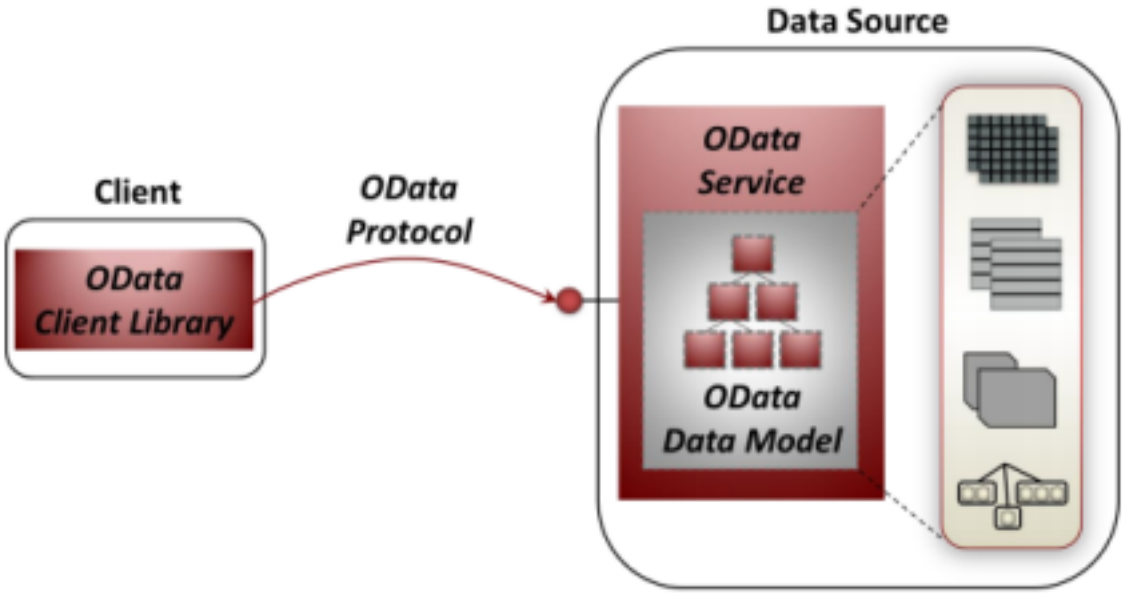
عنوان: **مروری بر Open Data Protocol و کاربردهای آن**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۰/۰۳ ۱۶:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: WCF RIA Services, OData, WCF Data Services

مقدمه

OData

قراردادی برای دسترسی به داده‌ها است که مایکروسافت آن را تحت مجوز Microsoft Open Specification Promise منتشر کرده است. این قرارداد استاندارد CRUD ایی را برای دسترسی به منبع داده از طریق وب سایت طراحی نموده است که از JDBC و ODBC ساده‌تر بوده و محدودیت ارتباط فقط با پایگاه داده‌های SQL ایی را ندارد.
OData از روی Atom Publishing Protocol و JSON ساخته شده و از مدل REST برای همه در خواست‌های خود استفاده می‌نماید.
OData در واقع یک راه مشترک برای هر نوع کلاینت برای دسترسی به هر نوع داده ای است.



OData چهار قسمت اصلی دارد :

OData data model که یک راه عمومی برای مدیریت و توصیف داده‌ها را فراهم می‌نماید
OData protocol که به کلاینت اجازه ایجاد درخواست و پاسخ از سرویس دهنده OData را می‌دهد.
OData client libraries که امکان ساخت ساده‌تر نرم افزارها برای دسترسی به داده‌ها با قرارداد OData را می‌دهد.
OData service سرویس دهنده و امکان دسترسی به داده‌ها را فراهم می‌سازد.

از مزیت‌های OData می توان به موارد زیر اشاره نمود:

