Tutorial: Real-time chat with SignalR 2 and MVC 5

Article • 02/20/2020 • 6 minutes to read

This tutorial shows how to use ASP.NET SignalR 2 to create a real-time chat application. You add SignalR to an MVC 5 application and create a chat view to send and display messages.

In this tutorial, you:

✓ Set up the project
✓ Run the sample
✓ Examine the code

⚠ **Warning**

This documentation isn't for the latest version of SignalR. Take a look at **ASP.NET Core SignalR**.

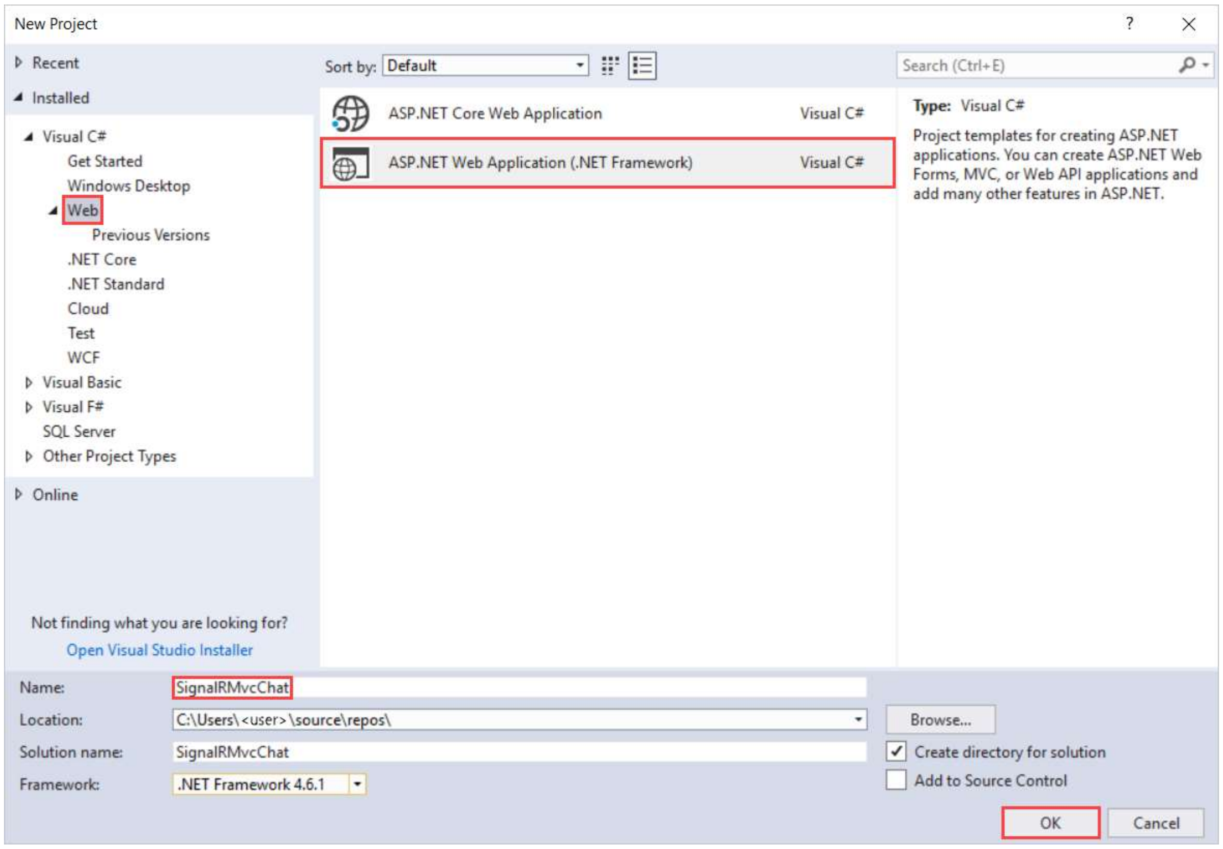
Prerequisites

- Visual Studio 2017 with the **ASP.NET and web development** workload.

Set up the Project

This section shows how to use Visual Studio 2017 and SignalR 2 to create an empty ASP.NET MVC 5 application, add the SignalR library, and create the chat application.

