Note introduttive su XML e tecnologie correlate

Prof. Andrea Omicini

II Facoltà di Ingegneria, Alma Mater Studiorum,
Università di Bologna a Cesena

andrea.omicini@unibo.it

Cosa dobbiamo imparare/capire?

- Perché / quando XML
  - Well-formedness / Validazione
  - Trasformazione
  - …
- Tecnologie correlate
  - DTD
  - XML Schema
  - XSL
    - XSLT, XPath, XSL-FO
  - WSDL
  - RDF
- Rapporto con le altre tecnologie

HTML vs. XML

- HTML
  - Orientato alla presentazione
  - Non rappresenta la struttura e la semantica dei dati
- XML
  - Orientato ai dati
  - Consente di rappresentare struttura e semantica
  - Può essere validato tramite grammatiche

XML

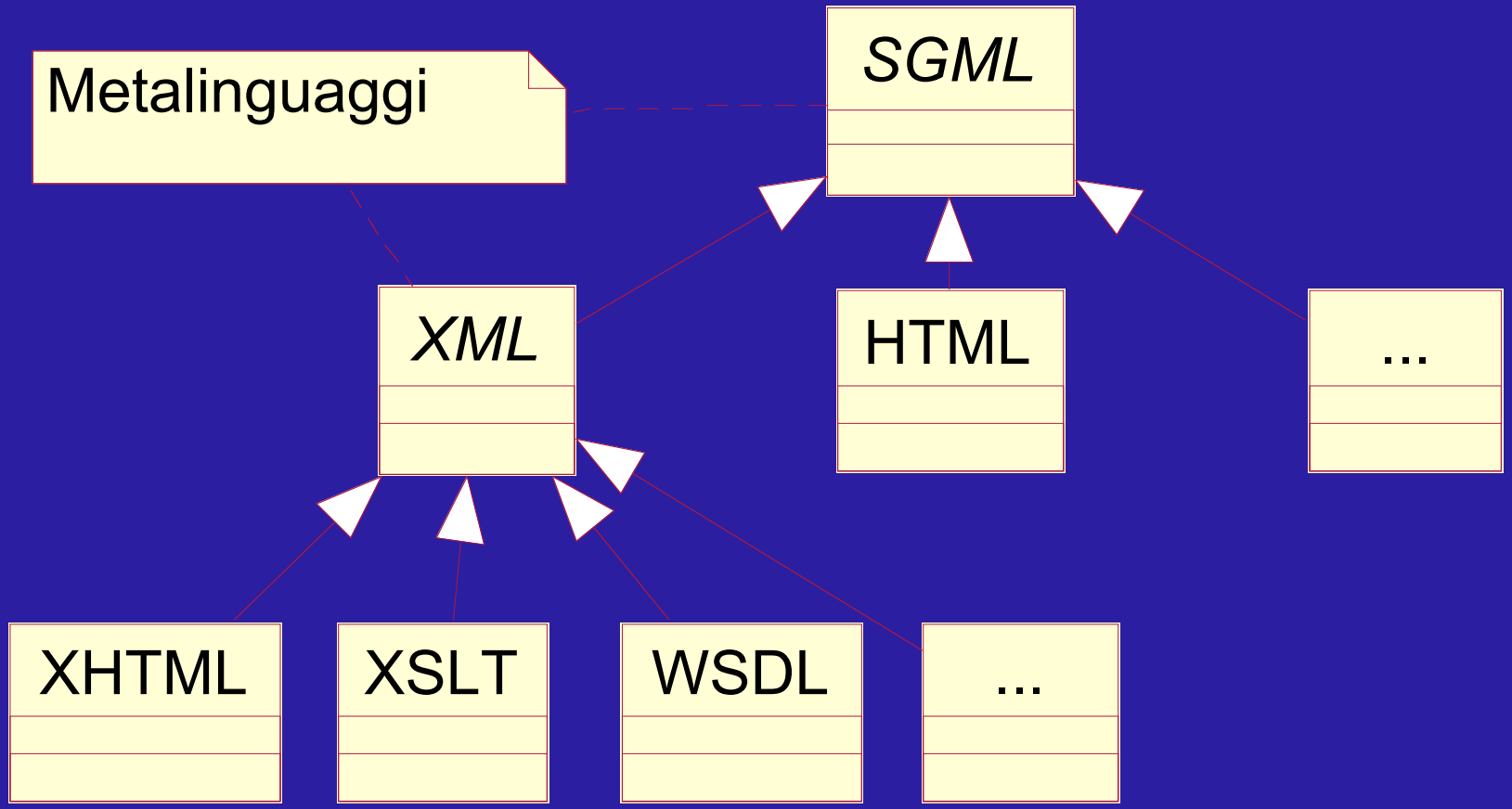
- eXtensible Markup Language
- Metalinguaggio di markup per la descrizione di dati strutturati
  - aperto
  - testuale
  - informazioni strutturali e semantiche
- Sottoinsieme di SGML (Standard Generalized Markup Language)
  - SGML è troppo complesso…
- Standard W3C
  - http://www.w3.org/XML/

