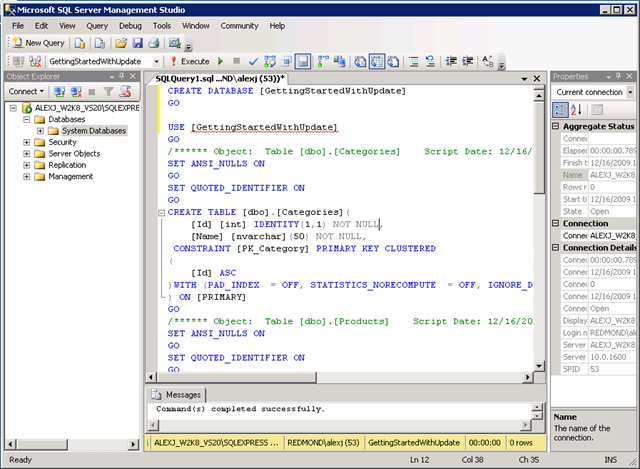
**[WCF Data Service QuickStart](http://www.cnblogs.com/shanyou/archive/2010/02/14/1668210.html)**

[开放数据协议(OData)](http://www.odata.org/)是 一个查询和更新数据的Web协议。OData是基于诸如HTTP和AtomPub的国际标准创建的，它提供了一个跨平台的数据通信的方案。OData应用 了web技术如HTTP、Atom发布协议(AtomPub)和JSON等来提供对不同应用程序，服务和存储的信息访问。SharePoint 2010, SQL Server 2008 R2, PowerPivot, Windows Azure Table Storage, 和第三方的产品像 [IBM’s WebSphere eXtreme Scale](http://www.stephenforte.net/ct.ashx?id=28df55a8-8811-4c4b-b319-75c5c58d1444&url=http%3a%2f%2fwww.ibm.com%2fdeveloperworks%2fwebsphere%2fdownloads%2fxs_rest_service.html)都使用OData。

首先，WCF Data Services是WCF服务，所以你可以使用所有现有的WCF知识。其次，WCF Data Services已经实现了OData拓扑，于是你可以致力于你的数据格式在你的程序中的表示，而不是AtomPub/JSON这些真正在网络上传递的数 据格式。再有，WCF Data Services致力于数据传输，而不是数据存储。你的数据可以存放在任何位置：本地的数据库，云端的数据库，外部的web services，xml文件，等等。无论数据是怎么来的，你都可以用同样的方式来发布/使用它们。

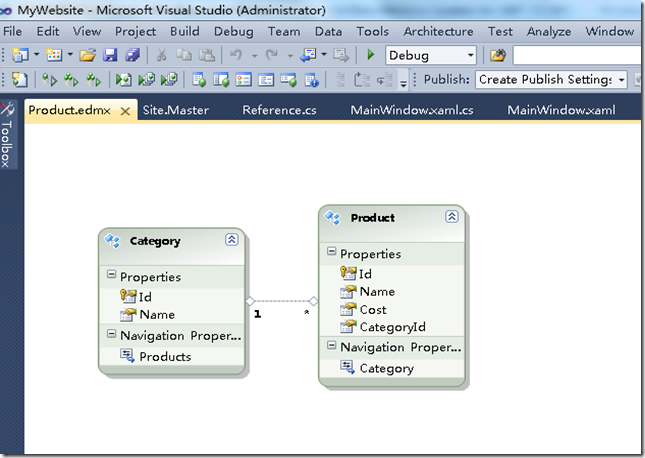
使用Visual Studio 2010里头使用WCF Data Service，就是使用OData发布的Restful服务，下面的例子演示2010里头的WCF Data Service.

1、创建一个数据库，使用SQL Server Management Studio并运行 [这个脚本](http://blogs.msdn.com/astoriateam/attachment/9938290.ashx).



创建了数据库，包含两个表Categories和Products，两个表里头分别插入了2条记录。

2、创建一个Web Application，然后创建一个Entity Data Model，如下图所示

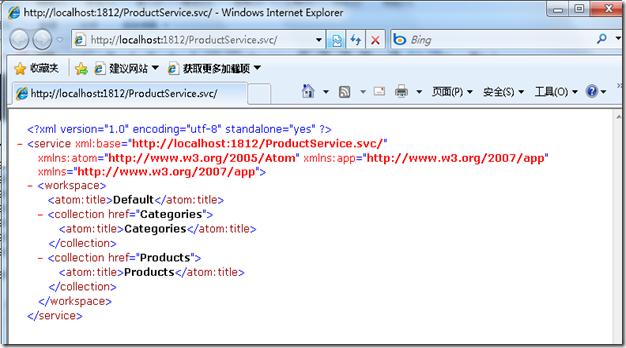
[](http://images.cnblogs.com/cnblogs_com/shanyou/WindowsLiveWriter/WCFDataServiceQuickStart_D31F/image_2.png)

3、创建一个Data Service暴露我们的模型：

using System;   
using System.Collections.Generic;   
using System.Data.Services;   
using System.Data.Services.Common;   
using System.Linq;   
using System.ServiceModel.Web;   
using System.Web;

namespace MyWebsite   
{   
    public class ProductService : DataService<GettingStartedWithUpdateEntities>   
    {   
        // This method is called only once to initialize service-wide policies.   
        public static void InitializeService(DataServiceConfiguration config)   
        {   
            // TODO: set rules to indicate which entity sets and service operations are visible, updatable, etc.   
            // Examples:   
            config.SetEntitySetAccessRule("\*", EntitySetRights.AllRead);   
            config.SetServiceOperationAccessRule("\*", ServiceOperationRights.All);   
            config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V2;   
        }   
    }   
}

4、验证ProductService服务，在浏览器中查看ProductService.svc,如下图所示：

[](http://images.cnblogs.com/cnblogs_com/shanyou/WindowsLiveWriter/WCFDataServiceQuickStart_D31F/image_4.png)

上图的内容就是AtomPub协议，首先,您向服务器的Atom链接发起带验证的GET请求(authenticated GET),以获得关于当前可用服的描述。服务器返回了一个(描述)Atom服务的XML文件，其中列出了一些列workspace，每个 workspace包含对应的collection集合。一个workspace可能是一个博客，wiki命名空间或内容集等通过您用户名/密码可以访问 的资源。

