

Materiale concesso da HTML.it: <http://www.html.it>

Tutti i capitoli sono a cura di : Cesare Lamanna

XHTML

1. INTRODUZIONE	3
STRUTTURA DELLA GUIDA	3
IDEE GUIDA	4
2. COS'È XHTML	6
VERSIONI DI XHTML	7
RIFERIMENTI	8
3. VANTAGGI DI XHTML	9
CODICE PULITO E BEN STRUTTURATO	9
PORTABILITÀ	10
ESTENSIBILITÀ	11
ACCESSIBILITÀ	11
RIFERIMENTI	12
4. REGOLE DI BASE	13
VALIDAZIONE	13
ELEMENTO RADICE	14
NAMESPACE XHTML	14
DICHIARAZIONE DOCTYPE	14
5. LA MIA PRIMA PAGINA XHTML	15
RIFERIMENTI	16
6. IL PROLOGO	17
DICHIARAZIONE XML	17
DEFINIZIONE DEL DOCTYPE	17
RIFERIMENTI	19
7. LE DTD XHTML 1.0	20
DTD STRICT	21
DTD TRANSITIONAL	22
DTD FRAMESET	22
RIFERIMENTI	22
8. L'ELEMENTO RADICE	23
<HTML> È OBBLIGATORIO	23
IL NAMESPACE	24
RIFERIMENTI	25
9. LA TESTATA DEL DOCUMENTO	26
RIFERIMENTI	27
10. IL CORPO DEL DOCUMENTO	28

ELEMENTI BLOCCO.....	28
ELEMENTI INLINE	29
ATTRIBUTI DI BODY	29
11. DIFFERENZE CON HTML.....	31
GLI ELEMENTI DEVONO ESSERE CORRETTAMENTE ANNIDATI	31
I NOMI DI ELEMENTI E ATTRIBUTI DEVONO ESSERE IN MINUSCOLO	31
GLI ELEMENTI NON VUOTI DEVONO ESSERE CHIUSI	32
I VALORI DEGLI ATTRIBUTI DEVON ESSERE POSTI TRA VIRGOLETTE	32
OGNI ATTRIBUTO DEVE AVERE UN VALORE.....	32
GLI ELEMENTI VUOTI DEVONO TERMINARE CON />	32
USO DI ID E NAME.....	33
12. ESTENDERE XHTML: FML (FORM MARKUP LANGUAGE).....	34
13. PRESENTAZIONE DEL TUTORIAL.....	37
COSA REALIZZEREMO.....	37
COME FUNZIONA UN BLOG	37
QUALI LINGUAGGI USEREMO.....	38
COME PROCEDEREMO.....	39
QUALI STRUMENTI CI SERVONO.....	39
MATERIALI	39
RIFERIMENTI	39
14. STRUTTURA E STILE DEL BLOG.....	41
LA STRUTTURA	41
LO STILE	41
CODICE	42
E LE TABELLE?	43
RIFERIMENTI	44
15. TABELLE ADDIO	45
CENTRARE	45
TESTATA	46
MENU.....	46
CONTENUTO.....	47
NAVIGAZIONE	47
POST	47
16. I CONTENUTI: I POST DEL BLOG.....	49
GESTIONE DEI CONTENUTI CON XML	49
PRESENTARE XML	50
17. RISORSE XHTML E XML SUL WEB.....	52
XHTML	52
XML.....	52
18. LIBRI SU XHTML	54
IN ITALIANO	54
IN INGLESE	54

1. Introduzione

26 gennaio 2000: il World Wide Web Consortium (W3C (<http://www.w3c.org>)) rilascia la prima specifica del linguaggio di markup destinato a sostituire HTML. Quel linguaggio si chiama XHTML. La specifica di XHTML occupa una sola pagina. Non ci sono nuovi tag, non troverete nulla di rivoluzionario rispetto ad HTML 4.0. Eppure, quello che è attualmente il linguaggio raccomandato dal Consortium per la realizzazione di pagine web, è davvero un passo decisivo e fondamentale.

L'elemento chiave è la X. Sta per EXTENSIBLE, è la stessa X su cui si fonda quella che è sicuramente la pietra angolare della comunicazione digitale del futuro: XML (Extensible Markup Language). La cosa "rivoluzionaria" è appunto questa: HTML è ora una parte della grande famiglia XML, ne condivide regole di base e potenzialità. Risultato: il tempo dell'anarchia, del codice "sporco ma basta che funzioni", degli elementi proprietari imposti è finito. E tutti quelli seriamente interessati allo sviluppo di un web migliore dovrebbero esserne felici. Con questa guida tenteremo di darvi un quadro complessivo di questo linguaggio, mettendone in evidenza la struttura, i vantaggi, i problemi di implementazione, gli scenari di applicazione presenti e futuri.

Struttura della guida

La guida si articola in due sezioni e farà riferimento soprattutto a XHTML 1.0. Le lezioni della prima parte, più "teoriche" ma con costanti e precise proposte di codice di esempio, copriranno i seguenti argomenti:

- Definizione e vantaggi di XHTML
- Le regole di base e le differenze con HTML 4.0
- Analisi e realizzazione di una pagina XHTML valida
- La validazione dei documenti
- La compatibilità con i browser

- Editor e tools per la realizzazione e la conversione di pagine in XHTML
- Esempi di applicazioni avanzate

La seconda parte sarà invece centrata su un tutorial che vi guiderà nella realizzazione di un blog interamente scritto in XHTML. Senza la pretesa di essere esaustivi, vedremo come questo linguaggio si integri naturalmente con gli altri standard proposti dal W3C (XML, XSL, CSS) e come questa integrazione potrà cambiare il nostro approccio alla progettazione di siti e applicazioni web.

Nel corso delle lezioni si farà costante riferimento alle risorse di HTML.it. In particolare, riteniamo utile, come lettura propedeutica o complementare, la consultazione della guida ad HTML, della guida ai CSS e del corso su XML. Alla fine delle singole lezioni daremo riferimenti ad articoli e risorse online sugli argomenti trattati e aggiungeremo alla fine della guida una bibliografia essenziale.

Idee guida

Nessun progetto nasce senza un'idea guida. Le idee fondamentali che ci hanno guidato alla scrittura di queste lezioni sono tre:

1. **Standard: se non ora quando?** Ormai non ci sono più scuse. Tutti gli attori sul mercato dei browser producono, oggi, programmi aderenti ai principali standard del W3C. La situazione non potrà che migliorare, con supporti più adeguati e bug residui finalmente colmati. Per la prima volta da quando esiste il web chi progetta una pagina "standard compliant" ha la certezza che sarà visualizzata senza problemi con questi strumenti e che lo sarà nel futuro. È l'ora della "forward compatibility".
2. **Gli standard: separati ma integrati.** Prima si faceva tutto con HTML. Layout, stile, contenuto. Lavorare con gli standard significa invece usare un linguaggio per lo scopo per cui è stato concepito. La forza sta nell'integrazione dei singoli elementi.
3. **Cambiare mentalità.** Separare il contenuto dalla presentazione e dalla logica di programmazione. Non è uno slogan vuoto. Significa progetti più gestibili. Informazioni più facili da ritrovare. Siti più accessibili e trasportabili su altri sistemi. Ma significa anche cambiare le modalità operative, il modo stesso di pensare un sito o una singola pagina. È una

sfida da raccogliere. Cercheremo di aiutarvi a capire se ne vale la pena.

Buona lettura.

2. Cos'è XHTML

Per definire cos'è XHTML possiamo iniziare con una semplice espressione:

$$\text{HTML} + \text{XML} = \text{XHTML}$$

Esaminiamo i termini dell'operazione.

HTML

Se siete su questo sito sapete di cosa parliamo. HTML è un linguaggio di marcatura per presentare i contenuti di una pagina web. La sua semplicità è la base dell'esplosione di Internet. L'ultima raccomandazione (<http://www.w3c.org/TR/html4/>) rilasciata dal W3C è la 4.01 (dicembre 1999).

XML

XML è una sorta di "super-linguaggio" che consente la creazione di nuovi linguaggi di marcatura. Potente, flessibile e rigoroso è alla base di tutte le nuove specifiche tecnologiche rilasciate dal W3C e adottate ormai come standard dall'industria informatica. I principali obiettivi di XML, dichiarati nella prima specifica ufficiale (<http://www.w3.org/TR/2000/REC-xml-20001006>)(ottobre 1998), sono pochi ed espliciti: utilizzo del linguaggio su Internet, facilità di creazione dei documenti, supporto di più applicazioni, chiarezza e comprensibilità. Nella visione di Tim Berners Lee è destinato ad essere il fondamento di un web finalmente universale.

XHTML

XHTML è la riformulazione di HTML come applicazione XML. Ciò significa essenzialmente una cosa: un documento XHTML deve essere valido e ben formato. Torneremo in seguito su questi concetti. Per ora è importante chiarire il punto di vista del W3C su XHTML.

Piuttosto che sfornare una nuova versione del linguaggio, un HTML 5.0, gli

uomini del Consortium hanno compiuto un'opera di ridefinizione. Niente nuovi tag, attributi o metodi. Questi rimangono essenzialmente quelli di HTML 4.0. In pratica: il vocabolario rimane uguale, ma cambiano le regole sintattiche. Se si comprende questo fatto, sarà chiaro come XHTML risponda a due esigenze fondamentali:

1. portare HTML nella famiglia XML con i benefici che ciò comporta in termini di estensibilità e rigore sintattico
2. mantenere la compatibilità con i software che supportano HTML 4.0

Visto così, XHTML è un ponte tra passato e futuro. È un modo per imparare a pensare in XML partendo da un linguaggio che conosciamo, senza dover rinunciare, dunque, alle conoscenze già acquisite.

Versioni di XHTML

Le versioni di XHTML attualmente disponibili e pubblicate come raccomandazioni dal W3C sono tre.

XHTML 1.0

Pubblicata il 26 gennaio 2000 e seguita da una versione rivista dell'ottobre 2001. Consiste, come detto, nella riscrittura in XML di HTML 4.0 e si basa sulle tre DTD già definite per questo linguaggio:

- DTD Strict
- DTD Transitional
- DTD Frameset

XHTML Basic

È una versione "ridotta" (<http://www.w3.org/TR/xhtml-basic>) del linguaggio. Specificamente pensata per dispositivi mobili (PDA, cellulari), contiene solo gli elementi che si adattano a questi dispositivi (esclude, ad esempio, i frames che non hanno ovviamente senso in tale contesto). È destinata a sostituire WML come linguaggio di base per le applicazioni WAP.

XHTML 1.1

Basata sulla DTD Strict della versione 1.0, rappresenta la prima formulazione pratica del concetto di modularità (<http://www.w3.org/TR/xhtml-modularization>). In questa visione, gli elementi fondamentali (in pratica l'insieme di tag che definiscono la struttura di un documento) sono

raggruppati in una serie di moduli indipendenti, che possono essere implementati o esclusi secondo le necessità. Secondo il W3C è la base della futura estensione di XHTML con altri set di linguaggi o moduli, anche personalizzati.

Riferimenti

La guida HTML (<http://xhtml.html.it/guide/leggi/51/guida-html/>) di HTML.it
Il guida XML (<http://xml.html.it/guide/leggi/58/guida-xml-di-base/>) di HTML.it

XHTML sul W3C: news, specifiche, definizioni e discussioni
(<http://www.w3.org/MarkUp/>)

Traduzione italiana della specifica XHTML 1.0 (di Patrizia Andronico)
(<http://www.w3c.cnr.it/traduzioni/xhtml1-it.html>)

3. Vantaggi di XHTML

Quando si propone una nuova tecnologia è fatale sollevare dubbi e obiezioni.

Consideriamo il quadro d'insieme degli attuali linguaggi web.

Milioni di pagine sono scritte in HTML. Una gran parte di esse non è valida rispetto alle raccomandazioni del W3C, ma funziona. L'adozione di un nuovo linguaggio (XHTML) non è obbligatoria nè forzata: tutti i browser in commercio sono in grado di interpretare HTML 4. XHTML non introduce nuove features rispetto al suo predecessore. "Perchè allora imparare un nuovo linguaggio?". "Perchè dovrei riscrivere tutto il mio sito?". Sono domande scontate, ma legittime. Ecco alcune possibili risposte ed un semplice elenco dei vantaggi di XHTML.

Codice pulito e ben strutturato

Il passaggio ad XHTML può essere visto come un ritorno alle origini. HTML è nato come linguaggio per definire la **struttura** di un documento. Non ha nulla a che vedere con la presentazione. Eppure, è stato stiracchiato da tutte le parti, costretto a svolgere compiti per cui non è stato creato. Il caso delle tabelle è quello macroscopico. Sono nate per la presentazione di dati tabulari. Sono state impiegate come l'unico mezzo per costruire il layout della pagina.

E ancora: quando si sentì la necessità di gestire la tipografia dei documenti, venne introdotto il tag con i relativi attributi. La conseguenza è quella di pagine cariche di elementi inutili, più pesanti, difficili da modificare.

