عنوان: **بررسی متد های یک طرفه در WCF**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۸/۱۶ ۹:۵
آدرس: www.dotnettips.info
برچسب‌ها: WCF, OneWay Operation

در WCF به صورت پیش فرض متدها به صورت Request-Response هستند. این بدین معنی است که هر زمان درخواستی از سمت کلاینت به سرور ارسال شود تا زمانی که پاسخی از سمت سرور به کلاینت برگشت داده نشود، کلاینت منتظر خواهد ماند. برای مثال:
پروژه ای از نوع Wcf Service App می‌سازیم و یک سرویس با یک متد که خروجی آن نیز void است خواهیم داشت. به صورت زیر:

```
[ServiceContract]
    public interface ISampleService
    {
        [OperationContract]
        void Wait();
    }
```

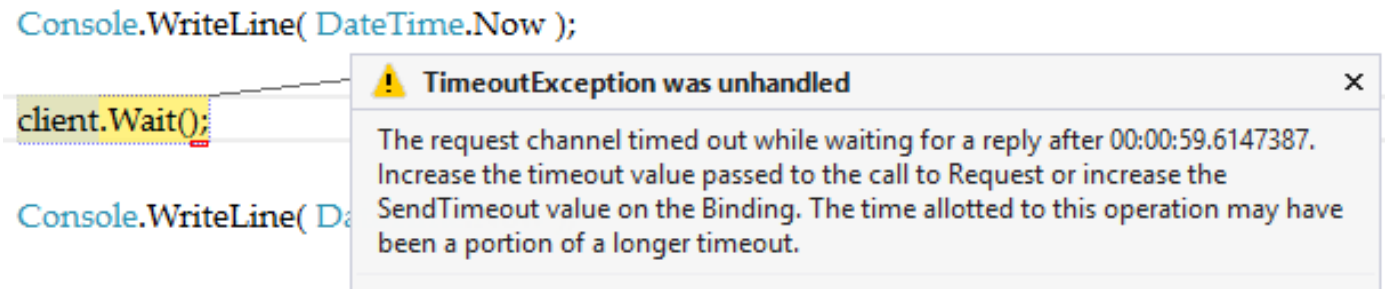
پیاده سازی Contract بالا:

```
public class SampleService : ISampleService
    {
        public void Wait()
        {
            Thread.Sleep( new TimeSpan( 0, 1, 0 ) );
        }
    }
```

در متد Wait، به مدت یک دقیقه اجرای برنامه سمت سرور را متوقف می‌کنیم. حال در یک پروژه از نوع Console App، سرویس مورد نظر را اضافه کرده و متد Wait آن را فراخوانی می‌کنیم. به صورت زیر:

```
class Program
    {
        static void Main( string[] args )
        {
            SampleService.SampleServiceClient client = new SampleService.SampleServiceClient();

            Console.WriteLine( DateTime.Now );

            client.Wait();

            Console.WriteLine( DateTime.Now );

            Console.ReadKey();
        }
    }
```

همان طور که می‌بینید قبل از فراخوانی متد Wait زمان جاری سیستم را نمایش داده و سپس بعد از فراخوانی دوباره زمان مورد را نمایش می‌دهیم. در مرحله اول با خطای زیر مواجه خواهیم شد:



دلیل اینکه Timeout Exception پرتاب شد این است که به صورت پیش فرض مقدار خاصیت sendTimeout برابر 59 ثانیه است، در نتیجه قبل از اینکه پاسخی از سمت سرور به کلاینت برگشت داه شود این Exception رخ می‌دهد. برای حل این مشکل کافیست در فایل app.config کلاینت در قسمت تنظیمات Binding ، تغییر زیر را اعمال کنیم:

```
<basicHttpBinding>
     <binding name="BasicHttpBinding_ISampleService" sendTimeout="0:2:0"/>
 </basicHttpBinding>
```

حال خروجی به صورت زیر است:

```
Before calling wait : 11/6/2013 9:58:29 PM
After calling wait : 11/6/2013 9:59:33 PM
```

مشخص است که تا زمانی که عملیات سمت سرور به پایان نرسد،(یا توجه به اینکه خروجی متد سمت سرور void است) اجرای برنامه در کلاینت نیز متوقف خواهد بود(اختلاف زمان‌های بالا کمی بیش از یک دقیقه است).

در این مواقع زمانی که باید متدی سمت سرور فراخوانی شود و قرار نیست که خروجی نیز در اختیار کلاینت قرار دهد بهتر است که از متدهای یک طرفه استفاده نماییم. متدهای یک طرفه یا به اصطلاح OneWay، هیچ پاسخی را به کلاینت برگشت نمی‌دهند و بلافاصله بعد از فراخوانی، کنترل اجرای برنامه را در اختیار کلاینت قرار خواهند داد. برای تعریف یک متد به صورت یک طرفه کافیست به صورت زیر عمل نماییم(مقدار خاصیت IsOneWay را در OperationContractAttribute برابر true خواهیم کرد):

```
[ServiceContract]
    public interface ISampleService
    {
        [OperationContract( IsOneWay = true )]
        void Wait();
    }
```

حال اگر سرویس سمت کلاینت را به روز کرده و برنامه را اجرا کنیم خروجی به صورت زیر تغییر می‌کند:

```
Before calling wait : 11/6/2013 10:08:36 PM
After calling wait : 11/6/2013 10:08:40 PM
```

می‌بینید که اختلاف زمان‌های بالا در حد چند ثانیه است که آن هم صرفا جهت فراخوانی متد سمت سرور بوده است. نکته مهم قابل ذکر این است که سرویس دهنده زمانی که با درخواستی جهت اجرای متد یک طرفه روبرو می‌شود، از آن جا که اجباری برای اجرای متد در همان زمان نیست در نتیجه این درخواست‌ها در بافر ذخیره می‌شوند و سپس در زمان مناسب اجرا خواهند شد. اگر بافر برای ذخیره اجرای متدهای یک طرفه پر باشد در این حالت کلاینت برای فراخوانی متدهای یک طرفه بعدی باید منتظر خالی شدن بافر سمت سرور بماند.