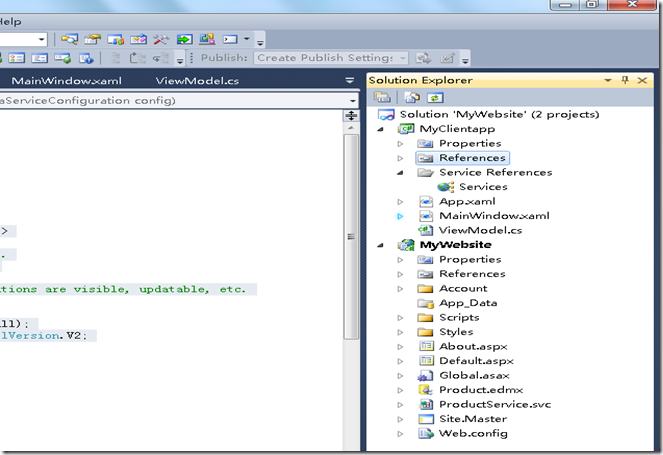
每个workspace可包含五种类型的集合：条目(entries)，类别(categories)，模板(templates)，用户(users)和集成资源(generic resources)。

既然一个workspace是一个产品，那麽产品一般都包含一系列的条目，产品等等这样的集合。所有这些集合都通过和HTTP谓词所指代的同样的方 式(GET,POSTD,ELETE,PUT)处理和作出响应。所有这些(集合)支持分页，因此，服务器可以将collections以一个易于处理的数 据块方式返回。同时还支持按日期查询(集合)，因此，您可以通过开始和结束日期过滤器来过滤collections。

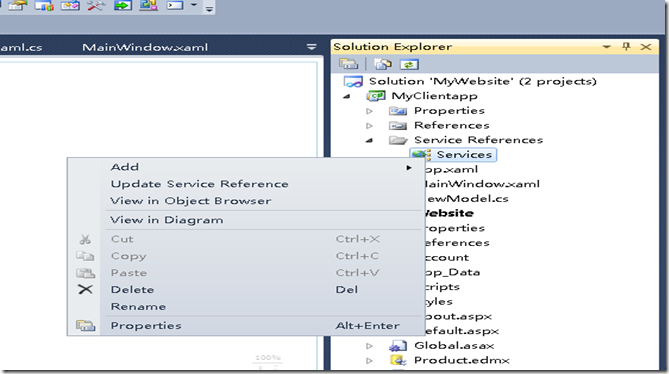
为了获得一个collection，可以向网址（就是服务文档collection的’ href ‘列出的地址） 发起一个GET请求。服务文档为每个collection都指定了一个URI。如果你向一个URI 发出GET请求，你会得到一个包含Atom Collection的XML文件，其中列出了前X个该collection中的成员。如果collection中的成员多于X个，那么该文件还将包含指 向下一批成员的URI，您可以使用它来获得下一批成员。您也可以在HTTP的头部通过Range来指定一日期范围，这样可以限制返回的 collection只包含那些在开始和结束日期之间的条目。

我们已经拥有了一个暴露V2 版本的OData Protocol的服务ProductService.svc,下面我创建一个WPF应用程序来消费这个服务。

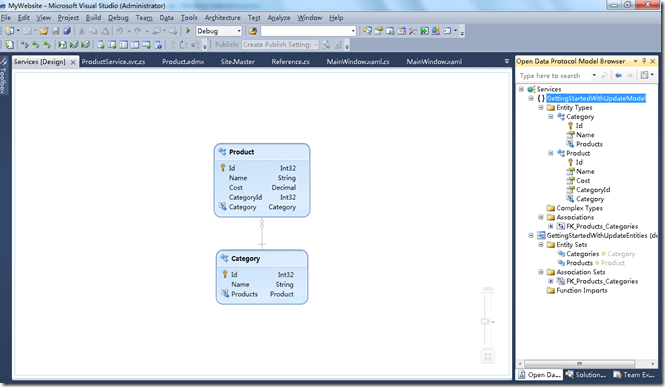
5、创建一个WPF应用程序，并添加ProductService.svc的服务引用。

[](http://images.cnblogs.com/cnblogs_com/shanyou/WindowsLiveWriter/WCFDataServiceQuickStart_D31F/image_6.png)

可以使用一个插件**Open Data Protocol Visualizer**查看服务返回的OData数据数据，这个工具的获取和安装可以参看[VS2010的扩展](http://www.cnblogs.com/shanyou/archive/2010/02/12/1667986.html" \t "_blank)。可以通过服务引用的“View in Diagram”进行查看。

[](http://images.cnblogs.com/cnblogs_com/shanyou/WindowsLiveWriter/WCFDataServiceQuickStart_D31F/image_8.png)

下图是ProductService.svc的的OData Model Brower：

[](http://images.cnblogs.com/cnblogs_com/shanyou/WindowsLiveWriter/WCFDataServiceQuickStart_D31F/image_10.png)

6、添加一个ViewModel，封装DataServiceContext，充当WPF表单和Data Service交互的中介。

public class ViewModel   
{   
       private GettingStartedWithUpdateEntities \_ctx;   
       private Category[] \_categories;   
       private DataServiceCollection<Product> \_products;

       public ViewModel()   
       {   
           \_ctx = new GettingStartedWithUpdateEntities(   
               new Uri("[http://localhost:1812/ProductService.svc/%22));](http://localhost:1812/ProductService.svc/%22%29%29;)   
           Load();   
       }

       public DataServiceCollection<Product> Products   
       {   
           get   
           {   
               return \_products;   
           }   
       }

       public Category[] Categories   
       {   
           get   
           {   
               return \_categories;   
           }   
       }

       public void SaveChanges()   
       {   
           \_ctx.SaveChanges();   
           Load();   
       }

       public void Load()   
       {   
           \_categories = \_ctx.Categories.ToArray();   
           \_products = new DataServiceCollection<Product>(\_ctx);   
           \_products.Load(from p in \_ctx.Products.Expand("Category")   
                          select p);   
       }   
   }

