FIREBRAND

Microsoft

MCSD: 70-487: Developing Windows Azure and Web Services Courseware

Version 1.1

www.firebrandtraining.com

Module 1
Overview of Service and Cloud Technologies

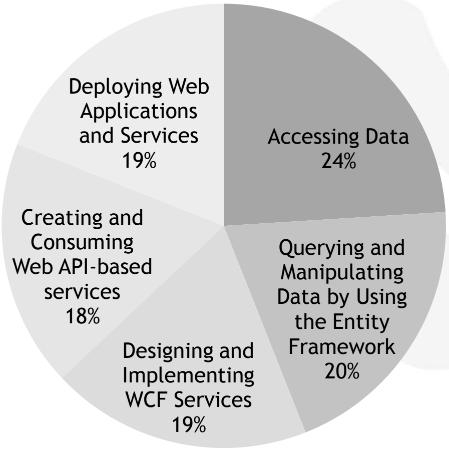
Developing Windows Azure and Web Services

Updated 11th April 2014

Developing Windows Azure and Web Services

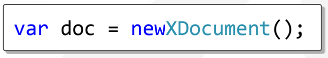
Course and Exam Contents

Microsoft Official Exam Guide



Jun 2013 to Jan 2014
130 minutes total
42 questions in total
22 in main section
2 case studies
(9, 11 questions)

Warnings!
Some of the code in questions miss spaces, commas, or semi-colons. Try not to let that put you off:



```
var doc = newXDocument();
```

50% code / 50% configuration
33% data, 33% WCF, 33% other

Microsoft Exam 70-487 Study Guide
http://www.bloggedbychris.com/2013/01/09/microsoft-exam-70-487-study-guide/

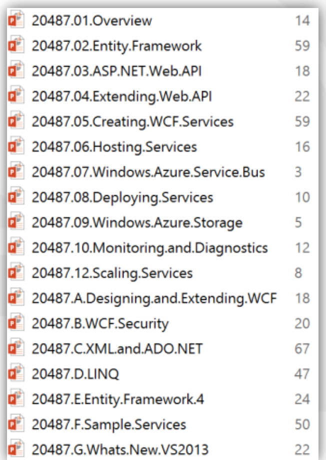
1.3

Developing Windows Azure and Web Services

Estimate of Number of Exam Questions per Module

Module	Qs
1: Overview of Service and Cloud Technologies	1
2: Querying and Manipulating Data Using Entity Framework	4
3: Creating and Consuming ASP.NET Web API Services	3
4: Extending and Securing ASP.NET Web API Services	3
5: Creating WCF Services	7
6: Hosting Services	3
7: Windows Azure Service Bus	1
8: Deploying Services	4
9: Windows Azure Storage	2
10: Monitoring and Diagnostics	1
12: Scaling Services	1
A: Designing and Extending WCF Services	4
B: Implementing Security in WCF Services	2
C: “Classic” XML and ADO.NET	4
D: LINQ	2
Total questions in exam	**42**

Firebrand Extra Slides



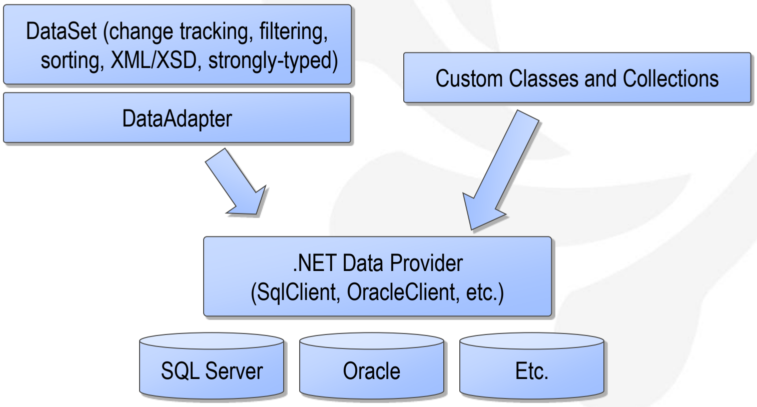
20487.01.Overview	14
20487.02.Entity.Framework	59
20487.03.ASP.NET.Web.API	18
20487.04.Extending.Web.API	22
20487.05.Creating.WCF.Services	59
20487.06.Hosting.Services	16
20487.07.Windows.Azure.Service.Bus	3
20487.08.Deploying.Services	10
20487.09.Windows.Azure.Storage	5
20487.10.Monitoring.and.Diagnostics	12
20487.12.Scaling.Services	8
20487.A.Designing.and.Extending.WCF	18
20487.B.WCF.Security	20
20487.C.XML.and.ADO.NET	67
20487.D.LINQ	47
20487.E.Entity.Framework.4	24
20487.F.Sample.Services	50
20487.G.Whats.New.VS2013	22

1.4

Overview

Data APIs in .NET (2002-2007)

- “Classic” ADO.NET
  - .NET Framework 1.0, 1.1, and 2.0


DataSet (change tracking, filtering, sorting, XML/XSD, strongly-typed)
DataAdapter
Custom Classes and Collections
.NET Data Provider (SqlClient, OracleClient, etc.)
SQL Server
Oracle
Etc.

* .NET 2.0 adds minor improvements like TableAdapters

1.5

Overview

Object-Relational Mapping

What are ORMs?

- Objects are more natural to work with for programmers...
- ...but relational data is better for storage
- Mapping converts CRUD on objects to CRUD on relational data

Philosophy of ORM

- If you do most work through stored procedures (SELECT, etc.) you will gain very little from using an ORM so use “Classic” ADO.NET instead

The objects should be “persistence ignorant”

- Members are mostly properties to store column values
- Can have methods for validation and business logic
- Should NOT have methods to store data

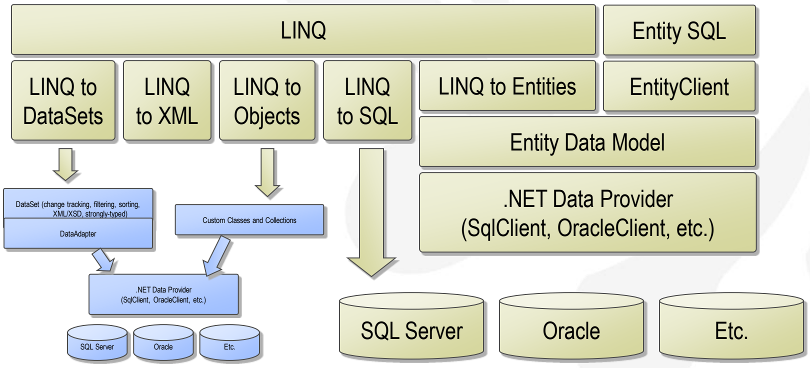
1.6

Overview

Data APIs in .NET (2008-2011)

LINQ and Entity Framework

- .NET Framework 3.5 SP1 and .NET 4


LINQ
Entity SQL
LINQ to DataSets
LINQ to XML
LINQ to Objects
LINQ to SQL
LINQ to Entities
EntityClient
Entity Data Model
.NET Data Provider (SqlClient, OracleClient, etc.)
DataSet (change tracking, filtering, sorting, XML/XSD, strongly-typed)
DataAdapter
Custom Classes and Collections
.NET Data Provider (SqlClient, OracleClient, etc.)
SQL Server
Oracle
Etc.
SQL Server
Oracle
Etc.

1.7

Overview

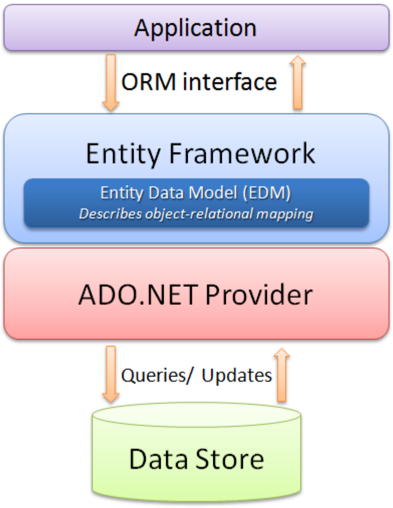
Entity Framework vs. LINQ to SQL

LINQ to SQL, .NET 3.5

- Created by C# team
- Simple ORM; one-to-one object-to-table mapping (although it does support a discriminator column for simple inheritance scenarios)
- SQL Server only
- Will be supported, but not improved

Entity Framework, .NET 3.5 SP1

- Created by SQL Server team
- Complex, powerful, flexible ORM
- Heterogeneous data sources
- Future of Microsoft Data APIs



Application

ORM interface

Entity Framework

Entity Data Model (EDM)

Describes object-relational mapping

ADO.NET Provider

Queries/ Updates

Data Store

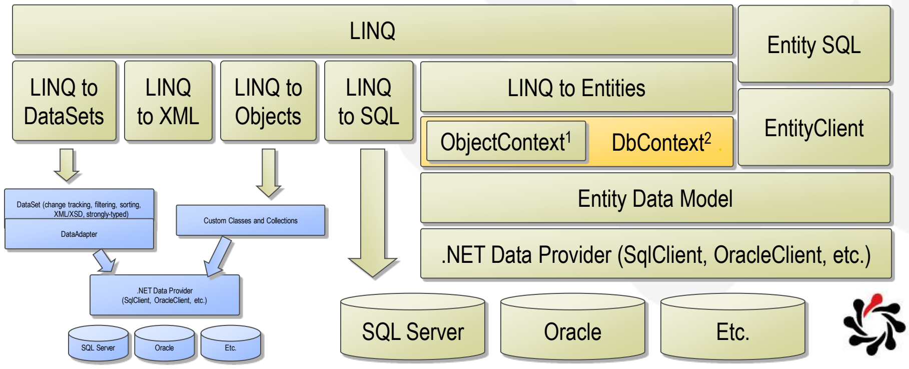
1.8

Overview

Data APIs in .NET (2012+)

.NET Framework 4.5

- Appendix C: “Classic” ADO.NET and XML
- Appendix D: LINQ (including common providers like LINQ to XML)
- Appendix E: Entity Framework 4 ([1]Visual Studio 2010)
- Module 2: Entity Framework 5 ([2]Visual Studio 2012)



LINQ

Entity SQL

LINQ to DataSets

LINQ to XML

LINQ to Objects

LINQ to SQL

LINQ to Entities

ObjectContext[1]

DbContext[2]

EntityClient

Entity Data Model

.NET Data Provider (SqlClient, OracleClient, etc.)

DataSet (change tracking, filtering, sorting, XML/XSD, strongly-typed)

DataAdapter

Custom Classes and Collections

.NET Data Provider (SqlClient, OracleClient, etc.)

SQL Server

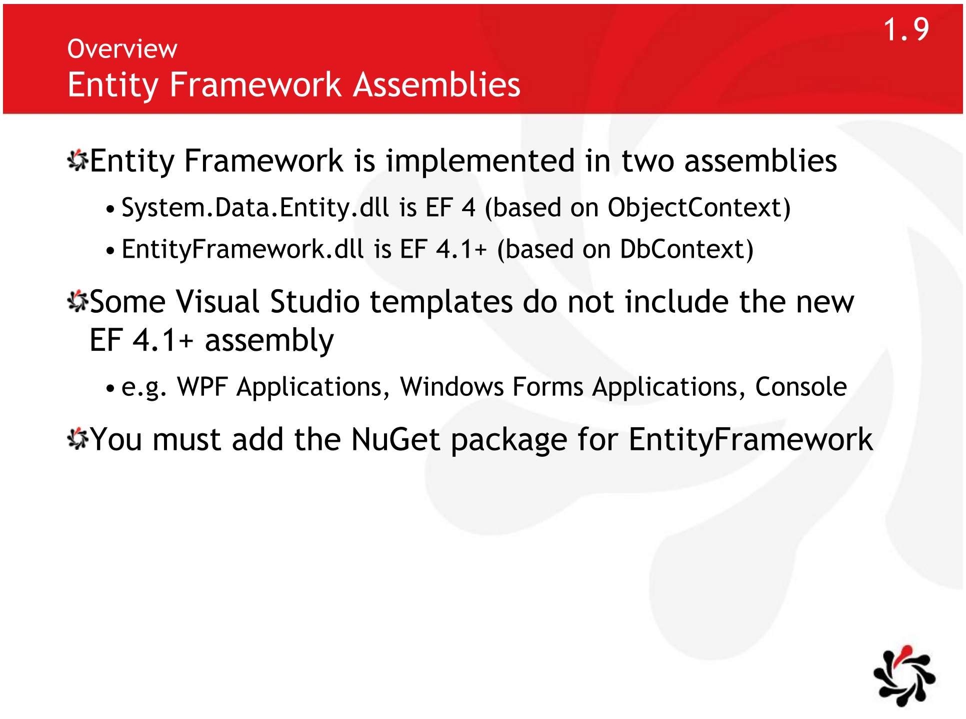
Oracle

Etc.

SQL Server

Oracle

Etc.

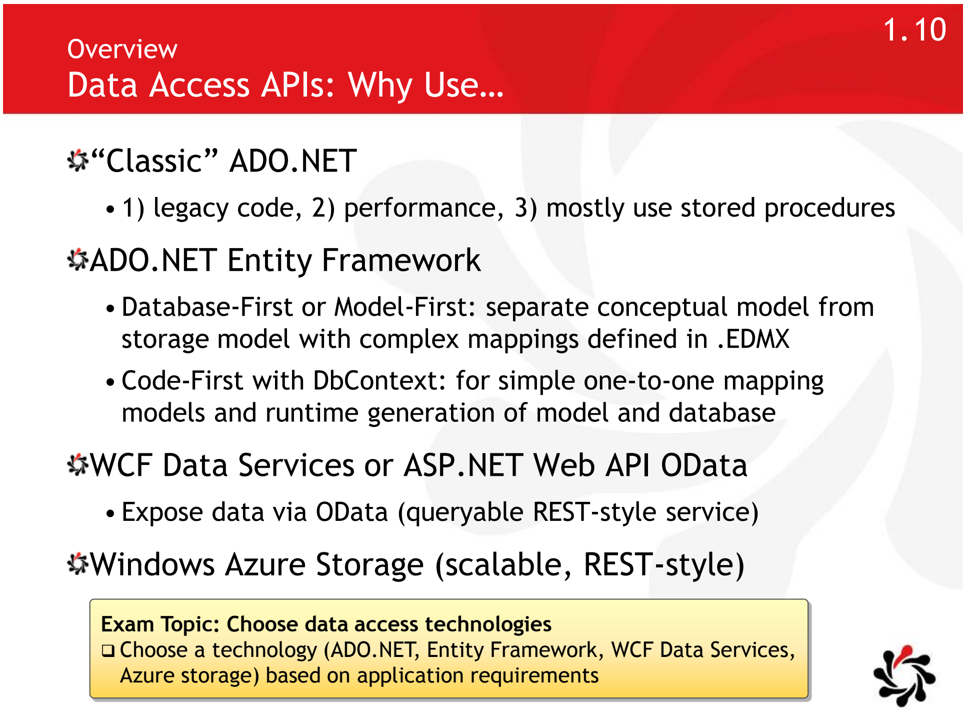


1.9

Overview

Entity Framework Assemblies

- Entity Framework is implemented in two assemblies
  - System.Data.Entity.dll is EF 4 (based on ObjectContext)
  - EntityFramework.dll is EF 4.1+ (based on DbContext)
- Some Visual Studio templates do not include the new EF 4.1+ assembly
  - e.g. WPF Applications, Windows Forms Applications, Console
- You must add the NuGet package for EntityFramework



1.10

Overview

Data Access APIs: Why Use…

- “Classic” ADO.NET
  - 1) legacy code, 2) performance, 3) mostly use stored procedures
- ADO.NET Entity Framework
  - Database-First or Model-First: separate conceptual model from storage model with complex mappings defined in .EDMX
  - Code-First with DbContext: for simple one-to-one mapping models and runtime generation of model and database
- WCF Data Services or ASP.NET Web API OData
  - Expose data via OData (queryable REST-style service)
- Windows Azure Storage (scalable, REST-style)

Exam Topic: Choose data access technologies
❑ Choose a technology (ADO.NET, Entity Framework, WCF Data Services, Azure storage) based on application requirements

Overview

Which Databases Work with Entity Framework?

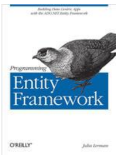
- Supported .NET Data Providers
  - MySQL, Oracle, Progress, VistaDB, SQL Lite, PostgreSQL, Virtuoso, IBM DB2, Informix, U2, Sybase, Sybase SQL Anywhere, Synergy, Firebird, Npgsql, and many others

Third-Party Providers
`http://msdn.microsoft.com/en-us/data/dd363565.aspx`

Further Study

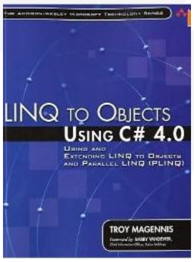
Entity Framework and LINQ

- To go deeper with Entity Framework
  - Programming Entity Framework, 2nd Edition
  - Experienced author, Julie Lerman
  - http://thedatafarm.com/blog/



- To go further into EF 4.1+

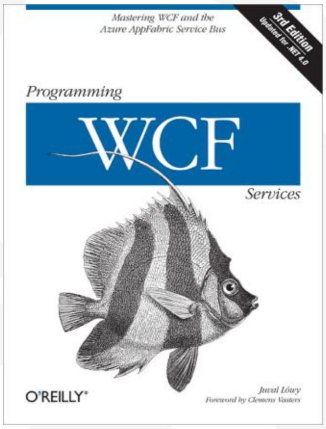
- LINQ to Objects using C# 4.0
  - Using and Extending LINQ to Objects and Parallel LINQ


LINQ to Objects Using C# 4.0
Using and Extending LINQ to Objects and Parallel LINQ (PLINQ)
Troy Magennis

1.13

Further Study
Windows Communication Foundation (WCF)

- To go deeper with WCF
  - Programming WCF Services, 3rd Edition
  - Experienced author, Juval Löwy
- Why this book?
  - Focuses on the possibilities and practical aspects of using WCF 4 so it complements the MOC
  - Contains many useful utilities, tools, and helper classes such as ServiceModelEx


Mastering WCF and the Azure AppFabric Service Bus
3rd Edition
Updated for .NET 4.0
Programming
WCF
Services
O'REILLY
Juval Löwy
Foreword by Clemens Vasters

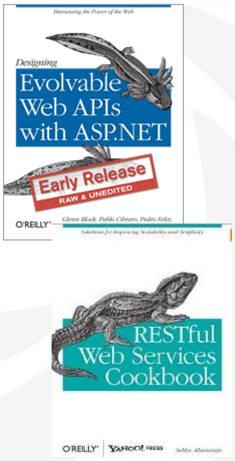
"Every Object Should be a WCF Service"
http://www.dotnetrocks.com/default.aspx?showNum=520

Programming WCF Services, 3rd Edition
http://shop.oreilly.com/product/9780596805494.do

1.14

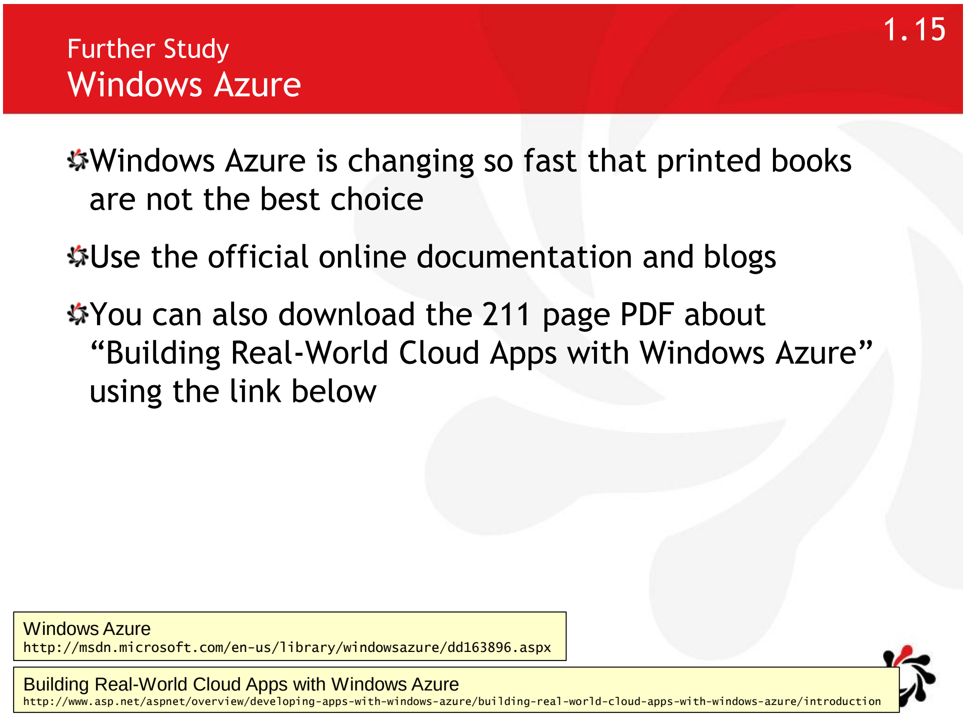
Further Study
ASP.NET Web API

- To go deeper with Web API
  - Designing Evolvable Web APIs with ASP.NET
  - Experienced authors
  - Due September 2013 but available now as an Early Release
- To go deeper with general concept of REST/HTTP style services
  - RESTful Web Services Cookbook



RESTful Web Services Cookbook
http://www.amazon.co.uk/RESTful-Web-Services-Cookbook-ebook/dp/B0043D2ESQ

Designing Evolvable Web APIs with ASP.NET
http://shop.oreilly.com/product/0636920026617.do



1.15

Further Study

Windows Azure

- Windows Azure is changing so fast that printed books are not the best choice
- Use the official online documentation and blogs
- You can also download the 211 page PDF about "Building Real-World Cloud Apps with Windows Azure" using the link below

Windows Azure
http://msdn.microsoft.com/en-us/library/windowsazure/dd163896.aspx

Building Real-World Cloud Apps with Windows Azure
http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/introduction

2.1

Module 2
Querying and Manipulating Data Using Entity Framework

Developing Windows Azure and Web Services

Updated 11th April 2014

2.2

Entity Framework
Contents

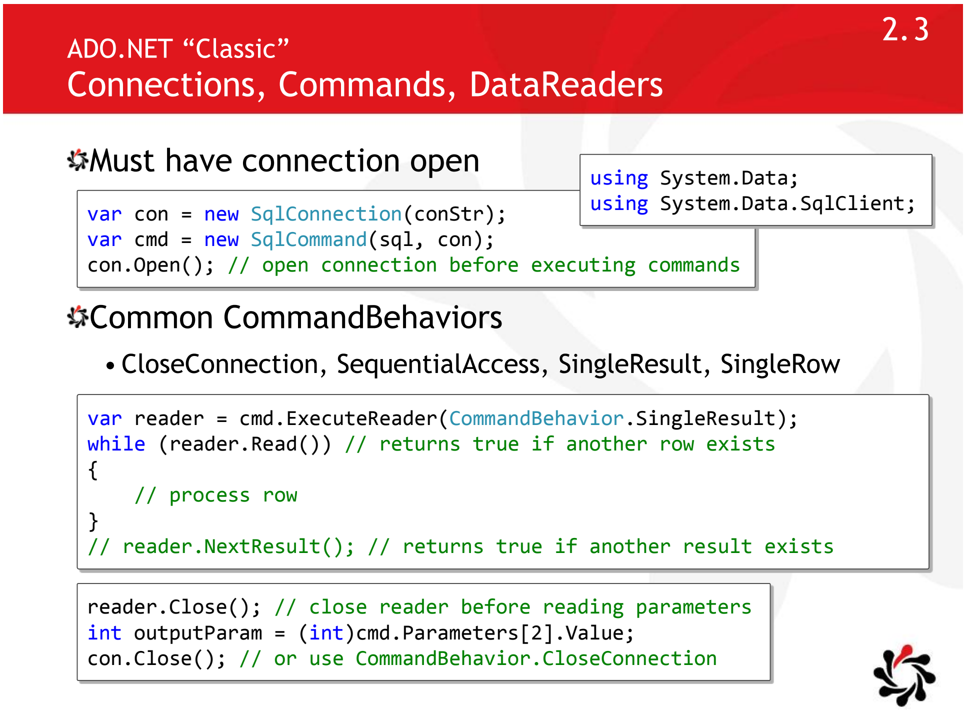
Topic	Slide
ADO.NET “Classic”	3
Connection Strings	6
Entity Data Models	11
Stored Procs and Functions	17
Code First (and POCO)	22
Inheritance Hierarchies	30
Designer Tools	33
Loading Patterns	38
Entity SQL	39
EntityClient	46
Querying Summary	48
Performance	49
Transactions	50

Exam Topic: Create an Entity Framework data model
- ❑ Structure the data model using Table per type, table per class, table per hierarchy
- ❑ Choose and implement an approach to manage a data model (code first vs. model first vs. database first)
- ❑ Implement POCO objects
- ❑ Describe a data model by using conceptual schema definitions, storage schema definition, and mapping language (CSDL, SSDL, MSL)

Exam Topic: Query data by using Data Provider for EF
- ❑ Query data by using Connection, DataReader, Command from the System.Data.EntityClient namespace
- ❑ Perform synchronous and asynchronous operations

Exam Topic: Query and manipulate data by using EF
- ❑ Query data by using DbContext
- ❑ Build a query that uses deferred execution
- ❑ Implement lazy loading and eager loading
- ❑ Query data by using Entity SQL
- ❑ Update, and delete data by using DbContext
- ❑ Create and run compiled queries

Exam Topic: Implement transactions
- ❑ Manage transactions by using the API from System.Transactions
- ❑ Implement distributed transactions
- ❑ Specify transaction isolation level



2.3

ADO.NET “Classic”
Connections, Commands, DataReaders

- Must have connection open

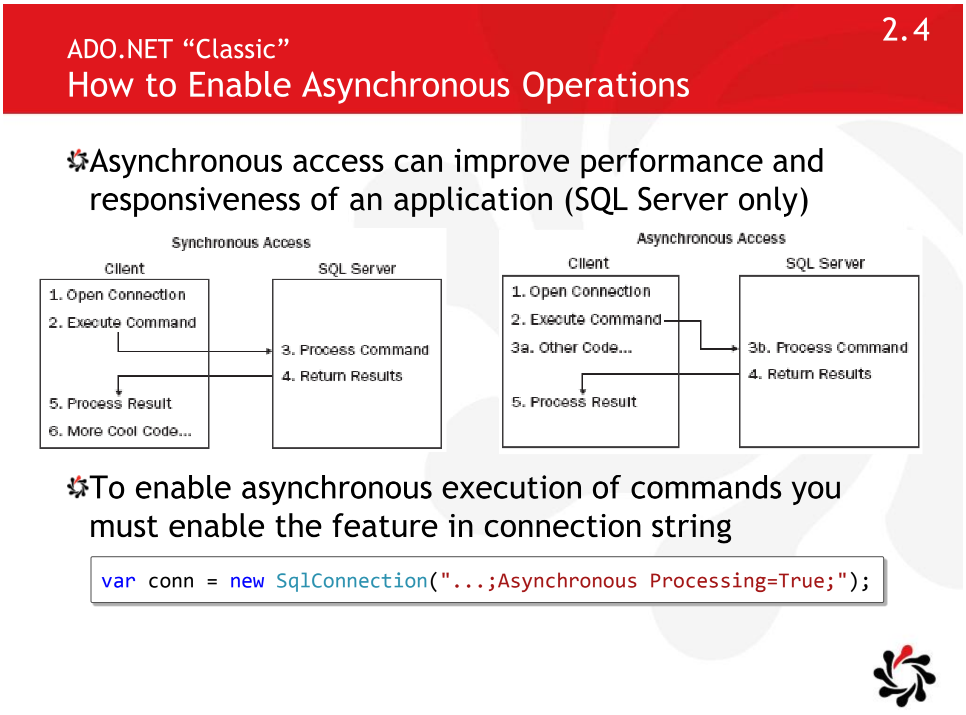
```
using System.Data;
using System.Data.SqlClient;
```

```
var con = new SqlConnection(conStr);
var cmd = new SqlCommand(sql, con);
con.Open(); // open connection before executing commands
```

- Common CommandBehaviors
  - CloseConnection, SequentialAccess, SingleResult, SingleRow

```
var reader = cmd.ExecuteReader(CommandBehavior.SingleResult);
while (reader.Read()) // returns true if another row exists
{
    // process row
}
// reader.NextResult(); // returns true if another result exists
```

```
reader.Close(); // close reader before reading parameters
int outputParam = (int)cmd.Parameters[2].Value;
con.Close(); // or use CommandBehavior.CloseConnection
```



2.4

ADO.NET “Classic”
How to Enable Asynchronous Operations

- Asynchronous access can improve performance and responsiveness of an application (SQL Server only)

Synchronous Access

Client: 1. Open Connection; 2. Execute Command; 5. Process Result; 6. More Cool Code...

