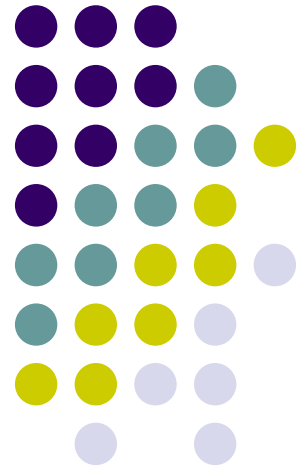
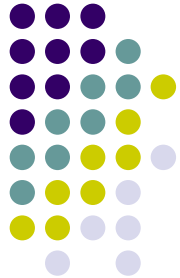
# Introduzione a XML

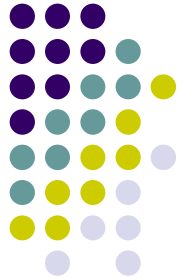


# Perché parliamo di XML



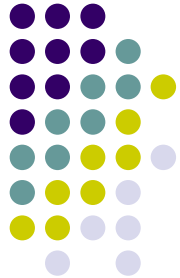
- Xml sta diventando lo standard de-facto per lo scambio di dati sul web e non solo.
- Sta alla base di quello che viene definito il semantic-web.
- E’ un linguaggio facile da imparare ed utilizzare, e nello stesso tempo è anche incredibilmente potente, permettendo di creare documenti della complessità desiderata.
- Non da ultimo, le più grandi società software, stanno investendo molto in XML, che, ad esempio, è una delle parti integranti dell’architettura .NET di Microsoft.

# Introduzione a Xml: cos'è?



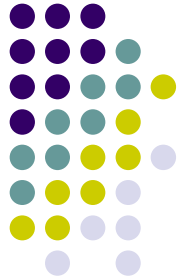
- Xml: eXtensible Markup Language
- Linguaggio di markup: permette di evidenziare il contenuto di un documento, all'interno di marcatori descrittivi, che ne definiscono gli attributi, ad es. linguaggio html.
- Estensibilità: in realtà Xml è un metalinguaggio di markup, cioè viene utilizzato per creare linguaggi di markup personalizzati.
- NON è un linguaggio di programmazione.

# Alcune caratteristiche di XML



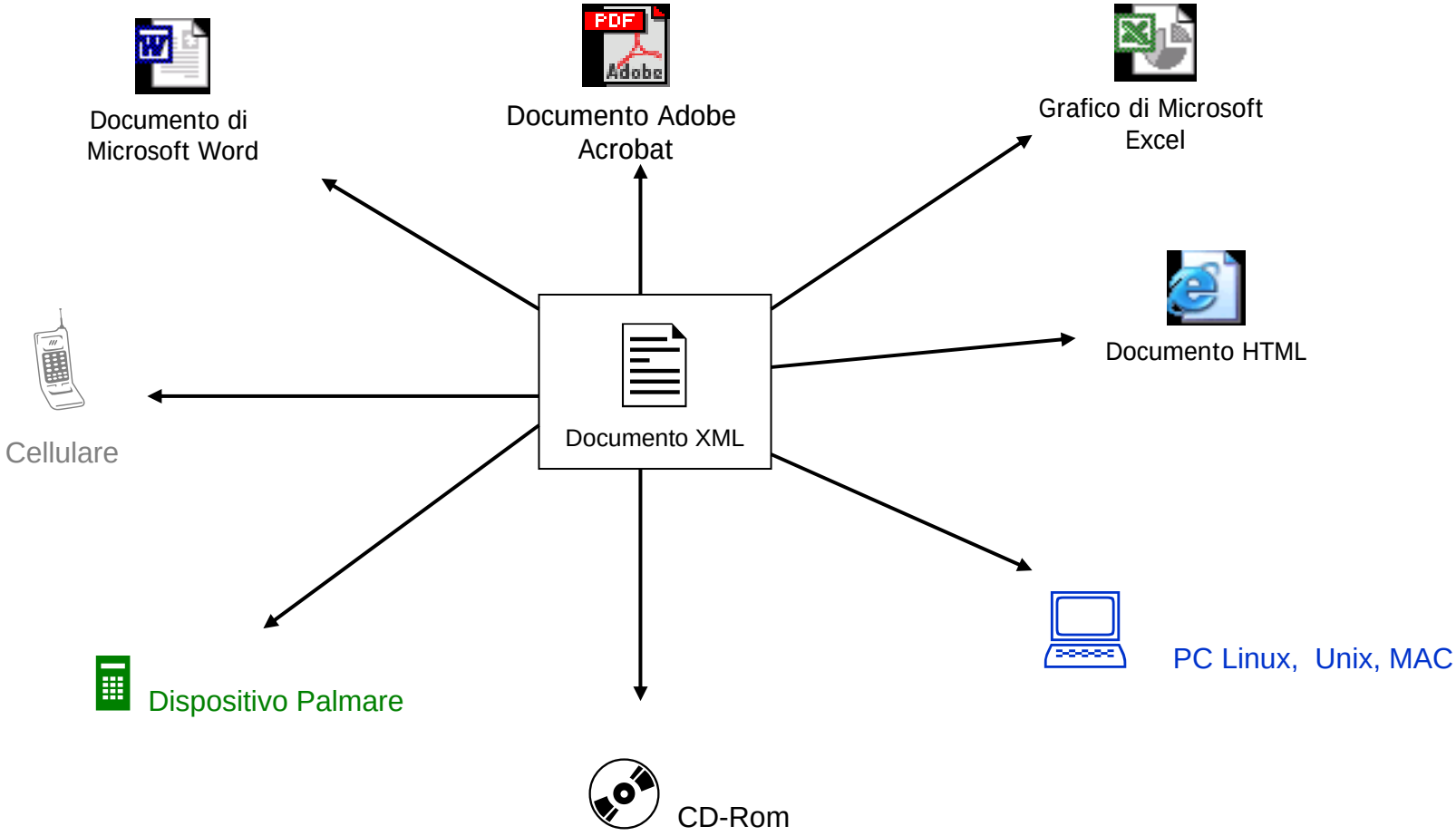
- Permette di organizzare a proprio piacimento le informazioni, consentendo di inviarle a chiunque in modo libero e gratuito.
- Standard aperto, no royalty, no brevetti, no copyright o segreti  industriali.
- Salvataggio dei dati: se avete dei file in un formato proprietario quale Visicalc, Lotus Jazz, magari su floppy da 5”¼, la possibilità di recuperarli è piuttosto bassa. Il formato di solo testo resiste abbastanza bene alle alterazioni magnetiche.
- Permette di condividere sul Web documenti con strutture anche molto complicate che contengono dati particolarmente complessi.

