

# 1. Conversione e formattazione di un documento in XHTML e CSS

*19 gennaio 2008 - 8 marzo 2008*

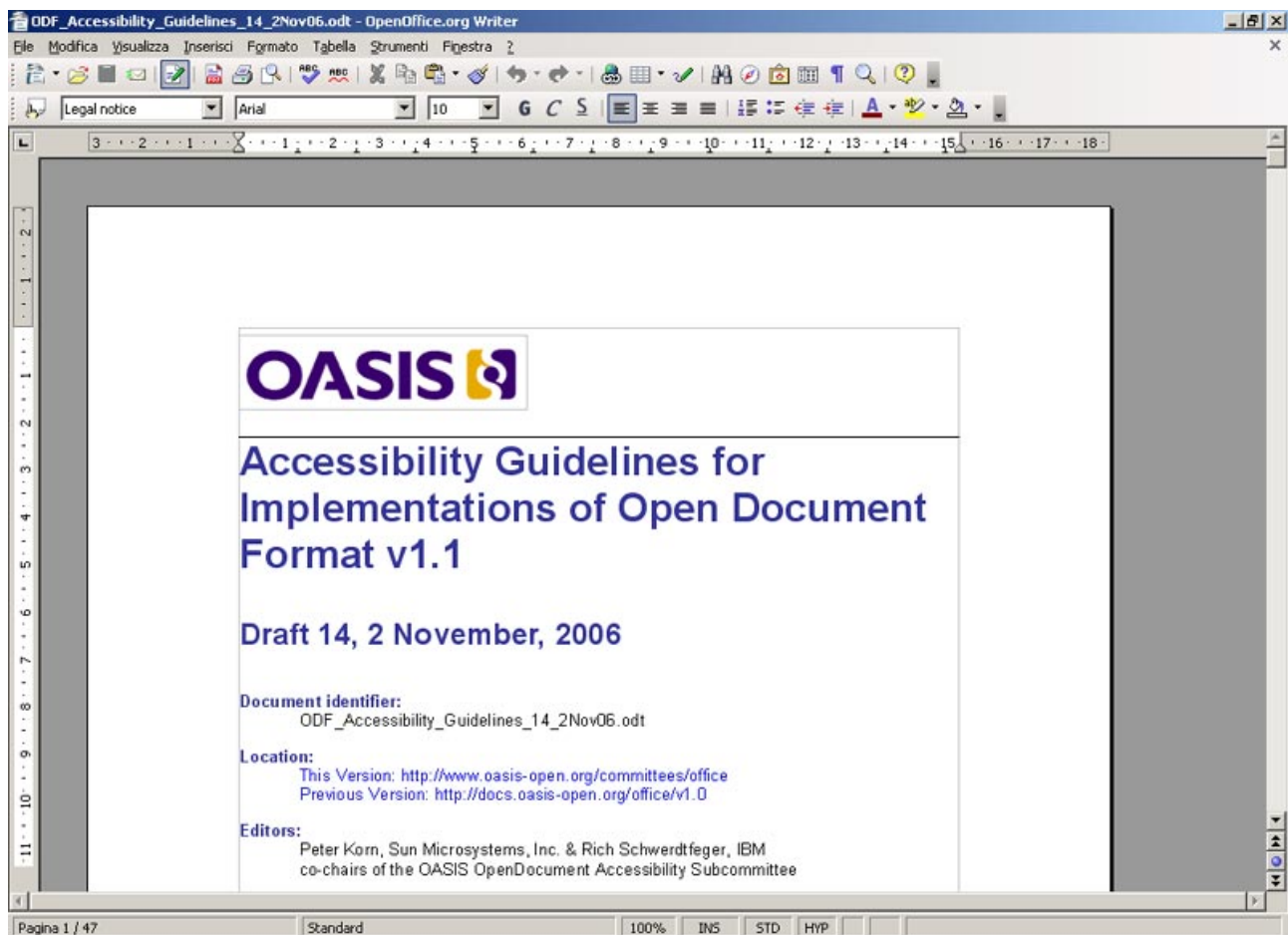
## 2. Documento di partenza

**Nome:** Accessibility Guidelines for Implementations of Open Document Format v1.1

**Formato:** File ODF

**Dimensione:** 93, 2 Kb

## 3. Screenshot del documento di partenza

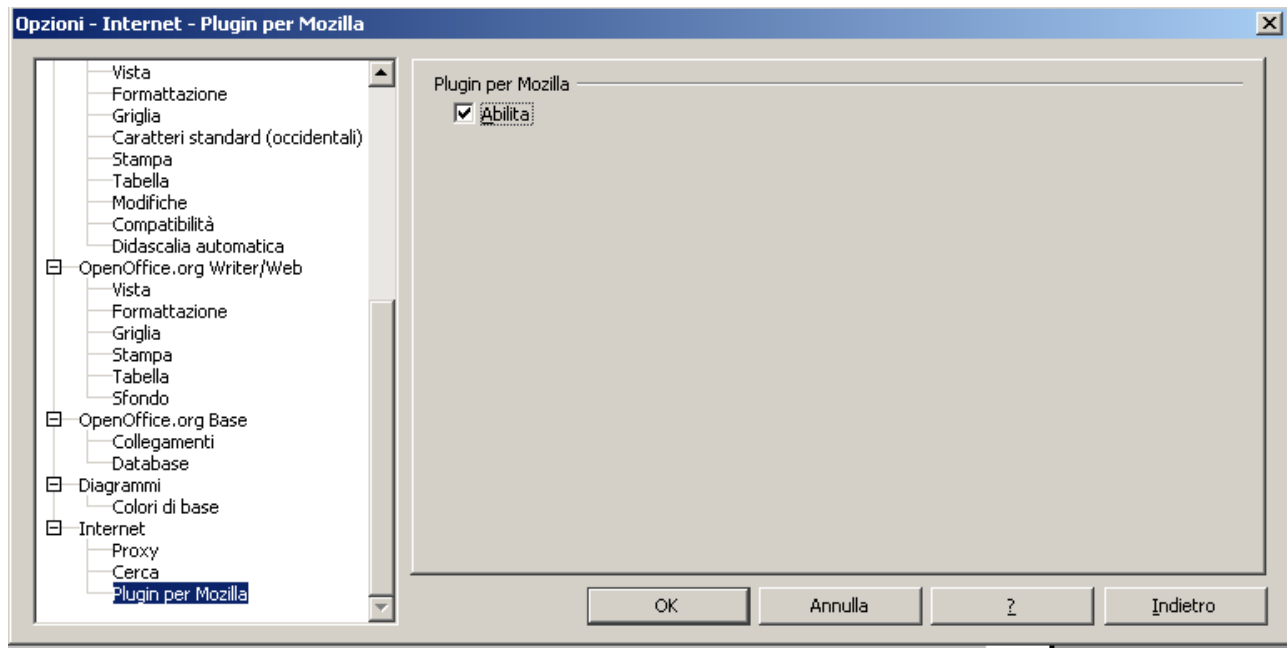


*Figura 1: Il documento aperto in Open Office*

[\[D\]](#)

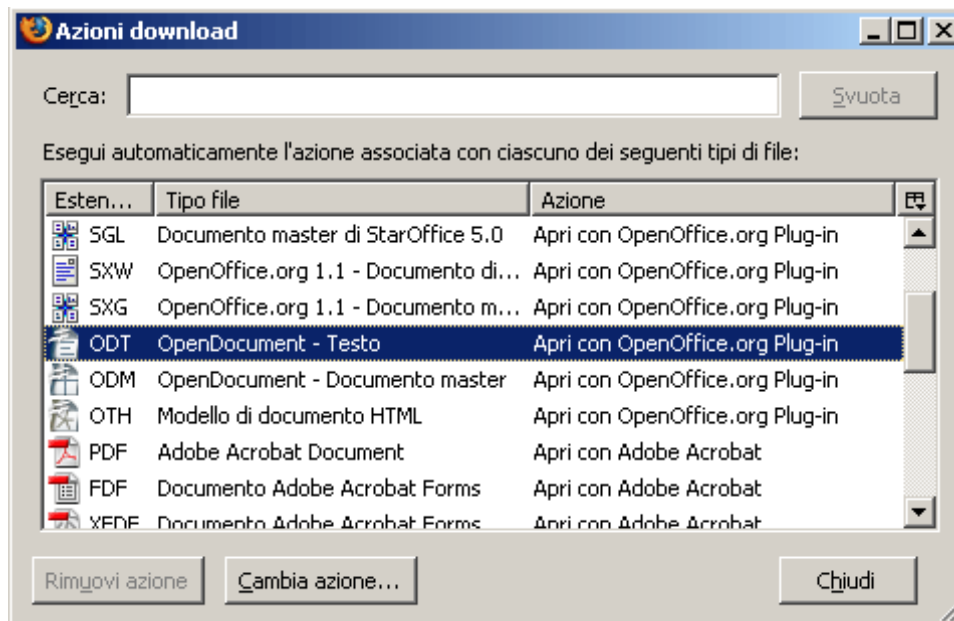
## 4. Problema principale con questo tipo di documento

Il formato del documento richiede la presenza di un particolare tipo di plug-in per il browser. Nello specifico, Open Office 2.1 permette di abilitare il plugin solo per i browser basati su Mozilla, come mostrato dalla seguente immagine.



*L'utente deve selezionare la voce **Opzioni** → **Internet** → **Plugin per Mozilla** e quindi **Abilita**.*

Subito dopo aver eseguito questa operazione, Firefox riconoscerà i file correlati con questo formato di documento, come mostrato dall'immagine che segue.



