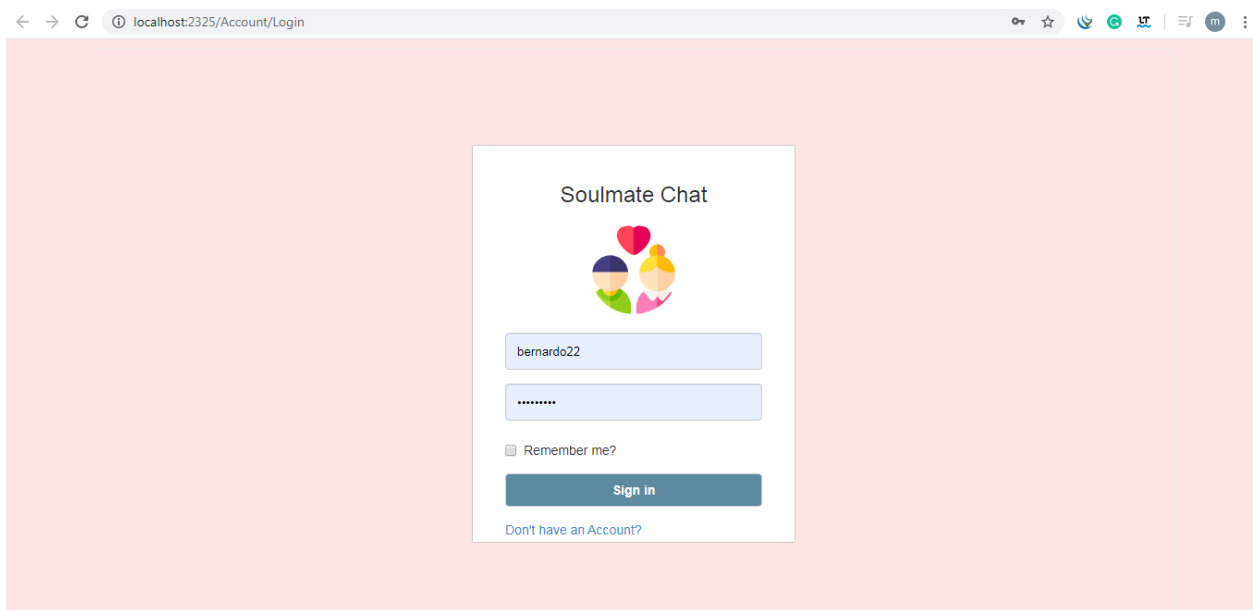


How to develop a chat system? The SoulMate Chat project

Martin Rupp

SCIENTIFIC AND COMPUTER DEVELOPMENT SCD LTD

Version 1



In this tutorial, we will learn how to develop chat applications in ASP.NET. The source code is attached and can be used for free for any possible purposes.

Our project consists of developing a chat application for people who have 'common affinities' so that they can match together and chat. This is just a sample project for educational purposes. It is untested and provides no guarantee whatsoever in terms of quality.

Step1. Designing the Chat system

A chat system, you will find tons over the internet. It is one of the most ancient web applications which is known and chat systems have their roots merged with the birth of the internet itself! Think about IRC for instance!

Nevertheless, chats are extremely popular nowadays. For example Skype, WeChat, telegraph, etc.

Chats are used by dozens of millions of users every day and a lot of business and collaborative work is conducted there! Chats can run as desktop applications or as mobile applications, both usage seems to be prevalent as well.

Now with the 'modern' web, chats can be very fluid, pleasant, and provided with a beautiful user interface - far from the old IRC systems. It is often a pleasure to write messages on modern chats because they are so elegant and fast.

Here, we do not pretend to provide such a chat system, we only aim to demonstrate some of the aspects of the technology involved (think about it as an IT school project).

The chat that we wish to develop will provide a way to find 'matching profiles' among the chat registered participants. These matching parties can therefore be contacted directly. What we will develop is an unfinished and untested application.

How will we provide the 'matching system' is one question and what framework will we use to provide the chat client and the chat server?

The matching system will consist of psychological quizzes. Therefore from the scoring, one can see what profiles are likely to be similar. We will provide questions that can lead to a unique personality trait definition.

As for the chat system, we will use an existing framework. There is a good open-source chat system written in C# using the Signal-R framework. We will use it as a basis and modify it for our purposes.

About the Signal-R framework

ASP.NET SignalR is a library for ASP.NET developers that allows them to create "real-time" web applications.

That 'real-time' functionality is the ability to have the server pushing content to connected clients instantly as this data becomes available, rather than having the server wait for a client to request new data (polling).

Signal-R usually relies on web sockets but has a fallback mechanism when such web sockets are not available. This is the same approach as the Bayeux channels, Meteor or Grizzly.

Signal-R is not a real "real-time" unless it is running on an RTOS platform but it provides real speed for interactive web applications such as chats, using RPC.

Signal-R is ideal for chat systems, indeed it provides the following features:

- Handles connection management automatically.
- Sends messages to all connected clients simultaneously. For example, a chat room.
- Sends messages to specific clients or groups of clients.
- Scales to handle increasing traffic.

Modifying the chat framework

The Signal-R chat can be downloaded [there](#). It is an MVC .NET application that can be used for the web and mobile phone applications.

It is also using the Knockout.js javascript framework.

When modifying a model-view-controller application, one should certainly add more views, more models, and more controllers. Here we aren't going to do that, we will simply add forms that will run independently of the MVC code, eventually sharing the same database.

This is, of course, not the recommended way to modify an MVC application but there we wish simply to demonstrate some technological aspects.

We will not use the mobile application client. We will only develop a web system that will run in a modern browser.

When the user registers, the chat offers only 4 basic icons as possible avatars. This is not enough, so we will have to add as many avatars as we can.

