

MANGO CHAT

# Mango Chat Developer Guide 1.0

---

Startup guide for the developers

**Mango Chat Support Team**

**2/1/2011**

This document is a startup guide for the developers who want to integrate the Mango Chat .Net Ajax based chat control to their websites.

## **Contents**

<a href="#">Contents.....</a>	<a href="#">2</a>
<a href="#">Overview.....</a>	<a href="#">3</a>
<a href="#">Getting Started.....</a>	<a href="#">3</a>
<a href="#">Names and Avatars.....</a>	<a href="#">7</a>
<a href="#">Contact Lists.....</a>	<a href="#">8</a>
<a href="#">Data Providers.....</a>	<a href="#">10</a>
<a href="#">Themes.....</a>	<a href="#">11</a>

# Overview

Mango Chat is a full-featured ASP.NET chat component which allows website owners to add chat functionality to their website. It includes features such as high load support, font/color/ customization, emoticons, and many more! The best thing about mango chat is that its a web control and is as easy as a simple drag and drop for the website owners.

## Getting Started

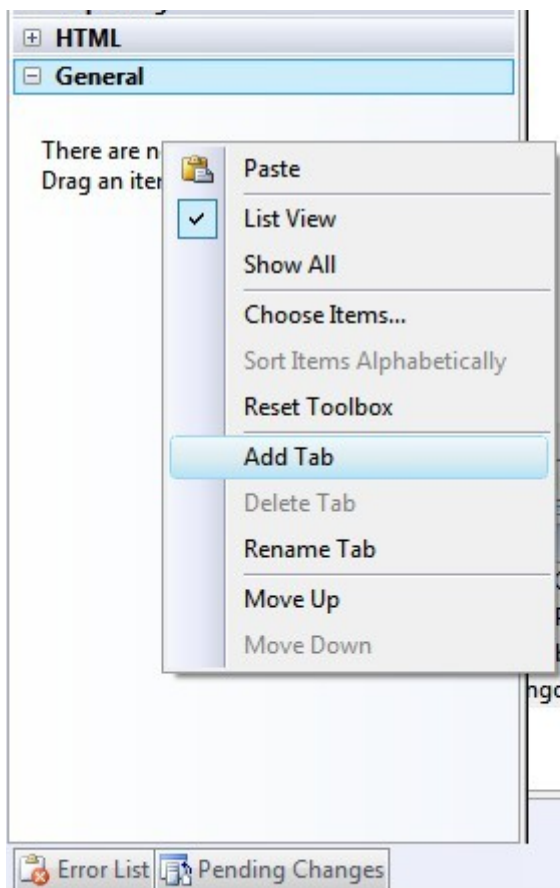
Integrating basic chat functionality to your website is super easy with MangoChat. Just follow along.

Open your webpage where u want the chat to be available in 'Design Mode'

### **\*\*Tip\*\***

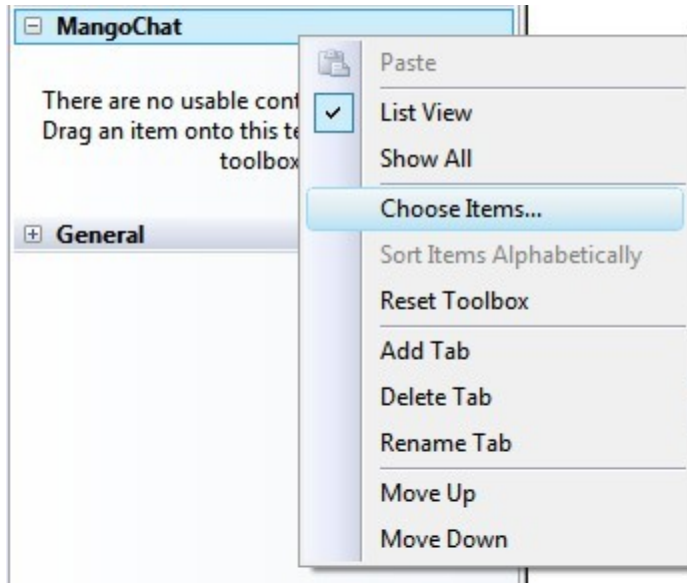
You can use a masterpage if you need the chat control to be available on multiple pages.