Perché un Markup Language?

- Markup
  - sistema di codifica incorporato nel documento, che specifica le proprietà del documento e dell'informazione in esso contenuta
    - per esempio, informazione di formattazione
    - ma può essere in generale informazione strutturale
      - struttura della conoscenza
- Marcatori (marks)
  - tag usati per qualificare segmenti di testo
  - esempio: tag HTML / XHTML
- Esempio XML

```
<studente>
  <nomestudente>
    <nome>Carlo</nome>
    <cognome>Nervo</cognome>
  </nomestudente>
  <matricola>0000145678</matricola>
  <corso>2036</corso>
</studente>
```

Tassonomia XML



Nota: troppi linguaggi?

- I domini applicativi divengono sempre più
  - numerosi
  - complessi
  - specifici
- Linguaggi speciali / specializzati come strumento dell'ingegnere
  - per rappresentare, denotare e esprimere comportamenti e computazioni
- L'ingegnere dei sistemi informatici / ICT dovrà non solo usare i linguaggi artificiali, ma pure
  - conoscere i paradigmi e i modelli computazionali
  - scegliere i paradigmi e i linguaggi
  - definire e costruire nuovi linguaggi
- Corsi di "Linguaggi e modelli computazionali" e "Ingegneria del SW" alla laurea magistrale in Sistemi ICT

XML: le basi

- Un metalinguaggio basato su testo
- Usa una notazione basata su tag (etichette)

  <tag> ... </tag> ogni tag aperto deve essere chiuso
  <tagVuoto /> abbrevia <tagVuoto></tagVuoto>

- Poche regole base
  - ogni tag aperto deve essere chiuso, e definisce un elemento
  - gli elementi ammettono attributi, e possono contenere elementi
  - va rispettata la corretta nidificazione dei tag aperti/chiusi
  - un solo tag radice racchiude tutti gli altri
  - all'inizio del documento va posto un tag particolare
- Un documento XML che rispetti queste regole si dice **ben formato** (well formed)