*In Firefox l'utente verifica il riconoscimento del plugin in **Opzioni** → **Contenuti** → **Tipi di file** → **Gestione**.*

*In questo caso viene evidenziato il tipo di file con estensione .odt.*

**Nota:**

In un test eseguito in locale, Open Office è andato in crash durante l'apertura di un file **.odt** inserito in un collegamento ipertestuale, per quanto il funzionamento di Firefox non ne abbia risentito. Per questo motivo si raccomanda di utilizzare, se possibile, l'ultima versione del programma.

## **5. Tentativi di conversione e risultati**

**Formato DOC:** 361 Kb

**Formato PDF:** 358 Kb

Entrambe le conversioni ci mostrano un significativo aumento delle dimensioni del documento di origine.

## **6. Pro e contro dei tre formati**

Formato	Vantaggi	Svantaggi
ODF	<ul style="list-style-type: none"> <li>• è un formato aperto</li> <li>• è basato su XML</li> <li>• le sue dimensioni sono inferiori agli altri due</li> </ul>	<ul style="list-style-type: none"> <li>• necessita di un particolare plug-in</li> <li>• non è leggibile dagli altri due</li> </ul>
DOC	<ul style="list-style-type: none"> <li>• è uno dei formati più diffusi</li> <li>• è facilmente stampabile</li> </ul>	<ul style="list-style-type: none"> <li>• è un formato proprietario</li> <li>• le sue dimensioni sono le maggiori tra i formati esaminati</li> <li>• è consultabile online a patto di modificare l'azione predefinita del browser</li> </ul>
	<ul style="list-style-type: none"> <li>• è uno dei formati più diffusi</li> <li>• è facilmente</li> </ul>	<ul style="list-style-type: none"> <li>• le sue dimensioni sono</li> </ul>

PDF	stampabile	maggiori del formato ODF
	<ul style="list-style-type: none"><li>• è consultabile anche online</li></ul>	



## 7. Quale formato scegliere?

Come alternativa al formato (X)HTML possiamo scegliere, in ordine di priorità:

1. formato PDF (consultabile online e scaricabile)
2. formato DOC (scaricabile)
3. formato ODF (scaricabile)

## 8. Il formato (X)HTML

Il formato (X)HTML presenta **tutti** i vantaggi sopra elencati ed è, come noto, il formato predefinito dei siti web. Tuttavia, va operata un'importante distinzione tra HTML ed XHTML.

# 9. Differenze tra HTML ed XHTML

Esistono numerose differenze tra i due formati.

## 10. HTML

- viene servito come `text/html`. Per questo motivo, **non** è compatibile con i programmi utente XML.
- i programmi utente usano due modalità di rendering per la sua rappresentazione:

1. **modalità quirk:** mancato rispetto dello standard CSS
2. **modalità standard:** pieno rispetto dello standard CSS

le due modalità vengono attivate dalla presenza o meno di una particolare DTD. il rispetto dello standard CSS dipende dal livello di supporto raggiunto dalle singole implementazioni.

- i programmi utente trattano la marcatura HTML con estrema permissività, a prescindere dalla modalità di rendering utilizzata. Ciò significa che la tolleranza all'errore è molto elevata. Esempio:

```
<p><span>Ciao<em>mondo</em></span></p>
```

un programma utente tratta di norma il precedente codice HTML come se fosse:

```
<p><span>Ciao<em>mondo</em></span></p>
```

il codice errato di cui sopra viene detto in gergo *zuppa di tag*. I programmi utente trattano comunque il formato HTML come zuppa di tag, e questo a prescindere dal fatto che la marcatura sia valida o meno.

# 11. XHTML

- XHTML 1.0 **dovrebbe** essere servito come **application/xhtml+xml**, mentre XHTML 1.1 **deve** essere servito come **application/xhtml+xml**. Questo tipo MIME lo rende compatibile con i programmi utente XML. Tuttavia esistono dei limiti all'applicazione del corretto formato:
  1. Internet Explorer 7 Windows (ed inferiori) non supporta il tipo MIME **application/xhtml+xml**
  2. nonostante molti server oggi supportino questo tipo MIME, non tutti permettono di scegliere la pagina



predefinita di una directory con estensione **.xht** o **.xhtml**

3. il supporto nei lettori di schermo e negli altri programmi assistivi è problematico.

- questo formato evita l'utilizzo di una modalità quirk da parte dei programmi utente, sia che venga servito come **application/xhtml+xml** che come **text/html**.
- se servito come **application/xhtml+xml** un documento XHTML viene letto seguendo le regole sintattiche dell'XML. Non vi è **nessuna** tolleranza all'errore. Ciò significa che la marcatura precedentemente esaminata non verrebbe rappresentata, ma produrrebbe una pagina di errore come la seguente.

**Errore interpretazione XML: tag corrispettivo mancante. Atteso: </em>.**  
**Indirizzo: file:///E:/update/template/xhtml11.xhtml**  
**Linea numero 12, colonna 25:**

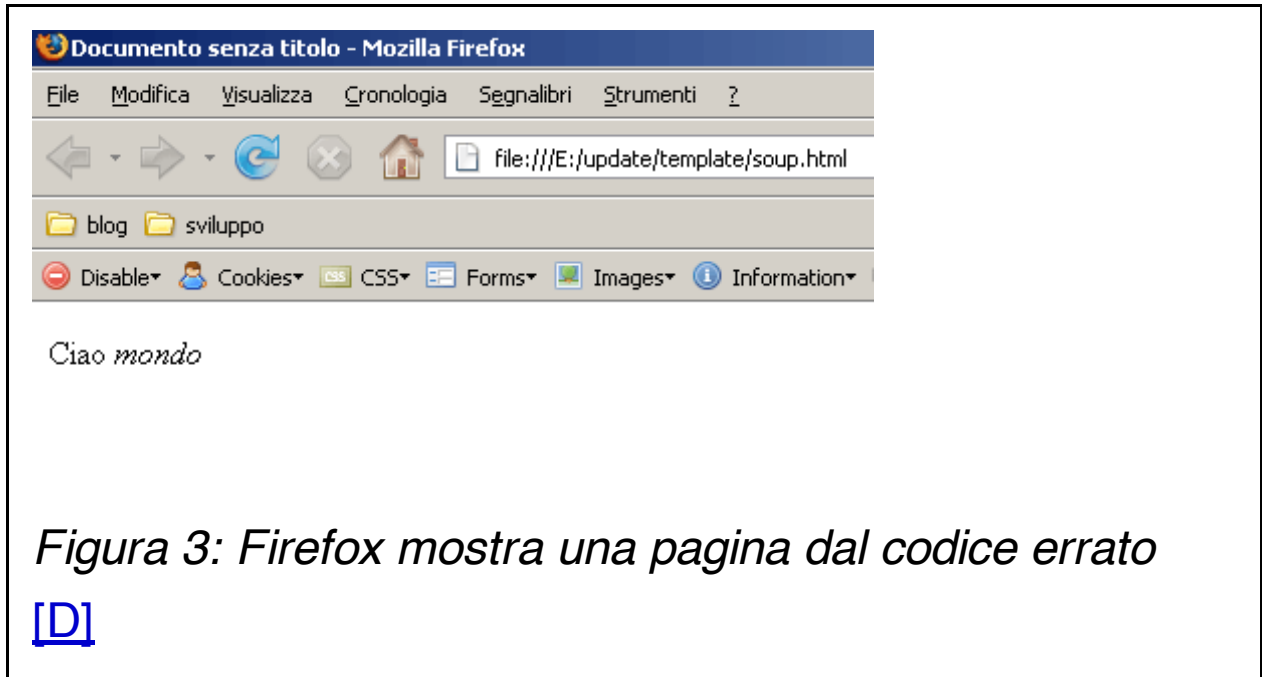
```
<p><span>Ciao<em>mondo</span></p>  
-----^
```

*Figura 2: Firefox mostra una pagina di errore nel parsing XML*

[\[D\]](#)

- se servito come **text/html** un documento XHTML viene letto esattamente come se fosse un documento HTML, ossia come zuppa di tag. Ciò significa che la marcatura

precedentemente esaminata verrebbe comunque rappresentata e non produrrebbe alcuna pagina di errore.



## 12. Vantaggi di XHTML

XHTML presenta numerosi vantaggi.

- se servito come **application/xhtml+xml** è compatibile con i nuovi formati derivati da SGML, come MathML, SVG ed XML
- è modularizzabile
- se servito come **text/html** è compatibile con i programmi utente obsoleti

## 13. Principale svantaggio di XHTML

Al momento Internet Explorer 7 (ed inferiori) non supporta il tipo MIME **application/xhtml+xml**, quindi i benefici sopra indicati vengono vanificati.

## 14. Possibili soluzioni

1. servire XHTML come **text/html**
2. servire XHTML come **text/html** a quei programmi utente che non supportano questo tipo MIME e come **application/xhtml+xml** a quei programmi utente che lo supportano

Il punto 2 può essere ottemperato tramite la negoziazione del contenuto.

## 15. Cenni sulla negoziazione

# del contenuto

La negoziazione del contenuto può essere implementata tramite alcuni linguaggi lato server, come PHP o ASP. Di seguito alcuni esempi.