7、制作WPF表单

<Window x:Class="MyClientapp.MainWindow"   
        xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation%22>   
        xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml%22>   
        Title="Products Catalog" Height="400" Width="425">   
    <Grid>   
        <StackPanel Orientation="Horizontal">   
            <Grid Margin="0,0,0,0" Name="grid1" Width="140" >   
                <ListBox ItemsSource="{Binding Path=Products}"   
                         Name="Products"   
                         IsSynchronizedWithCurrentItem="True">   
                    <ListBox.ItemTemplate>   
                        <DataTemplate>   
                            <StackPanel Orientation="Horizontal">   
                                <TextBlock Text="{Binding Path=Name}" FontWeight="Bold"/>   
                            </StackPanel>   
                        </DataTemplate>   
                    </ListBox.ItemTemplate>   
                </ListBox>   
            </Grid>   
            <StackPanel Orientation="Vertical" Width="260">   
                <StackPanel Orientation="Horizontal">   
                    <Label Name="lblName" Width="100">   
                        <TextBlock Width="150">Name:</TextBlock>   
                    </Label>   
                    <TextBox Name="txtName"   
                             Text="{Binding ElementName=Products, Path=SelectedItem.Name, Mode=TwoWay}"   
                             Width="150"/>   
                </StackPanel>   
                <StackPanel Orientation="Horizontal">   
                    <Label Name="lblCost" Width="100">   
                        <TextBlock Width="150" >Cost:</TextBlock>   
                    </Label>   
                    <TextBox Name="txtCost"   
                             Text="{Binding ElementName=Products, Path=SelectedItem.Cost, Mode=TwoWay}"   
                             Width="150"/>   
                </StackPanel>   
                <StackPanel Orientation="Horizontal">   
                    <Label Name="lblCategory" Width="100">   
                        <TextBlock>Category:</TextBlock>   
                    </Label>   
                    <ComboBox   Name="cmbCategory"   
                                ItemsSource="{Binding Path=Categories}"   
                                DisplayMemberPath="Name"   
                                SelectedValuePath="."   
                                SelectedValue="{Binding ElementName=Products, Path=SelectedItem.Category, Mode=TwoWay}"   
                                Width="140" />   
                </StackPanel>   
                <Button Height="23"   
                        HorizontalAlignment="Right"   
                        Name="btnSaveChanges"   
                        VerticalAlignment="Bottom"   
                        Width="136"   
                        Click="btnSaveChanges\_Click">Save Changes</Button>   
            </StackPanel>   
        </StackPanel>   
    </Grid>   
</Window>

编写如下codebehind代码:

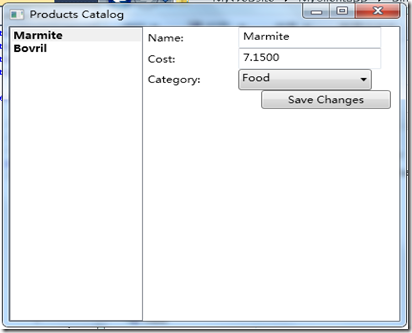
namespace MyClientapp   
{   
    /// <summary>   
    /// Interaction logic for MainWindow.xaml   
    /// </summary>   
    public partial class MainWindow : Window   
    {   
        ViewModel viewmodel = new ViewModel();

        public MainWindow()   
        {   
            InitializeComponent();   
            this.cmbCategory.DataContext = viewmodel;   
            this.grid1.DataContext = viewmodel;   
        }

        private void btnSaveChanges\_Click(object sender, RoutedEventArgs e)   
        {   
            viewmodel.SaveChanges();   
            this.grid1.DataContext = viewmodel;   
        }

    }   
}

运行程序结果如下：

[](http://images.cnblogs.com/cnblogs_com/shanyou/WindowsLiveWriter/WCFDataServiceQuickStart_D31F/image_12.png)

# [WCF Data Services 基础](http://www.cnblogs.com/2018/archive/2010/10/15/1852579.html)

把最近使用的WCF Data Service和WCF RIA Service的使用例子发布在站点[http://dskit.codeplex.com](http://dskit.codeplex.com/) ， 本系列文章就把WCF Data Service和WCF RIA Service涉及的主要方面描述一下，希望通过这些内容，可以比较顺畅的把这些框架使用起来，以提高开发的效率。

WCF Data Services中几个基础的概念

### Tenets of REST

* **HTTP method**

 Method information is kept in the HTTP header, not in the body of the request.

* **Resource based**

 Resources are expressed as “services” that clients consume.

* **Addressability**

The scoping information is kept in the URL.  Operations such as $filter $top $skip

* **Relationships**

How the data relates to each other is expressed in ADO.NET Data Services by resource links according to the CSDL [ specified in the Entity Data Model (EDM)]. A relationship link element such as <link href=”Customers(‘ALFKI’)/Orders

* **Statelessness**

No stateful messages are exchanged between resources

### MIME类型

承载的服务可以以JSON和ATOM返回【请求时说明MIME类型】用的返回类型如下：

|  |  |  |
| --- | --- | --- |
| MIME | MIME Type | 应用程序类型 |
| JSON | application/json | ASP.NET, AJAX, Silverlight |
| Atom | application/atom+xml | .NET Clients, Silverlight, ASP.NET mashups |

### HTTP Operation Success Return Codes

根据REST的定义对于不同的操作定义的服务端返回值情况：

Operation Success Code

GET 200 OK

POST 201 Created

PUT 204  No- Content

DELETE 204  No- Content

MERGE 204  No- Content

### 简单例子

VS2010中新建一个WEB工程，按如下步骤建立一个例子项目

建立微软SQL例子数据库NorthWind

1、  添加 Ado.net Entity: 选择SQL最初的例子NorthWind数据库

2、  添加WCF Data Services：WcfDataService.svc

3、  在WcfDataService.svc.cs配置访问规则：

            config.SetEntitySetAccessRule("\*", EntitySetRights.AllRead);

            config.SetServiceOperationAccessRule("\*", ServiceOperationRights.All);

            config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V2;