Additionally, there are but a few smileys, and we wish to add more with various types.

The chat does not also provide a profile for a user, e.g. a place where the chat user can input their birthdate, location, age, etc... we will have to add this as well.

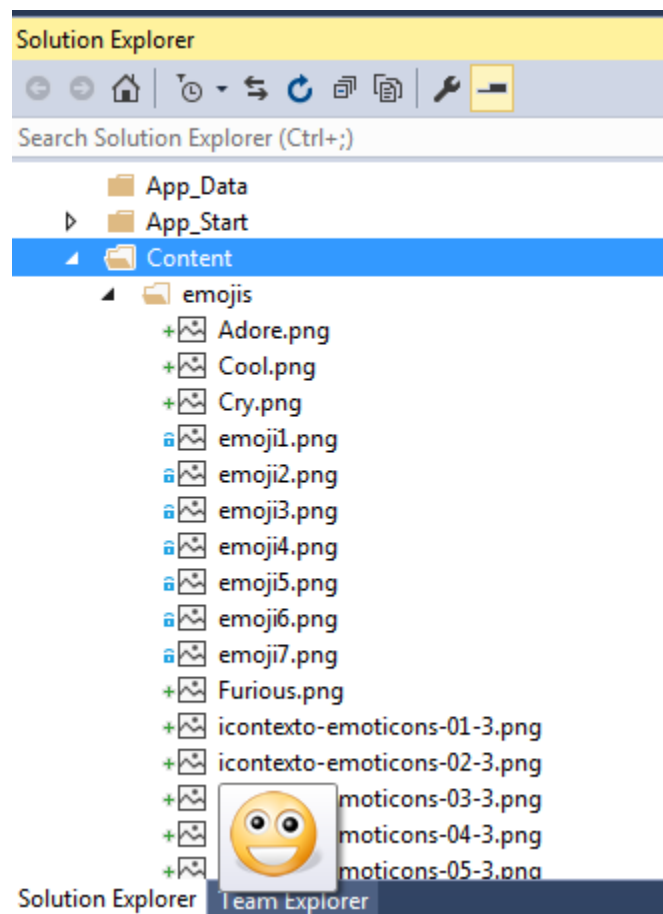
Finally, we must 'plug in' our psychological tests in such a way that they are fully integrated with the chat. We can use bootstrap for this. Bootstrap and JQuery allow us to smoothly display JavaScript pop-up dialogs in the browser.

Adding Emojis

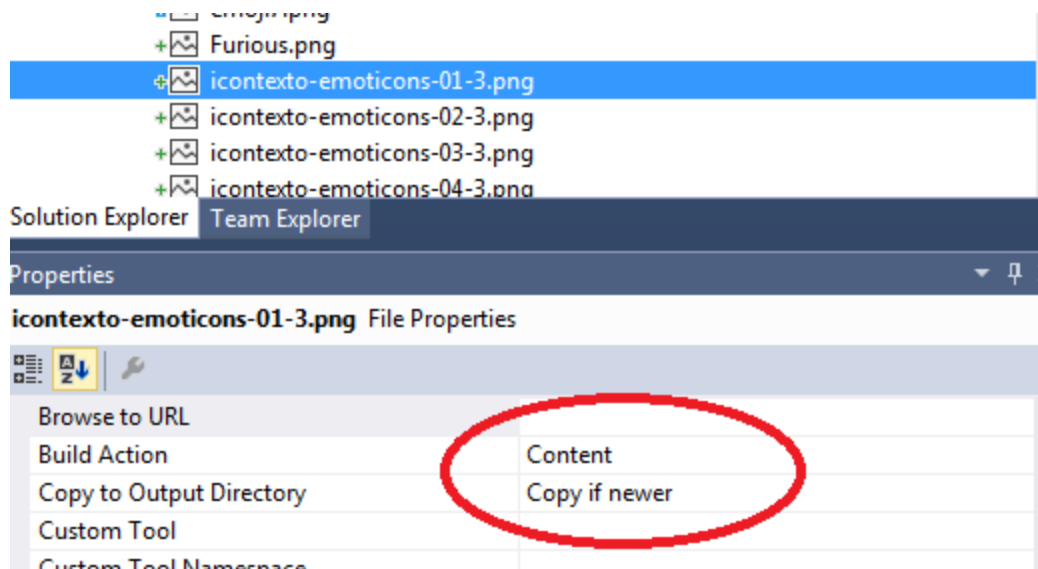
As we said, there are but a few Emojis ('Emoticons') in the base chat.

We must download new ones, either we design them ourselves, or we fetch them from some website that collects emojis.

We need to add them to the `emojis` folder, in the `content` folder of the chat.



Make sure that they get properly compiled and copied when a build is done. You can batch-do this.



We need to map our new emojis to a command such as :-) or :->
For this, we go to *BasicEmojis.cs* and add some new mappings:

```
content = content.Replace("12/", Img("icontexto-emoticons-01-3.png"));
content = content.Replace("13:",
Img("icontexto-emoticons-02-3.png"));
content = content.Replace("14Q",
Img("icontexto-emoticons-03-3.png"));
content = content.Replace("15:",
Img("icontexto-emoticons-04-3.png"));
content = content.Replace("16:",
Img("icontexto-emoticons-05-3.png"));
content = content.Replace("17Q",
Img("icontexto-emoticons-06-3.png"));
content = content.Replace("18:",
Img("icontexto-emoticons-07-3.png"));
content = content.Replace("19:",
Img("icontexto-emoticons-08-3.png"));
content = content.Replace("20]",
Img("icontexto-emoticons-09-2.png"));
content = content.Replace("21;",
Img("icontexto-emoticons-10-3.png"));
content = content.Replace("22/",
Img("icontexto-emoticons-11-3.png"));
content = content.Replace("23Q",
```

```

Img("icontexto-emoticons-12-3.png"));
        content = content.Replace("24/",
Img("icontexto-emoticons-13-3.png"));
        content = content.Replace("25:",
Img("icontexto-emoticons-14-3.png"));
        content = content.Replace("26]", Img("Laugh.png"));
content = content.Replace("27;", Img("Pudently.png"));
content = content.Replace("28]", Img("Struggle.png"));
content = content.Replace("29/", Img("Study.png"));
content = content.Replace("30Q", Img("Sweet-angel.png"));