# Alcune caratteristiche di XML

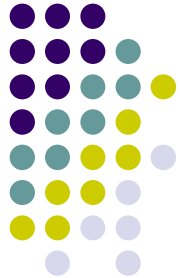


- L’importanza di  uno standard di comunicazione tra applicazioni, per evitare che i dati possano essere letti e “capiti” solo da pochissimi programmi.
- La gran parte dei flussi di informazioni viaggia attraverso sistemi proprietari e costosi, ad es. Oracle, SQL, DB2.
- Ciò che invece, possiamo ottenere con XML è questo:

# XML

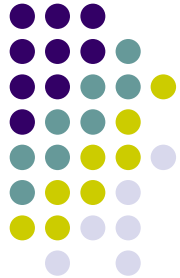

Documento di Microsoft Word
Documento Adobe Acrobat
Grafico di Microsoft Excel
Documento HTML
Documento XML
Cellulare
Dispositivo Palmare
PC Linux, Unix, MAC
CD-Rom

# Le “breve storia” non manca mai…



- Creato dal W3C si veda http://www.w3.org/XML, la prima edizione della specifica XML 1.0 risale al febbraio 1998, la seconda edizione è di ottobre 2000, mentre ad oggi è in fase “candiate recommendation” la versione 1.1 di XML.
- XML, deriva da SGML, che è un linguaggio di markup molto potente ma molto complicato.
- Anche HTML deriva da SGML, semplificandolo all’estremo, ma l’html nella sua semplicità si limita a formattare i dati.
- XML nasce dall’idea di creare un linguaggio che unisca potenza di SGML e semplicità di HTML.

# Alcuni punti chiave considerati nella progettazione dal W3C.



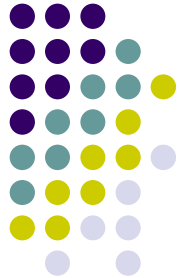
- Supporto di una vastissima gamma di applicazioni (non solo Web).
- I documenti XML devono rimanere leggibili da parte dell'uomo, e ragionevolmente chiari.
- Minimizzare le caratteristiche opzionali.
- Facile da redarre, ossia creabile anche con semplici editor di testo.
- Formale e conciso: piccoli insiemi di regole ma assolutamente precise.