以上的部分是配置服务的可访问性和权限、行为等各种参数，具体参考MSDN，如果不对互联网公布的，默认配置也可

4、服务访问测试：Build解决方案后即可访问

下表是几个访问的实例

|  |
| --- |
| 发布的服务描述  http://localhost:1206/WcfDataService.svc/ |
| 多个实体实例  http://localhost:1206/WcfDataService.svc/Suppliers |
| 单个实体实例  http://localhost:1206/WcfDataService.svc/Suppliers(1) |
| 对象关系导航  http://localhost:1206/WcfDataService.svc/Suppliers(1)/Products |

【工程可从[http://dskit.codeplex.com](http://dskit.codeplex.com/)下载】

可以看到发布一个数据并进行查询等处理写的代码简直是很少了，而提供的功能却非常强大，后续对查询等继续进行说明。

# [WCF Data Services查询](http://www.cnblogs.com/2018/archive/2010/10/17/1853384.html)

上一篇文章<http://www.cnblogs.com/2018/archive/2010/10/15/1852579.html>介绍了WCF Data Service的基础知识，下面介绍一下查询

查询在WCF Data Services十分的方便，更强的功能在这些类别中可以看到WCF Data Service的超级威力。可以直接在浏览器中输入查询表达式等进行访问

直接使用例子可从<http://dskit.codeplex.com>下载

##### 查询表达式Query Expressions

###### 一般的查询表达式【返回实体内容】

1. **$filter**

2. **$inlinecount**

3. **$orderby**

4. **$skiptoken**

5. **$skip**

6. **$top**

7. **$select**

8. **$expand**

###### 不返回实体内容的查询表达式

· Service root path - returns information about the entity sets that can be accessed on the data service.

· Service operation - defines a method that is exposed as an endpoint on the data service. For more information, see [Service Operations (WCF Data Services)](http://127.0.0.1:47873/help/1-3044/ms.help?method=page&id=583A690A-E60F-4990-8991-D6EFCE069D76&product=VS&productVersion=100&topicVersion=100&locale=EN-US&topicLocale=EN-US&embedded=true).

· **$batch** path segment - defines the URI that is used to submit requests that contain more than one operation.

· **$count** path segment - returns the number of entities in the set returned by the URI, without any additional response message metadata.

· **$value** path segment - returns the value of a property that is a primitive type, without any additional response message metadata. Also used to access binary data from a media resource.

· **$metadata** path segment - returns metadata, in conceptual schema definition language (CSDL), for the Entity Framework that describes the data model used by the data service.

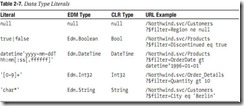
###### 语法和例子

[http://host/<service>/[/<EntitySet>][(<EntityKey>)][/<NavigationProperty>[(<EntityKey>)/](http://host/%3cservice%3e/%5b/%3cEntitySet%3e%5d%5b%28%3cEntityKey%3e%29%5d%5b/%3cNavigationProperty%3e%5b%28%3cEntityKey%3e%29/)...]]] [?$expand] & [?$filter] & [?$orderby] & [?&top] & [?$skip]

|  |
| --- |
| http://localhost:1206/WcfDataService.svc/Customers?$orderby=City desc |
|  |
| http://localhost:1206/WcfDataService.svc/Customers?$orderby=City desc&$skip=10 |
| http://localhost:1206/WcfDataService.svc/Customers?$skip=30&$top=10 |
|  |
| http://localhost:1206/WcfDataService.svc/Customers('ALFKI')?$expand=Orders |
| http://localhost:1206/WcfDataService.svc/Customers('ALFKI')?$expand=Orders/Order\_Details |
|  |
| http://localhost:1206/WcfDataService.svc/Customers?$filter=City eq 'london' |

以上这些在上面的那个WEB服务启动后都可以直接使用，加上这些表达式就可以完成比较复杂的查询了。

###### 查询表达式中的数据类型

[](http://images.cnblogs.com/cnblogs_com/2018/WindowsLiveWriter/WCFDataServices_8D1E/clip_image002_2.jpg)

##### Query Functions例子

[http://www.northwindtraders.com/service.svc/Customers?$filter=isof('Ns.SpecialCustomer')](http://www.northwindtraders.com/service.svc/Customers?$filter=isof%28%27Ns.SpecialCustomer%27%29)

更多的函数和例子参考MSDN

##### Query Operators例子

· Grouping operators: ()

· Logical operators: and or not eq ne lt gt le ge

· Arithmetic operators: add sub mul div mod

http://localhost:1206/WcfDataService.svc/Orders?$filter=OrderDate gt datetime'1997-01-01'

更多的约定和例子参考MSDN

分类: [Data Services](http://www.cnblogs.com/2018/category/262512.html)

# [WCF Data Services客户端访问](http://www.cnblogs.com/2018/archive/2010/10/18/1854847.html)

上一篇<http://www.cnblogs.com/2018/archive/2010/10/17/1853384.html>

讲述了查询的相关语法和例子，如果在程序中如何使用这些发布的服务呢？下面对在代码中访问这些服务的方法进行一下汇总

##### 客户端访问

|  |  |
| --- | --- |
| 查询 | 这些查询中可以结合上文的查询语法等使用  Ø 浏览器地址：输入地址，GET请求直接进行  Ø JavaScipt库：如ExtJS、DOJO、MS AJAX等支持JSON处理的JS库  Ø Service Reference引用  常用的形式，IDE直接添加引用，使用代理对象和上下文处理  //Generic泛型                      {                             DataServiceContext ctx = new DataServiceContext(u);                             var q = ctx.Execute<Order>(new Uri("/Orders(10402)", UriKind.Relative));                             foreach (var t in q)                             {                                    Console.WriteLine(t.OrderDate);                             }                      }    //Client Proxy代理                      {                             NorthwindEntities ctx = new NorthwindEntities(u);                             var q = from c in ctx.Suppliers select c;                             foreach (var t in q)                             {                                    Console.WriteLine(t.City);                             }  Ø HTTP协议支持处理         HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url + "/Orders(10402)");                             req.Method = "GET";                             req.Accept = "application/json";                             using (HttpWebResponse resp = (HttpWebResponse)req.GetResponse())                             {                                    using (StreamReader sr = new StreamReader(resp.GetResponseStream()))                                    {                                           Console.WriteLine(sr.ReadToEnd());                                    }                             }                               req = (HttpWebRequest)WebRequest.Create(url + "/Orders(10402)");                             req.Method = "GET";                             req.Accept = "application/atom+xml";                             using (HttpWebResponse resp = (HttpWebResponse)req.GetResponse())                             {                                    using (StreamReader sr = new StreamReader(resp.GetResponseStream()))                                    {                                           XmlDocument xml = new XmlDocument();                                           xml.Load(sr);                                             Console.WriteLine(xml.InnerXml);                                    }                             } |
| 增删改 | Ø HTTP形式  参考MSDN的规定传递参数和请求形式即可，处理有些麻烦  Ø Service Reference引用  比较常用的形式，具体参考“修改数据”一节  **需要处理并发冲突：**  DataServiceRequestException  · Entity Framework provider - In the data model, the **ConcurrencyMode** attribute of a property that is part of the concurrency token for an entity type is set to **Fixed**.  · Reflection provider - The [ETagAttribute](http://127.0.0.1:47873/help/1-884/ms.help?method=page&id=T%3ASYSTEM.DATA.SERVICES.ETAGATTRIBUTE&product=VS&productVersion=100&topicVersion=100&locale=EN-US&topicLocale=EN-US&embedded=true) is applied to the data class that is an entity type. This attribute declares the concurrency token based on the supplied property names. |

