

Giorno I

I linguaggi di markup

In questo primo capitolo, dopo un accenno alla natura del World Wide Web (il Web) e al ruolo del World Wide Web Consortium (W3C), affronteremo questi argomenti:

- Il ruolo che XML avrà nel mondo dell'e-business
- Alcune limitazioni dell'Hypertext Markup Language (Html), il linguaggio oggi predominante
- Che cos'è Sgml (Standard Generalized Markup Language) e quali sono i rapporti fra Sgml, Xml e Html
- Sette caratteristiche di Xml
- Come iniziare a creare documenti Xml con elementi di marcatura auto-descrittivi

Il Web è un fenomeno rivoluzionario

In pochi anni, il Web è diventato la biblioteca del mondo. Al cuore di questo archivio globale sta il potente concetto di "ipertesto", termine coniato nel 1965 da Ted Nelson per descrivere un modo per collegare flussi di dati da computer diversi. Ted Nelson presentò le sue idee a New York, nel corso della ventesima National Conference della Association for Computing Machinery, in un saggio intitolato *A File Structure for the Complex, the Changing, and the Indeterminate*. (Per maggiori informazioni su Ted Nelson e il suo progetto Xanadu: <http://jefferson.village.virginia.edu/elab/hf10155.html>.)

Verso la fine degli anni Sessanta Doug Engelbart ha creato un prototipo di "oNLine System" (NLS), di cui facevano parte la navigazione ipertestuale, la modifica dei collegamenti, la posta elettronica e altri componenti che oggi associamo al Web.

Ma il Web è diventato molto di più di questo. Il ricco tessuto di presentazioni multimediali con video, grafica, suoni, audio ha superato di gran lunga gli

obiettivi originali di Tim Berners-Lee, inventore riconosciuto del Web, che nel 1989 propose

un sistema universale di informazioni fra loro collegati, in cui la generalità e la portabilità sono molto più importanti della bella grafica (da <http://www.w3.org/History/1989/proposa1.html>).

Nel 1980 Berners-Lee iniziò con un programma chiamato “Enquire-Within-Upon-Everything”, pensando a una miriade di collegamenti ipertestuali che fornissero interconnessioni virtuali fra archivi di informazioni su computer diversi. Più tardi (ottobre 1990) cominciò a definire la sua creazione World Wide Web. Benché abbia cominciato con visioni piuttosto grandiose ed eteree (il Web avrebbe dovuto contenere tutto l’universo delle informazioni del genere umano), il metodo seguito per poter fruire di quella conoscenza era molto pratico, se non addirittura semplicistico.

La metodologia che ha sviluppato mentre lavorava al CERN (Conseil Européen pour la Recherche Nucléaire) doveva permettere lo scambio di documenti attraverso il TCP/IP (Transmission Control Protocol/Internet Protocol). Berners-Lee proponeva:

- Un nuovo metodo universale di indirizzamento dei documenti in Internet, l’URL (Universal Resource Locator)
- Un nuovo protocollo TCP/IP, che poi si sarebbe chiamato http (Hypertext Transfer Protocol)
- Un nuovo linguaggio per la descrizione dei documenti, l’HTML (Hypertext markup Language)

Il tutto sotto forma di un insieme funzionale di strumenti software (browser) in origine programmati sulla piattaforma NeXT in Objective C e poi portati in C per poter girare su altre piattaforme.



Potete leggere un resoconto, conciso ma molto interessante, degli sviluppi che hanno caratterizzato questa rivoluzione in un articolo dal titolo “A Little History of the World Wide Web”, in rete all’indirizzo <http://www.w3.org/History.html>.

Questo ci dice che, dietro i banner pieni di colori, le animazioni dal movimento fluido e i flussi audio/video in tempo reale che caratterizzano il Web di oggi stanno raccolte di semplici documenti di testo (principalmente in HTML e in altre tecnologie collegate), che possono essere trasferiti in modo efficiente e affidabile da un computer all’altro via Internet.

A che cosa serve un altro linguaggio di markup?

In fondo, sembrerebbe che la rivoluzione del Web sia andata avanti abbastanza in fretta basandosi semplicemente sulla marcatura con HTML. Con questo linguaggio, certo, un programmatore può creare semplici collegamenti ipertestuali che funzionano. Da una pagina HTML, è possibile rendere attivo un collegamento (per esempio, quando l'utente fa clic con il mouse su un ancoraggio ipertestuale in HTML presente su una pagina) per scaricare il contenuto di un'altra pagina.

L'HTML offre una portabilità che altre tecnologie, per la loro natura, non possono garantire. La portabilità è uno dei molti obiettivi primari del progetto profondo del Web.

La curva di apprendimento dell'HTML è relativamente poco ripida: un po' tutti, dai bambini che esplorano hobby e interessi comuni, fino ai nonni che vogliono condividere le immagini dei nipotini possono pubblicare sul Web con pochissima fatica. Editor e strumenti di sviluppo HTML con interfaccia grafica (GUI, Graphical User Interface) si trovano a basso costo, o addirittura gratuitamente scaricabili dal Web. Le interfacce browser sono a bassissimo costo, potenti e facili da usare.

Ma se l'HTML offre tutte queste cose, perché mai imparare XML?

Uno dei motivi è che HTML ha limiti seri. Quello che all'inizio era un linguaggio di marcatura dei dati è stato piegato e adattato come strumento di presentazione Web; in questo ruolo, però, è debole, offre scarso controllo sulla collocazione e la gestione degli "spazi bianchi" sulla pagina, per esempio. Ha problemi con la spaziatura dei caratteri, la giustificazione, la divisione in sillabe, e in HTML è problematica la manipolazione di dati in colonne (in stile giornale).