SQL Server: 3. Process Command; 4. Return Results

Asynchronous Access

Client: 1. Open Connection; 2. Execute Command; 3a. Other Code...; 5. Process Result

SQL Server: 3b. Process Command; 4. Return Results

- To enable asynchronous execution of commands you must enable the feature in connection string

```
var conn = new SqlConnection("...;Asynchronous Processing=True;");
```

2.5

ADO.NET “Classic”

How to Execute Asynchronous Operations

- ADO.NET 4 and earlier

```
IAsyncResult task = cmd.BeginExecuteNonQuery();
// do other work; can check task.IsCompleted
int i = cmd.EndExecuteNonQuery(task);
```

- ADO.NET 4.5

```
Task<int> t = cmd.ExecuteNonQueryAsync();
// do other work
int i = await t;
```

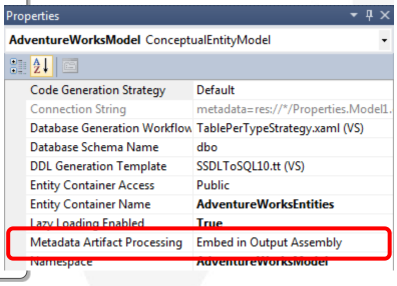
2.6

Connection Strings

EDM Embedded in Assembly

- By default the three XML files that define an EDM are embedded in the output assembly as resources

```
<add name="AdventureWorksEntities"
  providerName="System.Data.EntityClient"
  connectionString="metadata=
    res://*/Properties.Model1.csdl|
    res://*/Properties.Model1.ssdl|
    res://*/Properties.Model1.msl;
  provider=System.Data.SqlClient;
  provider connection string='
    data source=.\sqlexpress;
    initial catalog=AdventureWorks;
    integrated security=True;
    multipleactiveresultsets=True;
    App=EntityFramework'"
/>
```



AdventureWorksModel ConceptualEntityModel

Code Generation Strategy	Default
Connection String	metadata=res://*/Properties.Model1.
Database Generation Workflow	TablePerTypeStrategy.xaml (VS)
Database Schema Name	dbo
DDL Generation Template	SSDLToSQL10.tt (VS)
Entity Container Access	Public
Entity Container Name	**AdventureWorksEntities**
Lazy Loading Enabled	**True**
Metadata Artifact Processing	Embed in Output Assembly
Namespace	**AdventureWorksModel**

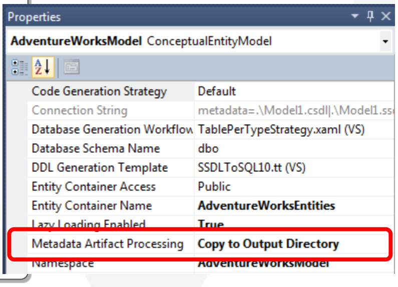
Entity Framework
http://msdn.microsoft.com/en-us/data/aa937723

2.7

Connection Strings
EDM Stored As Loose Files

- Any of the files can be stored outside the assembly but the connection string must be changed

```
<add name="AdventureWorksEntities"
  providerName="System.Data.EntityClient"
  connectionString="metadata=
    .\Model1.csdl|
    .\Model1.ssdl|
    .\Model1.msl;
  provider=System.Data.SqlClient;
  provider connection string='
    data source=.\sqlexpress;
    initial catalog=AdventureWorks;
    integrated security=True;
    multipleactiveresultsets=True;
    App=EntityFramework'"
/>
```



Properties

AdventureWorksModel ConceptualEntityModel

Code Generation Strategy	Default
Connection String	metadata=.\Model1.csdl|.\Model1.ss
Database Generation Workflow	TablePerTypeStrategy.xaml (VS)
Database Schema Name	dbo
DDL Generation Template	SSDLToSQL10.tt (VS)
Entity Container Access	Public
Entity Container Name	AdventureWorksEntities
Lazy Loading Enabled	True
Metadata Artifact Processing	Copy to Output Directory
Namespace	AdventureWorksModel

2.8

Connection Strings
Load Metadata from Resource or File System

```
metadata=res://<assemblyFullName>/<resourceName>
```

Option	Description
assemblyFullName	ResourceLib, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null If you specify wildcard (*) it searches: 1) calling assembly 2) referenced assemblies 3) assemblies in the bin folder `metadata=res://*/`
resourceName	The name of the resource, e.g. AdventureWorksModel.csdl If not specified, all metadata resources will be loaded

```
metadata=res://AdventureWorks, 1.0.0.0, neutral, a14...840/model.csdl
| model.ssdl | model.msl
```

Entity Framework Connection Strings
http://msdn.microsoft.com/en-us/library/cc716756.aspx

2.9

Connection Strings

EntityConnectionStringBuilder

- Should be used in conjunction with a SqlConnectionStringBuilder (or underlying provider)
  - The connection string for the underlying data source is supplied by the ProviderConnectionString property which is not checked for valid keyword/value pairs

```
var sqlBuilder = new SqlConnectionStringBuilder();
sqlBuilder.DataSource = serverName;
// set other provider-specific properties
```

```
var efBuilder = new EntityConnectionStringBuilder();
efBuilder.Provider = providerName;
efBuilder.ProviderConnectionString = sqlBuilder.ToString();
efBuilder.Metadata = @"res://*/AdventureWorksModel.csdl|...";
string s = efBuilder.ConnectionString;
```

EntityConnectionStringBuilder Class - http://msdn.microsoft.com/en-us/library/system.data.entityclient.entityconnectionstringbuilder.aspx

2.10

Connection Strings

Configuring Provider Connection Strings

- Connection strings provide the necessary information for an application to connect a data source
  - Server, database, security information, connection pool size
- Many parameters have multiple possible keywords for backwards compatibility e.g. these all mean the same
  - Data Source, server, address, addr, Network Address
- SQL Server (servername\instancename)

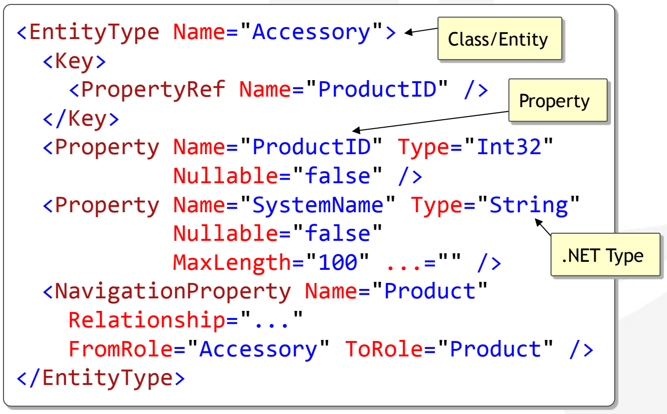
```
Data Source=FLORIDA\MIAMI;
Initial Catalog=Northwind;
Integrated Security=SSPI;
```

```
database=Northwind;
```

2.11

Entity Data Models
Defining an Entity

- CSDL (Conceptual Schema Definition Language)

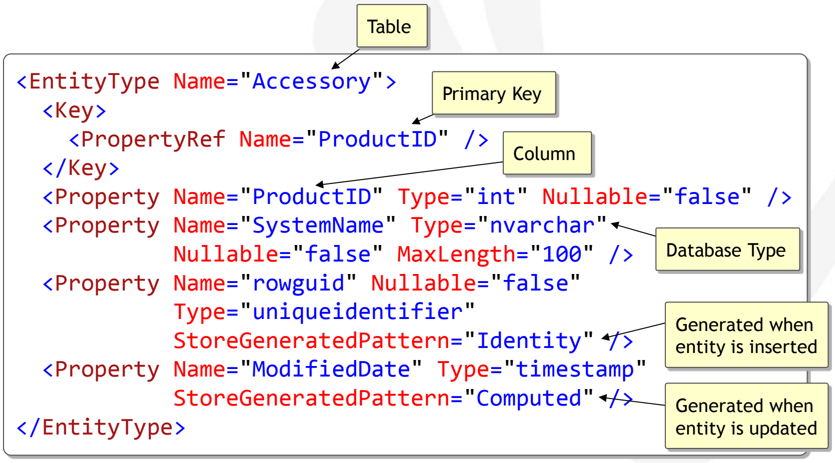


```
<EntityType Name="Accessory">
  <Key>
    <PropertyRef Name="ProductID" />
  </Key>
  <Property Name="ProductID" Type="Int32"
            Nullable="false" />
  <Property Name="SystemName" Type="String"
            Nullable="false"
            MaxLength="100" ...="" />
  <NavigationProperty Name="Product"
    Relationship="..."
    FromRole="Accessory" ToRole="Product" />
</EntityType>
```

2.12

Entity Data Models
Defining a Table

- SSDL (Store Schema Definition Language)



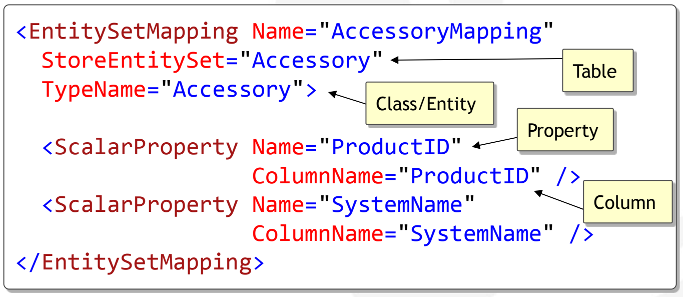
```
<EntityType Name="Accessory">
  <Key>
    <PropertyRef Name="ProductID" />
  </Key>
  <Property Name="ProductID" Type="int" Nullable="false" />
  <Property Name="SystemName" Type="nvarchar"
            Nullable="false" MaxLength="100" />
  <Property Name="rowguid" Nullable="false"
            Type="uniqueidentifier"
            StoreGeneratedPattern="Identity" />
  <Property Name="ModifiedDate" Type="timestamp"
            StoreGeneratedPattern="Computed" />
</EntityType>
```

Table
Primary Key
Column
Database Type
Generated when entity is inserted
Generated when entity is updated

Entity Data Models

Entity to Table

MSL (Mapping Specification Language)



```
<EntitySetMapping Name="AccessoryMapping"
  StoreEntitySet="Accessory"
  TypeName="Accessory">

  <ScalarProperty Name="ProductID"
                  ColumnName="ProductID" />
  <ScalarProperty Name="SystemName"
                  ColumnName="SystemName" />
</EntitySetMapping>
```

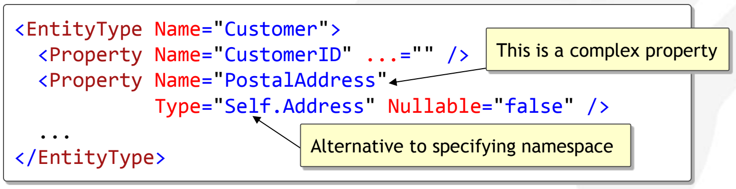
Entity Data Models

Define Complex Types and Properties in CSDL

Use ComplexTypes to group scalar properties for reuse

```
<Schema Namespace="AdventureWorksModel" Alias="Self"
```

```
<ComplexType Name="Address">
  <Property Name="Street" Type="String" />
  <Property Name="City" Type="String" />
  <Property Name="Region" Type="String" />
  <Property Name="PostalCode" Type="String" />
</ComplexType>
```



```
<EntityType Name="Customer">
  <Property Name="CustomerID" ...="" />
  <Property Name="PostalAddress"
            Type="Self.Address" Nullable="false" />
  ...
</EntityType>
```

This is a complex property
Alternative to specifying namespace

- Complex types are defined and used in the conceptual layer and have NO affect on storage structure

2.15

Entity Data Models

Mapping Complex Properties in MSL

- Use MappingFragment to map a complex property

```
<EntitySetMapping Name="Customers">
  <EntityTypeMapping TypeName="Customer">
    <MappingFragment StoreEntitySet="Customer">
      <ScalarProperty Name="CustomerID"
                      ColumnName="CId" />
      <ComplexProperty Name="PostalAddress"
                       TypeName="Address">
        <ScalarProperty Name="Street"
                        ColumnName="StreetAddress" />
        <ScalarProperty Name="City" ColumnName="City" />
        ...
      </ComplexProperty>
      ...
```

2.16

Entity Data Models

Conditional Mapping Fragments

- Use a Condition element to filter rows from table into an entity
  - In this example the Person table has a discriminator column
  - If the value is S it means the person is a student

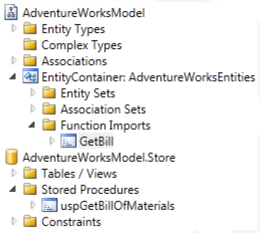
```
<EntityTypeMapping TypeName="Student">
  <MappingFragment StoreEntitySet="Person">
    <ScalarProperty Name="PersonID" ColumnName="pid" />
    <ScalarProperty Name="FirstName" ColumnName="fn" />
    <Condition ColumnName="PersonCategory" Value="S" />
  </MappingFragment>
</EntityTypeMapping>
```

```
<EntityTypeMapping TypeName="Employee">
  <MappingFragment StoreEntitySet="Person">
    <ScalarProperty Name="PersonID" ColumnName="pid" />
    <ScalarProperty Name="FirstName" ColumnName="fn" />
    <Condition ColumnName="PersonCategory" Value="E" />
```

2.17

Stored Procedures and Functions
Adding as a Method on ObjectContext or DbContext

```
Bill b = db.GetBill(23).First();
```



AdventureWorksModel
- Entity Types
- Complex Types
- Associations
- EntityContainer: AdventureWorksEntities
  - Entity Sets
  - Association Sets
  - Function Imports
    - GetBill

AdventureWorksModel.Store
- Tables / Views
- Stored Procedures
  - uspGetBillOfMaterials
- Constraints

- CSDL

```
<FunctionImport Name="GetBill">
  <Parameter Name="StartProductID"
    Mode="In" Type="Int32" />
```

- MSL

```
<FunctionImportMapping FunctionImportName="GetBill"
  FunctionName="AWModel.Store.uspGetBillOfMaterials" />
```

- SSDL

```
<Function Name="uspGetBillOfMaterials" ... Schema="dbo">
  <Parameter Name="StartProductID" Type="int" Mode="In" />
```

VS2010 adding a stored procedure adds <Function> to the SSDL, but you must also "import" the function to the CSDL which then adds the mapping and method
VS2012 adding a stored procedure does all four parts

2.18

Stored Procedures and Functions
Using Stored Procedures to Modify Entities

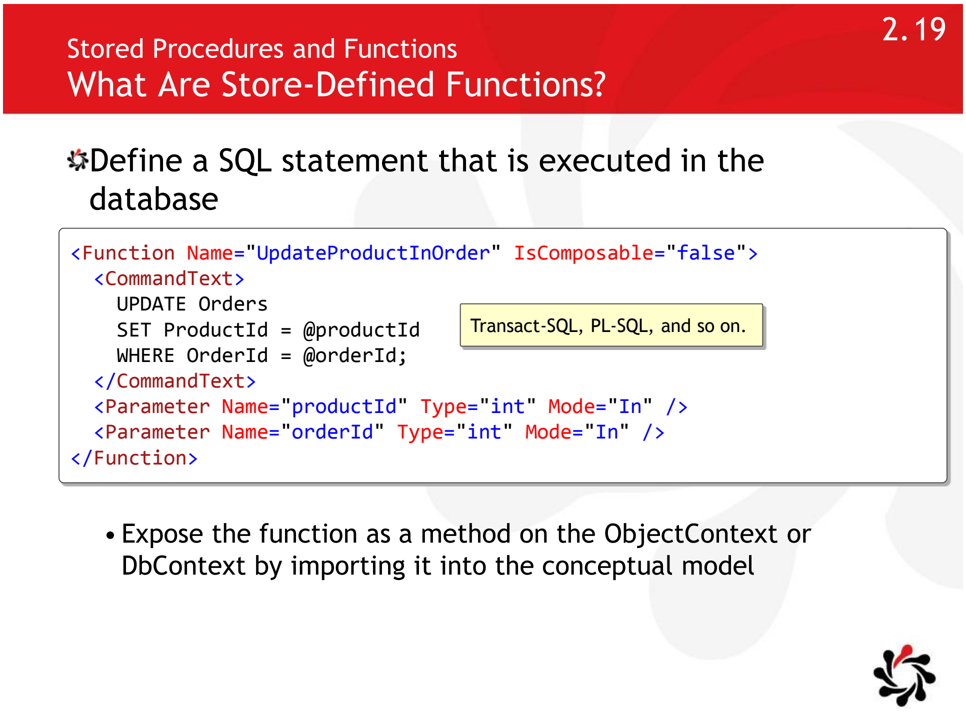
- MSL

```
<EntityTypeMapping TypeName="AdventureWorksModel.Product">
  <ModificationFunctionMapping>
    <UpdateFunction
      FunctionName="AWModel.Store.uspUpdateContact">
      <ScalarProperty Name="ProductID"
        ParameterName="ProductID" Version="Current" />
```

  - Version can be Current or Original

- SSDL

```
<Function Name="uspUpdateProduct" ... Schema="dbo">
  <Parameter Name="ProductID" Type="int" Mode="In" />
```



2.19

Stored Procedures and Functions

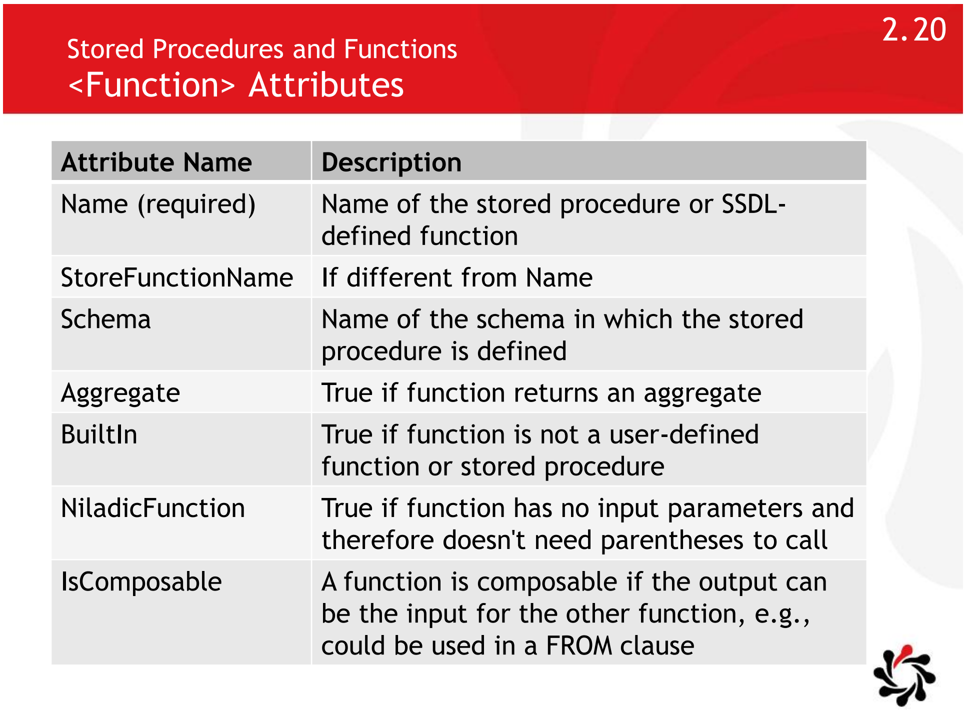
What Are Store-Defined Functions?

- Define a SQL statement that is executed in the database

```
<Function Name="UpdateProductInOrder" IsComposable="false">
  <CommandText>
    UPDATE Orders
    SET ProductId = @productId
    WHERE OrderId = @orderId;
  </CommandText>
  <Parameter Name="productId" Type="int" Mode="In" />
  <Parameter Name="orderId" Type="int" Mode="In" />
</Function>
```

Transact-SQL, PL-SQL, and so on.

  - Expose the function as a method on the ObjectContext or DbContext by importing it into the conceptual model



2.20

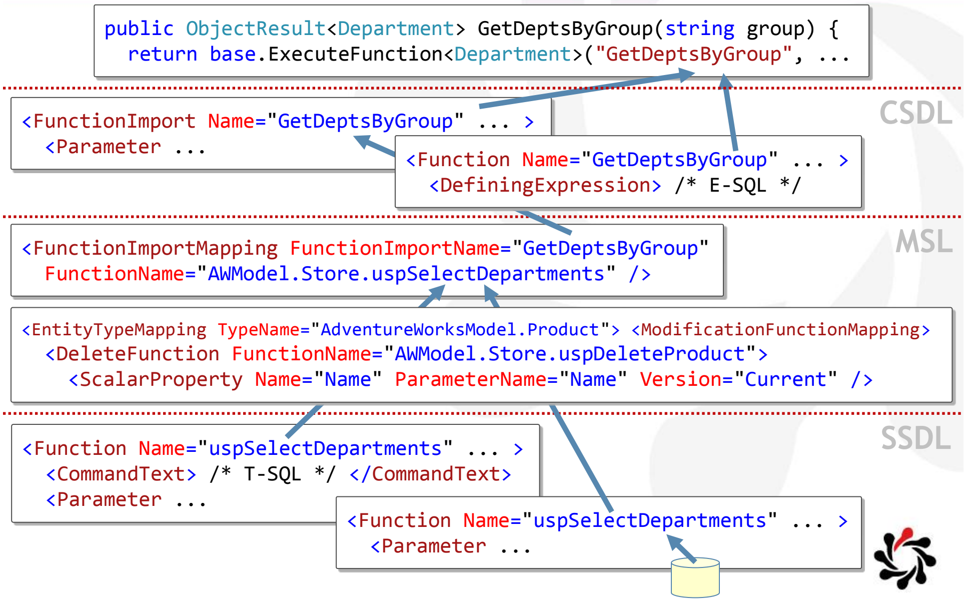
Stored Procedures and Functions

<Function> Attributes

Attribute Name	Description
Name (required)	Name of the stored procedure or SSDL-defined function
StoreFunctionName	If different from Name
Schema	Name of the schema in which the stored procedure is defined
Aggregate	True if function returns an aggregate
BuiltIn	True if function is not a user-defined function or stored procedure
NiladicFunction	True if function has no input parameters and therefore doesn't need parentheses to call
IsComposable	A function is composable if the output can be the input for the other function, e.g., could be used in a FROM clause

2.21

Stored Procedures and Functions
Summary



```
public ObjectResult<Department> GetDeptsByGroup(string group) {
  return base.ExecuteFunction<Department>("GetDeptsByGroup", ...
```

CSDL

```
<FunctionImport Name="GetDeptsByGroup" ... >
  <Parameter ...
```

```
<Function Name="GetDeptsByGroup" ... >
  <DefiningExpression> /* E-SQL */
```

MSL

```
<FunctionImportMapping FunctionImportName="GetDeptsByGroup"
  FunctionName="AWModel.Store.uspSelectDepartments" />
```

```
<EntityTypeMapping TypeName="AdventureWorksModel.Product"> <ModificationFunctionMapping>
  <DeleteFunction FunctionName="AWModel.Store.uspDeleteProduct">
    <ScalarProperty Name="Name" ParameterName="Name" Version="Current" />
```

SSDL

```
<Function Name="uspSelectDepartments" ... >
  <CommandText> /* T-SQL */ </CommandText>
  <Parameter ...
```

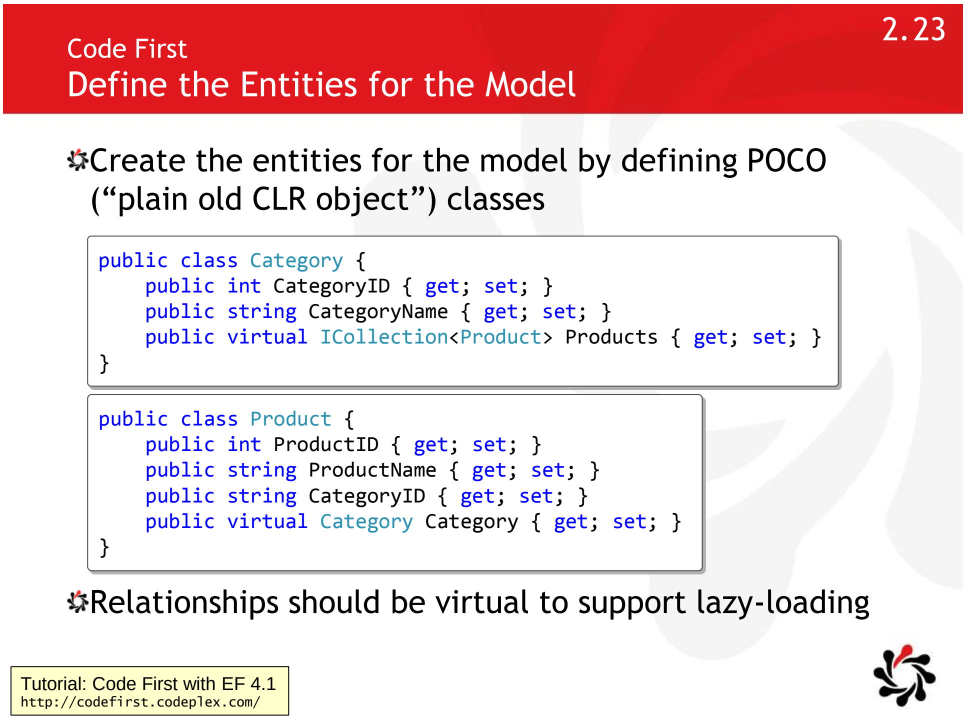
```
<Function Name="uspSelectDepartments" ... >
  <Parameter ...
```

2.22

Code First
What is Microsoft ADO.NET Entity Framework 4.1?

- aka “Magic Unicorn Edition” for VS2010 and later
- EF 4.1 introduces two new features
  - The **DbContext API** is a simplified abstraction over ObjectContext and a number of other types
  - **Code First** is a new development pattern that provides an alternative to the Database First and Model First patterns
- Code First is focused around defining your model using .NET classes
  - These classes can then be mapped to an existing database or be used to generate a database schema
  - Additional configuration can be supplied using Data Annotations or via a fluent API

EF 4.1 Released
http://blogs.msdn.com/b/adonet/archive/2011/04/11/ef-4-1-released.aspx



2.23

Code First
Define the Entities for the Model

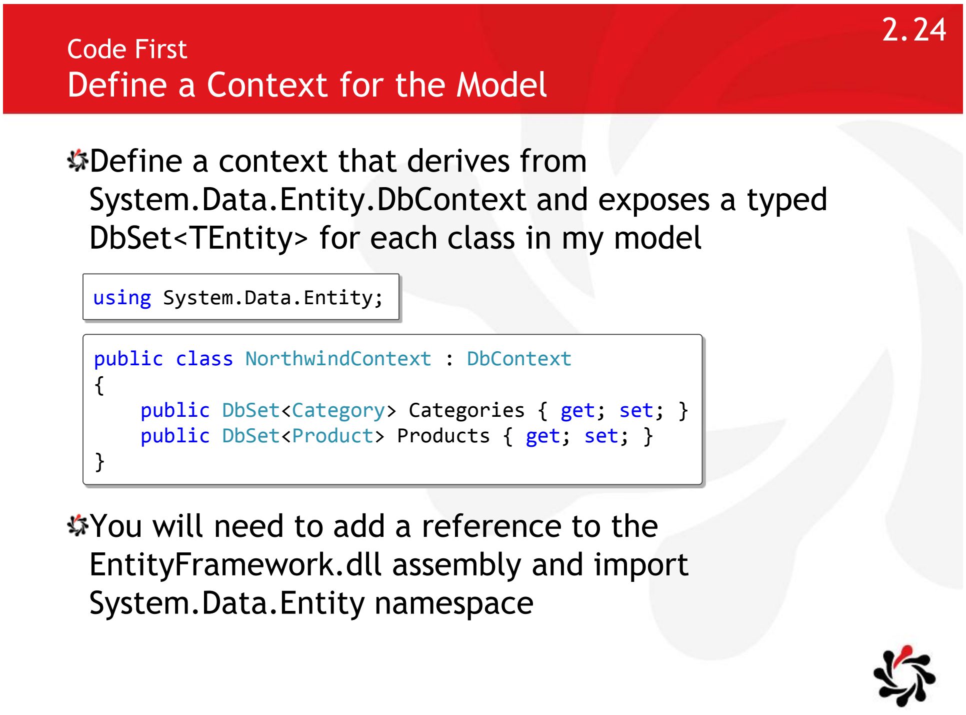
- Create the entities for the model by defining POCO (“plain old CLR object”) classes

```
public class Category {
    public int CategoryID { get; set; }
    public string CategoryName { get; set; }
    public virtual ICollection<Product> Products { get; set; }
}
```

```
public class Product {
    public int ProductID { get; set; }
    public string ProductName { get; set; }
    public string CategoryID { get; set; }
    public virtual Category Category { get; set; }
}
```

- Relationships should be virtual to support lazy-loading

Tutorial: Code First with EF 4.1
http://codefirst.codeplex.com/



2.24

Code First
Define a Context for the Model

- Define a context that derives from System.Data.Entity.DbContext and exposes a typed DbSet<TEntity> for each class in my model

```
using System.Data.Entity;
```

```
public class NorthwindContext : DbContext
{
    public DbSet<Category> Categories { get; set; }
    public DbSet<Product> Products { get; set; }
}
```

- You will need to add a reference to the EntityFramework.dll assembly and import System.Data.Entity namespace

2.25

Code First

Mapping to an Existing Database

- The easiest way to point Code First to an existing database is to add a .config connection string with the same name as your derived DbContext

```
<connectionStrings>
  <add name="NorthwindContext"
       providerName="System.Data.SqlClient"
       connectionString="Data Source=.\SQLEXPRESS;
                         Initial Catalog=Northwind;
                         Integrated Security=true;"
       />
</connectionStrings>
```

2.26

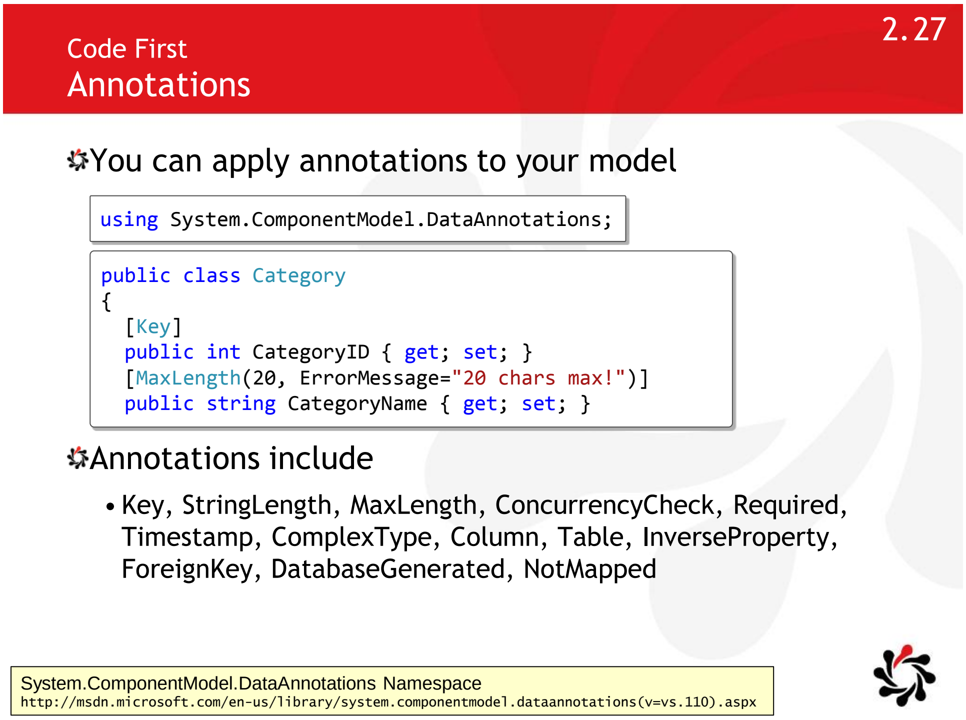
Code First

Modifying Data

- Use the DbContext

```
using (var db = new NorthwindContext())
{
  var food = new Category { CategoryName = "Foods" };
  db.Categories.Add(food);
  int recordsAffected = db.SaveChanges();
}
```

- If you do not specify a connection string for an existing database then DbContext by convention creates a database for you on localhost\SQLEXPRESS
  - The database will be named after the fully qualified name of your derived context



2.27

Code First
Annotations

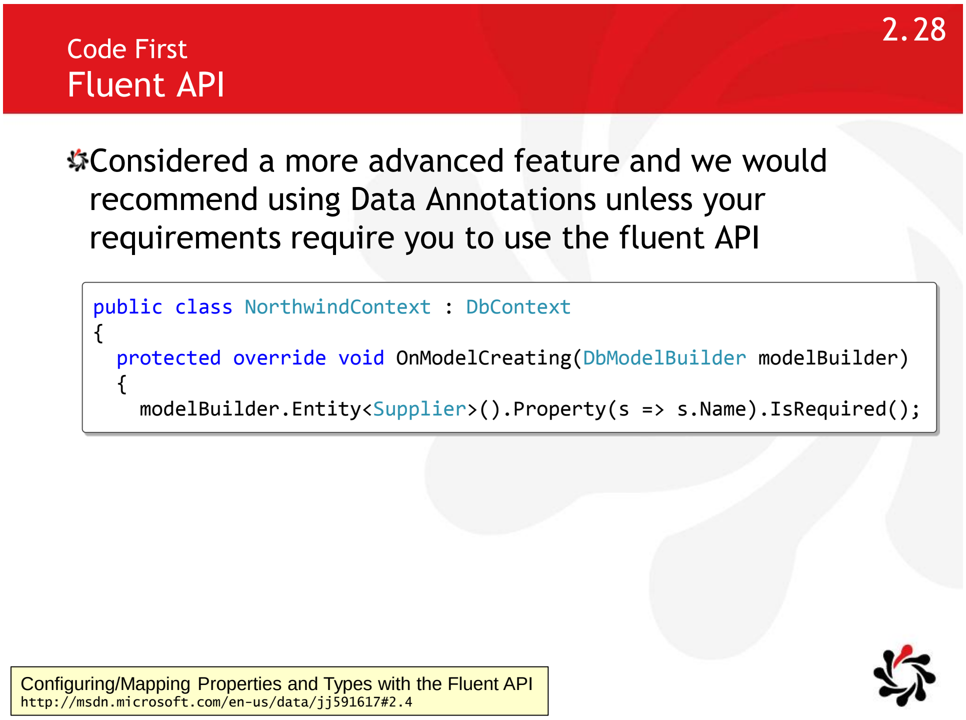
- You can apply annotations to your model

```
using System.ComponentModel.DataAnnotations;
```

```
public class Category
{
  [Key]
  public int CategoryID { get; set; }
  [MaxLength(20, ErrorMessage="20 chars max!")]
  public string CategoryName { get; set; }
```

- Annotations include
  - Key, StringLength, MaxLength, ConcurrencyCheck, Required, Timestamp, ComplexType, Column, Table, InverseProperty, ForeignKey, DatabaseGenerated, NotMapped

System.ComponentModel.DataAnnotations Namespace
http://msdn.microsoft.com/en-us/library/system.componentmodel.dataannotations(v=vs.110).aspx



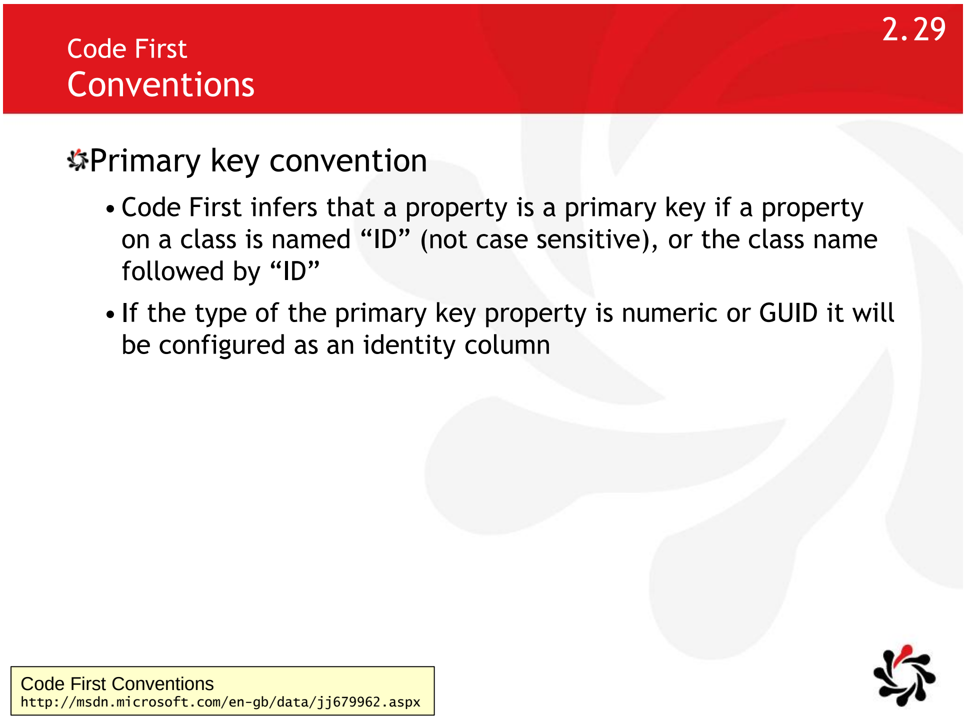
2.28

Code First
Fluent API

- Considered a more advanced feature and we would recommend using Data Annotations unless your requirements require you to use the fluent API

```
public class NorthwindContext : DbContext
{
  protected override void OnModelCreating(DbModelBuilder modelBuilder)
  {
    modelBuilder.Entity<Supplier>().Property(s => s.Name).IsRequired();
```

Configuring/Mapping Properties and Types with the Fluent API
http://msdn.microsoft.com/en-us/data/jj591617#2.4



2.29

Code First
Conventions

Primary key convention

- Code First infers that a property is a primary key if a property on a class is named “ID” (not case sensitive), or the class name followed by “ID”
- If the type of the primary key property is numeric or GUID it will be configured as an identity column

Code First Conventions
http://msdn.microsoft.com/en-gb/data/jj679962.aspx

2.30

Code First
Custom Conventions

Custom primary key convention

- Now any property in our model named Key will be configured as the primary key of whatever entity its part of

```
public class ProductContext : DbContext
{
    static ProductContext()
    {
        Database.SetInitializer(
            new DropCreateDatabaseIfModelChanges<ProductContext>());
    }
    public DbSet<Product> Products { get; set; }
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Properties()
                    .Where(p => p.Name == "Key")
                    .Configure(p => p.IsKey());
```