1. In Visual Studio, create a C# ASP.NET application that targets .NET Framework 4.5, name it SignalRChat, and click OK.


New Project

Recent
Installed
Visual C#
Get Started
Windows Desktop
Web
Previous Versions
.NET Core
.NET Standard
Cloud
Test
WCF
Visual Basic
Visual F#
SQL Server
Other Project Types
Online

Not finding what you are looking for?
Open Visual Studio Installer

Sort by: Default

Search (Ctrl+E)

ASP.NET Core Web Application — Visual C#
ASP.NET Web Application (.NET Framework) — Visual C#

Type: Visual C#
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

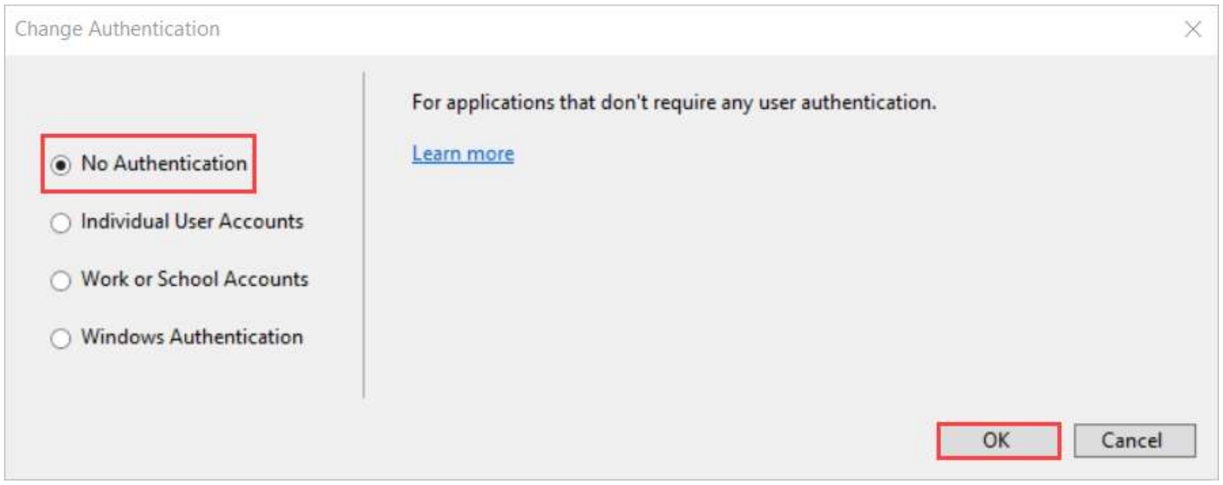
Name: SignalRMvcChat
Location: C:\Users\<user>\source\repos\
Solution name: SignalRMvcChat
Framework: .NET Framework 4.6.1

Browse...
[✓] Create directory for solution
[ ] Add to Source Control

OK Cancel

2. In **New ASP.NET Web Application - SignalRMvcChat**, select **MVC** and then select **Change Authentication**.

3. In **Change Authentication**, select **No Authentication** and click **OK**.


Change Authentication

● No Authentication
○ Individual User Accounts
○ Work or School Accounts
○ Windows Authentication

For applications that don't require any user authentication.

Learn more

OK Cancel

4. In **New ASP.NET Web Application - SignalRMvcChat**, select **OK**.

5. In **Solution Explorer**, right-click the project and select **Add** > **New Item**.

6. In **Add New Item - SignalRChat**, select **Installed** > **Visual C#** > **Web** > **SignalR** and then select **SignalR Hub Class (v2)**.

7. Name the class *ChatHub* and add it to the project.

   This step creates the *ChatHub.cs* class file and adds a set of script files and assembly references that support SignalR to the project.

8. Replace the code in the new *ChatHub.cs* class file with this code:

   C#

   ```
   using System;
   using System.Web;
   using Microsoft.AspNet.SignalR;
   namespace SignalRChat
   {
       public class ChatHub : Hub
       {
           public void Send(string name, string message)
           {
               // Call the addNewMessageToPage method to update clients.
               Clients.All.addNewMessageToPage(name, message);
           }
       }
   }
   ```

9. In **Solution Explorer**, right-click the project and select **Add** > **Class**.

10. Name the new class *Startup* and add it to the project.

11. Replace the code in the *Startup.cs* class file with this code:

   C#

   ```
   using Owin;
   using Microsoft.Owin;
   [assembly: OwinStartup(typeof(SignalRChat.Startup))]
   namespace SignalRChat
   {
       public class Startup
       {
           public void Configuration(IAppBuilder app)
           {
               // Any connection or hub wire up and configuration should go
   here
               app.MapSignalR();
           }
       }
   }
   ```

12. In **Solution Explorer**, select **Controllers** > **HomeController.cs**.

13. Add this method to the *HomeController.cs*.

    C#

    ```
    public ActionResult Chat()
    {
        return View();
    }
    ```

    This method returns the **Chat** view that you create in a later step.