ساده و انعطاف پذیر

سورس باز بودن

امکان استفاده در سیستم‌های با داده‌های رابطه ای و غیر رابطه ای

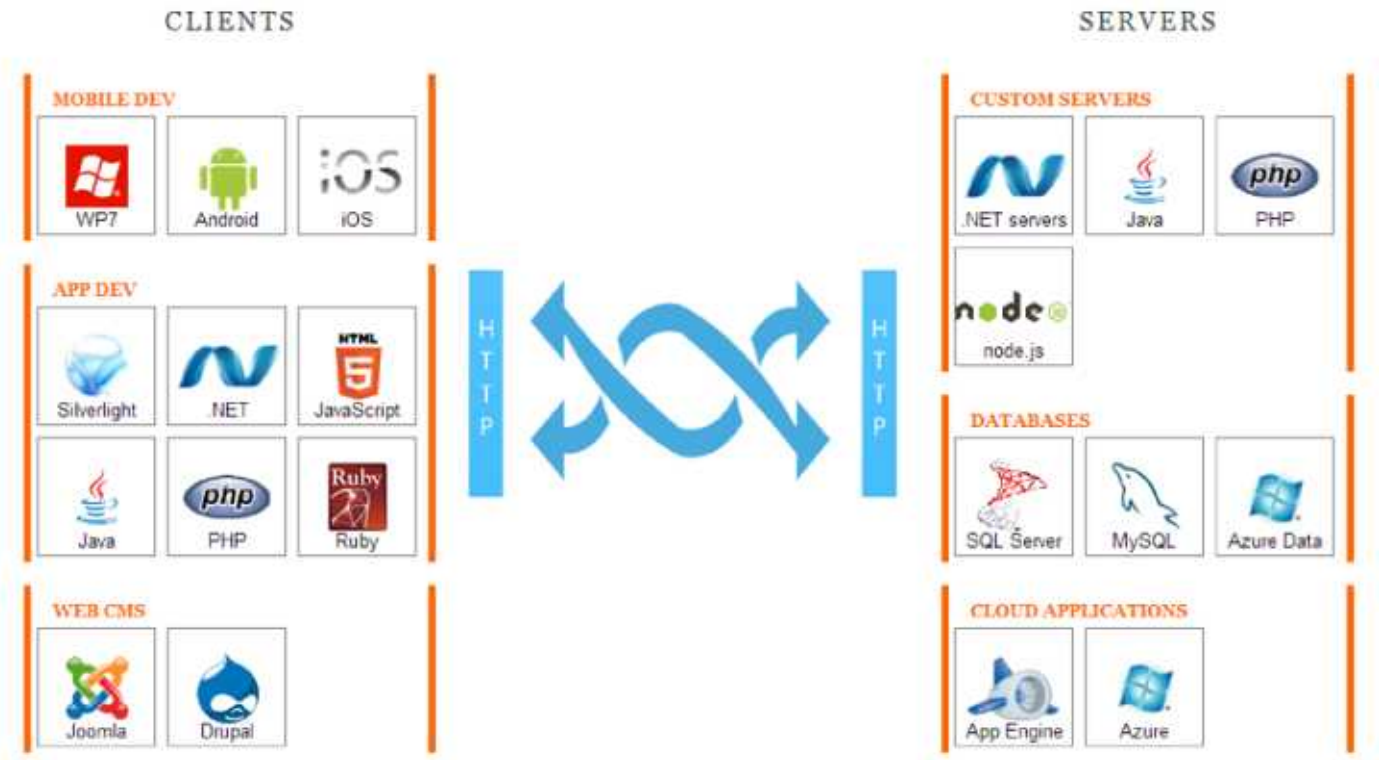
امکان استفاده از داده‌ها با منابع ای که آدرس پذیر هستند یعنی دسترسی از طریق Url

امکان دسترسی هر نوع گیرنده ای به داده ها

امکان نمایش خروجی با فرمت Json یا Xml

...

کتابخانه‌های کار بار OData:
کتابخانه‌های بسیاری برای odata نوشته شده است که امکان استفاده آن را در اکثر زبان‌ها مهیا می‌سازد.



اما بهترین کتابخانه WCF Data Services است که از سوی مایکروسافت ارائه شده و در اکثر تکنولوژی‌ها و محصولات خود قابلیت استفاده را دارد. WCF Data Services با پیاده سازی قرارداد OData ، توسعه دهندگان را از سطح پایین این قرارداد رهایی ساخته و به راحتی می‌توانند از ساختار شی گرا برنامه خود، در سرویس دهی با OData استفاده نمایند.

کاربردهای OData:
OData یک قرارداد سرویس دهی بر روی وب است که به هر نوع گیرنده که امکان دسترسی به وب را داشته باشد، امکان سرویس دهی دارد. به همین خاطر در اکثر برنامه‌های تحت وب یا نرم افزارهای موبایل که می‌خواهیم اطلاعاتی را مابین سرویس دهنده و گیرنده ردوبدل کنیم حتی زمانیکه platformهای مختلفی در کار باشند OData بهترین گزینه است.
در مطالب بعدی با پیاده سازی مثال‌های با استفاده از WCF Data Services بیشتر با OData آشنا خواهید شد. در این اینجا هدف آشنایی اولیه با Odata و کاربردهای آن بود که امیدوارم مفید واقع شده باشد.

نظرات خوانندگان

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۱۰/۰۳ ۲۱:۱۶

ممنون، لطفا ادامه بدید ...

نویسنده: پرهام
تاریخ: ۱۳۹۱/۱۰/۰۴ ۷:۵۷

ممنون. منتظر مطالب بعدی هستیم.

نویسنده: MegaMan
تاریخ: ۱۳۹۱/۱۰/۱۷ ۱۵:۴۷

جالب بود والبته مفید :)

عنوان: **آشنایی و استفاده از WCF Data Services در Visualstudio 2012**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۰/۱۳ ۱۷:۰
آدرس: www.dotnettips.info
گروه‌ها: WPF, Web Service, OData, WCF Data Services

مقدمه:

WCF Data Services جزئی از NET Framework . است که امکان ایجاد سرویس دهنده‌های با قرارداد OData را به روی وب یا Intranet با استفاده از REST مهیا می‌سازد. OData از داده هایی که با Url آدرس پذیر هستند استفاده می‌نماید. دسترسی و تغییر داده‌ها با استفاده از استاندارد HTTP و کلمات GET، PUT، POST و DELETE صورت می‌پذیرد. برای اینکه درک بهتری داشته باشید به یک مثال می‌پردازیم.

ایجاد یک برنامه سرویس دهنده WCF Data Service در VisualStudio 2012

یک ASP.NET Web Application با نام NorthwindService ایجاد نمایید و بر روی پروژه راست کلیک کنید و از منوی Add گزینه New Item را انتخاب نمایید از پنجره باز شده از دسته Data گزینه ADO.NET Entity Data Model را انتخاب و نام ان را Northwind بگذارید.

