Next2Friends AJAX.NET chat v1

Author: Lawrence Botley  
Date: 11 October 2007  
Platform: ASP.NET 3.5 / Linq Framework **/ Visual Studio 2008 beta**

1. **Overview**
2. **Chat Application Class**
   1. **Linq**
   2. **Poll requests**
   3. **Authorisation and security**
3. **In-Browser / DHTML Interface**
   1. **Online Friend Lister**
   2. **Chat console**
   3. **Using AJAX**

**1.0 Overview**

The N2F Ajax.NET chat application relies on a simple message posting/ retrieval mechanism. The AJAX.NET framework will be used to periodically poll the chat server at set intervals for any new messages and who return a list of friends who are online now. When a user sends a message to the server it is inserted into the recipients inbox (which is stored in a single array on the server). At the set interval the member will poll the server for any new messages. The messages are then deleted from the inbox so as not to send them again and returned to the browser client.

In the browser; should new messages be received by the Ajax call, the chat window is updated or created at the bottom left of the screen. Also , when users logout or login to the system, there status is updated the *OnlineFriendLister*

1. **Chat Application Class**

The Chatapplication is the accompanying project attached to this document. The Chat class contains a set of static methods that will allow members to login, logout, post and receive messages. All the messages are held in the Inbox list, this list will reside on the web server running ASP.NET and be stored as a globally accessible persistent variable.( Application[“InboxList”] ). Bear in mind that when the application scales and a cluster of IIS servers are needed, this Global variable should be readable/writable/accessible from each of the servers.

The Chat class contains a number of functions that allow the member to:

* StartChat() – logs into the system
* EndChat() – user closes browser or logs out of N2F
* SendMessage() – members posts a message to another member
* GetNewMessages() – gets all the new messages from the server
* GetFriendsOnline() – returns a list of all the users friends online right now These methods can be adapted if required.
* SetOnlineStatus() – This sets the members online status. This can be Online,Away,CustomMessage. These values will be returned when a user does a GetFriendsOnline() request.

**2.1 Linq**

Since the InboxList array could be large (hundreds of thousands ) then a fast mechanism needs to be used to search, post and retrieve messages from the array, Linq has been used to accomplish this. Linq is a new query language for objects in C# and is included in .NET 3.5.

In order to use Linq you must download Visual Studio 2008 beta. This can be downloaded here:

<http://msdn2.microsoft.com/en-us/evalcenter/bb655861.aspx>

In case you aren’t familiar with Linq at all. The following code selects an Inbox object from the InboxList where the MemberID = “1234”

string MemberID = "1234";

var MemberInbox =

from i in InboxList

where i.MemberID == MemberID

select i;

For a tutorial on linq go to: <http://msdn2.microsoft.com/en-us/vcsharp/aa336746.aspx>

Full code examples for the Chat class can be found at the end of this document in A.1

**2.2 Poll requests**

The client browser should make a poll request to check for new messages every 2 seconds by calling *GetNewMessages()* and another should be made every 20 seconds to determine new friends online or offline by calling *GetFriendsOnline()*

There will also need to be a final cleanup mechanism that removes members that have not made a polling call for longer than 1 minute. This would denote that the user has not log out of the system but simply closed the browser. This cleanup mechanism should be a separate process that runs in its own thread. Again, this will use Linq to perform a simple remove query.

* 1. **Authorisation and security**

For the development of this stage of the application there will be no password or token authentication. For the next phase of the development a token will be returned when the user successfully sends a password to the StartChat() method. The token will then be given on each request to the server to determine if the user is authorised.

**3.0 In-Browser / DHTML Interface**

From the end users perspective there will be two parts to the chat interface.

* *OnlineFriendLister* – this will lists all the friends that are currently online now
* *ChatConsole* – a chat console window for each of the friends that you are chatting to

**3.1 Online Friend Lister**

From the end users perspective there will be two parts to the chat interface. The first part is a list of all the member’s friends that are currently online (Online Chat Lister). This example has been taken from Gmail talk.

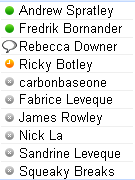


Figure 1.1 - Gmail talk online chat lister

The N2F AJAX.NET chat system will require similar features however, only friends that are currently online will be displayed (this is because a member may have hundreds of friends and it is not practical to show them all)

A chat icon will appear next to the friends that currently have a chat window open with them(white speech bubble)

The member’s friends are stored in a session variable (Session[“Friends”]) and every time a FriendOnline poll is performed, the MemberIDs are waiting serverside already, reducing the number of data transmitted to the server. When a new poll request is made via AJAX, the *GetFriendsOnline( )*method is called and a list of Friend objects will be returned in a list. It is up to you how you return this data to the client browser. I would recommend a string of delimited parameters to keep bandwidth footprint small.