Quando l'HTML è stato creato, non c'era l'intenzione di includervi aspetti di stile; anche marcatori per stili comuni come il corsivo (<I>) e il grassetto () sono stati aggiunti solo in seguito. Questi stili sono usati normalmente per l'evidenziazione, e l'intento era di includere nel linguaggio un marcatore di evidenziazione che sarebbe stato interpretato opportunamente da un processo diverso, se e quando si fosse reso necessario applicare gli stili. Questa separazione del contenuto dallo stile è caratteristica dell'XML, un linguaggio centrato sui metadati che definisce marcatori per i dati e non per lo stile di presentazione. Le questioni di stile sono trattate separatamente dai dati. Vedrete come si aggiunge uno stile alla marcatura dei dati in vari modi: nel Giorno 14, vedremo come aggiungere i CSS (*cascading stylesheets*) ai documenti XML, più o meno come si può aggiungerli a una pagina HTML 4.01. Nel Giorno 15, studieremo XSL (eXtensible Stylesheet Language) e le sue ricche componenti di formattazione (FO, Formatting Object). Nel Capitolo 16 infine useremo le trasformazioni di XSL (XSLT, eXtensible Stylesheet Language Transformations) per trasformare i documenti XML in HTML.

Come strumento di marcatura dei dati, l'HTML non è abbastanza flessibile. Prevede solo un numero limitato di marcatori predefiniti, e non consente di andare oltre quell'insieme prestabilito. Con lo sviluppo di nuove versioni del linguaggio sono state aggiunte nuove funzionalità, ma in genere con poca coerenza. Spesso sono stati introdotti nuovi marcatori che sono accettati solo da un particolare browser, infrangendo così le regole cardinali del markup per il Web, che dicono: tutte le applicazioni debbono interoperare e la presentazione dei dati deve essere indipendente dalla piattaforma.

Il W3C non vuole che vengano rilasciate altre versioni di HTML; quella attuale (4.01) sarà l'ultima. Forse avete sentito parlare di XHTML e vi siete domandati che cosa fosse; ebbene, è HTML riformulato come applicazione XML. Scoprirete le regole dell'XML in questa e nella prossima lezione, e in particolare vedrete che HTML può essere XML, se segue le sue regole sintattiche. In effetti, XHTML 1.0 e HTML 4.01 sono molto simili, ma gli elementi dell'uno e quelli dell'altro sono vincolati a spazi di nomi (cioè collezioni di elementi accettabili) diversi. Degli spazi di nomi o *namespace* parleremo nel Giorno 8.

HTML non supporta bene la *modularità* o *riusabilità* del codice che è tipica invece di altri linguaggi. In linguaggi a oggetti come C#, C++, Visual Basic e Java, ci sono le classi che permettono la condivisione e il reimpiego di oggetti e metodi in più programmi. Nel Giorno 18 vedremo Xinclude, una nuova tecnologia che promette di offrire funzioni di riusabilità del codice ai programmatori XML. Nel Giorno 10 esploreremo un concetto simile, utilizzando una tecnica di incorporamento che è disponibile per le applicazioni XLink-*aware*. HTML invece non permette nessuna di queste cose.

HTML presenta incoerenze che permettono di scrivere e diffondere codice che si può definire "cattivo". XML, con una applicazione coerente delle regole sintattiche, determina "pratiche di buona programmazione". Di molte fra queste regole parleremo in questa e nella prossima lezione, ma possiamo fare un esempio. Alcune versioni di HTML non richiedono sempre la presenza di un marcatore di fine a indicare il completamento di certe forme di markup, e i browser hanno dovuto accettare gli insiemi di regole misti che sono diventati caratteristici di HTML. Per esempio, in HTML un paragrafo inizia con un marcatore <P>, e nelle prime versioni di HTML non era necessario terminare un paragrafo con un corrispondente marcatore </P>. I browser dovevano "disattivare" una istruzione di marcatura di paragrafo determinando in quale punto del flusso di testo la marcatura fosse completa, il che non era sempre dove l'autore del documento voleva concludere il paragrafo. Lo standard più recente, HTML 4.01, richiede che un marcatore di paragrafo (e in effetti tutti i marcatori) abbiano un corrispondente marcatore di terminazione. I browser però supportano ancora le centinaia di milioni di siti Web che includono HTML non ben formato, e perciò non applicano strettamente né convalidano gli standard HTML 4.01. Parleremo di "buona formazione" dei documenti XML nel Giorno 2 e vedremo che i vincoli sintattici di XML sono rigorosi e applicati in modo rigido. Le medesime regole però offrono ai programmatori un livello di flessibilità che è del tutto assente in HTML.

Sette cose da sapere di XML

L'acronimo XML sta per eXtensible Markup Language, ma, in un certo senso, il nome è fuorviante: dovrebbe essere eXtensible Meta Language, perché si tratta di uno strumento standardizzato, ma estremamente flessibile, per creare altri linguaggi. Per essere forse ancora più precisi, XML permette la creazione di dialetti di linguaggi che seguono regole precise e rigorose per la struttura, la sintassi e la semantica, come stabilito dal W3C. Imparerete alcune di queste regole in questa lezione, e vedrete l'importanza di altre nel seguito della lettura e nello svolgimento degli esercizi.



Il World Wide Web Consortium (W3C) è l'organizzazione ufficiale che ha il compito di promuovere lo sviluppo di tecnologie interoperabili (cioè specifiche, direttive, software, strumenti). È stato fondato da Tim Berners-Lee, inventore riconosciuto del Web, come mezzo per condividere informazioni in modo coerente.



Il sito Web del W3C (<http://www.w3.org>) è una delle migliori fonti di informazione su XML e molte tecnologie correlate.

