HOTELIER

Indice

---

1. Guida utente e descrizione sintetica

1.1 Compilazione ed esecuzione:

compilazione:

posizionarsi nella directory "HOTELIER" ed eseguire:
[...]/HOTELIER$ **make classes**

esecuzione:

dopo aver compilato, eseguire:
$ **java ServerMain**
e
$ **java ClientMain**

in alternativa eseguire:
[...]/HOTELIER$ **java -jar HOTELIER_server.jar**
e
[...]/HOTELIER$ **java -jar HOTELIER_client.jar**

1.2 Librerie esterne:

gson-2.10.1.jar - usata per l'importazione ed esportazione dati da/verso file json

1.3 Argomenti file di configurazione, fra parentesi i valori di default:

server (in server.config):
port (10000) - porta usata per la connessione principale
threadpool_size (10) - dimensione threadpool
timeout (10000) - timeout clientsocket, timeout terminazione threads, tiomeout passato a serverexit
mport (1234) - porta usata per connettersi al gruppo multicast
hotelsFile (./Hotels.json) - json file contenente gli hotel
usersFile (./Users.json) - json file contenente gli utenti
reviewsFile (./Reviews.json) - json file contenete le review
RMIname (RMIRegister) - nome del servizio di registrazione
RMIport (2345) - porta usata dal servizio di registrazione (tramite RMI)
callbackName (RMIcallback) - nome del servizio di notifica
callbackPort (3456) - porta usata dal servizio di notifica (tramite RMI callback)
update_delay (10000) - intervallo di tempo fra un aggiornamento del ranking e l'altro

client (in client.config):
hostname (localhost) - hostname connessione principale
port (10000) - porta usata per la connessione principale
mport (1234) - porta usata per connettersi al gruppo multicast
mhostname (230.0.0.0) - hostname del gruppo multicast
RMIname (RMIRegister) - nome del servizio di registrazione
RMIport (2345) - porta del servizio di registrazione
callbackName (RMIcallback) - nome del servizio di notifica
callbackPort (3456) - porta usata dal servizio di notifica (tramite RMI callback)

1.4 Sintassi comandi:

server:
exit - termina il processo

client:
register username password - chiede la registrazione di un nuovo utente
login username password - chiede di effettuare il login
logout - chiede il logout dell'utente
searchHotel hotelname city - stampa informazioni su un particolare hotel
searchAllHotel city - stampa informazioni sugli hotel di una particolare citta'
insertReview hotelname city globalRate cleaning position services quality - chiede l'inserimento di una nuova review
showMyBagdes - stampa l'ultimo badge assegnato all'utente
quit - termina la connessione col server

1.5 Scelte effettuate:

Algoritmo ranking:
rate = quality*(3/5) + quantity*(1/5) + actuality*(1/5)

qualita' (quality): media aritmetica dei punteggi sintetici assegnati ad un hotel
quantita' (quantity): numero di recensioni di un hotel diviso per il totale di tutte le recensioni
attualita' (actuality): tempo dell'ultima recensione di un hotel diviso per il tempo attuale

Implementazione server: multithreaded server, gestione dei thread tramite threadpool.

1.6 Strutture dati:

server:
tre ConcurrentHashMap per gestire I dati relativi ad hotel (**hotels**), utenti (**users**) e recensioni (**reviews**).

UpdateRanking:
hotelsByCity (HashMap) - struttura dati ausiliaria usata per calcolare i ranking.
reviewsByHotel (HashMap) - struttura dati ausiliaria usata per calcolare i ranking.
lsiteners (ConcurrentHashMap) - contiene la lista di client da notificare tramite RMI callback.

Worker:
reviewsLocal - usato per salvare localmente le nuove review
Aux - usato per semplificare il parsing dei comandi inviati dal client.

client:
hotelsRanking (HashMap) - usata per salvare in locale i ranking di alcune citta'

1.7 Thread attivati:

server:
UpdateRanking
MulticastService (gestito da UpdateRanking)
ServerExit
Worker

client:
MulticastClient

2. Descrizione server

ServerMain legge i parametri di configurazione (v. sopra) dal file specificato nella variabile "configPath", questa operazione viene effettuatua tramite il metodo "importConfig".
Importa i file json relativi agli hotel, utenti e recensioni.
Attiva il servizio di registrazione tramite RMI.
Esegue il thread "updateRanking", responsabile dell'aggiornamento del ranking degli hotel, e della relativa notifica agli utenti.

Apre una socket (serverSocket), attiva un thread "ServerExit", e attiva un threadpool di processi "Worker". Quando arriva una richiesta di connessione, passa la relativa socket (clientSocket) ad uno dei Worker.

