WCF Data Services

Introduction

DEVELOPMENTOR

DEVELOPING PEOPLE WHO DEVELOP SOFTWARE

Agenda

- Overview (What and why)
- Exposing a WCF Data Service
- Consuming a WCF Data Service

What is WCF Data Services

- Purpose
  - Expose Data as RESTful Web service
- History
  - Project Astoria at Mix 07 (Mar 2007)
  - V1.0 Released as ADO.NET Data Services in .NET 3.5 SP1 (Aug 2008)
  - V1.5 Update released (Jan 2010)
  - V4.0 WCF Data Services in .NET 4.0 (Apr 2010)
  - V5.0 Released in (Apr 2012)
- OData protocol used by the service

WCF data problems

- Exposing Data via WCF meant exposing methods
- One method for each operation
  - Read and Write meant 5 operations (CRrUD)
- One method for each query type
  - GetOrdersByProduct, GetOrdersByCustomer, GetOrderByOrderId, GetOrdersByCategory, etc.
  - Controlled by service but should be controlled by client

WCF Data Services solution

- New way of thinking about the data over REST problem
- Rather than exposing operations on the data
  - Expose the data itself
- Rather than compiling the service to support client needs
  - The client send its needs in the URI
- Is NOT exposing the database to the web
  - Exposes a data model (layer of indirection)

Open Data Protocol [OData]

- OData protocol specifies formats for exposing data over REST web service
  - set of URI conventions for querying a data model
  - CRUD operations based on HTTP verbs
  - AtomPub and JSON data representations
  - batch multiple operations into a single HTTP request

