SCANLOG (analisi file di log HTTP)

Marco Ubertone La Rocca (302908)

MAGGIO 2011

Indice

Capitolo 1

Introduzione

Il progetto richiesto consiste nello sviluppare un'applicazione, la quale deve essere in grado di analizzare file di log HTTP, estrapolandone i contenuti, al fine di monitorare il traffico HTTP dei client. Per "traffico" si intente i siti visitati, il tempo ed i byte totali della navigazione di ciascun client.
Per ottenere gli obiettivi prefissati si è implementata una funzione di parsing sul file di log HTTP. Lo scopo di tale funzione è quello di suddividere il file in input in parti più piccole in modo da poter raccogliere le informazioni necessarie all'analisi del file stesso.

1.1 Parser

In informatica, il parsing o analisi sintattica è il processo atto ad analizzare uno stream continuo in input (letto per esempio da un file o una tastiera) in modo da determinare la sua struttura grammaticale grazie ad una data grammatica formale. Un parser è un programma che esegue questo compito.

Di solito i parser non sono scritti a mano ma generati attraverso dei generatori di parser.

Tipicamente, il termine italiano viene utilizzato per riferirsi al riconoscimento di una grammatica e alla conseguente costruzione di un albero sintattico, che mostra le regole utilizzate durante il riconoscimento dall'input; l'albero sintattico viene poi visitato (anche più volte) durante l'esecuzione di un'interprete o di un compilatore.

Nella maggior parte dei linguaggi, tuttavia, l'analisi sintattica opera su una sequenza di token in cui l'analizzatore lessicale spezzetta l'input. Pertanto, il termine inglese spesso viene usato per indicare l'insieme della analisi lessicale e della analisi sintattica vera e propria.

1.1.1 Tipi di parser

Il lavoro del parser è essenzialmente quello di determinare se e come l'input può essere derivato dal simbolo iniziale con le regole della grammatica formale. Questo può essere fatto essenzialmente in due modi:

- Analisi top-down - Un parser può partire con il simbolo iniziale e cercare di trasformarlo nell'input. Intuitivamente, il parser parte dal più grande elemento e lo divide in parti sempre più piccole.

- Analisi bottom-up - Un parser può partire con l'input e cercare di riscriverlo sino al simbolo iniziale. Intuitivamente, il parser cerca di trovare il più elementare simbolo, quindi elabora gli elementi che lo contengono, e così via.

1.1.2 Linguaggi di programmazione

Genericamente i parser sono utilizzati con i linguaggi di programmazione, i quali hanno delle grammatiche semplici e regolari; i parser di questo tipo tendono ad essere basati su grammatiche libere dal contesto poiché con queste grammatiche si possono scrivere parser veloci ed efficienti.

In realtà, le grammatiche libere dal contesto non riescono a descrivere da sole la maggior parte dei linguaggi di programmazione di uso comune. Informalmente, la ragione è che la memoria di ogni linguaggio è limitata; la grammatica non può ricordare la presenza di un costrutto dopo un'arbitraria lunghezza in input, è necessario per esempio in quei linguaggi dove i nomi possono essere referenziati.

Usare grammatiche più potenti quali quelle grammatiche dipendenti dal contesto, tuttavia, perdono in efficienza. Di conseguenza è una strategia comune quella di utilizzare grammatiche libere dal contesto per una versione rilassata (con minori vincoli) del linguaggio. Queste grammatiche accetteranno tutti i costrutti validi del linguaggio in considerazione, oltre ad altri costrutti non validi che vengono scartati durante l'analisi semantica dell'input. Per esempio, la grammatica potrebbe accettare un programma che usa una variabile non dichiarata in precedenza.

1.2 Struttura relazione

1. Applicazione .....
    - Specifiche file di log HTTP .....
    - Struttura applicazione .....
    - Funzionamento .....
    - Grafica .....
2. Manuale d'uso .....
3. Conclusioni .....

Capitolo 2

Applicazione

2.1 Specifiche file di log HTTP

Per implementare un parser apposito per file di log HTTP, dobbiamo prima definire come quest'ultimi sono formattati; un generico file di log HTTP è costituito dai seguenti campi:

- Client: indica l'indirizzo IP del client;
- Server: indica l'indirizzo del sito visitato;
- Protocol: indica il protocollo di trasferimento a livello di applicazione utilizzato;
- Method: indica il tipo di richiesta HTTP effettuata;
- URL: indica la pagina visita;
- HTTPReturnCode: indica il risultato della richiesta HTTP effettuata;
- Location;

- Referer;

- UserAgent: indica l'applicazione client che si connette al server;

- ContentType: indica le varie tipologie di estensioni;

- Bytes: indica i byte prodotti dalla navigazione per quella pagina;

- BeginTime: indica il tempo d'inizio della richiesta HTTP;

- EndTime: indica il tempo in cui si conclude la richiesta HTTP.

Tali campi sono separati da un token "TAB"; questo permetterà al parser di suddividere correttamente l'input prelevato dal file di log HTTP.