نظرات خوانندگان

نویسنده: reza110
تاریخ: ۱۳۹۲/۰۸/۱۹ ۱۴:۵۶

می‌خواستم بدانم اگر مثلا به جای دستور Thread.Sleep در خواست در سمت سرور اجرای یک دستور روی دیتابیس باشد و به هر دلیلی ارتباط کلاینت قطع شود چگونه می‌توان ادامه کار سمت سرور را متوقف کرد.
طبق بررسی روی task manager کرده ام حافظه مصرف شده همچنان افزایش می‌یابد.

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۸/۲۰ ۱۵:۵۶

زمانی که ارتباط بین سرور و کلاینت به هر دلیلی قطع شود یا به بنا به دلایلی طی انجام عملیات سمت سرور Exception رخ دهد، وضعیت ChannelFactory برای آن سرویس به حالت Faulted تغییر پیدا می‌کند. در نتیجه دیگر امکان استفاده از این کانال ارتباطی میسر نیست و باید یک کانال ارتباطی جدید تهیه نمایید. اما برای اینکه بعد از متوقف سازی عملیات سمت سرور حافظه مصرفی دوباره بازگردانده شود باید از مفهوم UnitOfWork بهره جست البته با مقداری تغییرات برای همگام سازی با درخواست‌های WCF.
روش مورد استفاده من به صورت زیر است(با فرض اینکه از EntityFramework به عنوان ORM استفاده میکنید):
» ابتدا یک Extension برای OperationContext تعریف می‌کنیم(با فرض اینکه IDatabaseContext نماینده کلاس DbContext پروژه است):

```
private class OperationContainerExtension : IExtension<OperationContext>
        {
            public OperationContainerExtension( IDatabaseContext dbContext, string contextKey )
            {
                this.CurrentDbContext = dbContext;
                this.ContextKey = contextKey;
            }

            public IDatabaseContext CurrentDbContext
            {
                get;
                private set;
            }

            public string ContextKey
            {
                get;
                private set;
            }

            public void Attach( OperationContext owner )
            {
            }

            public void Detach( OperationContext owner )
            {
            }
        }
```

بعد در هنگام نمونه سازی از UnitOfWork در لایه سرویس Extension بالا به OperationContext جاری اضافه خواهد شد(اگر از IOC Container خاصی استفاده می‌کنید باید کد عملیات وهله سازی کلاس UnitOfWork را با کدهای زیر مزین کنید):

```
if ( OperationContext.Current != null )
            {
                OperationContext.Current.Extensions.Add( new OperationContainerExtension( dbContext ,
CONTEXTKEY ) );
                OperationContext.Current.OperationCompleted +=
CurrentOperationContext_OperationCompleted;
                 OperationContext.Current.Channel.Faulted += Channel_Faulted;
             }
```

می بینید که برای رویداد OperationCompleted و رویداد Fault در Channel نیز کدهای Dispose کردن UnitOfWork و همچنین DbContext را قرار دادم. به صورت زیر:

```
void Channel_Faulted( object sender, EventArgs e )
        {
            IDatabaseContext dbContext = GetDbContext();
            if ( dbContext != null )
            {
                dbContext.Dispose();
                GC.Collect();
            }
        }

        private void CurrentOperationContext_OperationCompleted( object sender, EventArgs e )
        {
            IDatabaseContext dbContext = GetDbContext();
            if ( dbContext != null )
            {
                dbContext.Dispose();
                GC.Collect();
            }
        }
```

اگر به کدهای بالا دقت کنید متد GetDbContext نوشته شده برای به دست آوردن DbContext جاری در Session مربوطه است. کد آن نیز به صورت زیر می‌باشد

```
protected override IDatabaseContext GetDbContext()
        {
            if ( OperationContext.Current != null )
            {
                var operationContainerExtension =
OperationContext.Current.Extensions.OfType<OperationContainerExtension>().FirstOrDefault( e =>
e.ContextKey == CONTEXTKEY );
                if ( operationContainerExtension != null )
                {
                    return operationContainerExtension.CurrentDbContext;
                }
                return staticDbContext;
            }
            else
                return staticDbContext;
        }
```

نکته آخر هم این است که CONTEXTKEY صرفا یک فیلد از نوع string ولی با مقدار Guid برای به دست آوردن Extension مربوطه می‌باشد و تعریف آن در سازنده کلاس خواهد بود.

```
private string CONTEXTKEY = Guid.NewGuid().ToString();
```

در این صورت به طور قطع تمام منابع مورد استفاده سرویس‌های سمت سرور بعد از اتمام عملیات یا حتی وقوع هر خطا آزاد خواهند شد. اما اگر NHibernate را به عنوان ORM ترجیح می‌دهید به جای IDatabaseContext باید از ISession استفاده نمایید.