La responsabilità principale per questo uso improprio di HTML non ricade sugli sviluppatori. Nel 1996 (le date sono importanti!) il W3C ha rilasciato la versione definitiva di CSS1. Era questa la tecnologia destinata a definire la presentazione dei documenti. L'idea era chiara: HTML per la struttura, CSS per lo stile e il layout. Eppure, per avere browser che supportano decentemente questi standard si è dovuto attendere il 2000-2001. Quattro anni, tre generazioni di browser, milioni di siti costruiti tentando di risolvere incompatibilità e bug.

Con XHTML, almeno nella sua versione Strict, si torna ad un linguaggio che definisce solo la struttura. Semplicemente, se inserite elementi non supportati

(font, larghezza per le celle di tabelle o margini per il body, per citare solo alcuni esempi) il documento non è valido. Quindi: la formattazione si fa con i CSS. Mai più tag , mai più gif di un pixel. Tra poco, forse, niente più tabelle per il layout. Risultato: codice più pulito, più logico, più gestibile.

Esempio: due pagine realizzate secondo le due impostazioni, diciamo "old style" (http://html.it/guide/esempi/guida_xhtml/oldstyle.htm) e "new style" (http://html.it/guide/esempi/guida_xhtml/newstyle.html). Attenzione al codice riportato in fondo. Nel secondo caso tutta la formattazione è definita in un CSS e si vede! Peso delle due pagine: "old style" 3 kb, "new style" 1 kb.

Portabilità

Portabilità è la capacità/possibilità di un documento di essere visualizzato e implementato efficacemente su diversi sistemi: PC, PDA, cellulari WAP/GPRS, WebTV. Ora, se pensate alle limitazioni in termini di memoria, ampiezza dello schermo e usabilità di un terminale mobile, capirete perchè è importante un linguaggio essenziale, centrato sulla struttura del documento, privo di orpelli inutili. Ciò che ha senso è avere un titolo della pagina, un'intestazione, un paragrafo, una lista per scandire gli argomenti.

Sulla portabilità poggia l'enfasi con cui aziende del calibro di Nokia, Motorola, Ericsson o Siemens guardano ad XHTML. Dopo l'accettazione da parte del Wap Forum si può affermare con certezza che la piattaforma WAP 2 e tutta l'evoluzione dei servizi mobili sarà fondata sull'integrazione tra XHTML e CSS, con il supporto delle necessarie tecnologie sul lato server.

Nelle immagini (http://html.it/guide/esempi/guida_xhtml/xhtmllnokia.html), tratte da un documento di Nokia, è chiarissimo lo schema di distribuzione delle informazioni fondato su questa interazione. In breve:

- nella pagina XHTML incorporiamo diversi CSS per ciascun supporto
- il browser viene identificato
- su un PC vedremo il layout standard
- su un cellulare visualizziamo un layout "ridotto" e adatto alle caratteristiche del mezzo
- ciò che non cambia sono i contenuti

È solo uno degli scenari possibili. Se al quadro si aggiungono le enormi potenzialità del linguaggio XSL, si comprende come l'epoca delle tante versioni di uno stesso sito sia davvero al termine.

Estensibilità

Dal momento che XHTML è XML diventa estensibile. Significa che sarà facilissimo incorporare in un documento parti scritte in uno dei tanti linguaggi della famiglia XML.

Esempio. Dovete inserire in una pagina delle formule matematiche complesse. Basterà dichiarare il namespace relativo al linguaggio MathML e inserire nella pagina i tag specifici di quest'ultimo (il codice è tratto dal sito del W3C (<http://www.w3c.org/TR/xhtml1/>)).

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>A Math Example</title>
  </head>
  <body>
    <p>The following is MathML markup:</p>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply> <log/>
        <logbase>
          <cn> 3 </cn>
        </logbase>
        <ci> x </ci>
      </apply>
    </math>
  </body>
</html>
```

I frutti di questo approccio, che è il più semplice dei tanti possibili, sono ancora lontani dalla maturità. L'implementazione da parte dei browser è infatti carente, ma non mancano esempi funzionanti e di grande potenza. Parleremo in un'altra lezione di FML (Form Markup Language), un linguaggio che estende in maniera impressionante l'uso dei classici form di HTML.

Accessibilità

Un altro punto fondamentale. I documenti scritti in XHTML e validati sono naturalmente più accessibili. Da una parte la validazione richiede l'uso obbligatorio di funzionalità come il testo alternativo per le immagini. Dall'altra, una pagina che evita elementi non standard, ben definita nella struttura è di gran lunga meglio

gestibile da browser alternativi come quelli vocali o testuali. Per questi aspetti è obbligatoria la lettura delle linee guida per l'accessibilità del contenuto Web rilasciate dal W3C e tradotte anche in italiano (<http://www.point.uni.it/WAI/>).

Riferimenti

The fear of X

(<http://www.pcmag.com/article/0,2997,s%253D1487%2526a%253D7243,00.asp>): Molly Holzschlag è una delle personalità più influenti del web design americano. In questo articolo (tradotto sul numero di Novembre 2001 di PC Professionale) espone la sua visione su XHTML e sui vantaggi di questo linguaggio. Da leggere altri suoi articoli (<http://www.molly.com/molly/articles.html>) sull'argomento.

4. Regole di base

Entriamo nel vivo della conoscenza di XHTML esponendo le sue regole di base. Sono quelle che rendono un documento strettamente conforme (punto 3.1.1 (<http://www.w3c.cnr.it/traduzioni/xhtml1-it.html#docconf>) della specifica) e che ne definiscono i requisiti minimi essenziali. In pratica: se non si rispettano questi semplici punti il documento non può essere definito XHTML.

Validazione

Un documento deve essere convalidato rispetto ad una delle tre DTD XHTML del W3C. Vedremo nella prossima lezione come si effettua la convalida. Per ora è sufficiente chiarire che essa controlla essenzialmente due cose: che il documento sia **valido** e **ben formato**. Sono due concetti fondamentali derivati da XML e su cui vale la pena soffermarsi.

DOCUMENTI BEN FORMATI

XHTML, ricordiamolo, è fondato su XML. Da questo eredita il concetto di documento ben formato. Ciò significa che esso deve rispettare le regole della sintassi XML: presenza di un elemento radice, corretto annidamento degli elementi, chiusura obbligatoria dei tag vuoti, etc. Affronteremo nei dettagli questi aspetti nelle lezioni successive.

DOCUMENTI VALIDI

Un documento è valido se usa correttamente un linguaggio, vale a dire se usa nel modo giusto solo elementi specifici e consentiti. Ma dove vengono stabilite queste regole? Per XHTML (e per XML in genere, anche se in questo ambito si sta sviluppando l'uso degli schemi) le regole sono definite nelle DTD (Document Type Definition). Una DTD identifica gli elementi (tag) consentiti, cosa essi significano, come devono essere trattati (ad esempio, stabilisce

quali sono gli attributi possibili per ciascun elemento). In un documento XHTML la DTD deve essere obbligatoriamente specificata all'inizio. Per ora passiamo alla seconda regola di conformità.

Elemento radice

Ogni documento XML deve contenere un elemento radice. Si tratta dell'elemento che contiene al suo interno tutti gli altri:

```
<rubrica>
  <contatto>
    <nome>Marco</nome>
    <cognome>Rossi</cognome>
  </contatto>
</rubrica>
```

Nell'esempio l'elemento radice è <rubrica>. In un documento XHTML l'elemento radice deve essere <html>.

Namespace XHTML

L'elemento radice <html> deve contenere la dichiarazione di un namespace XML (**spazio dei nomi**) tramite l'attributo xmlns. Il namespace usato deve essere "http://www.w3.org/1999/xhtml".

Dichiarazione DOCTYPE

In un documento XHTML l'elemento radice deve essere preceduto da un elemento <!DOCTYPE>. All'interno di questo elemento è necessario specificare la DTD di riferimento e il suo URI.

Ora che conosciamo un pò meglio XHTML possiamo finalmente creare la nostra prima pagina.

5. La mia prima pagina XHTML

Ecco come si presenta il codice di una semplice pagina XHTML basata sulla DTD Strict (figura 1):

Figura 1 - La mia prima pagina XHTML

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      La mia prima pagina XHTML
    </title>
  </head>
  <body>
    <h1>Benvenuto!</h1>
    <p>Questo è il mondo di XHTML!</p>
  </body>
</html>
```

Nell'immagine abbiamo evidenziato con colori diversi le quattro sezioni essenziali di un documento XHTML:

- in rosso: **il prologo**
- in verde: **l'elemento radice (<html>)**
- in viola: **la testata (<head>)**
- in giallo: **il corpo del documento**

Le analizzeremo nelle prossime lezioni, definendo nei dettagli la funzione di ciascuna e le differenze con HTML. Per ora iniziamo con un pò di pratica. Il listato 1 (http://html.it/guide/esempi/guida_xhtml/listato1.html) contiene il codice della nostra prima pagina. Copiatelo e incollatelo, anche in un editor di testo. Salvate, nominando il documento "**test.html**", visualizzate nel vostro browser preferito.

A questo punto, dopo aver tanto insistito sul concetto di documento valido e ben formato, è opportuno imparare a fare la convalida. All'url <http://validator.w3.org> (<http://validator.w3.org>) è possibile effettuare la

validazione di qualunque documento presente in rete: basta inserire l'URI della pagina e cliccare su "Validate this page". Dal momento che la pagina appena creata è localizzata sulla nostra macchina dobbiamo usare un altro metodo.

Il W3C dà la possibilità di validare i documenti tramite l'upload del file sui suoi server. Usate quindi l'url <http://validator.w3.org/file-upload.html> (<http://validator.w3.org/file-upload.html>). Cercate il file da validare sul vostro PC e inviate la richiesta. Se il documento è valido riceverete, questo messaggio (figura 2):

Figura 2 - Il nostro documento è valido.

No errors found! *



Congratulations, this document validates as XHTML 1.0 Strict!

Bene. La prova è andata a buon fine, la pagina rispetta le regole. Possiamo ora passare all'analisi delle quattro sezioni del documento. Cominciamo dal prologo.

Riferimenti

Un validatore alternativo (<http://www.htmlhelp.com/tools/validator/>): potete validare i documenti anche su [htmlhelp.com](http://www.htmlhelp.com/) (<http://www.htmlhelp.com/>). Il sito fornisce utilissime notizie e suggerimenti su possibili problemi nella convalida.

6. Il prologo

Figura 1 - Prologo di un documento XHTML

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

La figura 1 mostra il prologo di un documento XHTML. Esso risulta composto da due parti: la dichiarazione XML e la definizione del DOCTYPE.

Dichiarazione XML

La nostra pagina inizia con una riga di codice: `<?xml version="1.0"?>`. La sua funzione è semplice: rendere esplicito il fatto che il documento è XML. Non è obbligatoria, ma il suo uso è consigliato dal W3C per tutti i documenti XML. Quando viene usata non deve essere preceduta da altre istruzioni.

All'interno della dichiarazione è possibile usare tre attributi. L'unico obbligatorio è **version** (il solo valore possibile è "1.0", in quanto non esistono altre versioni del linguaggio). Un altro attributo, opzionale, ma spesso usato è **encoding**. Serve a specificare la codifica del linguaggio:

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

L'ultimo attributo possibile è **standalone**. Se il valore è "yes" vuol dire che il documento non fa riferimento ad entità esterne.

Se il vostro obiettivo è la massima compatibilità potete omettere la dichiarazione XML. Molti browser hanno mostrato problemi così come alcuni editor (Dreamweaver). Se avete la necessità di specificare una codifica per i caratteri potete sempre usare un meta tag:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
```

Definizione del DOCTYPE

La dichiarazione del DOCTYPE (obbligatoria!) è composta da due sezioni:

1. Un FPI (Identificatore Formale Pubblico) riferito ad una delle tre DTD

© 1997-2006 [HTML.it](http://www.html.it)

La vendita, il noleggio, il prestito e la diffusione del contenuto di questa pagina sono vietate, tranne nei casi specificati nella pagina <http://www.html.it/info/note-legali.php>

XHTML (in rosso)

2. L'URI della DTD (in verde)

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

Essa, dunque, ha lo scopo di stabilire a quale delle tre DTD XHTML intendiamo conformarci e di dire al browser dove essa è collocata. Nel nostro esempio la DTD di riferimento è quella Strict, collocata sul sito del W3C. Il DOCTYPE, inoltre, non ha alcun effetto sulla presentazione della pagina.

Serve solo al validatore per stabilire le regole della convalida.

Notate anche la parola chiave PUBLIC. Significa che la DTD è pubblica, creata dal W3C. In effetti, in XML, è anche possibile definire DTD "private", specifiche per la nostra applicazione. In tal caso si usa la parola chiave SYSTEM.

Un'ultima notazione. Su molti siti (tra cui quello del W3C) vengono forniti schemi di dichiarazione DOCTYPE per le tre DTD che presentano URI relativi:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"DTD/xhtml1-strict.dtd">
```

Quest'uso comporta talvolta problemi in fase di validazione, pertanto è preferibile usare sempre URI assoluti. Riportiamo per comodità le tre dichiarazioni per ciascuna DTD. Potete usarle nelle vostre pagine con un semplice copia e incolla.