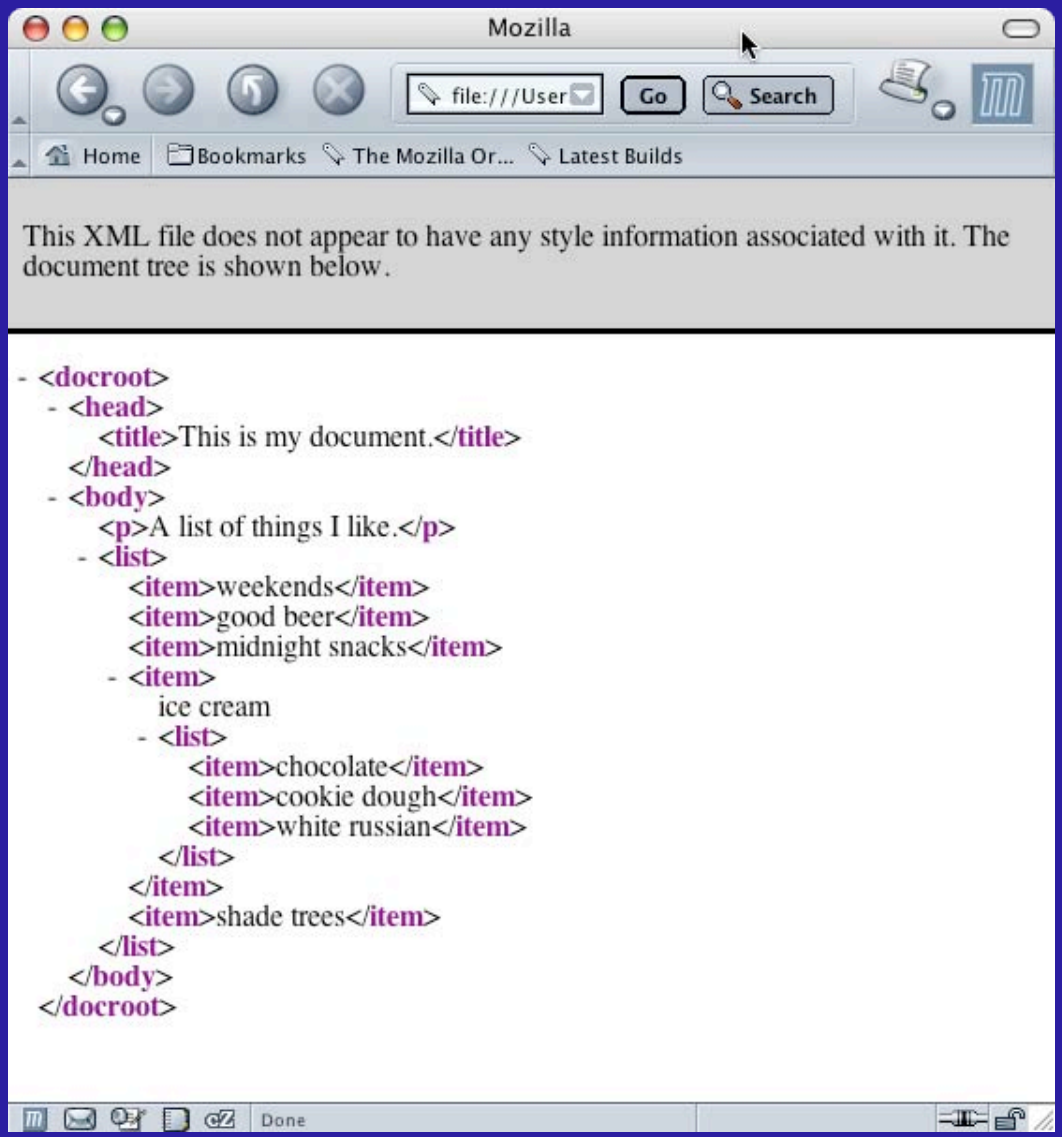
L'albero XML

- Siccome un documento XML **ben formato**
  - ha un solo elemento radice
  - e ogni elemento può contenere un numero qualunque di altri elementi
- prendendo
  - il tag radice come nodo radice dell'albero
  - gli elementi che contengono n>0 elementi come nodi dell'albero con n figli
  - gli elementi che non contengono altri elementi come foglie dell'albero
- possiamo costruire l'albero del documento XML
  - e usarlo per la rappresentazione del documento e per la estrazione dell'informazione

Esempio: il testo

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

<docroot>

  <head>
    <title>This is my document.</title>
  </head>

  <body>

    <p>A list of things I like.</p>

  <list>
    <item>weekends</item>
    <item>good beer</item>
    <item>midnight snacks</item>
    <item>ice cream
    <list>
      <item>chocolate</item>
    <item>cookie dough</item>
    <item>white russian</item>
    </list>
    </item>
    <item>shade trees</item>
  </list>