##### 修改数据

|  |  |
| --- | --- |
| Insert | HTTP POST  **Content-Type：需正确设置** |
| Update | HTTP Put/Merger |
| Delete | HTTP Delete |

###### 例子

需要修改数据对象，注意服务端需要设置正确的属性，如：

            config.SetEntitySetAccessRule("\*", EntitySetRights.All);

            config.SetServiceOperationAccessRule("\*", ServiceOperationRights.All);

            config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V2;

客户端通过代理访问：

//Client Proxy

{

NorthwindEntities ctx = new NorthwindEntities(u);

var q = from c in ctx.Suppliers select c;

foreach (var t in q)

{

Console.WriteLine(t.City);

}

///添加

var cust = new Customer()

{

CustomerID = "Test1",

Address = "Beijing",

City = "Peking",

CompanyName = "demo",

ContactName = "test",

ContactTitle = "mr.",

Country = "China",

Fax = "123",

Phone = "111",

PostalCode = "456",

Region = "HD"

};

ctx.AddToCustomers(cust);

ctx.SaveChanges();

///更新

var upCust = (from c in ctx.Customers where c.CustomerID == cust.CustomerID select c).FirstOrDefault();

upCust.ContactName += "-UPD";

ctx.UpdateObject(upCust);

ctx.SaveChanges();

///删除

upCust = (from c in ctx.Customers where c.CustomerID == cust.CustomerID select c).FirstOrDefault();

ctx.DeleteObject(upCust);

ctx.SaveChanges();

可见,在客户端可以使用LINQ语法进行查询

除了客户端，服务端也有很强的支持，下文再说。

# [WCF Data Services服务端处理汇总](http://www.cnblogs.com/2018/archive/2010/10/20/1856388.html)

和可以在客户端直接使用的查询对应，在服务端也有很多可以增强的功能

##### Service Operations

自己发布一些业务逻辑的处理

Service operations enable you to expose business logic in a data service, such as to implement validation logic, to apply role-based security, or to expose specialized querying capabilities.

具体要求如返回值、参数等参考MSDN，定义这些操作只要遵守这个要求即可

###### 例子

实现自定义的条件查询

· http://localhost:12345/Northwind.svc/GetOrdersByCity?city='London'

· http://localhost:12345/Northwind.svc/GetOrdersByCity?city='London'&$top=2

[WebGet]

public IQueryable<Order> GetOrdersByCity(string city)

{

    if (string.IsNullOrEmpty(city))

    {

        throw new ArgumentNullException("city",

            "You must provide a value for the parameter'city'.");

    }

    // Get the ObjectContext that is the data source for the service.

    NorthwindEntities context = this.CurrentDataSource;

    try

    {

        var selectedOrders = from order in context.Orders.Include("Order\_Details")

                             where order.Customer.City == city

                             select order;

         return selectedOrders;

    }

    catch (Exception ex)

    {

        throw new ApplicationException("An error occured: {0}", ex);

    }

}

##### Interceptors

拦截请求,这样可以加入自己的业务处理逻辑。目前系统预定义的有两个：

QueryInterceptor：由于限制查询的实体数据范围使用

ChangeInterceptor：修改实体时使用

WCF Data Services enables an application to intercept request messages so that you can add custom logic to an operation. You can use this custom logic to validate data in incoming messages. You can also use it to further restrict the scope of a query request, such as to insert a custom authorization policy on a per request basis.

Interception is performed by specially attributed methods in the data service. These methods are called by WCF Data Services at the appropriate point in message processing. Interceptors are defined on a per-entity set basis, and interceptor methods cannot accept parameters from the request like service operations can.

**Query interceptor** methods, which are called when processing an HTTP GET request, must return a lambda expression that determines whether an instance of the interceptor's entity set should be returned by the query results. This expression is used by the data service to further refine the requested operation.

###### 例子

[QueryInterceptor("Orders")]

public Expression<Func<Order, bool>> OnQueryOrders()

{

    // Filter the returned orders to only orders // that belong to a customer that is the current user.return o => o.Customer.ContactName ==

        HttpContext.Current.User.Identity.Name;

}

[ChangeInterceptor("Products")]

public void OnChangeProducts(Product product, UpdateOperations operations)

{

    if (operations == UpdateOperations.Add ||

       operations == UpdateOperations.Change)

    {

        // Reject changes to discontinued products.if (product.Discontinued)

        {

            throw new DataServiceException(400,

                        "A discontinued product cannot be modified");

        }

    }

    else if (operations == UpdateOperations.Delete)

    {

        // Block the delete and instead set the Discontinued flag.

        throw new DataServiceException(400,

            "Products cannot be deleted; instead set the Discontinued flag to 'true'");

    }

}

##### Data Service Provider

应用中我们一般使用的是Entity Framework provider[edmx]，从数据库生成模型或则从模型建立数据库都可以

由于Provider模型的强大功能，当然我们也可以自己实现一个Provider，如下例，是一个类实现的实例：

namespace DataServices

{

       [DataServiceKey("Name")]

       public class Inductee

       {

              public string Name { get; set; }

              public bool Group { get; set; }

              public int YearInducted { get; set; }

              public List<Song> Songs { get; set; }

              public static List<Inductee> MakeInducteeList()

