EBook Gratis

APRENDIZAJE signalr

Free unaffiliated eBook created from
Stack Overflow contributors.

#signalr

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: signalr

It is an unofficial and free signalr ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official signalr.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con signalr

Observaciones

Esta sección proporciona una descripción general de qué es signalr, y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de signalr, y vincular a los temas relacionados. Dado que la Documentación para signalr es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Versiones

Versión	Fecha de lanzamiento
2.2.1	2016-07-13
2.2.0	2014-12-11
2.1.2	2014-09-09
2.1.1	2014-07-22
2.1.0	2014-05-21
2.0.3	2014-03-25
2.0.2	2014-01-22
2.0.1	2013-10-22
2.0.0	2013-08-29
1.2.2	2014-08-18
1.2.1	2014-02-10
1.2.0	2013-12-11
1.1.4	2013-10-22
1.1.3	2013-07-31
1.1.2	2013-05-24
1.1.1	2013-05-16
1.1.0	2013-05-11

Examples

Levantarse y correr

Requisitos de la versión IIS / .NET : ver aquí

SignalR 2+

1. Instale el paquete NuGet `Microsoft.AspNet.SignalR` (esta es la solución completa de SignalR) y le pedirá que instale las dependencias de otros paquetes. Acepta los términos e instálalos también.

2. Ahora que tenemos los scripts `.dlls` y cliente necesarios para generar nuestro propio SignalR Hub, creamos uno. Haga clic en su proyecto web, agregue una carpeta llamada `Hubs` o `SignalR` , y en ella agregue una clase **`NameOfYourChoosing`** *`Hub`* . Voy a nombrar el mío `MessagingHub.cs` .

3. Necesitamos derivar de la clase base `Hub` que se encuentra en nuestra dll SignalR que descargamos a través de NuGet. El código se vería así:

```
[HubName("messagingHub")]
public class MessagingHub : Hub
{
    //empty hub class
}
```

   Y en nuestra clase Startup.cs podemos avisar a `IAppBuilder` que vamos a utilizar SignalR.

```
public partial class Startup
{
    public void Configuration(IAppBuilder app)
    {
        app.MapSignalR();
        ConfigureAuth(app);
    }
}
```

4. Para hacer referencia a nuestro código de concentrador en el Cliente, necesitaremos importar / hacer referencia a 2 scripts (Aparte de la obvia referencia de jQuery). La principal `jQuery.signalR-version.js` archivo y la generada `hubs.js` archivo que genera SignalR de nuestro centro en concreto. Estos recursos pueden verse así:

```
// script tag src="/YourAppPath/scripts/jquery-1.6.4.js"
// script tag src="/YourAppPath/scripts/jquery.signalR-2.2.0.js"
// script tag src="/YourAppPath/signalr/hubs"
```

5. Dado que el JavaScript de SignalR se construye sobre jQuery (requiere> = v1.6.4), el código para conectar y desconectar el hub debe parecer bastante trivial. Aquí está en todo su esplendor (envuelto en un IIFE):

```
$(function() {
    //notice the camel casing of our hub name corresponding to the [HubName()] attribute
on the server
    var connection = $.connection.messagingHub;

    $.connection.hub.start().done(function () {
        alert("Connection Successful!");
    }).fail(function (reason) {
        console.log("SignalR connection failed: " + reason);
    });
});
```

6. A partir de este momento, deberíamos poder ejecutar la aplicación y establecer una conexión con el concentrador SignalR.

Uso de SignalR con la API web y la aplicación web de JavaScript, con soporte de CORS.

Objetivo: utilizar SignalR para la notificación entre la API web y la aplicación web basada en TypeScript / JavaScript, donde la API web y la aplicación web están alojadas en un dominio diferente.

Habilitación de SignalR y CORS en la API web: cree un proyecto de API web estándar e instale los siguientes paquetes de NuGet:

- Microsoft.Owin.Cors
- Microsoft.AspNet.WebApi.Cors
- Microsoft.AspNet.WebApi.Owin
- Microsoft.AspNet.SignalR.Core

Después de eso, puede deshacerse de `Global.asax` y agregar una clase de inicio OWIN en su lugar.

```
using System.Web.Http;
using System.Web.Http.Cors;
using Microsoft.Owin;
using Owin;

[assembly: OwinStartup(typeof(WebAPI.Startup), "Configuration")]
namespace WebAPI
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {

            var httpConfig = new HttpConfiguration();

            //change this configuration as you want.
            var cors = new EnableCorsAttribute("http://localhost:9000", "*", "*");
            httpConfig.EnableCors(cors);

            SignalRConfig.Register(app, cors);
```

```
            WebApiConfig.Register(httpConfig);

            app.UseWebApi(httpConfig);
        }
    }
}
```

