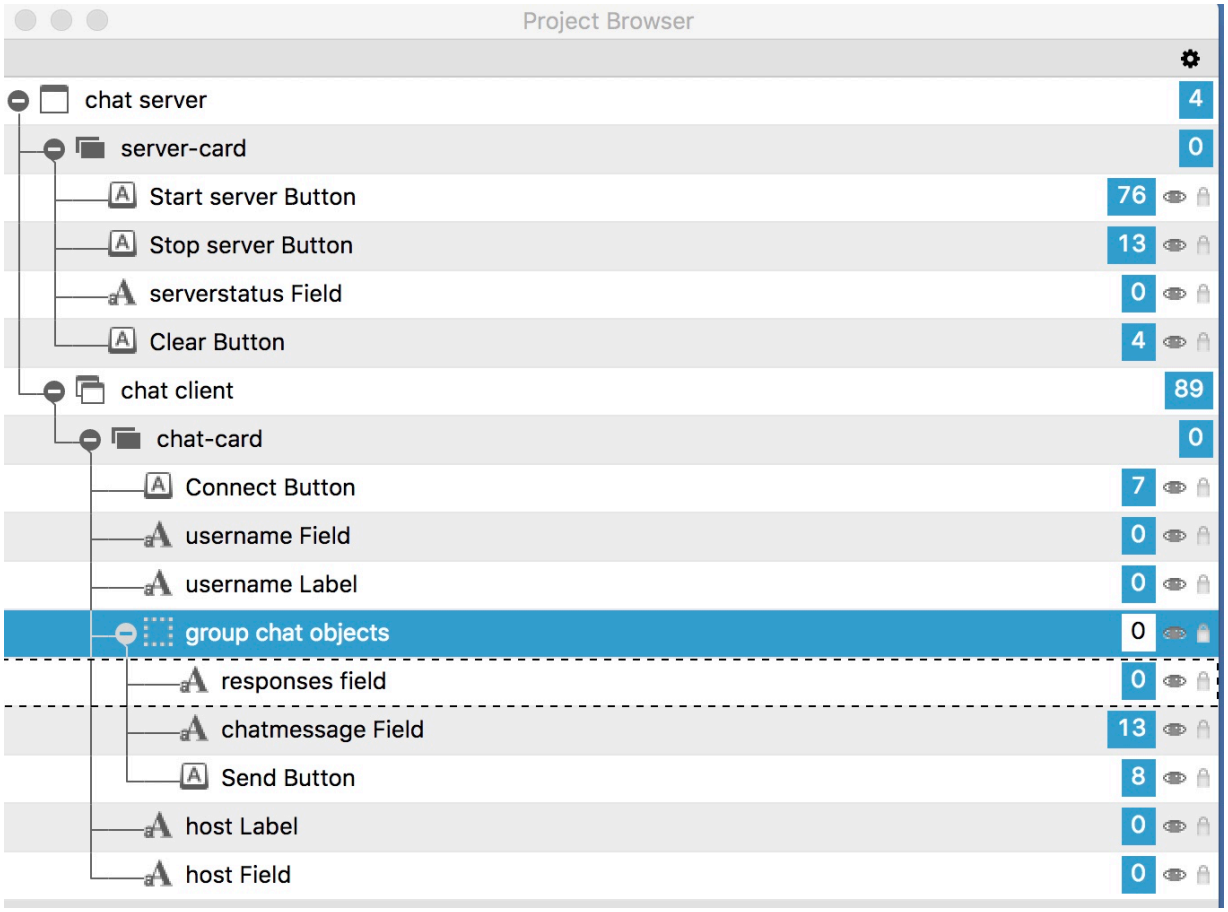
Chat Server



chat Server Stack script:

```
-- added 11/1/06 by O.K

on preOpenStack
  open stack "chat client" of this stack
end preOpenStack
```

_ _

server -card script: None

ON CARD “server-card”:

Start server Button:

```
-- declare a variable here to make it available to the entire script
local lChatterArray

-- when the mouse is clicked
on mouseUp
  -- provide visual feedback the server has been started
  disable me
  -- make it possible to stop the server
  enable button "Stop server Button”
  -- start accepting incoming connections
  -- the port has been chosen randomly, a high number
  -- is unlikely to be in use by anything else
  -- when a connection is received, send the message "chatConnected"
  accept connections on port 1987 with message chatConnected
end mouseUp

-- when a connection is recevied (this is first set up by mouseUp, above)
-- the "s" variable contains the address and port of the computer
-- that is connecting
on chatConnected s
  -- read in one line of data from the socket identified in the "s" variable
  read from socket s for 1 line
  -- remove any trailing return character
  put line 1 of it into tChatMessage
  -- add this new connection to the array containing a list of connections
  put tChatMessage into lChatterArray[s]
  -- call a handler to send a message to all clients informing them of the
  -- new connection
  broadcastToClients "*" & tChatMessage & " has joined the chat"
  -- put details of the new connection and a new line into the main field
  put tChatMessage && "connected" & return after field "serverstatus Field”
  -- start reading from the new connection contained in the "s" variable
  -- each time more data is received, call the chatMessage handler
  read from socket s with message chatMessage
end chatConnected

-- this handler is called when new data is received from a client
-- it is first set up by the chatConnected handler above
-- the variable "s" contains the host and port of the computer sending
-- the variable "data" contains the text that they sent
on chatMessage s,data
  -- put the chat message and a new line after the main field
  put data & return after field "serverstatus Field”
  -- send the chat message to all clients
  broadCastToClients data
  -- when more data is received from this client, send this message again
  read from socket s with message chatmessage
```

```
end chatMessage

-- this handler is called by the two handlers above
-- it sends the data contained in the "message" variable to all
-- the currently connected clients
on broadcasttoclients message
  -- get a list of all currently connected clients
  -- we add each client to this array when they connect in the handler above
  put keys(lChatterArray) into tChatterList
  -- cycle through all of the currently connected clients
  -- placing the host and port for each one into the variable "tSocket"
  repeat for each line tSocket in tChatterList
    -- send the data contained in the message variable to the client
    write message to socket tSocket
  end repeat
end broadcasttoclients

-- this message is sent when a client disconnects
-- the "s" variable contains the host and port of the client that disconnected
on socketClosed s
  -- look up the status of this client in the array we stored earlier
  put lChatterArray[s] into tChatter
  -- display this client disconnected to the main field
  put tChatter && "disconnected" & return after field "serverstatus Field”
  -- delete the reference to this client in the clients list array
  delete lChatterArray[s]
  -- tell all the remaining clients that this client has disconnected
  broadCastToClients "*" & tChatter && "has left"
end socketClosed
```

