Inhaltsverzeichnis

Aufbau des Chats

Ich habe versucht, einen Chat selbst zu programmieren, der auf meiner Homepage läuft. Nun gibt es verschiedene Arten Chats. Chats können in .exe-Programmen laufen, als Java-Applets oder einfach als JavaScript im Browser.
Ich habe, damit der Benutzer sich nicht erst lange ein Programm herunterladen muss, den Chat auf JavaScript und ASP (Active Server Pages) aufgebaut.

Die Clientseite

So sieht der Client (also der Benutzer) meinen Chat (hier in meiner Homepage eingebunden):

Abbildung 1: Mein Chat

Der Chat ist in mehrere Frames aufgeteilt, hier verdeutlicht:

Der von den roten Linien abgetrennte, große Bereich rechts unten ist der Chat.

Er besteht aus 4 Unterframes:
Options.asp: Hier können Sie den Nickname und die Farbe wechseln.
Leer.asp: Ist zu Anfang leer.
List.asp: Hat die Höhe Null, empfängt neue Mitteilungen und schreibt diese in leer.asp
Add.asp: Hier können Sie Mitteilungen schreiben.

Abbildung 2: Frames des Chat

Was der Chat alles bietet

Symbole

Abbildung 3: Symbole im Chat

Klicken Sie einfach auf ein Symbol, um es auszuwählen. Geben Sie Ihren Text ein und drücken Sie Enter.

Farben

Abbildung 4: Mehrere Farben

Wählen Sie einfach eine neue Farbe und klicken Sie auf Ändern. Jetzt lassen sich die Einträge der verschiedenen Leute gut unterscheiden!

Teilnehmerliste

Abbildung 5: Teilnehmerliste

Die Teilnehmerliste ist ein praktisches Werkzeug, um neben dem Chat-Fenster anzuzeigen, wer im Chat ist. Mit einem Klick auf das Fester wird dieses geschlossen. Die Teilnehmerliste aktualisiert sich alle 20 Sekunden von selbst. Sie zeigt auch die Farben der Benutzer an.

Anzeige von Usern, die den Chat verlassen haben

Benutzer, die den Chat verlassen wollen, können einfach Ihren Browser schließen. Nach 15 Sekunden wird automatisch eine Meldung angezeigt, dass der Benutzer den Chat verlassen hat. Natürlich in der passenden Farbe.

Zeitanzeige

Zeigt vor jedem Eintrag an, wann der Eintrag geschrieben wurde.

Wird die Einstellung der Zeitanzeige verändert, so wird die Seite in der Mitte neu geladen – alle Einträge älter als fünf Minuten verschwinden.
Auch praktisch, um aufzuräumen.

Abbildung 7: Zeitanzeige

Flüstern

Zum Flüstern einfach den Text eingeben und auf Flüstern klicken. Anklicken, zu wem geflüstert werden soll und los geht's.

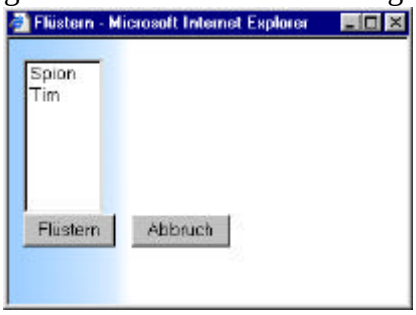

Abbildung 8: Flüstern

Dritte bemerken nicht, dass geflüstert wurde.

Abbildung 9: Flüstern

Die Technik dahinter: ASP

Nun stellt sich die Frage: Wie kommen die Texte von einem Computer zum anderen? Ein auf HTML und JavaScript basierender Chat kann keine direkte Verbindung zwischen Computern aufstellen.
Also benötigen wir einen **Server**.
Dies kann nicht ein normaler Server sein, auf dem ich Webspace habe, sondern es muss ein Server sein, der meinen Programmcode verarbeiten kann. Nun sind Server mit ASP-Unterstützung ziemlich teuer; außerdem läuft ASP nur auf Windows-Servern.
Zum Glück unterstützt der PWS (Personal Web Server), der bei Windows 98 mitgeliefert ist, ASP. Somit kann ich meinen Chat im normalen Netzwerk testen.

Was steckt hinter ASP?