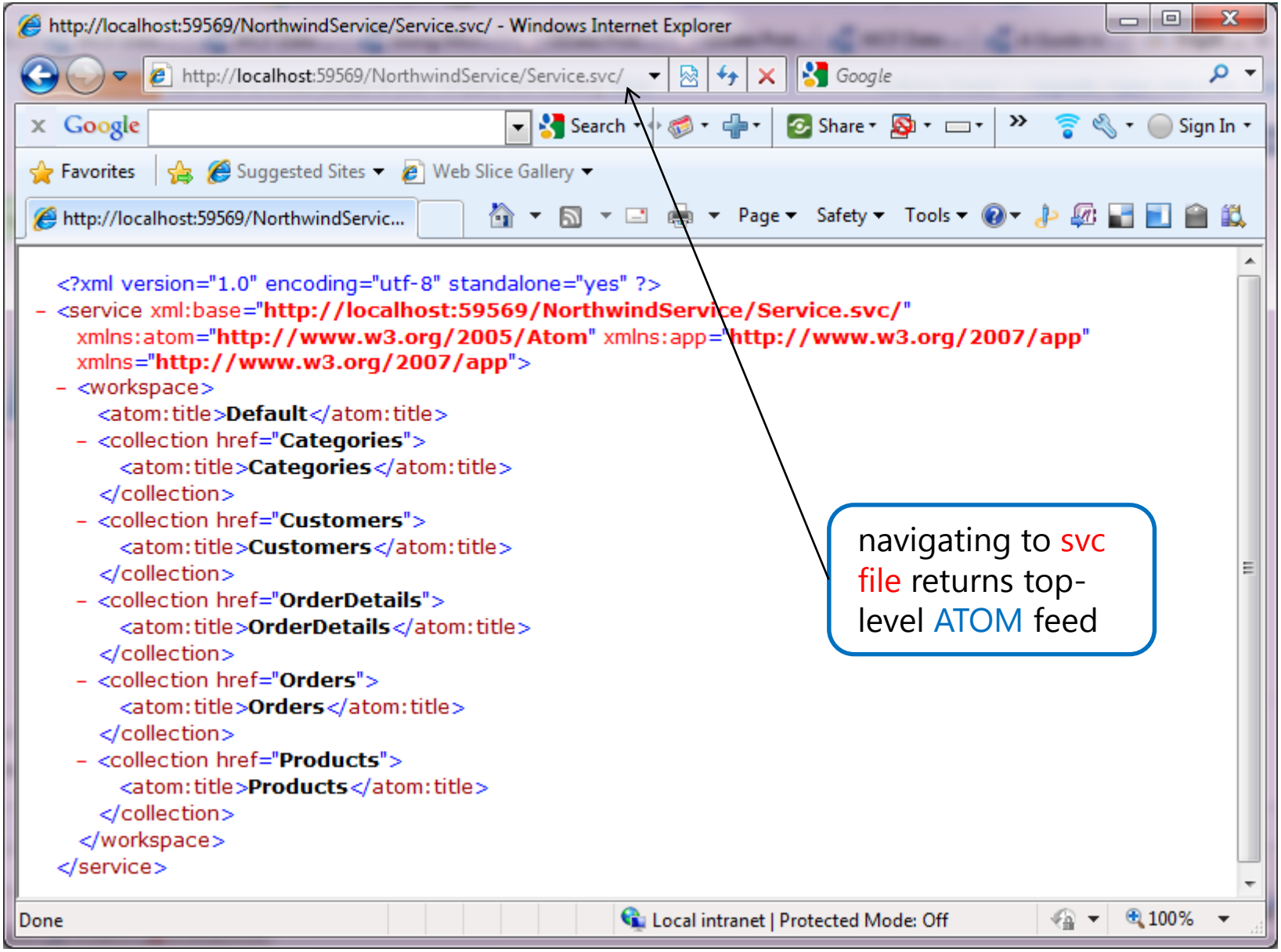
Exposing Entity Data Models

NorthwindDataService.svc

```
<%@ ServiceHost Language="C#"
                Factory="System.Data.Services.DataServiceHostFactory"
                Service="NorthwindDataService" %>

using System.Data.Services;

public class NorthwindDataService
:   DataService< NorthwindModel.NorthwindEntities >
{

    public static void InitializeService(IDataServiceConfiguration config)
    {
        config.SetEntitySetAccessRule("*", EntitySetRights.All);
    }

}
```

Enity Framework Object Context

Data Service shall provide access to all entity sets in object context


http://localhost:59569/NorthwindService/Service.svc/ - Windows Internet Explorer
http://localhost:59569/NorthwindService/Service.svc/
Google
Search
Share
Sign In
Favorites
Suggested Sites
Web Slice Gallery
http://localhost:59569/NorthwindServic...
Page
Safety
Tools
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<service xml:base="http://localhost:59569/NorthwindService/Service.svc/"
xmlns:atom="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
xmlns="http://www.w3.org/2007/app">
<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="OrderDetails">
<atom:title>OrderDetails</atom:title>
</collection>
<collection href="Orders">
<atom:title>Orders</atom:title>
</collection>
<collection href="Products">
<atom:title>Products</atom:title>
</collection>
</workspace>
</service>
navigating to svc file returns top-level ATOM feed
Done
Local intranet | Protected Mode: Off
100%

Data Services and ATOM

- ATOM syndication format provides structure
  - includes links to **edit** an entity or obtain **related entities**

```
<feed xml:base="http://MySite/NorthwindDataService.svc/" …
      xmlns="http://www.w3.org/2005/Atom">
  <entry>
    <id>…</id>
    <link rel="edit" title="Orders" href="Orders(10248)" />
    <link rel="…" title="…" href="Orders(10248)/Order_Details" />
    <content type="application/xml">
      <m:properties>
        <d:OrderID m:type="Edm.Int32">10248</d:OrderID>
        …
      </m:properties>
    </content>
  </entry>
  ...
```

uri to update this order

uri to get order details

Addressing entries

- Entries addressable by appending resource path of uri
  - **key value** in parentheses
  - **navigation** properties
  - **complex** types

service base address

```
http://localhost:1234/NorthwindService/Service.svc
```

```
/Customers
/Customers('ALFKI')
/Customers('ALFKI')/Orders
/Customers('ALFKI')/Orders(1)/Product
```

key values and properties

Query string parameters

- Execute query operators by adding query string parameters
  - each query option prefixed by $ character
  - logical, arithmetic and grouping operators supported
  - string, date, math, type operators also supported

LINQ statement	Request URI
entities.Orders.Take(2)	.../Orders?$top=2
entities.Orders.Skip(2).Take(2)	.../Orders?$skip=2&$top=2
from o in entities.Orders where o.ShipCountry=="USA"	.../Orders?$filter=ShipCountry eq 'USA'
from o in entities.Orders order by o.ShipCountry	.../Orders?$orderby=ShipCountry

Creating custom models

```
[DataServiceKey("Name")]
public class Person
{
    public string Name { get; set; }
    public int Age { get; set; }
}
```

```
public class PeopleRepository
{
    private List<Person> people = new List<Person>
    { /*...*/ };

    public IQueryable<Person> People
    {
        get { return people.AsQueryable(); }
    }
}
```

Customizing the Atom feed

- In a custom model

```
[EntityPropertyMapping("Name",
    SyndicationItemProperty.Title,
    SyndicationTextContentKind.Plaintext,
    keepInContent: true)]
```

- In an Entity model

```
<edmx:Edmx Version="3.0"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
```

```
<EntityType Name="Title">
  <Property Name="BookTitle"
     m:FC_TargetPath="SyndicationTitle"
     m:FC_ContentKind="text"
     m:FC_KeepInContent="true"/>
```