Custom Code First Conventions (EF6 onwards)
http://msdn.microsoft.com/en-gb/data/jj819164

Code First

Migration Support

2.31

- For example, if you wanted to add a new column to a Blogs table called Url

```
public partial class AddBlogUrl : DbMigration
{
  public override void Up()
  {
    AddColumn("Blogs", "Url", c => c.String());
  }
  public override void Down()
  {
    DropColumn("Blogs", "Url");
  }
}
```

- Not on the list of exam 70-487 objectives

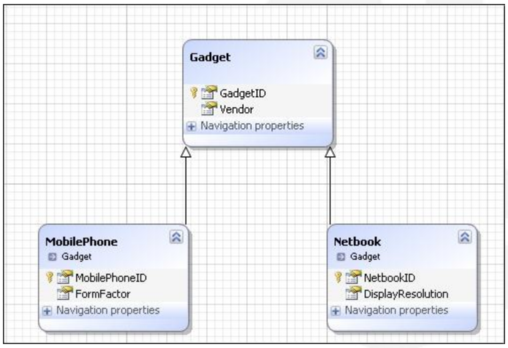
EF 4.3 Released
http://blogs.msdn.com/b/adonet/archive/2012/02/09/ef-4-3-released.aspx

Inheritance Hierarchies

Table-per-type vs. table-per-hierarchy

2.32

- If we start with a model



- The database schema can be implemented in two ways
  - Table-per-hierarchy (TPH)
  - Table-per-type (TPT): default when generating database from model (install EF Power Pack for more choices)

2.33

Inheritance Hierarchies

Table-per-hierarchy (TPH)

Characteristics

- One table with discriminator column

GadgetId	Discriminator	Vendor	FormFactor	DisplayResolution
1	M	Nokia	Monoblock	NULL
2	N	Sony	NULL	1024*768

Advantages

- High performance of CRUD operations
- Minimum number of tables in the database

Disadvantages

- Data redundancy which leads to the data integrity violation if data was changed manually
- Complexity of adding and deleting new entities because it is necessary to add or delete columns to/from the result table

2.34

Inheritance Hierarchies

Table-per-type (TPT)

GadgetId	Vendor
1	Nokia
2	Sony

Characteristics

- One table for each entity

MobilePhoneId	FormFactor
1	Monoblock

NetbookId	Resolution
2	1280*768

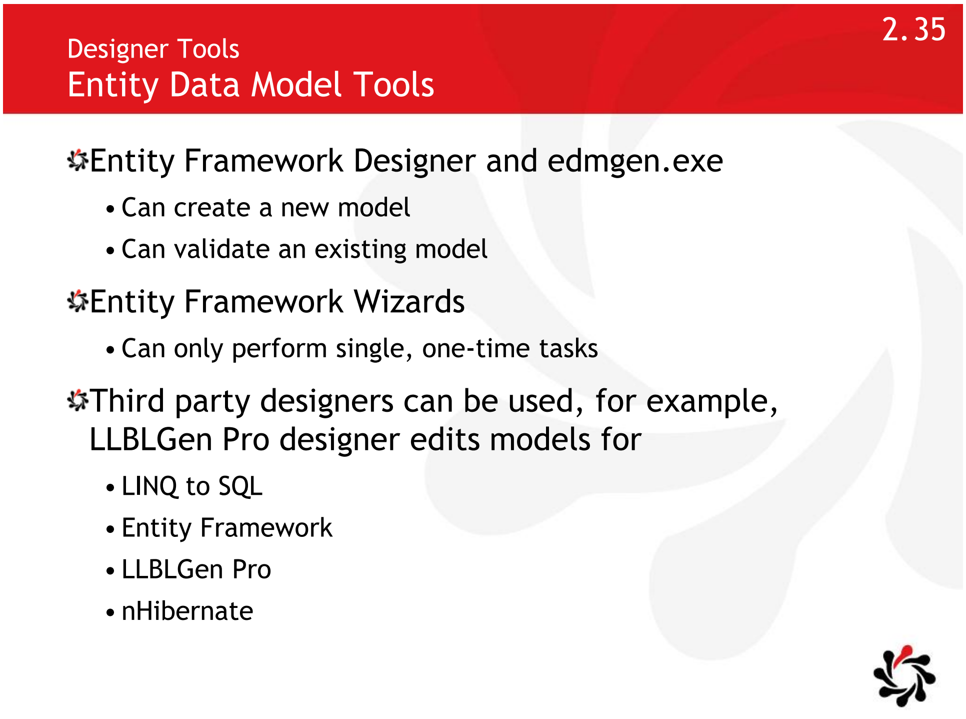
Advantages

- Data integrity is preserved with no data redundancy
- Flexibility in object model modification

Disadvantages

- The speed of the CRUD operation decreases when the number of classes in the hierarchy grows
- A large number of tables in the database

Entity Framework Mapping Scenarios
http://msdn.microsoft.com/en-us/library/cc716779.aspx

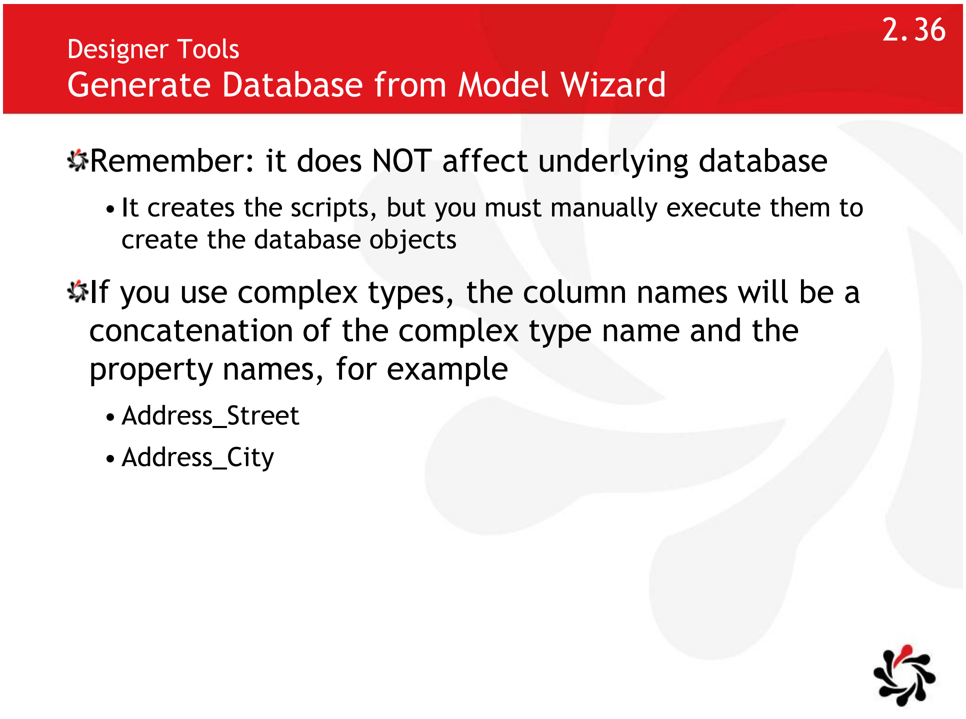


2.35

Designer Tools

Entity Data Model Tools

- Entity Framework Designer and edmgen.exe
  - Can create a new model
  - Can validate an existing model
- Entity Framework Wizards
  - Can only perform single, one-time tasks
- Third party designers can be used, for example, LLBLGen Pro designer edits models for
  - LINQ to SQL
  - Entity Framework
  - LLBLGen Pro
  - nHibernate



2.36

Designer Tools

Generate Database from Model Wizard

- Remember: it does NOT affect underlying database
  - It creates the scripts, but you must manually execute them to create the database objects
- If you use complex types, the column names will be a concatenation of the complex type name and the property names, for example
  - Address_Street
  - Address_City

2.37

Designer Tools

Comparing ObjectContext and DbContext

- DbContext is simply a wrapper around ObjectContext
  - For any DbContext instance, cast to IObjectContextAdapter interface and read the ObjectContext property

```
var db = new NorthwindEntities(); // a DbContext
var oc = (db as IObjectContextAdapter).ObjectContext;
```

```
ObjectContext
.AddObject("entityset", entity)
.DeleteObject("entityset", entity)
ObjectQuery.ToTraceString()
.ExecuteStoreCommand("T-SQL")
.ExecuteStoreQuery("T-SQL")
.LoadProperty(customer, "Orders")

.ContextOptions.LazyLoadingEnabled
.ContextOptions.ProxyCreationEnabled
```

```
DbContext
.DbSet.Add(entity)
.DbSet.Remove(entity)
DbQuery.ToString()
.Database.ExecuteSqlCommand("T-SQL")
.Database.SqlQuery("T-SQL")
.Entry(customer).Collection(
  "Orders").Load()
.Configuration.LazyLoadingEnabled
.Configuration.ProxyCreationEnabled
```

DbContext Class
http://msdn.microsoft.com/en-us/library/system.data.entity.dbcontext(v=vs.103).aspx

2.38

Designer Tools

DbSet<T>, DbQuery<T>, and IQueryable<T>

- DbContext uses DbSet<T> for the entity sets

```
public class NorthwindEntities : DbContext { ...
  public DbSet<Product> Products { ...
```

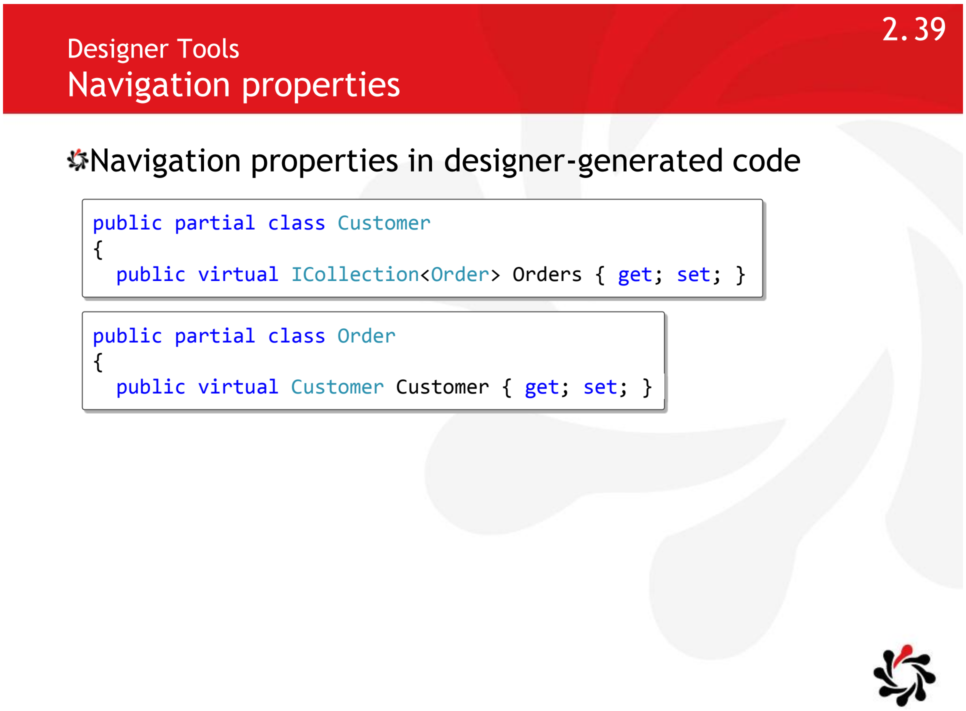
- DbSet<T> inherits from DbQuery<T> and implements some common interfaces

```
public class DbSet<TEntity> : DbQuery<TEntity>, IDbSet<TEntity>,
  IQueryable<TEntity>, IEnumerable<TEntity>, IQueryable, IEnumerable
  where TEntity : class
```

- Use ToString to see the T-SQL (or whatever)

```
string TSQL = query.ToString();
```

DbQuery<TResult> Class
http://msdn.microsoft.com/en-us/library/gg696530(v=vs.103).aspx



2.39

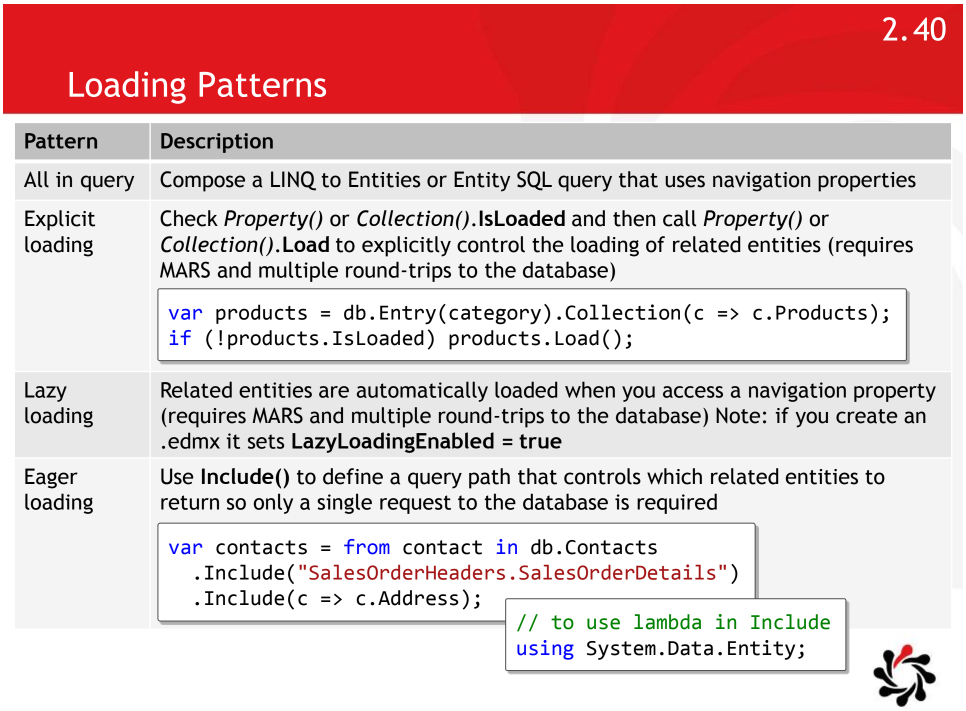
Designer Tools

Navigation properties

Navigation properties in designer-generated code

```
public partial class Customer
{
  public virtual ICollection<Order> Orders { get; set; }
```

```
public partial class Order
{
  public virtual Customer Customer { get; set; }
```



2.40

Loading Patterns

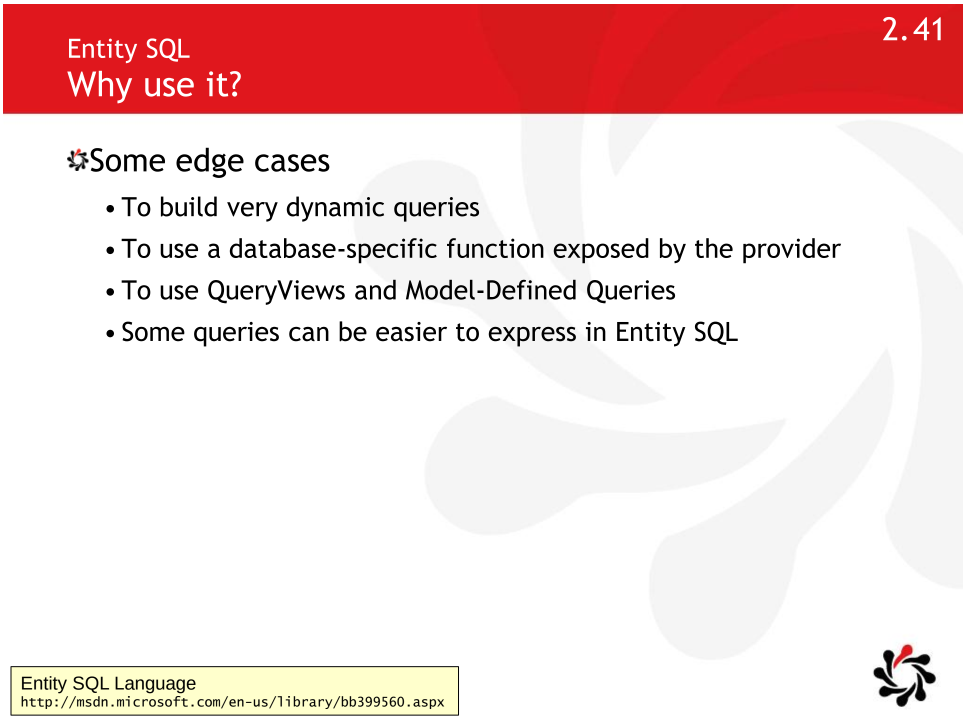
Pattern	Description
All in query	Compose a LINQ to Entities or Entity SQL query that uses navigation properties

Explicit loading	Check *Property()* or *Collection()*.**IsLoaded** and then call *Property()* or *Collection()*.**Load** to explicitly control the loading of related entities (requires MARS and multiple round-trips to the database)
	`var products = db.Entry(category).Collection(c => c.Products);`
	`if (!products.IsLoaded) products.Load();`

Lazy loading	Related entities are automatically loaded when you access a navigation property (requires MARS and multiple round-trips to the database) Note: if you create an .edmx it sets **LazyLoadingEnabled = true**

Eager loading	Use **Include()** to define a query path that controls which related entities to return so only a single request to the database is required
	`var contacts = from contact in db.Contacts`
	`  .Include("SalesOrderHeaders.SalesOrderDetails")`
	`  .Include(c => c.Address);`

```
// to use lambda in Include
using System.Data.Entity;
```

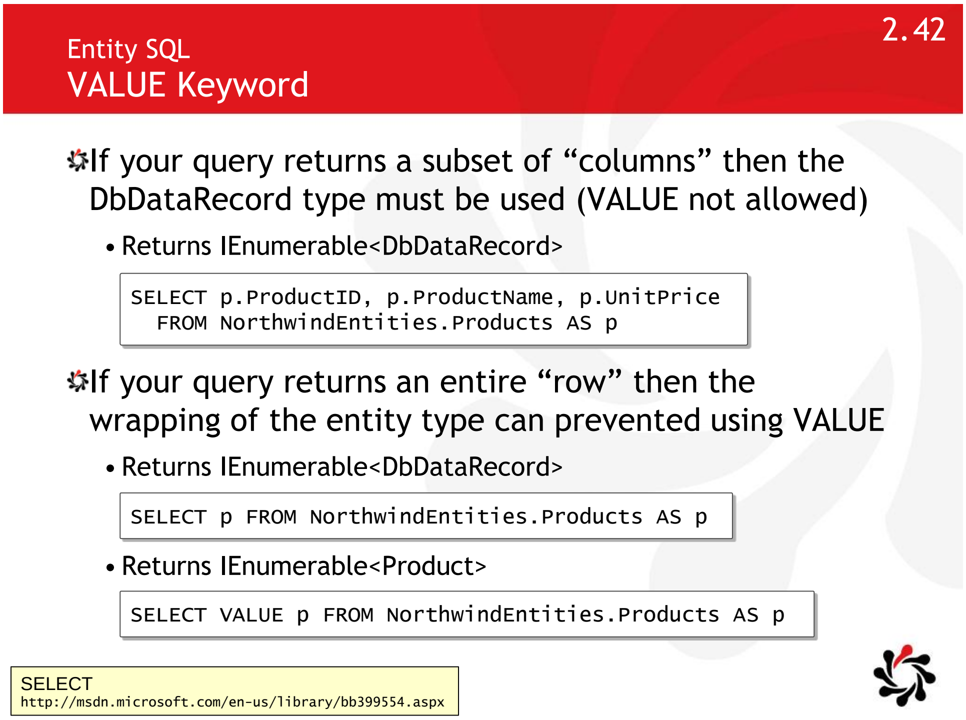


2.41

Entity SQL
Why use it?

- Some edge cases
  - To build very dynamic queries
  - To use a database-specific function exposed by the provider
  - To use QueryViews and Model-Defined Queries
  - Some queries can be easier to express in Entity SQL

Entity SQL Language
http://msdn.microsoft.com/en-us/library/bb399560.aspx



2.42

Entity SQL
VALUE Keyword

- If your query returns a subset of “columns” then the DbDataRecord type must be used (VALUE not allowed)
  - Returns IEnumerable<DbDataRecord>

```
SELECT p.ProductID, p.ProductName, p.UnitPrice
  FROM NorthwindEntities.Products AS p
```

- If your query returns an entire “row” then the wrapping of the entity type can prevented using VALUE
  - Returns IEnumerable<DbDataRecord>

```
SELECT p FROM NorthwindEntities.Products AS p
```

  - Returns IEnumerable<Product>

```
SELECT VALUE p FROM NorthwindEntities.Products AS p
```

SELECT
http://msdn.microsoft.com/en-us/library/bb399554.aspx

2.43

Entity SQL
Projections and the VALUE Keyword

- If your query returns a single “column” then simple types such as String can be used by using VALUE
  - Returns IEnumerable<DbDataRecord>

```
SELECT p.ProductName FROM NorthwindEntities.Products AS p
```

  - Returns IEnumerable<string>

```
SELECT VALUE p.ProductName FROM NorthwindEntities.Products AS p
```

2.44

Entity SQL
it

- In a query builder method you refer to the current ObjectQuery command by using an alias
- By default, the string “it” is the alias that represents the current command

```
var db = new NorthwindEntities(); // a DbContext
var oc = (db as IObjectContextAdapter).ObjectContext;
```

```
ObjectQuery<Product> productQuery =
  oc.Products.Where("it.StandardCost > @cost",
  new ObjectParameter("cost", 23));
```

- You can use a different name

```
productQuery.Name = "product";
ObjectQuery<Product> filteredProduct = productQuery
  .OrderBy("product.ProductID");
```

Entity SQL
Paging with TOP, LIMIT, SKIP

2.45

- The TOP sub-clause specifies that only the first set of rows will be returned

```
SELECT VALUE TOP(10) contact
FROM AdventureWorksEntities.Contacts AS contact
```

- Physical paging can be performed by using LIMIT and SKIP sub-clauses in ORDER BY clause
  - LIMIT or SKIP cannot be used separately from ORDER BY

```
SELECT VALUE p
FROM AdventureWorksEntities.Products AS p
ORDER BY p.ListPrice SKIP 50 LIMIT 10
```

  - An Entity SQL query is invalid if both the TOP modifier and the SKIP sub-clause are present in the same expression

Entity SQL
Entity Set Operators

2.46

- EXCEPT
  - Distinct values from the left that are not also from the right
- INTERSECT
  - Distinct values from both left and right sides
- UNION (and UNION ALL)
  - Distinct values combined from left and right sides (UNION ALL includes duplicates)
- IN
  - Determines whether a value matches any value in a collection
- EXISTS
  - Determines if a collection is empty

Comparison Semantics
http://msdn.microsoft.com/en-us/library/bb896323

2.47

Entity SQL

CREATEREF, DEREF, REF

- A REF is a reference to a *persisted* entity
  - It’s a pointer to an entity so you have to de-reference it (using the DEREF keyword or by referencing a property) to get to the actual entity
  - To get one of these REFs you can either start with an entity (the REF keyword) or you can make one from a key value (the CREATEREF keyword)
- To create references to the top 5 rows of Customers

```
SELECT TOP(5) REF(c) FROM NorthwindContext.Customers AS c
```

- Why?
  - A REF is a ‘lightweight’ entity in which we don’t need to spend resources in creating and maintaining the full entity state/values until it is really necessary

2.48

EntityClient

Using the EntityClient Data Provider

- Similar to any other .NET data provider

```
using System.Data.Entity.Core.EntityClient;
```

  - EntityConnection can load a connection string from .config

```
var con = new EntityConnection("name=NW");
con.Open();
```

  - EntityCommand uses Entity SQL

```
var cmd = new EntityCommand();
cmd.Connection = con;
cmd.CommandText ="SELECT VALUE p.ProductName FROM Products AS p";
```

  - EntityDataReader must be opened with sequential access

```
var reader = cmd.ExecuteReader(CommandBehavior.SequentialAccess);
```

EntityClient Provider for the Entity Framework
http://msdn.microsoft.com/en-us/library/bb738561.aspx

2.49

EntityClient

Query Plan Caching

- By default, Entity SQL queries are cached to improve performance if you re-execute the same statement
- The strings must be identical (case-sensitive) but can use parameters
- To disable, set EntityCommand object’s EnablePlanCaching property to false

Query Plan Caching
http://msdn.microsoft.com/en-us/library/bb738562.aspx

2.50

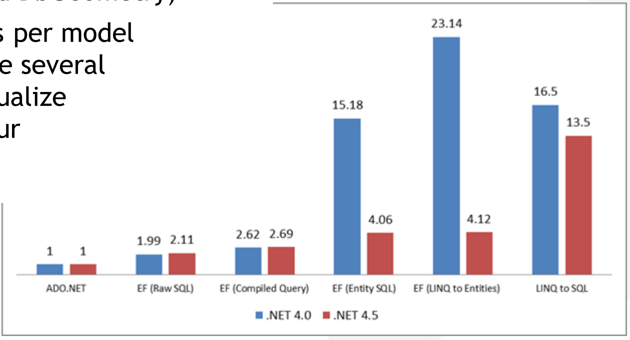
Querying Summary

When To Use…

- LINQ to Entities
  - Use this most of the time
  - Materializes data into entities tracked by DbContext
- Entity SQL with ObjectQuery (use ObjectContext)
  - When you need a feature supported by Entity SQL that is not supported by LINQ to Entities
  - Materializes data into entities tracked by ObjectContext
  - Set MergeOption.NoTracking to improve performance
- Entity SQL with EntityClient
  - For best read-only performance because it does NOT materialize data into entities that are change tracked

Performance
Comparing EF4 (2010) and EF5 (2012)

- Deployed with .NET Framework 4.5
  - Automatic compilation of LINQ to Entities queries
  - Support for: enums, table-valued functions, spatial data types (DbGeography and DbGeometry)
  - Multiple-diagrams per model allows you to have several diagrams that visualize subsections of your overall model
  - Shapes can have colour applied



Entity Framework 5.0 Performance Improvements
http://blogs.msdn.com/b/adonet/archive/2012/02/14/sneak-preview-entity-framework-5-0-performance-improvements.aspx

Transactions
ACID

- Atomic aka “all or nothing”
  - Groups multiple actions into a single, indivisible transaction
- Consistent
  - Leave the system in a consistent state after the transaction, for example, if we credit $100, we must also debit $100
- Isolated (different levels)
  - Apply locks while the transaction is occurring to isolate it
- Durable (or volatile)
  - A durable resource manager is one that can recover a failed transaction (for example, SQL Server uses logs)
  - A volatile resource manager cannot recover

2.53

Transactions

Durable and Volatile

- The durability of a resource manager refers to whether the resource manager supports failure recovery
  - If it supports failure recovery, it persists data to durable storage during Phase1 (prepare) such that if it goes down, it can re-enlist in the transaction upon recovery and perform the proper actions based on notifications from the transaction manager
- In order for a resource to participate in a transaction, it must enlist in the transaction
  - You use the EnlistVolatile methods for volatile resources, and the EnlistDurable method for durable resources

Implementing a Resource Manager
http://msdn.microsoft.com/en-us/library/ms229975.aspx

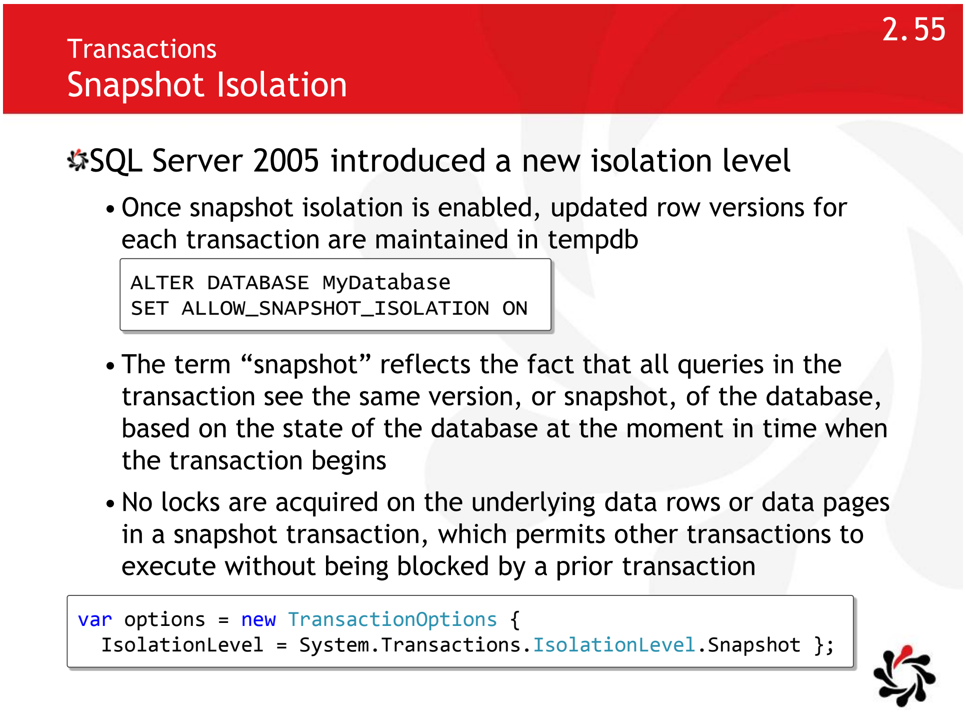
Volatile Resource Managers in .NET Bring Transactions to the Common Type
http://msdn.microsoft.com/en-us/magazine/cc163688.aspx

2.54

Transactions

Isolation Levels

- ReadUncommitted (weakest isolation)
  - Allows dirty reads, non-repeatable reads, and phantom data
- ReadCommitted
  - Allows non-repeatable reads and phantom data
- RepeatableRead
  - Allows phantom data
- Serializable (strongest isolation)
  - Completely isolates the transaction until it completes
- Defaults
  - SQL Server & ADO.NET “Classic” default to ReadCommitted
  - TransactionScope defaults to Serializable



2.55

Transactions

Snapshot Isolation

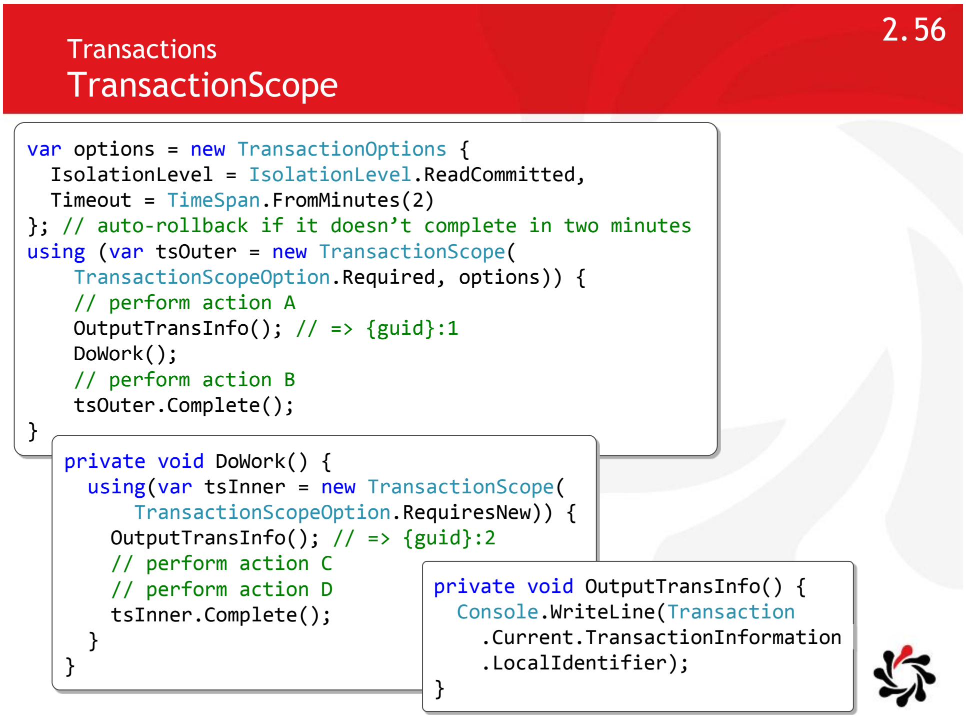
SQL Server 2005 introduced a new isolation level

- Once snapshot isolation is enabled, updated row versions for each transaction are maintained in tempdb

```
ALTER DATABASE MyDatabase
SET ALLOW_SNAPSHOT_ISOLATION ON
```

- The term “snapshot” reflects the fact that all queries in the transaction see the same version, or snapshot, of the database, based on the state of the database at the moment in time when the transaction begins
- No locks are acquired on the underlying data rows or data pages in a snapshot transaction, which permits other transactions to execute without being blocked by a prior transaction

```
var options = new TransactionOptions {
  IsolationLevel = System.Transactions.IsolationLevel.Snapshot };
```



2.56

Transactions

TransactionScope

```
var options = new TransactionOptions {
  IsolationLevel = IsolationLevel.ReadCommitted,
  Timeout = TimeSpan.FromMinutes(2)
}; // auto-rollback if it doesn’t complete in two minutes
using (var tsOuter = new TransactionScope(
    TransactionScopeOption.Required, options)) {
    // perform action A
    OutputTransInfo(); // => {guid}:1
    DoWork();
    // perform action B
    tsOuter.Complete();
}
```

```
private void DoWork() {
  using(var tsInner = new TransactionScope(
      TransactionScopeOption.RequiresNew)) {
    OutputTransInfo(); // => {guid}:2
    // perform action C
    // perform action D
    tsInner.Complete();
  }
}
```

```
private void OutputTransInfo() {
  Console.WriteLine(Transaction
    .Current.TransactionInformation
    .LocalIdentifier);
}
```

2.57

Transactions

TransactionScopeOption

- Required (default)
  - Uses the ambient transaction if one already exists, otherwise, it creates a new transaction before entering the scope
  - Use Transaction.Current to get the ambient transaction
- RequiresNew
  - A new transaction is always created for the scope
- Suppress
  - The ambient transaction context is suppressed when creating the scope and all operations within the scope are done without an ambient transaction context

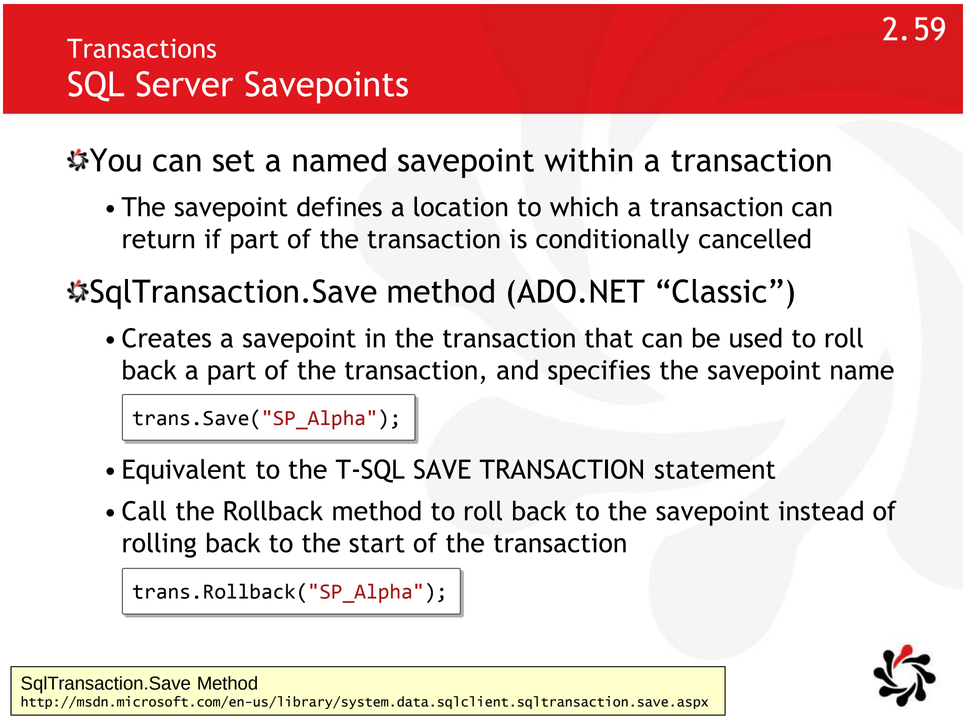
```
using (var tsPreventRollback = new TransactionScope(
    TransactionScopeOption.Suppress)) {
    // call method that could throw an exception
}
```

2.58

Transactions

DependentTransaction class

- A DependentTransaction is a clone of a Transaction object created using the DependentClone method
  - Its sole purpose is to allow the application to come to rest and guarantee that the transaction cannot commit while work is still being performed on the transaction (for example, on a worker thread)
  - When the work done within the cloned transaction is finally complete and ready to be committed, it can inform the creator of the transaction using the Complete method
- The DependentCloneOption enumeration is used to determine the behavior on commit
  - BlockCommitUntilComplete
  - RollbackIfNotComplete



Transactions

SQL Server Savepoints

- You can set a named savepoint within a transaction
  - The savepoint defines a location to which a transaction can return if part of the transaction is conditionally cancelled
- SqlTransaction.Save method (ADO.NET “Classic”)
  - Creates a savepoint in the transaction that can be used to roll back a part of the transaction, and specifies the savepoint name

```
trans.Save("SP_Alpha");
```

  - Equivalent to the T-SQL SAVE TRANSACTION statement
  - Call the Rollback method to roll back to the savepoint instead of rolling back to the start of the transaction

```
trans.Rollback("SP_Alpha");
```

SqlTransaction.Save Method
http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqltransaction.save.aspx

Module 3
Creating and Consuming ASP.NET Web API Services

Developing Windows Azure and Web Services

Updated 11th April 2014

ASP.NET Web API

Contents

Exam Topics: Design a Web API

❑ Define HTTP resources with HTTP actions
❑ Plan appropriate URI space, map URI space using routing
❑ Choose appropriate HTTP method (get, put, post, delete) to meet requirements
❑ Choose appropriate format (Web API formats) for responses to meet requirements
❑ Plan when to make HTTP actions asynchronous

Exam Topics: Implement a Web API

❑ Accept data in JSON format (in JavaScript, in an AJAX callback)
❑ Use content negotiation to deliver different data formats to clients
❑ Define actions and parameters to handle data binding
❑ Use HttpMessageHandler to process client requests and server responses
❑ Implement dependency injection, along with the dependency resolver, to create more flexible applications
❑ Implement action filters and exception filters to manage controller execution
❑ Implement asynchronous and synchronous actions
❑ Implement streaming actions

Exam Topics: Consume Web API web services

❑ Consume Web API services by using HttpClient synchronously and asynchronously
❑ Send and receive requests in different formats (JSON/HTML/etc.)

