Simulazione di un servizio stile Web per la gestione di file testuali

Traccia per l'esame di Linguaggi Dinamici

Indicazioni generali

Il progetto deve essere sviluppato in linguaggio Perl o Python a scelta dello studente, e deve possedere i seguenti requisiti implementativi:

- Sfruttare le caratteristiche di **dinamicità** del linguaggio;
- Sfruttare le **librerie** disponibili;
- Applicare la programmazione ad **oggetti** per almeno un aspetto del progetto;
- Sfruttare l'**I/O**;
- Essere **documentato**;
- Accedere ad un **database**;
- Eseguire **elaborazioni sul testo**;
- Essere provvisto di strumenti di **testing**.

Il progetto può essere svolto singolarmente o in gruppo (fino a 2 studenti), seguendo l'argomento e le specifiche proposte. Tali specifiche dovranno/potranno essere estese in modo opportuno in base al numero di persone coinvolte e ai particolari interessi. In ogni caso si chiede di concordare con il docente le modalità di sviluppo del progetto.

Devono essere consegnati al docente:

- un **CD-ROM** (o altro supporto magnetico) contenente:
    - il **software prodotto**, comprese le librerie esterne eventualmente utilizzate;
    - un file **README.txt** che contenga le istruzioni per potere utilizzare il software;
- una **relazione cartacea** che descriva adeguatamente il software e le scelte progettuali effettuate.

Il CD-ROM e la relazione cartacea devono essere consegnati **7 giorni lavorativi** prima della data in cui si intende effettuare l'esame.

Il progetto sarà valutato sulla base della **qualità** della soluzione proposta in termini di progettazione e sviluppo software e della **chiarezza** della relazione e delle istruzioni fornite.

Nel seguito del testo, i paragrafi evidenziati in azzurro (esempio) sono facoltativi.

Argomento proposto

L'obiettivo del progetto è di sviluppare un programma che implementa un **server** che fornisce un **servizio in stile Web**, che riceve una **richiesta** formattata secondo un **protocollo** ben definito, esegue una **elaborazione** e restituisce una sorta di **pagina** con i risultati. Il server offre specifiche funzionalità per la **gestione di file testuali**.

Indicizzazione del contenuto del server

Il server deve essere inizializzato specificando il percorso della cartella contenente i file testuali da gestire. Per ognuno di questi file, deve essere memorizzato in apposite tabelle del database:

1. **nome** del documento;
2. **statistiche** sul documento, come numero di parole e di frasi;
3. altre possibili **informazioni** per la gestione delle richieste dinamiche.

Interazione con il server

Le modalità di **interazione** con il server sono tre:

1. Tramite standard input e output;
2. Tramite file;
3. Tramite socket.

Si implementi il programma in modo tale che l'aggiunta di nuove modalità in futuro **non** richieda la riscrittura dell'intero programma ma solo l'aggiunta di moduli.

Il programma deve dare la possibilità di scegliere quale modalità di interazione seguire, tramite **parametro sulla riga di comando** o **configurazione letta da file**. La modalità viene determinata all'avvio del programma e rimane la stessa fino al termine.

L'implementazione dell'interazione tramite socket è facoltativa; se non la si implementa, si predisponga comunque un modulo apposito (che non fa niente).

Interazione tramite standard input e output

In questa modalità il server riceve la richiesta tramite lo **standard input** e, dopo aver fatto l'elaborazione richiesta, stampa su **standard output** la pagina con i risultati. Il programma funziona quindi come un filtro.

Interazione tramite file

In questa modalità la richiesta viene letta **da un file** con un nome recuperato da un file di configurazione, e la pagina con i risultati viene scritta in **un altro file**, il cui nome viene sempre recuperato nel file di configurazione.

Interazione tramite socket

In questa modalità il server rimane in ascolto su una **socket** collegata alla **porta** specificata nel file di configurazione; quando riceve una richiesta, la elabora, produce la pagina e la invia tramite la **stessa socket**; poi torna in attesa di altre richieste.