# Struttura XML

```
<?xml version="1.0"  ?>
<Media>
    <CDs>
        <CD quantity=“10”>
            <Title>Glory and consequence</Title>
            <Artist>Ben Harper</Artists>
            <Album>The wheel to live</Album>
            <Album>Hello</Album>

        </CD>
        <CD quantity=“6”>
            <Title>Bigmouth strikes again</Title>
            <Artist>The Smiths</Artists>
            <Album>Meat is murder</Album>
        </CD>
    </CDs>
    <Books>
            <Book quantity=“3”>
                <Title>Caves bird</Title>
                <Author>Ted Hughes</Author>
            </Book>
    </Books>
</Media>
```

# Regole di sintassi e strutturazione



- Quando un documento XML rispetta le regole strutturali definite dal W3C, si dice che è un documento “*ben formato”.* Gli interpreti XML rifiutano e ignorano i documenti mal formati. Ciò invece non accade in Html.
- Tutti i documenti XML devono avere una tag radice.
- Tutti i tag devono necessariamente essere chiusi.
- L’ annidamento dei tag deve rispettare la gerarchia di apertura. <tag1><tag2>Getafix</tag1></tag2> NON e’ consentito in XML (in Html si!)
- Il valore degli attributi DEVE sempre essere racchiuso tra virgolette.
- I nomi degli elementi e degli attributi devono iniziare con una lettera, un underscore _ o un segno di due punti (:), poi possono seguire un numero qualsiasi di lettere, numeri, due punti, punti singoli, trattini e underscore. Attenzione perché XML è “case sensitive”.

# Dtd: Document Type Definition

- Poiché abbiamo detto che XML ci permette di creare linguaggi di markup personalizzati, dobbiamo anche definire un dizionario per la leggibilità del documento. DTD è uno dei modi per farlo.
- E' un documento di testo che definisce tutte le parti di un documento XML: elementi, attributi, elenchi, entità, proprietà e relazioni.
- Le DTD possono far parte del documento XML stesso, oppu… …ato.
- E' pi… …o essere…
- Chiama… …ad una DTD esterna:
  <!DOCTYPE elemento_radice SYSTEM "nome_dtd">

Direttiva che inizia la riga in cui si specifica dtd

Nome dell'elemento radice

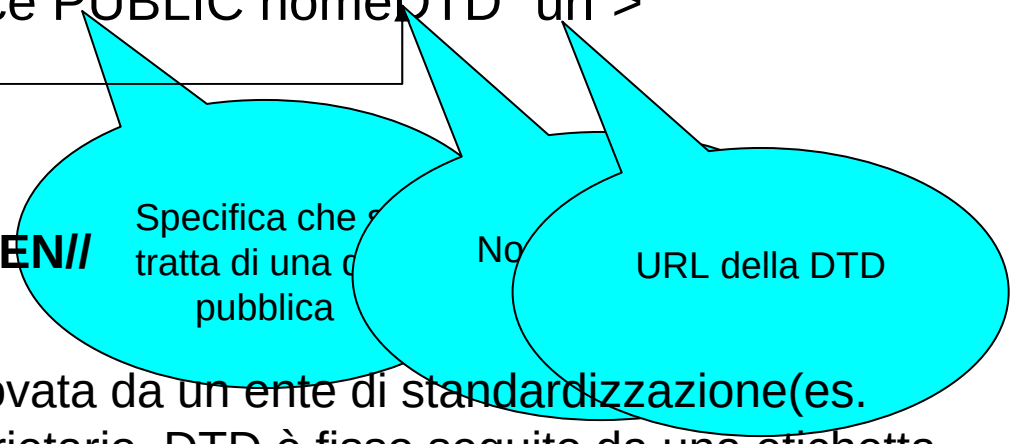
…ca che …na …loi …n un esterno

Nome del file di testo che contiene la dtd

# Dtd: Document Type Definition

- Nella maggior parte dei casi si utilizzano DTD pubbliche, ossia dtd già create e messe a disposizione da qualcuno.
- Chiamata ad una DTD esterna pubblica:

  <!DOCTYPE elemento_radice PUBLIC nomeDTD “url”>




Il meno indica una DTD non approvata da un ente di standardizzazione(es. ISO). Microsoft è il nome del proprietario, DTD è fisso seguito da una etichetta della DTD (una breve descrizione), EN è la lingua.