In questa sezione, vedremo sette affermazioni che caratterizzano XML. Il W3C ha una sezione che descrive l'XML in dieci punti (<http://www.w3.org/XML/1999/XML-in-10-points>), ma tre sono solo dei “segnaposto” per caratteristiche future. Ecco i sette punti fondamentali:

1. XML offre un metodo per inserire dati strutturati in un file di testo.
2. XML assomiglia un po' a HTML.
3. XML è leggibile dalle macchine, ma comprensibile agli esseri umani.
4. XML è costituito da una famiglia di tecnologie.
5. XML è prolisso.
6. XML è relativamente nuovo, ma ha radici di tutto rispetto.
7. XML non richiede licenza, è indipendente dalla piattaforma, ed è ben supportato.

I. XML offre un metodo per inserire dati strutturati in un file di testo

Quando si pensa ai programmi tradizionalmente usati per creare, manipolare e mantenere dati, normalmente i dati sono memorizzati in qualche formato binario, spesso proprietario. Word processor, database e fogli elettronici commerciali sono magari in grado di produrre dati in forma testuale, ma sono stati progettati per fare un uso ottimale di formati binari, diversi da un tipo di programma all'altro, e spesso da un produttore all'altro. Per questo il processo di con-

versione dei dati è nel migliore dei casi contorto, e qualche volta addirittura la natura proprietaria di certi formati lo rende impossibile.

Forse vi siete trovati in difficoltà di questo genere se vi è capitato, per esempio, di scambiare un documento creato in una particolare versione di Microsoft Word con qualcuno che usa Corel WordPerfect. Un documento salvato nel formato binario nativo di un prodotto non è leggibile dall'altro, e viceversa. Per scambiare documenti, dovete eseguire dei passaggi in più, e salvare il documento in un formato che sia accessibile a entrambi i prodotti. Spesso, con questa operazione, si finisce per perdere qualche caratteristica di formattazione ricca strettamente legata al pacchetto usato in origine.

XML permette la memorizzazione dei dati in forma di testo semplice: qualsiasi applicazione (o qualsiasi essere umano) che possa leggere un file di testo può leggere un documento XML. Non è necessario avere il programma di origine per accedere ai dati. In questo modo, risolvere un problema in un ambiente di sistema informativo è semplice: basta lanciare un editor di testo qualsiasi per rivedere e modificare il documento. La maggior parte dei sistemi operativi fornisce almeno un editor di testo senza costi aggiuntivi; altri editor, perfettamente adeguati a questo scopo, sono liberamente disponibili su Internet.

Come avrete notato, il primo punto dice che "XML offre un metodo per inserire dati *strutturati* in un file di testo". XML è un insieme di regole per creare formati di testo facili da generare e facilmente elaborabili da un computer. I file di testo risultanti sono strutturati, in modo tale da essere

- Privi di ambiguità
- Estendibili
- Indipendenti dalla piattaforma



La "X" in XML sta per Extensible: estendibilità significa che il linguaggio può essere ampliato per soddisfare esigenze specifiche. XML non è basato su un insieme finito di marcatori, e si possono creare marcatori descrittivi adatti alle proprie necessità.

Si può usare qualsiasi semplice editor di testi per creare e manipolare i documenti XML, ma nel corso dei prossimi capitoli parleremo di parecchi strumenti specializzati (e daremo le informazioni per ottenerli).

Per convenzione, i file XML generici hanno una estensione di file .xml: per esempio, `miodocumento.xml`. I dialetti specializzati creati con XML spesso usano un'estensione di file specifica, per esempio:

- .xsl indica un file Extensible Stylesheet Language
- .xsd indica un file di Extensible Schema Definition
- .xdr indica un file di XML Data Reduced Schema
- .mml indica un file di Mathematical Markup Language (MathML)
- .cdf indica un file di Channel Definition Format

2. XML assomiglia un po' a HTML

Ci sono molte somiglianze fra XML e HTML. Se conoscete già quest'ultimo, conoscete già alcune delle regole sintattiche di XML. Prendiamo per esempio il frammento del Listato 1.1: è un esempio di HTML o di XML?

Listato 1.1 Un esempio semplice di marcatura.

```
<P>
  Ecco un brano di testo
  <EM>marcato</EM> per
  la presentazione Web.
</P>
```

La risposta è “di tutti e due”. In effetti, questo è un esempio di un frammento ben formato tanto di HTML come di XML: segue le regole comuni a entrambi i linguaggi.

XML, però, è stato progettato per ovviare ai limiti e alle mancanze di HTML. L'autore del documento può definire elementi XML: non esiste un insieme di marcatori predefinito come in HTML. Dato che, come autori, potete definire i vostri marcatori, potete scegliere nomi più significativi. Invece di marcare un paragrafo con un marcatore `<P>`, potete scegliere di usare `<paragrafo>` o `<para>`. Forse avrà più senso per voi marcare paragrafi diversi di un documento sulla base del loro significato o della loro importanza. Per esempio, invece di avere solo un elemento `<paragrafo>`, potete decidere di averne diversi:

- `<introduzione>`
- `<sommario>`
- `<informazioni_commerciali>`
- `<indirizzo>`
- `<descrizione>`
- `<ecc>`

Per quanto riguarda le somiglianze, quindi, potete vedere che sia HTML sia XML hanno marcatori racchiusi fra i simboli `<` e `>`. Nei linguaggi di markup, un elemento semplice è formato da un marcatore di inizio, da un contenuto elementare e da un marcatore di fine, come si vede nel Listato 1.2.

