عنوان: **مدیریت تغییرات در سیستم های مبتنی بر WCF**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۳/۰۲ ۱۵:۲۰
آدرس: www.dotnettips.info
برچسب‌ها: WCF, versioning

تشریح مسئله : در صورتی که بعد از انتشار برنامه؛ در نسخه بعدی مدل سمت سرور تغییر کرده باشد و امکان بروز رسانی مدل‌های سمت کلاینت وجود نداشته باشد برای حل این مسئله بهترین روش کدام است.
نکته : برای فهم بهتر مطالب آشنایی اولیه با مفاهیم WCF الزامی است.
ابتدا مدل زیر را در نظر بگیرید:

```
[DataContract]
    public class Book
    {
        [DataMember]
        public int Code { get; set; }

        [DataMember]
        public string Name { get; set; }
    }
```

حالا یک سرویس برای دریافت و ارسال اطلاعات این مدل به کلاینت می‌نویسیم.

```
[ServiceContract]
    public interface ISampleService
    {
        [OperationContract]
        IEnumerable<Book> GetAll();

        [OperationContract]
        void Save( Book book );
    }
```

و سرویسی که Contract بالا رو پیاده سازی کند.

```
public class SampleService : ISampleService
    {
        public List<Book> ListOfBook
        {
            get;
            private set;
        }

        public SampleService()
        {
            ListOfBook = new List<Book>();
        }
        public IEnumerable<Book> GetAll()
        {
            ListOfBook.AddRange( new Book[]
            {
                new Book(){Code=1 , Name="Book1"},
                new Book(){Code=2 , Name="Book2"},
            } );
            return ListOfBook;
        }

        public void Save( Book book )
        {
            ListOfBook.Add( book );
        }
    }
```

متد GetAll برای ارسال اطلاعات به کلاینت و متد Save نیز برای دریافت اطلاعات از کلاینت.
حالا یک پروژه Console Application بسازید و از روش AddServiceReference سرویس مورد نظر را به Client اضافه کنید.
برنامه را تست کنید. بدون هیچ مشکلی کار می‌کند.

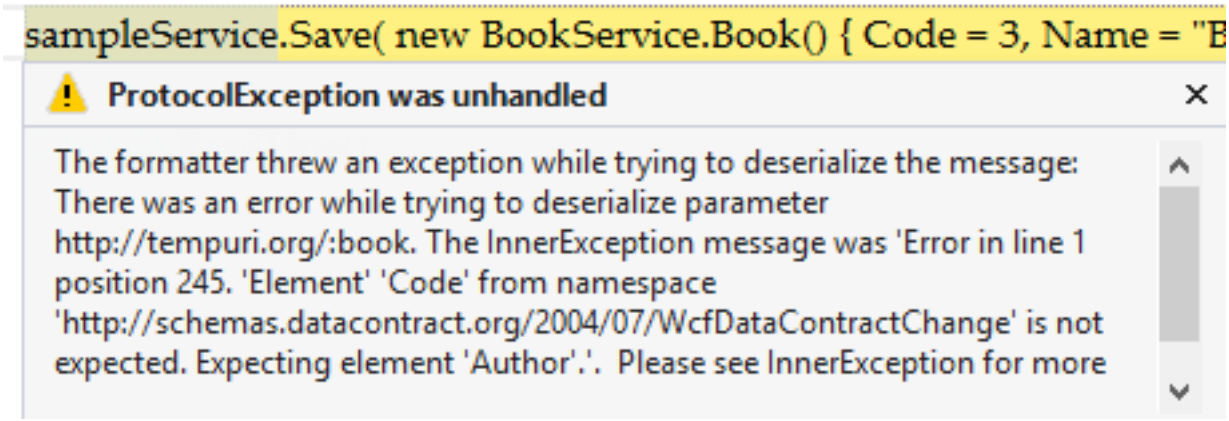
حالا اگر در نسخه بعدی سیستم مجبور شویم به مدل Book یک خاصیت دیگر به نام Author را نیز اضافه کنیم و امکان Update کردن سرویس در سمت کلاینت وجود نداشته باشد چه اتفاقی خواهد افتاد.
به صورت زیر:

```
[DataContract]
    public class Book
    {
        [DataMember]
        public int Code { get; set; }
        [DataMember]
        public string Name { get; set; }

        [DataMember]
        public string Author { get; set; }
    }
```