  </body>
</docroot>
```

Esempio: la vista ad albero



XML: X per estendibilità

- Idea fondamentale di XML
  - meta-linguaggio semplice per umani e automi
  - per costruire documenti elettronici
  - che consente di definire linguaggi di markup ad hoc
- Quindi
  - XML è molto "libero" in generale
  - poi può essere "esteso"
    - in realtà, specializzato
  - a definire linguaggi specifici
- Non c'è un insieme di tag predefinito e fisso, come in HTML
  - vanno definiti ad hoc: come?
  - li dobbiamo / possiamo definire noi?

XML: validazione

- L'insieme dei tag leciti in un dato documento lo decidiamo noi

→ necessità di una **grammatica** per specificare il "nostro linguaggio"

  - primo standard definito: **DTD** (Document Type Definition) MA: la sintassi DTD non segue le regole XML
  - nuovo standard più completo (e prolisso) : **XML Schema** (segue la sintassi XML)

- Un documento XML che rispetti le regole sintattiche stabilite da un certo DTD o XML Schema si dice
  - **valido**
  - rispetto a tale DTD o XML Schema
- Lo strumento che effettua tale validazione si chiama
  - parser XML
  - e lavora sull'albero XML del documento

XML: applicazioni

- XML in sè è "piccolo" e semplice
  - sono i linguaggi definiti tramite XML che sono tanti e complessi
- Applicazioni XML
  - sono specifici sottolinguaggi XML
    - definiti tramite DTD o Schema
  - possono essere standard o custom
- Le più importanti applicazioni XML standard sono W3C
  - Per esempio
    - XSLT
    - XML Schema
    - XHTML

XML: grammatica

- Tre classi di elementi
- Generati rispettivamente da
  - **direttive** (processing directives)
    - dicono come processare il testo
    - formato <?xml …?>
    - esempio: prologo XML (all'inizio di ogni documento)

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

  - **asserzioni** (declarative statements)
    - dichiarazioni / definizioni di entità usate nel documento
    - formato <! istruzioni >
    - esempio: DOCTYPE in XHTML

```
<! DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

  - **tag di marcatura** (markup tags)
    - il contenuto, la struttura e lo stile del documento
    - il resto è contorno, questo è l'essenziale…
    - tag, elementi e attributi: come visto in XHTML

XML e CSS

- Ogni file XML può essere preparato per la visualizzazione diretta da browser
- Direttiva di stile

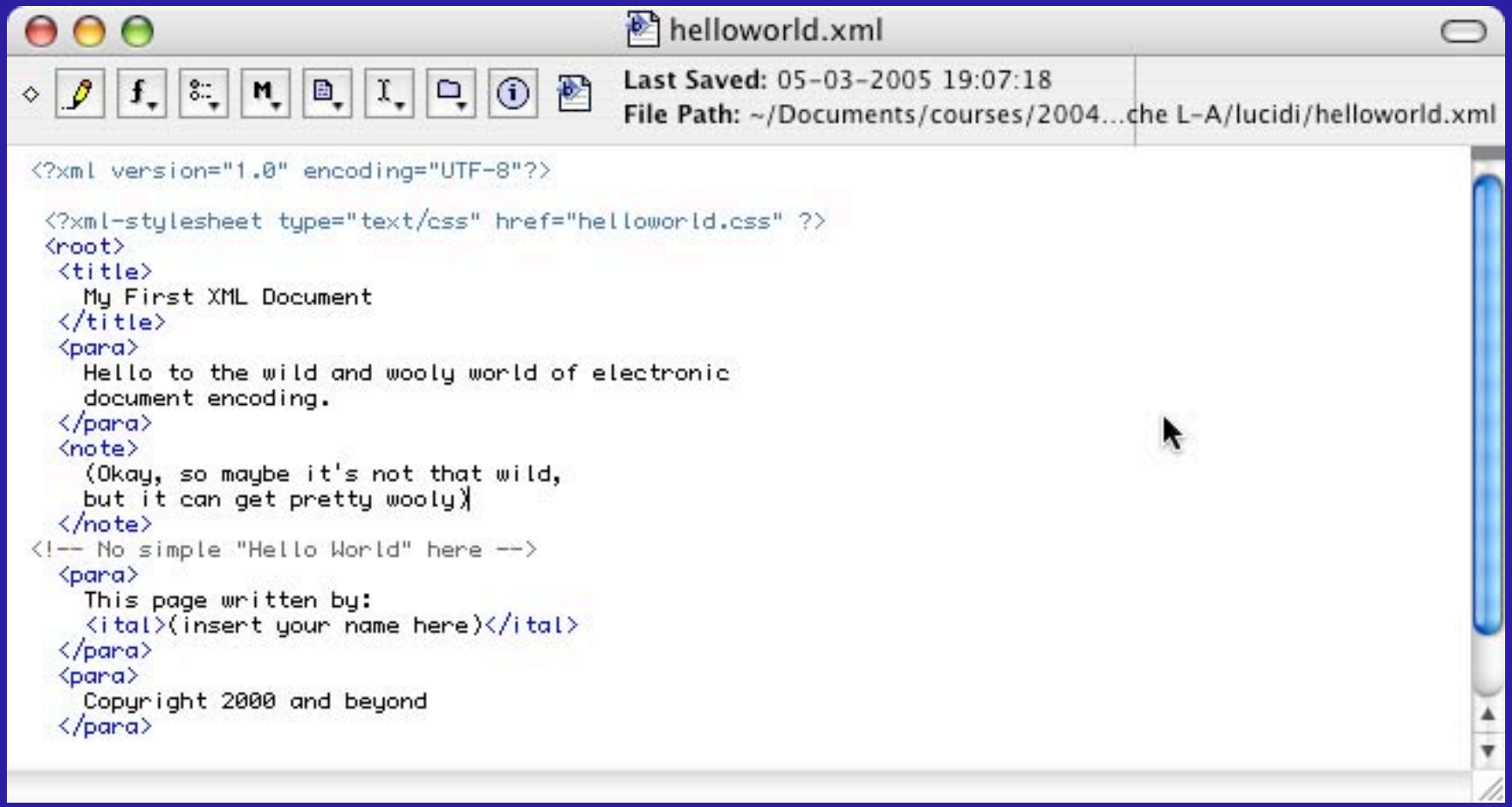
```
<?xml-stylesheet type="text/css" href="nomefile.css" ?>
```

