

Sistemi per l'elaborazione dell'informazione 2

Sfogliala la rete anche tu!

Lo scopo di questa esercitazione è creare un semplice browser che permette di scaricare una pagina HTML da un server web e visualizzarla in formato testuale. Il progetto è molto articolato, e potete svolgerlo in vari stadi. Al solito, niente panico: riutilizzate il più possibile codice che avete già realizzato, soprattutto per quel che riguarda la parte sul disegno MVC.

Nota bene: la rete del laboratorio non è disponibile direttamente, ma solo tramite proxy. Questo non è un problema di per sé, ma richiede la seguente opzione alla JVM con cui lanciate l'applicazione:

```
-Dhttp.proxyHost=159.149.43.252 -Dhttp.proxyPort=3128
```

Il browser

Dovete creare un'applicazione MVC che permetta di osservare una pagina web, dotata (per ora) di una sola vista, contenente tre controlli:

- un controllo di testo a linea singola, dove un utente può scrivere un URL;
- un pulsante “Go” (o qualcosa del genere);
- un controllo di testo a sola lettura e multilinea, cioè costruito usando le flag

```
SWT.BORDER | SWT.H_SCROLL | SWT.V_SCROLL | SWT.MULTI | SWT.READ_ONLY | SWT.WRAP,
```

dove verrà visualizzata la pagina scaricata.

Per scaricare la pagina ci sono, come visto a lezione, due modi:

1. semplice ma poco flessibile: usate il metodo `openStream()` della classe `URL`;
2. semplice e flessibile: usate il metodo `openConnection()` della classe `URL` per ottenere un oggetto della classe `URLConnection`, e ottenete da questo un `InputStream` tramite `getInputStream()` (il vantaggio di questo metodo è che più tardi potrete ottenere facilmente le intestazioni, i messaggi di risposta e così via);

Una volta che avete avuto accesso alla pagina, dovete memorizzare nel modello il contenuto e tutti i dati che vi interessano, e aggiornare la vista come indicato nei paragrafi che seguono.

Parsing semplificato

Dovete scorrere il documento e visualizzarlo nella finestra senza i tag. Ricordate che dovete rimpiazzare eventuali sequenze di spazi bianchi (spazi, tabulazioni e newline) con un unico spazio. Non preoccupatevi troppo della resa del documento HTML: per iniziare, è sufficiente inserire due caratteri di newline ogni volta che incontrate un tag `<p>` e un carattere di newline in corrispondenza di `<br>`. Assumete che il documento

che leggete sia ben formato, e cioè che i tag siano aperti e chiusi in modo corretto. Tenuto a mente che molte pagine che potreste tentare di visualizzare potrebbero violare l'assunzione sopra menzionata, è meglio fare riferimento alla pagina del corso o alla pagina del W3 consortium (<http://www.w3.org>), il cui codice HTML è sicuramente ben formato.

Questo semplice algoritmo di resa delle pagine HTML può essere arricchito a piacere, per esempio:

- aggiungendo un carattere di newline (`\n`) alla fine di ogni intestazione (e ogni volta che sia ragionevole farlo);
- evidenziando i titoli di paragrafo, sotto-paragrafo, ecc. (rendendo maiuscoli tutti i loro caratteri, oppure inserendo uno spazio tra un carattere e l'altro);
- inserendo un asterisco all'inizio di ogni tag `<li>`, e così via.

Molti dei costruttori e dei metodi che dovete usare lanciano eccezioni opportune per tutti i comportamenti indesiderati (fallimenti nel DNS, URL mal formati, ecc.). Dovete tracciare le eccezioni e riportare in maniera ragionevole gli errori all'utente (per esempio, all'interno del controllo di testo multilinea). Usate lo stesso approccio per segnalare che il documento che state tentando di visualizzare non è basato su codice HTML ben formato.

Parsing avanzato

Aggiungete al browser un tasto che apre una nuova finestra contenente la struttura ad albero dei collegamenti all'interno del documento corrente. Per fare questo dovete scorrere il documento e riconoscere le cosiddette ancore (i tag `<a>`), selezionare il valore dell'attributo `href` e costruire un nuovo oggetto URL (facendo attenzione alla possibilità che i valori di `href` facciano riferimento a URL relativi) a partire dall'argomento. Una volta ottenuto questo oggetto, dovete aprire il documento corrispondente ed eseguire ricorsivamente il parsing delle ancore, fino a un livello di profondità definito da una costante `MAX_LEVEL`.

I collegamenti devono essere visualizzati tramite un composto **Tree** (illustrato nel tutorial sui widget avanzati indicato nel sito del corso) che permette di visualizzare strutture ad albero. La pagina iniziale sarà la radice dell'albero, mentre i collegamenti in essa presenti corrisponderanno ai figli. Ogni documento figlio può a sua volta avere dei collegamenti (e quindi dei figli). Questa costruzione verrà iterata nella struttura **Tree** fino al livello predefinito (ricordatevi l'espedito che abbiamo visto a lezione per evitare i cicli quando si visita un composto).

Nota bene: il parsing di documenti HTML in Java viene fatto tramite metodi di classi opportune che tengono conto di tutte le peculiarità del linguaggio HTML. Il parsing che viene richiesto qui è molto semplice e non considera gran parte di queste peculiarità. Per questo motivo potrebbe non funzionare su alcuni documenti web. Inoltre, è molto facile ottenere un albero dei collegamenti molto grande anche considerando solo due o tre livelli. Per ovviare a questo, eseguite inizialmente i vostri test utilizzando questo documento pubblicato sul sito del corso:

<http://homes.dsi.unimi.it/~malchiod/SEI2/laboratorio/link-tree.html>

Altre viste

Potete aggiungere una vista (aperta su richiesta da un pulsante opportuno) che mostri una lista contenente tutte le intestazioni HTTP della pagina appena letta; chiaramente, dovete usare il secondo metodo di accesso (o quello descritto più avanti), usando il metodo `getHeaderFields()` di `URLConnection`.

L'interfaccia in questo caso può essere basata su una semplice casella di testo multilinea, oppure su una lista, o ancora su una tabella.

Parte facoltativa

Link ipertestuali

Potete aggiungere un'interessante funzionalità alla vista che mostra l'albero dei collegamenti ipertestuali, facendo in modo che quando l'utente seleziona uno degli elementi dell'albero venga caricato il corrispondente documento HTML.

Sempre più difficile

I più avventurosi possono misurarsi anche con un ulteriore procedimento, complesso ma molto flessibile: aprite direttamente la socket (per ottenere lo host potete utilizzare il metodo `getHost()` di `URL`), ottenete gli stream di input e output e gestite manualmente il protocollo. Per esempio, potete collegarvi al server all'indirizzo `homes.dsi.unimi.it` e imitare l'istruzione `GET`:

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

al fine di ottenere la pagina del corso (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 cos 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 di usare il primo o il secondo metodo (potete provare il terzo da casa).