              {

                    return (new List<Inductee>()

              {

                    new Inductee()

                    {

                           Name = "Rolling Stones",

                           Group = false,

                           YearInducted = 1990,

                           Songs = new List<Song>()

                    },

                    new Inductee()

                    {

                           Name = "Beatles",

                           Group = false,

                           YearInducted = 1986,

                           Songs = new List<Song>()

                    }

              });

              }

       }

       [DataServiceKey("SongTitle")]

       public class Song

       {

              public string SongTitle { get; set; }

              public static List<Song> MakeSongList()

              {

                    return (new List<Song>()

              {

                    new Song(){SongTitle="Satisfaction"},

                    new Song(){SongTitle="All you need is love"},

              });

              }

       }

       public class AssignInducteesToSongs

       {

              public static void Assign(List<Inductee> inductee, List<Song> songs)

              {

                    inductee[0].Songs.Add(songs[0]);

                    inductee[1].Songs.Add(songs[1]);

              }

       }

       /// <summary>

       /// Summary description for MyDataModel

       /// </summary>

       public class MyDataModel

       {

              static List<Inductee> inductees;

              static List<Song> songs;

              static MyDataModel()

              {

                    inductees = Inductee.MakeInducteeList();

                    songs = Song.MakeSongList();

                    AssignInducteesToSongs.Assign(inductees, songs);

              }

              public IQueryable<Inductee> Inductees

              {

                    get

                    {

                           return inductees.AsQueryable();

                    }

              }

              public IQueryable<Song> Songs

              {

                    get

                    {

                           return songs.AsQueryable();

                    }

              }

       }

//服务

       [System.ServiceModel.ServiceBehavior(IncludeExceptionDetailInFaults = true)]

       public class DemoOb : DataService<MyDataModel>

       {

              public static void InitializeService(DataServiceConfiguration config)

              {

                    config.SetEntitySetAccessRule("\*", EntitySetRights.AllRead);

                     config.SetServiceOperationAccessRule("\*", ServiceOperationRights.All);

                   config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V2;

              }

       }

这个服务发布后，就可以使用“WCF Data Services查询”的方法和客户端查询进行使用了。

该例子可以在<http://dskit.codeplex.com> 下载

##### Hosting

一般的服务承载在IIS中，以svc扩展名有IIS的HTTPHandler进行处理就行了

不过有时可能需要自己承载服务，此时就使用WCF的技术就行了

Using System.ServiceModel.Web;

WebServiceHost host = new WebServiceHost(typeof(SampleDataService));

host.Open();

其他的关于WCF的ABC只要在app.config中设置就行了

# [WCF Data Service安全分析和说明](http://www.cnblogs.com/2018/archive/2010/10/22/1858070.html)

首先需要知道服务运行在asp.net的承载环境中，具体参考：

<http://blogs.msdn.com/b/wenlong/archive/2006/01/23/516041.aspx>

关于安全有个详细的参考地址：

<http://blogs.msdn.com/b/astoriateam/archive/tags/authentication/>

讲述了涉及安全各个方面，是个很好的资料

本文就从常见的涉及安全的几个方面进行分析和说明

##### 服务端安全

由于服务承载在IIS中，因此可以使用Asp.net的安全模型，如一般的Forms验证

对于服务一般的处理是，把服务放到一个独立目录下如Services，使用Services\web.config设置目录的安全性，如下授予admins角色和d用户以访问权限：

<?xml version="1.0"?>

<configuration>

       <system.web>

              <authorization>

                    <allow roles="admins"/>

                    <allow users="d"/>

                    <deny users="\*"/>

              </authorization>

       </system.web>

</configuration>

以上设置后，匿名用户就不可以访问了

##### 服务对象的安全

DataService<T>对象的设置，仅授予可以察看的数据对象以相应的权限，这样客户端就不会看到其他的对象，增删改仅在必须的时候开放，或者把增

删改放到一个独立的不同权限的服务上

下例是设置一些限制的情况，如果不想让客户端看到更多的信息，可以把”\*”条件移除，这样就只能看到设置的对象了

              public static void InitializeService(DataServiceConfiguration config)

              {

                                         config.SetEntitySetAccessRule("Orders", EntitySetRights.AllRead);

                     config.SetServiceOperationAccessRule("\*", ServiceOperationRights.AllRead);

     config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V2;

              }

##### 默认查询的限制

对于授予了权限的实体，也可以根据角色或用户限定不同的范围[如果不限制的话，如果不设置的话，默认可以访问到整个表内容

下例设置后http://localhost:1287/Services/NorthWindService.svc/Orders 查询就会根据用户情况返回不同的值，这样可以减少了数据暴露的范围

       [QueryInterceptor("Orders")]

              public Expression<Func<Order, bool>> OrderFilter()

              {

                    if (Roles.IsUserInRole("admins"))

                           return (Order od) => true;

                    else

                           return (Order od) => od.ShipName.StartsWith("H");

              }

当然由于当前在WEB环境中，也可以直接使用HttpContext.Current.User或者HttpContext.Current其他对象完成用户的限制等安全处理

##### 自定义查询

这个根据查询条件自然就限制住了，定义linq查询条件时一定要把数据范围限制作

##### 设置安全后客户端的访问

Ø 网页形式：浏览器中访问网站时，会按照asp.net的安全模型进行处理，导航到登陆页面，输入用户名、密码，然后可以访问服务

Ø 一般的应用程序：如Console程序、WinForm程序的处理方式有些差别，最简单的步骤如下：

1、服务端开放Authentication\_JSON\_AppService.axd服务，只要web.config增加以下配置节后，客户端即可访问

       <system.web.extensions>

              <scripting>

                    <webServices>

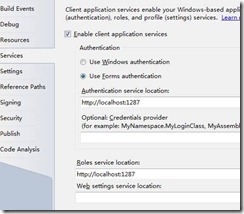
                           <authenticationService enabled="true" requireSSL="false"/>

