Projekt

EDA095

chattApp

Hanna Bayerlein, dic14hba
Felicia Carlsson, ctr14fca
Elin Hanstorp, dic14eha
Elsa Schröder, dat14esc
@student.lu.se

Handledare Filip Ström

20 maj 2016

1 Bakgrund

Detta är en rapport som behandlar resultatet av projektmomentet i kursen EDA095 Nätverksprogrammering. Vi valde att utveckla en chattapplikation som består av en serversida och en klientsida med ett grafiskt användargränssnitt som används för att koppla upp sig mot servern och skicka meddelanden till övriga deltagare i chatten.

2 Kravspecifikation

Vi utformade en kravspecifikation för applikationen, som kan delas in i två kategorier.

Användare

- Alla användare som är online syns i en lista för alla som är inloggade.
- Användare som loggar in på chatten får välja ett användarnamn.
- Användarnamnet som en användare valt är det som syns för övriga deltagare.
- Användarnamnet måste vara unikt under varje session.

Chattrummet

- Alla inloggade användare kan se konversationen.
- När en ny användare loggar in kan den endast se meddelanden som skickats från och med att hen loggade in.
- Konversationer sparas inte när en användare stänger ner chattapplikationen eller loggar ut.
- En användare kan välja att skapa en enskild konversation med en annan användare.
- Användare ska kunna skicka filer över chattrummet. Detta ska ske via trådar.

3 Modell

Vi använder oss av tre paket i vår modell; ett Serverpaket, ett Clientpaket och ett GUIpaket som använder sig av ActionListener. Då funktionaliteten ligget i Server- och Clientpaketet är det dessa två som kommer beskrivas. En kortfattad beskrivning av klasserna i alla tre paketen finns att läsa i tabellerna på sida 4 och 5.

När en användare väljer att logga in på chattrummet startas klassen Client i Messagepaketet. Clientklassen består av en tråd, en ReadMessagetråd, en

ActionListenermetod som lyssnar efter meddelanden från användaren och en socket som kopplar upp sig mot en port och en localhost.

ActionListenermetoden använder OutputStream- och PrintStreamklasserna för att läsa av meddelanden från användarens tangentbord och sedan skicka vidare detta meddelande via socketuppkopplingen till Sessionklassen. ReadMessagetråden använder InputStream- och BufferReaderklasserna för att läsa meddelanden via socketuppkopplingen från trådklassen MessageGetter och skriver ut meddelandet i alla uppkopplade användares chattruta.

Klassen Server är redo att ta emot nya uppkopplingar och meddelanden från de olika Clientobjekten genom en ServerSocket som hela tiden hålls uppkopplad mot port 30000. En Sessiontråd skapas för varje uppkoppling mot servern och ansvarar för att läsa av meddelanden med hjälp av InputStreamklassen. Meddelandet placeras sedan i MailBoxobjektet.

Det enda mailboxobjektet skapas i Serverklassen när denna exekveras, detta görs för att alla användare ska lagra sina meddelanden i samma objekt. MailBoxklassen har två stycken synkroniserade metoder då Sessiontrådarna från de olika klienterna och MessageGettertråden inte ska skriva över varandra och missa något meddelande. Mailboxobjektet innehåller maximalt ett meddelande. Om Mailboxen är tom underrättas Sessiontrådarna att det återigen går att lagra meddelanden och den tråd som är på tur lägga in sitt meddelande från sin användare. När ett meddelande har lagrats underrättas MessageGettertråden som hämtar meddelandet från Mailboxen och återigen underrättas Sessiontrådarna att Mailboxen är tom.

Det enda MessageGetterobjektet skapas i Serverklassen när denna exekveras. MessageGetter är en trådklass som hämtar lagrade meddelanden i Mailboxobjektet. Genom OutputStream- och PrintWriterklasserna skickar MessageGettertråden ut meddelandet antingen till alla inloggade användare eller ett privatmeddelande till en utvald användare.

Samma metod används för de olika Fileklasserna. Clientklassen innehåller också två en Filetråd, ReadFile som fungerar på samma sätt som ReadMessage, en metod för skickandet av filer som är kopplat till ActionListener med hjälp av FileInputStream- och FileOutputStreamklasserna. FileSession- och FileGettertrådarna använder sig av DataInputStream- och DataOutputStreamklasserna för att lagra och hämta filer i FileBoxen.