از پنجره باز شده Generate from Database را انتخاب و با انتخاب کانکشن از نوع Sql Server Compact 4 اتصال به فایل Northwind.sdf را انتخاب تا کلاس‌های لازم تولید شود.

برای تولید data service بر روی پروژه راست کلیک کنید و از منوی Add گزینه New Item را انتخاب نمایید از پنجره باز شده گزینه WCF Data Service را انتخاب و نام آن را Northwind.svc بگذارید. کد زیر خودکار تولید می‌شود

```
public class Northwind : DataService< /* TODO: put your data source class name here */ >
    {
        // 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("MyEntityset", EntitySetRights.AllRead);
            // config.SetServiceOperationAccessRule("MyServiceOperation", ServiceOperationRights.All);
            config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V3;
        }
    }
```

برای دسترسی به موجودیت‌های Northwind بجای عبارت put your data source نام مدل را تایپ کنید

```
public class Northwind : DataService<NorthwindEntities>
```

برای فعال کردن دسترسی به منابع data source متغیر config کلاس DataServiceConfiguration را بصورت زیر تنظیم نمایید. تابع SetEntitySetAccessRule با گرفتن نام موجودیت و نحوه دسترسی امکان استفاده از این موجودیت را با استفاده از WCF Data Service فراهم می‌نمایید. مثلا در زیر امکان دسترسی به موجودیت Orders را با امکان خواندن همه، نوشتن ادقامی و جایگزین فراهم نموده است.

```
config.SetEntitySetAccessRule("Orders", EntitySetRights.AllRead
      | EntitySetRights.WriteMerge
      | EntitySetRights.WriteReplace );
config.SetEntitySetAccessRule("Customers", EntitySetRights.AllRead);
```

اگر بخواهیم امکان خواندن همه موجودیت‌ها را فراهم کنیم از کد زیر می‌توانیم استفاده نمایید که * به معنای همه موجودیت‌های data model می‌باشد

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

دسترسی به WCF Data Service بوسیله مرورگر وب

برای دسترسی به وب سرویس برنامه را اجرا نمایید تا آدرس http://localhost:8358/Northwind.svc مشخصات وب سرویس را نمایش دهد

```
<service xmlns="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom"
xml:base="http://localhost:8358/Northwind.svc/">
<workspace>
<atom:title>Default</atom:title>
<collection href="Categories">
<atom:title>Categories</atom:title>
</collection>
<collection href="Customers">
<atom:title>Customers</atom:title>
</collection>
<collection href="Employees">
<atom:title>Employees</atom:title>
</collection>
<collection href="Order_Details">
<atom:title>Order_Details</atom:title>
</collection>
<collection href="Orders">
<atom:title>Orders</atom:title>
</collection>
<collection href="Products">
<atom:title>Products</atom:title>
</collection>
<collection href="Shippers">
<atom:title>Shippers</atom:title>
</collection>
<collection href="Suppliers">
<atom:title>Suppliers</atom:title>
</collection>
</workspace>
</service>
```

حال اگر آدرس را به http://localhost:8358/Northwind.svc/Products وارد نمایید لیست کالاها بصورت xml Atom قابل دسترس می‌باشد.

ایجاد یک برنامه گیرنده WCF Data Service در Visual Studio 2012

بر روی Solution پروژه جاری راست کلیک و از منوی Add گزینه New Project را انتخاب و یک پروژه از نوع WPF Application با نام NorthwindClient ایجاد نمایید.

در پنجره MainWindow مانند کد زیر از یک Combobox و DataGrid برای نمایش اطلاعات استفاده نمایید

```
<Window x:Class="MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Northwind Orders" Height="335" Width="425"
        Name="OrdersWindow" Loaded="Window1_Loaded">
    <Grid Name="orderItemsGrid">
        <ComboBox DisplayMemberPath="Order_ID" ItemsSource="{Binding}"
                  IsSynchronizedWithCurrentItem="true"
                  Height="23" Margin="92,12,198,0" Name="comboBoxOrder" VerticalAlignment="Top"/>
        <DataGrid ItemsSource="{Binding Path=Order_Details}"
                  CanUserAddRows="False" CanUserDeleteRows="False"
                  Name="orderItemsDataGrid" Margin="34,46,34,50"
                  AutoGenerateColumns="False">
            <DataGrid.Columns>
                <DataGridTextColumn  Header="Product" Binding="{Binding Product_ID, Mode=OneWay}" />
                <DataGridTextColumn  Header="Quantity" Binding="{Binding Quantity, Mode=TwoWay}" />
                <DataGridTextColumn  Header="Price" Binding="{Binding UnitPrice, Mode=TwoWay}" />
                <DataGridTextColumn  Header="Discount" Binding="{Binding Discount, Mode=TwoWay}" />
            </DataGrid.Columns>
        </DataGrid>
        <Label Height="28" Margin="34,12,0,0" Name="orderLabel" VerticalAlignment="Top"
               HorizontalAlignment="Left" Width="65">Order:</Label>
        <StackPanel Name="Buttons" Orientation="Horizontal" HorizontalAlignment="Right"
                    Height="40" Margin="0,257,22,0">
```

```
            <Button Height="23" HorizontalAlignment="Right" Margin="0,0,12,12"
                Name="buttonSave" VerticalAlignment="Bottom" Width="75"
                    Click="buttonSaveChanges_Click">Save Changes
            </Button>
            <Button Height="23" Margin="0,0,12,12"
                Name="buttonClose" VerticalAlignment="Bottom" Width="75"
                    Click="buttonClose_Click">Close</Button>
        </StackPanel>
    </Grid>
</Window>
```