- Foglio di stile che definisce lo stile per i tag del documento

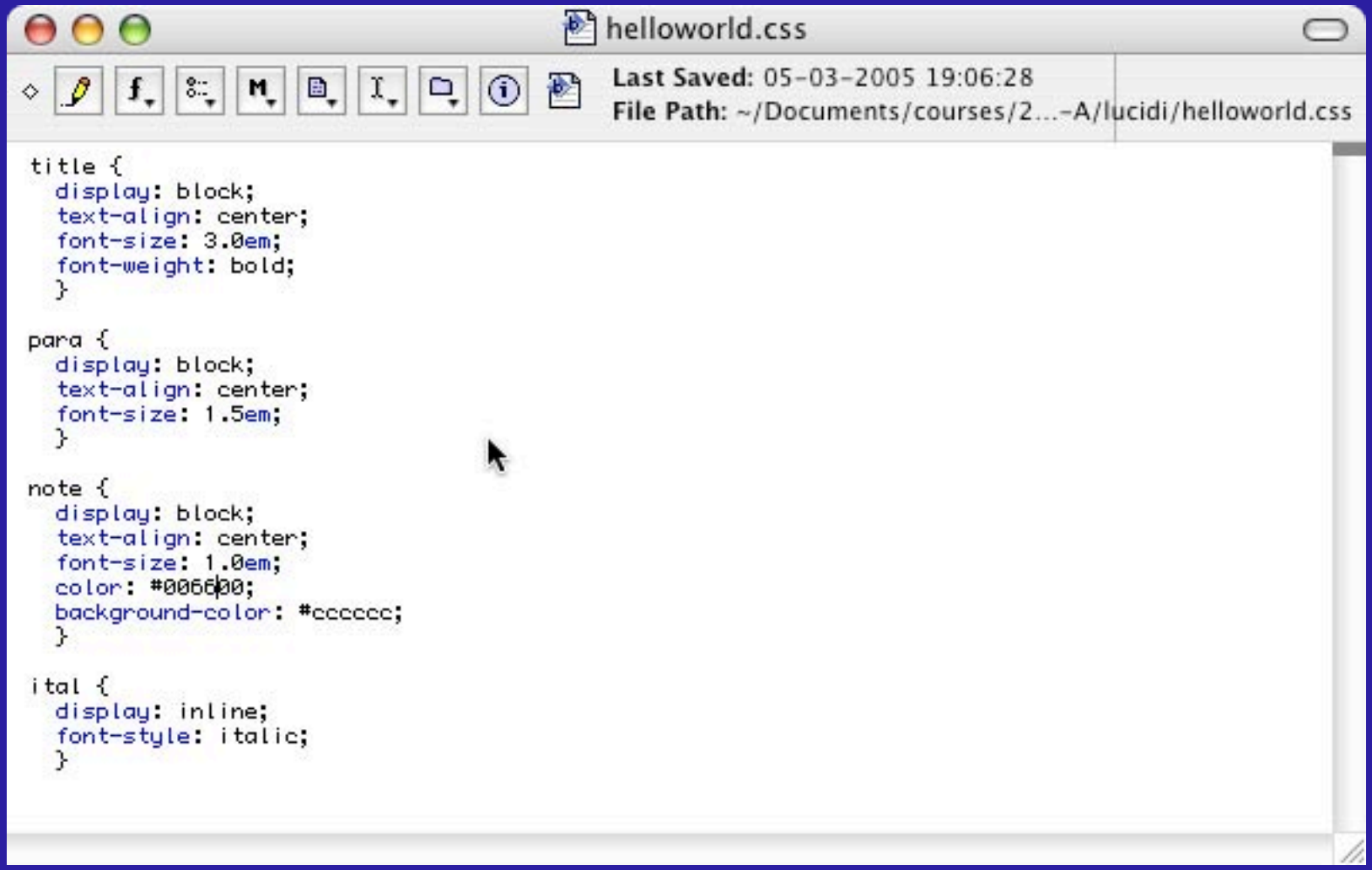
```
nometag {
   attributo1 : valore1;
   …
}
```

