999-05-21</shipDate>
    </item>
  </items>
</purchaseOrder>
```


XML e XML schema: esempio

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
     Purchase order schema for Example.com.
     Copyright 2000 Example.com. All rights reserved.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>
  <xsd:element name="comment" type="xsd:string"/>
  <xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items"  type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
```


XML e XML schema: esempio

```
  <xsd:complexType name="USAddress">
    <xsd:sequence>
      <xsd:element name="name"   type="xsd:string"/>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city"   type="xsd:string"/>
      <xsd:element name="state"  type="xsd:string"/>
      <xsd:element name="zip"    type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="country" type="xsd:NMTOKEN"
                   fixed="US"/>
  </xsd:complexType>
```


XML e XML schema: esempio

```
  <xsd:complexType name="Items">
    <xsd:sequence>
      <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="productName" type="xsd:string"/>
            <xsd:element name="quantity">
              <xsd:simpleType>
                <xsd:restriction base="xsd:positiveInteger">
                  <xsd:maxExclusive value="100"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="USPrice"  type="xsd:decimal"/>
            <xsd:element ref="comment"   minOccurs="0"/>
            <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
          </xsd:sequence>
          <xsd:attribute name="partNum" type="SKU" use="required"/>
        </xsd:complexType>
```

XML e XML schema: esempio

```
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>

<!-- Stock Keeping Unit, a code for identifying products -->
  <xsd:simpleType name="SKU">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{3}-[A-Z]{2}"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```


XML Schema

- L'elemento root del documento è rappresentato dal tag <xsd:schema>
- <xsd:element> definisce un elemento utilizzabile in un documento XML
  - possiamo indicare il tipo di dato dell'elemento
    - semplice
    - complesso
- <xsd:attribute> definisce un attributo

XML Schema

- Elementi e attributi globali:
  - le loro dichiarazioni sono figlie di <xsd:schema>
  - non fanno riferimento ad altri elementi
  - non contengono gli attributi
    - minOccurs, maxOccurs, use
  - <xsd:element> e <xsd:attribute> fanno riferimento agli altri elementi/attributi globali tramite l'attributo ref

XML Schema: tipo semplice

- non possono contenere altri elementi e non possono avere attributi
- previsti numerosi tipi predefiniti

  xsd:string, xsd:integer, xsd:decimal, xsd:boolean...
- si utilizza il tag xsd:simpleType
- possibilità di definire tipi di dato semplici personalizzati

XML Schema: tipo semplice

- Creazione di un nuovo tipo:
  - Estensione di un altro tipo semplice
  - Unione di più tipi
  - Lista

XML Schema: tipo complesso

- possono contenere altri elementi e possono avere attributi
- può estendere
  - un altro tipo semplice
  - un altro tipo complesso
- si utilizza il tag xsd:complexType

XML Schema: tipo complesso

- Per definire la sequenza di elementi che possono stare al suo interno
  - <xsd:sequence>

    sequenza ordinata di sottoelementi
  - <xsd:choice>

    elenco di sottoelementi alternativi
  - <xsd:all>

    sequenza non ordinata di sottoelementi

XML Schema: tipo complesso

- Attributi:
  - minOccurs e maxOccurs

    numero occorrenza minima con valori interi oppure con la parola chiave unbounded
  - required

    elemento richiesto
  - fixed

    valore fisso (esclude la presenza di default)
  - default

    valore di default (esclude la presenza di fixed)

XML Schema: attributo

- <xsd:attribute> permette la definizione degli attributi:

  <xsd:attribute name="titolo" type="xs:string" use="required" />
  - l'attributo use consente di specificare alcune caratteristiche come la presenza obbligatoria (required) o un valore predefinito (default) in combinazione con l'attributo default.

  <xsd:attribute name="titolo" type="xs:string" use="default" default="test" />
  - se non si specifica esplicitamente l'obbligatorietà dell'attributo, esso è considerato opzionale

XML Schema: gruppi di attributi

```
<xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="productName" type="xsd:string"/>
      <xsd:element name="quantity">
        <xsd:simpleType>
          <xsd:restriction base="xsd:positiveInteger">
            <xsd:maxExclusive value="100"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="USPrice"  type="xsd:decimal"/>
      <xsd:element ref="comment"   minOccurs="0"/>
      <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ItemDelivery"/>
  </xsd:complexType>
</xsd:element>
```


XML Schema: gruppi di attributi

```
<xsd:attributeGroup id="ItemDelivery">
  <xsd:attribute name="partNum"  type="SKU" use="required"/>
  <xsd:attribute name="weightKg" type="xsd:decimal"/>
  <xsd:attribute name="shipBy">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="air"/>
        <xsd:enumeration value="land"/>
        <xsd:enumeration value="any"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:attributeGroup>
```


XML Schema: gruppi di elementi

```
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:group   ref="shipAndBill"/>
      <xsd:element name="singleUSAddress" type="USAddress"/>
    </xsd:choice>
    <xsd:element ref="comment" minOccurs="0"/>
    <xsd:element name="items"  type="Items"/>
  </xsd:sequence>
  <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>