3.3

HTTP and REST

SOAP Services versus RESTful Services

- When developers talk about “web” services, they usually mean SOAP services, for example, ASMX files
  - Actually, SOAP services can use any protocol, since SOAP defines a message format, not an architectural style, which is why WCF supports so many bindings
- RESTful services are true web services since they are built on the architecture of the web
- Proponents of RESTful services are sometimes called RESTafarians and are often quite passionate about how “evil” SOAP is and how “beautiful” REST is
- In WCF 3.5 Microsoft embraced REST so that you can use WCF to create both SOAP and RESTful services
  - In ASP.NET 4.5 Microsoft added Web API which is even better

3.4

HTTP and REST

21st Century Service-Oriented Architecture

- In 2000 Roy Fielding wrote a doctoral dissertation
  - The web is the world’s largest and most scalable distributed application platform
  - He described the architecture of the web and distilled from it an architectural style based on the factors that led to its success
  - He named this style REST and suggested its use to build services
- WCF isn’t tied to SOAP so Microsoft was able to quickly embrace REST once its simple power was understood
- RESTafarians believe that REST should be your first choice when building services

3.5

HTTP and REST
Architecture of the Web

- Principles
  - Addressable resources (URIs)
  - Uniform interface for interacting with resources (HTTP verbs: GET, POST, DELETE, PUT, etc.)
  - Standard resource formats (HTML, JPEG, XML, JSON, etc.)
  - Statelessness between clients and servers (provides scalability and manageability)
  - Hyperlinking for navigation between resources (relationships)
- GET
  - The cacheability of the GET verb contributes to the scalability of the web
  - GETs are also considered “safe” in the sense that they should not cause side effects i.e. they don’t change resources

3.6

HTTP and REST
SOAP

- SOAP doesn’t follow the architecture of the web at all
  - Rather than URIs, SOAP uses *actions*, which are a thin veneer over method calls
  - SOAP services usually have only one URI and many different actions
  - SOAP is really an interoperable cross-platform remote procedure call (RPC) system
- When using HTTP, SOAP only uses one HTTP verb, POST
  - POSTs cannot be cached, so it’s not as scalable as GET
- But SOAP wasn’t designed for the web and goes out of its way to be protocol independent

3.7

HTTP and REST
REST versus SOAP

- REST services combine nouns (e.g. resources defined by URIs) with verbs (e.g. GET, DELETE)

```
PUT /AW.svc/Products(123)
Host: http://localhost:801
Content-Length: 223
Content-Type: application/json
```

```
{
  "ProductID" : "123",
  ...
```

- SOAP services use a message to contain the nouns (e.g. the payload in the body) with verbs (the action in the header)

```
<s:Envelope xmlns:s=" ... >
  <s:Header>
    <To>http://.../Sample.svc</To>
    <Action>AddProduct</Action>
```

```
<s:Body>
  <Product>
    <ProductID>123</ProductID>
    <ProductName>Fish</ProductName>
```

3.8

HTTP and REST
POST versus PUT

- “The actual function performed by the POST method is determined by the server” and “POST is designed to allow a uniform method to cover the following functions: […] Extending a database through an append operation”
- So POST can be used to insert and the server should respond with 201 (Created), or POST can be used for *any* meaning
- PUT “If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI”
- So PUT can be used to insert or update and the server should respond with either 201 (Created) or 200 (OK) or 204 (No content)

Method Definitions
http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

3.9

HTTP and REST

Designing the URIs

- Choose “common sense” URIs so developers can quickly work out how to access any resource and your service becomes almost “self-documenting”
  - Design your service API as if you were designing the URLs for a web site i.e. make them logical enough that an end user could work out how to use them if shown a few examples

Task	HTTP Method	Relative URI
Retrieve all entities	GET	/api/orders
Retrieve single entity	GET	/api/orders/*id*
Retrieve by custom	GET	/api/orders?category=*category*
Create new entity	POST	/api/orders
Update entity	PUT	/api/orders/*id*
Remove entity	DELETE	/api/orders/*id*

3.10

HTTP and REST

Uniform Interface

- With REST there is no need to design the semantics of the actions you want to perform because HTTP already defines them for you
  - GET (or HEAD): retrieves a resource (or without the body)
  - POST: Microsoft recommends this means create a new resource
  - PUT: Microsoft recommends this means replace an existing resource with a new version (idempotent[1])
  - PATCH: update a resource with a subset of new values
  - DELETE: remove a resource (idempotent[1])
  - CONNECT, TRANSFER, OPTIONS, TRACE
- The Atom protocol defines
  - MERGE: update a resource with a subset of new values

[1] Idempotent means that the effect of calling it many times is the same as calling it once

3.11

HTTP and REST

Resource Formats

- REST has no restrictions on resource formats
  - A REST service’s resource types are known as *media types*
- The media type is returned in the HTTP header as the Content-Type
  - XML: application/xml (text/xml is old standard)
  - Feeds: application/rss+xml or application/atom+xml
  - HTML and XHTML: text/html or application/xhtml+xml
  - JavaScript Object Notation: application/json (most popular media type because it is the most compact)
- ASP.NET Web API has built-in support for returning XML, JSON, and x-www-form-urlencoded, and you can add support for JSONP and custom formatters

Internet Media Types
http://en.wikipedia.org/wiki/Internet_media_type

3.12

ASP.NET Web API

Serializing Objects as XML, Atom, JSON

- To generate this JSON (or similar XML, Atom, etc.)

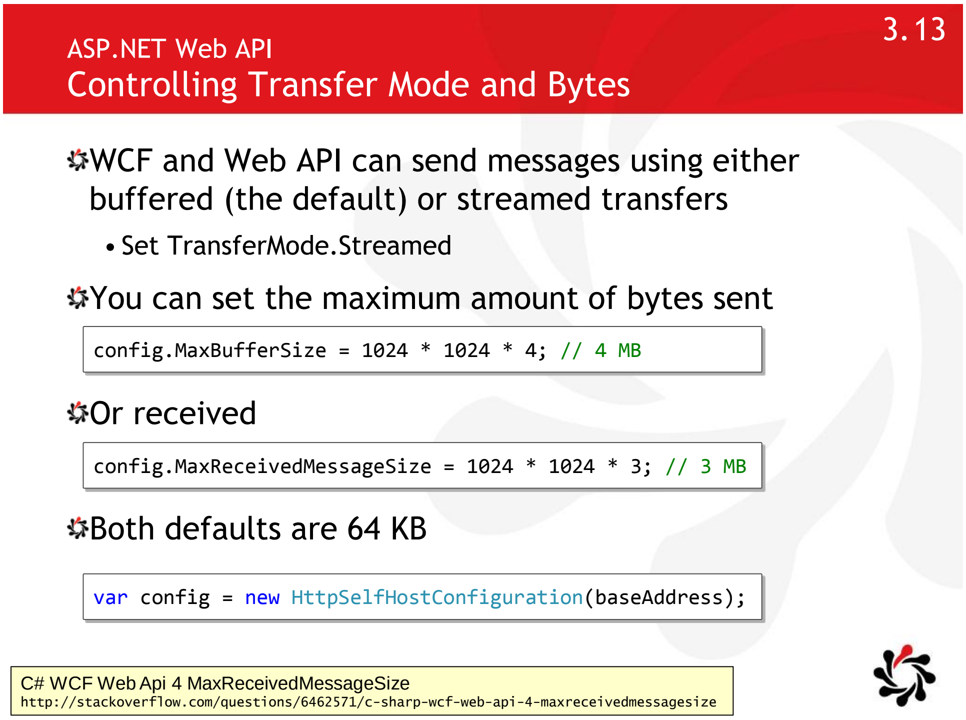
```
{"Name":"Alice","Age":23,"Pets":["Fido","Polly","Spot"]}
```

  - Return an anonymous type instance from an ApiController’s action method

```
public object Get()
{
  return new { Name = "Alice", Age = 23,
    Pets = new List<string> { "Fido", "Polly", "Spot" }
  };
}
```

  - Warning! The XML serializer that would be used with Chrome (because it adds Accept: XML) breaks with anonymous types so you must use a named class instead and mark with attributes

JSON and XML Serialization in ASP.NET Web API
http://www.asp.net/web-api/overview/formats-and-model-binding/json-and-xml-serialization



3.13

ASP.NET Web API

Controlling Transfer Mode and Bytes

- WCF and Web API can send messages using either buffered (the default) or streamed transfers
  - Set TransferMode.Streamed
- You can set the maximum amount of bytes sent

```
config.MaxBufferSize = 1024 * 1024 * 4; // 4 MB
```

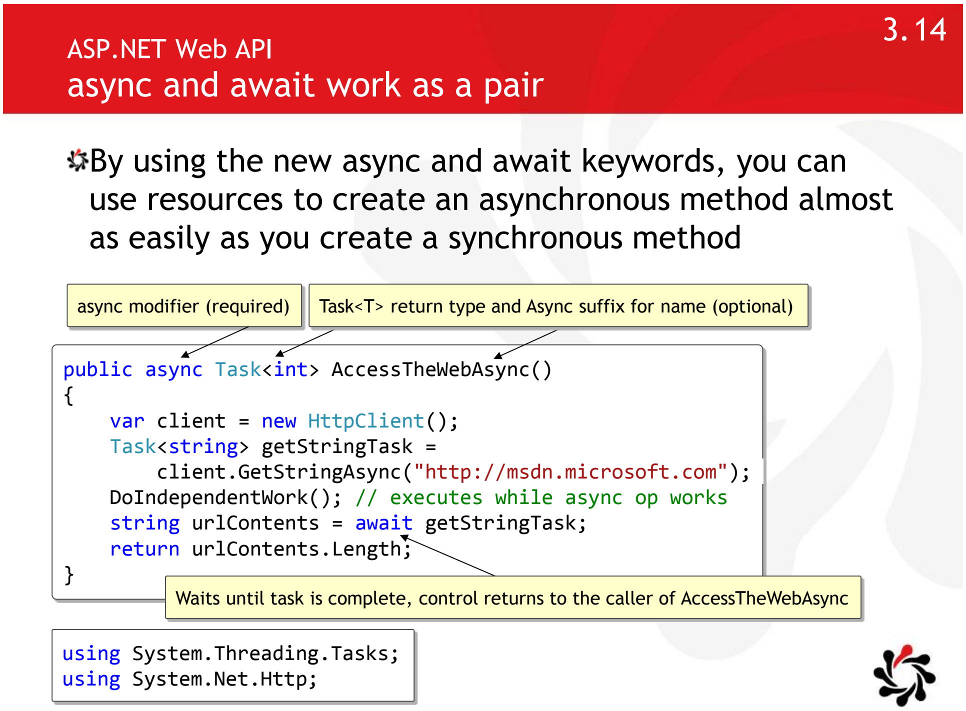
- Or received

```
config.MaxReceivedMessageSize = 1024 * 1024 * 3; // 3 MB
```

- Both defaults are 64 KB

```
var config = new HttpSelfHostConfiguration(baseAddress);
```

C# WCF Web Api 4 MaxReceivedMessageSize
http://stackoverflow.com/questions/6462571/c-sharp-wcf-web-api-4-maxreceivedmessagesize



3.14

ASP.NET Web API

async and await work as a pair

- By using the new async and await keywords, you can use resources to create an asynchronous method almost as easily as you create a synchronous method

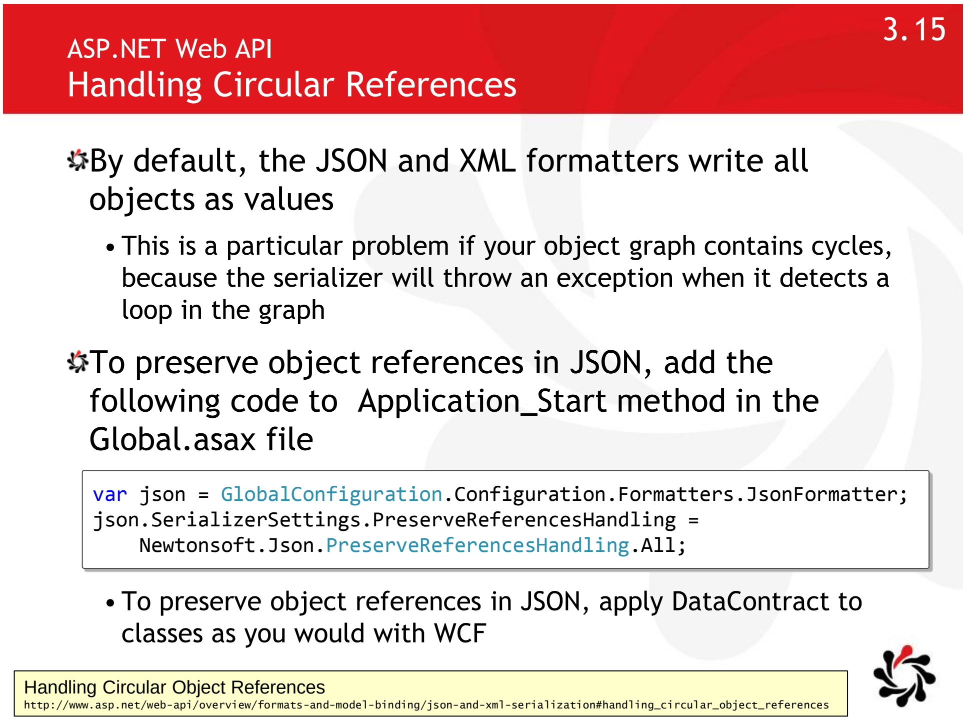
async modifier (required)

Task<T> return type and Async suffix for name (optional)

```
public async Task<int> AccessTheWebAsync()
{
    var client = new HttpClient();
    Task<string> getStringTask =
        client.GetStringAsync("http://msdn.microsoft.com");
    DoIndependentWork(); // executes while async op works
    string urlContents = await getStringTask;
    return urlContents.Length;
}
```

Waits until task is complete, control returns to the caller of AccessTheWebAsync

```
using System.Threading.Tasks;
using System.Net.Http;
```



3.15

ASP.NET Web API

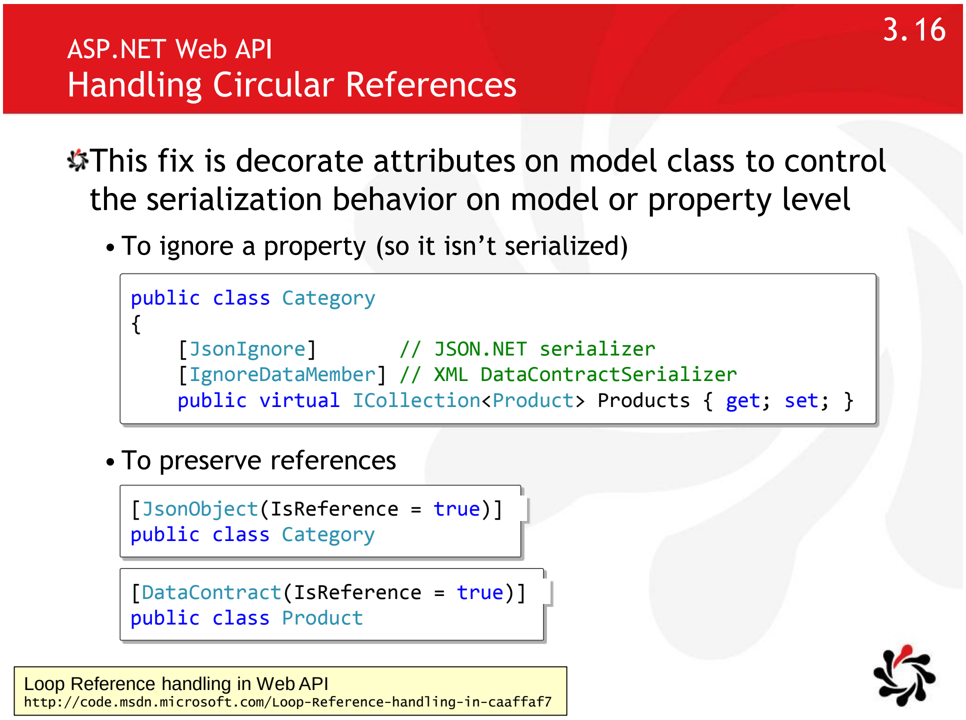
Handling Circular References

- By default, the JSON and XML formatters write all objects as values
  - This is a particular problem if your object graph contains cycles, because the serializer will throw an exception when it detects a loop in the graph
- To preserve object references in JSON, add the following code to Application_Start method in the Global.asax file

```
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
json.SerializerSettings.PreserveReferencesHandling =
    Newtonsoft.Json.PreserveReferencesHandling.All;
```

  - To preserve object references in JSON, apply DataContract to classes as you would with WCF

Handling Circular Object References
http://www.asp.net/web-api/overview/formats-and-model-binding/json-and-xml-serialization#handling_circular_object_references



3.16

ASP.NET Web API

Handling Circular References

- This fix is decorate attributes on model class to control the serialization behavior on model or property level
  - To ignore a property (so it isn’t serialized)

```
public class Category
{
    [JsonIgnore]       // JSON.NET serializer
    [IgnoreDataMember] // XML DataContractSerializer
    public virtual ICollection<Product> Products { get; set; }
```

  - To preserve references

```
[JsonObject(IsReference = true)]
public class Category
```

```
[DataContract(IsReference = true)]
public class Product
```

Loop Reference handling in Web API
http://code.msdn.microsoft.com/Loop-Reference-handling-in-caaffaf7

ASP.NET Web API

camelCase with JsonFormatter

3.17

- If you have this C# class

```
public class Person {
    public string FirstName { get; set; }
    public int Age { get; set; }
```

- The default serializer will generate this JSON

```
{"FirstName":"Bob","Age":25}
```

  - It feels awkward to work with JavaScript objects where the first letter of a property name is a capital letter

- The serializer will let us be smarter

```
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
json.SerializerSettings.ContractResolver =
    new CamelCasePropertyNamesContractResolver();
```

```
using Newtonsoft.Json.Serialization;
```

ASP.NET WebAPI Tip #3: camelCasing JSON
http://odetocode.com/blogs/scott/archive/2013/03/25/asp-net-webapi-tip-3-camelcasing-json.aspx

ASP.NET Web API

Returning Entities from a Model

3.18

- If you have an Entity Data Model and have created an ApiController using the scaffolding make sure you
  - Switch off proxy creation so that EF doesn't make DynamicProxy objects to track changes for you (they can't be serialized!)

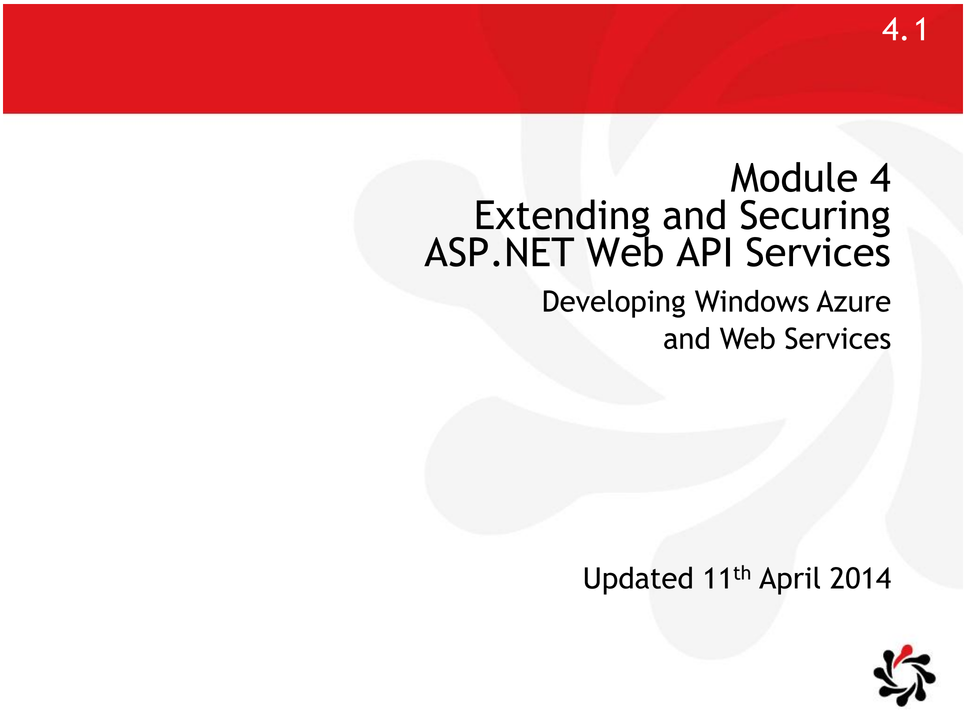
```
db.Configuration.ProxyCreationEnabled = false;
```

  - Switch off lazy loading so that EF doesn't automatically load related entities when serializing the objects you have chosen to return

```
db.Configuration.LazyLoadingEnabled = false;
```

  - Use explicit loading instead (call Include method in query)

```
var products = db.Products.Include(p => p.Category);
```

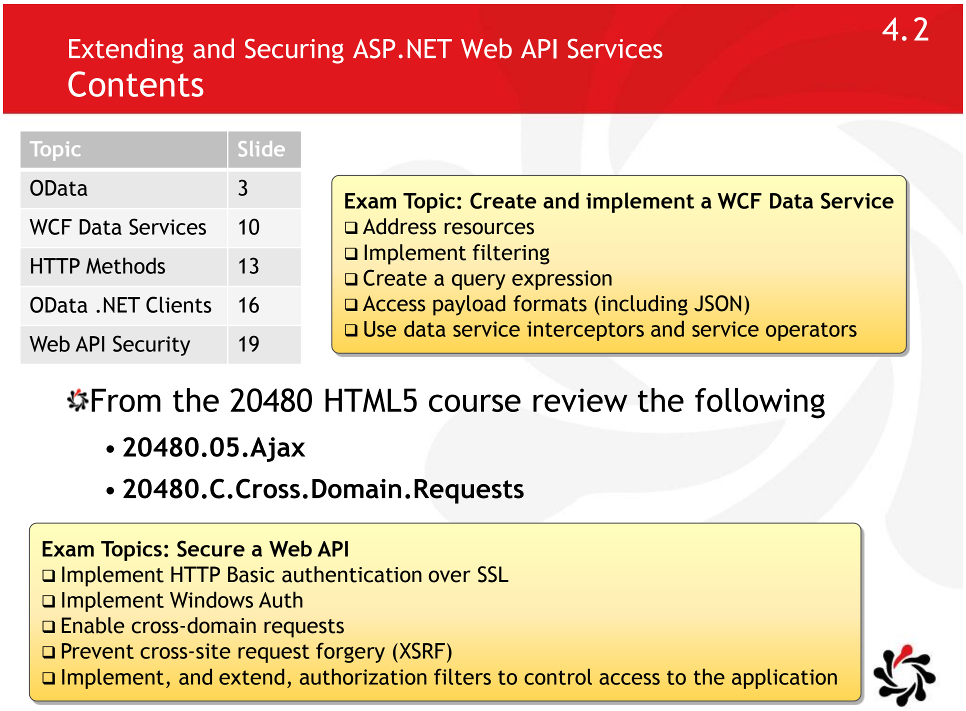


4.1

Module 4
Extending and Securing ASP.NET Web API Services

Developing Windows Azure and Web Services

Updated 11th April 2014



4.2

Extending and Securing ASP.NET Web API Services

Contents

Topic	Slide
OData	3
WCF Data Services	10
HTTP Methods	13
OData .NET Clients	16
Web API Security	19

Exam Topic: Create and implement a WCF Data Service
❑ Address resources
❑ Implement filtering
❑ Create a query expression
❑ Access payload formats (including JSON)
❑ Use data service interceptors and service operators

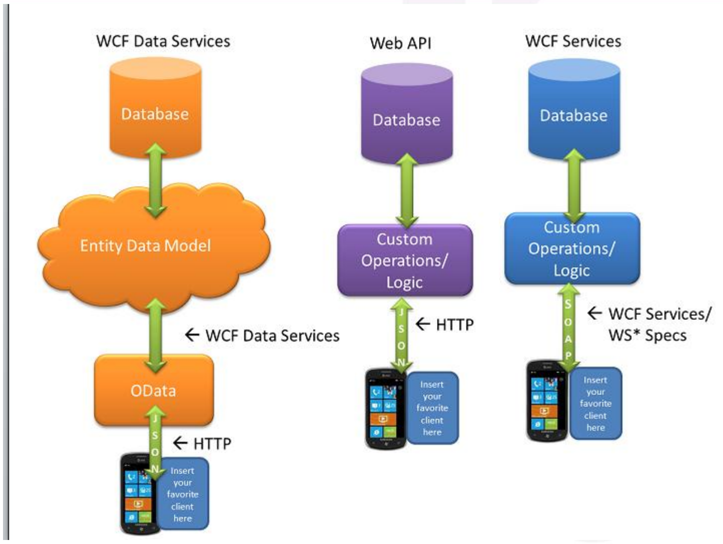
From the 20480 HTML5 course review the following

- **20480.05.Ajax**
- **20480.C.Cross.Domain.Requests**

Exam Topics: Secure a Web API
❑ Implement HTTP Basic authentication over SSL
❑ Implement Windows Auth
❑ Enable cross-domain requests
❑ Prevent cross-site request forgery (XSRF)
❑ Implement, and extend, authorization filters to control access to the application

4.3

OData
Comparing WCF, Web API, and WCF Data Services

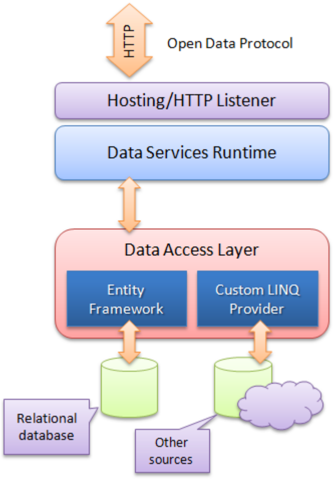


How I see Web API
`http://thedatafarm.com/blog/asp-net/how-i-see-web-api/`

4.4

OData
Overview

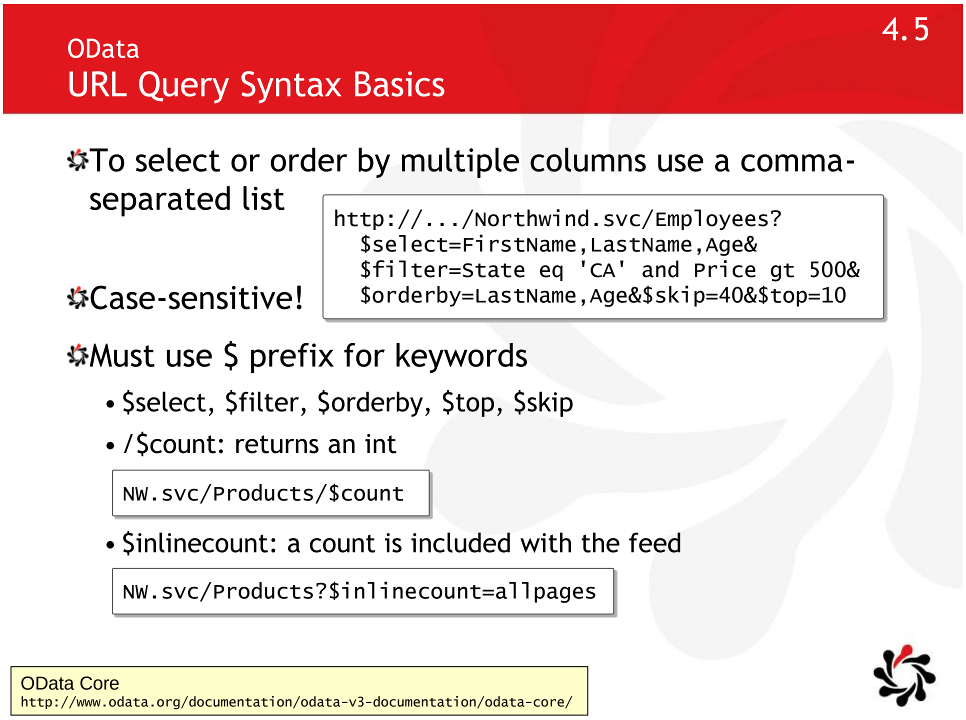
- OData is a standard for building HTTP services that follow standards for querying the data model
  - It defines a query syntax using URIs similar to SQL
- Two technologies for creating an OData service
  - WCF Data Services (.NET 3.5 +)
  - ASP.NET Web API OData (.NET 4.5 +)



WCF Data Services and OData At-a-Glance
`http://msdn.microsoft.com/en-us/data/aa937697`

WCF Data Services
`http://msdn.microsoft.com/en-us/data/odata.aspx`

WCF Data Services Blog
`http://blogs.msdn.com/b/astoriateam/`



4.5

OData

URL Query Syntax Basics

- To select or order by multiple columns use a comma-separated list
- Case-sensitive!

```
http://.../Northwind.svc/Employees?
  $select=FirstName,LastName,Age&
  $filter=State eq 'CA' and Price gt 500&
  $orderby=LastName,Age&$skip=40&$top=10
```