Cree la clase `SignalRConfig` la siguiente manera:

```
using System.Linq;
using System.Threading.Tasks;
using System.Web.Cors;
using System.Web.Http.Cors;
using Microsoft.Owin.Cors;
using Owin;

namespace WebAPI
{
    public static class SignalRConfig
    {
        public static void Register(IAppBuilder app, EnableCorsAttribute cors)
        {

            app.Map("/signalr", map =>
            {
                var corsOption = new CorsOptions
                {
                    PolicyProvider = new CorsPolicyProvider
                    {
                        PolicyResolver = context =>
                        {
                            var policy = new CorsPolicy { AllowAnyHeader = true,
AllowAnyMethod = true, SupportsCredentials = true };

                            // Only allow CORS requests from the trusted domains.
                            cors.Origins.ToList().ForEach(o => policy.Origins.Add(o));

                            return Task.FromResult(policy);
                        }
                    }
                };
                map.UseCors(corsOption).RunSignalR();
            });
        }
    }
}
```

Hasta ahora acabamos de habilitar SignalR con CORS en el lado del servidor. Ahora veamos cómo puedes publicar eventos desde el lado del servidor. Para esto necesitamos un `Hub` :

```
public class NotificationHub:Hub
{
    //this can be in Web API or in any other class library that is referred from Web API.
}
```

Ahora, finalmente, un código para transmitir el cambio:

```
public class SuperHeroController : ApiController
{
    [HttpGet]
    public string RevealAlterEgo(string id)
    {
        var alterEgo = $"The alter ego of {id} is not known.";
        var superHero = _superHeroes.SingleOrDefault(sh => sh.Name.Equals(id));
        if (superHero != null)
        {
            alterEgo = superHero.AlterEgo;

            /*This is how you broadcast the change.
             *For simplicity, in this example, the broadcast is done from a Controller,
             *but, this can be done from any other associated class library having access to
NotificationHub.
             */
            var notificationHubContext =
GlobalHost.ConnectionManager.GetHubContext<NotificationHub>();
            if (notificationHubContext != null)
            {
                var changeData = new { changeType = "Critical", whatHappened = $"Alter ego of
{id} is revealed." };

                //somethingChanged is an arbitrary method name.
                //however, same method name is also needs to be used in client.
                notificationHubContext.Clients.All.somethingChanged(changeData);
            }
        }
        return alterEgo;
    }
}
```

Por lo tanto, hasta ahora, hemos preparado el lado del servidor. Para el lado del cliente, necesitamos `jQuery` y el paquete `signalr` . Puede instalar ambos con `jspm` . Instale los tipos para ambos, si es necesario.

No utilizaremos el proxy de JavaScript generado por defecto. Preferiremos crear una clase muy simple para manejar la comunicación de SignalR.

```
/**
 * This is created based on this gist: https://gist.github.com/donald-
slagle/bf0673b3c188f3a2559c.
 * As we are crreating our own SignalR proxy,
 * we don't need to get the auto generated proxy using `signalr/hubs` link.
 */
export class SignalRClient {

    public connection = undefined;
    private running: boolean = false;

    public getOrCreateHub(hubName: string) {
        hubName = hubName.toLowerCase();
        if (!this.connection) {
            this.connection = jQuery.hubConnection("https://localhost:44378");
        }

        if (!this.connection.proxies[hubName]) {
            this.connection.createHubProxy(hubName);
        }
```

```
        return this.connection.proxies[hubName];
    }

    public registerCallback(hubName: string, methodName: string, callback: (...msg: any[]) =>
void,
        startIfNotStarted: boolean = true) {

        var hubProxy = this.getOrCreateHub(hubName);
        hubProxy.on(methodName, callback);

        //Note: Unlike C# clients, for JavaScript clients,
        //      at least one callback needs to be registered,
        //      prior to start the connection.
        if (!this.running && startIfNotStarted)
            this.start();
    }

    start() {
        const self = this;
        if (!self.running) {
            self.connection.start()
                .done(function () {
                    console.log('Now connected, connection Id=' + self.connection.id);
                    self.running = true;
                })
                .fail(function () {
                    console.log('Could not connect');
                });
        }
    }
}
```

Por último, use esta clase para escuchar las transmisiones de cambio, de la siguiente manera:

```
/**
 * Though the example contains Aurelia codes,
 * the main part of SignalR communication is without any Aurelia dependency.
 */
import {autoinject, bindable} from "aurelia-framework";
import {SignalRClient} from "./SignalRClient";

@autoinject
export class SomeClass{

    //Instantiate SignalRClient.
    constructor(private signalRClient: SignalRClient) {
    }

    attached() {
        //To register callback you can use lambda expression...
        this.signalRClient.registerCallback("notificationHub", "somethingChanged", (data) => {
            console.log("Notified in VM via signalr.", data);
        });

        //... or function name.
        this.signalRClient.registerCallback("notificationHub", "somethingChanged",
this.somethingChanged);
    }
```

```
    somethingChanged(data) {
        console.log("Notified in VM, somethingChanged, via signalr.", data);
    }
}
```

