[**Using ADO.NET Data Service - 服务端**](http://www.cnblogs.com/zlgcool/archive/2008/10/15/1312231.html)

Using ADO.NET Data Service

ADO.NET Data Service是随同Visual Studio 2008 SP1提供的用于构建在数据对象模型 (如EF-DEMX, LINQ-DBML) 之时来快速提供企业网内外的轻量级数据服务。ADO.NET Data Service Framework的目标是提供一个天生与web相结合的灵活的数据服务。它其中最醒目的特点就是通过URIs来指向以特定方式(JSON/ATOM)展现的数据(块)，并以REST方式来展现。所支持的URIs访问方式通过对标准HTTP verbs如GET, POST, PUT和DELETE的支持来完成对数据的CRUD.

本文会通过一个具体的例子来演示如何使用ADO.NET Data Service以及其部分特性。

         **选择数据源**

ADO.NET Data Service是由两部分组成的，一部分是其运行时，它提供了诸如Addressing(即URI Translation), Open Format for Data Representation(公开的数据表现形式-JSON/ATOM), 数据传输协议等。另一部分就是数据访问部分，这部分是可以选择的，LINQ或者Entity Framework都是可以的，Data Service本身的功能就是将这些数据源暴露成一个面向web的数据服务。

创建一个服务分为两步：

         用ADO.NET Entity Framework来创建一个Entity Data Model – 这是为了以EDM来展现你的数据架构

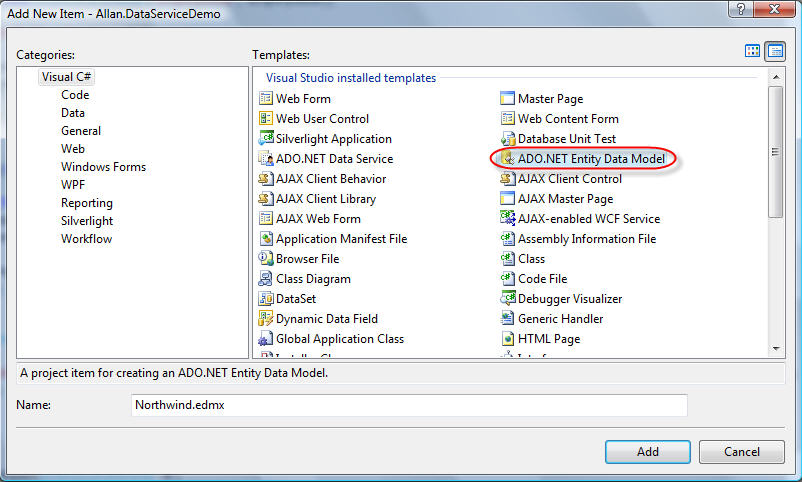
         建立数据服务

         配置用户对数据访问的权限

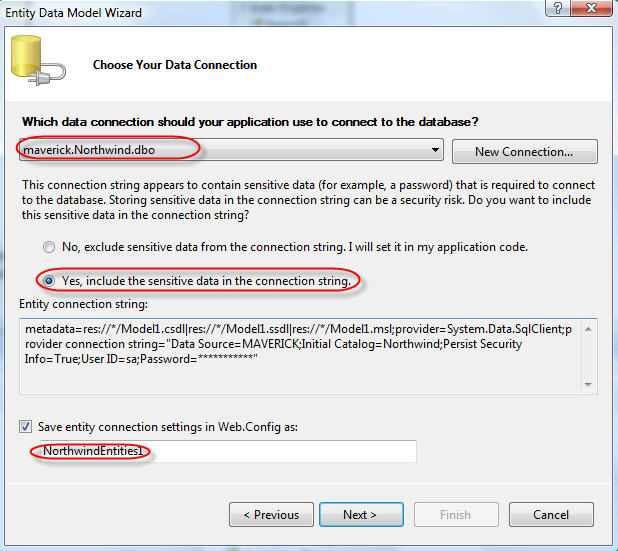
如何创建一个EDM我们在这里不想多讲，下边的图和简单步骤会告诉你如何来创建一个EDM。

1.       创建一个Web Application,这里我们命名为Allan.DataServiceDemo

2.       在Solution Explore中右键点击选择添加新项，选择添加ADO.NET Entity Data Model，在Name中输入Northwind.edmx.