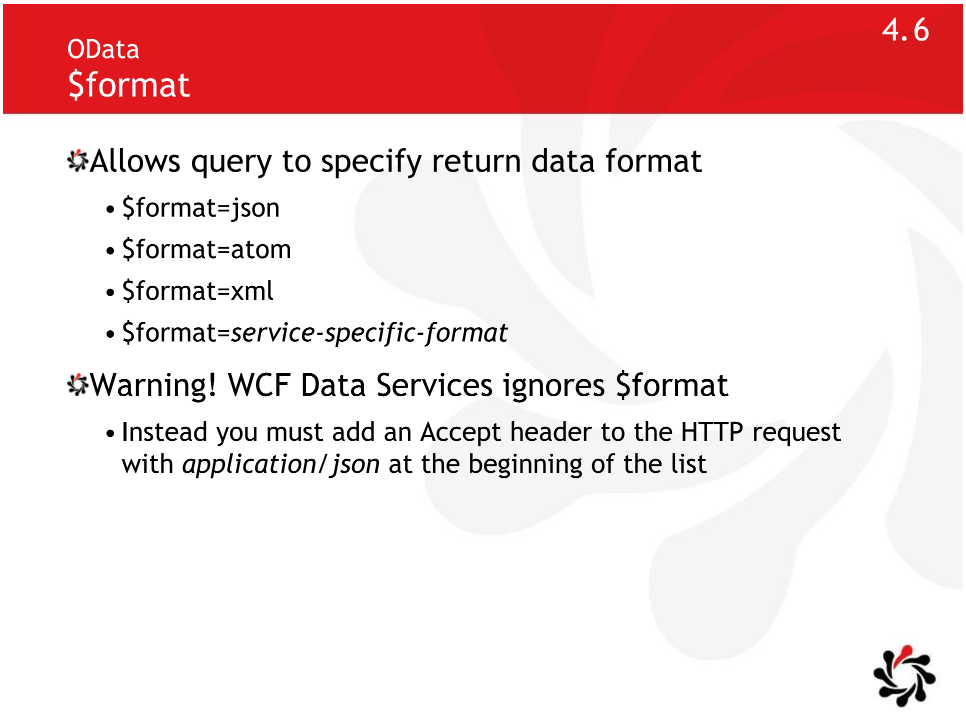
- Must use $ prefix for keywords
  - $select, $filter, $orderby, $top, $skip
  - /$count: returns an int

```
NW.svc/Products/$count
```

  - $inlinecount: a count is included with the feed

```
NW.svc/Products?$inlinecount=allpages
```

OData Core
http://www.odata.org/documentation/odata-v3-documentation/odata-core/



4.6

OData

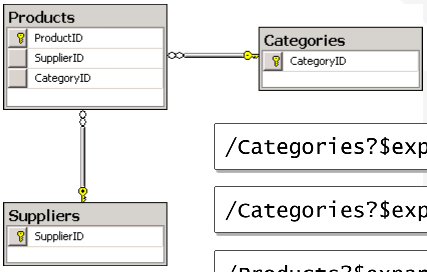
$format

- Allows query to specify return data format
  - $format=json
  - $format=atom
  - $format=xml
  - $format=*service-specific-format*
- Warning! WCF Data Services ignores $format
  - Instead you must add an Accept header to the HTTP request with *application/json* at the beginning of the list

4.7

OData
$expand

- The syntax of a $expand query option is a comma-separated list of Navigation Properties
  - Each Navigation Property can be followed by a forward slash and another Navigation Property to enable identifying a multi-level relationship



```
/Categories?$expand=Products
```

```
/Categories?$expand=Products/Suppliers
```

```
/Products?$expand=Category,Suppliers
```

Expand System Query Option ($expand)
http://www.odata.org/documentation/uri-conventions#ExpandSystemQueryOption

4.8

OData
$metadata

- Returns an EDMX document that contains a complete description of the feeds, types, properties, relationships exposed by the service in EDM
  - It isn't queryable so if you want to find Types that have an Address property you have to retrieve the whole EDMX and search the xml yourself

```
http://services.odata.org/Northwind/Northwind.svc/$metadata
```

Queryable OData Metadata
http://msopentech.com/odataorg/2010/04/queryable-odata-metadata/

4.9

OData
URI Query Syntax Examples

URI	Returns
/Customers('ALFKI')/ ContactName	An XML element that contains the ContactName property value for a specific Customer
/Customers('ALFKI')/ ContactName/$value	Only the string "Maria Anders" without the XML element
/Customers('ALFKI')/Orders	All the orders that are related to a specific Customer
/Orders(10643)/Customer	A reference to the Customer entity to which a specific Order entity belongs
/Orders?$filter=not endswith(ShipPostalCode,'100')	All the orders the postal codes of which do not end in 100
/Categories(1)/$links/Products	Links to the data instead of the actual data e.g. <uri>http://.../Products(4)</uri>
/Categories?$select=Name, Products&$expand=Products	Must select Products if expanding Products

Accessing Data Service Resources (WCF Data Services)
http://msdn.microsoft.com/en-us/library/dd728283.aspx

4.10

WCF Data Services
How to Create

- Project – Add New Item – WCF Data Service
- Create a context class that represents your data
  - ADO.NET Entity Data Model is easiest
  - Or any class that has properties of type IQueryable<T> where T is an “entity” (and optionally implements IUpdatable)
- Use context class in DataService<TContext>
- Set permissions for Entity Sets and Service Operations

```
public class BlogService : DataService<BlogDbContext>
{
  public static void InitializeService(DataServiceConfiguration config)
  {
    config.SetEntitySetAccessRule("Blogs", EntitySetRights.All);
    config.SetServiceOperationAccessRule("MyServiceOperation",
      ServiceOperationRights.All);
    config.DataServiceBehavior.MaxProtocolVersion =
      DataServiceProtocolVersion.V3;
```

4.11

WCF Data Services

json Support

- Differences in OData V2 and V3
  - In V2 you could ask for application/json in Accept
  - In V3 you must ask for application/json;odata=verbose
  - WCF Data Services 5.0 will return a 415 in response to a request for application/json if the service configuration allows a v3 response and if the client does not specify a MaxDataServiceVersion header or specifies a MaxDataServiceVersion header of 3.0
- Clients should always send a value for the MaxDataServiceVersion header to ensure forward compatibility

```
GET http://odata.example.org/Items?$top=5 HTTP/1.1
Accept: application/json
MaxDataServiceVersion: 2.0
```

What happened to application/json in WCF DS 5.0?
http://blogs.msdn.com/b/odatateam/archive/2012/04/11/what-happened-to-application-json-in-wcf-ds-5-0.aspx

4.12

WCF Data Services

Calling an OData Service from jQuery

- It’s possible to modify the headers with jQuery

```
$.ajax(url, {
    dataType: "json",
    beforeSend: function (xhr) {
        xhr.setRequestHeader("Accept", "application/json;odata=verbose");
        xhr.setRequestHeader("MaxDataServiceVersion", "3.0");
    },
    success: function (data) {
        // do something interesting with the data you get back.
    }
});
```

- How to enable $format support in WCF Data Services
  - You get the incoming request, remove the $format query option from the URI, and change the Accept header to application/json

Getting JSON Out of WCF Data Services
http://blogs.msdn.com/b/writingdata_services/archive/2011/02/25/getting-json-out-of-wcf-data-services.aspx

4.13

WCF Data Services

Intercepting Queries and Changes

- WCF Data Services enables an application to intercept request messages so that you can add custom logic
  - Define a query interceptor for the Orders entity set

```
[QueryInterceptor("Orders")]
public Expression<Func<Order, bool>> OnQueryOrders()
{
  return o => o.Customer.ContactName ==
    HttpContext.Current.User.Identity.Name;
}
```

  - Check UpdateOperations parameter to determine type

```
[ChangeInterceptor("Products")]
public void OnChangeProducts(
    Product product, UpdateOperations operation) {
  if(operation == UpdateOperations.Delete) {
```

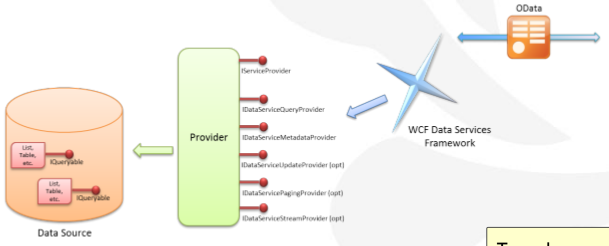
Interceptors (WCF Data Services)
http://msdn.microsoft.com/en-us/library/dd744842.aspx

4.14

WCF Data Services

Custom Data Service Providers

- A Data Service Provider is simply a .NET class that sits between the Data Services Framework and the underlying data source that's being exposed


OData
IServiceProvider
IDataServiceQueryProvider
IDataServiceMetadataProvider
IDataServiceUpdateProvider (opt)
IDataServicePagingProvider (opt)
IDataServiceStreamProvider (opt)
Provider
WCF Data Services Framework
List, Table, etc.
IQueryable
List, Table, etc.
IQueryable
Data Source

To make your service updatable

```
public class MyDataSource : IUpdatable {
  public IQueryable<Product> Products { get { ...
```

To make your service queryable

Custom Data Service Providers
http://msdn.microsoft.com/en-us/data/gg191846.aspx

4.15

HTTP Methods
MERGE and PATCH

- To update a column of a record without overwriting other columns, use MERGE or PATCH verbs and only pass the changed column values

```
PATCH /AW.svc/Contacts(23)
Host: AdventureWorks.com
Content-Type: application-json
{ State: 'CA' }
```

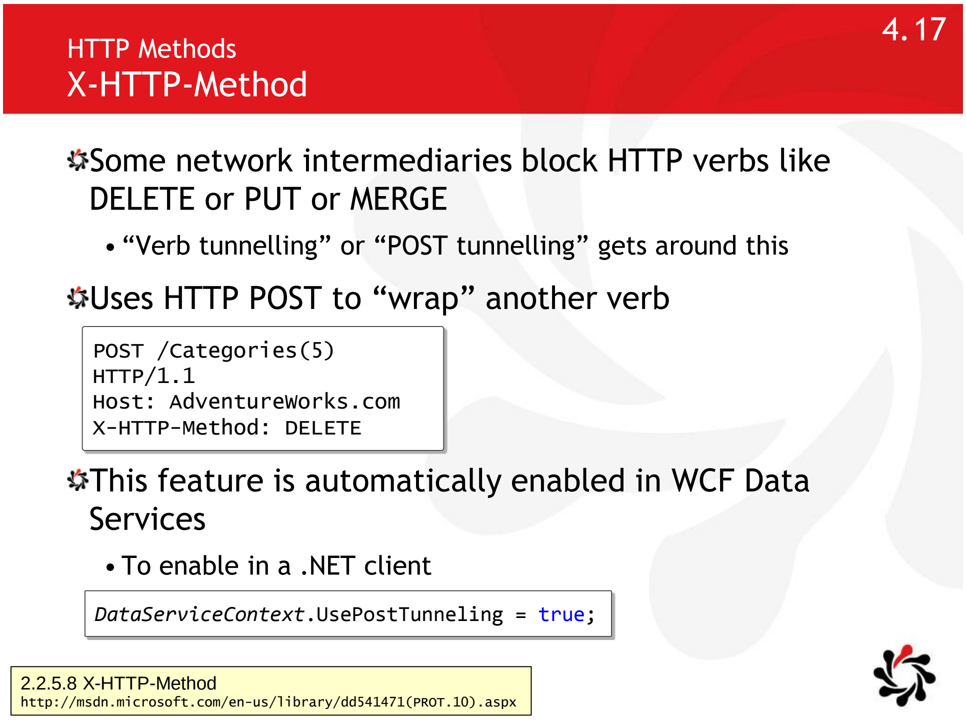
- Warning!
  - By default the WCF Data Services client library passes all properties using a MERGE, not just the ones that have changed
  - Use SaveChangesOptions.ReplaceOnUpdate to change to using a PUT (which requires all properties to be sent)

WCF Data Services: Optimizing bandwidth usage and performance with updates
http://blogs.infosupport.com/wcf-data-services-optimizing-updates-in-the-client-library/

4.16

HTTP Methods
Support for CRUD Operations

- To enable CRUD operations, IIS must allow the following methods on the .svc extension
  - PUT
  - DELETE



HTTP Methods
X-HTTP-Method

4.17

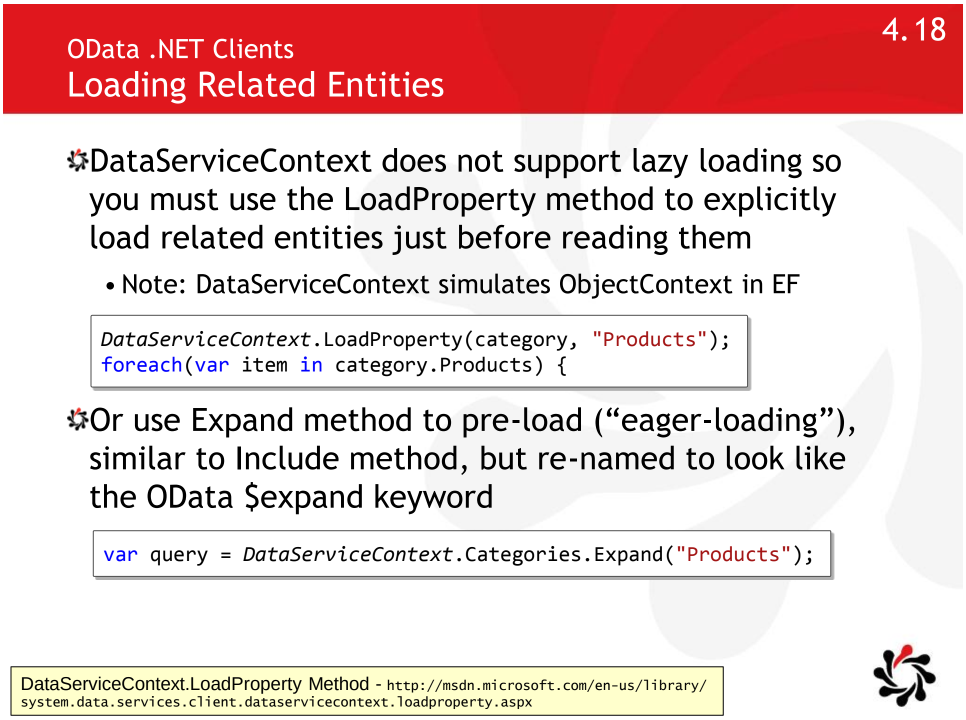
- Some network intermediaries block HTTP verbs like DELETE or PUT or MERGE
  - “Verb tunnelling” or “POST tunnelling” gets around this
- Uses HTTP POST to “wrap” another verb

```
POST /Categories(5)
HTTP/1.1
Host: AdventureWorks.com
X-HTTP-Method: DELETE
```

- This feature is automatically enabled in WCF Data Services
  - To enable in a .NET client

```
DataServiceContext.UsePostTunneling = true;
```

2.2.5.8 X-HTTP-Method
http://msdn.microsoft.com/en-us/library/dd541471(PROT.10).aspx



OData .NET Clients
Loading Related Entities

4.18

- DataServiceContext does not support lazy loading so you must use the LoadProperty method to explicitly load related entities just before reading them
  - Note: DataServiceContext simulates ObjectContext in EF

```
DataServiceContext.LoadProperty(category, "Products");
foreach(var item in category.Products) {
```

- Or use Expand method to pre-load (“eager-loading”), similar to Include method, but re-named to look like the OData $expand keyword

```
var query = DataServiceContext.Categories.Expand("Products");
```

DataServiceContext.LoadProperty Method - http://msdn.microsoft.com/en-us/library/system.data.services.client.dataservicecontext.loadproperty.aspx

OData .NET Clients

Troubleshooting

- To find out how a LINQ to OData query will translate into an OData URL use RequestUri

```
var query = from p in DataServiceContext.Products
            where p.Color == "Red"
            select p;
string uri = ((DataServiceQuery)query).RequestUri.ToString();
```

```
http://localhost:1034/AW.svc/Products()?$filter=Color eq 'Red'
```

OData .NET Clients

Set Headers in the Client Request

- Create an event handler for SendRequest2

```
DataServiceContext.SendingRequest2 += db_SendingRequest2;
```

- Add the header in the event handler

```
private void db_SendingRequest2(object sender, SendingRequest2EventArgs e)
{
  // Add an Authorization header that contains an
  // OAuth WRAP access token to the request
  e.RequestMessage.SetHeader(
    "Authorization", "WRAP access_token=\"123456789\"");
}
```

How to: Set Headers in the Client Request (WCF Data Services)
http://msdn.microsoft.com/en-us/library/gg258441.aspx

4.21

Web API Security
Overview

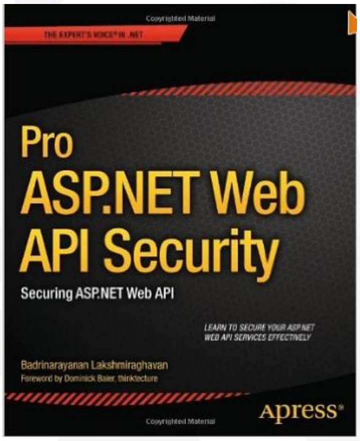
- For the most common scenario—JavaScript in a Web page accessing a Web API service on the same site—discussing security for ASP.NET Web API is redundant
  - Provided that you authenticate your users and authorize access to the Web Forms/Views holding the JavaScript that consumes your services, you've probably provided all the security your services need
- There's one exception (and it's an important one):
  - ASP.NET doesn't automatically protect you against Cross-Site Request Forgery (CSRF/XSRF) attacks

Enabling and Customizing ASP.NET Web API Services Security
http://msdn.microsoft.com/en-us/magazine/dn201748.aspx

4.22

Web API Security
Further Study

- Using this book, you can gain a solid understanding of the security techniques relevant to ASP.NET Web API
  - All the underlying concepts are introduced from basic principles and developed to the point where you can use them confidently, knowing what you are doing


Copyrighted Material
THE EXPERT'S VOICE® IN .NET
Pro
ASP.NET Web
API Security
Securing ASP.NET Web API
LEARN TO SECURE YOUR ASP.NET WEB API SERVICES EFFECTIVELY
Badrinarayanan Lakshmiraghavan
Foreword by Dominick Baier, thinktecture
Apress®
Copyrighted Material

Pro ASP.NET Web API Security: Securing ASP.NET Web API
http://www.amazon.co.uk/Pro-ASP-NET-Web-API-Security/dp/1430257822/

Module 5
Creating WCF Services

Developing Windows Azure and Web Services

Updated 11th April 2014

Creating WCF Services

Contents

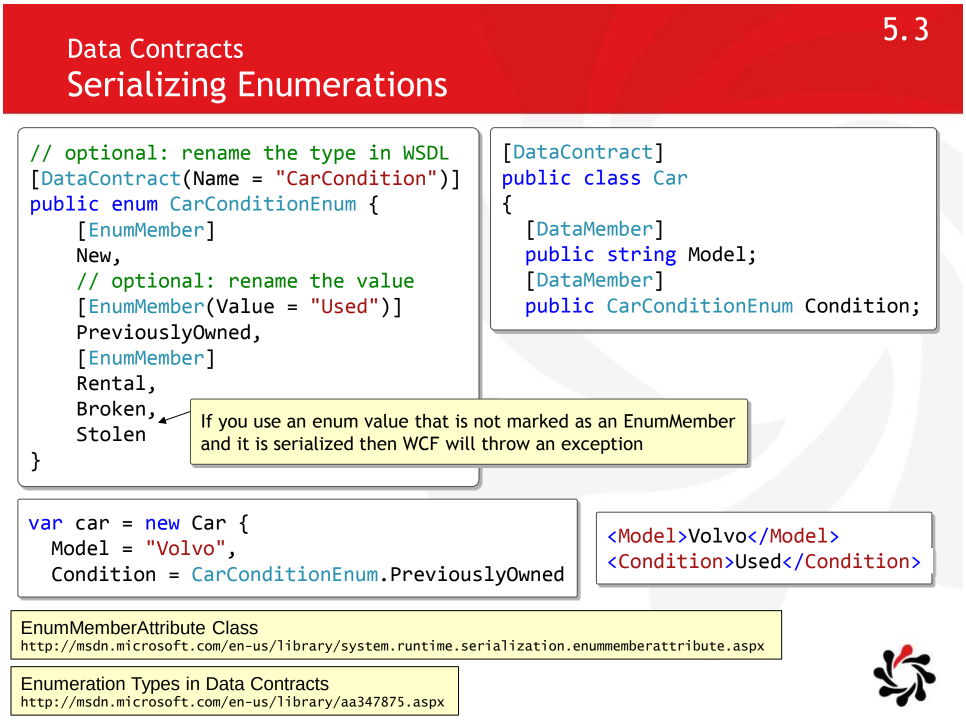
Exam Topics: Create a WCF service
- ❑ Create contracts (service, data, message, callback, and fault)

Exam Topics: Version a WCF service
- ❑ Version different types of contracts (message, service, data)
- ❑ Configure address, binding, and routing service versioning

Exam Topics: Consume WCF services
- ❑ Generate proxies by using SvcUtil
- ❑ Generate proxies by creating a service reference
- ❑ Create and implement channel factories

Exam Topics: Host and manage services
- ❑ Manage services concurrency (single, multiple, reentrant)
- ❑ Choose an instancing mode (per call, per session, singleton)

Exam Topics: Configure WCF services by using configuration settings or the API
- ❑ Configure service behaviors
- ❑ Configure service endpoints
- ❑ Configure binding
- ❑ Specify a service contract
- ❑ Expose service metadata (XSDs, WSDL, metadata exchange)



5.3

Data Contracts
Serializing Enumerations

```
// optional: rename the type in WSDL
[DataContract(Name = "CarCondition")]
public enum CarConditionEnum {
    [EnumMember]
    New,
    // optional: rename the value
    [EnumMember(Value = "Used")]
    PreviouslyOwned,
    [EnumMember]
    Rental,
    Broken,
    Stolen
}
```

If you use an enum value that is not marked as an EnumMember and it is serialized then WCF will throw an exception

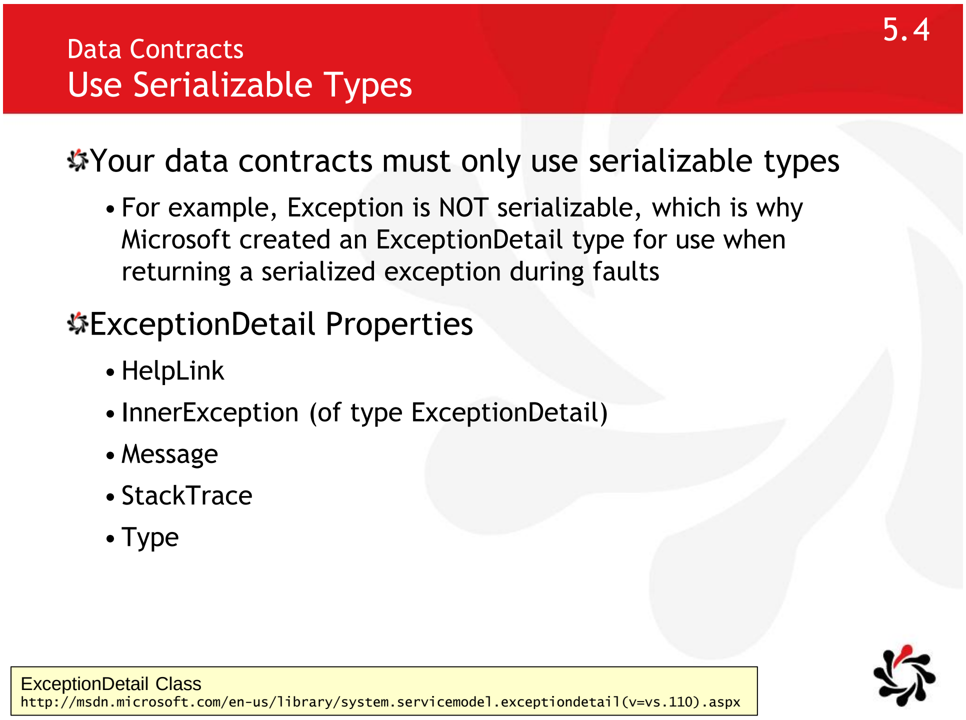
```
[DataContract]
public class Car
{
  [DataMember]
  public string Model;
  [DataMember]
  public CarConditionEnum Condition;
```

```
var car = new Car {
  Model = "Volvo",
  Condition = CarConditionEnum.PreviouslyOwned
```

```
<Model>Volvo</Model>
<Condition>Used</Condition>
```

EnumMemberAttribute Class
http://msdn.microsoft.com/en-us/library/system.runtime.serialization.enummemberattribute.aspx

Enumeration Types in Data Contracts
http://msdn.microsoft.com/en-us/library/aa347875.aspx

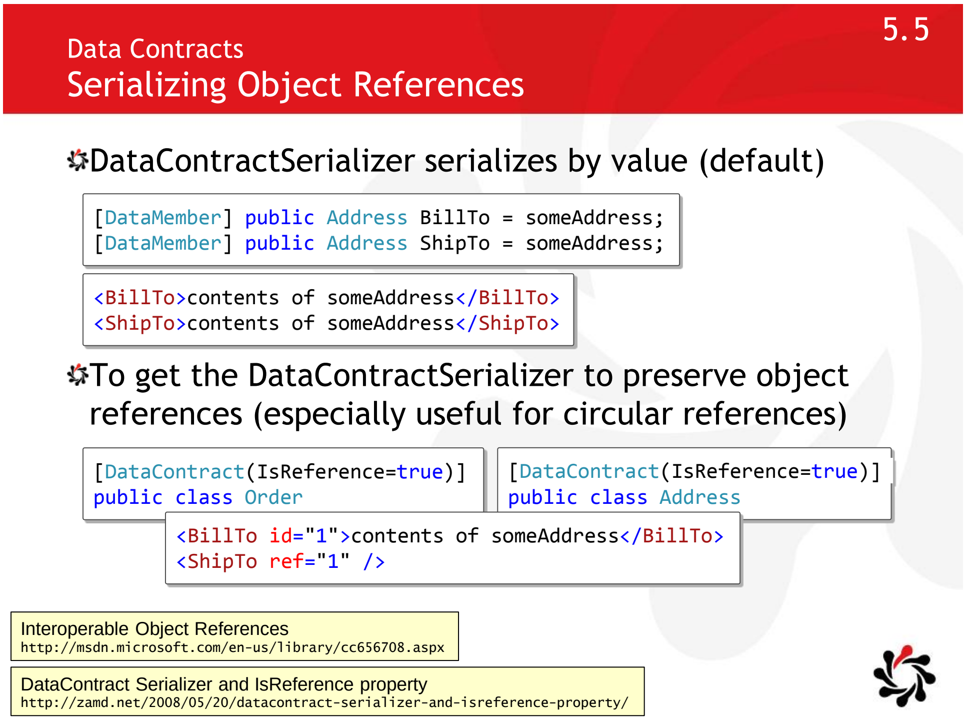


5.4

Data Contracts
Use Serializable Types

- Your data contracts must only use serializable types
  - For example, Exception is NOT serializable, which is why Microsoft created an ExceptionDetail type for use when returning a serialized exception during faults
- ExceptionDetail Properties
  - HelpLink
  - InnerException (of type ExceptionDetail)
  - Message
  - StackTrace
  - Type

ExceptionDetail Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.exceptiondetail(v=vs.110).aspx



5.5

Data Contracts

Serializing Object References

- DataContractSerializer serializes by value (default)

```
[DataMember] public Address BillTo = someAddress;
[DataMember] public Address ShipTo = someAddress;
```

```
<BillTo>contents of someAddress</BillTo>
<ShipTo>contents of someAddress</ShipTo>
```

- To get the DataContractSerializer to preserve object references (especially useful for circular references)

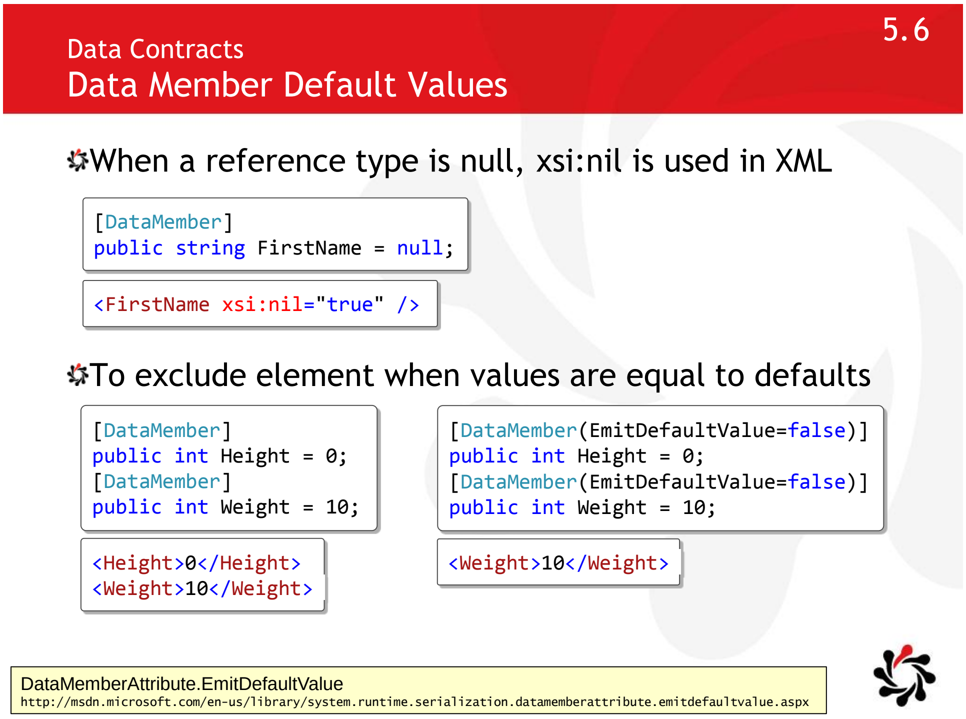
```
[DataContract(IsReference=true)]
public class Order
```

```
[DataContract(IsReference=true)]
public class Address
```

```
<BillTo id="1">contents of someAddress</BillTo>
<ShipTo ref="1" />
```

Interoperable Object References
http://msdn.microsoft.com/en-us/library/cc656708.aspx

DataContract Serializer and IsReference property
http://zamd.net/2008/05/20/datacontract-serializer-and-isreference-property/



5.6

Data Contracts

Data Member Default Values

- When a reference type is null, xsi:nil is used in XML

```
[DataMember]
public string FirstName = null;
```

```
<FirstName xsi:nil="true" />
```

- To exclude element when values are equal to defaults

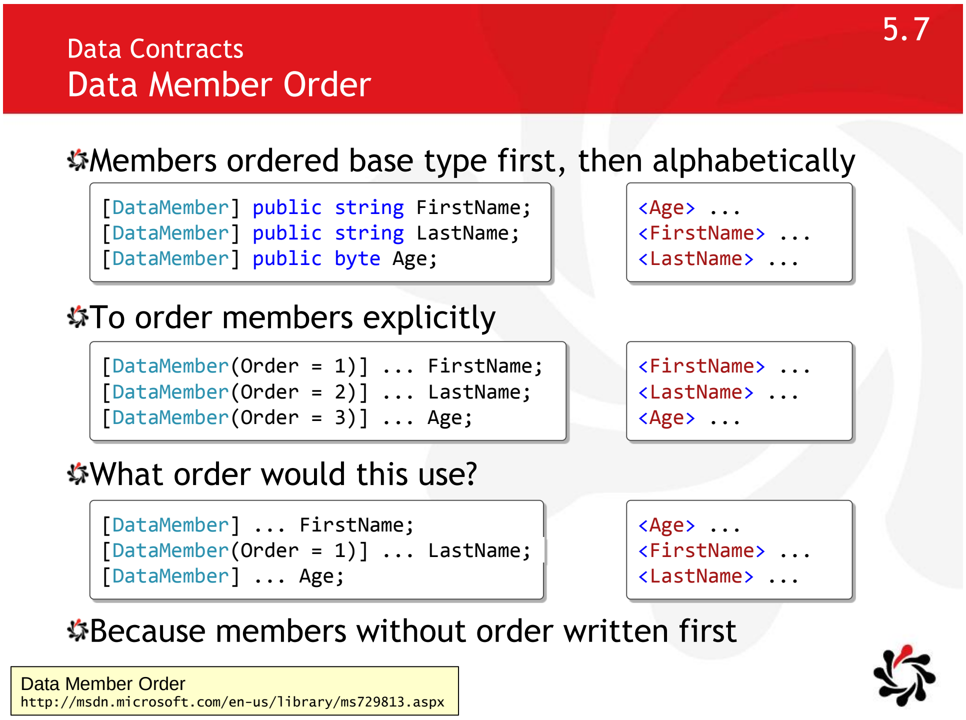
```
[DataMember]
public int Height = 0;
[DataMember]
public int Weight = 10;
```

```
[DataMember(EmitDefaultValue=false)]
public int Height = 0;
[DataMember(EmitDefaultValue=false)]
public int Weight = 10;
```

```
<Height>0</Height>
<Weight>10</Weight>
```

```
<Weight>10</Weight>
```

DataMemberAttribute.EmitDefaultValue
http://msdn.microsoft.com/en-us/library/system.runtime.serialization.datamemberattribute.emitdefaultvalue.aspx



5.7

Data Contracts

Data Member Order

- Members ordered base type first, then alphabetically

```
[DataMember] public string FirstName;
[DataMember] public string LastName;
[DataMember] public byte Age;
```

```
<Age> ...
<FirstName> ...
<LastName> ...
```