                    </webServices>

              </scripting>

       </system.web.extensions>

2、客户端设置”Client Application Service”[客户端需要引用完整的.net框架，而不是Profile框架]，再引用System.Web.dll

[](http://images.cnblogs.com/cnblogs_com/2018/WindowsLiveWriter/WCFDataService_9494/clip_image002_2.jpg)

如上图，URL设置到服务的根地址

具体的代码：

验证用户

System.Web.Security.Membership.ValidateUser("d", "demo1234!");

对Data Service的客户端代理OnContextCreated方法增加处理，把登陆的信息设置上

namespace TestDS.North

{

public partial class NorthwindEntities

{

partial void OnContextCreated()

{

this.SendingRequest +=

new EventHandler<SendingRequestEventArgs>(OnSendingRequest);

}

void OnSendingRequest(object sender, SendingRequestEventArgs e)

{

ClientFormsIdentity id = Thread.CurrentPrincipal.Identity as ClientFormsIdentity;

if(id!=null)

{

((HttpWebRequest)e.Request).CookieContainer = id.AuthenticationCookies;

}

}

}

}

以上验证后就可正常的访问服务了

NorthwindEntities ctx = new NorthwindEntities(u);

var ods = from c in ctx.Orders select c;

foreach (var o in ods)

{

Console.WriteLine(o.ShipName);

}

# How to: Add, Modify, and Delete Entities (WCF Data Services)

**.NET Framework 4.5**

[Other Versions](javascript:;)

http://i3.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

0 out of 2 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/dd756368.aspx#feedback)

With the WCF Data Services client libraries, you can create, update, and delete entity data in a data service by performing equivalent actions on objects in the [DataServiceContext](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.aspx). For more information, see [Updating the Data Service (WCF Data Services)](http://msdn.microsoft.com/en-us/library/dd756361.aspx).

The example in this topic uses the Northwind sample data service and autogenerated client data service classes. This service and the client data classes are created when you complete the [WCF Data Services quickstart](http://msdn.microsoft.com/en-us/library/cc668796.aspx).

## Example

The following example creates a new object instance and then calls the [AddObject](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.addobject.aspx) method on the [DataServiceContext](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.aspx) to create the item in the context. An HTTP POST message is sent to the data service when the [SaveChanges](http://msdn.microsoft.com/en-us/library/cc646716.aspx) method is called.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd756368.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

// Create the DataServiceContext using the service URI.

NorthwindEntities context = new NorthwindEntities(svcUri);

// Create the new product.

Product newProduct =

Product.CreateProduct(0, "White Tea - loose", false);

// Set property values.

newProduct.QuantityPerUnit = "120gm bags";

newProduct.ReorderLevel = 5;

newProduct.UnitPrice = 5.2M;

try

{

// Add the new product to the Products entity set.

context.AddToProducts(newProduct);

// Send the insert to the data service.

DataServiceResponse response = context.SaveChanges();

// Enumerate the returned responses.

foreach (ChangeOperationResponse change in response)

{

// Get the descriptor for the entity.

EntityDescriptor descriptor = change.Descriptor as EntityDescriptor;

if (descriptor != null)

{

Product addedProduct = descriptor.Entity as Product;

if (addedProduct != null)

{

Console.WriteLine("New product added with ID {0}.",

addedProduct.ProductID);

}

}

}

}

catch (DataServiceRequestException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}

The following example retrieves and modifies an existing object and then calls the [UpdateObject](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.updateobject.aspx) method on the [DataServiceContext](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.aspx) to mark the item in the context as updated. An HTTP MERGE message is sent to the data service when the [SaveChanges](http://msdn.microsoft.com/en-us/library/cc646716.aspx) method is called.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd756368.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

string customerId = "ALFKI";

// Create the DataServiceContext using the service URI.

NorthwindEntities context = new NorthwindEntities(svcUri);

// Get a customer to modify using the supplied ID.

var customerToChange = (from customer in context.Customers

where customer.CustomerID == customerId

select customer).Single();

// Change some property values.

customerToChange.CompanyName = "Alfreds Futterkiste";

customerToChange.ContactName = "Maria Anders";

customerToChange.ContactTitle = "Sales Representative";

try

{

// Mark the customer as updated.

context.UpdateObject(customerToChange);

// Send the update to the data service.

context.SaveChanges();

}

catch (DataServiceRequestException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}

The following example calls the [DeleteObject](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.deleteobject.aspx) method on the [DataServiceContext](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.aspx) to mark the item in the context as deleted. An HTTP DELETE message is sent to the data service when the [SaveChanges](http://msdn.microsoft.com/en-us/library/cc646716.aspx) method is called.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd756368.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

// Create the DataServiceContext using the service URI.

NorthwindEntities context = new NorthwindEntities(svcUri);

try

{

// Get the product to delete, by product ID.

var deletedProduct = (from product in context.Products

where product.ProductID == productID

select product).Single();

// Mark the product for deletion.

context.DeleteObject(deletedProduct);

// Send the delete to the data service.

context.SaveChanges();

}

// Handle the error that occurs when the delete operation fails,

// which can happen when there are entities with existing

// relationships to the product being deleted.

catch (DataServiceRequestException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}

The following example creates a new object instance and then calls the [AddRelatedObject](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.addrelatedobject.aspx) method on the [DataServiceContext](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.aspx) to create the item in the context along with the link to the related order. An HTTP POST message is sent to the data service when the [SaveChanges](http://msdn.microsoft.com/en-us/library/cc646716.aspx) method is called.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd756368.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

int productId = 25;

string customerId = "ALFKI";

Order\_Detail newItem = null;

// Create the DataServiceContext using the service URI.

NorthwindEntities context = new NorthwindEntities(svcUri);

try

{

// Get the specific product.

var selectedProduct = (from product in context.Products

where product.ProductID == productId

select product).Single();

// Get the specific customer.

var cust = (from customer in context.Customers.Expand("Orders")

where customer.CustomerID == customerId

select customer).Single();

// Get the first order.