**3.2 Chat Console**

The Chat console will popup when you either click on a friend’s name on the Online Friend Lister of when a friend sends a message to you. The window will look something like below. For the design of this application do not be concerned with ascetics as we will have our design change the look and feel

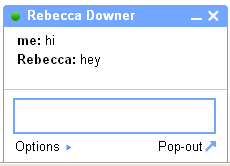


Figure 3.2 – Gmail Talk message console

Once the *GetNewMessages()* Ajax method has been Called and new messages are returned, they must be retained in a persistent data structure that it is accessible for the lifetime of being logged in. If the user is clicking around the site the chat console windows will always stay static and never reload with the page. For this a Frameset must be used to separate the two layers of the chat application and the web page behind it. (have a look at the chat windows in GMAIL: as you navigate around the site they don’t disappear or even flicker)

If more than one chat is taking place then the chat consoles will be stacked horizontally like below:

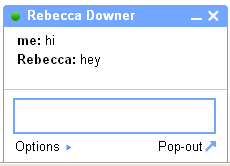
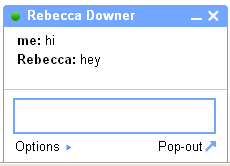
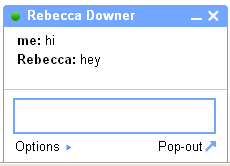


Figure 3.3 – Multiple chats are stacked horizontally

Should the number of chats exceed the screen size then they will be stacked upon a new row vertically.

For this program we have chosen to use AJAX.NET. It could be problematic using the update panel for the chat window consoles as they would need to be instantiated dynamically client side and this does not seem possible (??). A custom JavaScript script should be coded to instantiate a new chat window ( create a new <DiV> object in the browser DOM) and place it in the appropriate point on the screen (defaulting to bottom right).

The chat windows will pop up in the following two instances,

1 When the user single clicks on a friend in the Online Friend Lister.

1. When an incoming message is returned from polling the server.

From here, when the user enters a message into the box and hits the return key, the SendMessage() method is called on via an AJAX method. The console is then updated with the outgoing chat message IE (**me**: hi).

The GetNewMessages() method should be called at intervals of 2-3 seconds via an AJAX method. If the Ajax returns a list of new messages then they should be updated (or created) in the chat console in the appropriate users chat box.

A.1 Code example: this code is the main entry point for the attached C# project.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Next2Friends.Chat

{

/// <summary>

/// THIS APPLICATION MUST RUN IN VISUAL STUDIO 2008 WITH .NET 3.5 TO SUPPORT LINQ

/// </summary>

class Program

{

static void Main(string[] args)

{

// Initialise the server array. This would be an Global application

// variable accessible by all clustered servers in the ASP.NET web Application

Chat.InitialiseServerInboxList();

// Lawrence logs in giving his perminant Unique Chat/member ID and NickName

Chat.StartChat("12345","Lawrence");

// Hamid logs in

Chat.StartChat("67890","Hamid");

// lawrence sends a message to Hamid giving his nickname as well as memberid

// they are both passed to reduce refence lookup request from the list array

Chat.SendMessage("67890", "12345", "Lawrence", "Hello Hamid");

// Hamid sends a message to Lawrence

Chat.SendMessage("12345", "67890","Hamid", "Hello Lawrence");

// Lawrence polls the server and asks for any new messages. All messages are

// removed from the list

List<ChatMessage> LawrenceChatMessages = Chat.GetNewMessages("12345");

// Hamid polls the server and asks for any new messages

List<ChatMessage> HamidChatMessages = Chat.GetNewMessages("67890");

// write Lawrences new messages

foreach (ChatMessage c in LawrenceChatMessages)

{

Console.WriteLine(c.FromNickName + " said: " + c.Message);

}

// write Hamids new messages

foreach (ChatMessage c in HamidChatMessages)

{

Console.WriteLine(c.FromNickName + " said: " + c.Message);

}

///// if you check the server again, there will be no new messages

// Lawrence polls the server and asks for any new messages

List<ChatMessage> LawrenceChatMessages1 = Chat.GetNewMessages("12345");

// Hamid polls the server and asks for any new messages

List<ChatMessage> HamidChatMessages1 = Chat.GetNewMessages("67890");

foreach (ChatMessage c in LawrenceChatMessages1)

{

Console.WriteLine(c.FromNickName + " said: " + c.Message);

}

foreach (ChatMessage c in HamidChatMessages1)

{

Console.WriteLine(c.FromNickName + " said: " + c.Message);

}

// set hamids online status to CustomMessage and set the message

Chat.SetOnlineStatus("67890", OnlineStatus.CustomMessage, "Gone for a coffee!");

// returns a list of all lawrence's friends online. (this will be Hamid only)

List<Friend> FriendsOnline = Chat.GetFriendsOnline("12345", new string[] { "67890" });

// write hamids Online status

foreach (Friend f in FriendsOnline)

{

Console.WriteLine(f.NickName + " status = " + f.OnlineStatusToString());

}

}

}

}