<xsd:group id="shipAndBill">
  <xsd:sequence>
    <xsd:element name="shipTo" type="USAddress"/>
    <xsd:element name="billTo" type="USAddress"/>
  </xsd:sequence>
</xsd:group>
```

XML Schema: anyType

- Permette di inserire in un elemento qualsiasi cosa, anche altri elementi
- Tipo di default:

  <xsd:element name="anything" type="xsd:anyType"/>

  equivale a

  <xsd:element name="anything"/>

XML Schema: substitution group

- permette ad un elemento di essere sostituito da altri elementi
  - l'elemento sostituibile si chiama head
  - gli elementi di un substitution group devono avere lo stesso tipo dell'head o un tipo derivato
  - il gruppo di elementi che lo possono sostituire si chiama substitution group

XML Schema: substitution group

```
...
<complexType name="Vehicle" abstract="true"/>

<element name="transport" type="Vehicle"/>
...
```

```
....
<transport xmlns="http://cars.example.com/schema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="Car"/>
....
```


XML Schema: abstract

- Se dichiarato abstract un elemento non può apparire come istanza
  - deve essere per forza utilizzato come head
  - <xsd:element name="nomeElemento" type="string" abstract="true"/>
- Se un tipo di un elemento è dichiarato abstract l'istanza dell'elemento deve utilizzare xsi:type per dichiararne il tipo

XML Schema: substitution group

```
<element name="shipComment" type="string"
        substitutionGroup="ipo:comment"/>
<element name="customerComment" type="string"
        substitutionGroup="ipo:comment"/>
```

```
....
<items>
  <item partNum="833-AA">
    <productName>Lapis necklace</productName>
    <quantity>1</quantity>
    <USPrice>99.95</USPrice>
    <ipo:shipComment>Use gold wrap if possible</ipo:shipComment>
    <ipo:customerComment>Want this for the holidays!</ipo:customerComment>
    <shipDate>1999-12-05</shipDate>
  </item>
</items>
....
```


XML Schema: namespace

- TargetNamespace
  - indica il namespace a cui appartengono le definizione degli elementi e degli attributi
- Assenza di TargetNamespace
  - gli elementi e gli attributi non sono qualificati da un namespace

XML Schema: namespace

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:po="http://www.example.com/PO1"
        targetNamespace="http://www.example.com/PO1"
        elementFormDefault="unqualified"
        attributeFormDefault="unqualified">
  <element name="purchaseOrder" type="po:PurchaseOrderType"/>
  <element name="comment"       type="string"/>
  <complexType name="PurchaseOrderType">
    <sequence>
      <element name="shipTo"    type="po:USAddress"/>
      <element name="billTo"    type="po:USAddress"/>
      <element ref="po:comment" minOccurs="0"/>
      <!-- etc. -->
    </sequence>
  </complexType>
</schema>
```


XML Schema: namespace

```
<?xml version="1.0"?>
<apo:purchaseOrder xmlns:apo="http://www.example.com/PO1"
                   orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <!-- etc. -->
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <!-- etc. -->
  </billTo>
  <apo:comment>Hurry, my lawn is going wild<!/apo:comment>
  <!-- etc. -->
</apo:purchaseOrder>
```


XML Schema: namespace

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:po="http://www.example.com/PO1"
        targetNamespace="http://www.example.com/PO1"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

  <element name="purchaseOrder" type="po:PurchaseOrderType"/>
  <element name="comment"       type="string"/>

  <complexType name="PurchaseOrderType">
    <!-- etc. -->
  </complexType>

  <!-- etc. -->

</schema>
```


XML Schema: namespace

```
<?xml version="1.0"?>
<apo:purchaseOrder xmlns:apo="http://www.example.com/PO1"
                   orderDate="1999-10-20">
  <apo:shipTo country="US">
    <apo:name>Alice Smith</apo:name>
    <apo:street>123 Maple Street</apo:street>
    <!-- etc. -->
  </apo:shipTo>
  <apo:billTo country="US">
    <apo:name>Robert Smith</apo:name>
    <apo:street>8 Oak Avenue</apo:street>
    <!-- etc. -->
  </apo:billTo>
  <apo:comment>Hurry, my lawn is going wild<!/apo:comment>
  <!-- etc. -->
</apo:purchaseOrder>
```


XML Schema: namespace

```
<?xml version="1.0"?>
<purchaseOrder xmlns="http://www.example.com/PO1"
               orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <!-- etc. -->
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <!-- etc. -->
  </billTo>
  <comment>Hurry, my lawn is going wild<!/comment>
  <!-- etc. -->
</purchaseOrder>
```


XLink

- XML Linking Language definisce il supporto ipertestuale nei documenti XML
- namespace: http://www.w3.org/1999/xlink
- due tipologie di link
  - simple link
    - simile ad href di HTML
  - extended link
    - più potente, permette la connessione di più risorse con archi multipli

Riferimenti

- http://www.w3.org/TR/xml/
- http://www.w3.org/XML/Schema
- http://www.w3.org/TR/xlink/