Listato 1.2 Un documento XML con un solo elemento.

```
1: <titolo>XML Guida completa</titolo>
```

Questo elemento singolo costituisce un documento XML ben formato, per quanto semplice. Il nome dell'elemento è `titolo`, e compare fra il marcatore di inizio (`<titolo>`) e quello di fine (`</titolo>`) dell'elemento. Tutta la stringa (`<titolo>XML Guida completa</titolo>`), dalla parentesi angolare di apertura

del marcatore di inizio fino alla parentesi angolare chiusa del marcatore di fine, costituisce l'*elemento*. Il dato testuale contenuto fra i marcatori è il *contenuto dell'elemento*, o semplicemente il *contenuto*. L'espressione *contenuto dell'elemento* (*element content*) è usata anche da alcuni dei linguaggi per gli schemi, di cui parleremo più avanti.



Tutti gli elementi in XML debbono essere terminati. Gli elementi non vuoti debbono avere un marcatore di inizio e un marcatore di fine. Gli elementi vuoti debbono essere terminati correttamente.

Gli elementi XML nel markup sono i mattoni fondamentali del linguaggio, come i nomi nelle lingue parlate; sono gli oggetti, i luoghi, le cose. XML può avere anche attributi, simili a quelli dell'HTML, che servono a modificare o specificare gli elementi, come nelle lingue parlate gli aggettivi modificano i nomi. Gli attributi, se presenti, devono trovarsi sempre nel marcatore di inizio dell'elemento. Il Listato 1.3 presenta un elemento un poco più complesso, che incorpora attributi e nel contenuto dell'elemento presenta altri elementi. Quando un elemento contiene elementi figli, questi ultimi si dicono *annidati*.

Listato 1.3 Un documento XML con elementi figli annidati e attributi.

```
1: <conto tipo="risparmio" valuta="Dollari Usa">
2:     <nome>Smith</nome>
3:     <saldo>34,576.89</saldo>
4: </conto>
```

ANALISI

Le righe 1-2 sono tutto l'elemento *conto*. Due attributi modificano l'elemento *conto*, nello stesso modo in cui un aggettivo modifica un nome, cioè fornendo ulteriori informazioni sull'elemento. Gli attributi sono il tipo e la valuta, con i valori *risparmio* e *Dollari Usa*, rispettivamente. I valori degli attributi vengono sempre collocati fra apici singoli o doppi. L'elemento *conto* contiene solamente contenuto di elemento (righe 2-3): in altre parole, l'elemento *conto* non contiene dati carattere o testo, bensì elementi figli, cioè *nome* e *saldo* (righe 2-3). Ciascun elemento (*nome*, *saldo* e *conto*) ha un marcatore di inizio e un corrispondente marcatore di fine con una barra (/ , *slash* in inglese) nel marcatore di fine. Tutti gli elementi XML debbono essere terminati.



I valori degli attributi in XML debbono essere racchiusi fra apici singoli o doppi.

Qualche volta, decidere se usare un elemento o un attributo per caratterizzare certe informazioni è difficile. Se potete pensare che quelle informazioni operino come un aggettivo che modifica un certo nome, per esempio un conto *rispar-*

mio, o valuta *dollari*, probabilmente avete a che fare con attributi. Se invece le informazioni hanno un valore di dati reale, probabilmente è meglio marcarle come elemento. Gli elementi possono avere elementi figli; gli attributi no. Perciò, qualsiasi entità che sia un contenitore di altri elementi con tutta probabilità è un elemento. Queste non sono regole rigide; in effetti, non esistono regole. Per ogni singolo progetto di marcatura, dovrete effettuare un'analisi specifica, per stabilire che cosa funziona meglio, situazione per situazione.

Gli elementi XML possono contenere testo, altri elementi, una qualsiasi combinazione di testo e altri elementi, oppure essere vuoti. Un elemento vuoto è un elemento che non ha contenuto al di là di un marcatore di inizio e di un marcatore di fine. In HTML, `` è un esempio di elemento vuoto. La sintassi di un elemento vuoto è:

```
<nome_elemento />
```

Il Listato 1.4 mostra un esempio di elemento XML vuoto.

Listato 1.4 Un esempio di elemento XML vuoto.

```
1: <data mese='Settembre' anno='2002' />
```

L'elemento `data` è vuoto. Il terminatore (`/`) si trova infatti all'interno del marcatore di inizio: si tratta di una forma stenografica equivalente a `<data mese='Settembre' anno='2002'></data>`. Anche se le due forme sono funzionalmente equivalenti, è buona pratica di programmazione usare la forma compatta per gli elementi vuoti. Anche se questi elementi hanno attributi con valori, vengono comunque considerati vuoti, perché non sono presenti dati fra i marcatori di inizio e fine.

Notate che questo elemento, in particolare, ha un attributo. Com'è possibile che un elemento che contiene attributi con valori venga considerato vuoto? In fin dei conti, i valori degli attributi sono dati. Vuoto è un elemento che non marca contenuto. Il contenuto appare solo fra il marcatore di inizio e quello di fine. Gli attributi non sono considerati contenuto anche se forniscono informazioni qualitative sugli elementi. Nel Giorno 12, vedremo il Document Object Model (DOM), che offre il modo per accedere da programma a singole parti di un documento XML. Vedremo che i dati attributo occupano, rispetto ai dati contenuto, una parte diversa, cioè un "nodo" diverso nella struttura rappresentata dal documento XML.

Fin qui, abbiamo visto alcune somiglianze sintattiche fra HTML e XML: entrambi hanno elementi e attributi. Una delle differenze fondamentali è invece la possibilità di creare, in XML, propri elementi autodescrittivi. Confrontate i due esempi di codice nei Listati 1.5 e 1.6.