Stop server Button:

```
on mouseUp
  -- provide visual feedback that the server is stopped
  disable me
  -- make it possible to start the server again
  enable button "Start server Button”
  -- the openSockets contains a list of all socket connections that are open
  -- cycle through that list, putting each item in it into the variable "a"
  -- each time we go around the loop
  repeat for each line a in the opensockets
    -- close the connection contained in the variable "a"
    close socket a
  end repeat
```

```
end mouseUp
```

Clear Button:

```
on mouseUp
  -- clear the text in the main field
  put empty into field "serverstatus Field”
end mouseUp
```

chat client Stack script:

```
-- declaring a variable here will make it available to the entire script
-- the lChatSocket variable contains the host and port for the connection
local lChatSocket

-- this handler is called by the mouseUp handler in the
-- script of the connect button
-- it starts the connection to the chat server
on chatConnect
  -- clear the responses field
  put empty into field "responses Field"
  -- prevent the user from typing while waiting for the connection to open
  disable group 1
  -- open a connection to the host address specified in the host field
  -- using port 1987, a number chosen randomly.  a high port number
  -- is unlikely to conflict with another application
  -- send a message "chatConnected" when sucessfully connected to this host
  open socket field "host" & ":1987" with message "chatConnected"
end chatConnect

-- this handler is called by the mouseUp handler in the
-- script of the connect button
-- it stops the connection to the chat server
on chatDisconnect
  -- close the connection to the host and port stored in the lChatSocket variable
  close socket lChatSocket
  -- prevent the user from typing as the connection is now closed
  disable group 1
  -- change the connect button to show we are disconnected and to allow
connecting
  set the label of button "connect Button” to "Connect"
end chatDisconnect

-- this message is sent when the stack is closed
```

```
on closeStack
  -- call the disconnection handler (above)
  chatDisconnect
end closeStack

-- this message handler is set up in the chatConnect handler above
-- it is called when a connection is established
-- the "s" variable contains the host and port of the server we
-- are now connected to
on chatConnected s
  -- activate the controls in group 1 so the user can type
  enable group 1
  -- change the connect button to show we are successfully
  -- connected and to allow disconnecting
  set the label of button "connect Button" to "Disconnect"
  -- store the host and port of the server we are now connected to
  put s into lChatSocket
  -- send the user name to the chat server so it can broadcast
  -- this to other chat clients
  write field "username" & return to socket lChatSocket
  -- specify the message to be sent whenever any data is received from
  -- the chat server connection
  read from socket s with message chatReceived
end chatConnected

-- this message is called when data is received from the chat server
-- it is first set up in the handler chatConnected above
-- the variable "s" contains the host that connected
-- the variable "data" contains the data that was sent
on chatReceived s,data
  -- display the data that was sent
  put data & return after field "responses Field"
  -- specify that this message is to be sent again when more data is received
  read from socket s with message chatReceived
end chatReceived

-- this message is sent automatically in the event of an error
-- the "s" variable contains the host and port connected
-- the data variable contains the error message
on socketerror s,data
  -- prevent the user typing
  disable group 1
  -- show we are disconnected now and make it possible to start
  -- a new connection
  set the label of button "connect" to "Connect"
  -- display a dialog on the screen with the error message
  answer data
```

```
end socketerror

-- this message handler is called in the mouseUp handler of the
-- send button.  the "data" variable contains the message to send
-- it sends that data to the chat server
on chatMessage data
  -- send the user name followed by the data to the chat server
  -- connection is stored in the lChatSocket variable
  write field "username" & ":" & data to socket lChatSocket
end chatMessage
```

chat-card script: None

ON CARD “chat-card”:

Connect Button:

```
on mouseUp
  if the label of me is "Connect" then
    chatConnect
  else
    chatDisconnect
  end if
end mouseUp
```

Chatmessage Field:

```
on returnInField
  -- send a mouseUp message to the send button
  -- we use "click at" instead of "send mouseUp" so that
  -- we get the visual feedback associated with clicking on the button
  click at the location of button "Send Button"
end returnInField

on enterInField
  -- activate the handler above
  -- this is short hand for writing out the handler again, but would save
  -- time if we ever made the handler above more complex
  returnInField
end enterInField
```

Send Button:

```
on mouseUp
  -- chatMessage is a message handler in the stack script
  -- send this message together with the contents of the field
  -- the user typed in
  chatMessage field "chatmessage Field"
  -- clear the field so the user can type another message
  put empty into field "chatmessage Field"
end mouseUp
```