Right Click on the the ASP.NET Toolbox and select 'Add Tab'

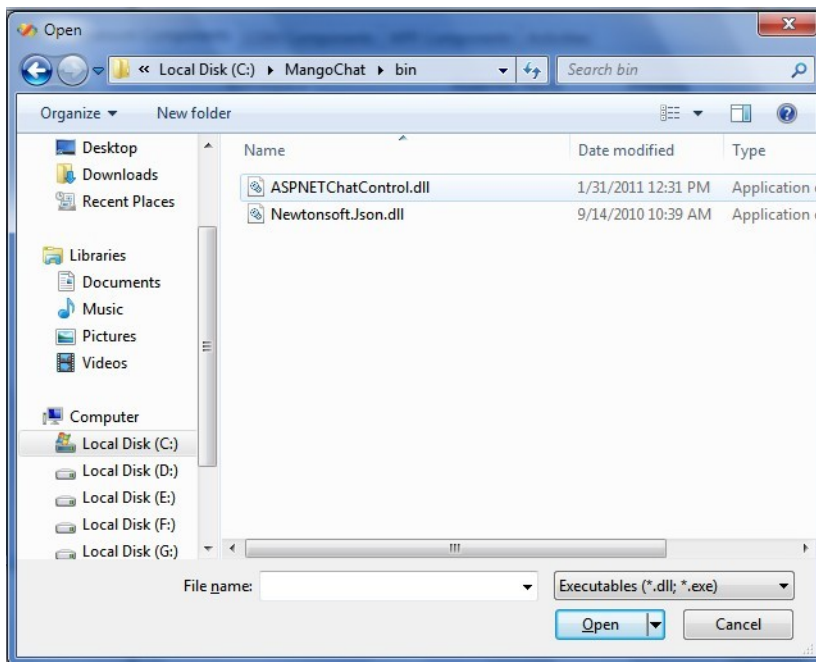


Name it 'MangoChat'

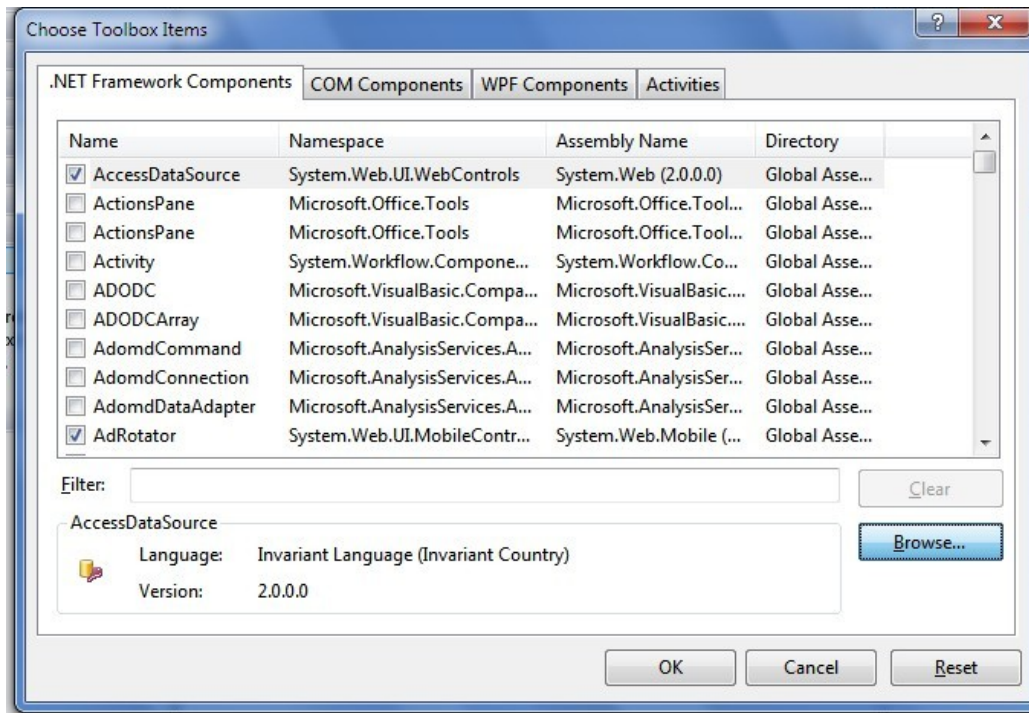
Right Click the 'MangoChat' tab and select 'Choose Items'



