Articles » Languages » Java » General

Java Chat With Customizable GUI

Jeeva S, 7 May 2007 CPOL

★★★★★ 4.76 (110 votes)

A complete Java (AWT) Chat Application with great customizable GUI Interface. It has features such as general chat and private chat, music when message arrives, sending images and more

Download final source code - 276.01

Download class files - 107.71 KB

Note: The latest source code has been uploaded along with this article now.

Introduction

This is my second article on The Code Project. I have already posted Tap Control in Java here.

This Java Chat is purely AWT based, no Swing Components used and still it has a great look and feel. For this application, I have developed my own Tab Control and Image Canvas. Also, I have uploaded the complete source code here. You can download it from there.

This is the updated article . Now I have posted all the concepts behind the chat instead of putting the source code as per the request of The Code Project members.

Features

1. Transfer Smilies with Text
2. Private Chat
3. Great Look and Feel with Customized Color
4. Audio Enabled

Description

In this Chat application, we have both server side and client side modules.. In server side, I have defined our own RFC Commands. Some of the commands which I have used in this application are listed below:

- `HELO` - Initialize connection to server
- `QUIT` - Remove users from chat
- `KICK` - Kickoff from chat
- `CHRO` - Change room
- `MESS` - Send general message
- `PRIV` - Send private message
- `ROCO` - Get users count in specified room
- `CALL` - Request for voice chat (not included with this one)

Server Side Module

I will briefly explain the concepts behind the server side.

- Created custom `UserObject` class which will have the client details like username, the socket of user, and the room name, etc.
- When the Chat Server runs, it opens the Server Socket at port 1436 (we can modify too) and listen for the client to connect. If the client connects to the server, it will open a separate thread to service. So, when the client sends `QUIT` command, it will close the thread too. If you take a look *ChatCommunication.java*, you will get all the details.

This is a sample code of getting connection from the Chat Client and creating a new object of `ChatCommunication`. In `ChatCommunication` class, we will create a thread to watch all the commands from the client and responds to the client too.

```
ChatServer.java
while(true
```

```
{
    Socket socket = serversocket.accept();
    ChatCommunication chat = new ChatCommunication(socket);
}
.........

ChatCommunication.java
..........
ChatCommuncation(Socket socket)
{
    personalsocket = socket;
    dout = new Dataoutputstream(personalsocket.getoutoutstream());
.....
}
```

Client Side Module

I will also briefly explains the concepts behind the Client side Chat.

- When the Chat Client runs, it will open a socket and connect to `ChatServer` by sending `HELO RFC` to Server.. Once it gets connected, the chat client will keep the socket connection and communicate with the server whenever the user commands it.
- Another important thing in the client module is the USER INTERFACE. I have created my OWN Custom Components like Tab Control and Image Supported Message Canvas.
- The basic idea of creating a message canvas is based on simple logic. Whenever users enter the message, I will store it in arraylist. Also, in the Arraylist, I keep the `XOffset` and `YOffSet` position of each message. If you have a look of this sample code, you might get an idea of what I mean.

```
Ex:
...........
for(int i =0; i < messagearraylist.size();i++)
{
PaintMessageToMessageCanvas((MessageObject)messagearraylist.get(i);
}.........

************************************************************
This is the Function To Paint Images and Text Messages
************************************************************
private void PaintMessageIntoCanvas(MessageObject messageObject)
{
int m_YPos = messageobject.StartY - YOffset;
int m_XPos = 5 - XOffset;
int CustomWidth = 0;
String Message = messageobject.Message;

/*************Print The User Name in UserName Font *************/
if(Message.indexOf(":") >= 0)
{
graphics.setFont(UserNameFont);
chatclient.getGraphics().setFont(UserNameFont);
fontmetrics = chatclient.getGraphics().getFontMetrics();
String m_UserName = Message.substring(0,Message.indexOf(":")+1);
graphics.drawString(m_UserName,m_XPos+CustomWidth,m_YPos);
CustomWidth+=fontmetrics.stringWidth(m_UserName)+HorizantalSpace;
Message = Message.substring(Message.indexOf(":")+1);
}

/*********Set the Text Font **********/
chatclient.getGraphics().setFont(TextFont);
graphics.setFont(TextFont);
```

```
fontmetrics =  chatclient.getGraphics().getFontMetrics();

/**********Print Image Area*******/
if(messageobject.IsImage == true)
{
tokenizer = new StringTokenizer(Message," ");
while(tokenizer.hasMoreTokens())
{
TokenString = tokenizer.nextToken();
if(TokenString.indexOf("~~") >= 0)
{
/********If its a Proper Image************/
try {
int m_ImageIndex = Integer.parseInt(TokenString.substring(2));
if((m_ImageIndex >= 0) && (m_ImageIndex < chatclient.IconCount))
{
graphics.drawImage(chatclient.IconArray[m_ImageIndex]
,m_XPos+CustomWidth,m_YPos - 15,messageobject.Width,messageobject.Height,this);
CustomWidth+=messageobject.Width+HorizantalSpace;
}
}catch(Exception _Exc) { }
}
else
{
graphics.drawString(TokenString,m_XPos+CustomWidth,m_YPos);
CustomWidth+=fontmetrics.stringWidth(TokenString)+HorizantalSpace;
}
if(TotalWidth < m_XPos+CustomWidth)
{
TotalWidth = m_XPos+CustomWidth;
scrollview.setValues(TotalWidth,TotalHeight);
}
.................
}
```

Conclusion

In this updated article, I have uploaded my complete source code. To download the full source code, click here.

If you still have any doubts, feel free to contact me at vavjeeva@gmail.com.

License

Share

About the Author

Jeeva S

Architect
United States

No Biography provided

You may also be interested in...

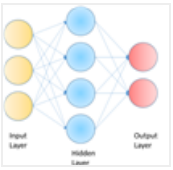
A Customizable Architecture for 3D Graphics Applications

An Introduction to Application Performance Management (APM)

A Java Chat Application

Learnings from a DevOps Hackfest with Orckestra

The Ultimate DevOps Toolkit



Deep Learning on Windows: A Getting Started Guide

Comments and Discussions

256 messages have been posted for this article Visit **http://www.codeproject.com/Articles/2440/Java-Chat-With-Customizable-GUI** to post and view comments on this article, or click **here** to get a print view with messages.