برای ارجاع به wcf data service بر روی پروژه راست کلیک و گزینه Add Service Reference را انتخاب نمایید در پنجره باز شده گزینه Discover را انتخاب تا سرویس را یافته و نام Namespase را Northwind بگذارید.
حال مانند کد زیر یک شی از مدل NorthwindEntities با آدرس وب سرویس ایجاد نموده ایم و نتیجه کوئری با استفاده از کلاس DataServiceCollection به DataContext گرید انتصاب داده ایم که البته پیش فرض آن آشنایی با DataBinding در WPF است.

```
private NorthwindEntities context;
        private string customerId = "ALFKI";
        private Uri svcUri = new Uri("http://localhost:8358/Northwind.svc");

        private void Window1_Loaded(object sender, RoutedEventArgs e)
        {
            try
            {
                context = new NorthwindEntities(svcUri);
                var ordersQuery = from o in context.Orders.Expand("Order_Details")
                                  where o.Customers.Customer_ID == customerId
                                  select o;
                DataServiceCollection<Orders> customerOrders = new
DataServiceCollection<Orders>(ordersQuery);
                this.orderItemsGrid.DataContext = customerOrders;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }
```

با صدا زدن تابع SaveChanges مدل می‌توانید تغییرات را در پایگاه داده ذخیره نمایید.

```
private void buttonSaveChanges_Click(object sender, RoutedEventArgs e)
{
    try
    {
        context.SaveChanges();
    }
    catch (DataServiceRequestException ex)
    {
        MessageBox.Show(ex.ToString());

    }
}
```

برنامه را اجرا نمایید تا خروجی کار را مشاهده نمایید. مقادیر Quantity را تغییر دهید و دکمه Save Changes را انتخاب تا تغییرات دخیره شود.
در اینجا در یک برنامه ویندوزی استفاده از WCF Data Service را تست نمودیم اما براحتی به همین شیوه در یک برنامه وب نیز قابل استفاده است.

نظرات خوانندگان

نویسنده: حامد
تاریخ: ۱۳۹۱/۱۰/۱۵ ۲۳:۳

با درود
لطفاً درباره‌ی امنیت استفاده از این روش هم توضیحاتی بنویسید.
مخصوصاً امنیت از نوع Message، آیا می‌توان این نوع امنیت را در کنار WCF Data Service داشت؟

با سپاس

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۴:۶

سلام
در صورتی که بخوایم سطح دسترسی به داده بر اساس کاربران داشته باشیم، چطور باید انجام بدیم؟ چون اینجا همه چیز انگاری خودکار و بدون واسطه انجام میشه

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۴:۵۱

Securing WCF Data Services

OData and Authentication – Part 7 – Forms Authentication

عنوان: **قراردادهای کوئری نویسی در OData و WCF Data Service - قسمت اول**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۰/۱۸ ۲۳:۲۰
آدرس: www.dotnettips.info
برچسب‌ها: OData, WCF Data Services

قبل از اینکه با کاربردهای OData بیشتر آشنا شوید می‌بایست قراردادهای کوئری نویسی با استفاده از آدرس وب سرویس را فراگیرید. در سمت گیرنده WCF Data Service زمانی که شما یک آدرس وب سرویس را به پروژه خود اضافه می‌نمایید مدل‌ها و روابط موجودیت‌ها بصورت خودکار تولید شده و دیگر لازم نیست از کوئری نویسی با آدرس وب سرویس استفاده نمایید و به جای آن از LINQ براحتی می‌توانید داده‌های خود را جستجو نمایید.اما اگر بخواهید وب سرویس را در بسترهای دیگر یا در سمت گیرنده وب استفاده نمایید آشنایی با کوئری نویسی به شما امکان جستجو و دسترسی به داده‌های مدنظرتان را می‌دهد گرچه کتابخانه‌های OData موجود این امکان را آسان‌تر می‌سازد.

اجزای OData Url

OData Url شامل سه جزء می‌باشد که ترکیب این سه جرء امکان کوئری نویسی را فراهم می‌سازد
Service Root Url یا ریشه آدرس سرویس که آدرس و نام وب سرویس را مشخص می‌سازد.
Resource Path یا مسیر منابع که امکان دسترسی به منابع موجود وب سرویس را فراهم می‌سازد.
Query Options یا گزینه‌های کوئری که امکان کوئری نویسی با استفاده از عملگرها و پارامترهای گرامر OData را فراهم می‌سازد.

در زیر این اجزا بهتر نشان داده شده است:

```
http://services.odata.org/OData/OData.svc/Category(1)/Products?$top=2&$orderby=name
\_____________________________________/ \_________________/ \_________________/
                 |                             |                    |
         service root URL                resource path        query options
```

مسیر منابع در OData

برای دسترسی به موجودیت‌های سرویس در قسمت مسیر منابع با نوشتن نام آن موجودیت این امکان فراهم می‌شود مثلا برای دسترسی به موجودیت Prodacts بعد از نام وب سرویس از یک / استفاده می‌نامیم که Backslash برای جدا کردن موجودیت‌ها استفاده می‌شود

```
http://services.odata.org/OData/OData.svc/Products
```