14. In **Solution Explorer**, right-click **Views** > **Home**, and select **Add** > **View**.

15. In **Add View**, name the new view **Chat** and select **Add**.

16. Replace the contents of **Chat.cshtml** with this code:

    CSHTML

    ```
    @{
        ViewBag.Title = "Chat";
    }
    <h2>Chat</h2>
    <div class="container">
        <input type="text" id="message" />
        <input type="button" id="sendmessage" value="Send" />
        <input type="hidden" id="displayname" />
        <ul id="discussion">
        </ul>
    </div>
    @section scripts {
        <!--Script references. -->
        <!--The jQuery library is required and is referenced by default in
    _Layout.cshtml. -->
        <!--Reference the SignalR library. -->
        <script src="~/Scripts/jquery.signalR-2.1.0.min.js"></script>
        <!--Reference the autogenerated SignalR hub script. -->
        <script src="~/signalr/hubs"></script>
        <!--SignalR script to update the chat page and send messages.-->
        <script>
            $(function () {
                // Reference the auto-generated proxy for the hub.
                var chat = $.connection.chatHub;
                // Create a function that the hub can call back to display
    messages.
                chat.client.addNewMessageToPage = function (name, message) {
    ```

```
                    // Add the message to the page.
                    $('#discussion').append('<li><strong>' + htmlEncode(name)
                        + '</strong>: ' + htmlEncode(message) + '</li>');
                };
                // Get the user name and store it to prepend to messages.
                $('#displayname').val(prompt('Enter your name:', ''));
                // Set initial focus to message input box.
                $('#message').focus();
                // Start the connection.
                $.connection.hub.start().done(function () {
                    $('#sendmessage').click(function () {
                        // Call the Send method on the hub.
                        chat.server.send($('#displayname').val(),
$('#message').val());
                        // Clear text box and reset focus for next comment.
                        $('#message').val('').focus();
                    });
                });
            });
            // This optional function html-encodes messages for display in the
page.
            function htmlEncode(value) {
                var encodedValue = $('<div />').text(value).html();
                return encodedValue;
            }
        </script>
}
```

17. In **Solution Explorer**, expand **Scripts**.

    Script libraries for jQuery and SignalR are visible in the project.

    Important

    The package manager may have installed a later version of the SignalR scripts.

18. Check that the script references in the code block correspond to the versions of the script files in the project.

    Script references from the original code block:

CSHTML

```
<!--Script references. -->
<!--The jQuery library is required and is referenced by default in
_Layout.cshtml. -->
```

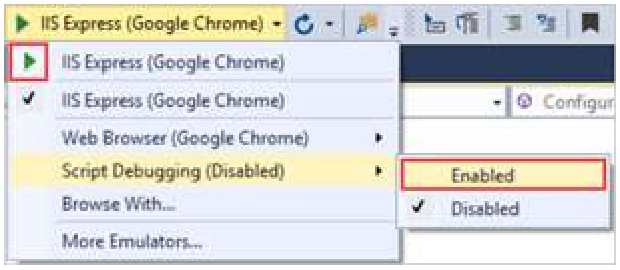
```
<!--Reference the SignalR library. -->
<script src="~/Scripts/jquery.signalR-2.1.0.min.js"></script>
```

19. If they don't match, update the *.cshtml* file.

20. From the menu bar, select **File** > **Save All**.

Run the Sample

1. In the toolbar, turn on **Script Debugging** and then select the play button to run the sample in Debug mode.


IIS Express (Google Chrome)
IIS Express (Google Chrome)
IIS Express (Google Chrome)
Web Browser (Google Chrome)
Script Debugging (Disabled)
Browse With...
More Emulators...
Enabled
Disabled
Configur