Click 'Browse' on the '.NET Framework Components' Tab

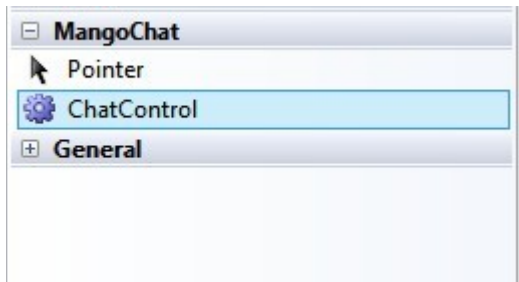


Select 'ASPNETChatControl.dll' from the 'Open File' dialog



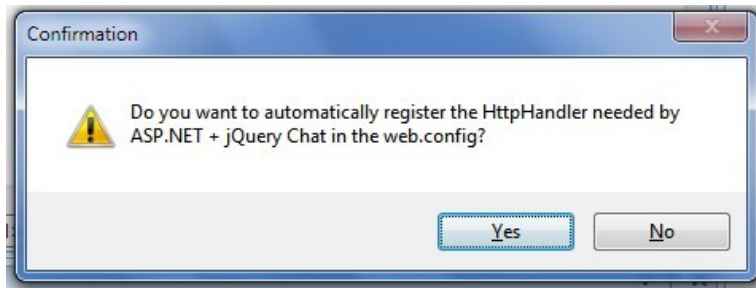
Click OK to close the 'Choose Toolbox Items' dialog

You should now have the 'ChatControl' webcontrol displayed in the 'MangoChat' tab.

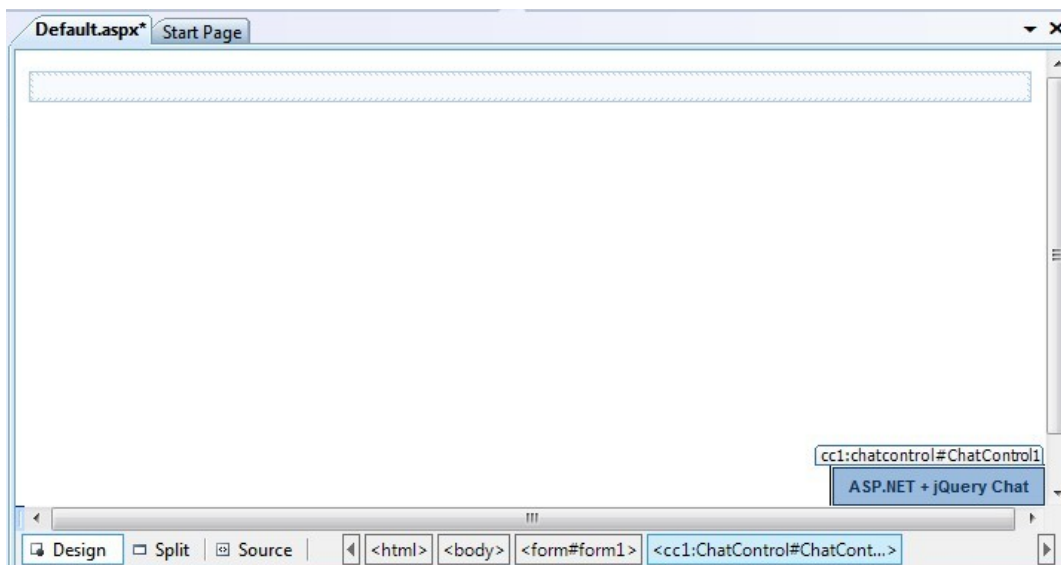


Drag the control onto the page

You'll get a confirmation box asking if you wish to add the necessary configuration setting to the web.config automatically. Click 'Yes' to continue



You should be able to see a placeholder for the chat control on your page on the lower right hand corner of the page



Go to the code behind for this page and import the 'ASPNETChatControl' namespace

Type the following code inside the Page\_Load event handler to start the chatting session on this page:

```
ChatControl.StartSession();
```

This is how it should look like.