- Non è necessario il DTD o lo Schema
  - ma il browser comunque si può lamentare…

XML e CSS: esempio XML



XML e CSS: esempio CSS



In breve, perché XML?

QUANTO CONVIENE "complicare" le cose?

- È realmente utile adottare una rappresentazione basata su un linguaggio XML-based più complesso al solo fine di esprimere compiutamente i vincoli con il DTD?
- È solo questione di eleganza/correttezza,

o serve anche a qualcosa… "dopo"?

- Altrimenti, tanto vale usare nodi di puro testo…

Una rappresentazione basata su elementi e attributi XML anziché sul "puro testo"
- può essere validata in modo preciso, anche in termini strutturali
- semplifica notevolmente l'elaborazione tramite **XSLT**, e integrata adeguatamente nei sistemi web

XSL: eXtensible Stylesheet Language

- XML-based stylesheet language
  - http://www.w3.org/Style/XSL/
- Famiglia di linguaggi / standard per la trasformazione e presentazione di documenti XML
  - XSL Transformations (XSLT)
    - http://www.w3.org/TR/xslt
    - linguaggio per trasformare XML
  - XML Path Language (XPath)
    - http://www.w3.org/TR/xpath
    - linguaggio di espressioni per denotare e accedere a parti di documento XML
  - XSL Formatting Objects (XSL-FO)
    - vocabolario XML per specificare la semantica della formattazione

XSLT in una slide

- Metalinguaggio (XML-based) per esprimere regole di trasformazione di documenti da XML a XML
  - "result tree"
- Le regole si esprimono definendo dei template
  - ogni template indica con quali "parti" del documento XML fa match, specificando nella sua intestazione una **Xpath** expression
  - il template viene attivato per tutti e soli i nodi dell'albero XML che fanno match con la sua espressione XPath
  - dovrebbe esistere un solo template che faccia match con una data espressione Xpath, altrimenti scelta non-deterministica
  - esiste sempre un template radice che attiva gli altri; fa match con l'espressione root "/"
  - ogni template può a sua volta attivare altri template mediante la regola ricorsiva <xsl:apply-templates/>
- Linguaggio dichiarativo, privo di effetti collaterali
  - variabili a singolo assegnamento, non distruttivo

Esempio di XSLT sheet

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

<xsl:template match="para">
  <p><xsl:apply-templates/></p>
</xsl:template>

<xsl:template match="emphasis">
  <i><xsl:apply-templates/></i>
</xsl:template>

</xsl:stylesheet>
```

- trasforma

```
<?xml version='1.0'?>
<para>This is a <emphasis>test</emphasis>.</para>
```

- in

```
<?xml version="1.0" encoding="utf-8"?>
<p>This is a <i>test</i>.</p>
```