2. When the browser opens, enter a name for your chat identity.

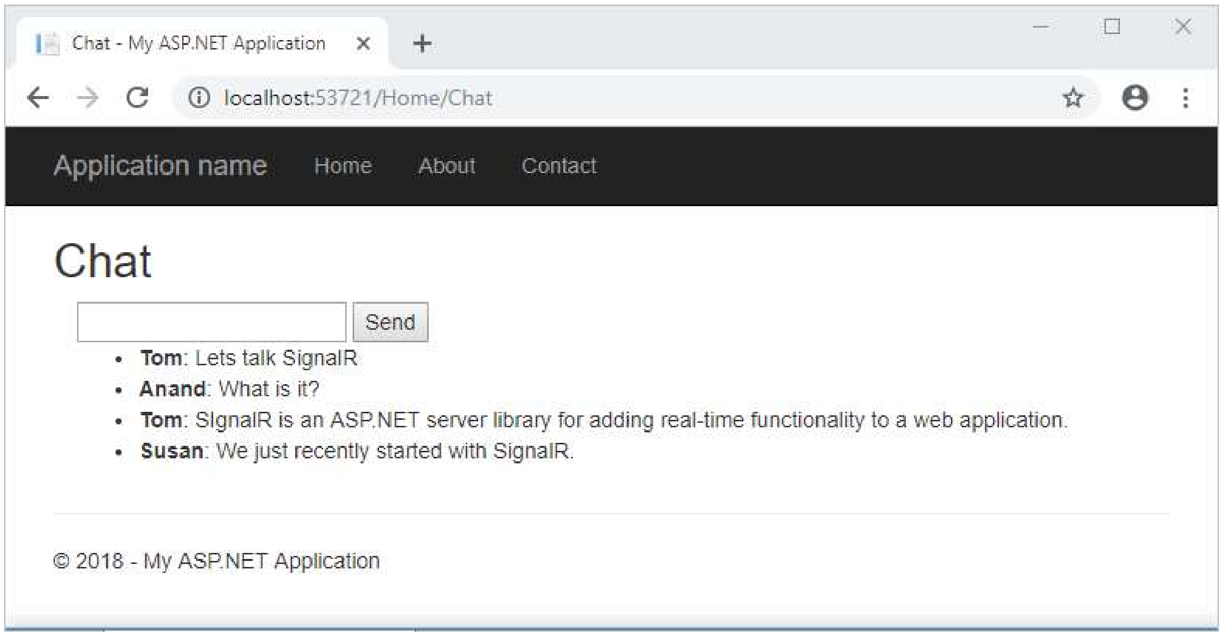
3. Copy the URL from the browser, open two other browsers, and paste the URLs into the address bars.

4. In each browser, enter a unique name.

5. Now, add a comment and select **Send**. Repeat that in the other browsers. The comments appear in real time.

> Note
>
> This simple chat application does not maintain the discussion context on the server. The hub broadcasts comments to all current users. Users who join the chat later will see messages added from the time they join.

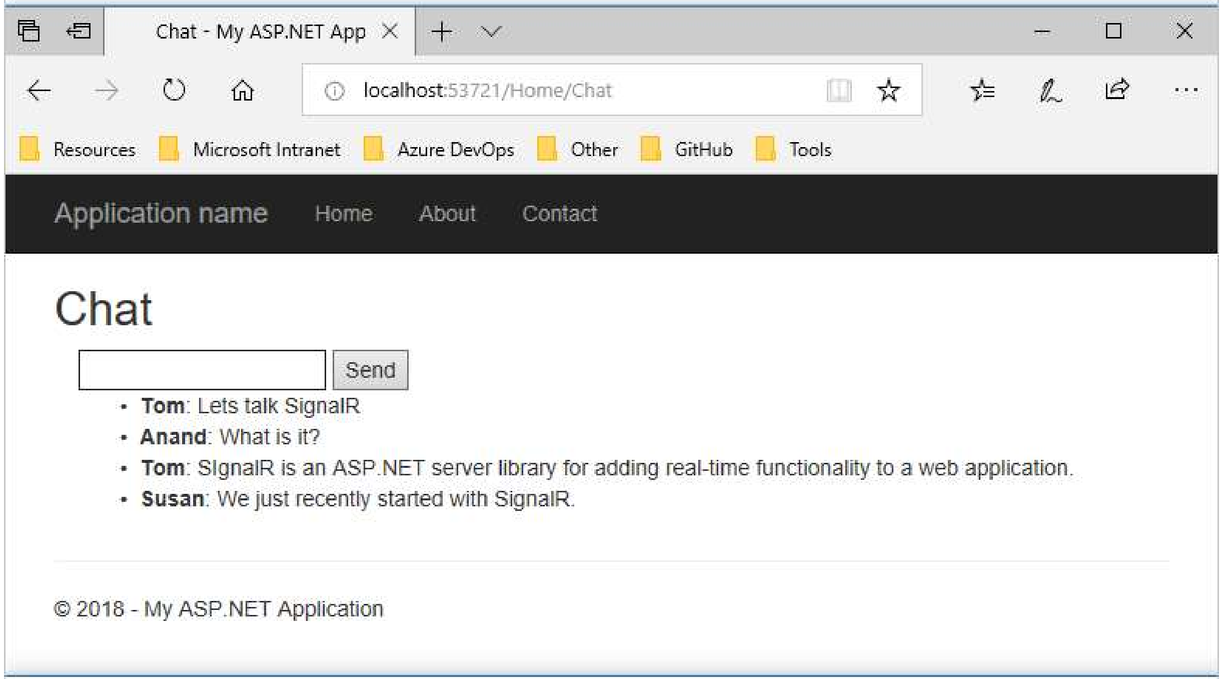
See how the chat application runs in three different browsers. When Tom, Anand, and Susan send messages, all browsers update in real time:



Chat

- **Tom**: Lets talk SignalR
- **Anand**: What is it?
- **Tom**: SIgnalR is an ASP.NET server library for adding real-time functionality to a web application.
- **Susan**: We just recently started with SignalR.

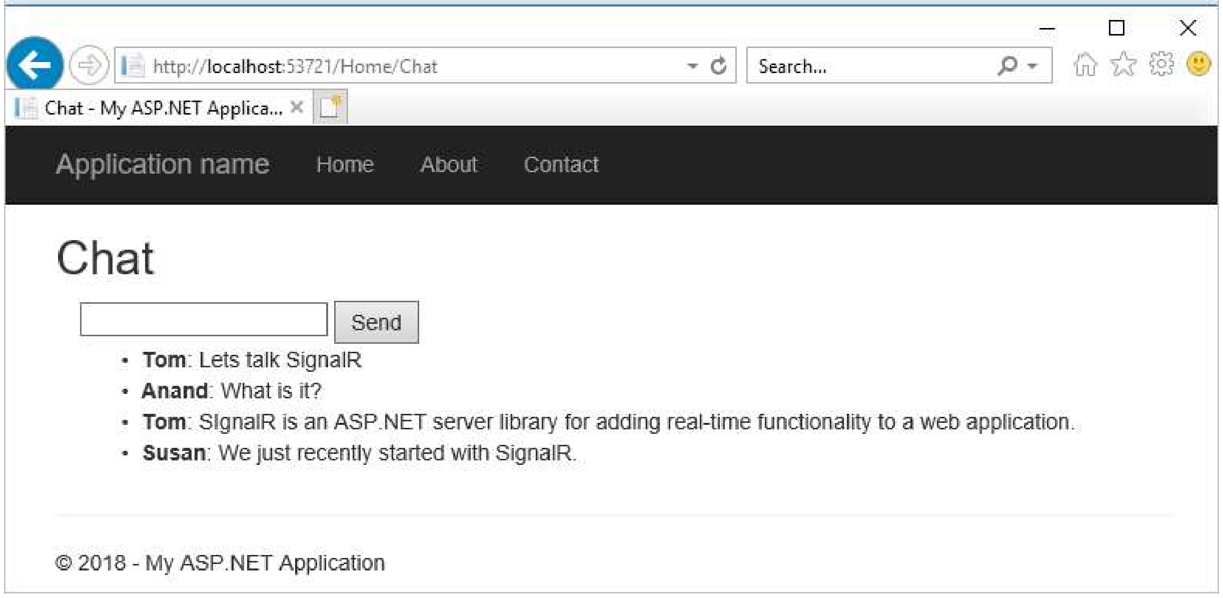
© 2018 - My ASP.NET Application



Chat

- **Tom**: Lets talk SignalR
- **Anand**: What is it?
- **Tom**: SIgnalR is an ASP.NET server library for adding real-time functionality to a web application.
- **Susan**: We just recently started with SignalR.