## PHP

```
$accept = $_SERVER["HTTP_ACCEPT"];
$ua = $_SERVER["HTTP_USER_AGENT"];
if (isset($accept) && isset($ua)) {
    if (stristr($accept,
        "application/xhtml+xml") || stristr($ua,
        "W3C_Validator")) {
        header("Content-Type:
            application/xhtml+xml");
    }
}
```

## ASP

```
Dim strAccept, strUA
strAccept =
Request.ServerVariables("HTTP_ACCEPT").Item
strUA =
Request.ServerVariables("HTTP_USER_AGENT").Item
If InStr(1, strAccept,
"application/xhtml+xml") > 0 Or InStr(1,
strUA, "W3C_Validator") > 0 Then
```

```
Response.ContentType = "application/xhtml+xml"
```

```
End If
```

Alcuni riferimenti:

- [Serving XHTML As XML](#) – articolo presente sul sito [XHTML.com](#)
- [Content negotiation](#) – articolo presente sul sito [Juicy Studio](#)

## 16. Importanza della validazione

Bisogna tenere presente che gli attuali programmi utente che supportano il tipo MIME `application/xhtml+xml`, non usano un parser validante nell'analisi dei documenti. In altre parole, essi controllano solo se il documento è ben formato ma non se il documento è valido secondo la DTD di riferimento. Per esempio, dato un documento XHTML 1.1 servito come `application/xhtml+xml` avente la seguente marcatura:

```
<body>
```

```
<foo>Ciao mondo</foo>
```

```
</body>
```

e i seguenti stili:

```
foo {  
display: block;  
width: 100px;  
height: 100px;  
line-height: 100px;  
background: red;  
color: white;  
margin: 2em auto;  
text-align: center;  
font-weight: bold;  
}
```

il risultato sarà il seguente:



*Figura 4: Firefox mostra ugualmente la pagina*

[\[D\]](#)

Occorre quindi validare sempre i propri documenti per assicurarsi che non vi siano errori sintattici. Questo si rivela fondamentale in fase di debugging al fine di verificare se il problema dipende dagli stili applicati o dalla marcatura

utilizzata.

**Nota:**

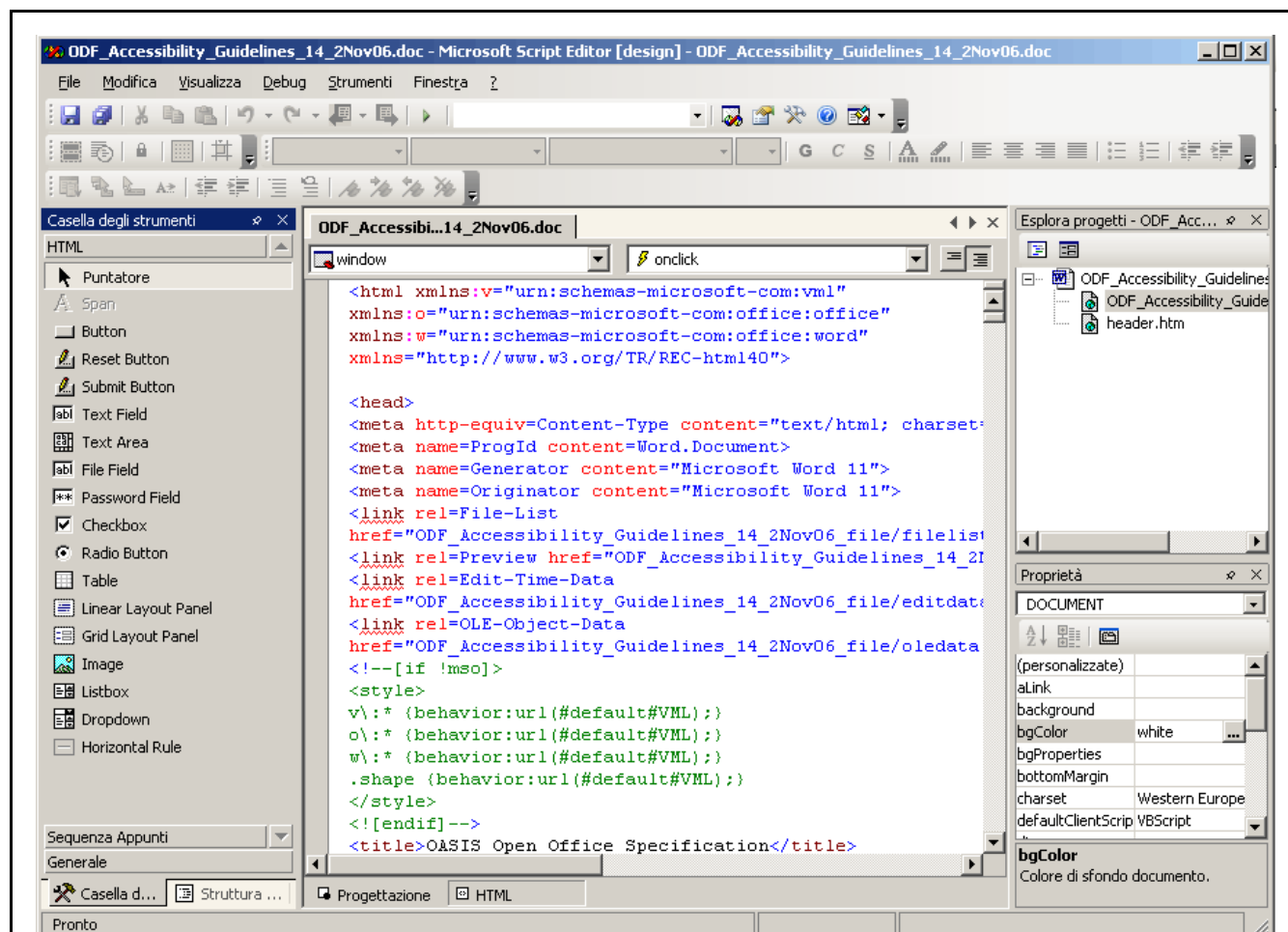
La directory `/res/dtd` contenuta nella directory di installazione di Mozilla Firefox **non** contiene le DTD dei vari linguaggi di marcatura, ma solo le entità SGML usate dal browser, come si evince dal seguente estratto:

```
<!--  
* Predefined HTML entities to be loaded when  
parsing XHTML documents.  
* The contents match  
mozilla/parser/htmlparser/src/nsHTMLEntityList.h,  
* except that Navigator entity extensions  
are not included.  
-->
```

## 17. Da dove partire

Il punto di partenza è ovviamente il documento originale. Lo sviluppatore web potrebbe essere tentato di usare strumenti automatici per la conversione del documento in un formato diverso. Per esempio, sia Open Office che Microsoft Word permettono il salvataggio del documento in formato HTML. Tuttavia, la marcatura prodotta da questi due programmi è per lo più ridondante e presentazionale, costringendo di fatto lo

sviluppatore ad un lungo e faticoso lavoro di "ripulitura". Ci si può rendere conto dell'output finale aprendo il documento convertito in formato **.doc** con l'editor interno di Microsoft Word.



*Figura 5: La marcatura prodotta da Microsoft Word*

[\[D\]](#)

La situazione non migliora in Open Office, che pur rinunciando ad una sintassi proprietaria, mostra una sovrabbondanza di elementi presentazionali usati per ricreare il layout di partenza. Occorre quindi cominciare da zero.



# 18. Domande e risposte

Quando si affronta la costruzione di un layout basato su un documento di un formato diverso da quello (X)HTML occorre porsi delle domande sul layout finale da ottenere.

## 1. Qual'è il layout del documento di partenza?

Il documento originale presenta un layout di tipo monolitico, con una sola colonna che racchiude tutto il contenuto.

## 2. Il documento di partenza ha delle dipendenze esterne (immagini, oggetti, ecc.)?

Il documento originale comprende delle immagini inserite al suo interno. Le immagini sono di due tipi:

1. un logo
2. immagini di esempio.

## 3. Quali sono i caratteri usati nel documento di partenza?

Un font principale della famiglia **sans-serif**, nello

specifico l'Arial, ed un font monospaziato (**monospace**) per il codice di esempio.

#### 4. **Qual'è lo schema di colori usato nel documento di partenza?**

Abbiamo intestazioni e link impostati sullo stesso colore (blu scuro). Le tabelle, il codice e il sommario presentano uno sfondo grigio scuro.

#### 5. **Lo schema di colori originale soddisfa i requisiti del W3C?**

No. Va modificato.

#### 6. **Qual'è la struttura del documento di partenza?**

Nel documento originale sono presenti i seguenti elementi strutturali:

1. un logo
2. un elenco degli autori e della versione del documento
3. un sommario
4. una sequenza ripetuta di
  1. intestazioni

2. paragrafi
3. elenchi
5. codice di esempio
6. tabelle di esempio
7. immagini di esempio
8. un piè di pagina con i ringraziamenti e le note finali.

## 7. **Sono presenti degli schemi di posizionamento?**

Vi sono immagini allineate a sinistra con didascalia sottostante.

## 8. **Quale formato sceglieremo per il documento di destinazione?**

Sceglieremo il formato XHTML servito come **`application/xhtml+xml`**.

# 19. Sulla scelta del formato

Scegliamo di usare XHTML servito come

**application/xhtml+xml** per alcuni motivi che andiamo a riassumere:

## 1. **Certezza dell'essere ben formato del documento**

Servendo il documento di destinazione come **application/xhtml+xml**, i browser rileveranno ogni errore riguardante il corretto annidamento degli elementi. Questo ci permetterà di risparmiare tempo e, al contempo, acquisire quella disciplina necessaria a scrivere codice sintatticamente valido.

## 2. **Possibilità di convertire il documento in formati diversi**

Servendo il documento con questo tipo MIME, la conversione in altri formati diventa più semplice. Data la correttezza formale necessaria ad ottenere un documento ben formato, si potrebbe convertire il documento in formato **text/html** (HTML 4.01 o XHTML 1.0 Strict) o **text/xml** (XML) venendo così incontro ad una vasta gamma di programmi utente.