```

We need also to convert these images into base64 format:

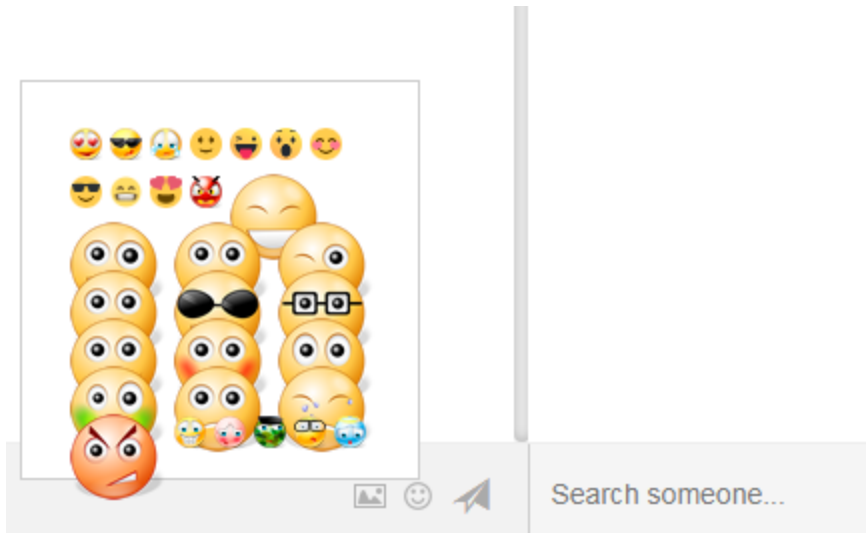
```

namespace Chat.Web.Helpers
{
    2 references | akouki, 1059 days ago | 1 author, 1 change
    public class StaticResources
    {
        /// <summary>
        /// Base64 Images
        /// </summary>
        public static List<string> Avatars = new List<string>()
        {
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAADEAAAAwCAYAAAC4wJK5AAAA8mJLR0QA/wD/AP+gvaeTAAAAACXBIIwXMAAA:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAADAAAAAwCAYAAABXAvmHAAAA8mJLR0QA/wD/AP+gvaeTAAAAACXBIIwXMAAA:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAADAAAAAwCAYAAABXAvmHAAAA8mJLR0QA/wD/AP+gvaeTAAAAACXBIIwXMAAA:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAADAAAAAwCAYAAABXAvmHAAAA8mJLR0QA/wD/AP+gvaeTAAAAACXBIIwXMAAA:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:
            "data:image/false;base64,iVBORw0KGgoAAAANSUgAAASwAAAEsCAYAAAB5FY51AAAAACXBIIwXMAAASTAAALewEAMPwYAAAAGXRFwHRTb2Z0d2FyZQ:

```

You can create a helper to convert automatically all the directory files into base64 strings, then copy them and paste them into the CS code.

These images will be displayed in the browser by javascript when the user picks up an emoji to send it to the chat.

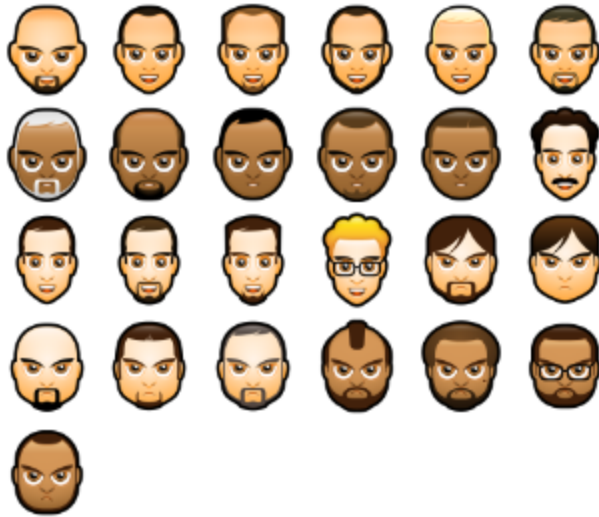


We could have changed the size of the display windows but we found it was more fun to have them mixed!

As for the new avatars (to be chosen at registration time) we display them by categories.



men #1 ▶ men #2



men #3 ▶ girls #1



Select your Avatar



men #1



men #2 ▶ men #3 ▶ girls #1 ▶ misc #1



Register

[Already have an Account?](#)

Adding user information

In the base chat, user information is scarce and almost non-existent. Therefore we modify the database structure to add extra-fields.

	avatar	sex	town	country	email	aboutme	age
▶	oppp		San Francisco, ...	United States	test@test.com	iiiiiiiiii	10
	bernardo22	male	storojinet	Ukraine	test@test.com	hi this is me	10
	test22	male	www	United Arab Em...	test@test	hi please go to ...	47
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

We will only offer here a very simple registration mechanism.

Profile mechanism

We add a mechanism that allows us to view users' profiles. Here we simply add a form that will return JSON data from a query.

```
String u = Request.Form["u"];
String id = Request.Form["id"];

String v;

if (id == null)
    v = SQLUtils.getAvatarFromUsername(u);
else
    v = id;

Chatprofile cp = SQLUtils.getProfileFromUsername(v);

if (cp == null)
    return;

String res = JsonConvert.SerializeObject(new String[] {
cp.aboutme, cp.age, cp.city, cp.country, cp.email, cp.sex, cp.town });

Response.Clear();
Response.Write(res);
Response.Flush();
```

We create a small helper SQL library to retrieve the profile data.

