SignalR - Development view

Olli Arokari
38218
oarokari@abo.fi

Table of contents

Developement view

Traditionally, HTTP links the server and client by requests (GET and POST are the most widely used) and if there are no requests by the client, the server and client are not communicating and no data is updated. The solution many websites use to this day are HTTP cookies [20], which enable storage of some stateful information (e.g. remembers that you have logged in). If the client does not request anything, it is impossible for the server to respond with any real-time information to the client.

When users interact with each other on a web application, real-time communication between client and server would decrease the amount of cookies. SignalR is a library that helps ASP.NET Core developers easily to add real-time functionality over HTTP to their existing applications by establishing a TCP connection between the server and client thus allowing information in real-time both ways.

Module Organization

SignalR Core consists of a C# library and client-side javascripts (since ASP.NET Core is a web framework). Clients can both be a .NET client or a JavaScript client. The library is sometimes obtainable as a NuGet package (one-click-install) but at the moment it is not. The most important thing from the library are the Hub -objects and mapping for the route that connecting clients use to connect to the Hubs (the mapping is done under .NET Core app Configuration by adding app.MapSignalR() using OWIN (the .NET web interface)). The Hubs work like controllers in an MVC software where the client is the view. The clients connect to a hub on the server and when connected the server may run functions that exist on the client and vice versa: the client can run functions that exist in the Hub that it is connected to. Examples with a JavaScript (webpage) client are displayed on the next page. [25][26][27]

The javascript client is obtainable as a npm package which is in theory a good thing since npm can keep the client files updated (Visual Studio 2017 can do this also). However since at the moment there is no way to tell npm to install the client in a specific folder the client is obtained to the root of the project. By manually moving the files we got it to work.

The following diagram contains information about the architecture itself. SignalR is divided in the server and client, but since the JS client is very simple and not of particular interest in terms of architecture we left it out.

Architecture table

Microsoft.AspNetCore.SignalR	Contains the RouteBuilder which is used to route connections to a Hub by name.
Microsoft.AspNetCore.SignalR.Core	Server library. Contains e.g. the Hub class and client identifiers such as groups and caller.
Microsoft.AspNetCore.SignalR.Client	Client library. This contains only the builder. Depends on: ● Microsoft.AspNetCore.SignalR.Client.Core ○ Contains the classes the builder uses, such as the HubConnection class. ● .NET Http ● Microsoft.AspNetCore.Sockets
Microsoft.AspNetCore.SignalR.Common	A common library for encoders, formatters and communication protocol functions.
samples	Works as documentation by giving working example projects.
test	Test enviroment used by contributors.
specs	Contains HubProtocol.md and TransportProtocols.md. Explains how SignalR works in theory.

Examples

Client invokes function on server (in the Hub).

Server (C#)	`[HubName("NameOfHub")]`
	`public class NameOfHub : Hub`
	`{`
	`  public void FunctionInHub(args)`
	`  {`
	`  //Client runs this code when connected`
	`  }`
	`}`

Client (JS)	`var connection = new signalR.HubConnection("serverIP/hubname");`
	`connection.start()`
	`  .then(() => { connection.server.invoke("FunctionInHub", arguments);}`
	`  .catch(err => { console.log("Error while connecting");}`
	`});`

Server invokes a function at client:

Client (JS)	`var connection = new signalR.HubConnection("serverIP/hubname");`
	
	`connection.on("FunctionAtClient", (arguments) => {`
	`  //Server can run this code at anytime`
	`});`
	
	`connection.start()`
	`  .then(() => { console.log("Connected"};`
	`  .catch(err => { console.log("Error while connecting");}`
	`});`

Server (C#)	`[HubName("NameOfHub")]`
	`public class NameOfHub : Hub`
	`{`
	`  public void SomeFunction(args)`
	`  {`
	`  Clients.Caller.FunctionAtClient(arguments);`
	`  }`
	`}`