بروزرسانی :حال اگر بخواهیم به Supplierهای Product با استفاده از کلید موجودیت به یک product دسترسی یابیم. (بعد Id و بعد) برای product و دوباره / و Supplier را می‌نویسیم

```
http://services.odata.org/OData/OData.svc/Products(1)/Supplier
```

حتی امکان دسترسی در توابع موجودیت‌ها نیز میسر است

```
http://services.odata.org/OData/OData.svc/Products/MostExpensive
```

اگر بخواهیم پارامتری را مقدار دهی نماییم بصورت زیر عمل می‌کنیم

```
http://services.odata.org/OData/OData.svc/GetProductsByCategoryId(categoryId=2)
```

برای دسترسی به property های موجودیت‌ها بصورت زیر عمل می‌کنیم

```
http://services.odata.org/OData/OData.svc/Products(1)/Name
```

گزینه‌های کوئری

OData پنج عملگر دارد که امکان دستکاری موجودیت‌ها را فراهم می‌سازد

$filter عناصر برگشتی را محدود می‌سازد

$orderby امکان مرتب سازی عناصر برگشتی را فراهم می‌سازد

$skip امکان گذشت از تعدادی عناصر را از ابتدای عناصر فراهم می‌سازد

$top تعداد عناصر برگشتی را محدود می‌سازد

$expand امکان برگشت محتوای وابسته به عناصر برگشتی را فراهم می‌سازد

در زیر مثال‌های از این گزینه‌ها آورده شده است

```
//filter
http://services.odata.org/OData/OData.svc/Products?$filter=Name eq 'Milk'
//orderby
http://services.odata.org/OData/OData.svc/Products?$orderby=Name
//skip
http://services.odata.org/OData/OData.svc/Products?$skip=5
//top
http://services.odata.org/OData/OData.svc/Products?$top=10
//expand
http://services.odata.org/OData/OData.svc/Products?$expand=Category
```

ادامه دارد...

نظرات خوانندگان

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۰/۱۹ ۹:۱۱

حتی الامکان Categoryها را به همراه Product هایشان برگردانید، نه بالعکس، به علت ساختار XML، این کوئری سایز کمتری دارد،
البته من تست نکردم و صرفا حدس می‌زنم، فرصت کنم چک اش می‌کنم
فیلتر چند شرطی را نیز قرار دهید
بد نیست کلا در مزیت Application Serverها نیز چند خطی توضیح دهید.

موفق و پایدار باشید

نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۰/۱۹ ۱۰:۱۴

در این قسمت هدف آشنایی اولیه با کوئری نویسی و اجزای آن بود. در اینجا کارایی و سرعت عمل کوئری‌ها مطرح نیست. البته این بسته به نیاز و جایگاه آن کوئری مورد نظر بر می‌گردد، خیلی جاها به categoryهای productها احتیاجی نیست.

عنوان: **قراردادهای کوئری نویسی در OData و WCF Data Service - قسمت دوم**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۰/۲۵ ۱۶:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: WCF RIA Services, OData, WCF Data Services

در مطلب قبلی قراردادهای کوئری نویسی در OData و WCF Data Service - قسمت اول با قراردادهای کوئری نویسی آشنا شدید در این مطلب به جزئیات بیشتر این قراردادها می‌پردازیم.

عمگرهای منطقی

در OData نه عملگر منطقی داریم که امکان مقایسه منطقی در عبارات‌های شرطی را در اختیارمان قرار می‌دهد.

eq عملگر برابری
ne عملگر مخالف
lt عملگر کوچکتری
le عملگر کوچکتر یا مساوی
gt عملگر بزرگنری
ge عملگر بزرگتر یا مساوی
and
or
not

عملگرهای ریاضی

پنج عمگر ریاضی وجود دارد که امکان انجام عملیات ریاضی در کوئری را میسر می‌سازد.

add جمع دو عملوند
sub تفریق دو عملوند
mul ضرب دو عملوند
div تقیسم دو عملوند
mod باقیمانده تقسیم دو عملوند

در آخر () برای گروه بنده و اولویت دهی عبارات کاربرد دارد.

توابع عبارت‌های کوئری نویسی

در OData چهار دسته توابع داریم
String Functions در جدول زیر این توابع با توضیح آن آورده شده است:

آیا رشته p0 شامل رشته p1 هست؟ مثال: http://services.odata.org//Northwind.svc/Customers?$filter=substringof('Alfreds', CompanyName) eq true شرح: مشتریانی که نام شرکت آنها شامل رشته Alfreds باشد	(bool substringof(string p0, string p1
آیا رشته p0 با رشته p1 پایان می‌یابد؟ مثال: http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=endswith(CompanyName, 'Futterkiste') شرح: مشتریانی که نام شرکت آنها با رشته FutterKiste پایان می‌یابد	(bool endswith(string p0, string p1
آیا رشته p0 با رشته p1 آغاز می‌شود؟ مثال: http://services.odata.org/Northwind.svc/Customers?$	(boolstartswith(string p0, string p1