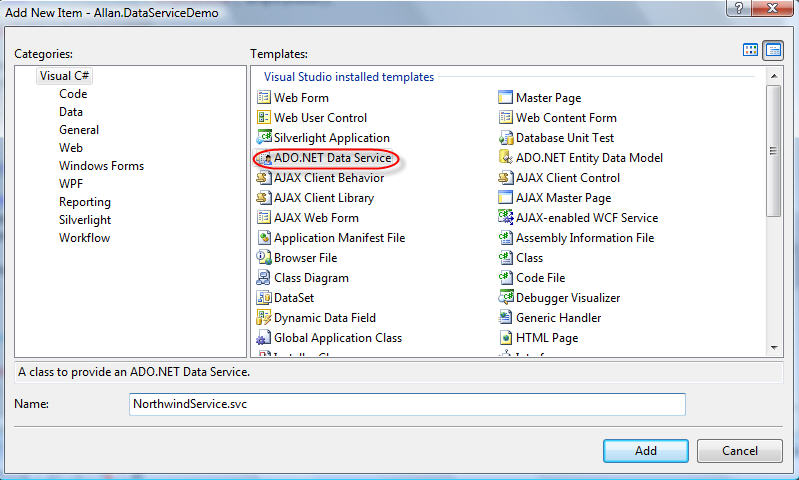
3.       在Entity Data Model Wizard中设置ConnectionString.点击下一步.



4.       在接下来的选择表、视图的窗体中选择你想要加入到EDM中的表。这里我们选择Employees, Territories和EmployeesTerrories表，然后点Finish. 到此我们的EDM已经完成了。接下来我们要做的事在EDM的基础上来构建Data Service.

         **创建数据服务**

5.       右键点击项目并在弹出菜单中选择添加新项目。在弹出的对话框中选择 ADO.NET Data Service, 并将名称更名为NorthwindService.svc.通过这里你会发现，其实ADO.NET Data Service本质上是一个WCF.

6.       Service本身已经建立，但现在还没有将数据暴露给客户端，并且没有相应的数据访问控制。当点击上一步中的Add按钮后，系统会打开NorthwindService.svc.cs文件，在这里我们需要做如下更改来暴露数据。

a.       将数据访问层类设立为被暴露的服务的数据源

b.      设定对相关数据实体集的权限和可访问性。

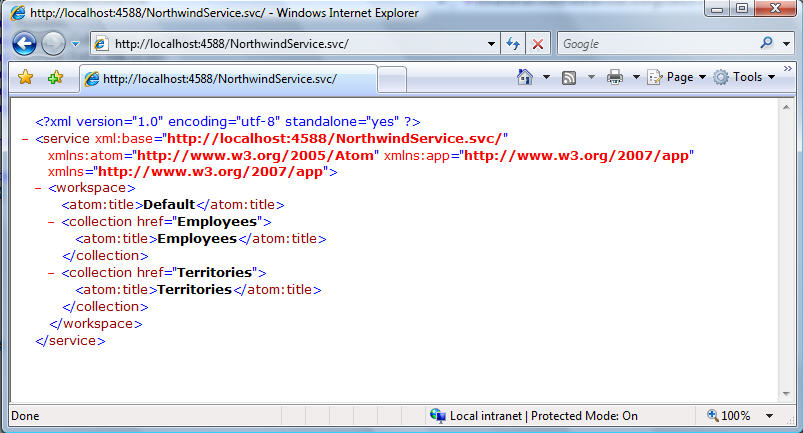
要做到这两点只需要更改一下所示的高两部分即可。第一个高两部分表示我们将NorthwindEntities作为我们的数据源。NorthwindEntities是我们刚才创建的EDMX的类名，Employees等表都已经作为其属性被Mapping成对象。第二个高亮部分是控制EntitySet(其实就是对应的表一级)的访问权限。例如你仍然可以通过下边的代码仅仅暴露Employees对象的只读权限：config.SetEntityAccessRule(“Employees”,EntitySetRights.AllRead.这样，服务只会暴露Employees集合并且只接受读取，而不能有更新操作。

|  |
| --- |
| public class NorthwindService : DataService<NorthwindEntities>  {  // This method is called only once to initialize service-wide policies.  public static void InitializeService(IDataServiceConfiguration config)  {  // TODO: set rules to indicate which entity sets and service operations are visible, updatable, etc.  // Examples:  config.SetEntitySetAccessRule("\*", EntitySetRights.All);  // config.SetServiceOperationAccessRule("MyServiceOperation", ServiceOperationRights.All);  }  } |

大功告成，你可以通过View In Browser来验证你的工作是否正确。

        **Addressing: Accessing Data Service via URIs**

在前边提到过，Data Service通过URIs来指向a piece of data。也就是说我们通过URL就可以来做到访问数据，并且具有一定的数据过滤功能。抛弃高级的查询选项不说，基本的URL就可以来访问EntitySet和Entity，以至于某个字段，字段的值等等。通过浏览你的Northwind.svc，我们可以看到以下数：