We could have created the profile system as a *REST web service* or integrated into the MVC.

Again, we want to proceed very fast so we use a form.

```
public class SQLUtils
{
    public static readonly char s = '\\';
    public static readonly char sep = ',';
    public static bool ExistsInDb(String id, int n_test)
    {
        try
        {
            String check_queryString = @"SET NOCOUNT OFF;SELECT avatar
FROM quizz1 WHERE EXISTS (SELECT avatar FROM quizz_" + n_test + " WHERE
avatar = '" + id + "')";

            String select_queryString = @"SET NOCOUNT OFF;SELECT avatar
FROM quizz_" + n_test + " WHERE avatar = '" + id + "'";

            using (SqlConnection connection = new
SqlConnection(System.Configuration.ConfigurationManager.
ConnectionStrings["DefaultConnection"].ConnectionString))
            {
                SqlCommand command = new SqlCommand(select_queryString,
connection);

                connection.Open();
                String r = (String)command.ExecuteScalar();

                if (r != id)
                    return false;
                else
                    return true;
            }
        }
        catch (Exception ex)
        {
```

```

    }

    return false;
}

public static string getAvatarFromUsername(string id_)
{
    Dictionary<String, Object[]> d = dumpUsers();

    foreach (KeyValuePair<String, Object[]> K in d)
    {

        if ((String)K.Value[13] == id_)
            return K.Key;

    }

    return null;
}

internal static bool setProfile(Chatprofile cp)
{

    try
    {

        String queryString = @"DELETE FROM [USERS2] WHERE AVATAR =
'" + cp.avatar + @"'";
INSERT INTO [USERS2] ([avatar], [sex], [town], [country], [email],
[aboutme], [age]) VALUES ('" + cp.avatar + '"', '" + cp.sex + '"', '" + cp.town
+ '"', '" + cp.country + '"', '" + cp.email + '"', '" + cp.aboutme + '"', '" +
cp.age + '"');

        using (SqlConnection connection = new
SqlConnection(System.Configuration.ConfigurationManager.
        ConnectionStrings["DefaultConnection"].ConnectionString))
        {
            SqlCommand command = new SqlCommand(queryString,
connection);

            connection.Open();
            command.ExecuteNonQuery();

```

```

    }
}
    catch (Exception ex)
    {

    }

    return false;

}

public static Chatprofile getProfileFromUsername(string id_)
{
    Dictionary<String, Object[]> d = dumpUsers();

    String ID = getAvatarFromUsername(id_);
    if (ID == null)
        return null;

    if (!d.ContainsKey(ID))
        return null;

    Chatprofile cp = new Chatprofile();

    cp.avatar = ID;
    // cp.email = (String)d[ID][3];
    cp.icon = (String)d[ID][2];
    // cp.phone= (String)d[ID][7];

    Dictionary<String, String[]> d2 = dumpUsers2();

    // String ID2 = getAvatarFromUsername(id_);
    //if (ID2 == null)
    //    return cp;

    if (!d2.ContainsKey(ID))
        return cp;

    cp.sex = d2[ID][1];
    cp.town=d2[ID][2];
}

```

```

        cp.country = d2[ID][3];
        cp.email = d2[ID][4];
        cp.aboutme = d2[ID][5];
        cp.age = d2[ID][6];

        return cp;
    }

```

This code reads Avatar's names from the database and returns them as strings. We do the same thing in general with everything that we need from the SQL database. We will use these routines to query data in the db via a form.

We can change many things in the base chat, the CSS, etc... What is important is to allow a chat user to view a profile by clicking another chat user.

We code the following small function in javascript:

```

    jQuery.post("profile2.aspx", {'id':u}, function (data) {
        // alert("success");

        // alert(data);

        var obj = JSON.parse(data);

        var profile = obj[5] + "\nage:" + obj[1] + "\ncity:" + obj[6] +
        "\ncountry:" + obj[3] + "\nAbout me:" + obj[0];

        alert(profile);

    })
    .done(function () {
        // alert("second success");
    })
    .fail(function (xhr, status, error) {
        alert("error while retrieving user profile: " +
        xhr.responseText + " " + error + " " + status);
    })
    .always(function () {

```

```

    //    alert("test submitted");
    });

}

```

We can simply use the `getdevice` function since we do not need it here. We change it in the *ChatHub.js* file:

```

private string IdentityName
{
    get { return Context.User.Identity.Name; }
}

private string GetDevice()
{
    return "javascript:displayprofile2('" + IdentityName + "')"; ;

    //    return "Web";
}

```

It will be displayed with the chat user icon.

```

chatHub.client.addUser = function (user) {

    //    user.Device = "javascript:displayprofile2('" +
user.DisplayName+"')" ;

    //    alert(user.Device);

    model.userAdded(new ChatUser(user.Username,
        user.DisplayName,
        user.Avatar,
        user.CurrentRoom,
        user.Device));

};

```

Here are the windows allowing a user to see/modify their profile:



Avatar:bernardo

Genre:

About me:

Age:

City:

Country:

Email:

Phone:

Edit profile

Email:

Genre:

male

female

Age:

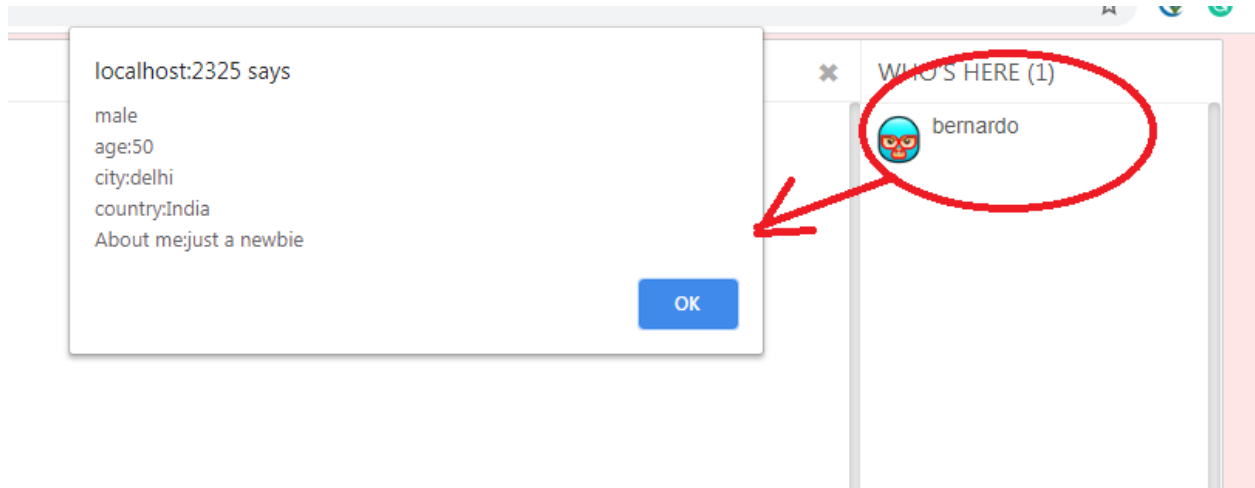
City:

Country:

Phone:

About me:

And this is what happens when you click on the profile of a chat user:



Here we use the `alert()` function, we should use a smoother way to display inline information like a jQuery pop-up for instance.

Quizzes

This is a really important part of the project. We want to build psychological quizzes. The idea is to have the chat user answering several questions and then the results are stored in the database. Later on, some matching coefficients can be computed by anyone on the chat to find the best 'matching souls' based on how many identical answers there are.

Designing the Quizzes

Designing Quizzes isn't always easy. Here we try to avoid the cliches with the 'sentimental' matching. After all, we do not want specifically to develop an online dating website!

We will try to develop original psychological quizzes that can reveal specific traits of the personality of the chat users!

As a start, we will use Bootstrap for displaying online and smooth inline windows.

At the start, we simply present a set of images and we ask the user to click which one he/she likes or hates the most.

```
<table id="quizz1_1" hidden>
  <tr>
    <td id="banana" onclick="quizz(this)">
      
```

```

        </td>
        <td onclick="quizz(this)"></td>
        <td onclick="quizz(this)"></td>
    </tr>
    <tr>
        <td onclick="quizz(this)">
            
        </td>
        <td onclick="quizz(this)"></td>
        <td onclick="quizz(this)"></td>
    </tr>
    <tr>
        <td onclick="quizz(this)">
            
        </td>
        <td onclick="quizz(this)"></td>
        <td onclick="quizz(this)"></td>
    </tr>
</table>

```

We code this as a big HTML sequence of tables that will be successively hidden/revealed with the progress of the quiz.

We need to take care that the user can close the windows and automatically come back where he/she left the quiz.

```
function quizz2(a) {
```

```
    if (quiz2 > 1)
        return;
    var s = a.src;

    if (s == null) {
        a.style.border = "5px solid #ff0000";
        quiz2++;

        return;
    }

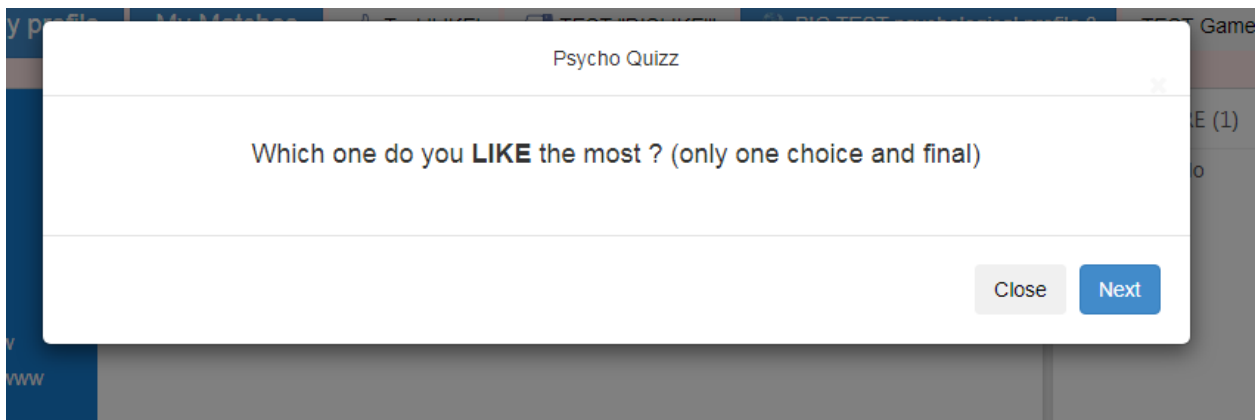
    var tbl = s.split('/');
    var n = tbl.length;

    var nm = tbl[n - 1];

    var answer = nm.replace('.jpg', '');
    quiz2++;

    test_results2[i_test2 - 1] = answer;
}
```

We upload via ajax each test as soon as it is done so, if there are like 10 groups of images we will progressively store the partial results of the tests 10 times.