آیا رشته p0 شامل رشته p1 هست؟ مثال:http://services.odata.org//Northwind.svc/Customers? $filter=substringof('Alfreds', CompanyName) eq true شرح: مشتریانی که نام شرکت آنها شامل رشته Alfreds باشد	(bool substringof(string p0, string p1
('$filter=startswith(CompanyName, 'Alfr شرح: مشتریانی که نام شرکت آنها با رشته Alfer آغاز می‌یابد	
محاسبه طول رشته دریافتی. مثال:http://services.odata.org/Northwind/Northwind.svc/C ustomers?$filter=length(CompanyName) eq 19 شرح: مشتریانی که طول نام شرکت آنها برابر 19 باشد	int length(string p0)
برگشت اندیس رشته وروری مثال:http://services.odata.org/Northwind.svc/Customers? $filter=indexof(CompanyName, 'lfreds') eq 1 شرح: مشتریانی که نام شرکت آنها با رشته lfreds که از کارکتر دوم شروع می‌یابد	int indexof(string arg)
جایگزینی یک رشته در رشته دیگر مثال:http://services.odata.org/Northwind.svc/Customers? $filter=replace(CompanyName, ' ', '') eq ''AlfredsFutterkiste شرح: مشتریانی که نام شرکت آنها بدون فاصله برابر AlfredsFutterkiste باشد با پرکردن فاصله با جای خالی	(string replace(string p0, string find, string replace
برگرداندن رشته ای از رشته دیگر از شماره اندیس ورودی یا از شماره اندیس تا طول رشته ورودی مثال:http://services.odata.org/Northwind.svc/Customers? $filter=substring(CompanyName, 1) eq 'lfreds Futterkiste شرح: مشتریانی که نام شرکت آنها از کاراکتر دوم برابر lfreds Futterkiste باشد http://services.odata.org/Northwind.svc/Customers?$fil' ter=substring(CompanyName, 1,2) eq 'lf	(string substring(string p0, int pos (string substring(string p0, int pos, int length
برگرداندن رشته ورودی با کارکتر بزرگ برگرداندن رشته ورودی با کاراکتر کوچک	string tolower(string p0) string toupper(string p0)
حذف کارکترهای Whitespace از دو طرف رشته	string trim(string p0)
الحاق رشته به هم	(string concat(string p0, string p1

Date Functions

برگرداندن سال تاریخ ورودی مثال:http://services.odata.org/Northwind.svc/Employees?$filter=year(BirthDate) eq 1971 ن	int year(DateTimep0)
برگرداندن ماه تاریخ ورودی مثال:http://services.odata.org/Northwind/Northwind.svc/Employees?$filter=month(BirthDate) eq 5	int month(DateTimep0)
برگرداندن روز تاریخ ورودی برگرداندن تعداد روز فاصله زمانی مثال: http://services.odata.org/Northwind.svc/Employees?$filter=day(BirthDate) eq 8	int days(DateTimep0) int day(DateTimep0)
برگرداندن ساعت تاریخ ورودی مثال:http://services.odata.org/Northwind.svc/Employees?$filter=hour(BirthDate) eq 4	int hour(DateTimep0)
برگرداندن دقیقه تاریخ ورودی مثال:http://services.odata.org/Northwind.svc/Employees?$filter=minute(BirthDate) eq 40	int minute(DateTimep0) int minutes(DateTimep0)
برگرداندن ثانیه تاریخ ورودی مثال:http://services.odata.org/Northwind.svc/Employees?$filter=second(BirthDate) eq 40	int second(DateTimep0)

Math Functions

برگرداندن سقف عدد ورودی مثال:http://services.odata.org/Northwind.svc/Orders?$filter=ceiling(Freight) eq 32	double ceiling(double p0) decimal ceiling(decimal p0)
برگرداندن کف عدد ورودی مثال:http://services.odata.org/Northwind.svc/Orders?$filter=floor(Freight) eq 32	double floor(double p0) decimal floor(decimal p0)
گردن کردن عدد ورودی مثال:ttp://services.odata.org/Northwind.svc/Orders?$filter=round(Freight) eq 32	double round(double p0) decimal round(decimal p0)

Type Functions

برگرداندن نوع داده وروری مثال:http://services.odata.org/Northwind.svc/Orders?$filter=isof(Customer, NorthwindModel.MVPCustomer) شرح: سفارشاتی که نوع مشتری آنها برابر MVPCustomer باشد	boolisof(type p0) boolisof(expression p0, type p1)
تبدیل نوع داده ورودی	cast(type p0> p0) cast(expression p0, type p1 > p1)

در آخر چند نکته

برای استفاده از رشته‌ها در عبارات از ' تک کوتشن استفاده نمایید

برای دستیابی به مقادیر پروپرتی‌ها از $value استفاده نمایید

برای دستیابی به آدرس روابطهای یک موجودیت از $links استفاده نمایید

مثال:http://services.odata.org/OData/OData.svc/Categories(1)/$links/Products

$select برای محدود کردن پروپرتی‌های یک موجودیت استفاده می‌شود

مثال:http://services.odata.org/OData/OData.svc/Products?$select=Price,Name

از ستاره برای دستیابی به همه پروپرتی‌های یک موجودیت می‌توان استفاده نمود

مثال:http://services.odata.org/OData/OData.svc/Products?$select=*

ادامه دارد...

