Progetto Programmazione Java

Servizio di messaggistica istantanea

A.A. 2015-2016

Progetto

Rambod Rahmani

Server di Log

Le classi del Server di Log, come mostrate graficamente nel diagramma UML, sono le seguenti:

Classe Server
Classe contenente la funzione main() del progetto.
Gestisce l'Input/Output da terminale per la gestione (avvio, arresto, stampa logs) del Server di Log.

Classe GestoreParametriConfigurazioneXML
Legge i parametri di configurazione dal file di configurazione .xml locale.
Valida i parametri tramite XML Schema con file .xsd locale.

Classe ParametriConfigurazione
Contiene i parametri di configurazione letti dal file di configurazione .xml locale dalla classe **GestoreParametriConfigurazioneXML**.
Serializzata/Deserializzata tramite XMLStream.

Classe GestoreLogsXML
Gestisce l'accesso sincronizzato di lettura e scrittura sul file di log .txt locale.
Valida i log ricevuti tramite XML Schema con file .xsd locale.

Classe ThreadServerSocket
Gestisce il ServerSocket in attesa delle connessioni da parte dei Clients.
Estende Thread.
Inviato alla classe **SocketClient**.

Classe SocketClient
Gestisce il Socket di comunicazione tra Server di Log e ciascun client Connesso.
Estende Thread.

Diagramma UML
Diagramma UML presente in allegato PDF [Diagramma UML Server.pdf**].**

Client

Le classi del Client, come mostrate graficamente nel diagramma UML, sono le seguenti:

Classe Client
Classe contenente la funzione main() del progetto.
Istanzia una variabile di tipo **FrameContatti** contenente la GUI principale del progetto.

Classe Contatto
Contiene le informazioni di ciascun contatto connesso.
Utilizzata dalla classe **GestoreBaseDiDati** per ottenere la lista dei contatti connessi.

Classe FrameContatti
Contiene il JFrame con la GUI principale (area connessione, lista contatti connessi, menu strumenti).
Istanzia una variabile di tipo **ServerSocketConversazioni** che avvia un ServerSocket che rimane in ascolto delle connessione da parte degli altri Contatti.

Classe FrameConversazione
Contiene il JFrame con la GUI della finestra di conversazione.
Istanzia una variabile di tipo **SocketConversazione** che si connette al **ServerSocketConversazioni** del contatto con cui si desidera conversare.

Classe FrameStatistiche
Contiene il JFrame con la GUI della finestra delle statistiche.

Classe GestoreBaseDiDati
Esegue tutte le query sulla Base di Dati.

Classe GestoreCacheConversazioni
Si occupa di salvare e prelevare da file binario locale gli ultimi messaggi scambiati con ciascun Contatto.

Classe GestoreParametriConfigurazioneXML
Legge i parametri di configurazione dal file di configurazione .xml locale.
Valida i parametri tramite XML Schema con file .xsd locale.

Classe ParametriConfigurazione
Contiene i parametri di configurazione letti dal file di configurazione .xml locale dalla classe **GestoreParametriConfigurazioneXML**.
Serializzata/Deserializzata tramite XMLStream.

Classe MessaggioDiLog
Implementa Serializable
Contiene i campi di ciascun messaggio di log inviato da Client a Server.

Classe ModelloTabellaContatti
Estende AbstractTableModel.
Gestisce i dati della tabella contenente la lista dei Contatti connessi.

Classe ServerSocketConversazioni
Gestisce il ServerSocket in attesa delle connessioni da parte degli altri Contatti.
Estende Thread.
Inviato alla classe **SocketConversazione**,

Classe SocketConversazione
Gestisce il Socket di comunicazione tra due Clients connessi che scambiano messaggi.
Estende Thread.

Classe SocketServerDiLog
Gestisce il Socket di comunicazione con il Server di Log.
Estende Thread.

Classe Storico
Contiene le informazioni riguardanti il numero di ore di connessione per ciascun Contatto.
Utilizzata dalla classe **GestoreBaseDiDati** per ottenere la lista delle ore spese connesse per ciascun contatti.

Diagramma UML
Diagramma UML presente in allegato PDF [Diagramma UML Client.pdf**].**

Classi Comuni

Client e Server di Log hanno in comune il package messaggio, che contiene le classi Messaggio e TipoMessaggio, e che verrà sviluppato sotto forma di una Java Class Library, in modo da ottenere un .jar da includere nei due progetti che ne necessitano. Essendo messaggio l'oggetto serializzato che viene scritto e letto da ObjectOutputStream e ObjectInputStream, conviene avere un unico file .jar da includere nel progetto Server e Client in modo da non incorrere in errori di incongruenza.

Classe Messaggio
Implementa Serializable
Contiene i campi di ciascun messaggio scambiato tra Server e Client o tra due Clients.

Classe TipoMessaggio
Implementa Serializable
Enumerazione contenente i tipi di messaggio che possono essere creati.

Diagramma UML
Diagramma UML presente in allegato PDF [Diagramma UML ServerClient.pdf**].**