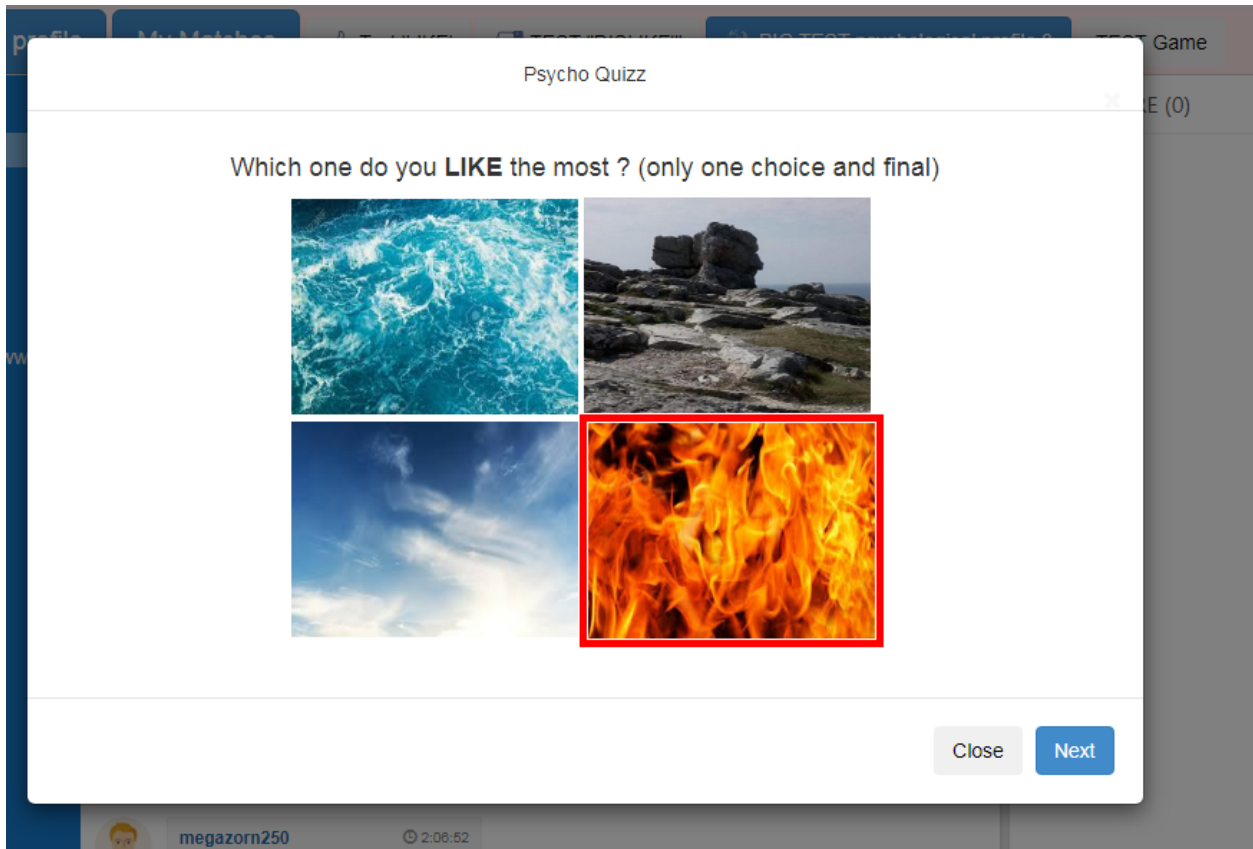


Which one do you **LIKE** the most ? (only one choice and final)



Close

Next



There are about 30 such tests... it is only when the current test is answered that the user can move to the next test.

Next, we create a more complicated psychological quiz. This will be using forms and ASP.NET Ajax, with a timer and an ASP.NET update panel.



Psycho Quizz 3

Question #1

Are we alone in the universe?

What will you answer?

- yes
- no and they're among us
- no but we will never know
- Does it matter?
- No, I know I'm one of them

next



megazorn250

2:08:52

Each time the server computes the partial results for matching.

As you can see from the design view, we have a script manager. This is because we use the ASP.NET Ajax features which allow you to code as always, using codebehind, but *without* a complete page reload.

```

<form id="form1" runat="server">
  <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>

    <div style="background-color:bisque;width:800px" class="modal-content"
  >

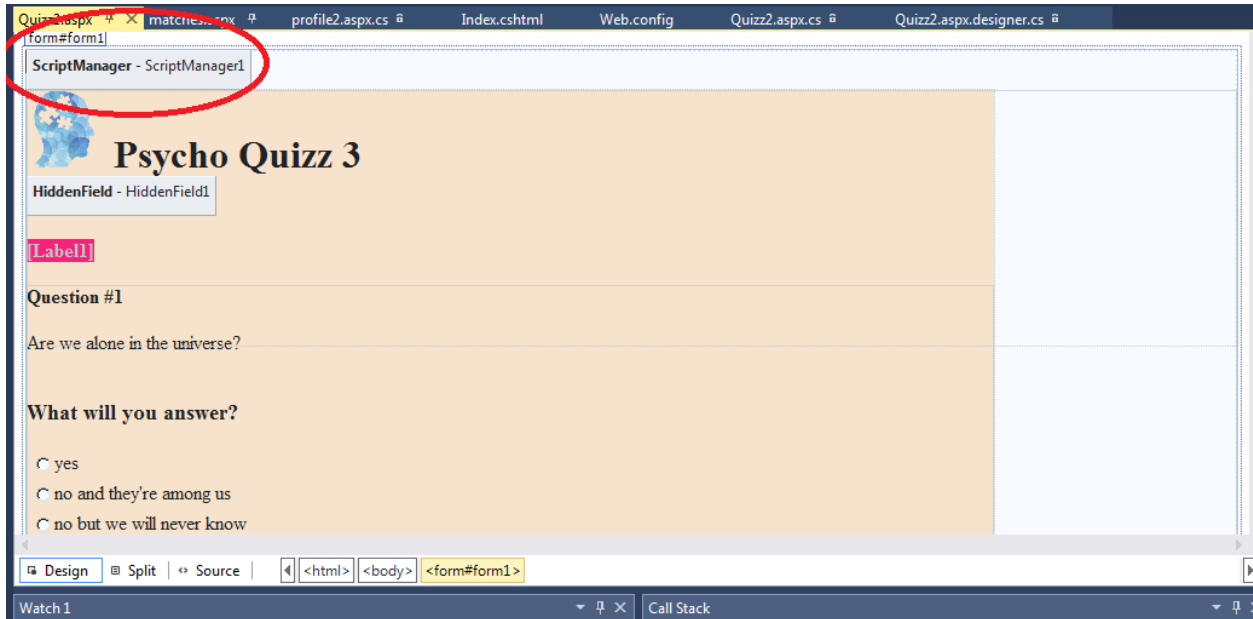
    <div class="modal-header">