عنوان: **استفاده از WCF Data Services در برنامه های وب به کمک JQuery**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۷:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: Web Service, jQuery, OData, WCF Data Services, JSON

در مطالب قبلی با پروتکل OData و WCF Data Service و قراردادهای کوئری نویسی آن آشنا شدید. حال می‌خواهیم با استفاده از Jquery به داده‌های وب سرویس WCF Data Service دسترسی یابیم. اما پیش نیازهای لازم است

پیش نیاز اول : دسترسی به خروجی Json وب سرویس WCF Data

خروجی پیش فرش وب سرویس WCF Data Services ساختار Xml دارد پس می‌بایست وب سرویس را متوجه سازیم که ما با خروجی Json نیاز داریم. از نسخه 5 به بعد اگر MaxProtocolVersion را بر روی V3 قرار دهیم دیگر با Accept Header برابر application/json کار نخواهد کرد و می‌بایست از application/json;odata=verbose استفاده نمود یا نسخه پروتکل را بر روی V2 یا پایین‌تر تنظیم کنید. علاوه بر آن کتابخانه‌های و قطعه کدهای تهیه شده است که با پارامتر $format این کار را برای ما انجام می‌دهد در زیر آدرس دو نمونه آورده شده است.

DataServicesJSONP

WCF Data Services Toolkit

قطعه کد اول یک Attribute است که با اضافه کردن آن به بالای کلاس WebService و استفاده از پارامتر $format=json در آدرس وب سرویس این کار را برای ما انجام می‌دهد.

```
[JSONPSupportBehavior]
public class Northwind : DataService<NorthwindEntities>
```

و نمونه آدرس

```
http://localhost:8358/Northwind.svc/Products?$format=json
```

دومی کتابخانه ای است که مانند روش اول عمل می‌کند اما به جای ارث برای از کلاس DataService می‌بایست از کلاس ODataService استفاده نماییم.
نکته: در صورتی که بخواهیم از نسخه V3 استفاده نماییم Accept Header را باید به application/json;odata=verbose تغییر دهیم

```
public class Northwind : ODataService<NorthwindEntities>
```

استفاده از WCF Data Services به کمک Jquery

تابع getJSON مخصوص درخواست‌های است خروجی بصورت json برگردانده می‌شود اما با نسخه V3 سازگار نمی‌باشد و از روش پارامتر $format می‌توان استفاده نمود

```
$.getJSON("Northwind.svc/Products?$format=json", function (data) {
                $.each(data.d, function (i, item) {
                    $("<p/>").html(item.Product_Name + " " + item.Unit_Price).appendTo("#products");
                });
            });
```

همچنین از تابع Ajax که امکانات بیشتری را در اختیارمان قرار می‌دهد به راحتی می‌توان استفاده نمود به مثال زیر دقت کنید:

```
$.ajax('Northwind.svc/Products', {
                dataType: "json",
                beforeSend: function (xhr) {
                    xhr.setRequestHeader("Accept", "application/json;odata=verbose");
                    xhr.setRequestHeader("MaxDataServiceVersion", "3.0");
                },
                success: function (data) {
                    $.each(data.d, function (i, item) {
                        $("<p/>").html(item.Product_Name + " " +
item.Unit_Price).appendTo("#products");
                    });
                }
            });
```

با اسفاده از beforeSend مقدار Accept Header و MaxDataServiceVersion را تعیین نموده ایم.

بنابراین به کمک قراردادهای کوئری نویسی که در مطالب قبلی گفته شد می‌توان با استفاده از Url تابع Ajax به داده مورد نظر خود رسید.

نظرات خوانندگان

نویسنده: مرادی
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۷:۴۲

با سلام، بهتر نیست از jay data یا از breeze.js استفاده کنید ؟

نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۸:۳۴

در مطلب بعدی به این دو اشاره خواهم کرد قطعا امکانات بیشتری در اختیارمان قرار می‌دهد

نویسنده: abdali
تاریخ: ۱۳۹۲/۰۴/۱۱ ۱۵:۱۹

لطف میکنید کد رو قرار بدین ، چند بار امتحان کردم با خطا مواجه شدم . مرسی

عنوان: **Http Batch Processing در Asp.Net Web Api**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۱۱/۱۶ ۲۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.NET Web API, OData, Performance, ASP.NET Web API 2, Batch Request

بعد از معرفی نسخه‌ی 2 از Asp.Net Web Api و پشتیبانی رسمی آن از OData بسیاری از توسعه دهندگان سیستم نفس راحتی کشیدند؛ زیرا از آن پس می‌توانستند علاوه بر امکانات جالب و مهمی که تحت پروتکل OData میسر بود، از سایر امکانات تعبیه شده در نسخه‌ی دوم web Api نیز استفاده نمایند. یکی از این قابلیت‌ها، مبحث مهم Batching Processing است که در طی این پست با آن آشنا خواهیم شد.
منظور از Batch Request این است که درخواست دهنده بتواند چندین درخواست (Multiple Http Request) را به صورت یک Pack جامع، در قالب فقط یک درخواست (Single Http Request) ارسال نماید و به همین روال تمام پاسخ‌های معادل درخواست ارسال شده را به صورت یک Pack دیگر دریافت کرده و آن را پردازش نماید. نوع درخواست نیز مهم نیست یعنی می‌توان در قالب یک Pack چندین درخواست از نوع Post و Get یا حتی Put و ... نیز داشته باشید. بدیهی است که پیاده سازی این قابلیت در جای مناسب و در پروژه‌هایی با تعداد کاربران زیاد می‌تواند باعث بهبود چشمگیر کارآیی پروژه شود.