به طور پیش فرض اگر در DataContractهای سمت سرور و کلاینت اختلاف وجود داشته باشد این موارد نادیده گرفته می‌شوند.
یعنی همیشه مقدار خاصیت Author برابر null خواهد بود.
نکته : برای Value Typeها مقادیر پیش فرض و برای Reference Typeها مقدار Null.
اگر برای DataMemberAttribute خاصیت IsRequired را برابر true کنیم از این پس برای هر درخواستی که مقدار Author آن مقدار نداشته باشد یک Protocol Exception پرتاب می‌شود. به صورت زیر:

```
[DataMember( IsRequired = true )]
public string Author { get; set; }
```



اما این همیشه راه حل مناسبی نیست.
روش دیگر این است که Desrialize کردن مدل را تغییر دهیم. بدین معنی که هر گاه مقدار Author برابر Null بود یک مقدار پیش فرض برای آن در نظر بگیریم. این کار با نوشتن یک متد و قراردادن OnDeserializingAttribute به راحتی امکان پذیر است.
کلاس Book به صورت زیر تغییر می‌کند.

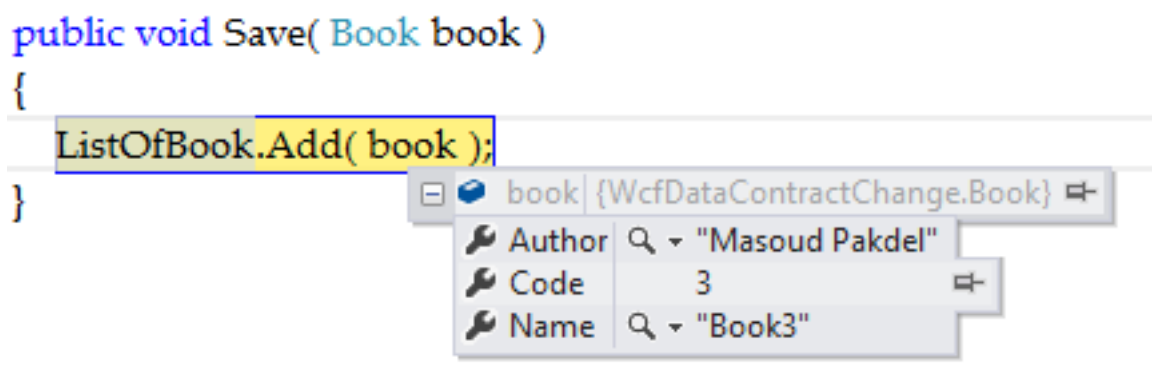
```
[DataContract]
    public class Book
    {
        [DataMember]
        public int Code { get; set; }
        [DataMember]
        public string Name { get; set; }

        [DataMember( IsRequired = true )]
        public string Author { get; set; }

        [OnDeserializing]
        private void OnDeserializing( StreamingContext context )
        {
            if ( string.IsNullOrEmpty( Author ) )
            {
```

```
                Author = "Masoud Pakdel";
            }
        }
    }
```

حال اگر از سمت کلاینت کلاس Book دریافت شود که مقدار خاصیت Author آن برابر Null باشد توسط متد OnDeserializing مقدار پیش فرض به آن اعمال می‌شود.مثل تصویر زیر:



روش بعدی استفاده از اینترفیس IExtensibleDataObject است. بعد از اینکه کلاس Book این اینترفیس را پیاده سازی کرد مشکل Versioning Round Trip حل می‌شود. به این صورت که سرویس یا کلاینتی که نسخه قدیمی را می‌شناسد اگر نسخه جدید را دریافت کند خصوصیاتی را که نمی‌شناسد مثل Author در خاصیت ExtensionData ذخیره می‌شود و هنگامی که کلاس Book برای سرویس یا کلاینتی که نسخه جدید را می‌شناسد DataContractSerializer اطلاعات مورد نظر را از خصوصیت ExtensionData بیرون می‌کشد و کلاس Book جدید را باز سازی می‌کند. بررسی کلاس ExtensionData توسط خود DataContractSreializer انجام می‌شود و نیاز به هیچ گونه ای کد نویسی ندارد.

```
[DataContract]
    public class Book : IExtensibleDataObject
    {
        [DataMember]
        public int Code { get; set; }
        [DataMember]
        public string Name { get; set; }

        [DataMember]
        public string Author { get; set; }

        public virtual ExtensionDataObject ExtensionData
        {
            get { return _extensionData; }
            set
            {
                _extensionData = value;
            }
        }
        private ExtensionDataObject _extensionData;
    }
```

اگر کد متد GetAll سمت سرور را به صورت زیر تغییر دهیم که خاصیت Author هم مقدار داشته باشد با استفاده از خاصیت ExtensionData کلاینت هم از این مقدار مطلع خواهد شد.

