Exercises in .NET with Andras Nemes

Tips and tricks in C# .NET

Introduction to WebSockets with SignalR in .NET Part 5: dependency injection in Hub

MAY 29, 2014 7 COMMENTS (HTTPS://DOTNETCODR.COM/2014/05/29/INTRODUCTION-TO-WEBSOCKETS-WITH-SIGNALR-IN-NET-PART-5-DEPENDENCY-INJECTION-IN-HUB/#COMMENTS)

Introduction

We've now gone through the basics of SignalR. We've looked at a messaging project and a primitive, but working stock ticker. Actually most of the work is put on the client in form of JavaScript code. The WebSockets specific server side code is not too complex and follows standard C# coding syntax.

However, the relative server side simplicity doesn't mean that our code shouldn't follow good software engineering principles. I mean principles such as SOLID (https://dotnetcodr.com/2013/08/12/solid-design-principles-in-net-the-single-responsibility-principle/), with special subjective weight given to my favourite, the letter 'D (https://dotnetcodr.com/2013/08/26/solid-design-principles-in-net-the-dependency-inversion-principle-and-the-dependency-injection-pattern/)'.

You'll recall from the previous post (https://dotnetcodr.com/2014/05/26/introduction-to-websockets-with-signalr-in-net-part-4-stock-price-ticker/) that the stock prices were retrieved from a concrete StockService class, an instance of which was created within the body of StartStockMonitoring(). The goal now is to clean up that code so that the stock prices are retrieved from an interface instead of a concrete class. The interface should then be a parameter of the ResultsHub constructor.

We'll use StructureMap (http://docs.structuremap.net/) to inject the dependency into ResultsHub.cs.

Demo

I'll use two sources from StackOverflow to solve the problem:

- Using StructureMap for dependency injection to SignalR 2.0.1 (http://stackoverflow.com/questions/20926728/using-structuremap-for-dependency-injection-to-signalr-2-0-1)
- How do you Resolve signalR v2.0 with StructureMap v2.6 (http://stackoverflow.com/questions/20705937/how-do-you-resolve-signalr-v2-0-with-structuremap-v2-6/)

Let's start by creating the abstraction. Add the following interface to the Stocks folder:

```
public interface IStockService
{
    dynamic GetStockPrices();
}
```

Then modify the StockService declaration so that it implements the abstraction:

```
public class StockService : IStockService
```

Change the ResultsHub constructor to accept the interface dependency:

```
private readonly IStockService _stockService;

public ResultsHub(IStockService stockService)
{
    if (stockService == null) throw new ArgumentNullException("StockService");
    _stockService = stockService;
    StartStockMonitoring();
}
```

Make sure you refer to the private field in the Task body of StartStockMonitoring:

```
Task stockMonitoringTask = Task.Factory.StartNew(async () =>
                {
                    while(true)
                    {
                        dynamic stockPriceCollection = _stockService.GetStockPrices();
                        Clients.All.newStockPrices(stockPriceCollection);
                        await Task.Delay(5000);
                    }
                }, TaskCreationOptions.LongRunning);
```

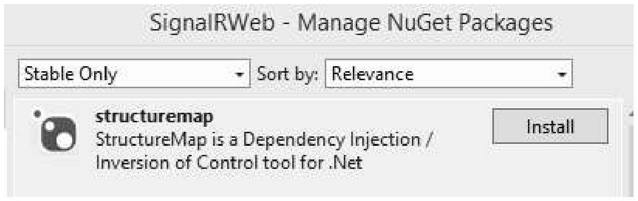
Next, create an interface for ResultsHub in the Stocks folder:

```
public interface IResultsHub
{
    void Hello();
    void SendMessage(String message);
}
```

…and have ResultsHub implement it:

```
public class ResultsHub : Hub, IResultsHub
```

Next import the StructureMap NuGet package:


SignalRWeb - Manage NuGet Packages

Stable Only Sort by: Relevance

structuremap
StructureMap is a Dependency Injection / Inversion of Control tool for .Net

Install

(https://dotnetcodr.files.wordpress.com/2014/04/structuremap-nuget-package.png)

We'll need a custom HubActivator to find the implementation of IResultsHub through StructureMap. Add the following class to the solution:

```
public class HubActivator : IHubActivator
{
    private readonly IContainer container;

    public HubActivator(IContainer container)
    {
        this.container = container;
    }

    public IHub Create(HubDescriptor descriptor)
    {
        return (IHub)container.GetInstance(descriptor.HubType);
    }
}
```

IContainer is located in the StructureMap namespace.