Listato 1.5 Un frammento di codice HTML.

```
<HTML>
  <H1>Fattura</H1>
  <P>Da: Devan Shepherd</P>
  <P>A: Sally Jones</P>
  <P>Data: 26 Luglio 2001</P>
  <P>Imponibile: $100.00</P>
  <P>Tasse: 21 %</P>
  <P>Totale: $121.00</P>
</HTML>
```

Listato 1.6 Un frammento confrontabile di codice XML.

```
<Fattura>
  <Da>Devan Shepherd</Da>
  <A>Sally Jones</A>
  <Data anno='2001' mese='Luglio' giorno='26' />
  <Imponibile valuta='USD'>$100.00</Imponibile>
  <TassaAliquota>21</TassaAliquota>
  <Totale valuta='USD'>121.00</Totale>
</Fattura>
```

Esaminando i due esempi, ponetevi queste domande:

- Quale documento ha più valore pertinente?
- Quale cattura le informazioni più utili per un'applicazione di elaborazione?
- Quale listato offre il maggior potenziale di utilizzazione?

XML, con i suoi elementi e attributi autodescrittivi, offre una marcatura più ricca. Se un'applicazione avesse bisogno dei dati di fatturazione, il frammento HTML nel Listato 1.5 non sarebbe in grado di fornire altro che una serie di paragrafi. Per differenziare il contenuto di un elemento `<P>` da quello di un altro, sarebbe necessaria una logica complessa di programmazione, di scripting o di *pattern-matching*. Il codice XML nel Listato 1.6, invece, offre componenti che hanno un significato, un po' come i diversi campi di un database "piatto". XML è centrato sui metadati, e conserva l'intelligenza e le informazioni utili sui dati che marca. Pensate alla struttura di un documento di markup. Un singolo elemento contiene tutti gli altri elementi del documento. Questo singolo elemento al livello più alto della struttura è il cosiddetto *elemento radice*. In una pagina HTML, l'elemento radice è sempre l'elemento HTML. In un documento XML, siete voi a decidere il nome dell'elemento radice. Nell'esempio del Listato 1,6, l'elemento radice è **Fattura**.

Un documento XML ben formato deve contenere un (e un solo) elemento radice, che contiene tutti gli altri.

Un'altra differenza fondamentale fra XML e HTML è che il secondo mescola contenuto e formattazione nello stesso flusso di markup. Per esempio, il marca-

tore <H1> in HTML serve a individuare una particolare stringa di dati come intestazione di livello 1, ma ha anche il compito di far sapere al browser che tutto ciò che è marcato in questo modo deve essere presentato con un carattere in corpo maggiore.

XML si basa sul presupposto che contenuto e aspetto (o stile) debbano essere tenuti separati dalla codifica di marcatura dei dati. XML si basa esclusivamente sui linguaggi dei fogli stile come Cascading Style Sheets (CSS) o eXtensible Style Sheet Language (XSL), per la presentazione dei documenti o per la loro trasformazione da una struttura a un'altra. Vedremo come applicare CSS e XSL per il rendering lato browser nei giorni 14 e 15. Nel Giorno 16, useremo le conoscenze acquisite per usare XSLT, il linguaggio di trasformazione di XSL, e convertire documenti XML in HTML entro script middleware per il trasferimento di dati XML via Web in modo indipendente dalla piattaforma.

3. XML è leggibile dalle macchine, ma comprensibile agli esseri umani

Come abbiamo visto, gli elementi autodescrittivi fanno sì che i documenti XML possano essere intuitivi per natura. La semantica effettiva dei dati viene conservata come intelligenza veicolata in XML insieme con contenuti e valori degli attributi. Comunque, XML è codice per computer, destinato a essere letto e usato da processori XML: nel Giorno 2, vedremo un tipo di processore – un *parser* o *analizzatore sintattico* – che interpreta un documento XML una riga alla volta.

4. XML è costituito da una famiglia di tecnologie

La suite XML comprende parecchie tecnologie importanti, elencate nella Tabella 1.1, con l'indicazione del giorno in cui sono trattate.

Tabella 1.1 Tecnologie XML

Giorno	Tecnologia	Descrizione
Giorni 1,2,3	XML Version 1.0	Raccomandazione tecnica per XML
Giorno 4	DTD	Document Type Definition (uno schema)
Giorno 5	XDR	XML Data Reduced (schema Microsoft)
Giorno 6	XSD	XML Schema Definition (schema W3C)
Giorno 8	Namespaces	Un metodo per qualificare i nomi di elementi e attributi
Giorno 9	XPath	XML Path Language
Giorno 10	XLink	XML Link Language
Giorno 11	XPointer	XML Pointer Language
Giorno 12	DOM	Document Object Model API

(segue)

Tabella I.1 Tecnologie XML *(continua)*

Giorno	Tecnologia	Descrizione
Giorno 13	SAX	Simple API for XML
Giorno 15	XSL	eXtensible Style Sheet Language
Giorno 15	XSL-FO	XSL Formatting Objects
Giorno 16	XSLT	XSL Transformation Language
Giorno 18	XInclude	Sintassi XML Include
Giorno 18	XBase	Sintassi XML Base URI

Leggendo la descrizione di queste tecnologie e sperimentando con gli esempi di codice, fate particolare attenzione allo stato W3C di ciascuna. Alcune di queste componenti sono ancora sotto forma di prima bozza di lavoro e nei mesi o negli anni a venire sono destinate a subire modifiche anche notevoli. I processori XML accettano anche altri membri della famiglia XML, in vario modo: la standardizzazione di molti di questi elementi è ancora in corso.