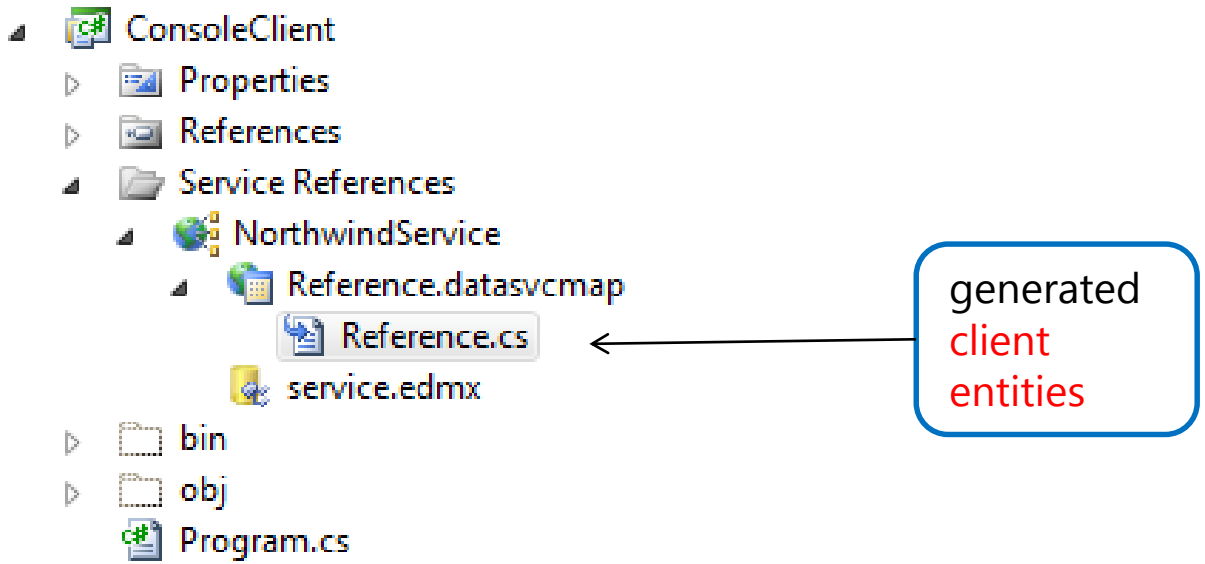
Retrieving data from WCF Data Service

```
using System;
using System.Net;
using System.IO;

class Program
{
  static void Main(string[] args)
  {
    DumpUri("http://MySite/NorthwindDataService.svc/Orders");
  }

  static void DumpUri(string uri)
  {
    WebRequest req = WebRequest.Create(uri);
    using (WebResponse rsp = req.GetResponse())
      Console.WriteLine(new StreamReader(
        rsp.GetResponseStream()).ReadToEnd());
  }
}
```

Returns data for all orders

Data Services client library

- Add a Service Reference to .svc file
  - $metadata option generates CSDL file describing entities
  - used by service reference to generate **client entities**



DataServiceContext

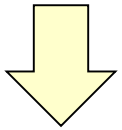
- Adding service reference creates a container class
  - derives from **`DataServiceContext`**
  - provides identity management and change tracking
  - ExecuteBatch method allows batched operations

```
NorthwindEntities ctx = new NorthwindEntities
   (new Uri("http://localhost:1234/NorthwindService/
    Service.svc"));

Uri reqUri = new Uri("Products?$filter=UnitPrice gt 20"
 + "&$orderby=ProductName", UriKind.Relative);

IEnumerable<Product> query = ctx.Execute<Product>(reqUri);
```

Client LINQ provider

- LINQ queries translated to uri's
  - cast to **`DataServiceQuery`** to obtain **request uri**

```
IQueryable<Product> query =
    from p in ctx.Products
    where p.Category.CategoryName == "Beverages"
    orderby p.ProductName
    select p;
DataServiceQuery dsQuery = (DataServiceQuery)query;
Console.WriteLine("\n{0}", dsQuery.RequestUri);
```



```
.../Products()?$filter=Category/CategoryName eq 'Beverages'
&$orderby=ProductName
```

Client change-tracking

- DataServiceContext tracks changes to entities
  - entity states: unchanged, modified, added, deleted
- Change-state must be set explicitly
  - UpdateObject, SetLink
  - AddObject, DeleteObject

```
product.UnitPrice++;
ctx.UpdateObject(product);

product.Category = confections;
ctx.SetLink(product, "Category", confections);

ctx.SaveChanges();
```

entities and relations marked as changed

Save changes

- SaveChangesOptions enum specifies how changes are done
  - options for batch updating, post vs merge, continue on error

SaveChangesOptions	Remark
None	One HTTP request for each modification
Batch	Single HTTP Post request containing all updates is sent to service
ReplaceOnUpdate	Use HTTP verb "POST" instead of "MERGE"
ContinureOnError	Multiple requests are sent. When request causes error, further requests are still sent.

Summary

- Expose data model as a REST service
  - service clients without SOAP (AJAX, Silverlight)
  - entities are treated as resources
  - include operations using [WebGet] and [WebInvoke]
- Leverage HTTP and AtomPub
  - HTTP verbs mapped to SQL commands
  - response includes links to related entities
- Can use JSON format for browser clients
- Add a service reference to utilize client library
  - use LINQ queries instead of raw uri's
  - call context methods to track changes
  - batch queries and updates possible