

Sistemi per l'elaborazione dell'informazione 2

Sfoggia la rete anche tu! (ma meglio...)

In quest'ultima esercitazione dovete adattare il browser sviluppato in quella precedente, utilizzando la classe `ParseDelegator` vista a lezione per effettuare l'analisi dei documenti scaricati.

Analisi del documento

Reimplementate i metodi di analisi del documento visti nell'esercitazione precedente utilizzando la classe `ParserDelegator` e i *callback*.

Attenzione: i link in una pagina possono essere *relativi*, cioè possono non contenere tutte le parti di un URL. In tal caso, i link vanno *risolti* relativamente all'URL della pagina in cui sono contenuti. La risoluzione viene effettuata automaticamente dal costruttore di `URL` che accetta come argomento, oltre al testo che descrive l'URL, anche un altro URL di *contesto* (che sarebbe poi l'URL della pagina corrente). Per esempio, se nella pagina

```
http://homes.dsi.unimi.it/~malchiod/P3/laboratorio/link-tree.html
```

viene trovato un URL della forma `link-uno.html`, l'URL risultante sarà

```
http://homes.dsi.unimi.it/~malchiod/P3/laboratorio/link-uno.html
```

Se vi resta del tempo, sbizzarritevi a scrivere altri *callback*. Per esempio:

1. un *callback* che stampa solo i titoli;
2. un *callback* che inserisce uno spazio tra un carattere e l'altro del contenuto testuale degli elementi `h1`, e mette un asterisco all'inizio di ogni elemento `li`;
3. un *callback* che stampa tutti i riferimenti a immagini (i valori dati agli attributi `src` degli elementi `img`);
4. un *callback* che estrae il testo, ma elimina tutti i tag; inoltre, rende interamente maiuscolo il titolo e il contenuto di tutte le intestazioni (elementi `h1`, `h2`, ecc.);
5. un *callback* che salva in un file (passato come argomento nel costruttore del *callback*) il documento, ma inverte il testo contenuto nel titolo e nelle intestazioni (potete utilizzare, a tal fine, il metodo `reverse()` della classe `StringBuffer`).

Dialogare con il server a basso livello

Se vi sentite particolarmente arditi, aprite direttamente la socket (per ottenere l'host potete utilizzare il metodo `getHost()` di `URL`), ottenete gli stream di input e output e gestite manualmente il protocollo. Potete imitare l'istruzione `GET` che, per esempio, fa riferimento alla pagina del corso, inviando sullo stream la seguente stringa:

```
"GET /~malchiod/SEI2/index.php HTTP/1.1\r\nHost: homes.dsi.unimi.it\r\n\r\n"
```

(una nota: l'intestazione `Host` presente immediatamente dopo la `GET` è necessaria per la versione 1.1 del protocollo HTTP). Attenti a copiare attentamente i caratteri di fine riga e ritorno carrello, e a chiamare `flush()` sullo stream di output.

Nei dati restituiti, compare innanzitutto la riga con il messaggio di risposta, seguito immediatamente dalle intestazioni HTTP, una per riga. Segue una riga vuota, e quindi il contenuto della pagina. Attenzione però: il contenuto potrebbe essere *chunked*, cioè diviso in blocchi preceduti dalla loro lunghezza in esadecimale (la lunghezza compare su una riga a parte). Quindi, se tra le intestazioni HTTP compare **Transfer-Encoding: chunked**, dovete iterativamente leggere una riga, interpretarla come numero esadecimale, leggere il numero di byte corrispondenti, leggere di nuovo una riga e interpretarla come numero esadecimale, e così via finché la lunghezza letta non è 0. È una buona idea farsi stampare tutti i byte emessi dalla connessione per controllare che cosa vi viene restituito effettivamente (purtroppo `telnet` non funziona a causa del proxy).

Attenzione: l'accesso diretto alla rete tramite socket potrebbe non funzionare dal laboratorio anche configurando correttamente il proxy. Si consiglia in tal caso provare questo metodo da casa. In questo caso potete anche utilizzare prima `telnet` per impratichirvi con i messaggi stampati dal protocollo.