这里暴露了Employees和Territories两个数据集，这和我们之前设定的是相同的。通过以下的访问列表，你可以很容易的来在这个数据集合中导航，并获取你想要的数据。

|  |  |
| --- | --- |
| **Resource** | **URI** |
| Service | http://localhost:4588/NorthwindService.svc/ |
| Entity Set | http://localhost:4588/NorthwindService.svc/Employees |
| Entity | http://localhost:4588/NorthwindService.svc/Employees(1) |
| Relationship | http://localhost:4588/NorthwindService.svc/Employees(1)/Address |
| Property | http://localhost:4588/NorthwindService.svc/Employees(1)/Address/City |
| Property | http://localhost:4588/NorthwindService.svc/Employees(1)/FirstName |

ADO.NET Data Service同样提供了以下的查询选项：

|  |  |  |
| --- | --- | --- |
| **Option** | **Description** | **Example** |
| expand | 类似于LINQ中的LoadOptions，以此来指定加载此对象相关的通过expand指定的对象。如果需要加载多个对象，用逗号分隔。 | [http://localhost:4588/NorthwindService.svc/Employees(1)?$expand=Territories](http://localhost:4588/NorthwindService.svc/Employees%281%29?$expand=Territories)  同事加载当前employee对应的Territories. |
| orderby | 指定排序方式。语法为：$orderby=Field [ASC|DESC], [Field[ASC|DESC]] | <http://localhost:4588/NorthwindService.svc/Employees?$orderby=FirstName> |
| Skip  Top | 类似于LINQ中的Skip(PageSize \* PageIndex).Take(PageSize)用来实现分页。表示越过多少条记录，取接下来开始的几条记录。也可分开始使用两个选项。 | <http://localhost:4588/NorthwindService.svc/Employees?$orderby=FirstName&$skip=3&$top=3>  http://localhost:4588/NorthwindService.svc/Employees? $skip=3 |
| Filter | 通过filter这个参数可以在URL里传递过滤条件。 | [http://localhost:4588/NorthwindService.svc/Employees?$orderby=FirstName&filter=substringof(FirstName,'A')](http://localhost:4588/NorthwindService.svc/Employees?$orderby=FirstName&filter=substringof%28FirstName,%27A%27%29) |

对于如上边所示的查询表达式，ADO.NET Data Service提供了一些函数和标记来帮助我们组建适合的查询表达式(Query Expression).下表总结了常见的组建Query Expression的标记、函数等，以供查询和参阅。

1.       Logical Operators – 逻辑运算符

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Operator** | **Description** |
| eq | Equal | ne | Not Equal |
| gt | Greater than | ge | Greater than or equal |
| lt | Less than | le | Less than or equal |
| and | Logical and | or | Logical or |
| not | Logical negation | () | Precedence grouping |

2.       Arithmetic Operators – 算数运算符

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| add | Addition | /Product?filter=UnitPrice add 5 gt 10 |
| sub | Subtraction | /Product?filter=UnitPrice sub 5 gt 10 |
| mul | Multiplication | /Orders?$filter=Freight mul 800 gt 2000 |
| div | Division | /Orders?$filter=Freight div 10 eq 4 |
| mod | Modulo | /Orders?$filter=Freight mod 10 eq 0 |

3.       Functions – 函数

|  |  |  |  |
| --- | --- | --- | --- |
| **String Functions** | | | |
| bool substringof(string p0, string p1) | | bool endswith(string p0, string p1) | |
| bool startswith(string p0, string p1) | | int length(string p0) | |
| int indexof(string arg) | | string insert(string p0,int pos, string p1) | |
| string remove(string p0, int pos) | | string remove(string p0, int pos, int length) | |
| string remove(string p0, string find, string replace) | | string substring(string p0, int pos) | |
| string substring(string p0, int pos, int length) | | string tolower(string p0) | |
| string toupper(string p0) | | string trim(string p0) | |
| string concat(string p0, string p1) | |  | |
| **Date Functions** | | | |
| int day(DateTime) | int hour(DateTime) | | int minute(DateTime) |
| int month(DateTime) | int second(DateTime) | | int year(DateTime) |
| **Math Functions** | | | |
| double round(double) | decimal round(decimal) | | double floor(double) |
| decimal floor(decimal) | double ceiling(double) | | decimal ceiling(decimal) |
|  |  |  |  |