Next insert the following class that will initialise the StructureMap container:

```
public static class IoC
{
    public static IContainer Initialise()
    {
        ObjectFactory.Initialize(x =>
            {
                x.Scan(scan =>
                    {
                        scan.AssemblyContainingType<IResultsHub>();
                        scan.WithDefaultConventions();
                    });
            });
        return ObjectFactory.Container;
    }
}
```

The last step is to register our hub activator when the application starts. Add the following code to Application_Start() in Global.asax.cs:

```
IContainer container = IoC.Initialise();
GlobalHost.DependencyResolver.Register(typeof(IHubActivator), () => new HubActivator(container));
```

That's it actually. Set a breakpoint within the ResultsHub constructor and start the application. If all goes well then code execution should stop at the breakpoint. Inspect the incoming stockService parameter. If it's null then something's gone wrong but it shouldn't be. Let the code execution continue and you'll see that it works as before except that we've come a good step closer to loosely coupled code.

Read the finishing post in this series here (https://dotnetcodr.com/2014/06/02/introduction-to-websockets-with-signalr-in-net-part-6-the-basics-of-publishing-to-groups/).

View the list of posts on Messaging here (https://dotnetcodr.com/messaging/).
FILED UNDER .NET, DESIGN PRINCIPLES, MESSAGING TAGGED WITH C#, DEPENDENCY INJECTION, SIGNALR, WEB SOCKETS

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

7 Responses to *Introduction to WebSockets with SignalR in .NET Part 5: dependency injection in Hub*

Kim says:
May 29, 2014 at 8:39 am
Been reading a couple of your posts and I like them. Good explanations and very detailed. This post is also good, but I must say that I think SignalR has done a crappy job on how they do DI/IoC.

Your work however writing about it was excellent. Keep it up!

If I may have a request for future posts I would like to ask for this:
You showed how SignalR can be used for sending messages to all, to others and to a specific client. But how do I send to clients based on age for example? If I want to target people between 20 – 30 years old how can I do that? Groups will not do it since they are not dynamic in that way.

Would love a post about that topic!

/Kim

Reply

Andras Nemes says:
May 29, 2014 at 1:28 pm
Hi Kim, thanks for your feedback. Your request sounds very interesting, I'll think about it.
//Andras

Reply

Andras Nemes says:
May 29, 2014 at 1:41 pm
…I'm aiming to have a solution next Monday when the next "long" is due.
//Andras

Reply

Kim says:
May 29, 2014 at 6:06 pm
Sounds great! Thank you!

What I would really like is a SignalR way of doing it like they do it in xsockets.net.
If using the age example in xsockets I would do…

this.SendTo(p => p.Age >= 20 && p.Age <= 30, objectToSend, "topic");

If you can show me that I will be very happy!

Andras Nemes says:
June 2, 2014 at 8:01 am
…there it is, check out today's post. It's not exactly the way you've specified but it's the closest I could get.
//Andras

Kim says:
June 2, 2014 at 10:42 am
Thank you very much for your efforts! I think you example (post nr 6) is a nice way of showing how to achieve some of functionality I need.

Unfortunatelly the way SIgnalR works does not cover my needs in this case. I have to combine Age, Gender and Interest. So I have 3 dynamic criterias… I am sure you can see my concern. I have solved it with one line in xsockets since I can use

this.SendTo(Func condition, object o, string topic);

I have been using SignalR for a long time but I just dont know how to get past this right now.

Thanks again for your extra post, I really enjoy reading your blog.

Thank you
Kim

سعيد says:
May 29, 2014 at 9:47 am
Hello,
Thanks for this introduction.
There are few notes about 'Task.Factory.StartNew' and also creating a new unnecessary thread in web applications. more info:
http://blog.stephencleary.com/2013/11/taskrun-etiquette-examples-dont-use.html

Reply