Nello specifico, per questo documento scegliamo di usare la DTD XHTML 1.1 ed il seguente DOCTYPE:

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en">
```

Il documento di destinazione avrà l'estensione **.xhtml**, indispensabile per il corretto riconoscimento del tipo MIME in un file statico.

## 20. La struttura di un layout monolitico

Un layout monolitico ha le seguenti caratteristiche:

1. Il flusso degli elementi segue di norma il flusso predefinito del documento. Gli elementi si dispongono dall'alto verso il basso seguendo le loro regole di visualizzazione predefinite (di blocco e inline), a meno che queste non vengano alterate da regole di stile specifiche.
2. Disponendosi su una sola colonna, questo layout è particolarmente adatto alla rappresentazione di documenti consultabili, quali tesi di laurea, articoli universitari ed altro.
3. La disposizione su una sola colonna facilita la lettura, a patto di non usare una lunghezza eccessiva delle singole

righe.

4. Questo tipo di layout viene per esempio usato da [Michele Diodati](#) per l'impaginazione dei documenti principali del suo sito.

## 21. Le regole di stile generali

Per essere efficace, un foglio di stile deve seguire un ordine preciso, cominciando da dichiarazioni generali per l'intero documento e passando quindi a definire delle dichiarazioni particolari per sezioni specifiche. Nello scrivere tale foglio di stile si dovrebbero seguire delle buone pratiche che andiamo a riassumere:

1. Dividere il foglio di stile in blocchi specifici.
2. Aggiungere un commento per ciascun blocco in modo da favorire la sua manutenzione.
3. Usare la giusta indentatura per il codice, evitando di scrivere i blocchi di dichiarazione su una singola riga.
4. Evitare dichiarazioni ridondanti, usando ove possibile la sintassi abbreviata delle proprietà.

Passiamo quindi alle dichiarazioni generali per il nostro

documento.

```
1.  body {
2.    margin-left: 5%;
3.    margin-right: 5%;
4.    font: 100%/1.5 Arial, Helvetica, sans-
      serif;
5.    background: #fff;
6.    color: #000;
7.  }

8.  a:link {
9.    color: #00c;
10.   background: transparent;
11.   text-decoration: none;
12.   border-bottom: 1px solid;
13.  }

14. a:visited {
15.   color: #800;
16.   background: transparent;
17.  }

18. a:hover, a:active {
19.   color: #400040;
20.   background: transparent;
21.  }

22. h1, h2, h3, h4, h5, h6 {
23.   line-height: normal;
```

```
24.     color: #339;
25.     background: transparent;
26. }
27. h3, h4, h5, h6 {
28.     letter-spacing: 0.1em;
29. }
30. h2, h3, h4, h5, h6 {
31.     margin-bottom: 0;
32. }
33. h5 {
34.     text-transform: uppercase;
35. }
36. h5.example {
37.     color: #000;
38.     font-size: 1em;
39.     font-weight: bold;
40.     text-transform: none;
41. }

42. ins {
43.     display: none;
44. }
```

Il codice generale appare diviso in quattro blocchi principali:

### 1. Righe 1-7

Vengono qui date le dichiarazioni generali per l'elemento



**body**. Si impostano i margini orizzontali su un valore in percentuale, in modo da avere un layout fluido. Si impostano il colore del testo e dello sfondo, l'interlinea e il tipo di font per il documento, nonché la sua dimensione.

## 2. Righe 8-21

Vengono qui date le dichiarazioni per lo stile dei collegamenti ipertestuali presenti nel documento, seguendo l'ordine **:link**, **:visited**, **:hover**, **:active**. Si noti che al fine della semplice conversione del documento sarebbe sufficiente attribuire uno stile solo alla pseudo-classe **:link**. Si noti altresì come la sottolineatura dei link sia stata eliminata a favore di un bordo inferiore, al fine di garantire un certo spazio tra la linea di base del testo e la sottolineatura.

## 3. Righe 22-41

Si attribuiscono degli stili alle intestazioni presenti nel documento, seguendo l'ordine di importanza **h1-h6**. Viene ridotta l'interlinea, assegnato un colore specifico, una spaziatura tra le lettere (elementi **h3-h6**), ridotto lo spazio inferiore tra le intestazioni **h2-h6** e gli elementi contigui, e si rende in lettere maiuscole il testo dell'elemento **h5**. Da ultimo, la classe **.example** applicata ai soli elementi **h5** fa in modo che questi ultimi abbiano il testo di colore nero,

non siano in lettere maiuscole ed abbiano una dimensione del font pari a quella dell'elemento **body** con il testo in grassetto.

#### 4. Righe 42-44

Nel documento sono presenti delle note di redazione, prive di valore contestuale ai fini della comprensione del documento. Per questo motivo sopprimiamo la loro visualizzazione marcandole con l'elemento **ins** a cui assegniamo un valore **none** per la proprietà **display**. Si noti che in questo modo tali note non verranno lette dai lettori di schermo che interpretano tale dichiarazione (come Jaws). Se si ritiene che tali note abbiano un valore ai fini della comprensione del documento, si può ricorrere a questa dichiarazione alternativa:

```
ins {  
  position: absolute;  
  top: -1000em;  
}
```

Assegnando un valore negativo altissimo alla proprietà **top** facciamo in modo che l'elemento scompaia dalla finestra di visualizzazione. Questa tecnica ci permette di avere un'ottima compatibilità con i lettori di schermo e gli altri programmi che non supportano i fogli di stile.

## 22. Il logo

Il logo è rappresentato dalla seguente immagine.



*Figura 11: Il logo del documento*

[\[D\]](#)

Per inserire il logo ricorriamo ad una marcatura molto semplice:

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

Volendo è possibile associare un link nascosto che rimandi ad una descrizione estesa dell'immagine. Questa tecnica andrebbe usata in combinazione con l'attributo **longdesc**, non ampiamente supportato. Possiamo nascondere il link con la tecnica vista in precedenza per l'elemento **ins**. Data la semplicità della marcatura, questa sezione non necessita di stili particolari.

## 23. L'elenco degli autori e della versione del documento

L'elenco degli autori e della versione del documento viene formattato con la seguente marcatura:

```
<dl id="pref">
  <dt><strong>Document identifier:</strong>
  </dt>
  <dd>ODF_Accessibility_Guidelines...</dd>
  ...omissis...
  <dt><strong>Authors:</strong></dt>
  <dd>
    <ul>
      <li>Janina Sajka, Capital
        Accessibility</li>
      <li>Stephen Noble, Design Science, Inc.
        </li>
      ...omissis...
    </ul>
  </dd>
</dl>
```

Scegliamo di usare un elenco di definizione per stabilire una precisa relazione semantica tra le voci dell'elenco. Ciascuna

voce viene infatti definita dalla voce che segue. A questo punto passiamo ad applicare gli stili.

```
1. dl#pref dt strong, {  
2.     color: #339;  
3.     background: transparent;  
4. }  
  
5. dl#pref {  
6.     font-size: 0.85em;  
7. }  
  
8. #pref ul {  
9.     margin-left: 0;  
10.     padding-left: 0;  
11. }  
12. #pref ul li {  
13.     display: block;  
14. }
```

Abbiamo assegnato un colore agli elementi **strong** presenti all'interno degli elementi **dt** e abbiamo quindi diminuito la dimensione del font per l'elenco di definizione (righe 5-7).

Le righe che vanno dalla 7 alla 14 presentano alcune caratteristiche interessanti nella formattazione degli elenchi non ordinati.

Il primo blocco di dichiarazioni (righe 8-11) annulla il classico rientro a sinistra degli elenchi non ordinati, azzerando il margine ed il padding sinistro dell'elemento `ul`. Questa doppia dichiarazione si rende necessaria in quanto alcuni browser (come Internet Explorer e Opera) usano il margine sinistro per creare il rientro, mentre altri (come Safari e Firefox) usano il padding sinistro.

L'ultimo blocco di dichiarazioni (righe 11-14) elimina il marcatore dell'elenco non ordinato dichiarando ogni voce di elenco come un elemento di blocco. Secondo le specifiche, infatti, un elemento può avere un marcatore solo se il valore della proprietà `display` è impostato su `list-item`. In questo modo evitiamo di dover ricorrere alla dichiarazione `list-style-type: none` che potrebbe confondere i lettori di schermo, come già segnalato da Michele Diodati nel sesto capitolo della sua *Accessibilità. Guida completa*.

Purtroppo questa soluzione non funziona in Internet Explorer 7 ed inferiori. Per questo browser, infatti, una voce di elenco viene già considerata come un elemento di blocco, e quindi il marcatore non viene soppresso. Se vogliamo ottenere una compatibilità anche con questo browser dobbiamo ricorrere ad un semplice stratagemma.

Un marcatore di elenco di norma eredita dalla voce cui appartiene il colore del testo. Quindi dichiarare per una voce di elenco un colore del testo uguale a quello di sfondo equivale di

fatto a rendere invisibile il marcatore oltre al testo della voce. Poiché quello che vogliamo è rendere invisibile solo il marcatore, dobbiamo modificare la marcatura aggiungendo un elemento all'interno delle voci di elenco che vada a racchiudere il testo. Nel nostro caso aggiungiamo un elemento **span**:

```
<dd>
  <ul>
    <li><span>Janina Sajka, Capital
      Accessibility</span></li>
    <li><span>Stephen Noble, Design
      Science, Inc.</span></li>
    ...omissis...
  </ul>
</dd>
```

Gli stili da applicare sono i seguenti:

```
1. #pref dd ul li {
2.   color: #fff;
3. }
4. #pref dd ul li span {
5.   color: #000;
6. }
```

Il primo blocco di dichiarazioni (righe 1-3) annulla il marcatore di elenco impostando un colore del testo uguale a quello di sfondo (qui ereditato dall'elemento **body**), mentre il secondo

(righe 4-6) rende di nuovo visibile il testo all'interno dell'elemento **span**. Possiamo apprezzare il lavoro svolto sinora in questa immagine.

**Document identifier:**

ODF\_Accessibility\_Guidelines\_14\_2Nov06.odt

**Location:**

**This Version:**

<http://www.oasis-open.org/committees/office>

**Previous Version:**

<http://docs.oasis-open.org/office/v1.0>

**Editors:**

Peter Korn, Sun Microsystems, Inc. & Rich Schwerdtfeger, IBM  
co-chairs of the OASIS OpenDocument Accessibility Subcommittee

**Authors:**

Janina Sajka, Capital Accessibility  
Stephen Noble, Design Science, Inc.  
Chieko Asakawa, IBM  
Pete Brunet, IBM  
Rich Schwerdtfeger, IBM  
Dave Pawson, Royal National Institute for the Blind  
Peter Korn, Sun Microsystems, Inc.

**Abstract:**

This document is a guide for Office Applications, that support version 1.1 of the OpenDocument format, to promote and preserve accessible ODF documents. This guide is not a comprehensive guide for content mapping to platform accessibility APIs.

**Status:**

This document is an early draft.

*Figura 12: L'elenco degli autori e della versione del documento*

[\[D\]](#)

## 24. Il sommario

Il sommario si presenta come una serie di elenchi ordinati annidati, con una numerazione progressiva per ciascuna voce nei diversi livelli di annidamento. La marcatura necessaria alla sua formattazione è la seguente:



```

<ol id="toc">
  <li><a href="#bg-overview">Background and
  Overview</a>
  <ol>
    <li><a href="#intro">Introduction</a>
    </li>
    <li><a href="#access">What is
    Accessibility?</a>
    <ol>
      <li><a href="#type-
      access">Types/Categories of
      Access</a></li>
      <li><a href="#type-
      disabilities">Types of Disabilities,
      and How they are Addressed</a></li>
    </ol>
  </li>
  ...omissis...
</ol>

```

Decidiamo di usare i contatori del contenuto generato per ricreare la numerazione progressiva ordinata, consci dei notevoli limiti nel supporto a tale caratteristica dei fogli di stile da parte di alcuni browser e dei lettori di schermo. Offriremo tuttavia un'alternativa per venire incontro alle loro esigenze. Diamo quindi gli stili necessari a formattare il sommario usando il contenuto generato.

```
1. #toc {
```

```
2.   counter-reset: item;
3. }
4. #toc li {
5.   display: block;
6. }
7. #toc li:before {
8.   counter-increment: item;
9.   content: counter(item) ". ";
10.  font-weight: bold;
11.  font-family: Georgia, serif;
12.  color: #800;
13.  background: transparent;
14. }
15. #toc li ol {
16.   counter-reset: item2;
17. }
18. #toc li ol li:before {
19.   counter-increment: item2;
20.   content: counter(item) ". "
21.          counter(item2) " ";
22.   font-weight: normal;
23. }
24. #toc li ol li ol {
25.   counter-reset: item3;
```

```
26. #toc li ol li ol li:before {
27.     counter-increment: item3;
28.     content: counter(item) ". "
           counter(item2) ". " counter(item3) " ";
29.     font-style: italic;
30.     font-weight: normal;
31. }
```

I contatori del contenuto generato si basano su quattro elementi fondamentali:

1. La proprietà **counter-reset**, che inizializza il contatore, a cui viene assegnato un nome come identificativo univoco.
2. La proprietà **counter-increment**, che incrementa il contatore dell'unità specificata. Nel nostro caso non abbiamo specificato un unità, quindi viene incrementato di 1.
3. La funzione **counter()** (o la sua variante **counters()**), che viene posta all'interno della proprietà **content** per controllare il contatore.
4. La proprietà **content**, che inserisce materialmente il contatore nel layout.

Il contenuto così creato viene inserito prima o dopo il contenuto effettivo di un elemento tramite gli pseudo-elementi **:before** (prima) e **:after** (dopo).

Il nostro sommario presenta tre livelli di annidamento (1, 1.1, 1.1.1), quindi creeremo tre contatori distinti, nominandoli rispettivamente **item**, **item2** e **item3**. Vediamo in dettaglio il processo di inserimento dei contatori.

## 1. Righe 1-6

Viene creata l'istanza del contatore **item**, il cui ambito di applicazione è l'intero sommario (elemento **#toc**). Tutte le voci dell'elenco principale e degli elenchi annidati vengono privati del marcatore, che altrimenti si andrebbe a sovrapporre al contenuto generato.

## 2. Righe 7-14

Il contatore **item** viene inserito prima delle voci dell'elenco principale. Gli vengono assegnati un colore, un font ed un peso del font particolari, oltre ad un punto dopo di esso (".").

## 3. Righe 15-22

Viene creata l'istanza del contatore **item2** per il primo sottoelenco annidato. Il contatore viene quindi inserito prima del contenuto delle voci del primo sottoelenco, ma dopo il contatore **item**. Il peso del font viene ridotto. Si aggiunge un ulteriore punto dopo l'ultimo contatore.

#### 4. Righe 23-31

Viene creata l'istanza del contatore `item3` per il secondo sottoelenco annidato. Il contatore viene quindi inserito prima del contenuto delle voci del secondo sottoelenco, ma dopo il contatore `item2` e dopo il contatore `item`. Lo stile del font viene impostato sul corsivo, e si aggiunge uno spazio (" ") dopo l'ultimo contatore.

La soluzione proposta ha tuttavia dei notevoli limiti nel supporto offerto da Internet Explorer 7 (ed inferiori), dai browser testuali e dai lettori di schermo. Occorre quindi ripiegare su una soluzione alternativa, modificando radicalmente la marcatura ed il foglio di stile. Per la marcatura avremo:

```
<ul id="toc">
```

```
<li><span class="tocnum first">1.</span> <a href="#bg-overview">Background and Overview</a>
```

```
<ul>
```

```
<li><span class="tocnum">1.1</span> <a href="#intro">Introduction</a></li>
```

```
<li><span class="tocnum">1.2</span> <a href="#access">What is Accessibility?</a>
```

```
<ul>
```

```
<li><span class="tocnum third">1.2.1</span> <a href="#type-access">Types/Categories of
```

```
Access</a></li>
```

```
<li><span class="tocnum  
third">1.2.2</span> <a href="#type-  
disabilities">Types of Disabilities,  
and How they are Addressed</a></li>
```

```
</ul>
```

```
...omissis...
```

```
</ul>
```

Abbiamo sostituito gli elenchi ordinati con elenchi non ordinati, inserendo per ciascuna voce un elemento **span** aggiuntivo al fine di formattare la numerazione progressiva inserita direttamente nel sorgente. Gli stili diventano quindi i seguenti:

```
1. #toc li {  
2.     display: block;  
3. }  
  
4. #toc span.tocnum {  
5.     font-family: Georgia, serif;  
6.     color: #800;  
7.     background: transparent;  
8. }  
  
9. #toc span.tocnum.first {  
10.    font-weight: bold;  
11. }  
  
12. #toc span.tocnum.third {
```

```
13.     font-style: italic;
```

```
14. }
```

Nello specifico, siamo costretti ad usare classi ad hoc per ottenere gli stessi effetti ottenuti in precedenza tramite il contenuto generato. In un browser testuale come Lynx ogni singola voce avrà comunque il proprio marcatore oltre alla numerazione. È un prezzo da pagare se vogliamo mantenere una certa compatibilità. L'immagine che segue ci mostra i progressi ottenuti sino ad ora.

## [1. Background and Overview](#)

### [1.1 Introduction](#)

### [1.2 What is Accessibility?](#)

#### [1.2.1 Types/Categories of Access](#)

#### [1.2.2 Types of Disabilities, and How they are Addressed](#)

### [1.3 Importance of the Accessibility of the ODF Application](#)

### [1.4 Importance of Ensuring Authors Encode Accessibility Information into Documents](#)

### [1.5 Putting the pieces together](#)

## [2. ODF Application Accessibility](#)

### [2.1 Keyboard Navigation or use without a mouse](#)

### [2.2 Theme Support \(including OS fonts & colors\)](#)

### [2.3 Interoperability with Assistive Technologies](#)

#### [2.3.1 Characteristics of Engineered Accessibility Frameworks](#)

#### [2.3.2 Recommended Engineered Accessibility Frameworks](#)

#### [2.3.3 Supporting an Accessibility Framework](#)

#### [2.3.4 Dealing with the Absence of an Accessibility Frameworks](#)

### [2.4 Special Issues for Web-based ODF Applications](#)

### [2.5 ODF Help System Accessibility](#)

*Figura 13: Il sommario del documento*

[\[D\]](#)

# 25. Intestazioni, paragrafi ed elenchi

La maggior parte del contenuto del nostro documento è inserita all'interno di una sequenza regolare di intestazioni di vari livelli, paragrafi ed elenchi (ordinati e non). La formattazione delle intestazioni è già stata definita nelle regole di stile principali del nostro CSS. I paragrafi non hanno alcuna formattazione particolare. Si noti come sia nel caso delle intestazioni che dei paragrafi, i valori dei margini verticali vengono definiti dal foglio di stile predefinito del browser. Tipicamente tali valori sono i seguenti:

```
h1 {  
  margin: .67em 0;  
}  
  
h2 {  
  margin: .83em 0;  
}  
  
h3 {  
  margin: 1em 0;  
}
```



```
h4 {  
    margin: 1.33em 0;  
}  
  
h5 {  
    margin: 1.67em 0;  
}  
  
h6 {  
    margin: 2.33em 0;  
}  
  
p {  
    margin: 1em 0;  
}
```

Questi valori sono ricavati dal foglio di stile predefinito di Firefox (**html.css**), che si trova nella directory **/res** del programma.

Per quanto riguarda gli elenchi, solo alcuni di essi hanno bisogno di una formattazione particolare, che riportiamo di seguito.