Protocollo della richiesta

La richiesta viene inviata al server con una delle modalità descritte in precedenza.

Il **formato** della richiesta deve essere conforme al seguente protocollo:

`ld://risorsa/modalità/opzioni`

Dove:

- `ld://` rappresenta il protocollo ed è fisso
- `risorsa` è il nome della risorsa richiesta (file di testo)
- `modalità` può essere `s` (statica) o `d` (dinamica)
- `opzioni` rappresenta una lista di opzioni separate da ';' (usate in modalità dinamica)

Esempio di richiesta:

`ld://report.txt/d/computer;mouse;lang=en`

Ogni opzione può essere un singolo **termine** (ad es. `computer` e `mouse`) oppure delle **coppie chiave=valore** (ad es. `lang=en`).

Elaborazione e risposta

Nel caso in cui la modalità sia **statica**, l'elaborazione consiste nel semplice recupero del file specificato dal file system locale. La risposta sarà comprensiva di:

a. statistiche sul file estratte dal database;
b. il contenuto del file stesso.

Se invece la modalità è **dinamica**, per ciascuno dei **termini** specificati nella richiesta, la risposta dovrà includere:

a. una lista dei possibili significati del termine secondo il dizionario WordNet;
b. le frasi in cui il termine compare;
c. il numero di occorrenze totali nel documento.

Opzionalmente, è possibile prevedere **altre elaborazioni** sia sul testo della risorsa che sulle informazioni presenti nel database. Ad esempio, per ciascuna delle frasi in cui la parola compare, la risposta potrà includere anche una stima del possibile significato del termine (ad esempio effettuata sulla base delle similarità tra i termini della definizione e quelli della frase in cui comparre nel testo). È possibile sfruttare le opzioni nel formato chiave=valore per specificare delle elaborazioni aggiuntive.

Il formato della pagina di risultati è lasciato allo studente. Potrebbe ad esempio essere una pagina HTML.

File di log

Il server deve mantenere un **file di log**, in cui scrive il lavoro svolto; in particolare si richiede di tenere traccia di:

- Richieste arrivate
- Elaborazioni effettuate
- Risposte generate (solo l'indicazione che è stata inviata)
- Errori occorsi

Ogni informazione è corredata dall'orario.

Il **nome** del file di log è recuperato dal file di configurazione.

File di configurazione

Il file di configurazione è un file di **testo**, in cui i commenti iniziano con '#', e che contiene coppie chiave-valore separate dal carattere '='. Deve gestire **almeno** le seguenti chiavi:

- Modus: modalità di **interazione**;
- Infile: **file di input** nella modalità di interazione a file;
- Outfile: **file di output** nella modalità di interazione a file;
- Logfile: file di **log**;
- Port: **porta** sulla quale rimanere in ascolto
- Path: **percorso** della cartella da indicizzare.

Il server deve analizzare la configurazione all'avvio, scartando i commenti e le chiavi sconosciute.

Possibili moduli utilizzabili

Nel progetto è possibile appoggiarsi a moduli esterni.

In particolare, per l'interfacciamento al dizionario **WordNet** e per le relative operazioni sul testo si consigliano:

- Perl:
  - WordNet.pm: http://people.csail.mit.edu/jrennie/WordNet/
  - WordNet Similarity: http://www.d.umn.edu/~tpederse/similarity.html
- Python:
  - Natural Language Toolkit (NLTK): http://www.nltk.org/
    (vedere anche: http://nltk.googlecode.com/svn/trunk/doc/howto/wordnet.html)

Per l'utilizzo di **socket** si può far riferimento alle seguenti pagine:

- Perl:
  - http://perldoc.perl.org/perlipc.html#Sockets%3a-Client%2fServer-Communication
- Python:
  - http://docs.python.org/library/socket.html