

# Introduction to WebSockets with SignalR in .NET Part 1: the basics

MAY 15, 2014   2 COMMENTS ([HTTPS://DOTNETCODR.COM/2014/05/15/INTRODUCTION-TO-WEBSOCKETS-WITH-SIGNALR-IN-NET-PART-1-THE-BASICS/#COMMENTS](https://dotnetcodr.com/2014/05/15/introduction-to-websockets-with-signalr-in-net-part-1-the-basics/#comments)).

## Introduction

The normal communication flow between a browser and a web app on a server is quite limited: the client sends a HTTP request to the server and receives a HTTP response back. If the client needs some new information then it will need to send another request to the server. Normally a web server will not just send out HTTP responses to a client without a HTTP request first to act upon.

This is true even for pages where you view the same type of data which is updated periodically. A typical example is a sports site where you can view the current standing of a game. You as the viewer would probably like to have real time updates so that you can be happy when your team scores – or smash your smart phone against the wall if it's losing.

There are various ways to solve this problem on a web site:

- Put a big “Refresh” button on the page which requests and updated set of objects from the server – very straightforward, but not too convenient for the client
- Put some kind of JavaScript timer function on the web page which periodically requests an update from the server and refreshes the page
- Same as above but use an Ajax call to only refresh the page partially

...and possibly others.

The underlying method with all three solutions is that the browser will need to keep sending the same request to the server over and over again. Imagine that this site is very popular so their server receives tens of thousands of essentially the same request. The company will need to invest in extra infrastructure, such as a web farm, a load balancer, CDN etc. Wouldn't it be easier if the browser could tell the server: “hey, server, I need updates from you on the same set of data periodically, thanks.” And then the communication channel is magically kept open between the client and the server. The server will be able to send updates over that channel automatically without waiting for new requests. This can potentially result in an alleviated server load, a very responsive web UI and significant savings for the company in terms of reduced infrastructure.

## Web Sockets

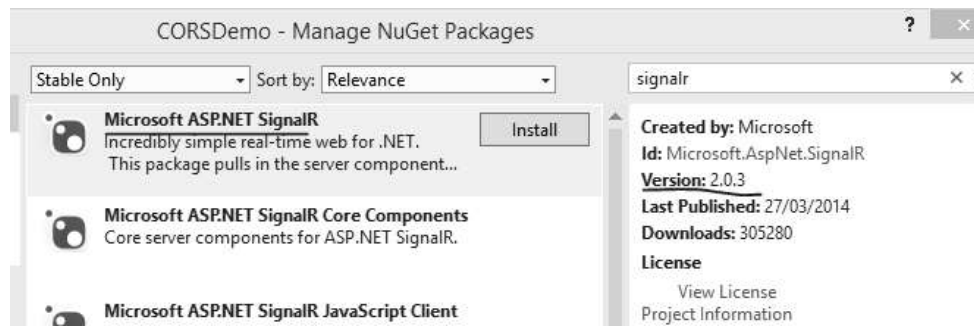
These communication channels through Web Sockets (<http://en.wikipedia.org/wiki/WebSocket>) have been available for some years now, so it's not some kind of a dream. A web socket is a TCP (<https://dotnetcodr.com/2014/01/13/net-developers-user-guide-for-troubleshooting-networking-problems-part-2/>) socket connection between the client and the server. This socket is maintained between two parties. It is a two-way full duplex communication channel (<https://dotnetcodr.com/2013/12/16/a-model-soa-application-in-net-part-1-the-fundamentals/>): the client can send data to the server and the server can send data back to the client over the same channel at any time.

The protocol for a web socket is “ws”, such as “ws://mysite.com”. Similarly a secure web socket follows the “wss” scheme. Note that not all browsers support web sockets. Check out the table on [this](http://caniuse.com/#search=sockets) (<http://caniuse.com/#search=sockets>) web site to see which browsers support it.

SignalR is a technology that alleviates the burden of setting up a web app which uses web sockets. There are many things you need to take care of with web sockets communication: serialising data, deserialising data, maintaining the connection, processing messages that arrive in no particular order. SignalR simplifies the construction of sockets-aware web apps by hiding a lot of the underlying complexity. You as a developer can then concentrate on the fun parts instead of worrying too much about low level details. SignalR can even help older browsers “understand” web sockets by emulating its behaviour.

## So what's SignalR?

As hinted on above SignalR is a Microsoft framework that makes working with web sockets easier for the programmer. You can make it available in your web project by downloading the Microsoft.AspNet.SignalR NuGet package. The most recent version at the time of writing this post is 2.0.3:



(<https://dotnetcodr.files.wordpress.com/2014/04/signalr-nuget.png>).

By the time you read this post there might be an updated package. If you see that some functionality described in this blog series is not compatible with a newer version then please comment about it and then I'll update the posts.