- To order members explicitly

```
[DataMember(Order = 1)] ... FirstName;
[DataMember(Order = 2)] ... LastName;
[DataMember(Order = 3)] ... Age;
```

```
<FirstName> ...
<LastName> ...
<Age> ...
```

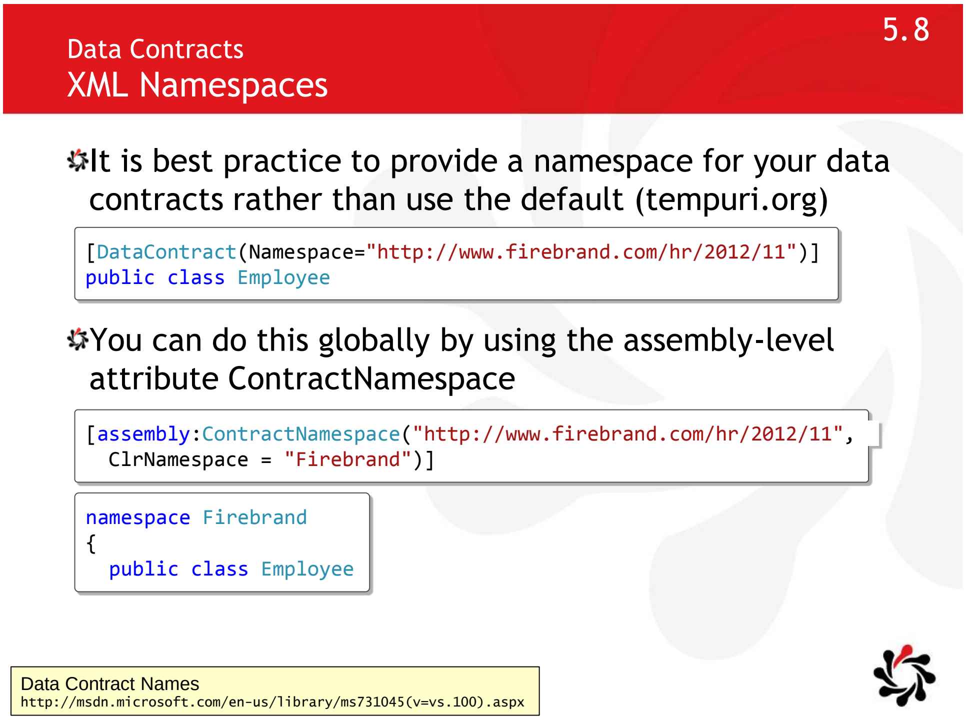
- What order would this use?

```
[DataMember] ... FirstName;
[DataMember(Order = 1)] ... LastName;
[DataMember] ... Age;
```

```
<Age> ...
<FirstName> ...
<LastName> ...
```

- Because members without order written first

Data Member Order
http://msdn.microsoft.com/en-us/library/ms729813.aspx



5.8

Data Contracts

XML Namespaces

- It is best practice to provide a namespace for your data contracts rather than use the default (tempuri.org)

```
[DataContract(Namespace="http://www.firebrand.com/hr/2012/11")]
public class Employee
```

- You can do this globally by using the assembly-level attribute ContractNamespace

```
[assembly:ContractNamespace("http://www.firebrand.com/hr/2012/11",
  ClrNamespace = "Firebrand")]
```

```
namespace Firebrand
{
  public class Employee
```

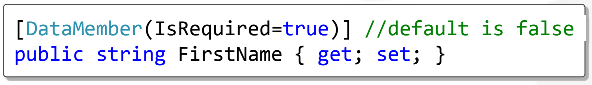
Data Contract Names
http://msdn.microsoft.com/en-us/library/ms731045(v=vs.100).aspx

5.9

Data Contracts

Versioning

- "Strict schema validity"
  - This means in both directions (new-to-old and old-to-new) so data and service contracts must be considered immutable
  - If a new version is required then a new data contract must be created with a different name or namespace and the service contract should add a new operation
- Automatic versioning is the default
  - Missing data members don't cause exceptions so to throw a SerializationException if a data member is missing when deserializing set IsRequired to true



```
[DataMember(IsRequired=true)] //default is false
public string FirstName { get; set; }
```

DataMemberAttribute.IsRequired Property
http://msdn.microsoft.com/en-us/library/system.runtime.serialization.datamemberattribute.isrequired.aspx

5.10

Data Contracts

Round-Tripping (1/2)

- If strict schema validity is NOT required, and you need to be able to round-trip an instance of a data contract with older clients, use **IExtensibleDataObject**
  - **IExtensibleDataObject** provides a data structure to store extra data encountered during deserialization
  - In a roundtrip operation where data is received, processed, and sent back, any extra data is returned to the sender intact
  - If you do not implement the interface, any extra data is ignored and lost during a roundtrip operation
  - Useful to store data received from future versions of the contract
  - Note: svcutil and VS-generated code implement this but you need to implement it yourself if reusing data contracts

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

5.11

Data Contracts
Round-Tripping (2/2)

- Must mark versions e.g. Person and PersonV2 with same Name and Namespace
  - Best practice to apply an order to new members

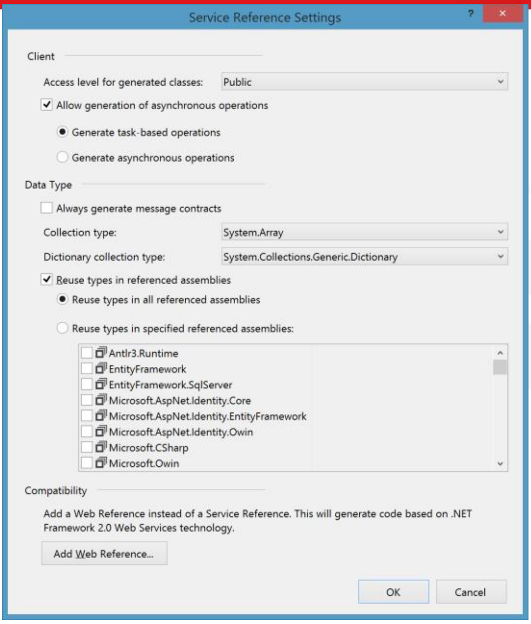
```
[DataContract(Name="Person", Namespace="http://.../")]
public class Person : IExtensibleDataObject
{
  public ExtensionDataObject ExtensionData { get; set; }
```

```
[DataContract(Name="Person", Namespace="http://.../")]
public class PersonV2 : IExtensibleDataObject
{
  public ExtensionDataObject ExtensionData { get; set; }
  [DataMember(Order=2)]
  public int Age { get; set; } // new member
```

Best Practices for Handling Change in Your WCF Applications
http://www.devx.com/dotnet/Article/42005/1954

5.12

Data Contracts
Returning Collections from Operations



Service Reference Settings

Client

Access level for generated classes: Public

☑ Allow generation of asynchronous operations

● Generate task-based operations

○ Generate asynchronous operations

Data Type

□ Always generate message contracts

Collection type: System.Array

Dictionary collection type: System.Collections.Generic.Dictionary

☑ Reuse types in referenced assemblies

● Reuse types in all referenced assemblies

○ Reuse types in specified referenced assemblies:

□ Antlr3.Runtime
□ EntityFramework
□ EntityFramework.SqlServer
□ Microsoft.AspNet.Identity.Core
□ Microsoft.AspNet.Identity.EntityFramework
□ Microsoft.AspNet.Identity.Owin
□ Microsoft.CSharp
□ Microsoft.Owin

Compatibility

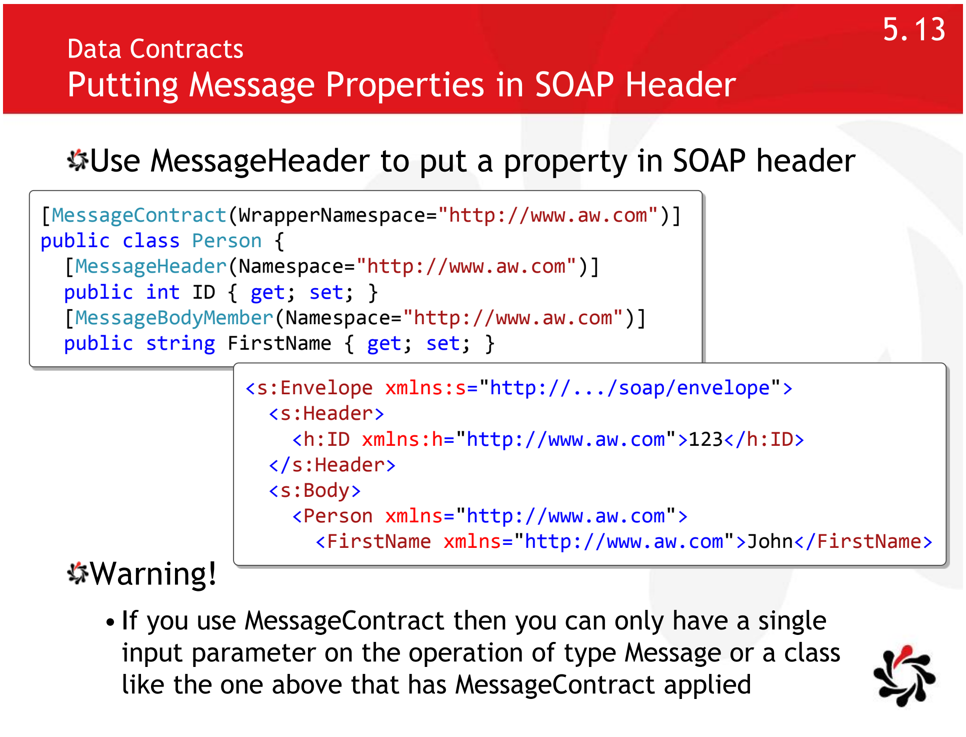
Add a Web Reference instead of a Service Reference. This will generate code based on .NET Framework 2.0 Web Services technology.

- Add Service Reference...

  Reuse only applies to data contracts, not service & operation contracts

- ChannelFactories

  The client should have a reference to the contracts so will know exactly what collection type to use



5.13

Data Contracts

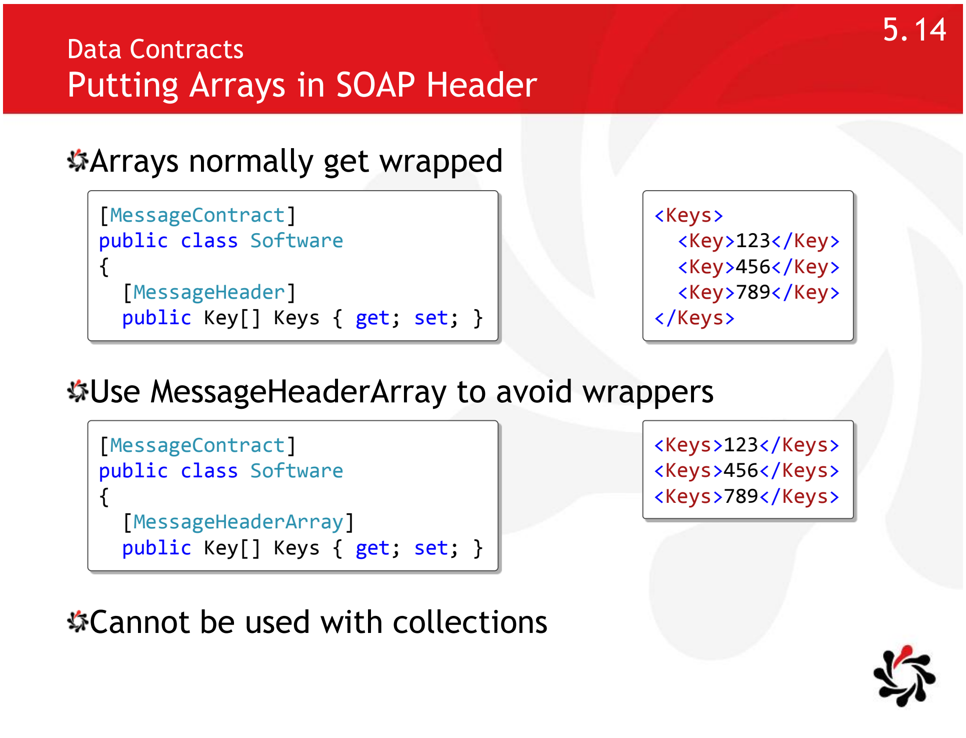
Putting Message Properties in SOAP Header

- Use MessageHeader to put a property in SOAP header

```
[MessageContract(WrapperNamespace="http://www.aw.com")]
public class Person {
  [MessageHeader(Namespace="http://www.aw.com")]
  public int ID { get; set; }
  [MessageBodyMember(Namespace="http://www.aw.com")]
  public string FirstName { get; set; }
```

```
<s:Envelope xmlns:s="http://.../soap/envelope">
  <s:Header>
    <h:ID xmlns:h="http://www.aw.com">123</h:ID>
  </s:Header>
  <s:Body>
    <Person xmlns="http://www.aw.com">
      <FirstName xmlns="http://www.aw.com">John</FirstName>
```

- Warning!
  - If you use MessageContract then you can only have a single input parameter on the operation of type Message or a class like the one above that has MessageContract applied



5.14

Data Contracts

Putting Arrays in SOAP Header

- Arrays normally get wrapped

```
[MessageContract]
public class Software
{
  [MessageHeader]
  public Key[] Keys { get; set; }
```

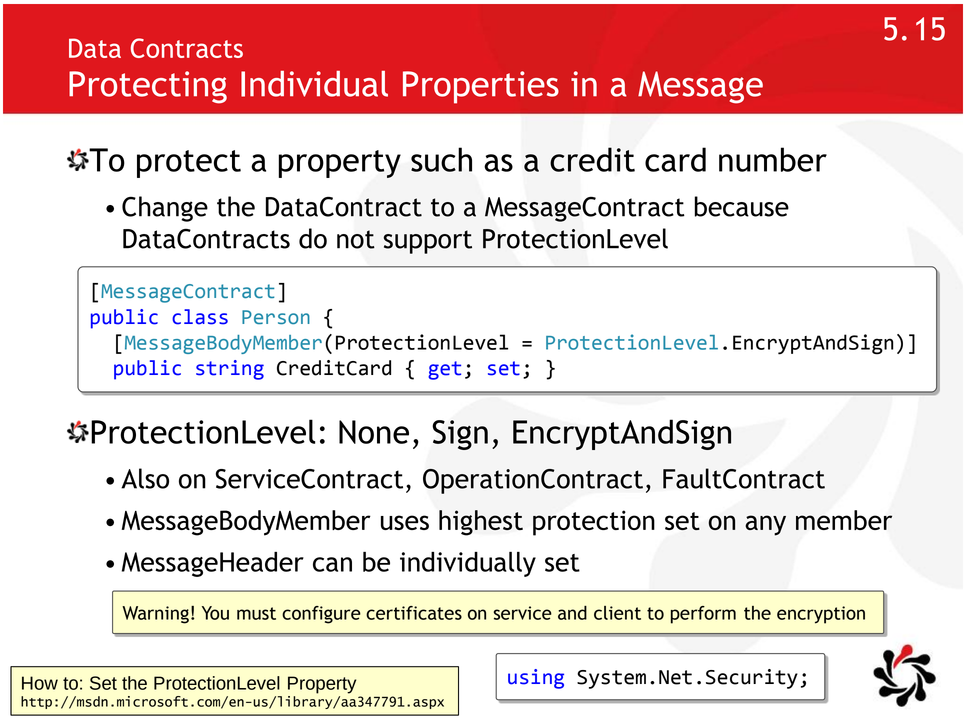
```
<Keys>
  <Key>123</Key>
  <Key>456</Key>
  <Key>789</Key>
</Keys>
```

- Use MessageHeaderArray to avoid wrappers

```
[MessageContract]
public class Software
{
  [MessageHeaderArray]
  public Key[] Keys { get; set; }
```

```
<Keys>123</Keys>
<Keys>456</Keys>
<Keys>789</Keys>
```

- Cannot be used with collections



5.15

Data Contracts
Protecting Individual Properties in a Message

- To protect a property such as a credit card number
  - Change the DataContract to a MessageContract because DataContracts do not support ProtectionLevel

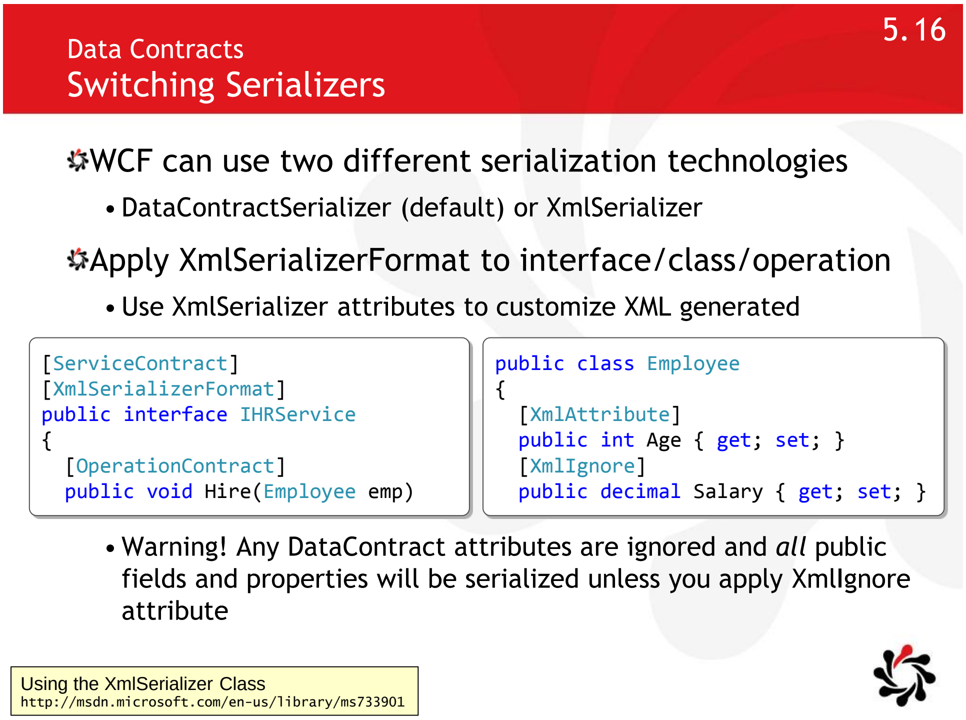
```
[MessageContract]
public class Person {
  [MessageBodyMember(ProtectionLevel = ProtectionLevel.EncryptAndSign)]
  public string CreditCard { get; set; }
```

- ProtectionLevel: None, Sign, EncryptAndSign
  - Also on ServiceContract, OperationContract, FaultContract
  - MessageBodyMember uses highest protection set on any member
  - MessageHeader can be individually set

Warning! You must configure certificates on service and client to perform the encryption

How to: Set the ProtectionLevel Property
http://msdn.microsoft.com/en-us/library/aa347791.aspx

```
using System.Net.Security;
```



5.16

Data Contracts
Switching Serializers

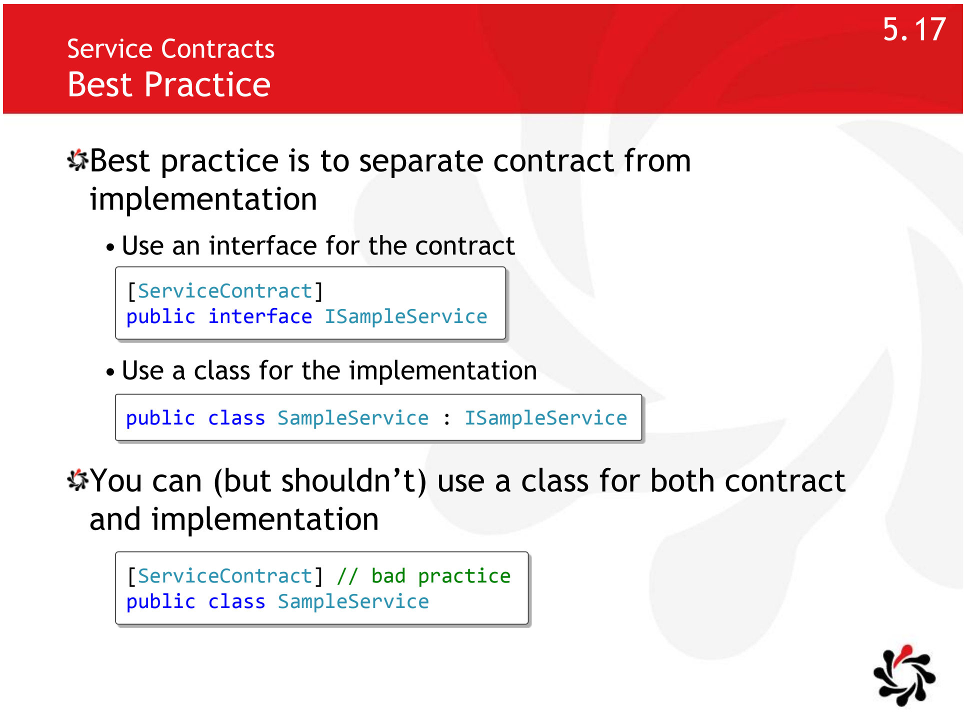
- WCF can use two different serialization technologies
  - DataContractSerializer (default) or XmlSerializer
- Apply XmlSerializerFormat to interface/class/operation
  - Use XmlSerializer attributes to customize XML generated

```
[ServiceContract]
[XmlSerializerFormat]
public interface IHRService
{
  [OperationContract]
  public void Hire(Employee emp)
```

```
public class Employee
{
  [XmlAttribute]
  public int Age { get; set; }
  [XmlIgnore]
  public decimal Salary { get; set; }
```

  - Warning! Any DataContract attributes are ignored and *all* public fields and properties will be serialized unless you apply XmlIgnore attribute

Using the XmlSerializer Class
http://msdn.microsoft.com/en-us/library/ms733901



5.17

Service Contracts

Best Practice

- Best practice is to separate contract from implementation
  - Use an interface for the contract

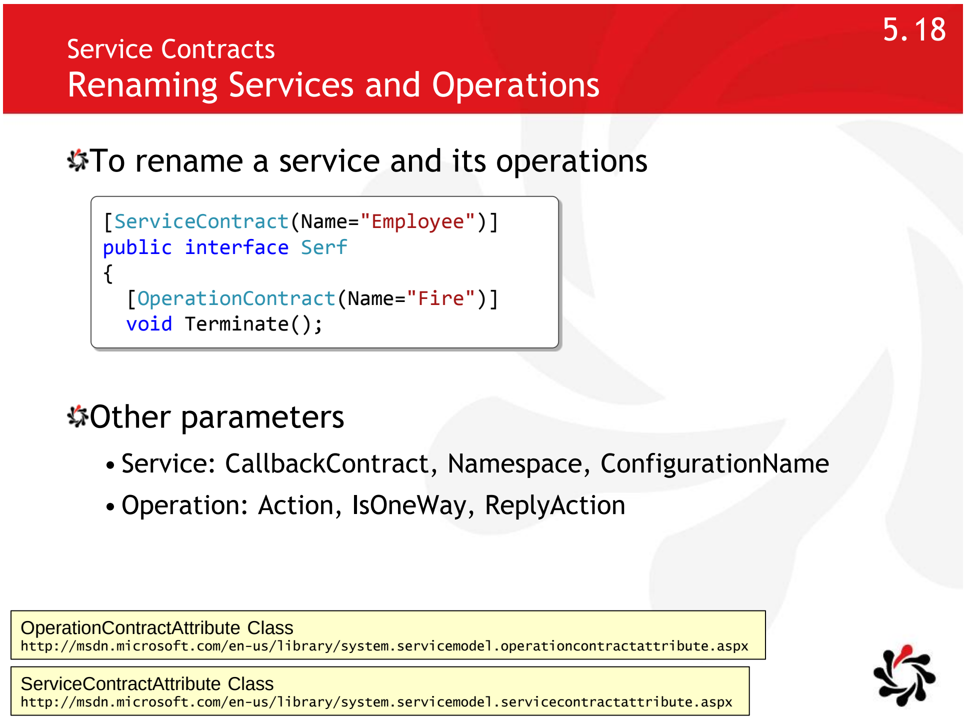
```
[ServiceContract]
public interface ISampleService
```

  - Use a class for the implementation

```
public class SampleService : ISampleService
```

- You can (but shouldn’t) use a class for both contract and implementation

```
[ServiceContract] // bad practice
public class SampleService
```



5.18

Service Contracts

Renaming Services and Operations

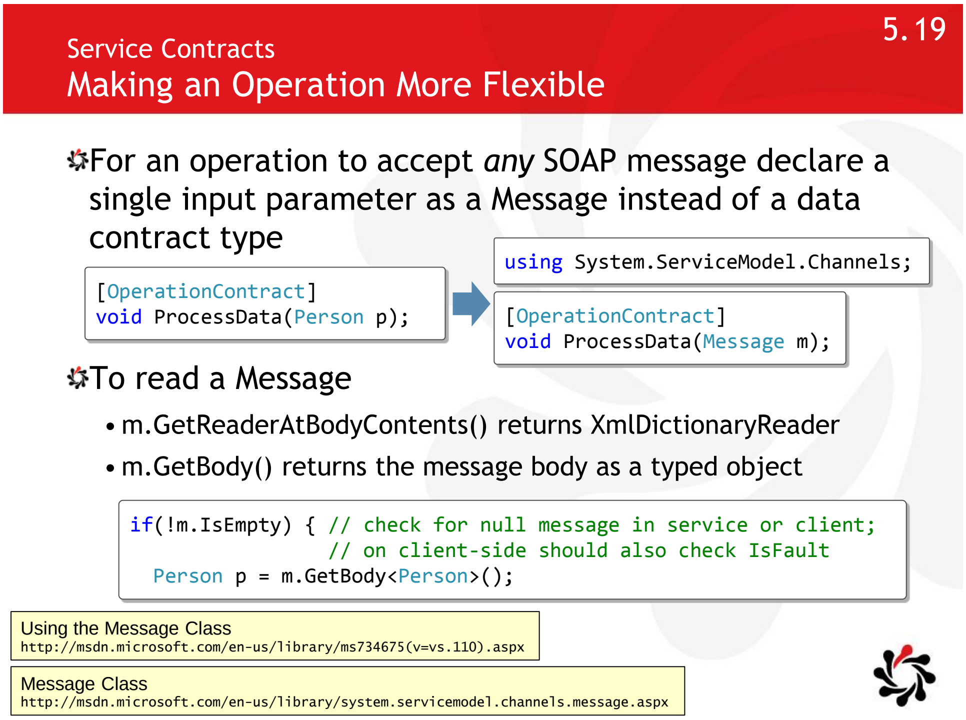
- To rename a service and its operations

```
[ServiceContract(Name="Employee")]
public interface Serf
{
  [OperationContract(Name="Fire")]
  void Terminate();
```

- Other parameters
  - Service: CallbackContract, Namespace, ConfigurationName
  - Operation: Action, IsOneWay, ReplyAction

OperationContractAttribute Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.operationcontractattribute.aspx

ServiceContractAttribute Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.servicecontractattribute.aspx



Service Contracts
Making an Operation More Flexible

- For an operation to accept *any* SOAP message declare a single input parameter as a Message instead of a data contract type

```
[OperationContract]
void ProcessData(Person p);
```

```
using System.ServiceModel.Channels;
```

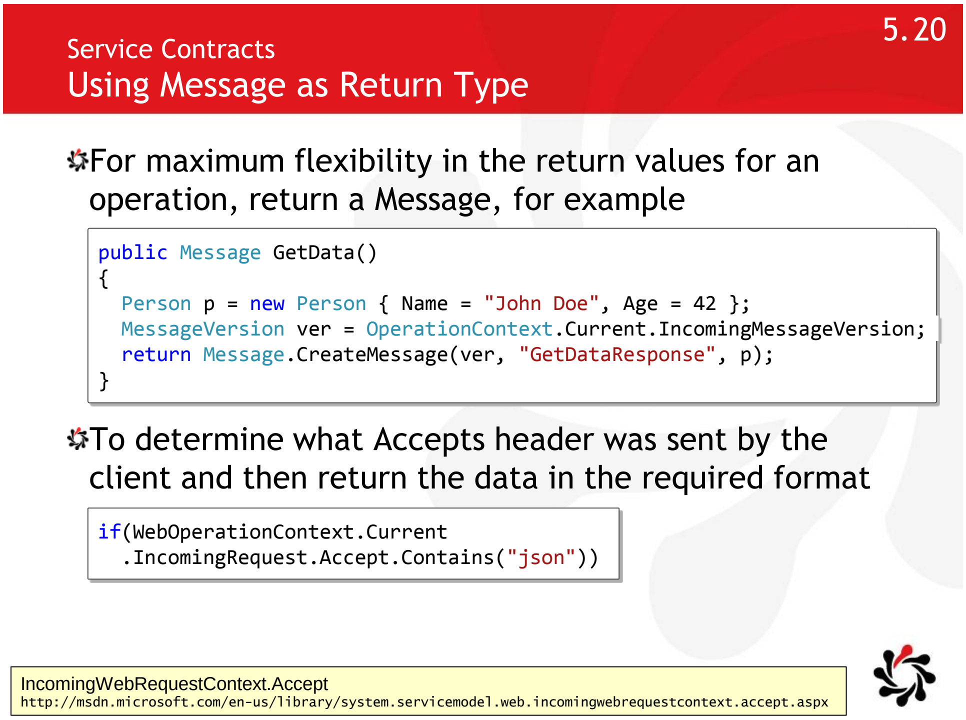
```
[OperationContract]
void ProcessData(Message m);
```

- To read a Message
  - m.GetReaderAtBodyContents() returns XmlDictionaryReader
  - m.GetBody() returns the message body as a typed object

```
if(!m.IsEmpty) { // check for null message in service or client;
                 // on client-side should also check IsFault
  Person p = m.GetBody<Person>();
```

Using the Message Class
http://msdn.microsoft.com/en-us/library/ms734675(v=vs.110).aspx

Message Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.message.aspx



Service Contracts
Using Message as Return Type

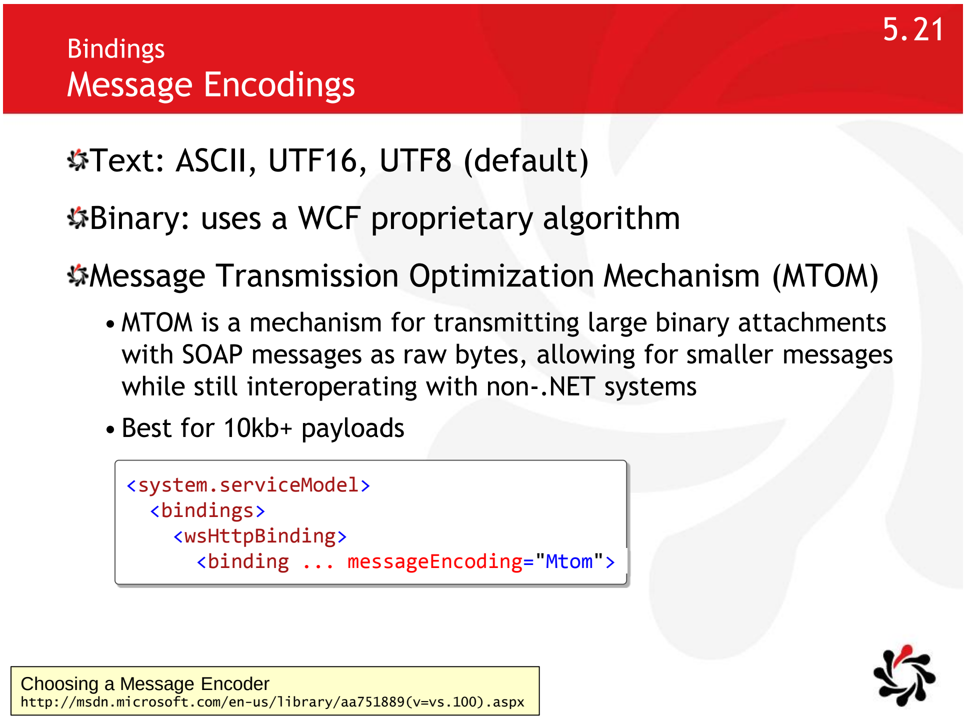
- For maximum flexibility in the return values for an operation, return a Message, for example

```
public Message GetData()
{
  Person p = new Person { Name = "John Doe", Age = 42 };
  MessageVersion ver = OperationContext.Current.IncomingMessageVersion;
  return Message.CreateMessage(ver, "GetDataResponse", p);
}
```

- To determine what Accepts header was sent by the client and then return the data in the required format

```
if(WebOperationContext.Current
  .IncomingRequest.Accept.Contains("json"))
```

IncomingWebRequestContext.Accept
http://msdn.microsoft.com/en-us/library/system.servicemodel.web.incomingwebrequestcontext.accept.aspx