Lea Empezando con signalr en línea: https://riptutorial.com/es/signalr/topic/5633/empezando-con-signalr

Capítulo 2: Manejando la Vida de Conexión

Parámetros

Parámetro	Detalles
parar Llamado	Este valor le indica cómo se desconectó un usuario, si se configuró como verdadero, entonces el usuario cerró explícitamente la conexión, de lo contrario, expiró el tiempo de espera.

Observaciones

Vale la pena señalar que mientras estas funciones todavía tienen acceso al Contexto, por lo tanto, puede obtener el ID de conexión e identificar quién / qué se ha desconectado.

Recuerde, un usuario puede tener ** más de un ID de conexión **, así que piense en cómo desea almacenar todas las ID de conexión para un usuario

Examples

Conectado

Cuando un usuario se conecta a su hub, se llama a `OnConnected()` . Puede anular esta función e implementar su propia lógica si necesita realizar un seguimiento o limitar el número de conexiones.

```
  public override Task OnConnected()
    {
        //you logic here

        return base.OnConnected();
    }
```

En desconectado

Sobrecargar la función de desconexión le permite manejar qué hacer cuando un usuario se desconecta.

```
public override Task OnDisconnected(bool stopCalled)
{
    //Your disconnect logic here

    return base.OnDisconnected(stopCalled);
}
```

Ejemplo de informar a otros en un grupo que un usuario se ha desconectado

```
    /// <summary>
    /// Overrides the onDisconnected function and sets the user object to offline, this can be
checked when other players interacts with them...
    /// </summary>
    /// <param name="stopCalled"></param>
    /// <returns></returns>
    public override Task OnDisconnected(bool stopCalled)
    {
        var user = GetUserDictionaryData();
        if (user != null)
        {
            user.Status = PlayerStatus.offline;
            var playerJson = ReturnObjectAsJSON(user.Player);
            Clients.OthersInGroup(user.GroupId).userDisconnected(playerJson);
        }

        return base.OnDisconnected(stopCalled);
    }
```

No hay que preocuparse por la conexión en signalr.

Si desea notificar a otros clientes / usuarios a través de la aplicación, no debe preocuparse por la conexión, ya que se crea una nueva conexión cada vez que visita otras páginas en la aplicación web.

Podemos aprovechar la característica de los usuarios de signalr para lograr lo mismo. Vea el ejemplo a continuación:

Aquí estamos creando en función del projectid y todos los demás usuarios de ese grupo reciben una notificación cuando accedemos a $ .connection.notificationHub.server.NotifyOthersInGroup (projectid, projectname); del archivo javascript del cliente

// Hub Ejemplo de SignalR

clase pública NotificationHub: Hub {

```
    public static readonly ConcurrentDictionary<string, HashSet<string>>
       ProjectLockUsers = new ConcurrentDictionary<string,HashSet<string>>();

    public void JoinGroup(int projectid)
    {
        string groupname = string.Format("ProjectLock_{0}", projectid);
        Groups.Add(Context.ConnectionId, groupname);

        AddUserToProjectLockGroup(groupname, Context.User.Identity.Name);
    }

    public void NotifyOthersInGroup(int projectid,string name)
    {
        string groupname = string.Format("ProjectLock_{0}", projectid);

        var allusers=null as HashSet<string>;
```

```
        if (ProjectLockUsers.TryGetValue(groupname, out allusers))
        {
            allusers.Remove(Context.User.Identity.Name);
            Clients.Users(allusers.ToList()).notifyUnlock(name);
        }
    }

    public override Task OnConnected()
    {
        return base.OnConnected();
    }
    public override Task OnDisconnected(bool stopCalled)
    {
        return base.OnDisconnected(stopCalled);
    }


    private void AddUserToProjectLockGroup(string projectId,string userId)
    {
        var userIds = null as HashSet<string>;


        if (Sessions.TryGetValue(projectId, out userIds) == false)
        {
            userIds = Sessions[projectId] = new HashSet<string>();
        }

        userIds.Add(userId);

        ProjectLockUsers[projectId] = userIds;
    }

}
```

Lea Manejando la Vida de Conexión en línea:
https://riptutorial.com/es/signalr/topic/6527/manejando-la-vida-de-conexion

Creditos

S. No	Capítulos	Contributors
1	Empezando con signalr	Community, Mark C., Sayan Pal, Scott Weldon
2	Manejando la Vida de Conexión	ashish, Jay