ASP ist, wie auch PHP oder Perl, eine Programmiersprache, die auf dem Server abläuft. Ein ASP-Programm liefert eine normale HTML-Webseite zurück. Hier erst mal eine ganz einfache ASP-Seite:

```
<HTML><HEAD><TITLE>ASP-Test</TITLE></HEAD>
<BODY>
Dies ist ein Test einer ASP-Seite.<br>
<%
If Hour(Time) < 12 Then
      Response.Write "Guten Morgen!"
Else
      Response.Write "Guten Tag!"
End If
%>
</BODY>
</HTML>
```

Diese Seite ist einer HTML-Seite ziemlich ähnlich, nur dass zeitabhängig ein anderer Text erscheint. Anders als bei JavaScript führt aber nicht der Client, sondern der Server das Programm aus. Der Benutzer sieht, wenn er sich den Quellcode der Seite anzeigen lässt, nur das Ergebnis.

ASP-Seiten werden normalerweise in Visual Basic Script geschrieben, können aber auch in JavaScript oder beides gemischt geschrieben werden.
ASP-Seiten können auch auf das Dateisystem des Servers zugreifen, deshalb erlauben auch nur wenige Provider ASP.
ASP-Code steht immer zwischen <% und %>.
Response.Write fügt einfach HTML-Code in das Dokument ein.
Variablen, die in ASP definiert wurden können auch direkt mit <%=Variable%> ohne Response.Write eingebunden werden.

Wie speichert ASP die eingegebenen Texte?

Zunächst muss jemand den Text schreiben. Das geschieht in der „Add.asp“.
Dies ist das relativ einfache Formular:

```
<FORM Action="Add.asp" Method="post" name=f1>
<INPUT name=text size="50">
<INPUT type="submit" value="Senden">
</FORM>
```

Dieses Formular schickt den eingegebenen Text an sich selbst.
ASP kann das übermittelte Formular nun auslesen:

```
<% User=Session("ChatNickname")
t= Request.Form("Text")
if len(t)>1 then 'Wenn Text vorhanden ist
      AddLine User & ": " & t
end if %>
```

Hier verwende ich das Session()-Objekt. Alle Einträge im Session-Objekt, die mit `Session(„Feld“)=“Wert“` zugewiesen werden, werden auf dem Server gespeichert, bis der Benutzer 20 Minuten lang nicht mehr auf eine Seite dieses Servers zugegriffen hat. Die Session()-Einträge sind für alle ASP-Seiten auf dem Server verfügbar, allerdings hat jeder Benutzer eine eigene Session. Im Session()-Objekt habe ich den Nickname des Benutzers gespeichert. `AddLine` ist eine selbstgeschriebene Funktion, die eine Zeile an den Chat anhängt. *Hinweis: Das angegebene Programmstück ist vereinfacht, da der Benutzer auch noch Symbole und eine Farbe wählen kann.*

AddLine schreibt die angegebene Zeile in eine Datei. Die Datei heißt in unserem Fall „04.12.01Roomwww.danielgrunwald.de.htm“. Somit kann man bei meinem Chat auch noch Tage später gucken, was die anderen geschrieben haben, wenn man Zugriff auf die Chat-Datei hat.

Wie bekommt der Browser mit, dass sich etwas verändert hat?

Der Browser kann nicht vom Server aus benachrichtigt werden, dass jemand etwas neues geschrieben hat, also muss er in einem bestimmten Zeitintervall eine Seite aktualisieren. Das ist hier der „unsichtbare“ Frame mit der Seite „list.asp“. Da diese Seite unsichtbar ist, sieht der Benutzer kein Flackern. Wie aktualisiert sich diese Seite aber nun?
Ganz einfach: Mit JavaScript.
JavaScript wird auf dem Client ausgeführt und kann damit das Aktualisieren erledigen. Nun müssen wir aber zwischen JavaScript und ASP einen dicken Trennstrich machen, auch wenn die ASP-Anweisung Response.Write auch JavaScript erstellen kann.
Dies ist die Aktualisierungsanweisung aus der ‚list.asp’:

```
<SCRIPT LANGUAGE="JavaScript">
function neuLaden(){
location.replace('List.asp?t=<%=li%>');
}

window.setTimeout('neuLaden()',3000);
```

```
</SCRIPT>
```

Was der Parameter `t=<%=li%>` bedeutet, wird später erklärt.
Wenn List.asp nun etwas Neues gefunden hat, wird es wie folgt ausgegeben:

```
<SCRIPT Language="JavaScript">
function AddLines()
{
parent.View.document.writeln ('Daniel Grunwald: Hallo!<br>');
parent.View.scrollBy(0,30);
}
AddLines();
</SCRIPT>
```

Die Anweisung sorgt dafür, dass im Frame mit Leer.asp der Text angehängt wird, die Anweisung darunter lässt den Bildschirmausschnitt falls nötig herunterscrollen.