© 2018 - My ASP.NET Application

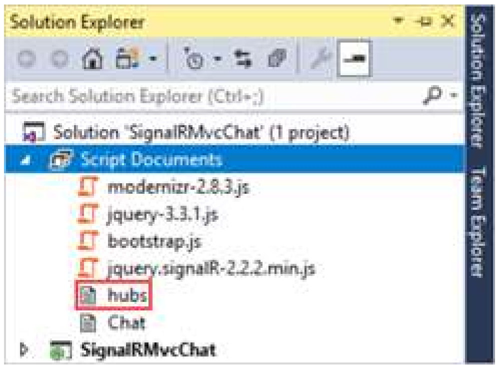

http://localhost:53721/Home/Chat
Search...
Chat - My ASP.NET Applica...
Application name
Home
About
Contact
Send

Chat

- **Tom**: Lets talk SignalR
- **Anand**: What is it?
- **Tom**: SIgnalR is an ASP.NET server library for adding real-time functionality to a web application.
- **Susan**: We just recently started with SignalR.

© 2018 - My ASP.NET Application

6. In **Solution Explorer**, inspect the **Script Documents** node for the running application. There's a script file named *hubs* that the SignalR library generates at runtime. This file manages the communication between jQuery script and server-side code.



Examine the Code

The SignalR chat application demonstrates two basic SignalR development tasks. It shows you how to create a hub. The server uses that hub as the main coordination object. The hub uses the SignalR jQuery library to send and receive messages.

SignalR Hubs in the ChatHub.cs

In the code sample, the `ChatHub` class derives from the `Microsoft.AspNet.SignalR.Hub` class. Deriving from the `Hub` class is a useful way to build a SignalR application. You can create public methods on your hub class and then access those methods by calling them from scripts in a web page.

In the chat code, clients call the `ChatHub.Send` method to send a new message. The hub in turn sends the message to all clients by calling `Clients.All.addNewMessageToPage`.

The `Send` method demonstrates several hub concepts:

- Declare public methods on a hub so that clients can call them.
- Use the `Microsoft.AspNet.SignalR.Hub.Clients` dynamic property to communicate with all clients connected to this hub.
- Call a function on the client (like the `addNewMessageToPage` function) to update clients.

C#

```
public class ChatHub : Hub
{
    public void Send(string name, string message)
    {
        Clients.All.addNewMessageToPage(name, message);
    }
}
```

SignalR and jQuery Chat.cshtml

The *Chat.cshtml* view file in the code sample shows how to use the SignalR jQuery library to communicate with a SignalR hub. The code carries out many important tasks. It creates a reference to the autogenerated proxy for the hub, declares a function that the server can call to push content to clients, and it starts a connection to send messages to the hub.

JavaScript

```
var chat = $.connection.chatHub;
```

Note

In JavaScript, the reference to the server class and its members is in camelCase. The code sample references the C# `ChatHub` class in JavaScript as `chatHub`.

In this code block, you create a callback function in the script.

HTML

```
chat.client.addNewMessageToPage = function (name, message) {
    // Add the message to the page.
    $('#discussion').append('<li><strong>' + htmlEncode(name)
        + '</strong>: ' + htmlEncode(message) + '</li>');
};
```

The hub class on the server calls this function to push content updates to each client. The optional call to the `htmlEncode` function shows a way to HTML encode the message content before displaying it in the page. It's a way to prevent script injection.

This code opens a connection with the hub.

JavaScript

```
$.connection.hub.start().done(function () {
    $('#sendmessage').click(function () {
        // Call the Send method on the hub.
        chat.server.send($('#displayname').val(), $('#message').val());
        // Clear text box and reset focus for next comment.
        $('#message').val('').focus();
    });
});
```

Note

This approach ensures that you establish a connection before the event handler executes.

The code starts the connection and then passes it a function to handle the click event on the **Send** button in the Chat page.

Get the code

Download Completed Project

Additional resources

For more about SignalR, see the following resources:

- SignalR Project
- SignalR GitHub and Samples
- SignalR Wiki

Next steps

In this tutorial, you:

✓ Set up the project
✓ Ran the sample
✓ Examined the code

Advance to the next article to learn how to create a web application that uses ASP.NET SignalR 2 to provide high-frequency messaging functionality.

Web app with high-frequency messaging