عنوان: **مدیریت نگاشت ConnectionIdها در SignalR به کاربران واقعی سیستم**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۰۸ ۱۱:۳۴
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, SignalR

SignalR تنها از Context.ConnectionId خود با خبر است و بس. کاربران واقعی سیستم، پس از اعتبارسنجی می‌توانند با چندین و چند ConnectionId به سیستم متصل شوند؛ برای مثال گشودن چندین مرورگر یا باز کردن برگه‌های مختلف یک مرورگر و یا حتی استفاده از سایر کلاینت‌هایی که SignalR قابلیت کار کردن با آن‌ها را دارد. بنابراین باید بتوان بین ConnectionIdها و کاربران واقعی سیستم، تناظری را برقرار کرد و همچنین نباید تصور کرد که الزاما یک کاربر مساوی است با یک ConnectionId.

اعتبار سنجی کاربران در SignalR

تمام مباحث عنوان شده در مورد نحوه‌ی کار با Forms Authentication استاندارد یک برنامه وب، در SignalR نیز قابل دسترسی است. پس از اینکه کاربری به سایت وارد شد (با استفاده از روش‌های متداول؛ مانند یک صفحه‌ی لاگین)، اطلاعات او در یک Hub نیز قابل استفاده است. برای مثال می‌توان به خاصیت this.Context.User.Identity.IsAuthenticated دسترسی داشت.
به علاوه در این حالت برای محدود کردن دسترسی کاربران اعتبار سنجی نشده به یک هاب فقط کافی است فیلتر Authorize را به هاب اعمال کنیم. باید دقت داشت که این فیلتر در فضای نام Microsoft.AspNet.SignalR تعریف شده است.

```
[Authorize]
public class ChatHub : Hub
{
  //...
}
```

نگاشت اتصالات، به کاربران واقعی سیستم

```
public class User
    {
        public int Id { set; get; }
        public string Name { get; set; }
        // سایر خواص کاربر


        public HashSet<string> ConnectionIds { get; set; }
    }
```

با توجه به توضیحات ابتدای بحث، هر کاربر با چندین ConnectionId می‌تواند به سیستم متصل شود. بنابراین کلاس کاربران، دارای یک خاصیت اضافی که نیازی هم نیست تا به بانک اطلاعاتی نگاشت شود، به نام ConnectionIds همانند کلاس فوق خواهد بود.
سپس باید لیست اتصالات کاربر را در هربار اتصال و قطع اتصال او به روز کرد:

```
using System.Collections.Concurrent;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNet.SignalR;

namespace SignalR05.Common
{
    public class User
    {
        public int Id { set; get; }
        public string Name { get; set; }
        // سایر خواص کاربر


        public HashSet<string> ConnectionIds { get; set; }
```

```
    }

    public class ChatHubHub : Hub
    {
        private static readonly ConcurrentDictionary<string, User> Users = new
ConcurrentDictionary<string, User>();

        public override Task OnConnected()
        {
            connect();
            return base.OnConnected();
        }

        private void connect()
        {
            var userName = Context.User.Identity.Name;
            var connectionId = Context.ConnectionId;

            var user = Users.GetOrAdd(userName,
                _ => new User
                {
                    Name = userName,
                    ConnectionIds = new HashSet<string>()
                });
            lock (user.ConnectionIds)
            {
                user.ConnectionIds.Add(connectionId);
            }
        }

        public override Task OnReconnected()
        {
            connect();
            return base.OnReconnected();
        }

        public override Task OnDisconnected()
        {
            var userName = Context.User.Identity.Name;
            var connectionId = Context.ConnectionId;

            User user;
            Users.TryGetValue(userName, out user);
            if (user != null)
            {
                lock (user.ConnectionIds)
                {
                    user.ConnectionIds.RemoveWhere(cid => cid.Equals(connectionId));

                    if (!user.ConnectionIds.Any())
                    {
                        User removedUser;
                        Users.TryRemove(userName, out removedUser);

                        ///Clients.Others.userDisconnected(userName);
                    }
                }
            }

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

در این مثال با بازنویسی متدهای اتصال، اتصال مجدد و قطع اتصال یک کاربر، توانسته‌ایم:
الف) نگاشتی را بین یک Id اتصال و یک User واقعی سیستم برقرار کنیم.
ب) لیست اتصالات یک کاربر را نیز در اختیار داشته و در زمان قطع اتصال یکی از برگه‌های مرورگر او، تنها یکی از این Idهای اتصال را از لیست حذف خواهیم کرد.

اگر این لیست دیگر Id متصلی نداشت، با فراخوانی متد فرضی Clients.Others.userDisconnected، می‌توان به سایر کاربران مثلا یک Chat، خروج کامل این کاربر را اطلاع رسانی کرد.
با داشتن لیست اتصالات یک کاربر، می‌توان به سایر کاربران اطلاع داد که مثلا کاربر جدیدی به Chat room وارد شده است:

```
Clients.AllExcept(user.ConnectionIds.ToArray()).userConnected(userName);
```

AllExcept در اینجا یعنی سایر کاربران منهای کاربرانی که Id اتصالات آن‌ها ذکر می‌شود. چون این Idها تمامی متعلق به یک کاربر هستند، فراخوانی فوق به معنای اطلاع رسانی به همه، منهای کاربر جاری متصل است.

نظرات خوانندگان

نویسنده: سعید صالحی
تاریخ: ۱۳۹۳/۰۴/۲۵ ۱۲:۵۰

با سلام
خسته نباشید
در صورتی که بخواهیم پیغام فقط به همین یوزری که لاگین کرده بره به جای ؟ توی دستور پایین باید چی بذاریم؟ یا اگه دستور دیگه ای باید استفاده کنیم ممنون می‌شم اگه راهنمایی کنید

```
context.Clients.User("?").displayNotification();
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۵ ۱۳:۳۶

مطابق مطلب فوق باید ConnectionIdهای او را یافته و به آن‌ها پیام ارسال کنید. روش مدیریت و جمع آوری این ConnectionIdها با مثالی در اینجا بحث شده.
به صورت خلاصه باید تناظری را بین مشخصات کاربر لاگین شده به سیستم یا Context.User.Identity.Name و تمام Context.ConnectionId او برقرار کرد.
بعد با داشتن لیستی از ConnectionId های متناظر (ConcurrentDictionary مثال فوق)، می‌توان به کاربر خاصی پیام ارسال کرد.
در این دیکشنری، به ازای یک Context.User.Identity.Name (مشخصات کاربر لاگین شده)، لیست Idهای اتصال او موجود است.
بعد برای ارسال پیام به یک اتصال:

```
Clients.Client(someConnectionId).sayhello("....");
```

ارسال پیام به چند اتصال، یا لیستی از ConnectionId ها:

```
Clients.Clients(connectionIdsList).sayhello("....");
```

نویسنده: س محمدرضا برنتی
تاریخ: ۱۳۹۳/۱۱/۱۸ ۱۳:۵۸

وقتی یک کاربر مرورگری با چند برگه باز را ببندد، تنها برای یکبار

```
public override Task OnDisconnected()
```

فراخوانی می‌شود، در نتیجه تنها یک ConnectionId از لیست ConnectionId های آن کاربر کم می‌شود،درصورتی که باید به تعداد آن برگه‌ها متد OnDisconnected فراخوانی شود، شما برای حل این مشکل راه حلی دارید ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۱۸ ۱۴:۱۲

- البته این متد با این امضاء از نگارش جدید SiganlR حذف شده‌است. نسخه‌ی به روز رسانی شده‌ی آن در اینجا .
+ بحثی در اینجا شده و پیشنهاد کردند این کد را در سمت کلاینت اضافه کنید:

```
window.onbeforeunload = function (e) {
    $.connection.hub.stop();
};
```

نویسنده: س محمدرضا برنتی

تاریخ: ۱۳۹۳/۱۱/۱۸ ۱۵:۷

با توجه به اینکه این کد سمت کلاینت قابل ویرایش است ، راه حل امنی برای تعیین متصل بودن و یا غیرمتصل بودن یک ConnectionId محسوب نمی‌شود و ظاهرا تنها راه حل برای بررسی وضعیت اتصال یک ConnectionId ، چک کردن اتصال آن در دوره‌های زمانی مشخص است .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۱۸ ۱۵:۳۶

- کلاینت سمت کاربر SiganlR که درون مرورگر اجرا می‌شود، اساسا جاوا اسکریپتی است. (البته برای جاوا یا دات نت و امثال آن هم کلاینت مخصوص دارد؛ ولی بحث مرورگر آن مشخص است)
+ این متد خاص هاب سمت سرور، در آخرین نگارش SiganlR به این نحو تغییر کرده‌است:

```
public override Task OnDisconnected(bool stopCalled)
{
    if (stopCalled)
    {
        // We know that Stop() was called on the client,
        // and the connection shut down gracefully.
    }
    else
    {
        // This server hasn't heard from the client in the last ~35 seconds.
        // If SignalR is behind a load balancer with scaleout configured,
        // the client may still be connected to another SignalR server.
    }

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

اگر پارامتر stopCalled با مقدار true فراخوانی شد، یعنی سمت کلاینت، با استفاده از کدهای جاوا اسکریپتی SignalR (فراخوانی شده به صورت خودکار در حین بستن یک تب یا مرورگر یا به صورت دستی به نحوی که عنوان شد)، درخواست بسته شدن صفحه را داده‌است. اگر مقدار آن false بود، یعنی سرور تشخیص داده‌است که در طی 35 ثانیه‌ی قبل کاربر فعالیتی نداشته‌است.