5. XML è prolisso

I file XML sono documenti di testo con delimitatori per la marcatura, perciò sono quasi sempre di dimensioni maggiori rispetto a file binari analoghi. Questa caratteristica è stata presa seriamente in considerazione nella fase di progettazione del linguaggio: i membri del W3C che avevano ricevuto l'incarico di creare lo standard hanno scelto di costruire un linguaggio prolisso, lasciando ampio spazio per la possibilità di estensioni e per l'autodescrittività di elementi e attributi. I progettisti di XML si sono resi conto che lo spazio di memoria è una risorsa che tende a diventare sempre meno costosa, perciò non hanno ritenuto che le dimensioni dei file fossero un problema; inoltre sono ormai ampiamente disponibili programmi di compressione a basso costo, spesso addirittura gratuiti, per molte piattaforme. Applicazioni come GZIP e ZIP sono veloci ed efficienti. Infine, XML è ottimizzato per il trasporto su Internet con il protocollo http/1.1, che, per implementazione, comprime al volo le stringhe di dati testuali, per una minore occupazione di banda.

6. XML è relativamente nuovo, ma ha radici di tutto rispetto

La prima idea dell'XML risale al 1996; lo standard W3C è del 10 febbraio 1998. XML è basato sullo Standard Generalized Markup Language (SGML), creato una decina d'anni prima, che è a sua volta un metalinguaggio per la creazione di altri linguaggi: una delle applicazioni di SGML è HTML. Questa relazione può essere fonte di qualche confusione, ma basta ricordare che XML è un sottoinsieme di SGML, mentre HTML è un'applicazione di SGML. XML e HTML non sono "fratelli". La Figura 1.1 visualizza questa relazione ed elenca varie applicazioni di

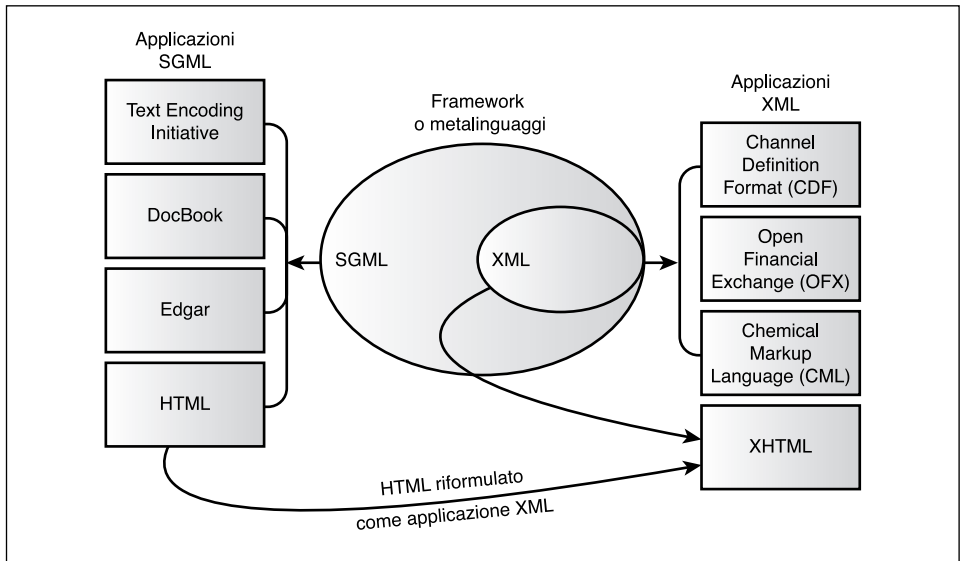


Figura 1.1

La relazione fra SGML, XML e HTML.

SGML e XML. Di alcune applicazioni particolarmente diffuse di XML parleremo nel Giorno 19, nell'ambito di una discussione sull'integrazione di XML in un modello di business.

Forse avete sentito parlare di XHTML, eXtensible Hypertext Markup Language, e vi state chiedendo che cosa sia e in quale rapporto stia con gli altri linguaggi di markup. In senso stretto, XHTML è la riformulazione di HTML come applicazione XML, con l'obiettivo di dare ad HTML un certo grado di estendibilità. Il W3C offre molte informazioni su XHTML nel suo sito <http://www.w3.org/Markup/>. XHTML porta il rigore dell'XML nel campo del rendering per il Web.

7. XML non richiede licenza, è indipendente dalla piattaforma, ed è ben supportato

Nessuno ha un'esclusiva su XML: non ci sono problemi di licenza, il linguaggio è disponibile per tutte le implementazioni, e anche le tecnologie componenti sono di pubblico dominio. L'indipendenza dalla piattaforma rende XML ideale per l'uso sul Web. Nuovi modelli di e-business richiedono sistemi trasparenti di scambio di dati basati sulle transazioni via Internet.

La maggior parte delle società di sviluppo ha contribuito con la creazione di strumenti, la promozione di standard e la preparazione di soluzioni esemplificative, e in genere offre ampio sostegno agli sviluppatori.

Il ruolo dell'e-business

L'esperienza degli utenti è stata arricchita dai metodi multimediali di presentazione dei dati, il che ha determinato un cambiamento significativo nel "look and feel" del Web. Le aziende però hanno bisogno di forme molto raffinate di trasferimento delle informazioni, e hanno fornito le risorse finanziarie necessarie per una forte accelerazione dello sviluppo di metodi universali e di ampia portata. In effetti, il modo in cui svolgiamo le nostre attività è cambiato drasticamente da quando le imprese hanno abbracciato l'idea del trasferimento di dati aziendali in forma elettronica.

Quando la comunità degli affari ha cominciato a sostenere l'uso del Web per attività commerciali, le attese dei clienti Web sono cresciute esponenzialmente. Le pagine Web statiche non possono fornire dati tempestivi, e così si sono sviluppati i modelli di contenuto dinamico. Le aziende hanno avuto bisogno di nuovi strumenti e di nuove tecnologie per distribuire non solo dati di tipo tradizionale, ma anche informazioni in un nuovo stile, in risposta alla raffinatezza crescente dei clienti Web. Probabilmente anche voi usate già servizi bancari via Web oppure prenotate i vostri viaggi via Internet.