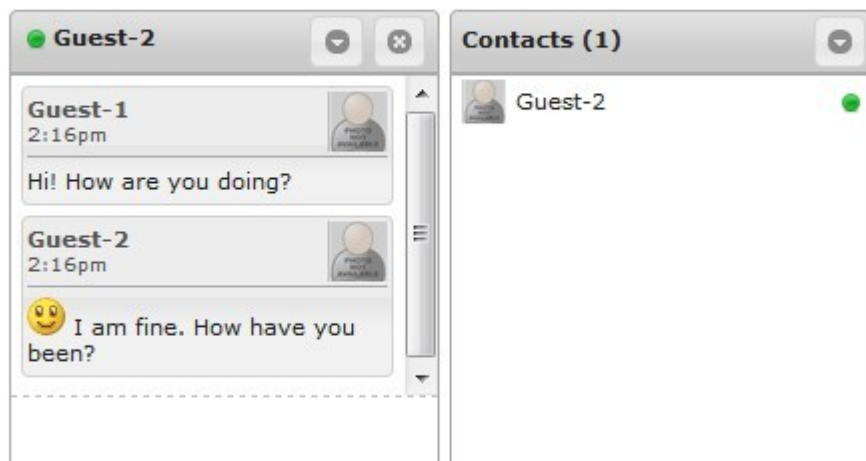
```

using ASPNETChatControl;

namespace SampleChatProject
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            ChatControl.StartSession();
        }
    }
}

```

Run the application in two different browsers (e.g. Chrome and FireFox) to try it out.



This is all you need to get the control to work on your website. With a little more work you can customize both the visual appearance and the behavior of the control to make it more integrated into your website. You'll be able to do things like:

- Assign custom names to users instead of the default Guest-x naming scheme.
- Assign custom avatars to the users
- Create contact lists for users instead of showing everyone to everyone
- Create custom data store providers for the control (NCache, MySQL, etc.)
- Change the default theme of the chat UI to match the look and feel of your website

We'll look at how to do these things in the next few sections.

## Names and Avatars

Assigning a custom name or avatar image to a user is as easy as using a different overload of the StartSession method that we saw in 'Getting Started'.

```

using ASPNETChatControl;

namespace SampleChatProject
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            ChatControl.StartSession(
                ▲3 of 4 ▼ void ChatControl.StartSession (string username, string chatPhotoName)
            );
        }
    }
}

```

---

So instead of using the following:

```
void ChatControl.StartSession();
```

use one of the following overloads to specify the username and/or the avatar image for the user

```

void ChatControl.StartSession(string username);
void ChatControl.StartSession(string username, string chatPhotoName);

```

The 'username' takes a string and sets it as the current users name.

The chatPhotoName takes a url string pointing to the desired avatar image for the current user.

You can pull this information from anywhere you want for e.g. from the user's session variables or a database.

## Contact Lists

The default behavior of the chat control is to display all users to all users in their contact list. While this might work for a demo or a really simple website, we think in most cases you'll need to customize this behavior. To do it just follow these steps:

First of all, we need to uniquely identify all the chat users and to do that we need to assign unique ids to them. While the chat control does this itself out of the box but the ids that it assigns to the users will be meaningless to your application. So first of all we have to assign custom user ids to the users. To do this just use the following overload of the 'StartSession' method to provide a custom userId in addition to providing the username and the avatar.

```
public static void StartSession(string userId, string username, string chatPhotoName);
```

OK, now lets move onto creating our custom contact list building logic.

Create a new class. Name it anything you like

Import the 'ASPNETChatControl.Extensibility' namespace.

Inherit the class from the 'ContactListProvider' base class.

Override the 'GetContactsByUserId' method.

```
using ASPNETChatControl.Extensibility;
using System.Collections.Generic;

namespace SampleChatProject.Extensions
{
    public class MyContactListProvider:ContactListProvider
    {
        public override Dictionary<string, string>
            GetContactsByUserId(string userId)
        {
            return base.GetContactsByUserId(userId);
        }
    }
}
```

This method will be called whenever the chat control needs to provide a contact list for a user. It provides you the 'userId' for which it needs the contact list and in return you provide it a generic Dictionary object of type 'Dictionary<string, string>' where the key is the 'userId' and the value is the 'username' of the contact.

As you can see that by default it calls its base class version and simply returns the results. In order to customize the behavior you need to implement your own logic inside this method.