DTD STRICT

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

DTD TRANSITIONAL

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

DTD FRAMESET

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

L'argomento DTD merita comunque un approfondimento. È ciò di cui parleremo nella prossima lezione.

Riferimenti

Come vivere meglio con XHTML

(<http://www.alistapart.com/stories/betterliving/>): articolo apparso su **A list apart** con trucchi e suggerimenti su un uso appropriato del DOCTYPE.

7. Le DTD XHTML 1.0

Giunti a questo punto dovrebbe essere ormai chiara la funzione di una DTD: descrivere in un linguaggio comprensibile da una macchina la sintassi e la grammatica di un linguaggio XML. Il tutto allo scopo di verificare la validità di un documento che a quella DTD fa riferimento.

Le DTD XHTML 1.0 sono contenute in tre documenti che potete anche scaricare (<http://www.w3.org/TR/xhtml1/xhtml1.zip>) sul vostro computer, sia per imparare come sono fatte sia per usarle direttamente nel vostro sito. In questo modo dovrete modificare l'URI nella dichiarazione DOCTYPE:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.miodominio.it/DTD/xhtml1-strict.dtd">
```

Nell'esempio si suppone che la DTD sia localizzata nella cartella DTD di un server miodominio.it. Se ciò può sembrare una bizzarria ora non lo sarà forse tra qualche anno quando milioni di browser tenteranno di convalidare documenti XHTML contro le DTD del W3C. È evidente che il pericolo è quello di rallentamenti e strozzature.

L'aspetto su cui ora vogliamo soffermarci è il contenuto di una DTD. Il codice è quanto di più criptico si possa immaginare, ma non mancano manuali che spiegano bene come interpretarle. Facciamo un esempio. Ecco come viene definito nella DTD Strict l'uso dell'elemento :

```
<!ELEMENT img EMPTY>  
<!ATTLIST img  
  %attrs;  
  src %URI; #REQUIRED  
  alt %Text; #REQUIRED  
  longdesc %URI; #IMPLIED  
  height %Length; #IMPLIED  
  width %Length; #IMPLIED  
  usemap %URI; #IMPLIED  
  ismap (ismap) #IMPLIED>
```

Traduzione in italiano:

L'elemento immagine è vuoto (EMPTY). Può assumere tutti gli attributi fondamentali (%attrs;): sono quelli comuni alla maggior parte degli elementi (id, class, style, title). Altri attributi possibili sono: src, alt, longdesc, width, height, usemap, ismap. Src e alt sono obbligatori (#REQUIRED). Gli altri opzionali (#IMPLIED).

Conoscere il contenuto di una DTD è dunque importante. Se non inserite, per esempio, l'attributo alt per un'immagine e validate la pagina vi verrà segnalato l'errore e potrete correggerlo. Ma se si conoscono le regole in partenza è certamente meglio, si eviterà di procedere per prova e correzione d'errore.

Per fortuna non è necessario imparare una DTD. Esistono per lo scopo ottime reference, elenchi di tutti i tag, degli attributi e degli eventi consentiti che spiegano in dettaglio il loro uso. Nella sezione riferimenti in fondo alla pagina segnaliamo un paio di reference online, ma le trovate anche nei migliori manuali su XHTML o HTML.

Il riferimento ad HTML non deve stupire. Le tre DTD XHTML 1.0 sono infatti basate sulle corrispondenti DTD elaborate per HTML 4.0. Vedremo ora quali sono le principali caratteristiche di ciascuna.

DTD Strict

È la DTD più rigida, centrata esclusivamente sulla struttura del documento. Elimina diversi elementi ed esclude tutti gli attributi che definiscono la presentazione. Per questo scopo vanno usati i CSS. Segue un elenco degli elementi non supportati:

<applet>, <basefont>, <center>, <dir>,
, <frame>, <frameset>, <iframe>,
<isindex>, <menu>, <noframes>, <s>,
<strike>, <u>

Oltre agli elementi non consentiti particolare attenzione va posta ad attributi molto usati nella comune pratica del web design. Elenchiamo alcuni casi e rimandiamo ad una buona reference per i dettagli:

- sono esclusi tutti gli attributi del tag <body> tranne quelli comuni (vedi sopra (#stilicom))
- non si può usare align per l'allineamento del testo in paragrafi e altri elementi
- non è supportato l'attributo target per i link e i form
- per una tabella (<table>) non si possono specificare il bordo, il colore

- di sfondo (**bgcolor**) o l'allineamento (**align**)
- le celle di tabella (**<td>**) non supportano il colore di sfondo, la larghezza (**width**), l'altezza (**height**). Supportano invece l'allineamento del testo (**align**)

DTD Transitional

Basata sull'omologa DTD di HTML 4.0 è attualmente quella più usata. La spiegazione è semplice. Nelle intenzioni del W3C essa deve essere una sorta di passaggio verso una ridefinizione più rigida del linguaggio. In effetti è utile quando si voglia passare ad XHTML mantenendo il massimo grado di compatibilità con i vecchi browser. Supporta tutti gli elementi e gli attributi di presentazione di HTML 4.0, anche quelli ritenuti sconsigliati. Se dovete e volete ancora usare le tabelle per il layout o fare uso del tag `<font>` è la DTD che fa per voi.

DTD Frameset

È identica alla Transitional, ma va usata quando si utilizzano i frame. L'unica differenza è in pratica la sostituzione del tag `<body>` con `<frameset>` nella pagina principale.

Riferimenti

XHTML Reference di W3Schools

(http://www.w3schools.com/xhtml/xhtml_reference.asp): W3Schools è una specie di HTML.it americano. Tante guide e tutorial su vari aspetti del web design con un linguaggio semplice ed efficace. La reference XHTML è un capolavoro di chiarezza e facilità d'uso.

XHTML Reference di zvon.org

(<http://www.zvon.org/xxl/xhtmlReference/Output/index.html>): ottimo sito che spazia da HTML ai linguaggi emergenti. La reference XHTML è completa ma poco chiara.

8. L'elemento radice

Riprendiamo l'analisi delle quattro sezioni della nostra pagina XHTML. E consideriamo l'elemento radice:

Figura 1: Dichiarazione dell'elemento radice

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Come abbiamo visto, l'elemento radice di un documento XHTML deve essere `<html>`, che deve a sua volta contenere tutti gli altri elementi. Niente di nuovo direte, tutte le pagine HTML iniziano più o meno così:

```
<html>
  <head>
    <title>Lezione 7 - L'elemento radice</title>
  </head>
  <body bgcolor="#FFFFFF" text="#000000">
    ..... Contenuto della pagina .....
  </body>
</html>
```

Ma le differenze ci sono e qui potete capire davvero cosa significa il rigore di XML.

<HTML> è obbligatorio

Perchè prima non lo era? Incredibilmente la risposta è: no! `<html>` era considerato di default. Piccola prova:

```
<head>
  <title>Lezione 7 - L'elemento radice</title>
</head>
<body bgcolor="#FFFFFF" text="#000000">
  <h1>Questo è HTML!</h1>
</body>
```

Copiate e incollate queste poche righe in Blocco Note; salvate come "**prova.html**". Aprite in un browser. La scritta "Questo è HTML!" comparirà

bella ed evidente. Ma cosa volete farci, i nostri browser sono fatti così. Sanno perdonare molto, forse troppo. Si calcola che l'80% della potenzialità del parser HTML di un browser venga usata per risolvere gli errori di scrittura presenti nelle pagine. È evidente che sottoponendo questa pagina a validazione non sarete nemmeno presi in considerazione. Semplicemente questo non è un documento XHTML. Se un giorno i browser (come è auspicabile) effettueranno la verifica la nostra paginetta di test verrà sdegnosamente rifiutata perchè non corretta. Ma affronteremo l'argomento browser in una prossima lezione. Ora ribadiamo: `<html>` è obbligatorio.

Il namespace

L'elemento `<html>` può assumere questi attributi:

dir	Determina la direzione del testo
Lang	Specifica il linguaggio di base dell'elemento quando è interpretato come HTML
xml:lang	Specifica il linguaggio di base dell'elemento quando è interpretato come XML
Xmlns	Specifica il namespace predefinito per XHTML

L'unico attributo obbligatorio è **xmlns**. Il W3C, come visto, specifica anche il valore obbligatorio di tale attributo: "`http://www.w3.org/1999/xhtml`" (vedi figura 1 (#figura08a)).

La necessità di un namespace (spazio di nomi) dipende dalla derivazione da XML. Linguaggio estensibile per definizione. È possibile, infatti, estendere il set di tag di XHTML con elementi di altri linguaggi, anche creati personalmente. Specificare uno o più namespace evita la possibilità di conflitti tra i tag. Chiariamo anche qui con un esempio.

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//OVERFLOW//DTD XHTML-FML 1.0//EN"
"http://www.mozquito.org/dtd/xhtml-fml1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:x="http://www.mozquito.org/xhtml-fml">
<head>
  <title>Untitled</title>
</head>
<body>
```

```
<x:form>
</x:form>
</body>
</html>
```

Il codice riporta l'implementazione di FML (Form Markup Language) in un documento XHTML. FML, su cui torneremo, è un linguaggio che ridefinisce e potenzia l'uso dei form. Come si può notare all'interno dell'elemento `<html>` sono stati definiti due namespace: in rosso è evidenziato quello standard di XHTML, in blue quello di FML.

La differenza tra i due, a parte l'URI, sta nel prefisso. Il primo non ne ha. Il secondo ha come prefisso "**x**". Significa che i tag non preceduti da prefisso sono quelli standard di XHTML e come tali verranno interpretati. Ora date uno sguardo al codice. Nel corpo della pagina è stato inserito un form. Ma il tag è preceduto da **x**! Vuol dire che esso appartiene al namespace FML e come tale deve essere trattato dal browser. Senza specificare il prefisso avremmo avuto un classico form XHTML.

L'argomento namespace è attualmente uno dei più dibattuti nella comunità XML e dunque vi rimandiamo ad uno dei tanti siti sull'argomento o ad un manuale aggiornato per chiarificazioni e approfondimenti.

Riferimenti

Nozione e uso dei namespace: dal corso XML di HTML.it una buona introduzione all'argomento.

Form Markup Language (<http://www.mozquito.org/html/lang-english/xhtml.html>): articoli e reference su questa interessante estensione di XHTML

9. La testata del documento

Presente sin dalle prime specifiche di HTML, la testata (<head>...</head>) è ovviamente supportata in tutte le DTD XHTML:

Figura 1: Testata tipica di un documento XHTML

```
<head>
  <title>
    La mia prima pagina XHTML
  </title>
</head>
```

La funzione principale della sezione <head> è quella di contenere informazioni che non vengono direttamente visualizzate nella pagina, ma che sono comunque di grande rilievo. Ecco l'elenco degli elementi che possono apparire nella testata:

<base>	Usato per definire l'URL di base della pagina. Utilissimo per la risoluzione dei link relativi.
<isindex>	Sconsigliato. È un modo per creare elementi simili alle caselle di testo.
<link>	Contiene informazioni su documenti esterni collegati. Usato soprattutto per i CSS.
<meta>	Specifica informazioni di vario tipo sul documento.
<noscript>	Usato per visualizzazioni alternative nei browser che non supportano gli script.
<object>	Racchiude un oggetto.
<script>	Contiene script di programmazione.
<style>	Definisce le regole di formattazione per il documento corrente.

<title>

Specifica il titolo del documento che compare nella barra del titolo del browser

Di questi elementi l'unico richiesto nelle DTD XHTML 1.0 è title. In realtà se esso viene omissso, il validatore del W3C non segnala errori. Rimane il fatto che dare un titolo ad una pagina è una buona norma da seguire sempre ed è anche utilissimo per i motori di ricerca, che sempre più considerano questo elemento piuttosto che le classiche keywords definite nei meta tag.

Tutti gli elementi della testata sono comunque ben noti a chi ha dimestichezza con HTML e nel passaggio ad XHTML non hanno subito variazioni (a parte, come vedremo, il tag script). Vi rimandiamo pertanto ai riferimenti per approfondimenti o per un semplice ripasso.

Riferimenti

I meta tag (<http://xhtml.html.it/guide/lezione/1710/i-meta-tag/>): dalla guida di HTML.it una panoramica sull'uso di questi fondamentali elementi.

Inserire i fogli di stile in un documento

(<http://css.html.it/guide/lezione/16/inserire-i-fogli-di-stile-in-un-documento/>): la guida ai CSS di HTML.it spiega il più classico uso del tag <link>.