# Dtd: definizione degli elementi

- La parola chiave piu' importante all'interno di una DTD è "ELEMENT", essa ci permette di definire gli elementi (tag) del nostro linguaggio di markup.
- <!ELEMENT media (#PCDA
  definisce l'elemento "media
  contenere (#PCDATA signi
  qualsiasi altra cosa che nor
- L'annidamento degli eleme
- <!ELEMENT Media (Books)
- <!ELEMENT Books (Book)>
- <!ELEMENT Book (Title,Au
- <!ELEMENT Title (#PCDAT
- <!ELEMENT Author (#PCDATA)>

Oltre definire la struttura degli elementi, è possibile indicare anche alcuni comportamenti quali:

- Numero di volte che un elemento può comparire: N. volte, da 0 a 1, da 1 a qualsiasi, da 0 a qualsiasi

- Indicare se la presenza di un elemento ne esclude un'altra

# Dtd: Separatori

Separano specifiche determinando l'ordine o l'obbligatorietà:

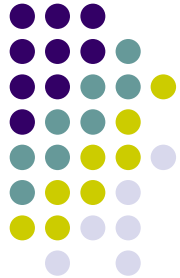
- ‘,’(virgola): richiede la presenza di entrambe le specifiche nell'ordine precisato.

  Es.: (a , b): ci devono essere sia a che b, e prima ci deve essere a e poi b.

- ‘|’(barra verticale): ammette la presenza di una sola delle due specifiche.

  Es.: (a | b): ci può essere o a, oppure b, ma solo uno di essi.

# Operatori di ripetizione (1)



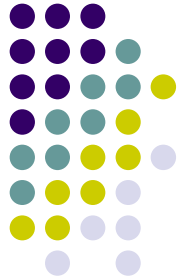
Permettono di specificare se un gruppo può comparire esattamente una volta, almeno una volta, oppure zero o piùvolte.

Niente: la specifica precedente deve comparire esattamente una volta.

- Es.: c, (a, b): a e b devono comparire in quest'ordine esattamente una volta. È lecito solo: cab. Si dice che è una specifica richiesta e non ripetibile.
- ? (punto interrogativo): la specifica precedente può e può non comparire, ma solo una volta.

Es.: c, (a , b)?: a e b possono comparire una volta, ma possono non comparire. Sono lecite: c, cab. Si dice che è una specifica facoltativa e non ripetibile.

# Operatori di ripetizione (2)



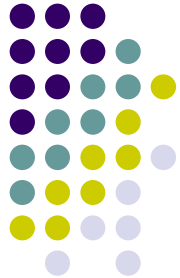
- + (più): la specifica precedente deve comparire almeno una volta. Es.: c, (a , b)+: a e b devono comparire almeno una volta, ma possono comparire anche più di una. Sono lecite: cab, cabab, cababababab, ma non c, ca, cb, cba, cababa. Si dice che èuna specifica richiesta e ripetibile.
- * (asterisco): la specifica precedente deve comparire zero o più volte. Es.: c, (a, b)*: a e b possono comparire o no, a scelta e in numero libero. Sono lecite: c, cab, cabab, cababababab, ma non ca, cb, cba, cababa. Si dice che è una specifica facoltativa e ripetibile.

# Dtd: definizione degli attributi

- La definizione degli attributi avviene in questo modo:
- <!ATTLIST nomeElemento nomeAttributo type default>
- Type indica il tipo di attributo, può essere:
  - PCDATA = qualunque cosa sia digi
  - Enumerazione = un insieme di poss
  - ID = valore univoco all'intern
  - IDREF = si riferisce al codice con
  - NMTOKEN = specifica che il valore dell'attributo segue le regole dei nomi XML
  - ENTITY = permette di includere come attributo oggetti esterni

Default permette di inserire un valore predefinito, oppure può essere:
- #REQUIRED (obbligatorio)
- #IMPLIED (facoltativo)
- #FIXED (fisso)

# Dtd: entità e notazioni



- Le entità servono per contenere dati: può trattarsi di testo, immagini o qualunque altra forma di dati binari non testuali.
- L'esempio più semplice:

**<!ENTITY azalea "Azalea, biblioteca digitale in oncologia per pazienti, familiari, cittadini.">**
**Nel documento XML si richiama così: <desc>&azalea;</desc>**

- L'esempio "meno" semplice, entità non interpretata esterna

**<!NOTATION gif SYSTEM "image/gif">**
**<!ENTITY foto SYSTEM "immagine.gif" NDATA gif"**

**A questo punto per includere l'entità nel documento XML è necessario utilizzare il tipo di attributo ENTITY:**
**<!ATTLIST utente fotografia ENTITITY #REQUIRED"**

**Il documento XML avrà quindi una riga di questo genere:**
**<utente fotografia="foto" </utente>**