Il passo successivo è stato l'individualizzazione e la personalizzazione dei contenuti, in modo che i siti Web potessero "riconoscere" un visitatore e fornirgli un'esperienza personalizzata, tagliata secondo le sue esigenze specifiche, individuali. Vi sarà sicuramente capitato di tornare a visitare un sito e di vedervi salutare per nome; forse siete stati anche in un sito di commercio elettronico che ricordava l'ultimo acquisto che avevate fatto. Il sito magari suggerisce oggetti a cui potreste essere interessati, sulla base degli acquisti precedenti: per esempio, un libro dello stesso autore oppure un disco dello stesso genere musicale. Ci sono portali che identificano la vostra zona e indicano l'ora locale, le temperature correnti e le previsioni del tempo. Sono tutti esempi di personalizzazione dei contenuti.

Nuovi modelli di business contribuiscono a un rapido progresso delle tecnologie di distribuzione delle informazioni: la cosiddetta *syndication* ne è un esempio. La maggior parte dei siti che offre un servizio del genere raccoglie il vostro codice o indirizzo postale e poi memorizza sul vostro computer un piccolo numero identificativo di riferimento (un *cookie*). Quando tornate allo stesso sito, un programma recupera il cookie e cerca il vostro record nel database dei visitatori; poi mette in corrispondenza il luogo in cui risiedete con le previsioni del tempo, ricavandole da un grande database online, e in questo modo crea una pagina Web personalizzata per voi. I dati utilizzati normalmente sono ottenuti da un servizio che fornisce le stesse informazioni di base a tutti quelli che vi si sono abbonati.

I modelli a micropagamenti offrono un altro esempio di innovazione che dipende dal Web. Si può andare, per esempio, a una libreria online e acquistare, di un libro, solo i capitoli che interessano. Lo scambio di informazioni necessario per effettuare una transazione del genere possono essere automatizzati su un si-

to Web, che può addirittura aiutare a scegliere capitoli simili da più libri, per costruire un documento personalizzato su un argomento particolare. Questo procedimento si sta rapidamente diffondendo nell'industria musicale, dove ormai è molto facile creare e acquistare un CD contenente solo i brani che si desiderano.

Anche transazioni di e-business fuori linea hanno contribuito a promuovere i progressi tecnologici. Pensate alle transazioni elettroniche che avvengono quando decidete di pagare la benzina con una carta di credito: inserite la vostra carta in un lettore che ricava il numero della carta e altri dati memorizzati sulla banda magnetica e trasferisce quei dati a un servizio di elaborazione (una stanza di compensazione). Questa controlla la validità della carta e verifica che il conto corrispondente abbia un credito sufficiente per consentire l'acquisto. Una volta convalidata la carta, alla pompa viene restituita una transazione di accettazione, che la attiva. Mentre riempite il serbatoio, vengono scambiate altre transazioni fra il distributore di benzina e la banca, viene calcolato il totale, si scambiano i messaggi di chiusura e, quando la pompa si ferma, viene stampata una ricevuta.

Se leggerete questo libro e studierete gli esempi di programmazione, vedrete che XML è ideale per creare soluzioni per questi e molti altri problemi di business. Manipolerete tecnologie XML, creerete documenti personalizzati con i vostri elementi di dati autodescrittivi ed esplorerete analizzatori sintattici e interfacce d'applicazione. Vedrete anche come trasformare i marcatori da un linguaggio all'altro, come usare XML con l'elaborazione su database e dal lato server, come integrare una serie di tecnologie per creare soluzioni valide.

Il primo documento XML

Abbiamo visto che XML è costituito da elementi intuitivi, autodescrittivi, contenuti in un elemento radice. Ci sono ancora un po' di regole, abbastanza semplici, da conoscere, ma ne parleremo nel secondo giorno. A questo punto, ne sapete già abbastanza da poter creare il vostro primo documento XML. Ecco gli strumenti che vi servono per svolgere questo esercizio:

- Un editor di testo (Blocco Note, SimpleText o qualcosa del genere). Potete usare qualsiasi editor che consenta di salvare documenti in formato testo ASCII, ma se usate un word processor state attenti a non salvare i vostri documenti in formato binario.
- Useremo più volte, in questo libro, il browser Microsoft Internet Explorer (versione 5.0 o successiva). Se non ne avete già una copia caricata sulla macchina, potete scaricarlo all'indirizzo <http://www.microsoft.com/downloads/search.asp>.

Il documento che creeremo è molto semplice: conterrà il vostro nome, il cognome e una battuta. Chiamate il primo elemento <nome>, il secondo <cognome>. La

battuta può essere qualsiasi stringa di testo di una riga, in un elemento denominato <battuta>. Cominciate con il lanciare l'editor che avete scelto, poi create il documento XML. Non dimenticate che dovete avere un elemento radice e un elemento per ciascuna componente distinta dei dati. Dovete assegnare un nome all'elemento radice. Dovete anche terminare ciascun elemento: in questo caso, terminerete con un marcatore di fine.

Quando avete finito, salvate il documento come `io.xml`, poi lanciate Internet Explorer e guardate il risultato. Se non salvate il documento con l'estensione `.xml`, probabilmente non otterrete i risultati desiderati.

Per vedere un documento in Internet Explorer, dovete inserire il percorso e il nome di file nel campo dell'indirizzo nella finestra del browser. Quando un documento viene visualizzato in Internet Explorer, l'applicazione browser fa passare il documento XML in un particolare processore, l'analizzatore sintattico o *parser*. L'analizzatore controlla che il documento sia conforme alle regole sintattiche dei documenti XML ben formati. Se le marcature sono corrette, l'analizzatore trasforma il documento in un formato di facile lettura, con le etichette dei marcatori in colore diverso da quello del testo racchiuso fra le etichette. Questo processo di trasformazione prende il nome di Extensible Stylesheet Language Transformation (XSLT). Approfondiremo l'argomento nel Giorno 16.