Bindings

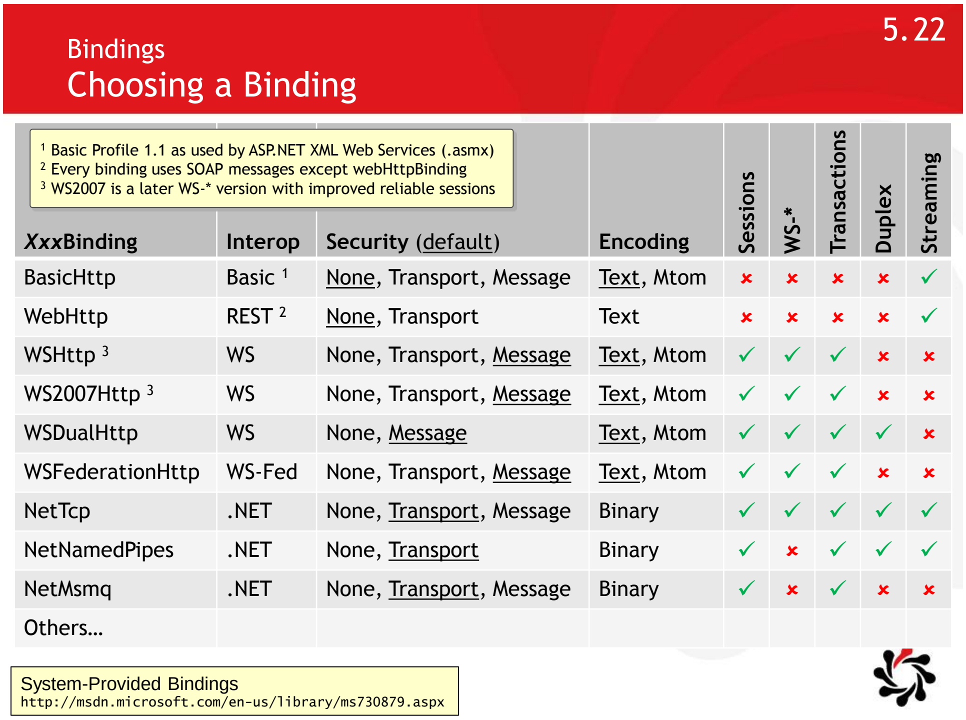
Message Encodings

5.21

- Text: ASCII, UTF16, UTF8 (default)
- Binary: uses a WCF proprietary algorithm
- Message Transmission Optimization Mechanism (MTOM)
  - MTOM is a mechanism for transmitting large binary attachments with SOAP messages as raw bytes, allowing for smaller messages while still interoperating with non-.NET systems
  - Best for 10kb+ payloads

```
<system.serviceModel>
  <bindings>
    <wsHttpBinding>
      <binding ... messageEncoding="Mtom">
```

Choosing a Message Encoder
http://msdn.microsoft.com/en-us/library/aa751889(v=vs.100).aspx



Bindings

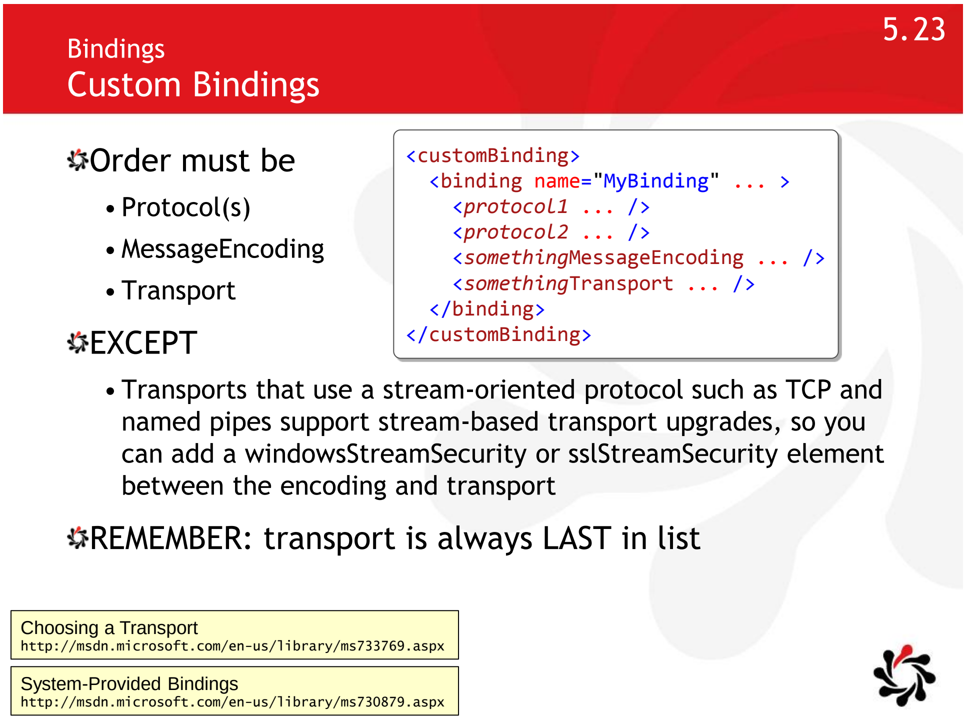
Choosing a Binding

5.22

[1] Basic Profile 1.1 as used by ASP.NET XML Web Services (.asmx)
[2] Every binding uses SOAP messages except webHttpBinding
[3] WS2007 is a later WS-* version with improved reliable sessions

***Xxx*Binding**	**Interop**	**Security (default)**	**Encoding**	**Sessions**	**WS-***	**Transactions**	**Duplex**	**Streaming**
BasicHttp	Basic [1]	None, Transport, Message	Text, Mtom	✗	✗	✗	✗	✓
WebHttp	REST [2]	None, Transport	Text	✗	✗	✗	✗	✓
WSHttp [3]	WS	None, Transport, Message	Text, Mtom	✓	✓	✓	✗	✗
WS2007Http [3]	WS	None, Transport, Message	Text, Mtom	✓	✓	✓	✗	✗
WSDualHttp	WS	None, Message	Text, Mtom	✓	✓	✓	✓	✗
WSFederationHttp	WS-Fed	None, Transport, Message	Text, Mtom	✓	✓	✓	✗	✗
NetTcp	.NET	None, Transport, Message	Binary	✓	✓	✓	✓	✓
NetNamedPipes	.NET	None, Transport	Binary	✓	✗	✓	✓	✓
NetMsmq	.NET	None, Transport, Message	Binary	✓	✗	✓	✗	✗
Others…								

System-Provided Bindings
http://msdn.microsoft.com/en-us/library/ms730879.aspx



5.23

Bindings

Custom Bindings

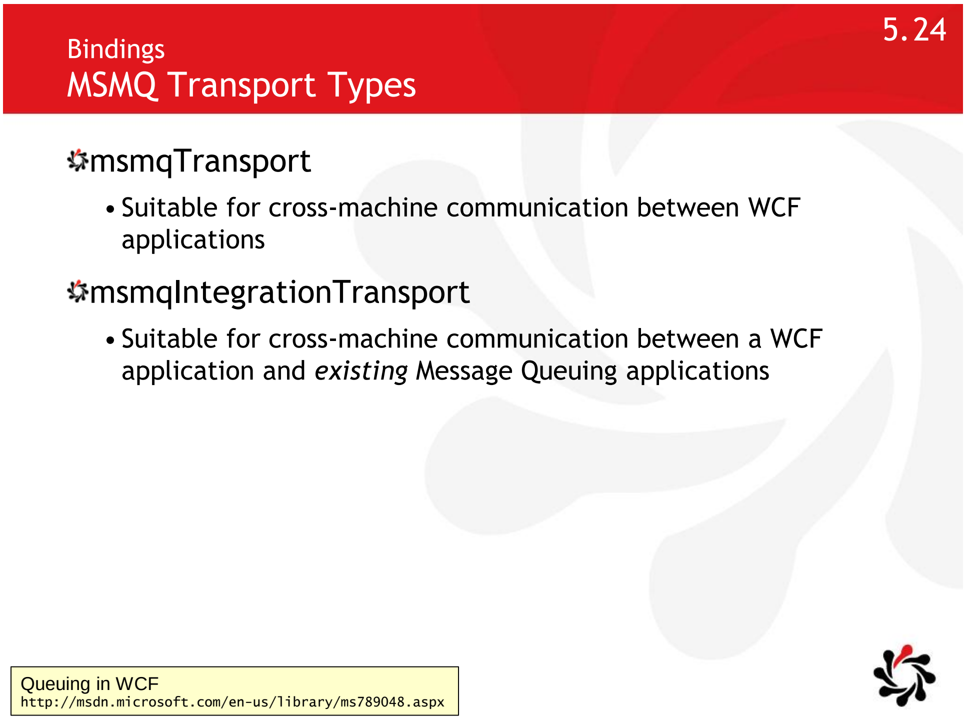
- Order must be
  - Protocol(s)
  - MessageEncoding
  - Transport

```
<customBinding>
  <binding name="MyBinding" ... >
    <protocol1 ... />
    <protocol2 ... />
    <somethingMessageEncoding ... />
    <somethingTransport ... />
  </binding>
</customBinding>
```

- EXCEPT
  - Transports that use a stream-oriented protocol such as TCP and named pipes support stream-based transport upgrades, so you can add a windowsStreamSecurity or sslStreamSecurity element between the encoding and transport
- REMEMBER: transport is always LAST in list

Choosing a Transport
http://msdn.microsoft.com/en-us/library/ms733769.aspx

System-Provided Bindings
http://msdn.microsoft.com/en-us/library/ms730879.aspx



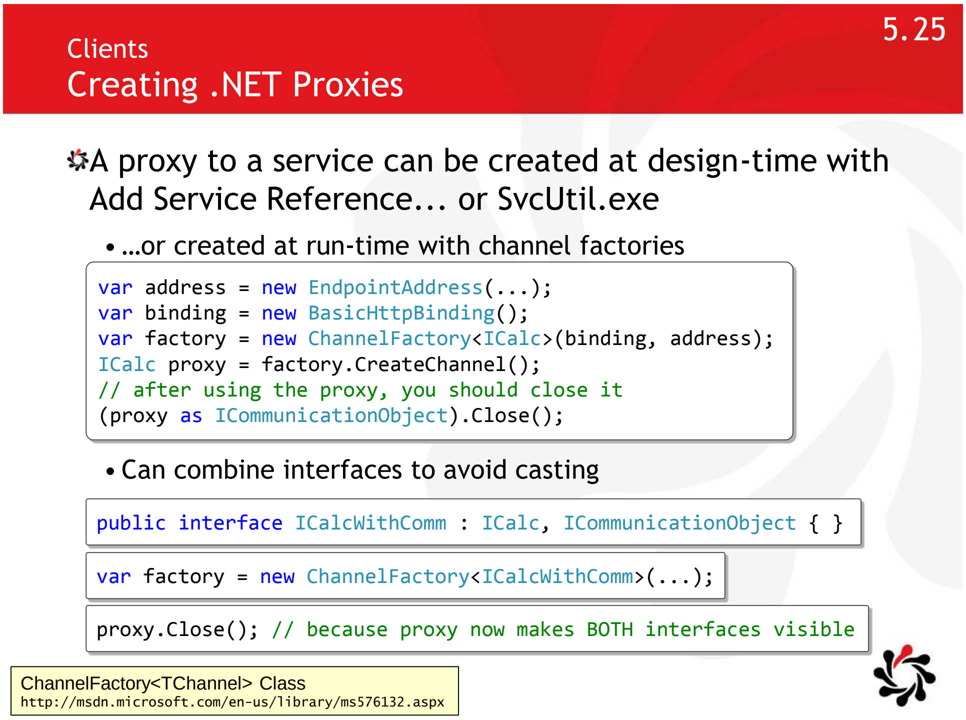
5.24

Bindings

MSMQ Transport Types

- msmqTransport
  - Suitable for cross-machine communication between WCF applications
- msmqIntegrationTransport
  - Suitable for cross-machine communication between a WCF application and *existing* Message Queuing applications

Queuing in WCF
http://msdn.microsoft.com/en-us/library/ms789048.aspx



5.25

Clients

Creating .NET Proxies

- A proxy to a service can be created at design-time with Add Service Reference… or SvcUtil.exe
  - …or created at run-time with channel factories

```
var address = new EndpointAddress(...);
var binding = new BasicHttpBinding();
var factory = new ChannelFactory<ICalc>(binding, address);
ICalc proxy = factory.CreateChannel();
// after using the proxy, you should close it
(proxy as ICommunicationObject).Close();
```

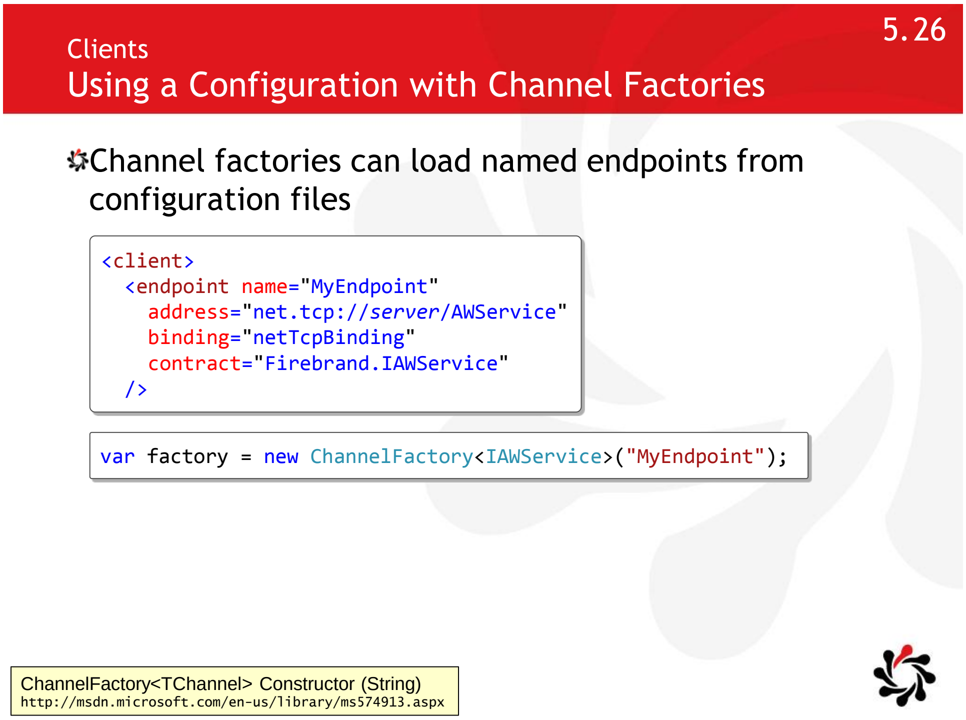
  - Can combine interfaces to avoid casting

```
public interface ICalcWithComm : ICalc, ICommunicationObject { }
```

```
var factory = new ChannelFactory<ICalcWithComm>(...);
```

```
proxy.Close(); // because proxy now makes BOTH interfaces visible
```

ChannelFactory<TChannel> Class
http://msdn.microsoft.com/en-us/library/ms576132.aspx



5.26

Clients

Using a Configuration with Channel Factories

- Channel factories can load named endpoints from configuration files

```
<client>
  <endpoint name="MyEndpoint"
    address="net.tcp://server/AWService"
    binding="netTcpBinding"
    contract="Firebrand.IAWService"
  />
```

```
var factory = new ChannelFactory<IAWService>("MyEndpoint");
```

ChannelFactory<TChannel> Constructor (String)
http://msdn.microsoft.com/en-us/library/ms574913.aspx

5.27

Clients
How To Reset a Proxy on the Client

- After an exception on the client the proxy will be in a faulted state
  - Calls to Close (or Dispose!) will cause another exception
  - Call Abort and then re-create the proxy before trying again

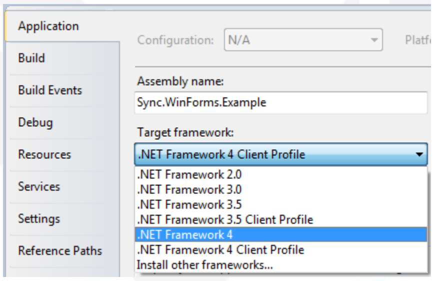
```
try
{
    proxy.ProcessData(data);
}
catch
{
    if (proxy.State == CommunicationState.Faulted)
    {
        proxy.Abort();
        proxy = new ServiceClient();
    }
}
```

ICommunicationObject.Abort Method
http://msdn.microsoft.com/en-us/library/system.servicemodel.icommunicationobject.abort.aspx

5.28

Clients
.NET Framework Client Profile (in VS2010 only)

- Subset of .NET Framework optimized for client apps
  - Does NOT include ASP.NET and advanced WCF functionality, for example, [WebGet] and [WebInvoke] methods for REST in System.ServiceModel.Web namespace
- Project templates that target the client profile
  - Empty Project
  - WPF and Windows Forms
  - Console Application
  - WCF Service Library
  - Windows Service
- To get full WCF features you need to change the target to .NET Framework 4



Configuration: N/A

Assembly name: Sync.WinForms.Example

Target framework: .NET Framework 4 Client Profile

5.29

Clients

WCF Test Client

- C:\Program Files\Microsoft Visual Studio 11.0\Common7\IDE\
  - The left pane of the main window lists all the available services, along with their respective endpoints and operations, and editable App.config file (right-click to edit)
- Supported
  - Service Invocation: Request/Response and One-way message
  - Bindings: all bindings supported by Svcutil.exe
  - Controlling Session
- NOT supported
  - Duplex, transactions, Security: Certificate & Username/Pwd
  - Bindings: WSFederation, Https, WebHttpBinding

WCF Test Client (WcfTestClient.exe)
http://msdn.microsoft.com/en-us/library/bb552364.aspx

5.30

Faults

Including Exception Details in Faults

- Set to true to troubleshoot exceptions in a service
  - As an attribute on the class implementation

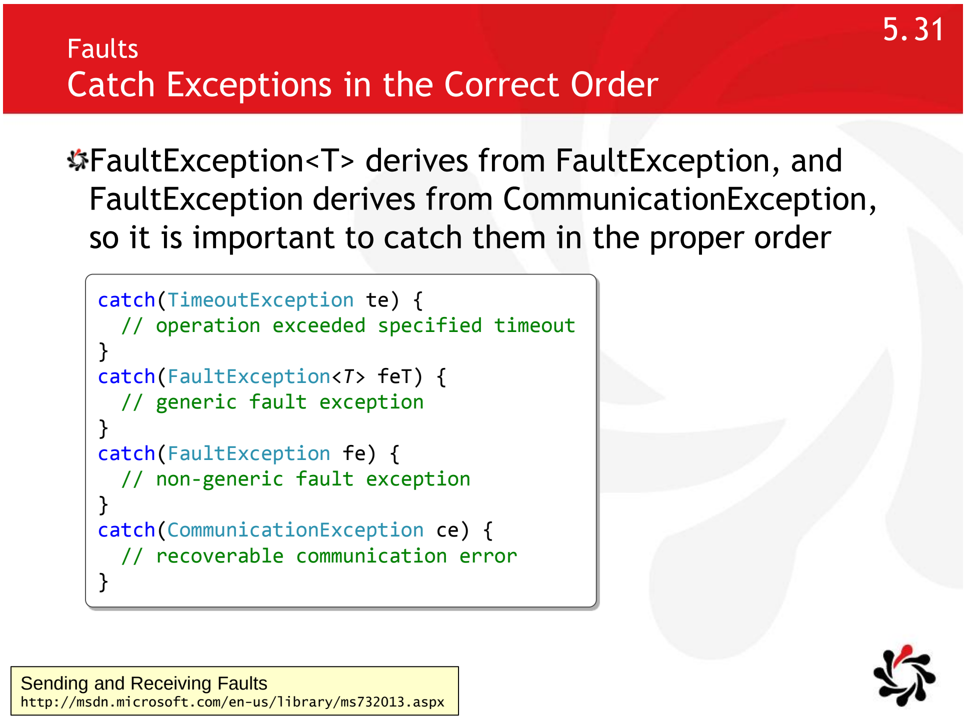
```
[ServiceBehavior(IncludeExceptionDetailInFaults = true)]
```

  - As a service behavior in .config

```
<serviceDebug includeExceptionDetailInFaults="true" />
```

- Serializes any service-side exceptions into a SOAP fault
  - Allows the service to return any exception that is thrown by the service code even if the exception is not declared using the FaultContractAttribute *including the server stack trace*
  - Specifies that general unhandled execution exceptions are to be converted into a FaultException<ExceptionDetail> and sent as a fault message

http://msdn.microsoft.com/en-us/library/system.servicemodel.servicebehaviorattribute.includeexceptiondetailinfaults.aspx



5.31

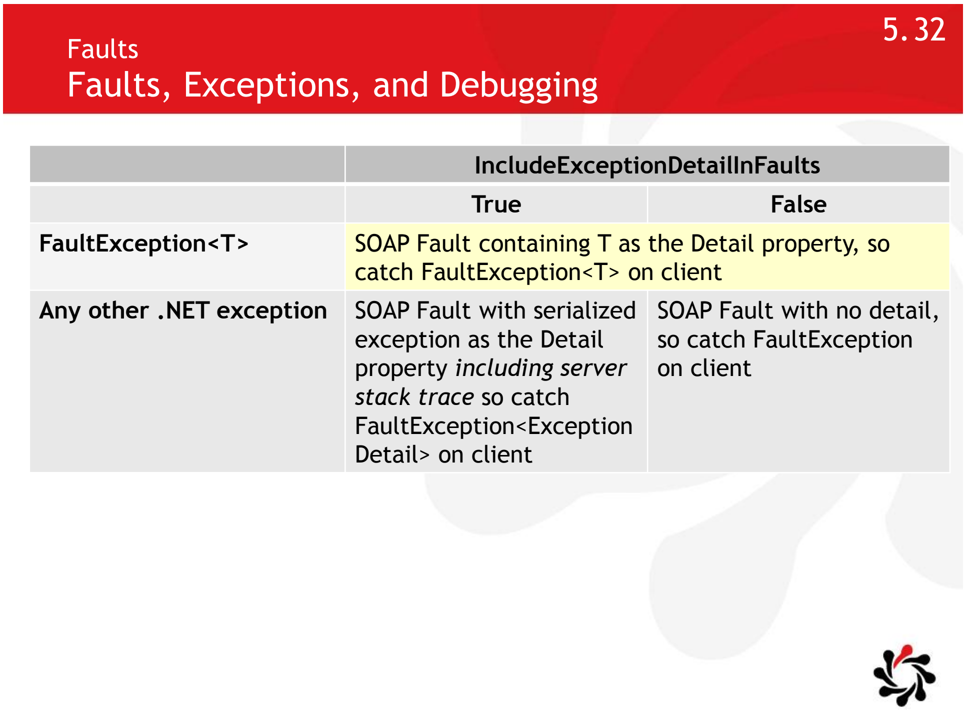
Faults

Catch Exceptions in the Correct Order

- FaultException<T> derives from FaultException, and FaultException derives from CommunicationException, so it is important to catch them in the proper order

```
catch(TimeoutException te) {
  // operation exceeded specified timeout
}
catch(FaultException<T> feT) {
  // generic fault exception
}
catch(FaultException fe) {
  // non-generic fault exception
}
catch(CommunicationException ce) {
  // recoverable communication error
}
```

Sending and Receiving Faults
http://msdn.microsoft.com/en-us/library/ms732013.aspx

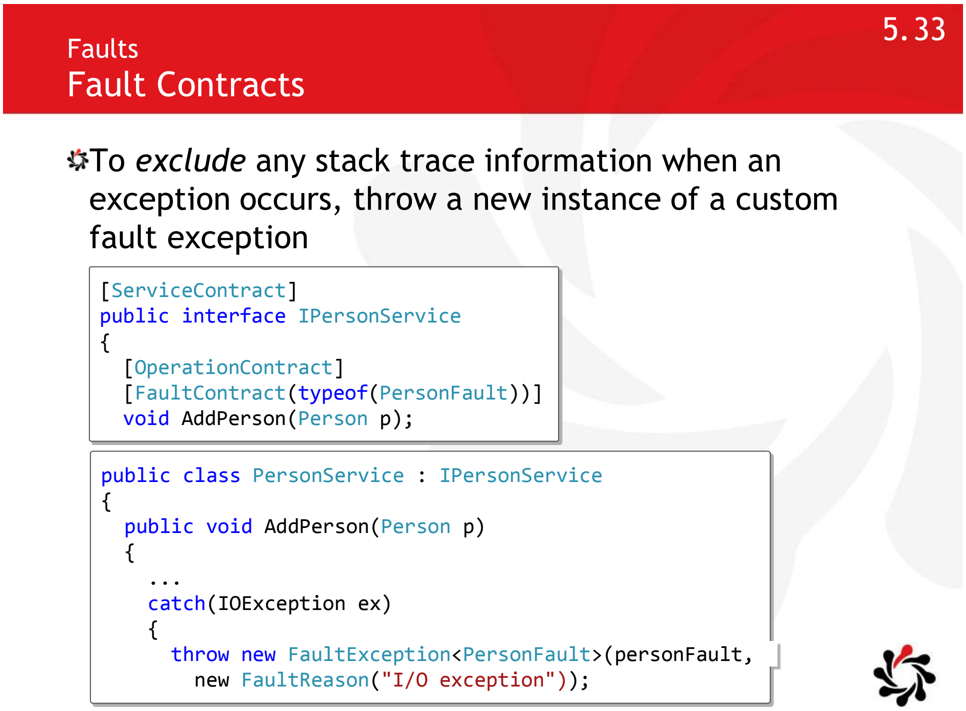


5.32

Faults

Faults, Exceptions, and Debugging

	IncludeExceptionDetailInFaults	
	True	**False**
FaultException<T>	SOAP Fault containing T as the Detail property, so catch FaultException<T> on client	
Any other .NET exception	SOAP Fault with serialized exception as the Detail property *including server stack trace* so catch FaultException<Exception Detail> on client	SOAP Fault with no detail, so catch FaultException on client



5.33

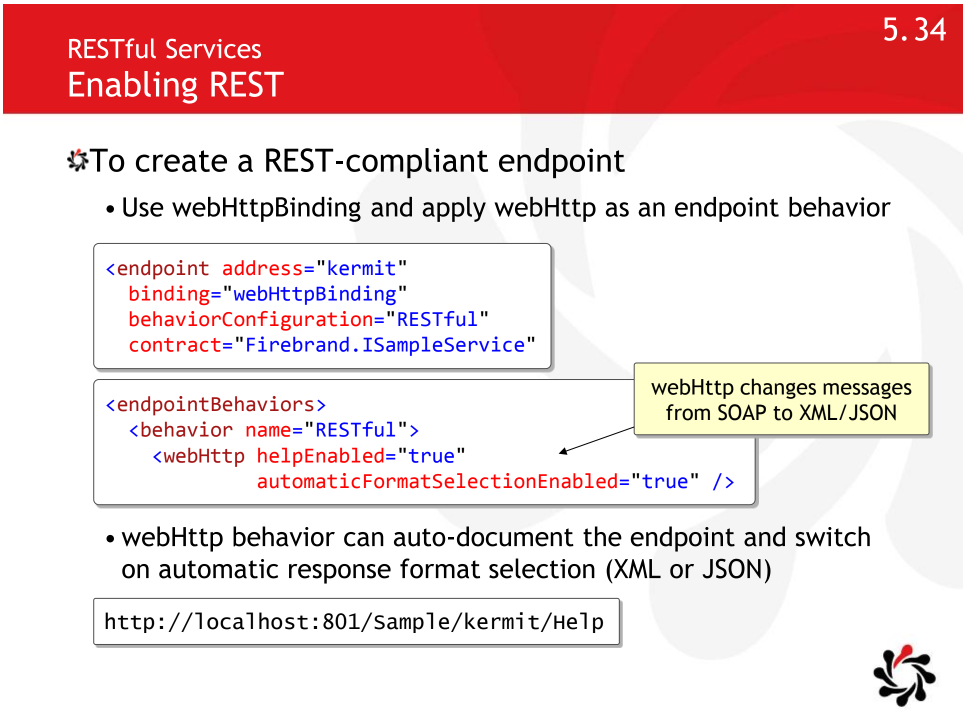
Faults

Fault Contracts

- To *exclude* any stack trace information when an exception occurs, throw a new instance of a custom fault exception

```
[ServiceContract]
public interface IPersonService
{
  [OperationContract]
  [FaultContract(typeof(PersonFault))]
  void AddPerson(Person p);
```

```
public class PersonService : IPersonService
{
  public void AddPerson(Person p)
  {
    ...
    catch(IOException ex)
    {
      throw new FaultException<PersonFault>(personFault,
        new FaultReason("I/O exception"));
```



5.34

RESTful Services

Enabling REST

- To create a REST-compliant endpoint
  - Use webHttpBinding and apply webHttp as an endpoint behavior

```
<endpoint address="kermit"
  binding="webHttpBinding"
  behaviorConfiguration="RESTful"
  contract="Firebrand.ISampleService"
```

```
<endpointBehaviors>
  <behavior name="RESTful">
    <webHttp helpEnabled="true"
             automaticFormatSelectionEnabled="true" />
```

webHttp changes messages from SOAP to XML/JSON

  - webHttp behavior can auto-document the endpoint and switch on automatic response format selection (XML or JSON)

```
http://localhost:801/Sample/kermit/Help
```

RESTful Services

Invoking REST

- SOAP only uses POST; REST uses POST (by default)
  - Or you can apply [WebInvoke] to map an operation to another HTTP verb such as PUT

WebInvoke/WebGet require System.ServiceModel.Web assembly and namespace

```
[OperationContract]
[WebInvoke(Method = "PUT")]
public string AddName(string newName);
```

```
PUT http://localhost:801/Sample/kermit/AddName
...
<newName>Alice</newName>
```

- Apply [WebGet] to pass simple types in a query string

```
[OperationContract]
[WebGet]
public string AddName(string newName);
```

```
GET http://localhost:801/Sample/kermit/AddName?newName='Alice'
```

RESTful Services

Quirks with REST Operations

- REST does not allow multiple arguments in the body without a "wrapper"
  - Since we have an operation with multiple arguments we could: (1) change to a single argument using a data contract…

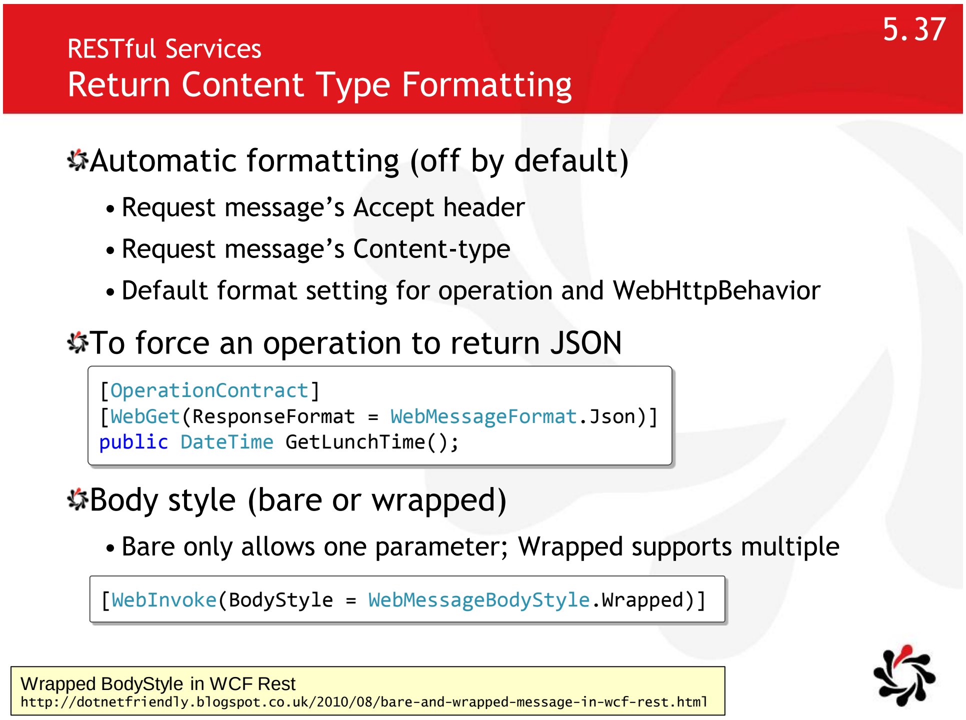
```
public class Numbers {
  public int A { get; set; }
  public int B { get; set; }
```

```
[OperationContract]
public int AddNumbers(Numbers n);
```

  - Note: this is good practice anyway since it allows versioning
  - …or (2) apply [WebGet] since the types are simple and could therefore be passed in a query string

```
[OperationContract]
[WebGet]
public int AddNumbers(int a, int b);
```

```
http://localhost:801/Sample/kermit/AddNumbers?a=4&b=5
```



5.37

RESTful Services
Return Content Type Formatting

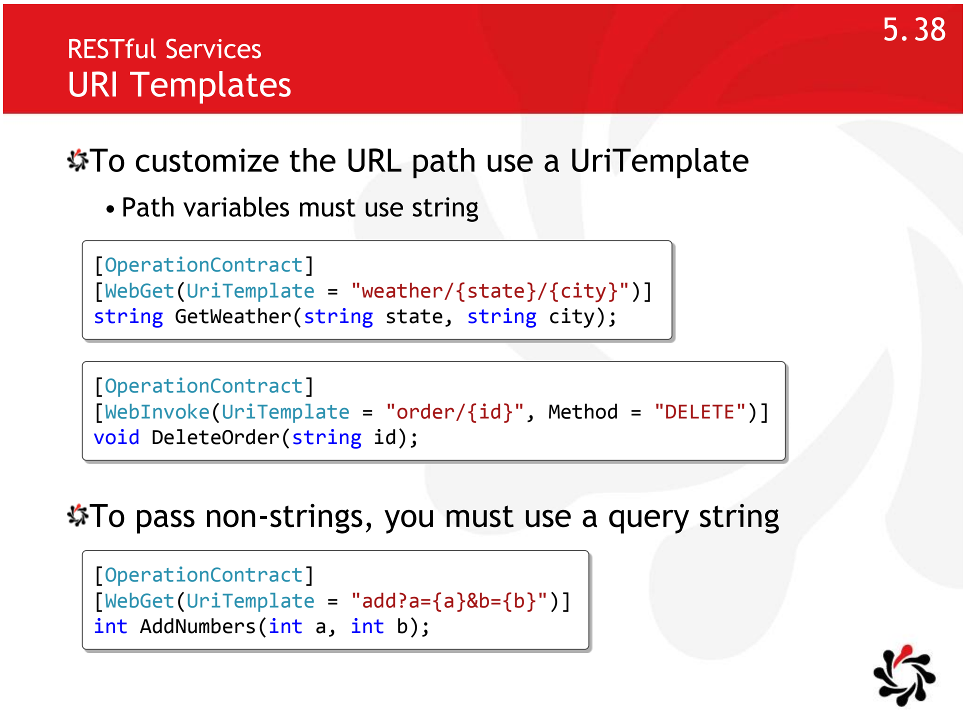
- Automatic formatting (off by default)
  - Request message’s Accept header
  - Request message’s Content-type
  - Default format setting for operation and WebHttpBehavior
- To force an operation to return JSON

```
[OperationContract]
[WebGet(ResponseFormat = WebMessageFormat.Json)]
public DateTime GetLunchTime();
```

- Body style (bare or wrapped)
  - Bare only allows one parameter; Wrapped supports multiple

```
[WebInvoke(BodyStyle = WebMessageBodyStyle.Wrapped)]
```

Wrapped BodyStyle in WCF Rest
http://dotnetfriendly.blogspot.co.uk/2010/08/bare-and-wrapped-message-in-wcf-rest.html



5.38

RESTful Services
URI Templates

- To customize the URL path use a UriTemplate
  - Path variables must use string

```
[OperationContract]
[WebGet(UriTemplate = "weather/{state}/{city}")]
string GetWeather(string state, string city);
```

```
[OperationContract]
[WebInvoke(UriTemplate = "order/{id}", Method = "DELETE")]
void DeleteOrder(string id);
```

- To pass non-strings, you must use a query string

```
[OperationContract]
[WebGet(UriTemplate = "add?a={a}&b={b}")]
int AddNumbers(int a, int b);
```

5.39

Endpoints

Define separate endpoints for different transports

Scenario

- You already have a service with a HTTP endpoint

Requirement

- Expose the service over TCP

Solution

- Leave the existing endpoint as-is and define a new endpoint for TCP

```
<service name="Firebrand.SampleService">
  <endpoint address="net.tcp://.../"
    binding="netTcpBinding"
    contract="Firebrand.ISampleService" />
  <endpoint address="http://.../"
    binding="wsHttpBinding"
    contract="Firebrand.ISampleService" />
```

5.40

Endpoints

ClientVia

A service can use a logical address while using a different address and binding for the transport

```
<client>
  <endpoint address="http://www.aw.com/Calc"
            binding="netTcpBinding"
            behaviorConfiguration="viaBehavior"
            contract="ServiceReference.ICalc" />
```

```
<endpointBehaviours>
  <behaviour name="viaBehavior">
    <clientVia viaUri="net.tcp://www.aw.com/ICalc" />
```

```
// or use code
client.Endpoint.Behaviors.Add(new ClientViaBehavior(
  new Uri("net.tcp://www.aw.com/ICalc"));
```

ClientViaBehavior Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.description.clientviabehavior.aspx

Endpoints

Why Use ClientVia?