Order order = cust.Orders.FirstOrDefault();

// Create a new order detail for the specific product.

newItem = Order\_Detail.CreateOrder\_Detail(

order.OrderID, selectedProduct.ProductID, 10, 5, 0);

// Add the new item with a link to the related order.

context.AddRelatedObject(order, "Order\_Details", newItem);

// Add the new order detail to the collection, and

// set the reference to the product.

order.Order\_Details.Add(newItem);

newItem.Order = order;

newItem.Product = selectedProduct;

// Send the changes to the data service.

DataServiceResponse response = context.SaveChanges();

// Enumerate the returned responses.

foreach (ChangeOperationResponse change in response)

{

// Get the descriptor for the entity.

EntityDescriptor descriptor = change.Descriptor as EntityDescriptor;

if (descriptor != null)

{

if (descriptor.Entity.GetType() == typeof(Order\_Detail))

{

Order\_Detail addedItem = descriptor.Entity as Order\_Detail;

if (addedItem != null)

{

Console.WriteLine("New {0} item added to order {1}.",

addedItem.Product.ProductName, addedItem.OrderID.ToString());

}

}

}

}

}

catch (DataServiceQueryException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}

// Handle any errors that may occur during insert, such as

// a constraint violation.

catch (DataServiceRequestException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}

# How to: Define Entity Relationships (WCF Data Services)

**.NET Framework 4.5**

[Other Versions](javascript:;)

http://i3.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

This topic has not yet been rated - [Rate this topic](http://msdn.microsoft.com/en-us/library/dd758799.aspx#feedback)

When you add a new entity in WCF Data Services, any relationships between the new entity and related entities are not automatically defined. You can create and change relationships between entity instances and have the client library reflect those changes in the data service. For more information, see [Updating the Data Service (WCF Data Services)](http://msdn.microsoft.com/en-us/library/dd756361.aspx).

The example in this topic uses the Northwind sample data service and autogenerated client data service classes. This service and the client data classes are created when you complete the [WCF Data Services quickstart](http://msdn.microsoft.com/en-us/library/cc668796.aspx).

## Example

The following example creates a new object instance and then calls the [AddRelatedObject](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.addrelatedobject.aspx) method on the [DataServiceContext](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.aspx) to create the item in the context along with the link to the related order. An HTTP POST message is sent to the data service when the [SaveChanges](http://msdn.microsoft.com/en-us/library/cc646716.aspx) method is called.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd758799.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

int productId = 25;

string customerId = "ALFKI";

Order\_Detail newItem = null;

// Create the DataServiceContext using the service URI.

NorthwindEntities context = new NorthwindEntities(svcUri);

try

{

// Get the specific product.

var selectedProduct = (from product in context.Products

where product.ProductID == productId

select product).Single();

// Get the specific customer.

var cust = (from customer in context.Customers.Expand("Orders")

where customer.CustomerID == customerId

select customer).Single();

// Get the first order.

Order order = cust.Orders.FirstOrDefault();

// Create a new order detail for the specific product.

newItem = Order\_Detail.CreateOrder\_Detail(

order.OrderID, selectedProduct.ProductID, 10, 5, 0);

// Add the new item with a link to the related order.

context.AddRelatedObject(order, "Order\_Details", newItem);

// Add the new order detail to the collection, and

// set the reference to the product.

order.Order\_Details.Add(newItem);

newItem.Order = order;

newItem.Product = selectedProduct;

// Send the changes to the data service.

DataServiceResponse response = context.SaveChanges();

// Enumerate the returned responses.

foreach (ChangeOperationResponse change in response)

{

// Get the descriptor for the entity.

EntityDescriptor descriptor = change.Descriptor as EntityDescriptor;

if (descriptor != null)

{

if (descriptor.Entity.GetType() == typeof(Order\_Detail))

{

Order\_Detail addedItem = descriptor.Entity as Order\_Detail;

if (addedItem != null)

{

Console.WriteLine("New {0} item added to order {1}.",

addedItem.Product.ProductName, addedItem.OrderID.ToString());

}

}

}

}

}

catch (DataServiceQueryException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}

// Handle any errors that may occur during insert, such as

// a constraint violation.

catch (DataServiceRequestException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}

The following example shows how to use the [AddObject](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.addobject.aspx) method to add an Order\_Details object to a related Orders object with a reference to a specific Products object. The [AddLink](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.addlink.aspx) and [SetLink](http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.setlink.aspx) methods define the relationships. In this example, the navigation properties on the Order\_Details object are also explicitly set.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd758799.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

int productId = 25;

string customerId = "ALFKI";

Order\_Detail newItem = null;

// Create the DataServiceContext using the service URI.

NorthwindEntities context = new NorthwindEntities(svcUri);

try

{

// Get the specific product.

var selectedProduct = (from product in context.Products

where product.ProductID == productId

select product).Single();

// Get the specific customer.

var cust = (from customer in context.Customers.Expand("Orders")

where customer.CustomerID == customerId

select customer).Single();

// Get the first order.

Order order = cust.Orders.FirstOrDefault();

// Create a new order detail for the specific product.

newItem = Order\_Detail.CreateOrder\_Detail(

order.OrderID, selectedProduct.ProductID, 10, 5, 0);

// Add the new order detail to the context.

context.AddToOrder\_Details(newItem);

// Add links for the one-to-many relationships.

context.AddLink(order, "Order\_Details", newItem);

context.AddLink(selectedProduct, "Order\_Details", newItem);

// Add the new order detail to the collection, and

// set the reference to the product.

order.Order\_Details.Add(newItem);

newItem.Product = selectedProduct;

// Send the changes to the data service.

DataServiceResponse response = context.SaveChanges();

// Enumerate the returned responses.

foreach (ChangeOperationResponse change in response)

{

// Get the descriptor for the entity.

EntityDescriptor descriptor = change.Descriptor as EntityDescriptor;

if (descriptor != null)

{

if (descriptor.Entity.GetType() == typeof(Order\_Detail))

{

Order\_Detail addedItem = descriptor.Entity as Order\_Detail;

if (addedItem != null)

{

Console.WriteLine("New {0} item added to order {1}.",

addedItem.Product.ProductName, addedItem.OrderID.ToString());

}

}

}

}

}

catch (DataServiceQueryException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}

// Handle any errors that may occur during insert, such as

// a constraint violation.

catch (DataServiceRequestException ex)

{

throw new ApplicationException(

"An error occurred when saving changes.", ex);

}