Dalla stessa guida (<http://css.html.it/guide/leggi/2/guida-css-di-base/>) consigli utili su come formattare un documento con l'elemento <style>.

10. Il corpo del documento

Figura 1 - Corpo del documento

```
<body>
  <h1>Benvenuto!</h1>
  <p>Questo è il mondo di XHTML!</p>
</body>
```

Il corpo del documento è la sezione in cui si sviluppa il contenuto. È racchiusa, come in HTML, tra i tag `<body>...</body>`. Gli elementi che possono comparire all'interno del corpo sono in genere suddivisi in due categorie: elementi blocco ed elementi inline.

Elementi blocco

Gli elementi blocco sono quelli che definiscono la struttura del documento. Possono contenere altri elementi blocco, elementi inline o testo. Quando sono inseriti danno origine ad una nuova riga nel flusso del documento. Chiariamo: se si inseriscono in una pagina queste due righe di codice:

```
<h1>Titolo</h1>
```

```
<p>Paragrafo</p>
```

il testo "Paragrafo" si troverà su una nuova riga, in quanto abbiamo inserito un nuovo elemento blocco (`<p>`).

Abbiamo riportato in allegato

(http://html.it/guide/esempi/guida_xhtml/block.html)l'elenco di tutti gli elementi blocco XHTML, specificando per ciascuno il supporto nelle tre DTD.

GERARCHIE E ANNIDAMENTO DEGLI ELEMENTI BLOCCO

Nella strutturazione del documento è fondamentale rispettare alcune semplici regole relative alla gerarchia e all'annidamento degli elementi blocco. Il primo elemento della gerarchia dovrebbe essere `<div>`, che definisce in pratica una sezione del documento. Al suo interno trovano posto gli altri elementi. La cosa importante è che bisogna evitare annidamenti errati, che i browser fanno passare senza problemi, ma che il validatore segnala impietosamente

in quanto violano le regole delle DTD. Esempio:

```
<p><div>Qui inserisco il mio testo</div></p>
```

Se inserite questa breve riga di codice in un documento visualizzerete regolarmente il testo. Niente problemi allora? Non proprio. In realtà il documento non è valido, in quanto l'elemento `<p>` non può contenere altri elementi blocco. Il giusto annidamento è questo:

```
<div><p>Qui inserisco il mio testo</p></div>
```

Avete imparato allora una cosa importante: non fidatevi dei browser per verificare se il documento che scrivete funziona. Molto spesso ciò che funziona non è valido e in XHTML la correttezza formale è obbligatoria. La consultazione di un buon riferimento XHTML o la consultazione diretta della DTD potrà chiarirvi ogni dubbio sul corretto annidamento degli elementi. Nella nostra lista (http://html.it/guide/esempi/guida_xhtml/block.html) abbiamo specificato le regole di contenuto di base per i principali elementi blocco.

Elementi inline

Gli elementi inline si distinguono da quelli di tipo blocco per due motivi: quando sono inseriti non danno origine a una nuova riga e possono contenere solo dati (essenzialmente testo) o altri elementi inline. Chiariamo anche qui con un esempio:

```
<p>Questo tasto è<b>grassetto</b></p>
```

La parte delimitata dai tag `<b>...</b>` non sarà posta su una nuova riga. Anche per gli elementi inline va posta molta attenzione all'annidamento. Esempi come questo:

```
<b><p>Testo in grassetto</p></b>
```

sono tollerati dai browser, ma non reggono al giudizio della validazione in quanto un elemento inline non può contenerne uno di tipo blocco. Evitiamoli! Anche per gli elementi inline forniamo in allegato una lista commentata (http://html.it/guide/esempi/guida_xhtml/inline.html).

Attributi di body

Gli attributi per il testo, i link, il colore di sfondo e i margini dell'elemento `<body>` sono espressamente vietati solo nella DTD Strict, ma erano già considerati sconsigliati in HTML 4.0. Non vanno pertanto usati e devono essere sostituiti dai CSS. Li ricordiamo:

- **alink**
- **background**
- **bgcolor**
- **link**
- **text**
- **vlink**

11. Differenze con HTML

Nella prima lezione abbiamo evidenziato come XHTML sia essenzialmente una ridefinizione di HTML e l'exkursus tra le varie sezioni di una pagina ha confermato questa affermazione. Al di là delle differenze tra le tre DTD, non abbiamo incontrato nuovi tag o elementi inusuali. Anche quelle che possono sembrare novità assolute, come la dichiarazione del DOCTYPE, erano in realtà già previste nelle precedenti versioni di HTML.

Le novità sostanziali riguardano piuttosto la sintassi, il modo in cui tag, attributi e valori vengono usati. Le esamineremo ora una per una, proponendo sempre il confronto con HTML. Attenzione: l'uso di queste convenzioni è normativo. XHTML è un'applicazione XML e alla sintassi di questo deve conformarsi. Non rispettare queste regole significa non avere documenti validi e ben formati. La cosa che le accomuna è che queste regole pongono fine ad una serie di procedure che HTML consentiva e che ora sono invece vincolate ad usi ben definiti.

Gli elementi devono essere correttamente annidati

HTML	XHTML
<code><i>un test</i></code>	<code><i>un test</i></code>

Il primo esempio non è corretto in XHTML. Il tag `<i>` si sovrappone a `<b>`. La seconda colonna mostra invece un corretto annidamento degli elementi. La prima pratica è consentita in HTML. Certo, il browser dovrà interpretare qualcosa che è ambiguo, ma alla fine ci restituirà un testo in grassetto-corsivo (ciò che volevamo). In XHTML non possono sorgere ambiguità, tutto segue una regola.

I nomi di elementi e attributi devono essere in minuscolo

XML è un linguaggio *case sensitive*. Significa che `<table>` e `<TABLE>` sono

cose diverse. In XHTML è consentito solo l'uso delle minuscole per i nomi di elementi e attributi. Anche qui regole più vincolanti.

HTML	XHTML
<code><TABLE><TR><TD>un test</TD></TR></TABLE></code>	<code><table><tr><td>un test</td></tr></table></code>

Gli elementi non vuoti devono essere chiusi

HTML	XHTML
<code><p>Test1 <p>Test2</code>	<code><p>Test1</p> <p>Test2</p></code>

In HTML la pratica esposta a sinistra è tollerata, in XHTML non lo è. Ogni elemento non vuoto (sono quelli che contengono testo o altri elementi) deve avere dopo il tag di apertura quello di chiusura.

I valori degli attributi devono essere posti tra virgolette

Anche questa pratica è tollerata in HTML. E, soprattutto, è causata da molti editor che la "dimenticano" con molta disinvoltura.

HTML	XHTML
<code></code>	<code></code>

Ogni attributo deve avere un valore

Alcuni attributi di HTML non hanno un valore (si dice che sono *standalone*). È il caso dell'attributo **selected** usato per identificare l'opzione di un menu a tendina selezionata all'apertura del documento. In XHTML anche questi attributi devono essere valorizzati.

HTML	XHTML
<code><option selected>test</option></code>	<code><option selected="selected">test</option></code>

Il problema si pone anche nei form. Se ad esso non si assegna un'azione avremo questa situazione:

```
<form action=""></form>
```

Si può risolvere assegnando un valore fittizio:

```
<form action="action"></form>
```

Gli elementi vuoti devono terminare con />

In XML tutti i tag devono essere chiusi. Ma cosa fare con gli elementi vuoti? Si tratta di quelli che non possono contenere nulla al loro interno ma semplicemente ricevere attributi: `<meta>`, `<br>`, `<hr>`, `<img>` sono i più comuni. In XHTML tali elementi vengono chiusi terminando la dichiarazione con `</>`.

HTML	XHTML
<code>
</code>	<code>
</code>
<code></code>	<code></code>

Uso di id e name

Per identificare un elemento si deve usare l'attributo `id` e non `name`. Apparentemente la cosa non fa una grinza. In questo modo si ha una perfetta conformità con XML dove `id` è l'attributo standard per l'identificazione dei frammenti. In realtà qui il cambiamento con HTML è notevole, perchè per elementi come `<img>` o `<a>` l'attributo di identificazione è proprio `name`. Il passaggio a `id` non pone problemi nei browser più recenti, ma con altri (Netscape 4, per esempio) non funziona. In questo caso e se la compatibilità all'indietro è assolutamente necessaria, la stessa specifica XHTML 1.0 suggerisce di usare entrambi gli attributi, anche se ciò è contro le regole:

HTML	XHTML
<code></code>	<code></code>
	Per compatibilità:
	<code></code>

Nelle specifiche successive (cosa che è accaduta con XHTML 1.1) l'attributo `name` è stato abolito completamente.

12. Estendere XHTML: FML (Form Markup Language)

Attenzione! il sito Mozquito.com, cui si fa continuo riferimento all'interno di questa lezione, ha purtroppo chiuso alcuni mesi dopo la pubblicazione di questa guida. Tuttavia, pur con la mancanza di questi link, la lezione conserva la sua utilità didattica.

Mozquito (#) è una delle aziende più impegnate nella ridefinizione dei linguaggi Web basati su XML e partecipa attivamente ai lavori del W3C. Tra le sue realizzazioni ci occupiamo qui di FML (Form Markup Language). È un linguaggio che estende XHTML e che ridefinisce l'uso e le potenzialità dei classici form HTML. Stando alla documentazione ufficiale dell'azienda, FML dovrebbe:

- ottimizzare e potenziare l'interazione tra l'utente e il server
- ottenere con un solo linguaggio di markup risultati raggiungibili solo con uso complesso di Javascript o plug-in esterni
- creare facilmente documenti distribuibili su varie piattaforme

Concretamente significa, ad esempio, creare velocemente slide-show, quiz, applicazioni di e-commerce, sondaggi, motori di ricerca. Per apprezzare le possibilità di questo linguaggio potete visualizzare alcuni esempi o il tutorial presente sul sito di Mozquito: tutta l'applicazione è realizzata in una sola pagina, senza uso di frame (un'altra delle potenti caratteristiche del linguaggio).

L'implementazione di FML è un classico esempio funzionante di modularizzazione. Riprendiamo l'esempio già visto in un'altra lezione:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//OVERFLOW//DTD XHTML-FML 1.0//EN"
"http://www.mozquito.org/dtd/xhtml-fml1.dtd">
<html xmlns=http://www.w3.org/1999/xhtml
xmlns:x="http://www.mozquito.org/xhtml-fml">
<head>
```

```

<title>Untitled</title>
</head>
<body>
  <x:form>
  </x:form>
</body>
</html>

```

Apparentemente è un normale documento XHTML, ma le differenze ci sono:

1. La DTD (in verde) non è quella pubblica del W3C ma una versione "modificata" che implementa le caratteristiche di FML
2. Vengono dichiarati due namespace: quello standard di XHTML (in rosso), quello per FML (in blue)

Questo è ciò che vediamo nel documento. In realtà, il cuore di tutto risiede nella DTD. Essa risulta come un mix della DTD XHTML 1.1 per certi moduli e della DTD FML per altri. Come si ottiene questo risultato? Escludendo i moduli riguardanti form, applet oppure oggetti e sostituendoli con quelli di FML. Il codice di esempio riporta la dichiarazione di esclusione dei moduli di XHTML:

```

<!ENTITY % xhtml-applet.module "IGNORE" >
<!ENTITY % xhtml-script.module "IGNORE" >
<!ENTITY % xhtml-object.module "IGNORE" >
<!ENTITY % xhtml-form.module "IGNORE" >
<!ENTITY % xhtml-model.module "IGNORE" >

```

Subito dopo, scorrendo la DTD, troviamo la dichiarazione dei moduli FML che sostituiscono i form:

```

<!-- This section declares parameter entities used to provide
namespace-qualified names for all FML element types.
-->
<!-- module: fml-form-1.mod -->
<!ENTITY % FML.form.qname "%FML.pfx;form" >
<!ENTITY % FML.button.qname "%FML.pfx;button" >
<!ENTITY % FML.textinput.qname "%FML.pfx;textinput" >

```

Non entriamo in dettagli estremamente complessi: il meccanismo dovrebbe essere chiaro.

Realizzare documenti in XHTML-FML non è difficile. Per provare basta scaricare la demo di Mosquito Factory 1.5

(http://html.it/guide/esempi/guida_xhtml/download/mozquito-factory151-

win.exe). Si tratta di un editor che implementa i form FML. I file prodotti dall'applicazione hanno come estensione XHTML e sono di dimensione davvero ridotte se si pensa a quali caratteristiche implementano. Ma con un ottimo meccanismo di codifica possono essere immediatamente convertite in HTML. Avrete le stesse funzionalità, ma con codice Javascript e purtroppo con dimensioni maggiori: da 12kb si può passare a 71kb! Cosa fare? Aspettare che ci siano browser in grado di leggere direttamente le pagine XHTML-FML, trasformare in HTML ma con gli svantaggi visti oppure appoggiarsi al Matrix Server della stessa Mozquito che effettua le trasformazioni sul server mantenendo dimensioni accettabili.

Uno dei motivi per scaricare la demo di Mozquito Factory è anche la presenza di molti esempi funzionanti, che potranno rendere la giusta idea di questa eccellente estensione di XHTML.

13. Presentazione del tutorial

Cosa realizzeremo

Obiettivo del tutorial è la realizzazione di un blog. Per i pochi che non lo sanno: blog è l'abbreviazione di weblog. È una sorta di cronaca digitale personale con una serie di post su vari argomenti disposti in ordine cronologico. Il fenomeno è esploso da un paio di anni ed ha dato vita a sperimentazioni interessantissime: sui contenuti, sul design, sulla creazione di sistemi di gestione dei contenuti (CMS).

Per fare un blog serve essenzialmente una cosa: avere qualcosa da dire. Tecnicamente, ci si può appoggiare a servizi dedicati (Blogger (<http://www.blogger.com>), Manila (<http://manila.userland.com>), LiveJournal (<http://www.livejournal.com>), Pitas (<http://www.pitas.com>)) o fare tutto da sè. Se questa è la vostra soluzione, sappiate che per una gestione facile è fondamentale avere a disposizione uno spazio web con supporto di tecnologie server-side. Tutto va bene: Windows + ASP, come Apache con PHP e MySQL.

Come funziona un blog

Il funzionamento di un blog è semplicissimo. Sono essenzialmente basati su una sola pagina principale che ospita i post quotidiani, ma possono contenere una serie di link accessori interni (biografia, contatti, archivi e cose del genere) o esterni (siti interessanti, segnalazioni, etc). Una visita a blogger.com (<http://www.blogger.com>) potrà darvi un'idea su stili e tendenze.

La fase cruciale è quella dell'aggiornamento e dell'inserimento dei nuovi post. Il metodo deve essere semplice e deve consentire possibilmente la conservazione dei contenuti in database o sistemi simili. Ecco perchè è importante avere a disposizione tecnologie server-side. Per il nostro blog non ci baseremo su database, ma useremo il più semplice XML. E visto che si

parla di linguaggi, vediamo quali sono le tecnologie che adotteremo.

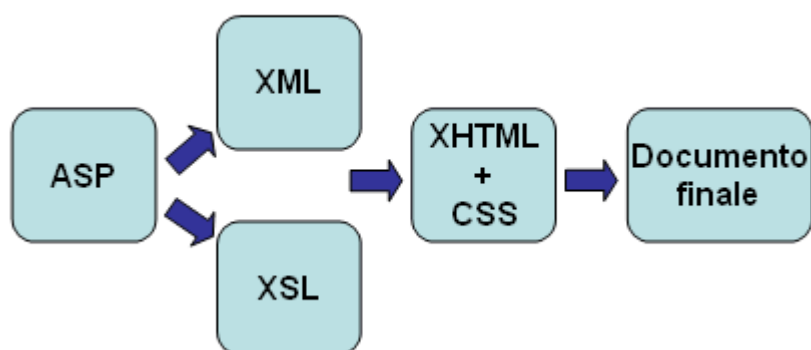
Quali linguaggi useremo

L'idea guida di questo tutorial è che i linguaggi standard del W3C danno il meglio di sé se usati per lo scopo per cui sono stati creati. Sono come singoli musicisti che vanno messi insieme per fare un'orchestra. Ecco, allora, la lista ragionata dei nostri "suonatori":

Linguaggio	Funzione
XHTML	Lo useremo per definire la struttura delle nostre pagine.
CSS	Con i CSS imposteremo il layout e lo stile grafico/tipografico del blog.
XML	In un file XML conserveremo i post che andremo poi a incorporare nella pagina principale.
XSL	Con XSL (Extensible Stylesheet Language) trasformeremo in XHTML i contenuti del documento XML.
ASP	Grazie ad ASP, che lavorerà sul lato server, supereremo le difficoltà di implementazione di XML nei browser.

Il workflow dell'applicazione può essere così schematizzato:

Figura 1. Workflow della generazione dei documenti



Con ASP carichiamo il documento XML e il foglio di stile XSL che lo trasformerà. Il risultato della trasformazione, che avviene sul server, sarà un documento XHTML con un CSS collegato che ne definisce lo stile. Questo documento sarà la pagina principale del blog.

Come procederemo

Le lezioni del tutorial affronteranno, nell'ordine, questi punti:

1. creazione dei prototipi grafici
2. creazione della struttura di base in XHTML
3. definizione dello stile e del layout con i CSS
4. gestione dei post con XML
5. trasformazioni con XSL
6. gestione del sito con ASP
7. aggiornamento del blog

Nell'anno accademico in corso non tratteremo i punti 5), 6) e 7).

Quali strumenti ci servono

Potrebbe bastare un editor di testo. Sia perchè scrivendo si ha un controllo perfetto sul codice, sia perchè forniremo di ogni documento il codice sorgente (basta un copia e incolla). Comunque, se lavorate in Windows, vi suggerisco il terzetto Macromedia Home Site + XML Spy + Top Style Pro 2.10. Per testare tutto è opportuno installare e settare correttamente il Personal Web Server o Internet Information Server.

Materiali

Abbiamo riunito in un file zip tutti i file completati e commentati della nostra applicazione. Potete scaricarli per seguire meglio le lezioni o testare subito il blog. Scompattando il file verrà creata automaticamente una cartella "blog". Scarica il file «blog.zip» (http://html.it/guide/esempi/guida_xhtml/blog.zip).

Riferimenti

Tra i riferimenti sono compresi ovviamente i siti citati nella lezione. Aggiungo i link di altri blog dedicati al web design e accomunati da alcune specifiche tecniche per noi importanti: scrittura in XHTML, validità, rispetto degli standard, utilizzo dei CSS. Tra essi (perdonatemi) il link del mio blog. Questo tutorial non è altro che la spiegazione di come è stato realizzato, anche se cambiano alcune funzioni avanzate e la grafica che è stata per l'occasione notevolmente semplificata:

Zeldman.com (<http://www.zeldman.com>): più di un blog: un riferimento assoluto.

Waferbaby (<http://waferbaby.com>): stile essenziale, contenuti originali. Un concentrato di intelligenza.

Hivelogic (<http://www.hivelogic.com/weblog/>): pulito, ottima tipografia, è il blog di Dan Benjamin.

14. Struttura e stile del blog

Un passo preliminare e importante in un progetto web è la realizzazione di un prototipo grafico. Esso è utile per evidenziare la struttura e gli elementi stilistici di base che saranno poi tradotti in codice XHTML (per la struttura) o CSS (per lo stile).

La struttura

Iniziamo con la struttura. Ecco l'immagine del prototipo strutturale (http://html.it/guide/esempi/guida_xhtml/struttura.gif - vedi Appendice A) realizzato con Fireworks.

La pagina principale del nostro blog (X-blog) è costituita da sei sezioni:

1. **Sezione principale:** è quella che contiene tutte le altre e che definisce lo spazio della pagina destinato al blog
2. **Testata:** qui inseriremo il logo
3. **Menu:** barra di navigazione orizzontale con link alle sezioni interne del sito
4. **Contenuto:** è una sorta di box che definisce la sezione centrale, a sua volta distinta in due sottosezioni
5. **Navigazione:** contiene link esterni suddivisi in categorie
6. **Post:** qui verranno inseriti i post di X-blog

Al momento di tradurre in XHTML questa struttura non faremo uso di tabelle. Ogni elemento deve fare il suo lavoro. Quando vorremo mostrare la classifica della serie A, useremo una tabella. Qui si tratta di definire sezioni, blocchi di contenuto generici e in XHTML l'elemento da usare è `<div>`.

Lo stile

A questo punto possiamo dare un pò di vita alla nuda collezione di box annidati vista in precedenza. È il momento creativo. Qui imposteremo colori,

margini, font, spaziature. Ed anche questa volta è opportuno, prima di mettere mano al nostro editor CSS preferito, preparare il terreno con un buon prototipo grafico.

Da questa immagine (http://html.it/guide/esempi/guida_xhtml/stile.gif- vedi Appendice B) potete farvi un'idea abbastanza precisa di come apparirà X-blog. Analizziamo alcuni dettagli:

- il font principale è **Verdana**, ma nel CSS vanno definiti caratteri alternativi
- la sezione **menu** ha uno sfondo giallo con testo rosso
- lo sfondo della sezione **navigazione** richiama quello della **testata**
- l'intestazione delle categorie va differenziata dai link: la prima sarà definita con un tag `<h1>`, i secondi con un `<p>`
- nella sezione **post** abbiamo questa struttura: titolo del post (`<h1>`), data (`<h2>`), testo (`<p>`)

Codice

Ora si può finalmente tradurre l'idea in codice. Ovviamente definiremo la struttura in XHTML, lo stile in un CSS.

Nel listato 3 (http://html.it/guide/esempi/guida_xhtml/listato3.html - vedi Appendice C) abbiamo riportato il codice del documento "**blog.html**", la pagina principale. Notate innanzitutto il rispetto delle regole di base XHTML (DOCTYPE, namespace, chiusura degli elementi, uso delle minuscole). Osservate poi la pulizia e l'essenzialità. La DTD usata è quella Strict. Il listato 4 (http://html.it/guide/esempi/guida_xhtml/listato4.html - vedi Appendice D) contiene invece una versione commentata dello stesso documento: vi aiuterà a chiarire ogni aspetto relativo alla struttura. Attenzione però: il codice di questo documento vi serva da punto di riferimento, non consideratelo come un punto d'arrivo. Nella nostra applicazione esso sarà generato automaticamente grazie alla magia di XSL. Certo, il gioco potrebbe anche finire qui. Ma pensate al momento in cui dovrete inserire nuovi post. Vi sembra comodo dover intervenire ogni volta sul documento XHTML? Il nostro obiettivo è separare i contenuti dalla presentazione. Quindi, ancora un pò di pazienza, ci arriveremo.

Nel foglio di stile "**mainstyle.css**" definiamo invece le regole di formattazione. Anche di questo file forniamo, nel listato 5 (http://html.it/guide/esempi/guida_xhtml/listato5.html - vedi Appendice E), una versione commentata. Il CSS è piuttosto semplice.

Dopo le regole generiche per il tag <body>, contiene sei **id** (contraddistinti dal simbolo #), uno per ciascuna sezione:

- **sezprinc**
- **testata**
- **menu**
- **contenuto**
- **navigazione**
- **post**

Di ognuno sono definite le regole di formattazione (larghezza, margini, bordi, colori, font, etc) e ognuno viene associato ad un <div> nel file XHTML tramite l'attributo id:

<div id="sezprinc">, <div id="menu">, etc.

Inoltre, sono stati creati dei selettori contestuali. Visto che intendiamo usare il tag <p> o <h1> in diverse sezioni e visto che devono avere una formattazione diversa si usa questa particolare forma di selettore. Con #navigazione h1 stabilisco come deve apparire l'elemento <h1> all'interno della sezione **navigazione**, con #post h1 come appare nella sezione **post**.

Ricordiamo che tutti i documenti sono contenuti nell'allegato file dei materiali (http://html.it/guide/esempi/guida_xhtml/download/blog.zip). Volendo testare il tutto, scompattate lo zip e aprite nel browser il documento "blog.html".

E le tabelle?

Ritorniamo ad un punto appena accennato. Il layout di X-blog è fatto senza tabelle. È una tecnica ancora non completamente matura, ma che comincia a dare buoni frutti proprio nel settore dei siti personali e dei blog. La non perfetta implementazione dei browser costringe ancora a usare vari trucchetti per superare alcune incompatibilità. Nei browser recenti (quinta e sesta generazione di Explorer, Netscape e Opera, più Mozilla) non avrete problemi. Rimangono i browser datati. Per questi il supporto di questo tipo di layout è carente. Ma un blog è un sito personale e se non si sperimenta qui non vedo dove. Le tecniche per l'implementazione di layout senza tabelle sono molte. La sezione riferimenti contiene un elenco di ottimi siti dedicati all'argomento. Nella prossima lezione ci limiteremo a spiegare il metodo usato per X-blog.

Riferimenti

La sezione riferimenti di questa lezione intende darvi un quadro dell'implementazione di layout senza tabelle, basati solo sui CSS:

Introduzione al layout con i CSS

(<http://developer.apple.com/internet/css/introcsslayout.html>): introduzione?

È un articolo completo, ben scritto, ricco di esempi, il migliore sull'argomento. Eric Costello sul sito di Apple.

CSS layout techniques (<http://www.glish.com/css/>): il sito personale dello stesso Costello

CSS Edge (<http://www.meyerweb.com/eric/css/edge/>): esperimenti con i CSS di Eric Meyer

Box Lesson

(http://www.thenoodleincident.com/tutorials/box_lesson/index.html):

raccolta incomparabile di trucchi, tecniche e layout pronti per l'uso.

15. Tabelle addio

In questa lezione analizzeremo alcuni dettagli relativi al foglio di stile "**mainstyle.css**". In particolare spiegheremo in che modo sia stato realizzato il layout senza fare uso di tabelle.

Centrare

Abbiamo visto nella precedente lezione che gli elementi fondamentali del CSS sono i sei **#id** che definiscono le proprietà di formattazione delle corrispondenti sezioni (identificate con il tag <div> nel documento XHTML). La prima cosa da evidenziare è il metodo usato per centrare il box principale (**#sezprinc**).

In HTML avremmo inserito una tabella centrandola con l'attributo align:

```
<table align="center">
```

Nei CSS non è possibile usare un attributo simile se non per il testo (proprietà text-align). L'ostacolo può essere superato con un uso accorto della proprietà margin. L'id **#sezprinc** è così definito:

```
#sezprinc {  
    background: #FFFFFF;  
    -----  
    margin : 20px auto;  
    text-align : left;  
    width : 574px;  
}
```

Per centrare si procede in questo modo:

1. Si assegna al box una larghezza esplicita (width: 574px;)
2. Si assegna ai margini sinistro e destro il valore auto. In questo modo essi avranno una larghezza identica.

Nel nostro caso il valore della proprietà margin va così interpretato: il

margin superiore è di 20 pixel, quello degli altri lati sarà calcolato automaticamente (auto). Questo è importante, perchè ci consente un design fluido e ci fa essere certi che cambiando risoluzione il box sarà sempre centrato. Alla risoluzione di 800x600 figura 1 - Appendice F (http://html.it/guide/esempi/guida_xhtml/lez22_fig1.jpg), ad esempio, i margini sinistro e destro saranno di 113 pixel ($800-574=226/2=113$). A 1024x768 di 225 pixel figura 2 - Appendice F (http://html.it/guide/esempi/guida_xhtml/lez22_fig2.jpg). Però...C'è un piccolo problema nell'uso di questo metodo: Explorer 5 per Windows non interpreta bene la tecnica. Ma il modo per ingannarlo c'è. Il browser di casa Microsoft non gestisce secondo gli standard nemmeno l'attributo text-align. Infatti, applica l'allineamento anche ai box e non solo al testo che contengono. Allora, visto che **#sezprinc** è contenuto nell'elemento **body** basta aggiungere tra gli attributi di quest'ultimo l'istruzione "text-align:center;". Ecco il codice del nostro CSS, dove lo stile dell'elemento body occupa infatti il primo posto:

```
body {  
    background : #FFFFFF;  
    -----  
    text-align : center;  
}
```

Tutto bene? Un attimo. Explorer 5 in questo modo centrerà correttamente il box, ma sorge un problema. Se definiamo l'attributo "text-align:center;" per l'elemento body succede che in tutti i box sottostanti il testo sarà centrato. Rimediare è ancora facile. Basta specificare nelle proprietà di **#sezprinc** l'attributo "text-align:left;", cosa che, come potete osservare sopra, abbiamo fatto puntualmente.

Testata

L'id **#testata** non presenta particolari problemi. In questo caso abbiamo settato esplicitamente l'altezza (height: 75px;), ma non la larghezza: essa sarà pertanto identica a quella del box parente (**#sezprinc**).

Menu

Poche osservazioni anche sull'id **#menu**. Abbiamo voluto che fosse meno ampio della testata. La sua larghezza è infatti di 512px. Con i margini abbiamo quindi impostato la sua posizione rispetto agli altri box:

margin : 0px 26px 1px;

Ricordiamo qui che l'ordine in cui vanno letti i valori di margin è il seguente: TOP, RIGHT, BOTTOM, LEFT. In pratica, bisogna seguire, a partire dall'alto un senso orario. Nel nostro caso, la sezione **menu** sarà praticamente unita alla **testata** (top = 0px), avrà un margine di 26px a destra e di un solo pixel in basso. E a sinistra? Bene, nei CSS esistono delle semplici regole per evitare la replica dei valori. Quando manca il valore per il lato sinistro (dovrebbe essere il quarto), esso sarà uguale a quello del lato destro (il secondo). Quindi, il margine sinistro è di 26px.

Contenuto

Il box **#contenuto** è una sorta di scatola vuota che serve semplicemente a delimitare la sezione centrale e a stabilirne le misure di base. La larghezza e i margini sono identici a quelli della sezione **#menu**. La sua utilità consiste anche nel facilitare la gestione dei due box che contiene.

Navigazione

A parte i bordi, il colore di fondo e il padding, il menu di navigazione a sinistra ha due proprietà importanti che influenzano il layout generale della pagina.

La prima è la dichiarazione **float: left;**. Per capire la proprietà float dovete pensare al modo usato in HTML per far scorrere il testo attorno ad un'immagine. Se si usa l'attributo `<img align="left">` l'immagine rimane a sinistra e il testo scorre intorno alla sua destra figura 3 - Appendice F (http://html.it/guide/esempi/guida_xhtml/lez22_fig3.gif). Allo stesso modo, definendo per l'id **#navigazione** `float: left;` facciamo in modo che esso rimanga a sinistra e che gli altri elementi adiacenti vengano spostati alla sua destra, ma non su una riga diversa, cosa che avviene per il box **#post**. Ecco cosa accade se eliminiamo l'istruzione figura 4 - Appendice F (http://html.it/guide/esempi/guida_xhtml/lez22_fig4.gif).

La seconda proprietà importante è la larghezza, che è uguale a 140px. Rimane la questione dell'altezza. Vi renderete conto che se non si specifica nessun valore, essa sarà basata sul contenuto. Se volete un'altezza fissa, usate `height` come per la testata.

Post

Siamo alla fine. L'id **#post** ha una larghezza di 350px ed è posizionato alla destra di **#navigazione**. Ma dove va posizionato precisamente? Qui entra in gioco il margine sinistro. Esso ha come valore 142px (margin : 0px 0px 0px 142px;), calcolati a partire dall'elemento parente (l'id **#contenuto**). Non è casuale. Se avessimo settato una distanza minore esso si sarebbe sovrapposto a **#navigazione**. Infatti, i 142px non sono altro che uguali alla larghezza di **#navigazione** + 2 pixel. L'immagine figura 5 - Appendice F (http://html.it/guide/esempi/guida_xhtml/lez22_fig5.gif) potrà chiarirvi questi particolari che possono sembrare un pò astrusi.

Bene. Il consiglio a questo punto è quello di sbizzarrirvi. Giocate un pò con il foglio di stile. Cambiate colori, margini, bordi, font. Capirete che la creatività si esprime proprio nei CSS, mentre il documento XHTML rimane intatto nella sua struttura.

Ora però dobbiamo riempire X-blog con i nostri post. È il momento di pensare ai contenuti.

16. I contenuti: i post del blog

Nelle lezioni precedenti abbiamo evidenziato soprattutto una delle caratteristiche di XML: il suo rigore. Correttezza formale e validità rispondono all'esigenza di avere un markup finalmente pulito e controllato. Ora, affronteremo un'altra importante dimensione di questo linguaggio.

Gestione dei contenuti con XML

XML è una sorta di lingua franca. Non ha nulla dei linguaggi proprietari. Un documento XML è semplicemente un file di testo e come tale leggibile su qualunque macchina. Si capisce, dunque, come si riveli uno strumento potentissimo in applicazioni centrate sullo scambio di informazioni e sulla distribuzione di contenuti su vari supporti. Inoltre, grazie alla possibilità di creare tag personalizzati, i documenti sono autoesplicativi. Ogni tag mi dice esattamente la funzione che svolge e il tipo di dati che definisce. La lettura di un buon manuale o della Guida XML (<http://xml.html.it/guide/leggi/58/guida-xml-di-base/>) potrà chiarire questo e altri aspetti. Ora passiamo alla pratica e torniamo al nostro blog.

I dati che intendiamo conservare in XML sono i post che quotidianamente arricchiscono la nostra cronaca digitale. Il motivo fondamentale per cui operiamo questa scelta è semplice: vogliamo tenere i contenuti separati dal resto. In questo modo, le operazioni di aggiornamento e archiviazione dei post saranno estremamente semplificate e volendo potremmo facilmente distribuirli su sistemi alternativi.

La struttura tipo di un post per X-blog è molto semplice. Ciascuno di essi avrà:

1. **un titolo**
2. **la data di invio**
3. **un testo**

4. eventuali link nel testo

Nel file "blog.html" listato 3

(http://html.it/guide/esempi/guida_xhtml/listato3.html - vedi Appendice C) abbiamo visto come questo schema si traduce in una struttura:

Sezione dei post

```
<div id="post">
  <h2>08/03/2002</h2>
  <h1>Titolo 1</h1>
  <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed diam nonummy
nibh euismod tincidunt ut laoreet dolore magna
aliquam erat volutpat.....</p>
```

Nel listato 6 (http://html.it/guide/esempi/guida_xhtml/listato6.html), invece, abbiamo riportato il codice del documento base di X-blog. Copiate, incollate e salvate come "**news.xml**". Aprendo il file con Internet Explorer apprezzerete al meglio la struttura ad albero del documento, con i singoli nodi strutturali contenuti all'interno dell'elemento radice. Qualche nota di commento:

1. il documento inizia con la dichiarazione XML
2. la seconda dichiarazione serve a incorporare un foglio di stile XSL (ne parleremo più avanti)
3. l'elemento radice è <news> che racchiude tutti i post
4. il tag <post> definisce un singolo aggiornamento del blog
5. <post> ha tre elementi figlio: <titolo>, <data>, <testo>
6. all'interno dell'elemento <testo> è possibile avere un altro nodo, <link> (per collegamenti a siti esterni citati nel testo)

Presentare XML

Come è evidente, il documento "news.xml" non contiene alcuna informazione su come verrà presentato. Non è una limitazione. È così che deve essere. Per effettuare la formattazione di un documento XML in vista della sua presentazione si possono seguire due metodi.

Il primo non vi risulterà nuovo: si usano i CSS. Applicare un foglio di stile CSS ad un documento XML è infatti semplicissimo. Supponiamo di voler collegare a "news.xml" il foglio di stile "news.css". Dovremo aggiungere, subito dopo la dichiarazione XML, quest'istruzione:

```
<?xml version="1.0" ?>
```

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

Il secondo metodo è quello che andremo ad analizzare nei dettagli nelle prossime due lezioni. È basato su XSL (Extensible Stylesheet Language), un linguaggio ben più potente dei semplici CSS in grado di trasformare qualunque documento XML in vari formati: testo semplice, HTML, XHTML, persino PDF. Finalmente capirete come "blog.html" sia generato automaticamente.

17. Risorse XHTML e XML sul web

In questa sezione della guida forniamo utili riferimenti su XML e XHTML. È un universo in continua evoluzione, che cambierà sicuramente il modo di lavorare di chi opera sul Web. La nostra speranza è quella di aver stimolato la vostra curiosità.

XHTML

XHTML sul W3C (<http://www.w3.org/TR/xhtml1/>): sito di riferimento per novità, specifiche, aggiornamenti.

XHTML Guru (<http://www.xhtmlguru.com/articles.htm>): contiene introduzioni alle varie versioni del linguaggio, una sezione di link e libri, una piccola raccolta di FAQ.

XHTML School (<http://www.w3schools.com/xhtml/>): un buon tutorial introduttivo e un'eccellente reference sul sito di W3Schools

Startkabel (<http://xhtml.startkabel.nl/>): mini-portale con link ad articoli e risorse

XHTML.org (<http://www.xhtml.org/>): pagina con news e riferimenti su XHTML, comprese mailing-list e newsgroup sull'argomento.

Ottimi articoli su XHTML si trovano anche nei più importanti siti di web-design e nei siti su XML riportati qui sotto.

XML

XML sul W3C (<http://www.w3.org/XML/>): la voce del Consortium

XML Directory (<http://www.xmldir.com/directory.asp?dir=518&ob=5>): raccolta di link e risorse

XML School (<http://www.w3schools.com/xml/default.asp>): tutorial ed esempi da W3Schools

XML.com (<http://www.xml.com/index.csp>): il sito su XML legato alla casa editrice O'Reilly&Associates. Una garanzia di eccellenza.

XML Pitstop (<http://www.xmlpitstop.com/>): sito ricchissimo sotto ogni aspetto. Incomparabile la raccolta di modelli di fogli XSLT.

XML/XSL Portal (<http://www.bayes.co.uk/xml/index.xml>): non solo è interamente costruito in XML e XSL, ma fornisce una serie incredibile di trucchi e risorse.

Perfect XML (<http://www.perfectxml.com/>): ottimo e ricco. La cosa migliore è la raccolta di capitoli di esempio tratti dai migliori manuali sui linguaggi legati a XML.

18. Libri su XHTML

I riferimenti sono ai libri ritenuti più validi dall'autore di questa guida.

In italiano

Tra i libri in italiano o tradotti in italiano su XHTML solo uno è davvero eccellente:

Molly E. Holzschlag, *XML, HTML, XHTML Magic*, Addison Wesley, 2002

Un ottima sezione su XHTML si trova anche nel manualone di Steven Holzner suXML:

Steven Holzner, *XML Tutto & Oltre*, Apogeo, 2001

In inglese

Steven Holzner, *XHTML Black Book*, Coriolis Group, 2000

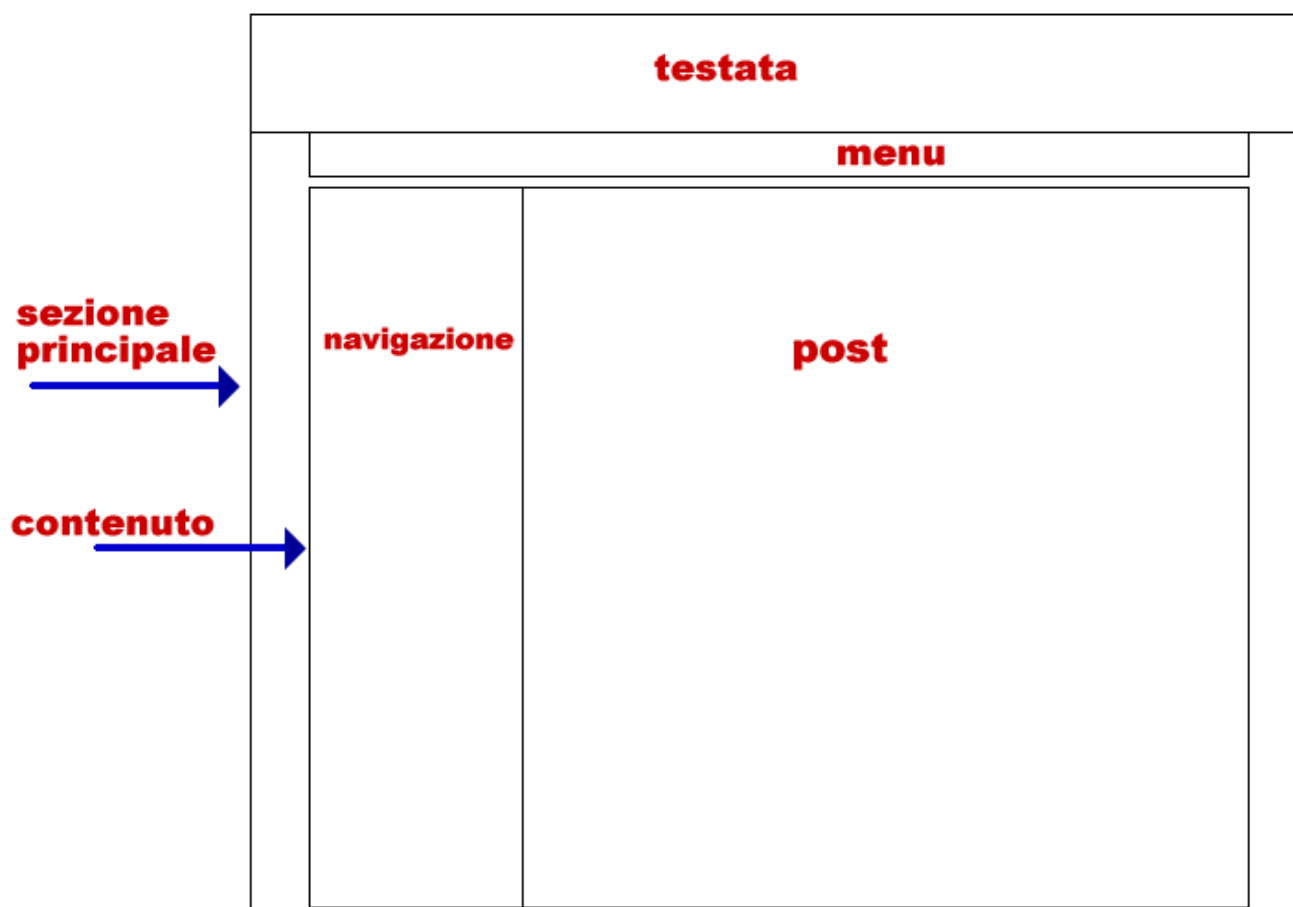
AA.VV., *Beginning XHTML*, Wrox Press, 2000

Chuck Musciano & Bill Kennedy, *HTML & XHTML : The Definitive Guide*, O'Really, 2000

Michael Morrison & Dick Oliver, *Sams Teach Yourself HTML and XHTML in 24 Hours*, Sams, 2001

Altri volumi li potete trovare nella sezione «libri (errore404.html)» di HTML.it.

Appendice A



Appendice B



Appendice C

Listato 3 - Codice del documento "blog.html"

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<title>X-blog</title>
<link type="text/css" rel="stylesheet" href="mainstyle.css" />
</head>
<body>
<div id="sezprinc">
<div id="testata">
</div>

<div id="menu">
<p><a href="#">home page</a> - <a href="#">about</a> - <a href="#">archivi</a>
- <a href="#">contatti</a> ----- <a href="#">e-mail me</a></p>
</div>

<div id="contenuto">

<div id="navigazione">
<h1>sezione 1</h1>
<p><a href="#">link 1</a></p>
<p><a href="#">link 2</a></p>
<p><a href="#">link 3</a></p>
<h1>sezione 2</h1>
<p><a href="#">link 1</a></p>
<p><a href="#">link 2</a></p>
<p><a href="#">link 3</a></p>
</div>

<div id="post">
<h1>Titolo 1</h1>
<h2>05/03/2002</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy
nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut
laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis
nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea
commodo consequat.</p>
<h1>Titolo 2</h1>
<h2>04/03/2002</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy
nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet,
consectetur adipiscing elit.</p>
</div>
</div>
```

```
</div>  
</body>  
</html>
```

Appendice D

Listato 4 - Codice commentato di "blog.html"

```
<!-- Escludiamo per compatibilità la dichiarazione XML -->
<!-- Inseriamo il DOCTYPE e usiamo la DTD Strict -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- L'elemento html contiene l'attributo XHTML -->
<html xmlns="http://www.w3.org/1999/xhtml">

<!-- Inizia la testata del documento -->
<head>
<!-- Dichiariamo con un meta-tag la codifica dei caratteri -->
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<!-- Diamo un titolo al documento: è obbligatorio -->
<title>X-blog</title>
<!-- Colleghiamo un foglio di stile esterno -->
<link type="text/css" rel="stylesheet" href="mainstyle.css" />
</head>
<!-- Fine della testata -->

<!-- Inizia il corpo del documento -->
<body>
<!-- Definiamo con l'elemento div le sezioni del documento badando al corretto
annidamento -->
<!-- Il div "sezprinc" contiene tutti gli altri. il suo tag di chiusura è in
fondo al documento-->
<div id="sezprinc">

<!-- Inizia la sezione "testata".-->
<div id="testata"></div>
<!-- Fine della sezione "testata" -->

<!-- Inizia la sezione "menu" -->
<div id="menu">
<p><a href="#">home page</a> - <a href="#">about</a> - <a href="#">archivi</a>
- <a href="#">contatti</a> ----- <a href="#">e-m@il me</a></p>
</div>
<!-- Fine della sezione "menu" -->

<!-- Inizia la sezione "contenuto". Essa contiene altre due sezioni, il tag di
chiusura è alla fine -->
<div id="contenuto">

<!-- Inizia la sezione "navigazione" -->
<div id="navigazione">
<!-- Con l'elemento h1 identifico il titolo della sezione -->
<h1>sezione 1</h1>
<!-- Con p iniziano i link veri e propri -->
<p><a href="#">link 1</a></p>
<p><a href="#">link 2</a></p>
<p><a href="#">link 3</a></p>
<h1>sezione 2</h1>
```

```

<p><a href="#">link 1</a></p>
<p><a href="#">link 2</a></p>
<p><a href="#">link 3</a></p>
</div>
<!-- Fine della sezione "navigazione" -->

<!-- Inizia la sezione "post"-->
<div id="post">
<h1>Titolo 1</h1>
<h2>05/03/2002</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy
nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud
exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.</p>
<h1>Titolo 2</h1>
<h2>04/03/2002</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy
nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet,
consectetur adipiscing elit.</p>
</div>
<!-- Fine della sezione "post"-->
</div> <!-- Questo tag chiude la sezione "contenuto" -->
</div> <!-- Questo tag chiude la sezione "sezprinc" -->
</body>
</html>

```

Appendice E

Listato 5 - Codice di "mainstyle.css"

```
/* Stili generici per l'elemento body */
body {
background : #FFFFFF;
color : #FFFFFF;
font : 12px Verdana, Geneva, Arial, Helvetica, sans-serif;
margin : 0px;
padding-bottom : 0px;
padding-left : 0px;
padding-right : 0px;
padding-top : 0px;
text-align : center;
}

/* Stile della sezione "sezprinc" */

#sezprinc {
background: #FFFFFF;
border-bottom : 1px solid black;
border-left : 1px solid black;
border-right : 1px solid black;
border-top : 1px solid black;
color : #000000;
margin : 20px auto;
text-align : left;
width : 574px;
}

/* Stile della sezione "testata" */

#testata {
background : #336699;
border-bottom : 1px solid black;
height : 75px;
margin : 0px;
padding-bottom : 1px;
padding-left : 26px;
padding-right : 26px;
padding-top : 0px;
}

/* Stile della sezione "contenuto" */

#contenuto {
border-bottom : #CCCCCC 0px dotted;
border-left : #CCCCCC 0px dotted;
border-right : #CCCCCC 0px dotted;
border-top : #CCCCCC 0px dotted;
margin : 0px 27px;
padding-bottom : 5px;
padding-left : 0px;
padding-right : 0px;
}
```

```
padding-top : 6px;
voice-family : inherit;
width : 512px;

}

/* Stile della sezione navigazione */

#navigazione {
background : #336699;
border-bottom : #CCCCCC 0px;
border-left : #CCCCCC 0px;
border-right : #CCCCCC 0px dotted;
border-top : #CCCCCC 0px;
color : #000000;
float : left;
padding-bottom : 0px;
padding-left : 0px;
padding-right : 0px;
padding-top : 0px;
voice-family : inherit;
width : 140px;
min-height : 500px;
}

/* Stili dei link della sezione "navigazione" */

#navigazione a {
background : #336699;
color : #99FFFF;
}

#navigazione a:hover {
background : #336699;
color : #DC143C;
}

#navigazione a:visited {
background : #336699;
color : #99FFFF;
}

/* Selettore contestuale. Specifica come appare l'elemento h1 all'interno della
sezione "navigazione" */

#navigazione h1 {
background : #336699;
border-bottom : 1px dotted #808080;
border-left : 1px dotted #808080;
border-right : 1px dotted #808080;
border-top : 1px dotted #808080;
color : white;
font : bold 13px Tahoma, Verdana, Helvetica, sans-serif;
margin-bottom : 3px;
margin-left : 5px;
margin-right : 5px;
margin-top : 15px;
padding-bottom : 2px;
}
```

```

text-align : center;
}

/* Selettore contestuale. Specifica come appare l'elemento p all'interno della
sezione "navigazione" */

#navigazione p {
background : #336699;
color : Black;
font : 11px Verdana, Geneva, Arial, Helvetica, sans-serif;
margin : 5px 5px;
}

/* Stile della sezione "menu" */
#menu {
border-bottom : black 1px solid;
border-left :black 1px solid;
border-right : black 1px solid;
border-top : black 0px solid;
margin : 0px 26px 1px;
padding-bottom : 5px;
padding-left : 2px;
padding-right : 0px;
padding-top : 4px;
width : 512px;
background : #FFCC00;
}

/* Stili dei link della sezione "menu" */
#menu a {
background : #FFCC00;
color : #CC0000;
}

#menu a:visited {
background : #FFCC00;
color : #CC0000;
}

/* Selettore contestuale. Specifica come appare l'elemento p all'interno della
sezione "menu" */
#menu p {
background : #FFCC00;
color : black;
font : bold 11px Verdana, Geneva, Arial, Helvetica, sans-serif;
text-align : center;
}

/* Stile della sezione "post" */
#post {
background-color : #FFFFFF;
border-bottom : #CCCCCC 0px;
border-left : #CCCCCC 0px dotted;
border-right : #CCCCCC 0px;
border-top : #CCCCCC 0px;
color : #000000;
margin : 0px 0px 0px 142px;
}

