Immagine che contiene esterni, nero, segnale, fotografia

Descrizione generata automaticamente

Relazione Progetto Programmazione di Reti

a cura di Parrinello angelo

Parrinello Angelo | Programmazione di Reti | A.A. 19/20

# INDICE

1. INTRODUZIONE
2. SVILUPPO
3. COMMENTI FINALI

# INTRODUZIONE

L’obbiettivo di questa relazione è spiegare in maniera chiara la realizzazione di un Web Server in Python per una testata giornalistica di sport, con relative directory e file accessori, che sia anche in grado di supportare più connessioni in contemporanea.

Il Web Server, che sarà realizzato su local host, ovvero il nome associato all’interfaccia di loopback con indirizzo IP 127.0.0.1 in IPv4, dovrà consentire la visualizzazione della pagina inziale (definita dal file index.html) nella quale saranno presenti immagini e titoli degli articoli.

I titoli degli articoli principali dovranno contenere dei link ad altre pagine web (interne al web server) nelle quali il testo dell’articolo viene mostrato per esteso.

Nella pagina principale saranno presenti un link per il download di un file pdf da parte del client e altre piccole feature.

# SVILUPPO

La prima cosa fatta è stata capire bene, nel dettaglio del suo codice, il funzionamento del web server multithread di base. Il funzionamento di esso è spiegato nelle slide della quinta lezione di laboratorio del prof. Piroddi e nel file http\_multithread.py di base, che si trova su IOL: il codice è man mano commentato, quindi non mi dilungherò sul suo funzionamento di base.

Immagine che contiene fotografia, monitor, televisione, schermo

Descrizione generata automaticamenteDopo la prima fase, in cui ho capito per bene cosa fare, mi son chiesto come dovesse apparire questo web server. Così mi son messo a scrivere la pagina web in HTML, index.html.

Una prima versione della mia pagina web appariva così molto minimale, ma rispettava tutte le regole della realizzazione della traccia 2: c’era la pagina iniziale con titoli e immagini; i titoli degli articoli avevano in sé altri link ai vari articoli con il testo integrale. Inoltre era presente un link per il download.

Il web server (sempre di base) non aveva problemi a gestire una paginina del genere con così pochi dati.

Immagine che contiene screenshot, sedendo, nero, tenendo

Descrizione generata automaticamenteUna volta creata una base iniziale dalla quale partire mi sono messo a raffinarla. Per prima cosa volevo rendere la pagina più carina esteticamente. Il modo più carino e veloce era tramite CSS. Così, prendendo pezzi di codice CSS base in giro per il web e modificandoli e unendoli assieme, sono riuscito ad ottenere un aspetto un po’ più carino per la mia pagina web.

Ora non solo la mia pagina rispettava tutti i canoni di base, ma aveva anche una galleria immagini (cliccabili che portano ai vari articoli), aveva una sezione iniziale dove erano elencati i vari articoli e sotto di sé, gli intro degli stessi. Inoltre in fondo c’è la sezione “About me” che racconta in breve chi sono; in basso c’è un link cliccabile che porta alla mia pagina Facebook proprio per scoprire di più su di me. Sotto quest’ultima sezione ho inserito la possibilità di tornare in cima alla pagine in cui si sta navigando (singoli articoli o home), oppure tornare alla home.

L’ultima cosa che ci tenevo a inserire era la barra di ricerca: l’ho creata usando anche un piccolo script di Javascript. Questo permette di mostrare a video i vari articoli presenti (in questo caso 3) : man mano che si cerca l’articolo desiderato gli altri non cercati “scompaiono”.

Immagine che contiene screenshot, uomo, monitor, schermo

Descrizione generata automaticamente Qui accanto invece una pagina web esempio di uno degli articoli presenti all’interno del web server. Come si vede si può leggere l’intero articolo e si possono fare gran parte delle cose presenti anche nella home.

Una volta sistemato a livello web le pagine le ho testate con il web server. Il tutto funzionava perfettamente. Arrivato a questo punto volevo migliorare il lato web server con Python. Non avendo mai programmato in Python questa è stata la parte più dura ma anche più entusiasmante perché ne ho scoperto la bellezza.

Quando si accede al web server, all’utente viene chiesto di registrarsi o fare il login. Se si scrive “access” o “sign in”( o più semplicemente “a” o “s”), viene chiesto di inserire in entrambi i casi id e password. Nel caso d’ “access”, il web server controlla che id e password siano presenti su file. Infatti i vari dati verranno salvati molto banalmente su un file .txt. Se son presenti il web server parte, altrimenti viene richiesta l’identità per un massimo di 3 volte. Cosa analoga per il ”sign in”, solo che in questo caso si andrà a salvare su file l’id e password inseriti. Il programma va a controllare che l’id non sia già usato al momento dell’inserimento dei dati. Per far sì che i file vengano salvati sicuramente riga per riga all’input da tastiera è sempre aggiunto un “\n” in modo tale da andare a capo. Il controllo su file, appunto, prende il testo riga per riga e un”\n”.

Per usi futuri ho creato la classe, ServerHandler che mantiene le funzioni di SimpleHTTPRequestHandler e implementa il metodo get nel caso in cui si vogliano aggiungere funzioni. Infatti quando si creerà il web server gli viene passato il ServerHandler.

In fase di chiusura, ovvero quando viene premuto CTRL+C, il sistema chiederà all’utente se si è sicuri o meno di chiudere il web server.

# COMMENTI FINALI

Il seguente progetto non è stato particolarmente facile per me, in quanto non ho mai lavorato con Python e ho lavorato molto poco con HTML, CSS e Javascript. Infatti ho impiegato parecchie ore di lavoro, non tanto per la realizzazione vera e propria di esso, ma per capire e studiare come fare le cose. Ma sono comunque soddisfatto in quanto ritengo di aver capito a fondo il funzionamento di un web server , di aver migliorato la mia abilità con dei linguaggi come HTML, CSS e Javascript e di aver iniziato a scrivere codice con Python, linguaggio davvero importante e utilizzabile per mille scopi. In definitiva il web server funziona a dovere e fa tutto ciò che di base si può chiedere a un web server. L’id e la password di default sono student e reti2, come si può vedere dai rispettivi file.