Wie bekommt List.asp mit, welche Texte noch fehlen?

Nun ja, dies ist ein wirkliches Problem. Wir wollen ja nicht alle 3 Sekunden den kompletten Text des Tages unten angehängt haben, sondern nur dass, was fehlt. Dafür ist eine ziemlich komplizierte ASP-Programmroutine nötig, das Herzstück meines Chats.
Es ist die `function ViewLastMinutes(ZeigeZeit) 'Zeigt die neuesten Einträge an`

Diese Funktion rechnet zunächst von der aktuellen Uhrzeit fünf Minuten ab. Dann liest sie den Wert „t“ aus. „t“ ist die Nummer des neuesten Eintrages, den der Browser hat und wird von der Aktualisierungsroutine übergeben.
Nun wird die Chat-Datei des Tages geöffnet und alle Einträge einzeln überprüft, ob diese jünger als 5 Minuten sind. Damit ist mein Chat einer der wenigen, die beim Betreten die letzten 5 Minuten anzeigen. Ist der Eintrag jung genug, wird anhand der Eintragsnummer überprüft, ob der Client den Eintrag bereits gesehen hat. Falls er den Eintrag noch nicht kennt, wird dieser wie oben angegeben an ihn übermittelt.
Außerdem prüft mein Programm noch, ob ein Text geflüstert wurde, falls ja, sendet es den Text nur, wenn der Benutzer Absender oder Empfänger des geflüsterten Textes ist.
Hier der Programmcode (vereinfacht):

```
function ViewLastMinutes(ZeigeZeit) 'Zeigt die neuesten Einträge an
Response.Write "<SCRIPT Language=""JavaScript"">"
Response.Write "function AddLines(){" 'Sende zunächst den SCRIPT-Tag
St= Hour(Time) : Min=Minute(Time)-5  'Hole aktuelle Zeit minus 5 Minuten
Do While Min<0   '  Ändere  16:-03 in 15:57
       Min=Min+60
       St=St-1
loop
if St<0 then St=0:Min=0:Sec=0  ' Einträge vom vorherigen Tag nicht anzeigen
strli=request.QueryString("t") ' Wie viele Einträge hat der Client schon?
if strli="" then ' keine Angabe = 0
      li=0
else
      li=cint(strli) ' wandle String in Zahl um
end if
i=0 'Zähler
Set Txt=GetTextStream(ForReading) 'Öffne Datei
do while not txt.atendofstream 'Solange nicht am Dateiende
      i=i+1  'Zähler erhöhen
      t=txt.read(8)  'Lese ersten 8 Zeichen ein (Uhrzeit)
      couldview=0  'Überprüfe, ob der Eintrag jünger als
      if hour(t) > St then 'fünf Minuten ist
```

```
            couldview=1
      elseif hour(t) = St then
            if minute(t) > Min then
                  couldview=1
            elseif minute(t) = min then
                  if second(t) >= sec then couldview=1
            end if
      end if
      if couldview=1 then  'Wenn der Eintrag jung genug ist:
            if li<i then  'prüfe ob Client den Eintrag schon hat
                  text=txt.readline  'Lese den Rest der Zeile
                  if ZeigeZeit then text= t & text  'Zeitangabe
                  %>
            parent.View.document.writeln ('<%=replace(Text,"'","\'")%>');
            parent.View.scrollBy(0,30); // Schreibe den Text mit JavaScript
                        // in anderen Frame und scrolle dort.
                  <%
                  li=i  ' jetzt hat der Browser den Eintrag
            else 'Wenn der Browser den Eintrag schon hat
                  txt.skipline 'Überspringe die Zeile
            end if
      else 'Wenn der Eintrag zu alt ist
            txt.skipline 'Überspringe die Zeile
      end if
loop
txt.close 'Schließe Text-Datei
%>
   } // end function   Ende der JavaScript-Funktion AddLines
<%if strli="" then
      response.write  setTimeout('AddLines();',1500);" 'Rufe AddLines()
                  ' beim ersten Mal zeitverzögert auf, damit Leer.asp
                  ' auch schon geladen ist
else
      response.write "AddLines();"
         ' Ansonsten rufe AddLines() direkt auf
end if%>
   </SCRIPT>
<%
end function%>
```

