Introduction to WebSockets with SignalR in .NET Part 6: the basics of publishing to groups

JUNE 2, 2014 5 COMMENTS (HTTPS://DOTNETCODR.COM/2014/06/02/INTRODUCTION-TO-WEBSOCKETS-WITH-SIGNALR-IN-NET-PART-6-THE-BASICS-OF-PUBLISHING-TO-GROUPS/#COMMENTS)

Introduction

So far in this series on SignalR we've published all messages to all connected users. The goal of this post is to show how you can direct messages to groups of people. It is based on a request of a commenter on part 5 (https://dotnetcodr.com/2014/05/29/introduction-to-websockets-with-signalr-in-net-part-5-dependency-injection-in-hub/). You can think of groups as chat rooms in a chat application. Only members of the chat room should see messages directed at that room. All other rooms should remain oblivious of those messages.

We'll build upon the SignalR demo app we've been working on in this series. So have it open in Visual Studio. We'll simulate the following scenario:

- When a client connects for the first time they are assigned a random age between 6 and 100 inclusive
- The client joins a "chat room" based on this age by calling a function in ResultsHub
- When the client sends a message then only those already in the chat room will be notified

The real-life version would of course involve some sign-up page where the user can define which room to join or what their age is. You can also store those users if the chat application is closed to unanonymous users. However, all that would involve too much extra infrastructure irrelevant to the main goals.

Joining a group

We first need to assign an age to the client when they navigate to the Home page. Locate HomeController.cs. The Index action currently only returns a View. Extend it as follows:

```
public ActionResult Index()
{
    Random random = new Random();
    int next = random.Next(6, 101);
    ViewBag.Age = next;
        return View();
}
```

Normally you'd pass the age into the View in a view-model object but ViewBag will do just fine. In Index.cshtml add the following markup just below the Register message header:

```
<div>
    Your randomised age is <span id="ageSpan">@ViewBag.Age</span>
</div>
```

Now we want to call a specific Hub function as soon as the connection with the hub has been set up. The server side function will put the user in the appropriate chat room based on their age. Insert the following code into results.js just below the call to hub.start():

```
$.connection.hub.start().done(function ()
{
        var age = $("#ageSpan").html();
        resultsHub.server.joinAppropriateRoom(age);
});
```

When the start() function has successfully returned we'll read the age from the span element and run a method called joinAppropriateRoom in ResultsHub. Open ResultsHub.cs and define the function and some private helpers as follows:

```
public void JoinAppropriateRoom(int age)
{
    string roomName = FindRoomName(age);
    string connectionId = Context.ConnectionId;
    JoinRoom(connectionId, roomName);
    string completeMessage = string.Concat("Connection ", connectionId, " has joined the room called
    Clients.All.registerMessage(completeMessage);
}

private Task JoinRoom(string connectionId, string roomName)
{
    return Groups.Add(connectionId, roomName);
}

private string FindRoomName(int age)
{
    string roomName = "Default";
    if (age < 18)
    {
        roomName = "The young ones";
    }
    else if (age < 65)
    {
        roomName = "Still working";
    }
    else
    {
        roomName = "Old age pensioners";
    }
    return roomName;
}
```

We'll first find the appropriate room for the user based on the age. We've defined 3 broad groups: youngsters, employed and old age pensioners. We extract the connection ID and put it into the correct room. Note that we didn't need to define a group beforehand. It will be set up automatically as soon as the first member is added to it. We then also notify every client that there's a new member.

While we're at it let's define the function that will be called by the client to register the message together with their age. The age is again needed to locate the correct group. Add the following method to ResultsHub.cs:

```
public void DispatchMessage(string message, int age)
{
    string roomName = FindRoomName(age);
    string completeMessage = string.Concat(Context.ConnectionId
        , " has registered the following message: ", message, ". Their age is ", age, ". ");
    Clients.Group(roomName).registerMessage(completeMessage);
}
```

We again find the correct group, construct a message and send out the message to everyone in that group.

There's one last change before we can test this. Locate the newMessage function definition of the messageModel prototype in results.js. It currently calls the simpler sendMessage function of ResultsHub. Comment out that call. Update the function definition as follows:

```
newMessage: function () {
            var age = $("#ageSpan").html();
            //resultsHub.server.sendMessage(this.registeredMessage());
            resultsHub.server.dispatchMessage(this.registeredMessage(), age);
            this.registeredMessage("");
        },
.
.
.
```

Run the application. The first client should see that they have joined some room based on their age. In my case it looks as follows:

(https://dotnetcodr.files.wordpress.com/2014/05/first-client-joining-a-group.png)

Open some more browser windows with the same URL and all of them should be assigned an age and a group. The very first client should see them all. In my test I got the following participants:

(https://dotnetcodr.files.wordpress.com/2014/05/first-client-sees-all-subsequent-chat-participants.png)

I started up 6 clients: 3 old age pensioners, 2 youngsters and 1 still working. It's not easy to copy all windows so that we can see everything but here are the 6 windows:


(https://dotnetcodr.files.wordpress.com/2014/05/all-clients-joining-some-group-and-conversing.png)

The top left window was the very first client, a 77-year-old who joined the pensioners' room. The one below that is a 65-year-old pensioner. Under that we have a 24-year-old worker. Then on the right hand side from top to bottom we have a 15-year-old, a 90-year-old and finally a 6-year-old client. Then the 65-year-old client sent a message which only the other pensioners have receiver. Check the last message in each window: the clients of 24, 6 and 15 years of age cannot see the message. Then the 6-year-old kid sent a message. This time only the 2 youngsters were sent the message. You'll see that the 24-year old worker has not received any of the messages.

We've seen a way how you can direct messages based on group membership. You can define the names of the rooms in advance, e.g. in a database, so that your users can pick one upon joining your chat application. Then instead of sending in their age they can send the name of the room which to join. However, if you're targeting specific users without them being aware of these groups you'll need to collect the criteria from them based on which you can decide where to put them. In the above example the only criteria was their age. In other case it can be a more elaborate list of conditions.

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

FILED UNDER .NET 4.5, MESSAGING TAGGED WITH C#, SIGNALR, WEBSOCKETS

About Andras Nemes

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

5 Responses to *Introduction to WebSockets with SignalR in .NET Part 6: the basics of publishing to groups*

Marco Medrano says:

August 21, 2015 at 4:40 pm

Hi man!, thank you for your posts, they are very detailed so anyone can follow those. You know everyone would like to make it run, is it the code hosted in any place?

Reply

Marco Medrano says:

August 21, 2015 at 4:47 pm
Hoo, I just found you in github, awesome https://github.com/andras-nemes/signalrdemo

Reply

Andras Nemes says:
August 21, 2015 at 9:00 pm
Hi Marco, I'm glad you've found the link. You can find all my Github repos under the GITHUB menu of the blog.
//Andras

Marco Medrano says:
August 22, 2015 at 4:29 pm
I see thank you!

Alex says:
March 27, 2018 at 8:53 pm
I have a webapp that provides a websocket at http://site/notify for getting realtime data. Is it possible for signalr to get data from a site that doesn't use signalr?

Reply