**\*\*Tip\*\***

If you need to access all currently available user sessions inside this method, you can call the following method.

```
protected List<UserChatSession> GetAllSessions();
```

This provides you a list of 'UserChatSession' objects. UserChatSession is a simple class that contains properties for a user's session such as the user's username, userid, avatar.

Now we need to tell MangoChat to use your ContactListProvider instead of the default one. To do this

Add 'Global.asax' to the root of your website (If its not already there).

Import the 'ASPNETChatControl' namespace.

In the 'Application\_Start' handler set your custom contact list provider as the one to be used as follows

```
ChatControl.ContactListProvider = new MyContactListProvider();
```

Done.

## Data Providers

MangoChat provides you with two chat data storage options out of the box.

In-Memory (Default)  
Microsoft SQL Server

The In-Memory option offers the best performance but isn't horizontally scalable. The SQL Server option offers scalability with some performance overhead.

To use the in-memory data storage provide you don't need to do anything as its the default data provider.

If you want to use the SQL Server data storage provider, follow these steps.

In your web.config file add an appSetting with the following key

```
'ASPNETChatControl.DAL.SqlServer.DSN'
```

and specify your SQL Server connection string as the value. For example:

```
<appSettings>
  <add key="ASPNETChatControl.DAL.SqlServer.DSN" value="Data
    Source=.\SQLEXPRESS;Initial Catalog=ChatDB;User
    ID=sa;Password=xxx"/>
</appSettings>
```

Next, add the 'Global.asax' file to the root of your application (If you don't have it there already)

Import the 'ASPNETChatControl' and 'ASPNETChatControl.DAL.SqlServer' namespaces in your 'Global.asax.cs' file.

Set the SQL Server data provider in the 'Application\_Start' handler as follows:

```
ChatControl.DataProvider = new SqlDataProvider();
```

Done.

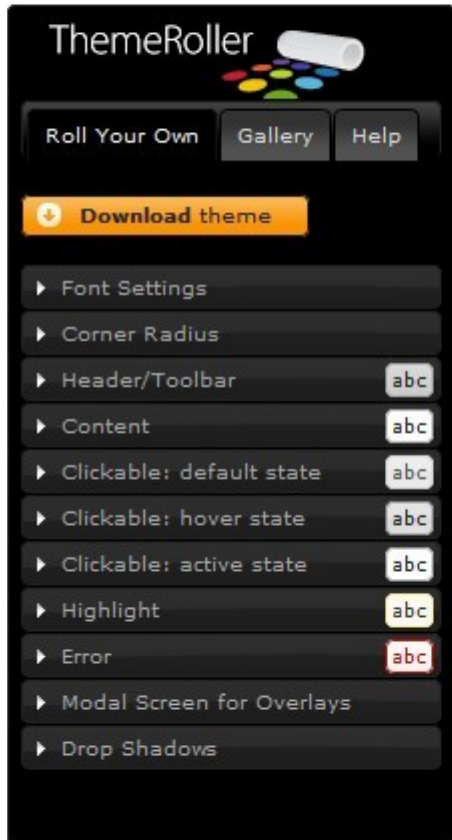
While these should cover most of the applications out there, if you like to use something else you have the option to do that too. In a nutshell you need to write a class, have it implement the 'IChatDataProvider' interface and let MangoChat know that you want to use it.

For more information on writing a custom data provider for mango chat, refer to the 'Custom Data Providers' document.

## Themes

MangoChat uses the jQuery UI CSS Framework to style itself. jQuery is the most used javascript library due to its wide array for plugins and exceptional ease of use. jQuery UI is the official set of UI widgets provider by the jQuery team.

What this means is that if you already use jQuery UI for your UI components, MangoChat can automatically use the theme that you are using for your application. If not you can design a custom theme using the brilliant 'ThemeRoller' tool provided by jQuery UI. You can find it at <http://jqueryui.com/themeroller/>. Once you have designed your theme just press download and you'll get a zip file containing all the required css and image files.



Once you've imported the css files into your pages, you just need to do the following to have MangoChat use it:

Open your page where you put the MangoChat control in 'Design Mode'

Select the MangoChat control. It should be in the bottom right hand corner of the page.

Open the 'Properties' window and set 'IncludeDefaultTheme' property to 'False'

Done.