Ausgewählte Probleme bei der Programmierung

Teilnehmer

Ein ziemlich großes Problem ist die Verwaltung von Teilnehmern. Dazu zählt die Teilnehmerliste, die Farben und die Meldung „Benutzer verlässt den Chat“.
Jeder Benutzer hat, wie oben erklärt, sein eigenes Session()-Objekt. Dort kann die Farbe des Benutzers problemlos gespeichert werden. Wenn der Benutzer in der Add.asp nun einen Text abschickt, so liest Add.asp die Farbe aus dem Session()-Objekt aus und fügt vor den eingebenen Text „<font color=<%=Farbe%>>“ ein.
Aber wie bekommt die Teilnehmerliste mit, welche Farben die anderen haben?
Die Farbe müsste dazu Session-unabhängig erreichbar sein. Dafür gibt es in ASP das Application()-Objekt. Wie ins Session()-Objekt können hier Werte für längere Zeit gespeichert werden. Die Application()-Werte sind Session-unabhängig erreichbar und bestehen, solange der Server nicht neu gestartet wird.
Damit gibt es eigentlich keine Probleme, aber wie werden die Benutzer aus der Liste entfernt?

Benutzer verlässt den Chat

Was passiert eigentlich, wenn der Benutzer den Browser schließt?
Gar nichts!!!
Und genau an dieser Stelle setzt mein Chat an. Wenn ein Benutzer 15 Sekunden lang nicht mehr neue Mitteilungen abruft, so hat er den Chat verlassen. Zwar ist seine Session noch längst nicht abgelaufen, aber die Meldung kann nun angezeigt werden.
Im Application()-Objekt wird zusätzlich zu Nickname und Farbe auch noch gespeichert, wann List.asp sich das letzte mal aktualisiert hat. Aber welche Programmroutine kontrolliert nun die Werte? Wenn sich der letzte Benutzer verabschiedet wird keine ASP-Seite mehr aufgerufen. Für dieses Problem habe ich bisher noch keine Lösung gefunden. Aber wenn sich nur einer von mehreren Benutzern verabschiedet, so prüft List.asp, die von einem übergebliebenem Benutzer aufgerufen wird, alle Benutzer. Wenn ein Benutzer den Chat verlassen hat, wird einfach sein Benutzername im Application-Objekt auf „ CLOSED “ gesetzt. Die Meldung „Benutzer verlässt den Chat“ wird angezeigt. Erst nach weiteren fünf Minuten wird sein Benutzername erneut geändert: Diesmal von „ CLOSED “ auf „“. Damit können andere Benutzer auch den Platz im Session()-Objekt benutzen. Warum diese Zeitverzögerung? Das liegt am Flüstern. Wenn ein neuer Benutzer den Chat betritt, so sieht er, was die anderen in den letzten fünf Minuten gesagt haben. Er soll aber nicht sehen, was sein Vorgänger, dessen Platz er jetzt eingenommen hat, geflüstert hat.
Falls der Benutzer, der den Chat verlassen hat, aber der letzte im Chat war, so bleibt er im Chat, bis der Server neugestartet oder der Chat wieder benutzt wird. Dann wird die Meldung „Benutzer verlässt den Chat“ womöglich Stunden zu spät angezeigt.

Doppelte Benutzernamen vermeiden

Wenn zwei Benutzer den gleichen Nickname wählen und womöglich auch noch die gleiche Farbe, so können andere Benutzer nicht sehen, wer etwas gesagt hat oder zu wem sie gerade flüstern.
Mein Chat hat eine einfache Routine um doppelte Benutzernamen zu vermeiden. Bei jedem Ändern des Benutzernamens prüft das Programm, ob der Name bereits existiert und hängt falls nötig [2], [3] usw. an den Benutzernamen an.

Symbole verwenden

Der Benutzer kann zusätzlich zu seinem Text auch noch Symbole senden. Er muss einfach nur vor dem Klicken auf „Senden“ auf ein Symbol klicken.
Welches Symbol gewählt ist, wird mit dem HTML-Formular an Add.asp übergeben. Add.asp fügt einfach den HTML-Code „`<IMG SRC="images/<%=image%>.gif">`“ wie bei den Farben vor den Text. `<%=image%>` ist dabei der Wert, der vom Formular übergeben wurde.

Bildverzeichnis:

Daniel Grunwald

Daniel Grunwald
Bierberger Str. 9
31174 Schellerten

daniel@danielgrunwald.de
www.danielgrunwald.de