Tabell 1: Paket Server

Klass	*Beskrivning*
Server	Denna klass accepterar uppkopplingar från de olika användarna
User	Innehåller ett unikt användarnamn och en personlig socket
Message	Innehåller ett meddelande och dess avsändare
Mailbox	I denna klass lagras det meddelande som ska skickas vidare till de inloggade användarna
MessageGetter	Denna tråd hämtar ut det meddelande som lagras i MailBox och skickar vidare det till de inloggade användarna
Session	Dessa trådar lagrar meddelanden deras användare skickat i Mailboxen
FileBox	I denna klass lagras den fil som ska skickas vidare till de inloggade användarna
FileGetter	Denna tråd hämtar ut den fil som lagras i FileBox och skickar vidare den till de inloggade användarna
FileSession	Dessa trådar lagrar filer deras användare skickat i FileBoxen

Tabell 2: Paket Client

Klass	*Beskrivning*
ClientMain	Mainmetoden i detta paket. Här skapas en Client
Client	Denna klass kopplar upp användaren mot servern och skapar trådarna ReadMessage, ReadFile och SendFile
ReadMessage	Denna trådade klass tar emot meddelanden från MessageGettertråden
ReadFile	Denna trådade klass tar emot filer från FileGettertråden

4 Användarhandledning

När användaren startat programmet kommer det först upp en dialogruta "Login", där hen uppmanas att skriva in ett användarnamn och därfefter trycka på "OK". Om användarnamnet är upptaget, kommer hen in i chattrummet där ett nytt namn kan anges i textfältet. Är användarnamnet ledigt kommer hen vidare till chattrummet och kan börja skriva till andra deltagare. Varje gång en ny deltagare loggar in eller ut meddelas det i ChattroomPane.

Om användaren vill byta namn till t ex NewName, skriver den in: "N: NewName" i textfältet. Namnbytet meddelas därefter till samtliga chattdeltagare.

För att föra över en fil trycker användaren på knappen "Send File", varpå en

Tabell 3: Paket GUI

Klass	*Beskrivning*
Main	Mainmetoden som skapar gränssnittet och en Client
Controller	Controllern kommunicerar händelser från GUI:t till servern som uppdaterar modeln
Model	Uppdateras från Controllern och underrättar de lyssnande obsservers på ändringarna som skett
GUI	Gränsnitt som innehåller utseendet och de observers som lyssnar på Modelklassen
LoginButtonListener	Lyssnar efter händelser för knappen OK och lägger till det valda användarnamnet i UserPane
SendButtonListener	Lyssnar efter händelser för knappen Send, skickar då vidare användarens meddelande via OutputStream
SendFileButtonListener	Lyssnar efter händelser för knappen Send File och låter då användaren välja fil från sin dator
FileChooser	Användaren väljer en fil och FileChooser skickar då vidare användarens fil via FileOutputStream
ChattroomPane	Visar alla meddelanden i chattrummet
UserPane	Visar de inloggade användarna

filhanterare öppnas. När användaren navigerat till filen som ska skickas, markerat den och tryckt på knappen "Open" skickas den till alla chattdeltagare, inklusive användaren själv. Den förvalda platsen för nedladdade filer är varje enskild användares skrivbord (desktop).

För att skicka ett privat meddelande till en utvald chattdeltagare, t ex texten "HemligtMeddelande" till användaren "AnnanAnvändare", skrivs kommandot: "P: AnnanAnvändare: HemligtMeddelande". Endast den valda mottagaren kan då se meddelandet.

För att logga ut anger användaren kommandot: "Q".

5 Utvärdering

Vi har uppfyllt alla krav fullständigt förutom en del i sista punkten som rör filöverföringen. Det är möjligt att föra över filer via konsolen i Eclipse, men det har visat sig vara svårare än vi trodde att implementera den lösningen för GUI:t. Vi ville att chattdeltagarna skulle fortsätta ha möjligheten att skriva meddelanden under filöverföring, varför vi valde att ha överföringen för meddelanden och filer med två olika anslutningar. Detta för att förhindra att hela chatten fryses tills filen är överförd. Vi tror att det är möjligt att väva ihop de två typerna av överföring, men att det skulle kräva mycket mer arbete än vad vi har tid med innan projektinlämningen.

Angående projektplanering lade vi upp en plan när projektmomentet introducerades och har följt den under kursens gång. Vi har inte haft så många

möten med handledaren för att vi känt att vi haft det mesta under kontroll. I efterhand inser vi att det kanske hade varit bra ändå, och att problem kunnat lösas snabbare med en hjälpande hand. Vi delade upp arbetet mellan oss i projektgruppen och tror att vi med handledarens närvaro kanske hade formulerat våra arbetsuppgifter tydligare varje vecka och därmed kunna ge varandra bättre återkoppling.

Projektet som helhet tycker vi var ett roligt inslag i kursen. Dock var laborationshandledningarna extremt svåra att förstå, och i och med att vårt projekt byggde på labb 3 försenade det vårt arbete markant. Handledningarna behöver vara mycket mer konkreta och inte i så lösa ordalag som de varit under den här terminen. Det är möjligt att de är utformade på det här sättet för att D4:orna som har möjlighet att välja den här kursen redan har goda kunskaper inom det här ämnet, men för C2:orna som inte läst något sådant här tidigare blir det ett ganska stort kunskapshopp.

6 Programlistor

Källkoden går att hitta på länken `http://users.student.lth.se/dat14esc/chattProjekt/ChattApp/src/`.