通过这些看似简单的函数和逻辑/算数操作符，可以在很大程度上通过URL来做一些简单的查询和过滤。通过分析查询结果你会清楚的看到：

         当有多个对象被返回是，在atom模式下，数据以feed的方式返回。

         每个entry包含一个对象。

         每个entry会提供link标签来给出自己的URL和其关联对象的URL

         属性会被包含在Content内部以application/xml表示

         每个属性都是content内一个子节点。

         所有的数据和关系均以纯XML表现

         默认情况下关联对象不被加载（expand可以加载）

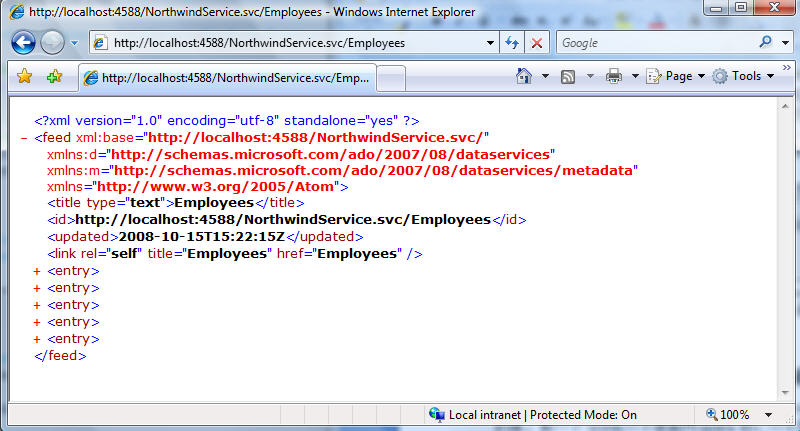
        **Query Interceptors – 查询拦截**

查询拦截非常有用，它起个代理或者把关的作用。当你仅仅想把具有某种状态或者特征的数据返回给客户端时，用拦截查询就可以实现。比如，数据库中的Employee信息，对于已经离开公司的员工我们会将其状态设置为Leave，而在我们不打算将这些信息流露出去时，写一个Query Interceptor仅仅返回状态<>Leave的记录，那么所有客户端请求与Employee有关的数据时将被强制只能得到状态不为Leave的记录。换句话说，Query Interceptor的代理函数返回的数据在表面上替换了Employees的记录。

下边的示例限定了客户端仅能得到国家是USA的Employees信息：

|  |
| --- |
| [QueryInterceptor("Employees")]  public Expression<Func<Employees, bool>> ReturnUSAEmployees()  {  return e => e.Country == "USA";  } |

Query Interceptors是面向EntitySet(也就是Table)的，通过修改QueryInterceptorAttribute的参数，你可以给任何一个EntitySet来创建一个查询拦截器。看看我们刚才设置的查询拦截器是否工作呢？

没错，仅仅返回了5条Country=USA的记录。Nice!

        **Change Interceptors – 数据更新拦截**

看起来，Change Interceptors和Query Interceptors很相似，都是用来拦截客户端操作的。不同的是，前一个word,一个是面向query的，一个是面向change的。顾名思义，Change Interceptors面向的就是提交到服务器的的数据更新操作：包括Add, Change 和Delete，分别对应在数据添加操作，数据更改操作和数据删除操作—这些是由UpdateOperations参数来提供的。下边的示例实现了在执行添加操作时，如果Employee实例的Country属性为空，则默认设置为USA:

|  |
| --- |
| [ChangeInterceptor("Employees")]  public void AssignDefaultCountry(Employees c, UpdateOperations operation)  {  if (operation == UpdateOperations.Add)  {  if (c.Country.Trim().Length == 0)  {  c.Country = "USA";  }  }  } |

不要忘记给这个方法加入ChangeInterceptor属性。而对于所拦截的操作对象，同样是通过更改ChangeInterceptorAttribute的参数来实现的。

        **扩展ADO.NET Data Service**

ADO.NET Data Service 为我们提供了快速构建一个数据服务的平台，但到目前为止，除了数据库中的对象（还不包括存储过程），我们仍然无法加入自己的方法，这显得好像很不灵活。当然，ADO.NET Data Service还是为我们提供了这样的一种方式来扩展已有的整套生成好的数据服务的。你可以添加一些你自己的方法，例如操作存储过程，自己的逻辑，返回特殊处理后的数据这样的方法，然后通过添加WebGetAttribute来使其变成一个ServiceOperation来暴露给客户。

|  |
| --- |
| [WebGet]  public IQueryable<Employees> GetEmployeesByCountry(string country)  {  var employees = from c in this.CurrentDataSource.Employees  where c.Country == country  select c;    return employees;  } |

同时还要设置这个方法的权限（InitializeService方法中）：

|  |
| --- |
| config.SetServiceOperationAccessRule("GetEmployeesByCountry", ServiceOperationRights.All); |

如何去访问这个方法呢？因为你无法再meta中（实际上就是http://localhost:4588/NorthwindService.svc）找到对这个方法的描述。很简单，直接试验一下下边的URL是否工作呢?

|  |
| --- |
| http://localhost:4588/NorthwindService.svc/GetEmployeesByCountry?Country='USA' |

服务端的设置好像该有的都有了哦？除了Data Service的表现方式：Atom还是JSON，这个另行解决吧，起码得有传输的数据包来展现。另外的有关客户端消费的(服务端消费和客户端AJAX消费)我们在下一个话题中来介绍吧:)

点击链接下载练习代码： <http://files.cnblogs.com/zlgcool/Allan.DataServiceDemo.rar>