برای شروع همانند سایر مطالب می‌توانید از این پست جهت راه اندازی هاست سرویس‌های Web Api استفاده نمایید. برای فعال سازی قابلیت batching Request نیاز به یک MessageHandler داریم تا بتوانند درخواست‌هایی از این نوع را پردازش نمایند. خوشبختانه به صورت پیش فرض این Handler پیاده سازی شده‌است و ما فقط باید آن را با استفاده از متد MapHttpBatchRoute به بخش مسیر یابی (Route Handler) پروژه معرفی نماییم.

```
public class Startup
    {
        public void Configuration(IAppBuilder appBuilder)
        {
            var config = new HttpConfiguration();

            config.Routes.MapHttpBatchRoute(
                routeName: "Batch",
                routeTemplate: "api/$batch",
                batchHandler: new DefaultHttpBatchHandler(GlobalConfiguration.DefaultServer));

            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "Default",
                routeTemplate: "{controller}/{action}/{name}",
                defaults: new { name = RouteParameter.Optional }
            );

            config.Formatters.Clear();
            config.Formatters.Add(new JsonMediaTypeFormatter());
            config.Formatters.JsonFormatter.SerializerSettings.Formatting =
Newtonsoft.Json.Formatting.Indented;
            config.Formatters.JsonFormatter.SerializerSettings.ContractResolver = new
CamelCasePropertyNamesContractResolver();

            config.EnsureInitialized();
            appBuilder.UseWebApi(config);
        }
    }
```

مهم‌ترین نکته‌ی آن استفاده از DefaultHttpBatchHandler و معرفی آن به بخش batchHandler مسیریابی است. کلاس DefaultHttpBatchHandler برای وهله سازی نیاز به آبجکت سروری که سرویس‌های WebApi در آن هاست شده‌اند دارد که با دستور GlobalConfiguration.DefaultServer به آن دسترسی خواهید داشت. در صورتی که HttpServer خاص خود را دارید به صورت زیر عمل نمایید:

```
var config = new HttpConfiguration();
 HttpServer server = new HttpServer(config);
```

تنظیمات بخش سرور به اتمام رسید. حال نیاز داریم بخش کلاینت را طوری طراحی نماییم که بتواند درخواست را به صورت دسته‌ای ارسال نماید. در زیر یک مثال قرار داده شده است:

```
using System.Net.Http;
using System.Net.Http.Formatting;

public class Program
    {
        private static void Main(string[] args)
        {
            string baseAddress = "http://localhost:8080";
            var client = new HttpClient();
            var batchRequest = new HttpRequestMessage(HttpMethod.Post, baseAddress + "/api/$batch")
            {
                Content = new MultipartContent("mixed")
                {
                    new HttpMessageContent(new HttpRequestMessage(HttpMethod.Post, baseAddress +
"/api/Book/Add")
                    {
                        Content = new ObjectContent<string>("myBook", new JsonMediaTypeFormatter())
                    }),
                    new HttpMessageContent(new HttpRequestMessage(HttpMethod.Get, baseAddress +
"/api/Book/GetAll"))
                }
            };

            var batchResponse = client.SendAsync(batchRequest).Result;

            MultipartStreamProvider streamProvider =
batchResponse.Content.ReadAsMultipartAsync().Result;
            foreach (var content in streamProvider.Contents)
            {
                var response = content.ReadAsHttpResponseMessageAsync().Result;
            }
        }
    }
```

همان طور که می‌دانیم برای ارسال درخواست به سرویس Web Api باید یک نمونه از کلاس HttpRequestMessage وهله سازی شود سازنده‌ی آن به نوع HttpMethod اکشن نظیر (POST یا GET) و آدرس سرویس مورد نظر نیاز دارد. نکته‌ی مهم آن این است که خاصیت Content این درخواست باید از نوع MultipartContent و subType آن نیز باید mixed باشد. در بدنه‌ی آن نیز می‌توان تمام درخواست‌ها را به ترتیب و با استفاده از وهله سازی از کلاس HttpMessageContent تعریف کرد.
برای دریافت پاسخ این گونه درخواست‌ها نیز از متد الحاقی ReadAsMultipartAsync استفاده می‌شود که امکان پیمایش بر بدنه‌ی پیام دریافتی را می‌دهد.

مدیریت ترتیب درخواست ها

شاید این سوال به ذهن شما نیز خطور کرده باشد که ترتیب پردازش این گونه پیام‌ها چگونه خواهد بود؟ به صورت پیش فرض ترتیب اجرای درخواست‌ها حائز اهمیت است. بعنی تا زمانیکه پردازش درخواست اول به اتمام نرسد، کنترل اجرای برنامه، به درخواست بعدی نخواهد رسید که این مورد بیشتر زمانی رخ می‌دهد که قصد دریافت اطلاعاتی را داشته باشید که قبل از آن باید عمل Persist در پایگاه داده اتفاق بیافتد. اما در حالاتی غیر از این می‌توانید این گزینه را غیر فعال کرده تا تمام درخواست‌ها به صورت موازی پردازش شوند که به طور قطع کارایی آن نسبت به حالت قبلی بهینه‌تر است.
برای غیر فعال کردن گزینه‌ی ترتیب اجرای درخواست‌ها، به صورت زیر عمل نمایید:

```
config.Routes.MapHttpBatchRoute(
                routeName: "WebApiBatch",
                routeTemplate: "api/$batch",
                batchHandler: new DefaultHttpBatchHandler(GlobalConfiguration.DefaultServer)
                {
                    ExecutionOrder = BatchExecutionOrder.NonSequential
                });
```

تفاوت آن فقط در مقدار دهی خاصیت ExecutionOrder به صورت NonSequential است.