```

```
<h1 class="modal-title" id="test34">

    Psycho Quizz 3<asp:HiddenField ID="HiddenField1"
runat="server" />

</h1>
```



The codehind code is a sequence of panels that get displayed and hidden with the progress of the text


```
Quizz2.aspx matches.aspx profile2.aspx.cs Index.cshtml Web.config Quizz2.aspx.cs
352
353 <asp:panel id="Panel13" style="display:none" runat="server">
354
355     <h4>Question #13
356     </h4>
357
358     <div>
359         
360     <br /><br />
361     <b>What does that picture evoke for you?
362 </b>
363
364     </div>
365     <asp:RadioButtonList id="RadioButtonList13" runat="server">
366     <asp:ListItem>Fear, anger, terror, hate</asp:ListItem>
367     <asp:ListItem>Pain</asp:ListItem>
368     <asp:ListItem>Pity, sadness</asp:ListItem>
369     <asp:ListItem>Fun</asp:ListItem>
370     <asp:ListItem>Confidence, joy, love</asp:ListItem>
371     <asp:ListItem>Nothing</asp:ListItem>
372     </asp:RadioButtonList>
373     <br />
374     </asp:panel>
375
376 <asp:panel id="Panel14" style="display:none" runat="server">
377
378     <h4>Question #14
```

We choose around 30 “tricky” psychological questions which are not the average questions found in dating websites.

Matching

Once the quizzes have been performed, users need to be able to search for a matching soul. We will therefore compare the identical amount of answers with other users.

There is nothing very complicated there but it needs to get done!

Again, we will code that into a form that will return the matching results.

```
public static Dictionary<int, string[]> getMatches2(ref int p_progress,
string id, int n_test, int n_tests, string id_target = null)
```

```

    {
        Dictionary<int, string[]> dic_ret = new Dictionary<int,
string[]>();

        Dictionary<String, String[]> target =
SQLUtils.dumpQuizz(n_test, n_tests);

        Dictionary<String, String[]> dic_ = target;

        if (id_target != null)
        {
            if (!dic_.ContainsKey(id_target))
            {

                return dic_ret;

            }

            // KeyValuePair<String, String[]> X = new
KeyValuePair<String, String[]>(id_target, dic_[id_target]);

            dic_ = new Dictionary<String, String[]>();
            dic_.Add(id_target, dic_[id_target]);

        }

        //ref record

        if (!dic_.ContainsKey(id))
            return null;

        String[] rec = dic_[id];

        //remove caller
        dic_.Remove(id);

        foreach (System.Collections.Generic.KeyValuePair<String,
String[]> K in dic_)
        {
            int c = 0;

```

```

    for (int i = 0; i < n_tests; i++)
    {
        if (K.Value[i + 1] == rec[i + 1])
            c++;
    }

    int percent = (int)((100 * c) / n_tests);

    if (!dic_ret.ContainsKey(percent))
        dic_ret.Add(percent, new String[] { K.Key });
    else
    {
        List<String> l0 = dic_ret[percent].ToList();

        l0.Add(K.Key);

        dic_ret[percent] = l0.ToArray();

        l0.Clear();
    }

}

//sort by decreasing values

return dic_ret;
}

```

```

private string printMatches(Dictionary<int, string[]> dic)
{
    Dictionary<String, Object[]> dic_users= SQLUtils.dumpUsers();

    var myList = dic.ToList();
    String res = "";

```

```

        myList.Sort((pair1, pair2) => pair1.Key.CompareTo(pair2.Key));

        myList.Reverse();

        foreach (System.Collections.Generic.KeyValuePair<int, String[]>
K in myList)
        {
            if ((K.Value != null) && (K.Value.Length > 0))
            {
                //    String[] Value2 = K.Value.Reverse().ToArray();

                res = res + "<b>" + K.Key + "% matching </b>(" +
K.Value.Length + ")";

                for (int i = 0; i < Math.Min(K.Value.Length, 5); i++)
                {

                    String avatar = K.Value[i];

                    String img = dic_users.ContainsKey(avatar) ?
(String)dic_users[avatar][2] : null;

                    res = res + "<div>" + avatar;

                    if (img!=null)
                    res=res+ "<img src=\"" +img+"\" width=\"64px\"
height=\"64px\">";
                    else
                    res = res + "<img src=\"Content/icons/anyone.png\"
width=\"64px\" height=\"64px\">";

                    Chatprofile cp =
SQLUtils.getProfileFromUsername(avatar);

                    if (cp != null)
                    {
                        if (cp.sex == null)
                            cp.sex = "unknown";
                    }
                }
            }
        }
    }
}

```

```
        res = res + cp.sex + "," + cp.age + " yrs," +
cp.town + "," + cp.country;

        res = res + "<a
href=\"javascript:displayprofile3('" + avatar + "')\">...</a>";

    }
    res = res + "</div>";

}

if (K.Value.Length > 20)
    res = res + "<div>...</div>";

}

}
return res;
}
```

Matching

Matching will be done with :

- All Chatters
- A specific chatter

newChatuser ▾

0 matches done

Compute Match results

Matching

Computing matches... may take a while please be patient



Matching will be done with

- All Chatters
- A specific chatter

newChatuser ▾

0 matches done

Compute Match results

Matching



Matching results for Like-test

100% matching (1)



90% matching (2)

megazorn250



unknown, yrs,,...

megazorn249



unknown, yrs,,...