Quando termina, chiude le risorse attivate in precedenza, ed esporta le strutture dati relative agli hotel, utenti e recenzioni in json.

ServerExit:
aspetta che venga inserito in input "exit" e chiude la serverSocket.

Worker:
gestisce la connessione con un client.
Mantiene una struttura dati locale per memorizzare le recensioni inserite dal client, quando termina le salva nella ConcurrentHashMap relativa alle recensioni.
Effettua invece direttamente le modifiche alla ConcurrentHashMap relativa agli utenti, per evitare piu' sessioni contemporanee per lo stesso utente.

UpdateRanking
aggiorna periodicamente il ranking di ogni hotel.
A tal fine usa alcune strutture dati ausiliarie:
hotelsByCity: mantiene una corrispondenza fra ogni citta' e i relativi hotel. Le liste di hotel in hotelsByCity vengono ordinate in base al "rate".

reviewsByHotel: mantiene una corrispondenza fra ogni hoitel e le relative recensioni.

In caso di variazioni nella prima posiione di un ranking locale, attiva un thread "MulticastService", per inviare le notifiche al gruppo multicast.
In caso di variazione in un qualche ranking locale, vengono notificati i client in ascolto tramite RMI callback.

Al momento della terminazione, esce dal loop di aggiornamento e chiude le risorse attivate in precedenza.

MulticastService
invia al gruppo multicast le notifiche relative alla variazione della prima posizione in un qualche ranking locale.

RMIRegisterInterface
interfaccia usata per implementare il servizio di registrazione tramite RMI.

RMIRegister
implementa l'interfaccia RMIRegisterInterface.

RMIevent_interface
interfaccia usata per implementare il servizio di notifica tramite RMI callback.

CallbackServer
implementa l'interfaccia RMIevent_interface.

3. Descrizione client

ClientMain legge i parametri di configurazione (v. sopra) dal file specificato nella variabile "configPath", questa operazione viene effettuata tramite il metodo "importConfig".
Se l'importazione va a buon fine, il client crea una socket e prova a connettersi al server.
Una volta effettuata la connessione, entra in un while loop (controllato dalla variabile "done") e, tramite stdin, chiede all'utente il comando da inviare al server.
Vengono inviati solo i comandi specificati di seguito, ad ogni altro input il client riscponde stampando "invalid command".

Input validi:
quit
invia il relativo comando al server.

Chiama il metodo locale "ClientExit": in caso sia stato effettuato il login in precedenza, termina il thread "multicastClient" (v. seguito).
Se e' attivo il servizio di notifica, rimuove l'utente dalla lista di utenti in ascolto.

setta la variabile "done" a "true": il programma esce dal while loop e chiude la socket.

register
chiede all'utente il nome utente e la password.
Tenta di connettersi al servizio di registrazione tramite RMI, invia i dati, e restituisce il risultato all'utente.

login
chiede all'utente il nome utente, la password, se vuole ricevere notifiche sul ranking di alcune citta' e, in caso affermativo, le citta' di interesse, che vengono salvate nella variabile locale "cities".

Invia al server il nome utente, la password, e il comando stesso. Attende poi risposta del server.
In caso di risposta positiva da parte del server, vengono settate le variabili locali "_username" e "loggedin", e viene creato il thread "multicastClient", istanza della classe "MulticastClient" (v. seguito).

In caso di richiesta notifiche, tramite la funzione locale "callback", viene aggiunto l'utente nella lista di utenti interessati alle notifiche.

logout
se non e' stato effettuato il login in precedenza, resetta le variabili locali "loggedin" e "_username", invia il comando al server, e attende risposta.

searchHotel
chiede all'utente il nome dell'hotel e la citta' di interesse, li invia al server insieme al comando, e attende risposta.

searchAllHotel
chiede all'utente la citta' di interesse, la invia al server insieme al comando, e attende risposta.

insertReview
chiede all'utente il nome dell'hotel e la citta' di interesse, oltre ai valori da assegnare alla recensione (punteggio globale e 4 punteggi specifici), li invia al server insieme al comando, e attende risposta.

showMyBadges
invia il relativo comando al server e attende risposta.

Attesa risposta: in un while loop, il client legge e stampa sullo stdout ogni messaggio ricevuto dalla socket. Termina alla ricezione della stringa "END".

MulticastClient
si connette ad un gruppo multicast, in un while loop (controllato dalla variabile locale "done") attende e stampa i messaggi ricevuti.
Se viene invocato il metodo "finish", esce la while loop, esce dal gruppo multicast, chiude la socket e interrompe il thread.

RMIlistener_interface
interfaccia usata per implementare il servizio di notifica tramite RMI callback.

CallbackClient
implementa RMIlistener_interface.