Il Listato 1.7 offre una possibile soluzione, ma la vostra può essere diversa, con nomi diversi per gli elementi, un annodamento diverso e, ci auguriamo, una battuta più divertente.

Listato 1.7 Un primo documento XML

```
<dati>
  <persona>
    <nome>Devan</nome>
    <cognome>Shepherd</cognome>
  </persona>
  <battuta>ASCII silly question, get a silly ANSI</battuta>
</dati>
```

La Figura 1.2 mostra come viene visualizzato questo semplice documento XML con gli stili incorporati nel browser Microsoft Internet Explorer. La trasformazione viene effettuata in modo automatico, se il documento è ben formato.

Come abbiamo già detto, IE mostra i nomi degli elementi in un colore e i contenuti in un altro. Questo risultato viene ottenuto trasformando XML in una pagina HTML, contenuta in memoria, con un "foglio stile" (*stylesheet*) associato. Lo stile è interno alla distribuzione fornita con IE. In effetti il browser ha al suo interno un processore XML, un analizzatore sintattico.

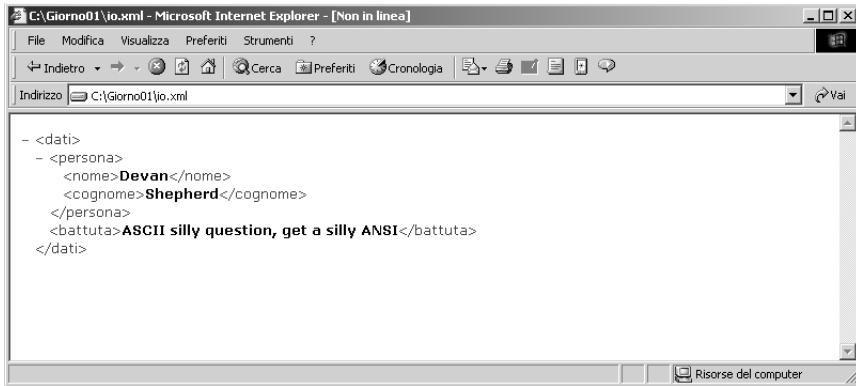


Figura 1.2

Il testo del listato 1.4, visualizzato in IE.

Riepilogo

Abbiamo introdotto XML come framework, ovvero come metalinguaggio per la creazione di altri linguaggi. Abbiamo visto l'importanza del markup nel mondo dell'e-commerce e come la comunità degli affari abbia contribuito a favorire la rapida accelerazione delle tecnologie Web. Abbiamo analizzato le sette caratteristiche fondamentali dell'XML e abbiamo studiato alcune delle regole sintattiche dello standard; infine abbiamo creato un primo documento XML e l'abbiamo fatto elaborare all'analizzatore sintattico di Microsoft Internet Explorer. Nella prossima lezione approfondiremo ulteriormente il tema della buona formazione degli enunciati XML e potremo capire meglio i vantaggi che XML offre a chi programma per il Web.

Domande e risposte

D. In che modo la comunità degli affari ha svolto un ruolo nello sviluppo del Web?

R. Le aziende hanno favorito uno sviluppo accelerato delle tecnologie Web non solo con i loro finanziamenti, ma anche con la creazione di nuovi modelli per l'e-business e lo scambio di dati.

D. A che cosa ci serve un altro linguaggio di markup?

R. In senso stretto, XML non è un linguaggio di markup, ma è necessario in parte per estendere i metodi di programmazione per il Web, oltre i limiti intrinseci di HTML.

D. C'è qualcosa che HTML fa particolarmente bene?

R. Ci dà un modo per creare ipertesti eseguibili. È facile da apprendere e per questo ha goduto di un'amplissima diffusione nel Web. È supportato da un'ampia gamma di interfacce browser a basso costo ma molto potenti.

D. Posso visualizzare il mio documento in Netscape o in qualche altro browser?

R. Netscape versione 6.0 supporta i documenti XML. Potete visualizzare un documento XML con Netscape, ma il documento non apparirà nella stessa forma che presenta in Internet Explorer. In effetti, val la pena di provare a visualizzare il documento in un altro browser per capire quali siano le differenze. Ricordate che IE effettua una trasformazione da XML a HTML al fine di presentare il documento con simboli di markup: la trasformazione viene effettuata attraverso una Extensible Stylesheet Language Transformation (XSLT) con un foglio stile di default interno a IE (di XSLT parleremo nel Capitolo 16).

D. Quali sono i problemi di HTML?

R. HTML difetta di flessibilità, a causa del suo insieme limitato di marcatori predefiniti. Non può essere esteso facilmente e ha un insieme rigido di stili predefiniti.

D. Quali sono le sette caratteristiche fondamentali di XML?

R. Le sette caratteristiche sono

1. XML offre un metodo per inserire dati strutturati in un file di testo.
2. XML assomiglia un po' a HTML.
3. XML è leggibile dalle macchine, ma comprensibile agli esseri umani.
4. XML è costituito da una famiglia di tecnologie.
5. XML è prolisso.
6. XML è relativamente nuovo, ma ha radici di tutto rispetto.
7. XML non richiede licenza, è indipendente dalla piattaforma, ed è ben supportato.

Esercizio

Questo esercizio vi permetterà di mettere alla prova quanto avete imparato in questo Capitolo. La risposta è nell'Appendice A.

Create un documento XML che presenti il vostro nome, l'indirizzo di casa, l'indirizzo di posta elettronica e la data di nascita. Usate per la data gli attributi. Quando avete finito, salvate il documento e visualizzatelo con Microsoft Internet Explorer 5.0.