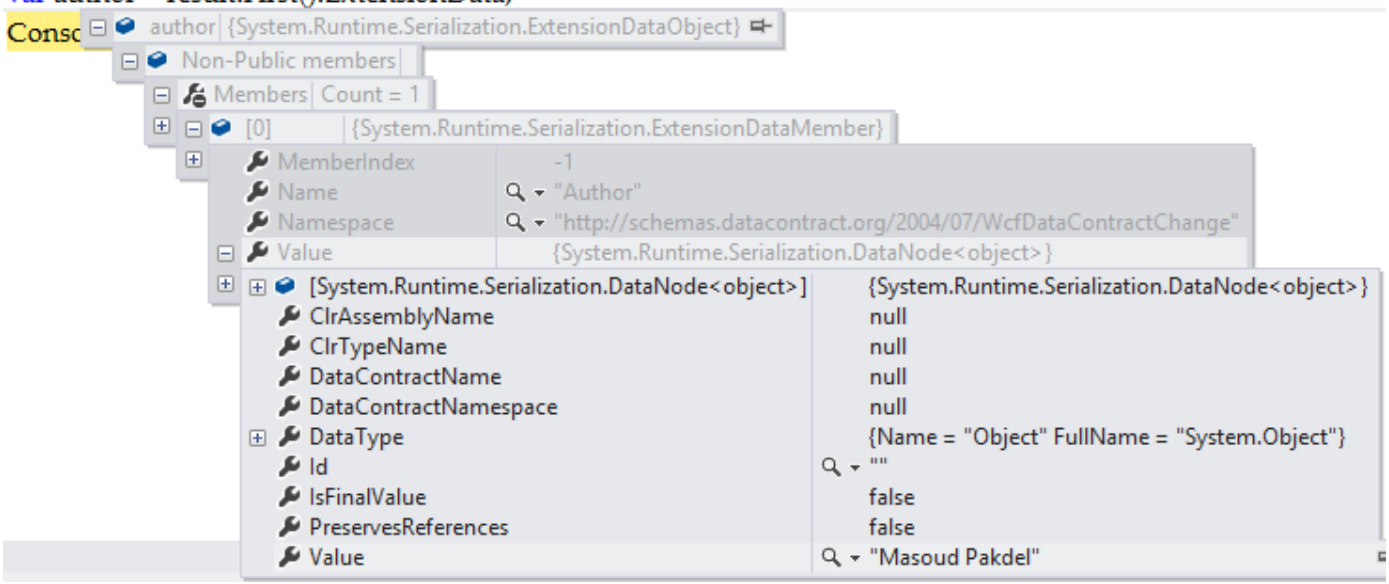
```
public IEnumerable<Book> GetAll()
        {
            ListOfBook.AddRange( new Book[]
            {
                new Book(){Code=1 , Name="Book1", Author="Masoud Pakdel"},
                new Book(){Code=2 , Name="Book2" },
```

```
            } );
            return ListOfBook;
        }
```

کلاینت هم به صورت زیر :

```
var result = sampleService.GetAll();
```

```
var author = result.First().ExtensionData;
```



همان طور که می‌بینید این نسخه از کلاینت هیچ گونه اطلاعی از وجود یک خاصیت به نام Author ندارد ولی از طریق ExtensionData متوجه می‌شود یک خاصیت به نام Author به مدل سمت سرور اضافه شده است.

اما در صورتی که قصد داشته باشیم که یک سرویس خاص از همان نسخه قدیمی کلاس Book استفاده کند و نیاز به نسخه جدید آن نداشته باشد می‌توانیم این کار را از طریق مقدار دهی True به خاصیت IgnoreExtensionDataObject در ServiceBehaviorAttribute انجام داد. بدین شکل

```
[ServiceBehavior( IgnoreExtensionDataObject = true )]
  public class SampleService : ISampleService
```

از این پس سرویس بالا از همان مدل Book بدون خاصیت Author استفاده می‌کند.

منابع :

http://msdn.microsoft.com/en-us/library/system.runtime.serialization.iextensibledataobject.aspx

http://msdn.microsoft.com/en-us/library/ms731083.aspx

http://msdn.microsoft.com/en-us/library/ms733832.aspx

و...