عنوان: **تزریق خودکار وابستگی‌ها در SignalR**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۵ ۱۶:۷
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, jQuery, SignalR

فرض کنید لایه سرویس برنامه دارای اینترفیس و کلاس‌های زیر است:

```
namespace SignalR02.Services
{
    public interface ITestService
    {
        int GetRecordsCount();
    }
}
```

```
namespace SignalR02.Services
{
    public class TestService : ITestService
    {
        public int GetRecordsCount()
        {
            return 10; // It's just a sample to test IOC's.
        }
    }
}
```

قصد داریم از این لایه، توسط تزریق وابستگی‌ها در Hub برنامه استفاده کنیم:

```
[HubName("chat")]
    public class ChatHub : Hub
    {
        //جهت آزمایش تزریق خودکار وابستگی‌ها
        private readonly ITestService _testService;
        public ChatHub(ITestService testService)
        {
            _testService = testService;
        }

        public void SendMessage(string message)
        {
            var msg = string.Format("{0}:{1}", Context.ConnectionId, message);
            Clients.All.hello(msg);

            Clients.All.hello(string.Format("RecordsCount: {0}", _testService.GetRecordsCount()));
```

برنامه، همان برنامه‌ای است که در دوره جاری تکمیل گردیده است. فقط در اینجا سازنده کلاس اضافه شده و سپس اینترفیس ITestService به عنوان پارامتر آن تعریف گردیده است. در ادامه می‌خواهیم کار وهله سازی و تزریق نمونه مرتبط را توسط StructureMap به صورت خودکار انجام دهیم.

برای این منظور یک کلاس جدید را به نام StructureMapDependencyResolver به برنامه اضافه کنید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.AspNet.SignalR;
using StructureMap;

namespace SignalR02.Utils
{
    public class StructureMapDependencyResolver : DefaultDependencyResolver
    {
        private readonly IContainer _container;
        public StructureMapDependencyResolver(IContainer container)
        {
            if (container == null)
            {
                throw new ArgumentNullException("container");
```

```
            }
            _container = container;
        }
        public override object GetService(Type serviceType)
        {
            return !serviceType.IsAbstract && !serviceType.IsInterface && serviceType.IsClass
                                ? _container.GetInstance(serviceType)
                                : (_container.TryGetInstance(serviceType) ??
base.GetService(serviceType));
        }
        public override IEnumerable<object> GetServices(Type serviceType)
        {
            return
_container.GetAllInstances(serviceType).Cast<object>().Concat(base.GetServices(serviceType));
        }
    }
}
```

کار این کلاس، تعویض DefaultDependencyResolver توکار SignalR با StructureMap است. از این جهت که برای مثال در سراسر برنامه از StructureMap جهت تزریق وابستگی‌ها استفاده شده است و قصد داریم در قسمت Hub آن نیز یکپارچگی کار حفظ گردد.

برای استفاده از این کلاس تعریف شده فقط کافی است Application_Start فایل Global.asax.cs برنامه هاب را به نحو ذیل تغییر دهیم:

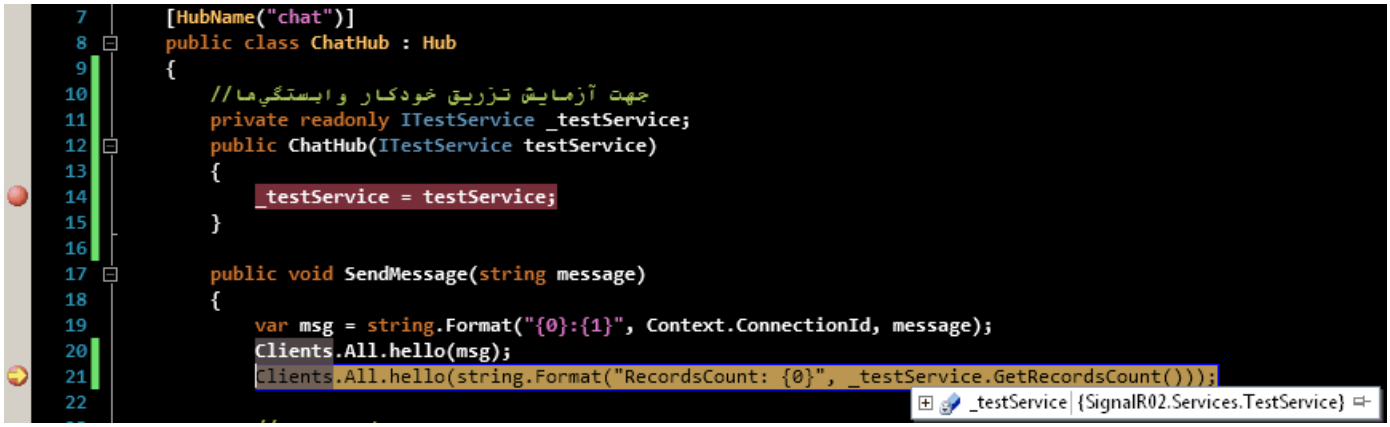
```
using System;
using System.Web;
using System.Web.Routing;
using Microsoft.AspNet.SignalR;
using SignalR02.Services;
using SignalR02.Utils;
using StructureMap;

namespace SignalR02
{
    public class Global : HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            ObjectFactory.Initialize(cfg =>
            {
                cfg.For<IDependencyResolver>().Singleton().Add<StructureMapDependencyResolver>();
                // the rest ...
                cfg.For<ITestService>().Use<TestService>();
            });
            GlobalHost.DependencyResolver = ObjectFactory.GetInstance<IDependencyResolver>();

            // Register the default hubs route: ~/signalr
            RouteTable.Routes.MapHubs(new HubConfiguration
            {
                EnableCrossDomain = true
            });
        }
    }
}
```

در اینجا در ابتدای کار IDependencyResolver توکار StructureMap با کلاس StructureMapDependencyResolver وهله سازی می‌گردد. سپس تعاریف متداول تنظیمات کلاس‌ها و اینترفیس‌های لایه سرویس برنامه اضافه می‌شوند. همچنین نیاز است GlobalHost.DependencyResolver توکار SignalR نیز به نحوی که ملاحظه می‌کنید مقدار دهی گردد.

اینبار اگر برنامه را اجرا کنید و سپس یکی از کلاینت‌های آن‌را فراخوانی نمائید، می‌توان مشاهده کرد که کار وهله سازی و تزریق وابستگی سرویس مورد استفاده به صورت خودکار انجام گردیده است:


[HubName("chat")]
public class ChatHub : Hub
{
//جهت آزمایش تزریق خودکار وابستگی‌ها
private readonly ITestService _testService;
public ChatHub(ITestService testService)
{
_testService = testService;
}
public void SendMessage(string message)
{
var msg = string.Format("{0}:{1}", Context.ConnectionId, message);
Clients.All.hello(msg);
Clients.All.hello(string.Format("RecordsCount: {0}", _testService.GetRecordsCount()));
_testService {SignalR02.Services.TestService}