As we said before Web Sockets can be quite challenging to work with if you try to do so using some low-level JavaScript package. WebSockets is not the only type of connection that enables real-time two-way communication between the client and the server. Here come some more:

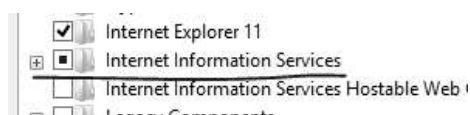
- Server-send events: a HTML5 based technology where the server sends automatic messages to the browser via HTTP ([http://en.wikipedia.org/wiki/Server-sent\\_events](http://en.wikipedia.org/wiki/Server-sent_events)).
- Comet – forever frame: a trick where JavaScript creates a hidden iFrame on the browser which holds the connection open so that the server can continue to send messages to the client ([http://en.wikipedia.org/wiki/Comet\\_\(programming\)](http://en.wikipedia.org/wiki/Comet_(programming))).
- Long polling: periodic requests to the server to get an updated set of information ([http://en.wikipedia.org/wiki/Push\\_technology](http://en.wikipedia.org/wiki/Push_technology)).

SignalR will test all 4 of these until it finds one that's available. It always tries WebSockets first as it is the the most efficient one of the 4 as far as overhead is concerned. Normally if the browser supports WebSockets then it can be used for communication. However, this is not always the case. In that case it will test the other three one after the other.

For WebSockets to succeed then even the web server will need to support this technology. You are presumably a .NET web developer so chances are high that you'll deploy your web app on IIS. Only IIS8 and above will support WebSockets. I have Window8.1 on the PC I'm building the demo on but IIS8 should be available on Windows8 as well. The built-in IIS Express server of Visual Studio 2013 supports WebSockets. If you'd like to deploy and test your web app locally on your PC then search for IIS on your PC. If you don't find it then you'll need to enable it first:

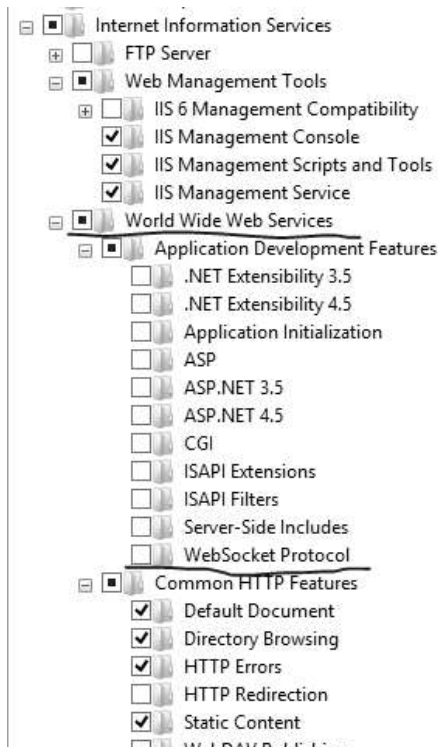


(<https://dotnetcodr.files.wordpress.com/2014/04/turn-windows-features-on-and-off.png>)



(<https://dotnetcodr.files.wordpress.com/2014/04/enable-iis.png>).

Also enable the WebSocket feature:



(<https://dotnetcodr.files.wordpress.com/2014/04/enable-websockets.png>).

Start IIS and check its version:



(<https://dotnetcodr.files.wordpress.com/2014/04/check-iis-version.png>).

If you're deploying your app on Windows Server then you'll need Windows Server 2012 and you'll need to enable the WebSocket feature as shown above.

The SignalR NuGet package will enable us to work with both the client side and server side part of the communication. On the server side the most important component we'll work with is called a Hub which is a class that hides the implementation details of the communication on the server side. On the client side we'll need some JavaScript code that can be called by SignalR to connect to this Hub.

In the [next post \(https://dotnetcodr.com/2014/05/19/introduction-to-websockets-with-signalr-in-net-part-2-code-basics/\)](https://dotnetcodr.com/2014/05/19/introduction-to-websockets-with-signalr-in-net-part-2-code-basics/), we'll start our demo.

View the list of posts on Messaging [here \(https://dotnetcodr.com/messaging/\)](https://dotnetcodr.com/messaging/).

FILED UNDER [.NET](#), [MESSAGING](#) TAGGED WITH [C#](#), [SIGNALR](#), [WEB SOCKETS](#)

### About Andras Nemes

I'm a .NET/Java developer living and working in Stockholm, Sweden.

## 2 Responses to *Introduction to WebSockets with SignalR in .NET Part 1: the basics*

**Wolfgang Kinkeldei** says:

May 15, 2014 at 7:47 am

I just wanted to simply say “thank you” for all your regular posts and your huge archive containing very very very valuable knowledge. I am a .NET newbie and find everything extremely well written and helpful. Sites like yours help me a lot to learn C# and its environment.

**Reply**

**SNithish** says:

December 15, 2017 at 6:30 pm

thanks for your post on this topic.


I have a web application using SignalR 1.1.2 and i want to enable websockets transport for that.

I tried enabling websocket protocol in IIS of Server 2012 and updated my project httpruntime and compilation tags with targetFramework = “4.5” and still i dont see the my application using websockets protocol. Is there any other settings i am missing to initiate websocket transport type for my application?

**Reply**

**Create a free website or blog at WordPress.com.**

### Advertisements

 JustAnswer  
**Chat w/ an expert Online Now**

REPORT THIS AD