```

```
padding-bottom : 10px;
padding-left : 16px;
padding-right : 0px;
padding-top : 0px;
voice-family : inherit;
width : 350px;

}

/* Selettore contestuale. Specifica come appare l'elemento h1 all'interno della
sezione "post" */
#post h1 {
background : white;
color : #191970;
font : bold 13px Tahoma, Verdana, Helvetica, sans-serif;
margin : 5px;

}

/* Selettore contestuale. Specifica come appare l'elemento h2 all'interno della
sezione "post" */
#post h2 {
background-color : #FFFFFF;
color : #CC0000;
font : bold 10px Tahoma, Verdana, Helvetica, sans-serif;
margin : 5px;
}

/* Selettore contestuale. Specifica come appare l'elemento p all'interno della
sezione "post" */
#post p {
background-color : #FFFFFF;
border-bottom : 1px dotted black;
color : #000000;
font : 11px Verdana, Geneva, Arial, Helvetica, sans-serif;
line-height : 1.5em;
margin : 5px;
padding-bottom : 4px;
}

```

Appendice F

Figura 1

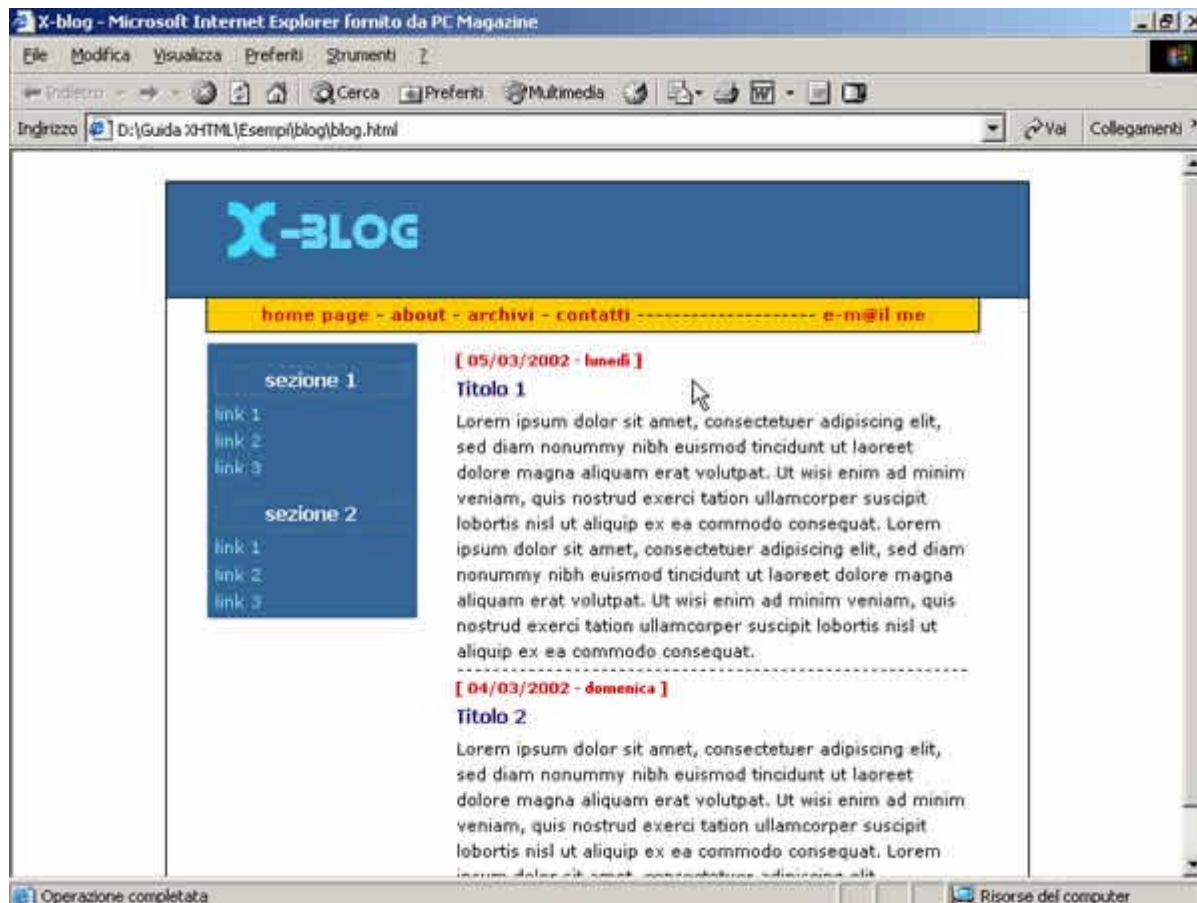


Figura 2

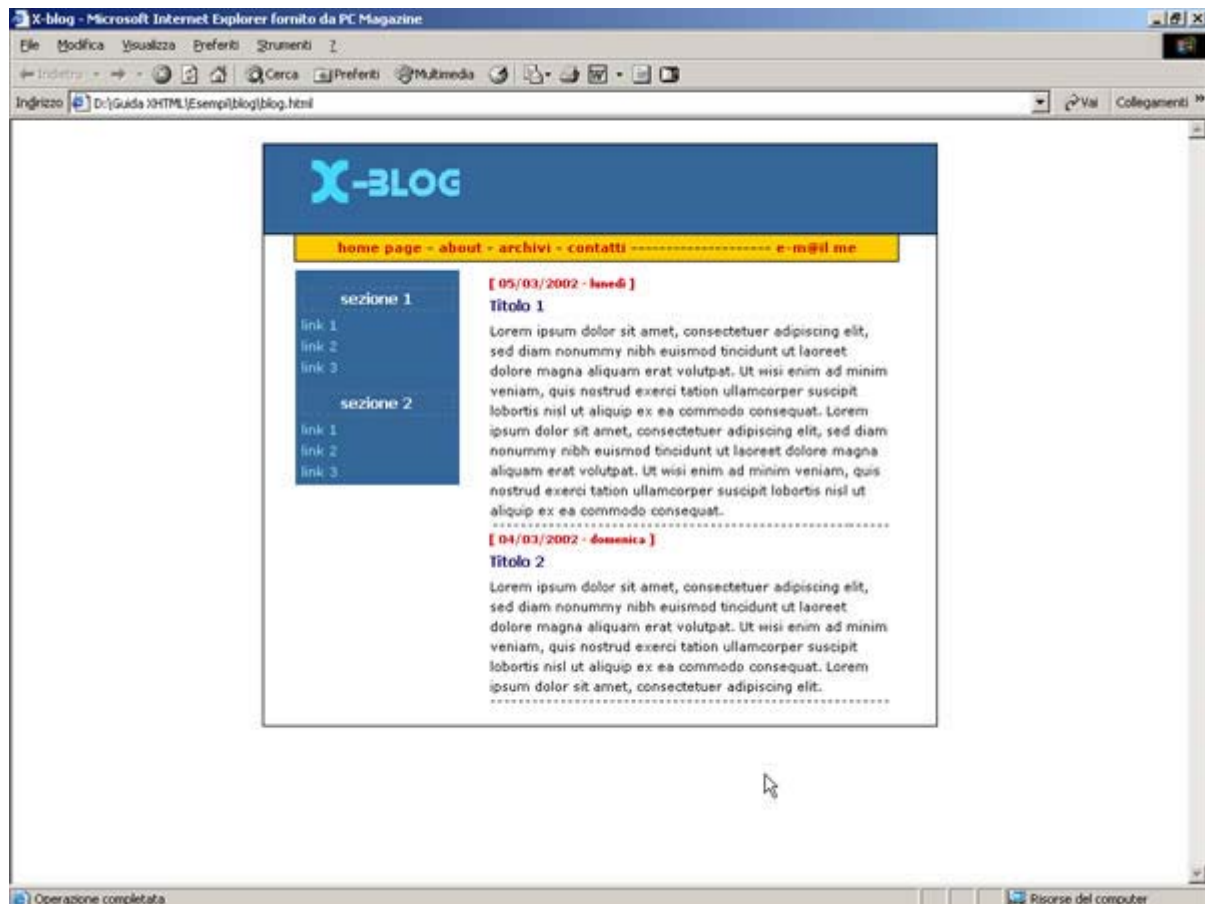


Figura 3

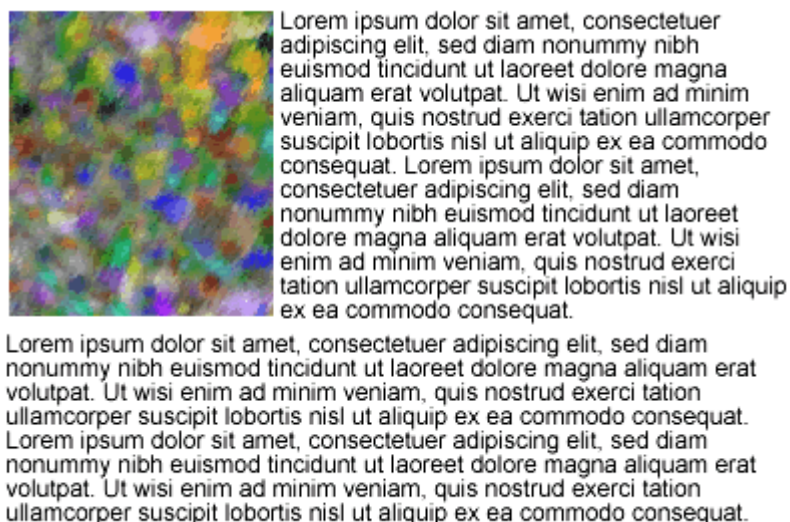


Figura 4

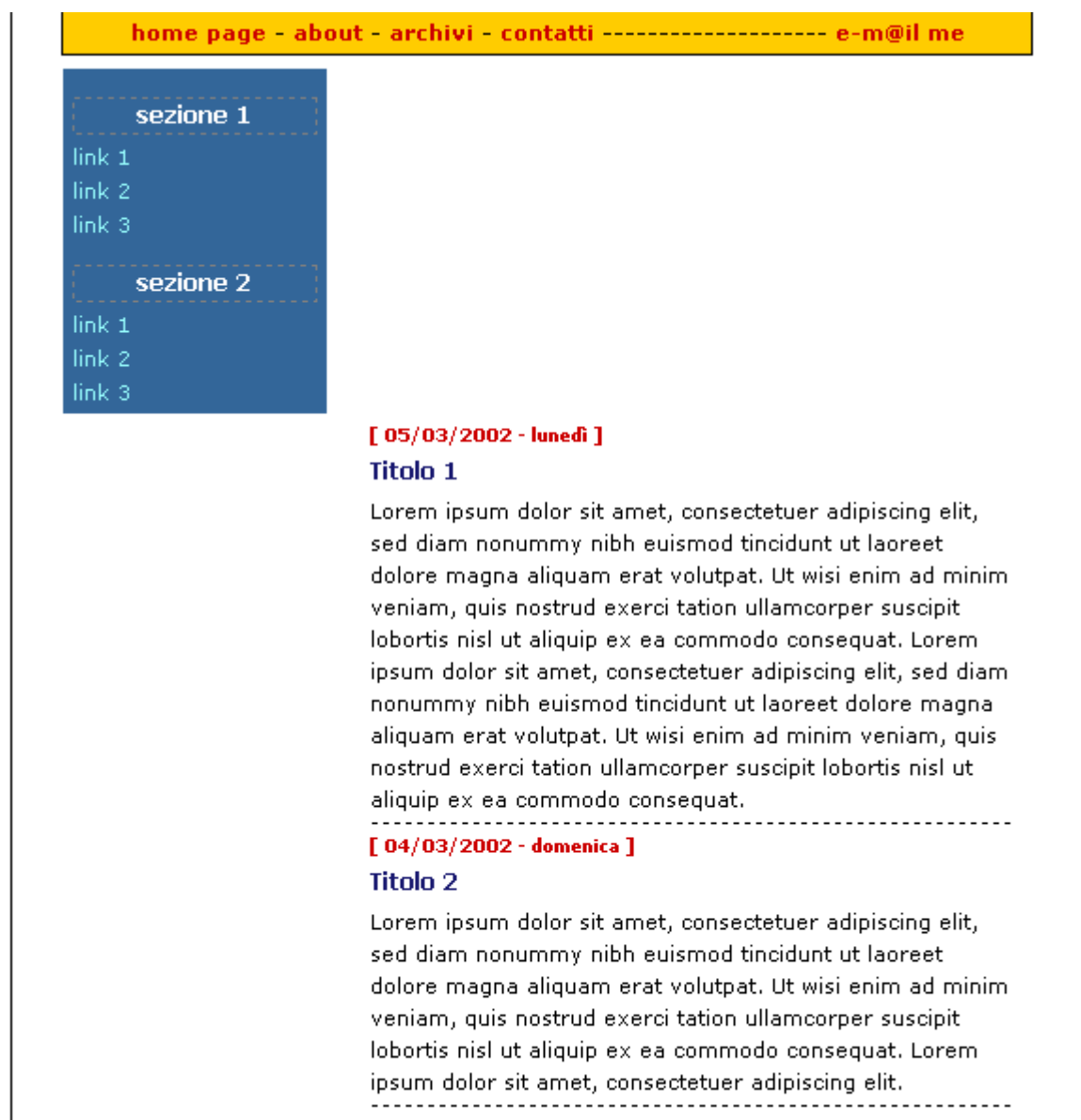


Figura 5