```
1. ol.literal {  
2.     list-style-type: lower-alpha;  
3. }
```

```
4.  ul.squared {
5.      list-style-type: square;
6.  }

7.  ul.dash li {
8.      display: block;
9.      padding-bottom: 1em;
10. }

11. ul.dash li:before {
12.     content: "\2013";
13.     padding-right: 0.2em;
14. }
```

Impostiamo due classi (righe 1-6) per definire due tipi di marcatori diversi per un elenco ordinato e non ordinato. Nel primo caso (classe **.literal**) usiamo il valore **lower-alpha** della proprietà **list-style-type** per ottenere l'effetto dei marcatori costituiti da lettere minuscole dell'alfabeto. Nel secondo caso (classe **.squared**) usiamo invece il valore **square** per avere un marcatore costituito da un piccolo quadrato.

Nelle righe che vanno dalla 7 alla 14, usiamo il contenuto generato per inserire un trattino prima del contenuto delle voci dell'elenco con classe **.dash**. Ovviamente questo esempio non funziona in Internet Explorer 7 ed inferiori, quindi siamo costretti ad inserire il trattino nel sorgente. Possiamo osservarne l'effetto nell'immagine che segue.

- On UNIX systems, use the GNOME Accessibility framework. This can be done via one of several specific user interface toolkits, including GTK+, UNO, XUL, and Java/Swing. Or it can be done by implementing support for ATK or the Java Accessibility API directly, or by AT-SPI directly. In any case, it is highly likely that either ATK or AT-SPI support will need to be implemented for the editing/content portion of the ODF application. This is well supported by UNIX assistive technologies such as the Orca, LSR, and Gnopernicus screen reader/magnifiers, and the GNOME On-screen Keyboard.
- On the Java platform, use the Java Accessibility API. This can be done by using Java/Swing directly, or by implementing support for the Java Accessibility API directly. This is well supported on UNIX systems via the GNOME Accessibility framework (which bridges the Java Accessibility API out of the Java Virtual Machine), and is unfortunately only poorly supported on Microsoft Windows by assistive technologies via the Java Access Bridge for Windows.
- On the Macintosh (OS X v10.4 or later), use the Apple Accessibility API. This is supported by the built-in VoiceOver screen reader, and the built-in magnifier.
- For web-based ODF applications, use the WAI-ARIA interface, which today is supported by the Firefox 2.0 web browser on Microsoft Windows and several commercial assistive technology products including the JAWS and WindowEyes screen readers. Support for WAI-ARIA is anticipated on UNIX with the release of Firefox 3.0.

*Figura 14: L'elenco con le voci con il trattino*

[\[D\]](#)

## 26. Il codice di esempio

Nel nostro documento è presente anche del codice di esempio. Tale codice viene rappresentato in blocchi con sfondo grigio scuro e con carattere monospaziato. Il contrasto di colore, pur essendo in questo caso corretto, mal si combina con il tipo di carattere scelto, risultando a volte di difficile lettura. Per questo motivo abbiamo deciso di cambiare il contrasto di colore, pur mantenendo un font simile a quello originale.

Di norma quando si deve scegliere di formattare del codice si ricorre quasi sempre all'elemento **pre**. Tuttavia questa scelta non è la più indicata per una serie di ragioni:

## 1. Difficile manutenzione

L'elemento **pre** ha per impostazione predefinita il valore **pre** per la proprietà **white-space**. Ciò significa che le righe non andranno a capo a meno che non le si mandi a capo direttamente nel sorgente. In questo modo, se si cambiano le dimensioni del layout o si ridimensiona la finestra o l'utente decide di ridimensionare il font, il testo presente in tale elemento potrebbe fuoriuscire dall'elemento stesso. Questo fenomeno comporta, in Internet Explorer 6 ed inferiori, la comparsa di una barra di scorrimento orizzontale a determinate risoluzioni di schermo.

## 2. Mancanza di ordine logico nel codice

Se formattiamo il codice di esempio usando l'elemento **pre**, viene a mancare l'ordine logico nel codice usato. Se per esempio volessimo far riferimento ad una riga precisa nel codice, dovremmo aggiungere un elemento contenitore attorno a tale riga, formattarlo ed inserirvi un identificatore univoco. Attualmente i CSS possono selezionare solo la prima riga di un blocco (pseudo-elemento **:first-line**), ma non dispongono di un meccanismo per selezionare la n-esima riga di un blocco

Per questi motivi decidiamo di usare un elenco ordinato in cui

per ogni voce andremo ad inserire la corrispettiva riga di codice. La marcatura usata sarà la seguente:

```
<div class="code">
<ol>
  <li><H1 CLASS="western">Heading 1</H1>
  </li>
  <li><P>Sample text</P></li>
  <li><H2 CLASS="western">Heading 1.1</H2>
  </li>
  <li><P>More sample text</P></li>
  <li><H3 CLASS="western">Heading 1.1.1</H3>
  </li>
  <li><H1 CLASS="western">Heading 2</H1>
  </li>
  <li>Where did the class attribute come
  from?</li>
</ol>
</div>
```

Gli stili saranno i seguenti:

```
1. .code {
2.   padding: 0.5em;
3.   background: #e0e0e0;
4.   color: #000;
5.   font: 1em Courier, monospace;
6. }
```

```
7.  .code ol {
8.    padding: 0;
9.    margin-left: 2.5em;
10.   margin-right: 2.5em;
11.  }

12.  .code ol li {
13.    background: #f5f5f5;
14.    display: block;
15.    padding: 0.2em;
16.    margin-bottom: 3px;
17.  }
```

In questo codice usiamo un contenitore per circondare l'elenco ordinato. Volendo si può fare a meno di tale contenitore ed assegnare gli stili delle righe 1-6 direttamente all'elenco ordinato. Per eliminare il marcatore dell'elenco usiamo la tecnica vista in precedenza, ossia dichiariamo di blocco le voci dell'elenco. Tuttavia per ottenere una compatibilità anche con Internet Explorer 7 (ed inferiori) è necessario ricorrere alla strategia vista in precedenza, modificando la marcatura come segue:

```
<div class="code">
  <ol>
    <li><code>... </code></li>
    ...omissis
  </ol>
```

A questo punto si imposta il colore del testo dell'elemento **li** sullo stesso colore dello sfondo dell'elemento con classe **.code** e poi si reimposta il colore appropriato per l'elemento **code** che contiene il testo all'interno delle voci di elenco. Si noti che in questo caso abbiamo usato un elemento con valenza semantica (**code**), in quanto il testo all'interno delle voci dell'elenco è del codice di computer.

Da ultimo, per ricreare l'effetto tipico dell'indentazione delle righe di codice si può utilizzare una soluzione come la seguente:

```
1. .code li.indent1 {  
2. text-indent: 1em;  
3. }  
  
4. .code li.indent2 {  
5. text-indent: 2em;  
6. }  
  
7. /* eccetera */
```

Assegniamo una classe differente a ciascuna voce, con un'indentazione del testo progressiva. In questo caso, avendo dichiarato le voci dell'elenco come blocchi, possiamo applicare la proprietà **text-indent** alle voci. Nel caso volessimo distanziare un blocco di codice dall'altro, possiamo usare

questa soluzione:

```
1. .code li.vspace {  
2.     margin-bottom: 1em;  
3. }
```

Assegniamo un margine inferiore ad una particolare voce dell'elenco, che si distanzierà dalle altre della misura specificata. L'immagine che segue ci mostra il risultato ottenuto.

```
<text:h text:style-name="Heading_20_1" text:outline-level="1">Heading 1</text:h>  
<text:p text:style-name="Text_20_body">Sample text</text:p>  
<text:h text:style-name="Heading_20_2" text:outline-level="2">Heading 1.1</text:h>  
<text:p text:style-name="Text_20_body">More sample text</text:p>  
<text:h text:style-name="Heading_20_3" text:outline-level="3">Heading 1.1.1</text:h>  
<text:h text:style-name="Heading_20_1" text:outline-level="1">Heading 2</text:h>
```

*Figura 15: I blocchi di codice formattati*

[\[D\]](#)

## 27. Le tabelle di esempio

Il documento presenta due tipi di tabelle di esempio: un tipo puramente presentazionale, senza dati di effettiva rilevanza ed un tipo strutturale con dati effettivi. La formattazione richiede quindi degli stili completamente diversi. È importante notare a



questo punto che le tabelle del primo tipo non sono di fatto necessarie alla comprensione del testo, e quindi potrebbero essere sostituite da immagini con opportuna descrizione. Decidiamo di inserirle come marcatura a scopo puramente dimostrativo. Nel documento originale tali tabelle si presentano come segue.

A1	B1	C1
A2	.B2.A1	.B2.B1
	.B2.A2	

Sample 1

*Figura 16: Una tabella di esempio*

[\[D\]](#)

La marcatura utilizzata per rendere questo tipo di tabella è la seguente:

```
<table summary="Example table" class="example-table">
```

```
<tr>
```

```
<td class="a1">A1</td>
```

```
<td class="b1">B1</td>
```

```
<td class="c1">C1</td>
```

```
</tr>
```

```
<tr>
```

```
<td class="a2">A2</td>
```

```
<td class="table" colspan="2">
```

```

<table summary="Nested table">
  <tr>
    <td class="b2a1">.B2.A1</td>
    <td class="b2b1">.B2.B1</td>
  </tr>
  <tr>
    <td class="b2a2"
      colspan="2">.B2.A2</td>
  </tr>
</table>
</td>
</tr>
</table>
<p class="caption">Sample 1</p>

```

Abbiamo usato l'attributo **summary** per dare informazioni aggiuntive sulle tabelle presentate. In questo caso avremmo potuto anche aggiungere una breve nota sul fatto che le tabelle sono puramente dimostrative e non presentano dati essenziali.

La didascalia in fondo alla tabella non è stata marcata con l'elemento **caption**, come ci si attenderebbe, per un motivo molto semplice: attualmente per ottenere l'effetto della didascalia in fondo alla tabella si sarebbe dovuto ricorrere alla dichiarazione **caption-side: bottom**, che non è ampiamente supportata dai browser. Questo è un altro motivo per cui tabelle dimostrative come questa andrebbero evitate

ove possibile.

A questo punto passiamo all'applicazione degli stili:

```
1.  table.example-table {
2.    width: 450px;
3.    border-collapse: collapse;
4.    border: 1px solid #000;
5.    table-layout: fixed;
6.    background: #ccc;
7.    color: #000;
8.  }

9.  table.example-table td {
10.   border: 1px solid;
11.   vertical-align: middle;
12.   text-align: center;
13.   padding: 0;
14.  }

15. table.example-table .a1, table.example-
    table .a2 {
16.   width: 90px;
17.   font-weight: bold;
18.  }

19. table.example-table .b1 {
20.   width: 110px;
21.   font-weight: bold;
```

```
22. }
23. table.example-table .c1 {
24.     width: 250px;
25.     font-weight: bold;
26. }
27. table.example-table .table {
28.     width: 360px;
29. }
30. table.example-table table {
31.     background: #eee;
32.     border-collapse: collapse;
33.     width: 360px;
34.     table-layout: fixed;
35. }
36. table.example-table table .b2a1 {
37.     border-width: 0 1px 0 0;
38.     width: 110px;
39. }
40. table.example-table table .b2b1 {
41.     border-width: 0 0 1px 0;
42.     width: 250px;
43. }
44. table.example-table table .b2a2 {
```

```
45.     width: 360px;
46.     border-width: 1px 0 0 0;
47. }

48. .caption {
49.     font-size: 0.85em;
50.     font-style: italic;
51.     margin: 0;
52.     padding-top: 2px;
53. }
```

Le 53 righe di codice necessario alla formattazione di questo tipo di tabelle ci convincono del fatto che un approccio del genere è controproducente. Tuttavia, il codice usato presenta alcune caratteristiche interessanti nella formattazione delle tabelle.

1. L'algoritmo del layout delle due tabelle è impostato su **fixed**. Questo significa che la somma complessiva della larghezza delle singole celle (compresi i bordi), dovrà essere uguale alla larghezza generale impostata sulla prima tabella (450 pixel). Si noti comunque che anche se si supera tale larghezza, i browser si limiteranno semplicemente ad allargare la tabella principale, senza nascondere il contenuto.
2. Le tabelle presentano un modello di bordi collassato (**border-collapse: collapse**), che fa in modo che i

bordi delle singole celle si fondono insieme per formare un unico bordo.

Tuttavia, Internet Explorer 7 (ed inferiori), presenta un problema nel collassamento dei bordi, per cui si rende necessario aggiungere un commento condizionale con il seguente codice:

```
1. <--[if lt IE 8]>
2. <style type="text/css" media="screen">
3.   table.example-table table .b2a1 {
4.     border-bottom: 1px solid;
5.   }
6. </style>
7. <![endif]-->
```

Il commento condizionale viene indirizzato alle versioni di Internet Explorer inferiori alla 8 (riga 1). Il risultato si può osservare nell'immagine che segue.

A1	B1	C1
A2	.B2.A1	.B2.B1
	.B2.A2	

Sample 1

*Figura 17: La tabella di esempio realizzata con i fogli di stile*

[\[D\]](#)

Passiamo alle tabelle di dati effettivi. Nel documento originale,

esse hanno il seguente layout:

<i>ODF 1.1 Element</i>	<i>HTML element using alt=</i>	<i>HTML element using title=</i>
<draw:path> <draw:circle> <draw:ellipse> <draw:g> <draw:page-thumbnail> <draw:frame> <draw:measure> <draw:caption> <draw:connector> <draw:control> <dr3d:scene> <draw-custom-shape>		
<text:a>	<a>	
<draw:layer>		
<draw:image>	<IMG alt=	
draw:area-rectangle	<AREA Shape="rect"	

*Figura 18: La tabella di dati effettivi*

[\[D\]](#)

La marcatura necessaria è la seguente:

```

<table summary="svg:title" class="code-
example">
  <tr>
    <th scope="col">ODF 1.1 Element</th>
    <th scope="col">HTML element using alt=
</th>
    <th scope="col">HTML element using title=
</th>

```

```
</tr>
```

```
<tr>
```

```
<td><draw:rect><br />
```

```
<draw:line><br />
```

```
<draw:polyline><br />
```

```
<draw:polygon><br />
```

```
<draw:regular-polygon></td>
```

```
<td><code><IMG alt=></code></td>
```

```
<td></td>
```

```
</tr>
```

```
...omissis...
```

```
</table>
```

In questa tabella abbiamo usato l'attributo **scope** sulle intestazioni di tabella in modo da associare ciascuna colonna ad un'intestazione. Abbiamo usato un elemento presentazionale (**br**) per mandare a capo il testo. Volendo si sarebbe potuta inserire ogni riga di testo in un elemento **span** e poi trasformarlo in blocco. Gli stili, meno ridondanti di quelli della precedente tabella, sono:

1. **table.code-example {**
2. **border: 1px solid #000;**
3. **border-collapse: collapse;**
4. **}**
5. **table.code-example td,**
6. **table.code-example th {**



```

7.   width: 30%;
8.   border: 1px solid #000;
9.   padding: 0.2em;
10.  vertical-align: middle;
11.  }

```

In questo caso viene usato il layout automatico di tabella, e il risultato è quello di una tabella fluida. Possiamo osservarlo nell'immagine che segue.

ODF 1.1 Element	HTML element using alt=	HTML element using title=
<draw.rect> <draw.line> <draw.polyline> <draw.polygon> <draw.regular-polygon>	<IMG alt=>	
<draw.path> <draw.circle> <draw.ellipse> <draw.g> <draw.page-thumbnail> <draw.frame> <draw.measure> <draw.caption> <draw.connector> <draw.control> <dr3d.scene> <draw-custom-shape>		

*Figura 19: La tabella di dati effettivi realizzata con i fogli di stile*

[\[D\]](#)

## 28. Le immagini di esempio

Il documento originale presenta anche delle immagini di esempio, solitamente allineate a sinistra rispetto al testo dei paragrafi e con una breve didascalia sottostante. Ne vediamo un esempio nell'immagine che segue.



*Figura 20: Un'immagine di esempio*

[\[D\]](#)

La marcatura usata per questo elemento è la seguente:

```
<span class="left">  
    
  <span>Illustration 1:<br /> A picture of a  
  pig</span>  
</span>
```

Ovviamente anche in questo caso sarebbe opportuno inserire una descrizione estesa dell'immagine, al fine di aumentare l'accessibilità della pagina. Gli stili relativi sono i seguenti:

```
1. .left {
```

```
2.   float: left;
3.   margin: 0 0.5em 0.5em 0;
4.   border: 1px solid;
5.   font-size: 0.85em;
6.   padding-top: 0.25em;
7. }

8. .left img {
9.   margin: 0 auto;
10.  display: block;
11. }

12. .left span {
13.  display: block;
14.  font-style: italic;
15.  padding: 0.25em;
16. }

17. .left.noborder {
18.  border: none
19. }
```

L'elemento **span** con classe **.left** è stato flottato a sinistra, gli sono stati attribuiti dei margini, un bordo ed il suo font è stato ridimensionato (righe 1-7). Si noti come l'elemento flottato non abbia una dimensione esplicita. In questo caso, le specifiche prevedono che venga applicato l'algoritmo *shrink-to-fit*, ossia la dimensione dell'elemento verrà determinata dal

contenuto presente al suo interno. Questo permette di fatto di avere dei float liquidi, che avranno le dimensioni necessarie ad ospitare il loro contenuto.

Successivamente (righe 8-11) abbiamo reso di blocco l'immagine e l'abbiamo centrata all'interno del suo blocco contenitore tramite i margini orizzontali automatici. Quindi abbiamo formattato la didascalia (righe 12-16), rendendola di blocco, cambiando lo stile del font ed assegnandole del padding.

Infine (righe 17-19) abbiamo creato una classe ad hoc per eliminare il bordo dalle immagini che non ne hanno bisogno. Si noti la sintassi usata, con i nomi delle classi scritti in sequenza (classi multiple).

Possiamo apprezzare l'effetto finale nell'immagine che segue.



When a non-visual user re  
within it. Attached to this |  
technology, the provision  
is what the author genera  
(generally a textual altern:

*Figura 21: Un'immagine di esempio formattata con i CSS*

[\[D\]](#)

Purtroppo Internet Explorer 6 (ed inferiori) presenta dei problemi nel computo delle dimensioni dell'elemento flottato. È necessario modificare il codice CSS come segue:

```
1.  .left {
2.    float: left;
3.    margin: 0 0.5em 0.5em 0;
4.    border: 1px solid;
5.    font-size: 0.85em;
6.    padding-top: 0.25em;
7.    text-align: center;
8.  }

9.  .left span {
10.   display: block;
11.   font-style: italic;
12.   padding: 0.25em;
13.   text-align: left;
14.  }
```

In pratica si deve evitare di centrare l'immagine tramite margini orizzontali automatici, ricorrendo invece alla dichiarazione **text-align: center** (riga 7). Ovviamente in tal senso l'immagine non va dichiarata come blocco.

## 29. Il piè di pagina con i ringraziamenti e le note finali

Il piè di pagina è molto simile all'elenco degli autori e della

versione del documento visto in precedenza. La marcatura usata in questo caso è la seguente:

```
<h2 id="appendix-a">Appendix A.  
Acknowledgments</h2>
```

```
<dl id="ack">
```

```
<dt><strong>Contributors:</strong></dt>
```

```
<dd>
```

```
<ul>
```

```
<li>Stephen Noble, Design Science, Inc.  
</li>
```

```
<li>Chieko Asakawa, IBM</li>
```

```
<li>Pete Brunet, IBM</li>
```

```
<li>Hiro Takagi, IBM</li>
```

```
<li>Richard Schwerdtfeger, IBM</li>
```

```
<li>Janina Sajka, Individual</li>
```

```
<li>David Clark, Institute for  
Community Inclusion</li>
```

```
<li>David Pawson, Royal National  
Institute for the Blind</li>
```

```
<li>Peter Korn, Sun Microsystems, Inc.  
</li>
```

```
<li>Malte Timmermann, Sun Microsystems,  
Inc.</li>
```

```
</ul>
```

```
</dd>
```

```
</dl>
```

```
<h2 id="appendix-b">Appendix B. Notices</h2>
```

```
<div id="notices">
```

```
<p>Copyright &copy; OASIS Open 2006. All  
Rights Reserved.</p>
```

```
<p><span class="underline">All capitalized  
terms...</p>
```

```
</div>
```

Gli stili da applicare sono estremamente semplici:

```
1. dl#ack dt strong {  
2.   color: #339;  
3.   background: transparent;  
4. }  
  
5. dl#ack, #notices {  
6.   font-size: 0.85em;  
7. }  
  
8. #notices .underline {  
9.   border-bottom: 1px solid #555;  
10. }  
  
11. #ack ul {  
12.   margin-left: 0;  
13.   padding-left: 0;  
14. }  
  
15. #ack ul li {  
16.   display: block;
```

Le righe 8-10 sono interessanti: in questo caso veniva richiesta una sottolineatura del testo, che abbiamo realizzato usando un bordo inferiore. Tale soluzione si rivela più incisiva della dichiarazione **text-decoration: underline**, in quanto quest'ultima non ci permette di controllare lo spazio tra la linea di base del testo e la sottolineatura. L'immagine che segue mostra il risultato ottenuto.

## Appendix A. Acknowledgments

### Contributors:

Stephen Noble, Design Science, Inc.  
Chieko Asakawa, IBM  
Pete Brunet, IBM  
Hiro Takagi, IBM  
Richard Schwerdtfeger, IBM  
Janina Sajka, Individual  
David Clark, Institute for Community Inclusion  
David Pawson, Royal National Institute for the Blind  
Peter Korn, Sun Microsystems, Inc.  
Malte Timmermann, Sun Microsystems, Inc.

## Appendix B. Notices

Copyright © OASIS Open 2006. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

*Figura 22: Il piè di pagina e le note finali*

[D]

# 30. Conclusioni

Alla fine del nostro percorso ci troviamo di fatto di fronte ad un



bivio. Infatti abbiamo creato due documenti, diversi nella struttura e negli stili applicati.

1. Il [primo documento](#), servito come **application/xhtml+xml** è compatibile con i browser che attualmente supportano maggiormente gli standard. Abbiamo quindi una compatibilità nel presente e nel futuro.
2. Il [secondo documento](#), servito come **text/html** è compatibile con tutti i browser, anche con quelli dal supporto agli standard più problematico. Abbiamo quindi una compatibilità nel presente, nel passato e nel futuro.

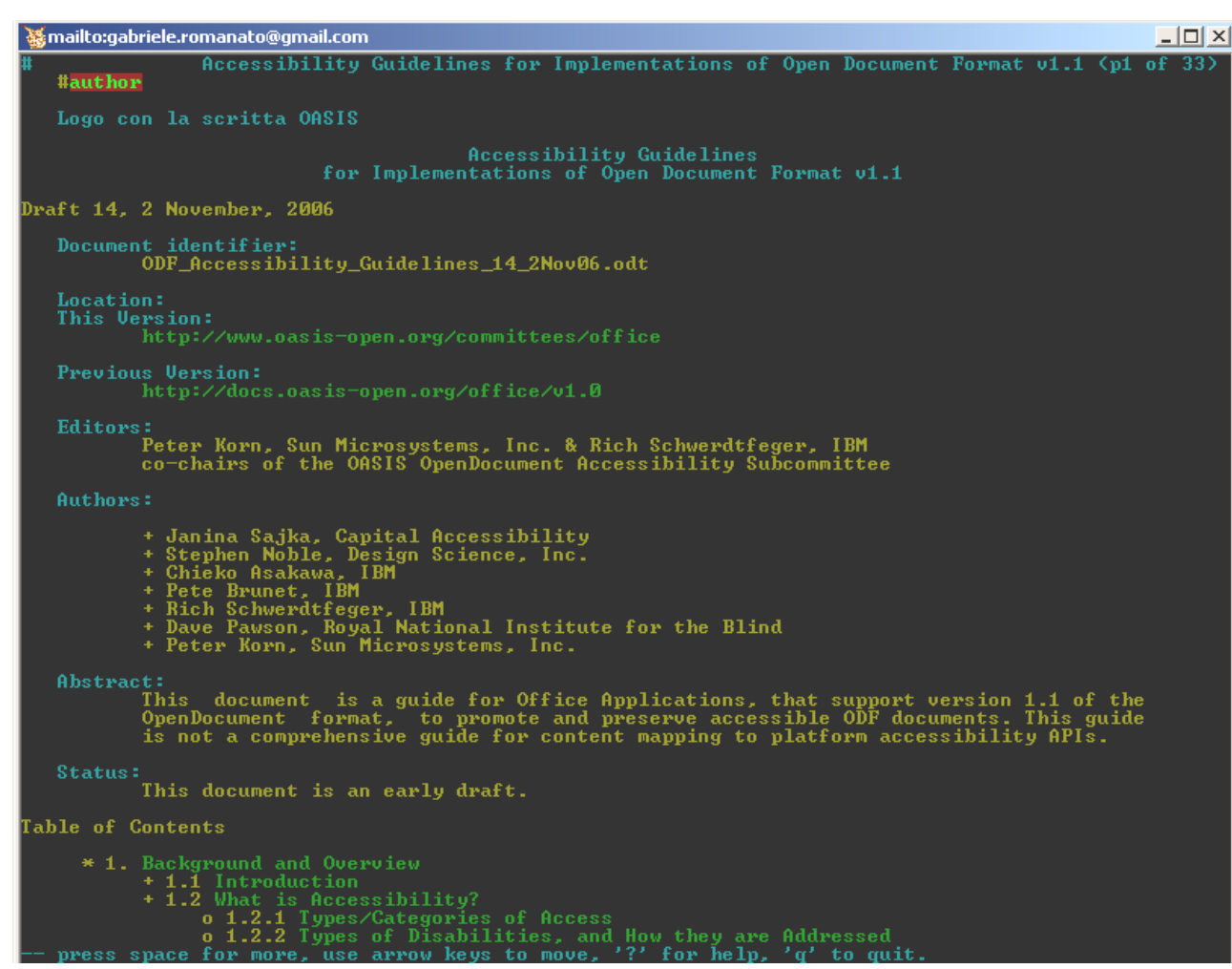
Quale scegliere? A nostro avviso si dovrebbe operare, se possibile, una scelta onnicomprensiva, fornendo il secondo documento come documento principale ed il primo come alternativa per quei browser che supportano il tipo MIME specificato. Come alternativa praticabile, si può optare per la negoziazione del contenuto, come visto in precedenza.

Scrivere un documento servito come **application/xhtml+xml** ci abitua ad un rigore formale e ad una precisione non sempre possibile nel formato **text/html**, data la già citata permissività dei browser con quest'ultimo formato.

Tale rigore è necessario se pensiamo al futuro sviluppo del Web e alle nuove tecnologie che si stanno affermando. Esse

richiedono una disciplina nella scrittura del codice impensabile fino a qualche tempo fa. Prepararci al futuro è il miglior modo per affrontare le sfide del presente.

Da ultimo, osserviamo il risultato ottenuto in un browser testuale come Lynx:



```
mailto:gabriele.romanato@gmail.com
# Accessibility Guidelines for Implementations of Open Document Format v1.1 (pl of 33)
#author
Logo con la scritta OASIS
Accessibility Guidelines
for Implementations of Open Document Format v1.1
Draft 14, 2 November, 2006
Document identifier:
ODF_Accessibility_Guidelines_14_2Nov06.odt
Location:
This Version:
http://www.oasis-open.org/committees/office
Previous Version:
http://docs.oasis-open.org/office/v1.0
Editors:
Peter Korn, Sun Microsystems, Inc. & Rich Schwerdtfeger, IBM
co-chairs of the OASIS OpenDocument Accessibility Subcommittee
Authors:
+ Janina Sajka, Capital Accessibility
+ Stephen Noble, Design Science, Inc.
+ Chieko Asakawa, IBM
+ Pete Brunet, IBM
+ Rich Schwerdtfeger, IBM
+ Dave Pawson, Royal National Institute for the Blind
+ Peter Korn, Sun Microsystems, Inc.
Abstract:
This document is a guide for Office Applications, that support version 1.1 of the
OpenDocument format, to promote and preserve accessible ODF documents. This guide
is not a comprehensive guide for content mapping to platform accessibility APIs.
Status:
This document is an early draft.
Table of Contents
* 1. Background and Overview
+ 1.1 Introduction
+ 1.2 What is Accessibility?
o 1.2.1 Types/Categories of Access
o 1.2.2 Types of Disabilities, and How they are Addressed
-- press space for more, use arrow keys to move, '?' for help, 'q' to quit.
```

Figura 23: Il documento mostrato in Lynx