2.2 Struttura applicazione

L'applicazione è costituita da 2 package. Il primo (default packacge) contenente 6 classi dove quella principale (contenente il main) è Scanlog.java, mentre le restanti 5 vengono utilizzate per implementare i vari oggetti utili all'analisi del file di log proposto. Il secondo (images) contenente le immagini utilizzate per lo sviluppo dell'interfaccia grafica dell'applicativo.

2.2.1 Classi

1. **Scanlog.java**: questa è il centro dell'applicazione vera e propria in quanto contiene il main ed alcune tra le funzioni più rilevanti. All'avvio del programma, costruisce un'interfaccia grafica per l'utente (tramite la funzione *createAndShowGUI()*) in modo tale da rendere chiaro ed

immediato il funzionamento dell'applicativo ed il suo scopo. L'interfaccia grafica è costruita tramite l'ausilio delle librerie AWT, SWING e all'utilizzo dei relativi oggetti.

Per quanto riguarda le funzioni implementate da tale classe, le più importanti sono quelle adibite all'analisi del file di log HTTP ed al salvataggio di un file di report. La funzione di analisi (*analyzing()*) e quella relativa al salvataggio (*rescue()*) sono entrambe richiamate dalla funzione *actionPerformed(ActionEvent e)*, la quale gestisce la selezione dei pulsanti di azione del programma stesso.

Tale gestione prevede:

- la selezione del file di log HTTP;
- l'esecuzione dell'analisi;
- il salvataggio dei risultati ottenuti dall'analisi in un file di report.

Queste 3 azioni devono essere eseguite esattamente in quest'ordine in quanto non sarà possibile eseguire un'analisi se nessun file di log HTTP è stato scelto e, non sarà possibile salvare un file di report se non è stata eseguita nessun analisi.

2. **Client.java**: essa costituisce un oggetto di tipo client. Quest'ultimo mantiene informazioni quali l'indirizzo IP del client (identificazione), la lista dei siti visitati, i byte di navigazione ed il tempo totale di navigazione.

3. **Server.java**: essa rappresenta un oggetto di tipo server il quale contiene il nome del server visitato dal client.

4. **Byte.java**: questa classe rappresenta un oggetto di tipo byte ed ha il compito di memorizzare (e restituire alla fine dell'analisi) il numero totale di byte associati alla navigazione di un client.

5. **Time.java**: questa classe rappresenta un oggetto di tipo time ed ha il compito di memorizzare (e restituire alla fine dell'analisi) il tempo totale di navigazione (in ms in quanto le operazioni vengono effettuate raccogliendo i dati dal file di log HTTP dove il tempo mantiene tale struttura). Questo oggetto ha il compito preciso di monitorare esattamente il tempo di navigazione.

6. **LineLOG.java**: essa rappresenta un oggetto particolare, ossia in fase di analisi del file di log, viene passata una riga al costruttore di tale classe, il quale ne verifica i contenuti andando a salvare i valori nelle variabili opportune. In questo modo riusciamo a memorizzare tutte le informazioni necessarie riguardo all'analisi da effettuare

2.3 Funzionamento

L'analisi genera un file di report contenente per ogni client siti visitati e durata totale navigazione in tempo e byte.
I campi byte, start e end del file di log vengono utilizzati per l'analisi del tempo in byte e in hh:mm:ss. Per ogni linea del file letta, al client associato incremento i byte di navigazione:
byteNavigation = byteNavigation + byte.
Per quanto riguarda il tempo, viene salvato lo start più piccolo e l'end più

grande al fine di ottenere, tramite una semplice sottrazione (end-start), il tempo totale di navigazione.

2.4 Grafica

La grafica dell'applicazione consiste in una finestra contente 3 bottoni e 1 area di testo.
I bottoni sono:

- openButton (funzione: far scegliere il file di log HTTP da monitorare);
- execButton (funzione: analizza il file di log HTTP passato salvando i dati in memoria);
- saveButton (funzione: salva i dati presenti in memoria in un file di report).

L'area di testo ha il compito di visualizzare in real time lo stato dell'applicazione a seconda delle funzione scelte dall'utente.

Capitolo 3

Manuale d'uso

Avviare con un doppio click l'applicazione ScanLog 1.1.

Premere il bottone "Seleziona File" per scegliere il file di log da monitorare.

A questo punto siamo pronti per effettuare il monitoraggio del file. Premere il bottone "Esegui". In questa fase, se si dovessero riscontrare errori nell'esecuzione dell'applicazione, questi verranno segnalati all'utente direttamente nell'apposita finestra di dialogo.

Successivamente salvare i risultati dell'analisi in un file di testo (txt) premendo il bottone "Salva Report".

Infine chiudere l'applicazione.

Ora possiamo controllare i risultati dell'analisi andando ad aprire il file di report creato.

Conclusioni

Attualmente l'applicazione utilizza solamente i campi opportuni al raggiungimento del suo scopo, ma qualora si volessero implementare funzioni aggiuntive di analisi, basterebbe prelevare i campi salvati nell'oggetto LineLog. L'implementazione di tale oggetto è stata pensata al fine di poter ramificare ed estendere con facilità il progetto.

Bibliografia

[1] Wikipedia: *http://www.wikipedia.org*