81% matching (1)

test22



male,47 yrs,www,United Arab Emirates...

63% matching (1)


oppo




,10 yrs,San Francisco, CA,United States...


54% matching (1)

75% matching (1)


megazorn250  unknown, yrs,....


50% matching (1)


megazorn249  unknown, yrs,....

 Matching results for BIG Psycho-test


42% matching (3)

test22  male,47 yrs,www,United Arab Emirates...

testuser_4921527 

testuser_943624 

38% matching (7)

testuser_8174790 

Finally, the chat can be used to find matching souls and chat with them

My profile

My Matches









👍 Test 'LIKE'

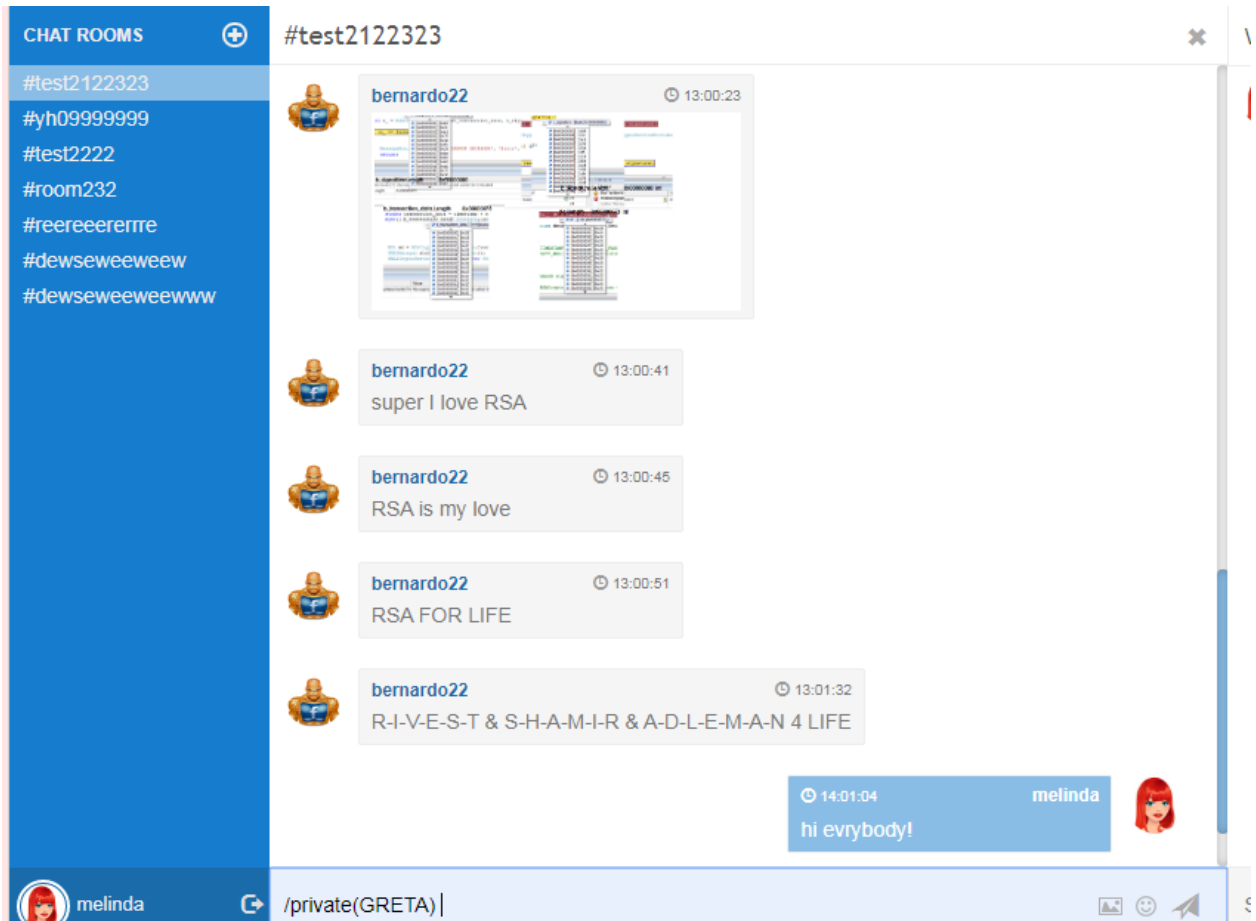
👎 TEST "DISLIKE"

🧠 BIG TEST psychological profile 3

- CHAT ROOMS +
- #test2122323
 - #yh09999999
 - #test2222
 - #room232
 - #reereeererre
 - #dewseweewew
 - #dewseweewewww

#yh09999999 ✕ WHO'

-  megazorn250 9:40:51
hi
-  oppp 14:46:02
-  oppp 14:46:02
🤔
-  oppp 14:46:16
-  oppp 14:46:16
😬😡
-  bernardo22 12:59:46
-  bernardo22 12:59:46
1 🤪
-  test22 5:01:38
hi



[We provide the source code for the chat](#), use it as you wish for any purpose you may want. We give no guarantee whatsoever.

Conclusion

We saw how to use an existing chat based on Signal-R and how to tune it to transform it into a system where users can find 'matching souls' using quizzes.

We saw how ASP.NET and AJAX ASP.NET allowed us to create rich web applications that run smoothly into the browser. We hope you enjoyed the tutorial and that you have found there some useful information.