5.41

- Facilitates debugging scenarios
  - Normally the client sends the message to the service directly and to ensure against attacks the service checks the requesting address with the actual address from which the request was sent
  - However this causes problems if you have intermediate services that might be used for debugging or other reasons because the service will receive the message from the intermediary
  - If you set the viaUri to the address of the intermediary then the service won’t reject the forwarded messages
- For another example see link below

WCF Intermediate Service between Client and Server
http://www.codeproject.com/Articles/29775/WCF-Intermediate-Service-between-Client-and-Server

Endpoints

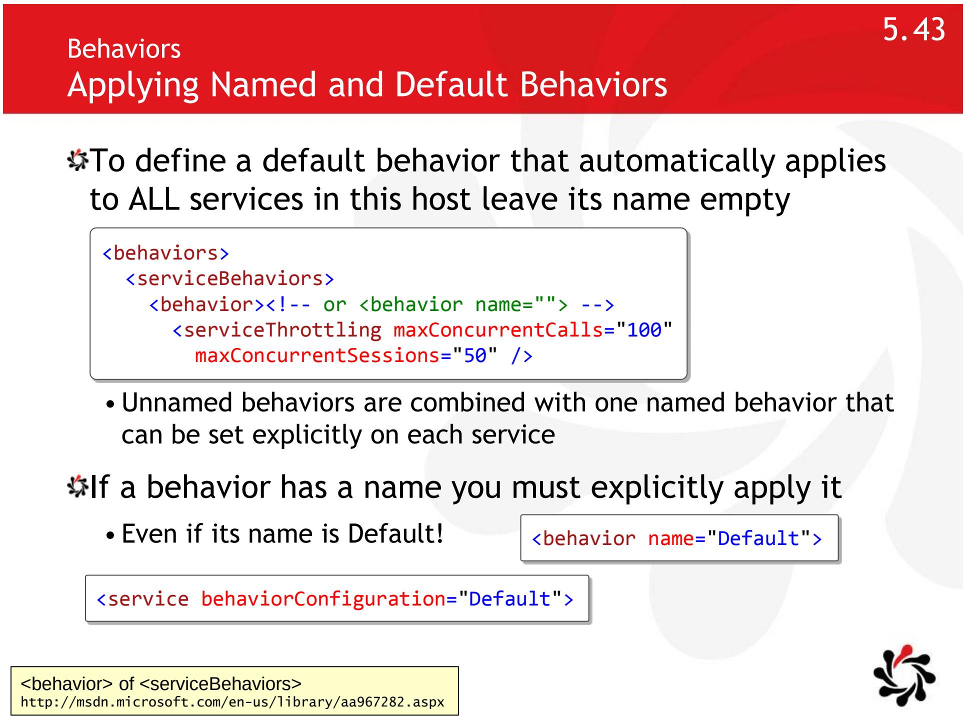
Standard Endpoints

5.42

- Standard endpoints enable a developer to define an endpoint that has default values or where one or more endpoint’s properties does not change
  - For example, to enable cross-domain script access for all instances of a web script endpoint

```
<system.serviceModel>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true" />
  <standardEndpoints>
    <webScriptEndpoint>
      <standardEndpoint crossDomainScriptAccessEnabled="true" />
```

Standard Endpoints
http://msdn.microsoft.com/en-us/library/ee358762(v=vs.110).aspx



5.43

Behaviors

Applying Named and Default Behaviors

- To define a default behavior that automatically applies to ALL services in this host leave its name empty

```
<behaviors>
  <serviceBehaviors>
    <behavior><!-- or <behavior name=""> -->
      <serviceThrottling maxConcurrentCalls="100"
        maxConcurrentSessions="50" />
```

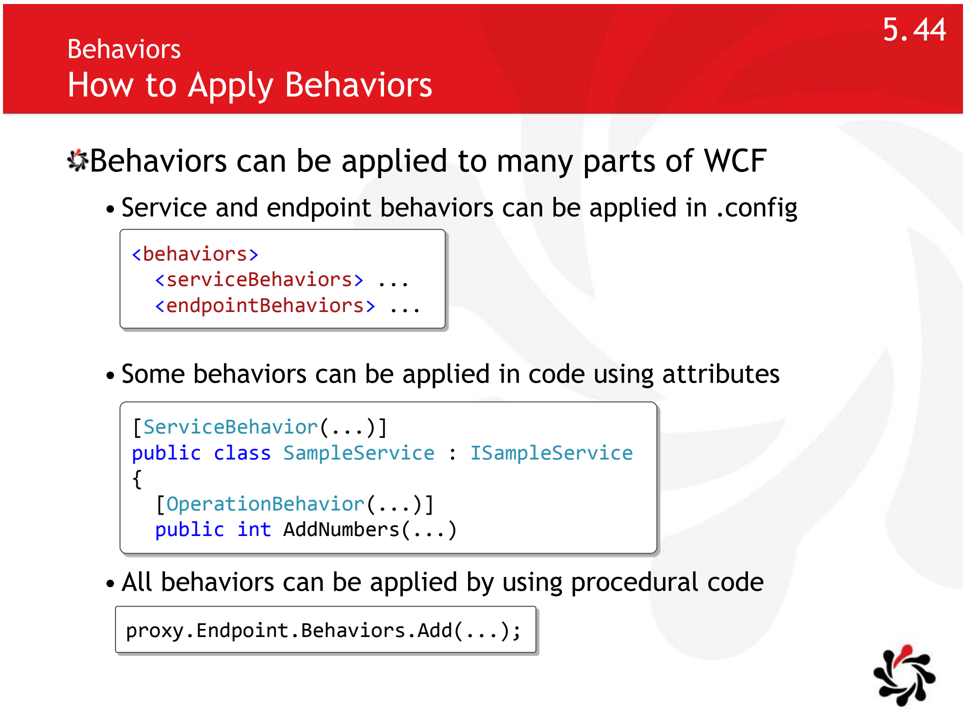
  - Unnamed behaviors are combined with one named behavior that can be set explicitly on each service

- If a behavior has a name you must explicitly apply it
  - Even if its name is Default!

```
<behavior name="Default">
```

```
<service behaviorConfiguration="Default">
```

<behavior> of <serviceBehaviors>
http://msdn.microsoft.com/en-us/library/aa967282.aspx



5.44

Behaviors

How to Apply Behaviors

- Behaviors can be applied to many parts of WCF
  - Service and endpoint behaviors can be applied in .config

```
<behaviors>
  <serviceBehaviors> ...
  <endpointBehaviors> ...
```

  - Some behaviors can be applied in code using attributes

```
[ServiceBehavior(...)]
public class SampleService : ISampleService
{
  [OperationBehavior(...)]
  public int AddNumbers(...)
```

  - All behaviors can be applied by using procedural code

```
proxy.Endpoint.Behaviors.Add(...);
```

5.45

Behaviors

ConcurrencyMode Service Behavior

- Specifies whether a service class supports single-threaded or multi-threaded modes of operation
- Single (default)
  - Service instance is single-threaded and does not accept reentrant calls
- Multiple
  - Service instance is multi-threaded
- Reentrant
  - Service instance is single-threaded and accepts reentrant calls when you call another service
  - It is your responsibility to leave your object state consistent before callouts and you must confirm that operation-local data is valid after callouts

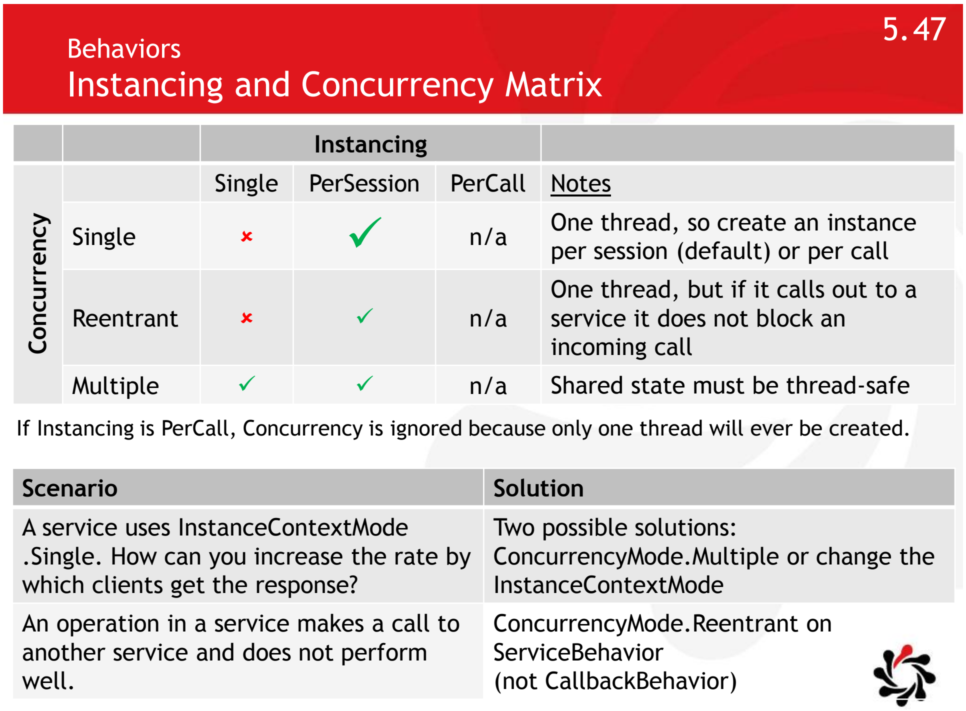
5.46

Behaviors

InstanceContextMode Service Behavior

- PerSession (default)
  - A new InstanceContext object is created for each session
- PerCall
  - A new InstanceContext object is created prior to and recycled subsequent to each call
- Single
  - Only one InstanceContext object is used for all incoming calls and is not recycled subsequent to the calls
  - If a service object is not passed to the ServiceHost constructor, one (and only one) is created automatically on first call
  - Your service can only process one message at a time unless you also set the ConcurrencyMode value to Multiple

InstanceContextMode Enumeration
http://msdn.microsoft.com/en-us/library/system.servicemodel.instancecontextmode.aspx



5.47

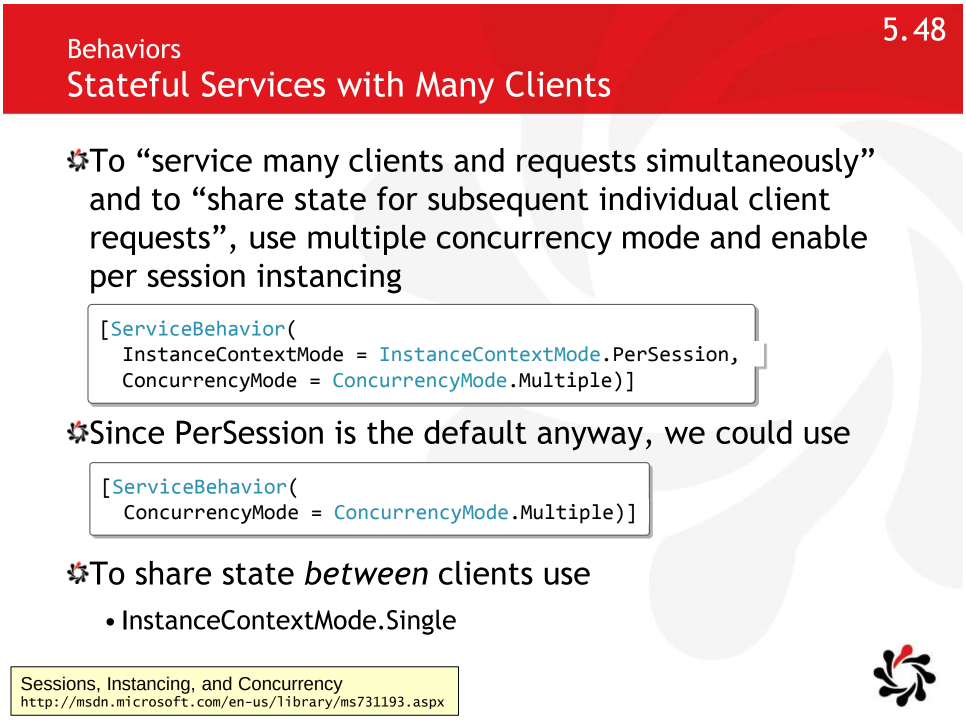
Behaviors

Instancing and Concurrency Matrix

		Instancing			
		Single	PerSession	PerCall	Notes
Concurrency	Single	✗	✓	n/a	One thread, so create an instance per session (default) or per call
	Reentrant	✗	✓	n/a	One thread, but if it calls out to a service it does not block an incoming call
	Multiple	✓	✓	n/a	Shared state must be thread-safe

If Instancing is PerCall, Concurrency is ignored because only one thread will ever be created.

Scenario	Solution
A service uses InstanceContextMode .Single. How can you increase the rate by which clients get the response?	Two possible solutions: ConcurrencyMode.Multiple or change the InstanceContextMode
An operation in a service makes a call to another service and does not perform well.	ConcurrencyMode.Reentrant on ServiceBehavior (not CallbackBehavior)



5.48

Behaviors

Stateful Services with Many Clients

- To “service many clients and requests simultaneously” and to “share state for subsequent individual client requests”, use multiple concurrency mode and enable per session instancing

```
[ServiceBehavior(
  InstanceContextMode = InstanceContextMode.PerSession,
  ConcurrencyMode = ConcurrencyMode.Multiple)]
```

- Since PerSession is the default anyway, we could use

```
[ServiceBehavior(
  ConcurrencyMode = ConcurrencyMode.Multiple)]
```

- To share state *between* clients use
  - InstanceContextMode.Single

Sessions, Instancing, and Concurrency
http://msdn.microsoft.com/en-us/library/ms731193.aspx

Behaviors

Publishing Metadata

- HTTP(S) GET metadata “endpoints” are easily enabled and use the base address

```
<serviceBehaviors>
  <behavior> <!-- and httpsGetEnabled -->
    <serviceMetadata httpGetEnabled="true" />
```

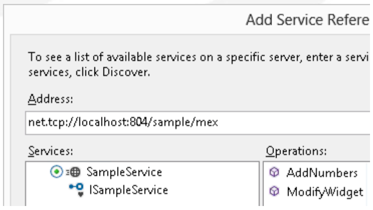
- For other metadata endpoints use WS-MetadataExchange

```
<serviceBehaviors>
  <behavior>
    <serviceMetadata />
```

```
<endpoint address="mex"
  binding="mexTcpBinding"
  contract="IMetadataExchange" />
```

Do NOT specify the full namespace i.e. NOT System.ServiceModel.Description

- mexHttpsBinding, mexNamedPipeBinding, mexTcpBinding



To see a list of available services on a specific server, enter a servi services, click Discover.

Address: net.tcp://localhost:804/sample/mex

Publishing Metadata
http://msdn.microsoft.com/en-us/library/aa751951.aspx

Behaviors

Publishing Custom Metadata

- You can specify an alternative location for returning the metadata instead of it being automatically generated

```
<serviceMetadata httpGetEnabled="true"
  externalMetadataLocation="MyCustomWsdl.wsdl" />
```

- To return custom WSDL for different scenarios, create a service and operation, and reference that instead

```
[ServiceContract]
public interface ICustomWsdl
{
    [OperationContract]
    [WebGet]
    Stream Metadata(string name, string ext);
}
```

```
externalMetadataLocation="http://.../CustomWsdl/Metadata?name=BBC&ext=wsdl"
```

Custom wsdl from WCF – David’s Blog
http://www.codehosting.net/blog/BlogEngine/post/Custom-wsdl-from-WCF.aspx

5.51

Sessions
What Is a Session?

- Correlates a group of messages into a conversation
  - Do NOT confuse with Session state in ASP.NET
- Session can be provided by the transport protocol (e.g. TCP) or by some other mechanism (e.g. WS-*)
- Although PerSession is the default instance mode, a session is only created if the binding supports sessions
  - NO support for sessions: BasicHttpBinding and WebHttpBinding
- SessionId can be read from proxy’s InnerChannel

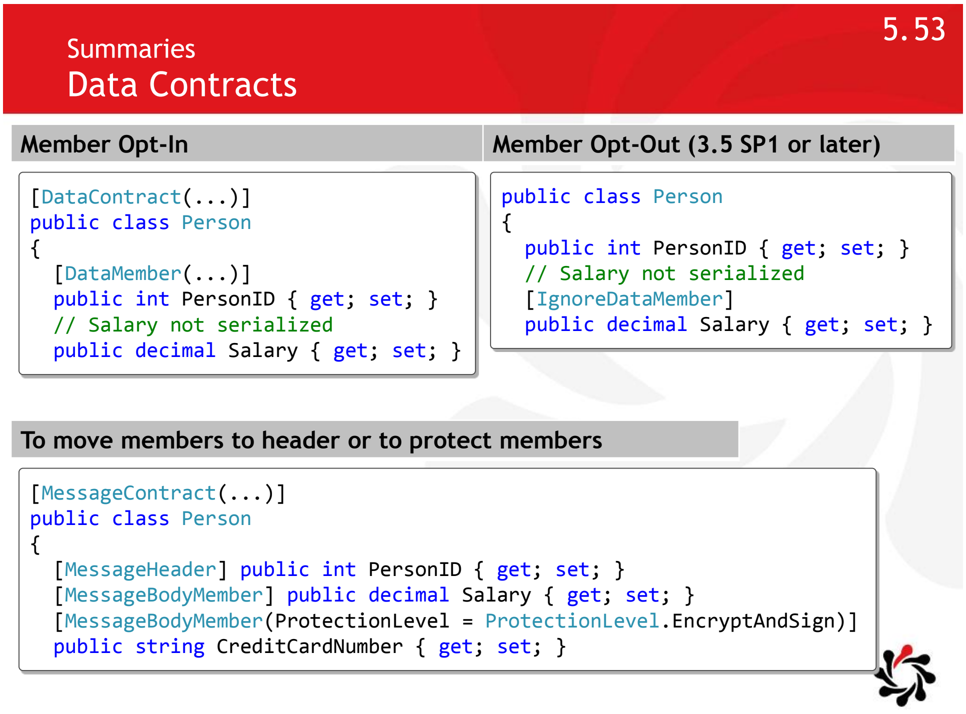
WCF Sessions - Brief Introduction
http://www.codeproject.com/Articles/188749/WCF-Sessions-Brief-Introduction

5.52

Sessions
Initiating and Terminating Operations

- Operations can be marked so they either initiate a session or terminate a session
  - Service must require sessions
  - By default, IsInitiating=true and IsTerminating=false

```
[ServiceContract(SessionMode = SessionMode.Required)]
public interface IMyService
{
    [OperationContract]
    void CreateShoppingCart();
    [OperationContract(IsInitiating = false)]
    void AddItemToCart();
    [OperationContract(IsInitiating = false, IsTerminating = true)]
    void CheckOut();
}
```



5.53

Summaries

Data Contracts

Member Opt-In

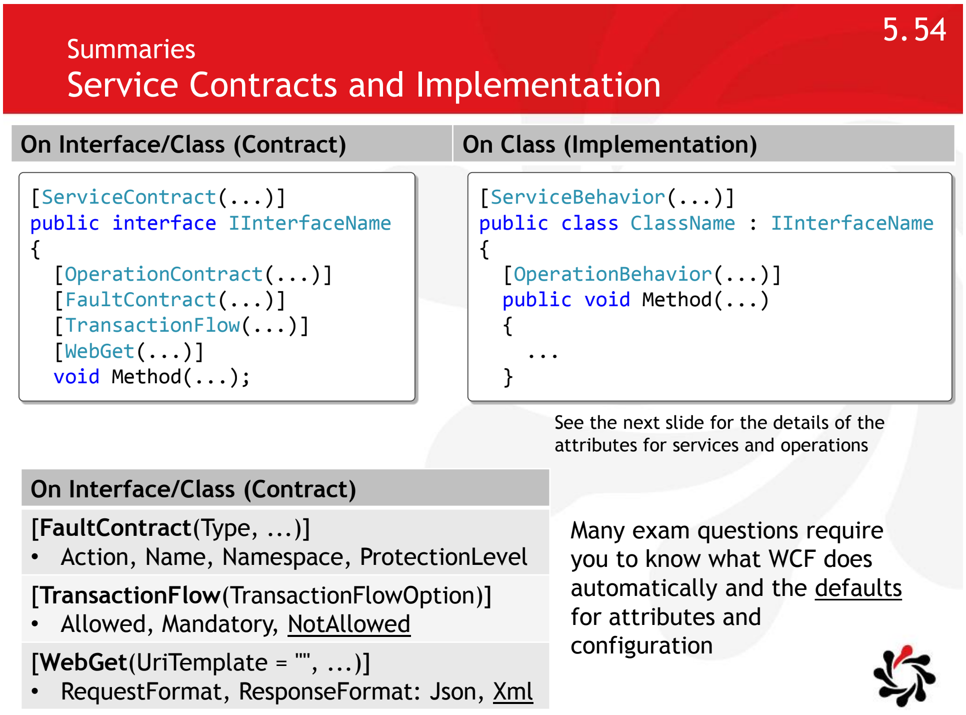
```
[DataContract(...)]
public class Person
{
  [DataMember(...)]
  public int PersonID { get; set; }
  // Salary not serialized
  public decimal Salary { get; set; }
```

Member Opt-Out (3.5 SP1 or later)

```
public class Person
{
  public int PersonID { get; set; }
  // Salary not serialized
  [IgnoreDataMember]
  public decimal Salary { get; set; }
```

To move members to header or to protect members

```
[MessageContract(...)]
public class Person
{
  [MessageHeader] public int PersonID { get; set; }
  [MessageBodyMember] public decimal Salary { get; set; }
  [MessageBodyMember(ProtectionLevel = ProtectionLevel.EncryptAndSign)]
  public string CreditCardNumber { get; set; }
```



5.54

Summaries

Service Contracts and Implementation

On Interface/Class (Contract)

```
[ServiceContract(...)]
public interface IInterfaceName
{
  [OperationContract(...)]
  [FaultContract(...)]
  [TransactionFlow(...)]
  [WebGet(...)]
  void Method(...);
```

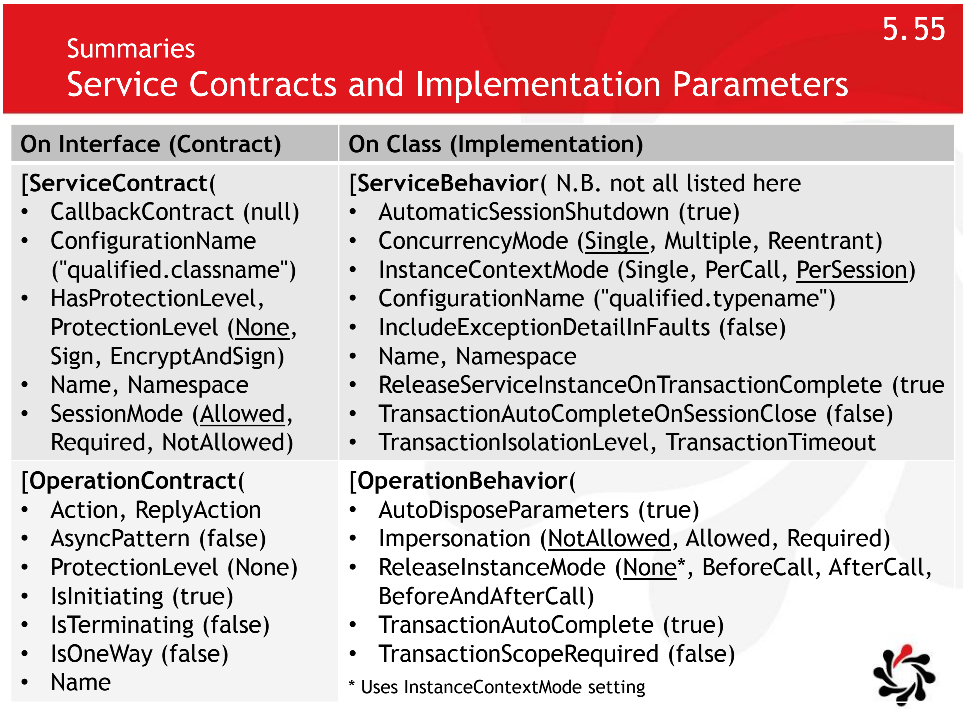
On Class (Implementation)

```
[ServiceBehavior(...)]
public class ClassName : IInterfaceName
{
  [OperationBehavior(...)]
  public void Method(...)
  {
    ...
  }
```

See the next slide for the details of the attributes for services and operations

On Interface/Class (Contract)
[FaultContract(Type, ...)] • Action, Name, Namespace, ProtectionLevel
[TransactionFlow(TransactionFlowOption)] • Allowed, Mandatory, NotAllowed
[WebGet(UriTemplate = "", ...)] • RequestFormat, ResponseFormat: Json, Xml

Many exam questions require you to know what WCF does automatically and the defaults for attributes and configuration

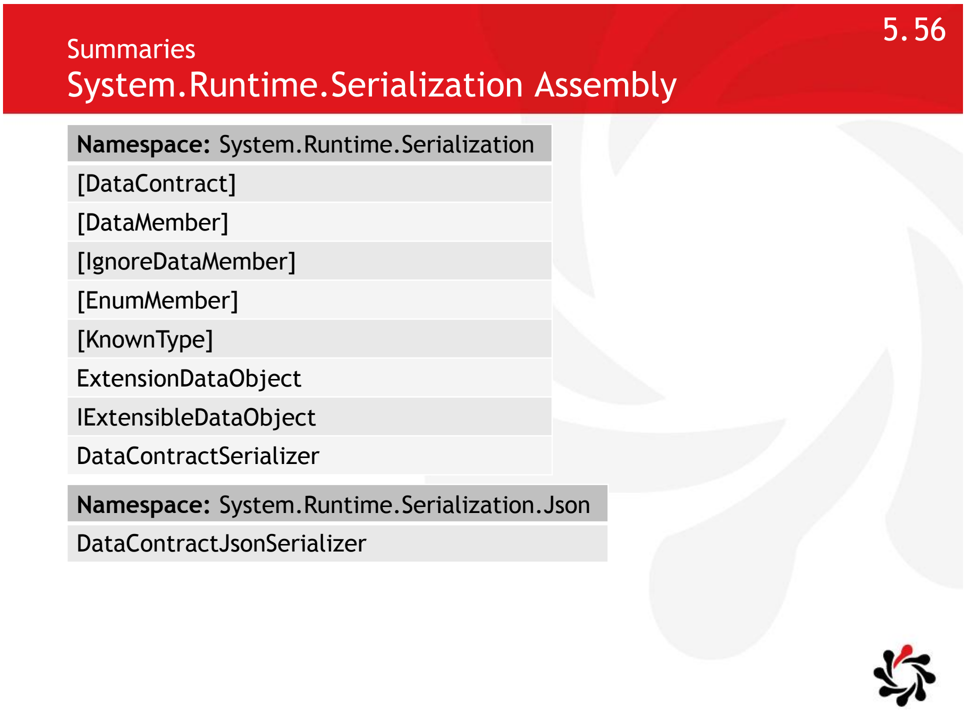


5.55

Summaries

Service Contracts and Implementation Parameters

On Interface (Contract)	On Class (Implementation)
[ServiceContract(• CallbackContract (null) • ConfigurationName ("qualified.classname") • HasProtectionLevel, ProtectionLevel (None, Sign, EncryptAndSign) • Name, Namespace • SessionMode (Allowed, Required, NotAllowed)	**[ServiceBehavior**(N.B. not all listed here • AutomaticSessionShutdown (true) • ConcurrencyMode (Single, Multiple, Reentrant) • InstanceContextMode (Single, PerCall, PerSession) • ConfigurationName ("qualified.typename") • IncludeExceptionDetailInFaults (false) • Name, Namespace • ReleaseServiceInstanceOnTransactionComplete (true • TransactionAutoCompleteOnSessionClose (false) • TransactionIsolationLevel, TransactionTimeout
[OperationContract(• Action, ReplyAction • AsyncPattern (false) • ProtectionLevel (None) • IsInitiating (true) • IsTerminating (false) • IsOneWay (false) • Name	**[OperationBehavior**(• AutoDisposeParameters (true) • Impersonation (NotAllowed, Allowed, Required) • ReleaseInstanceMode (None*, BeforeCall, AfterCall, BeforeAndAfterCall) • TransactionAutoComplete (true) • TransactionScopeRequired (false) * Uses InstanceContextMode setting



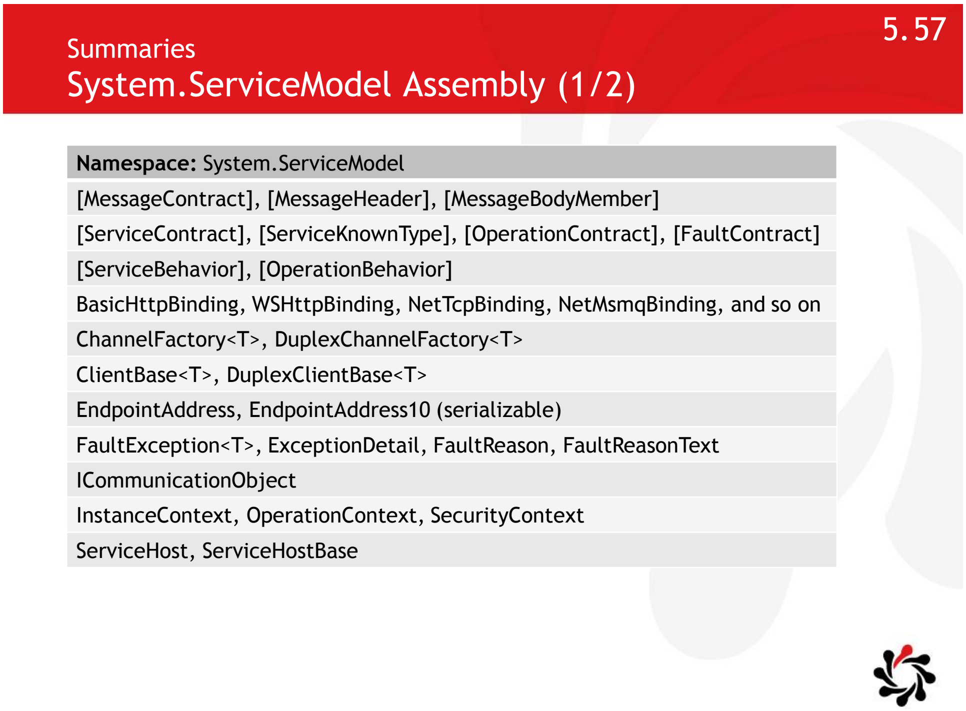
5.56

Summaries

System.Runtime.Serialization Assembly

Namespace: System.Runtime.Serialization
[DataContract]
[DataMember]
[IgnoreDataMember]
[EnumMember]
[KnownType]
ExtensionDataObject
IExtensibleDataObject
DataContractSerializer

Namespace: System.Runtime.Serialization.Json
DataContractJsonSerializer

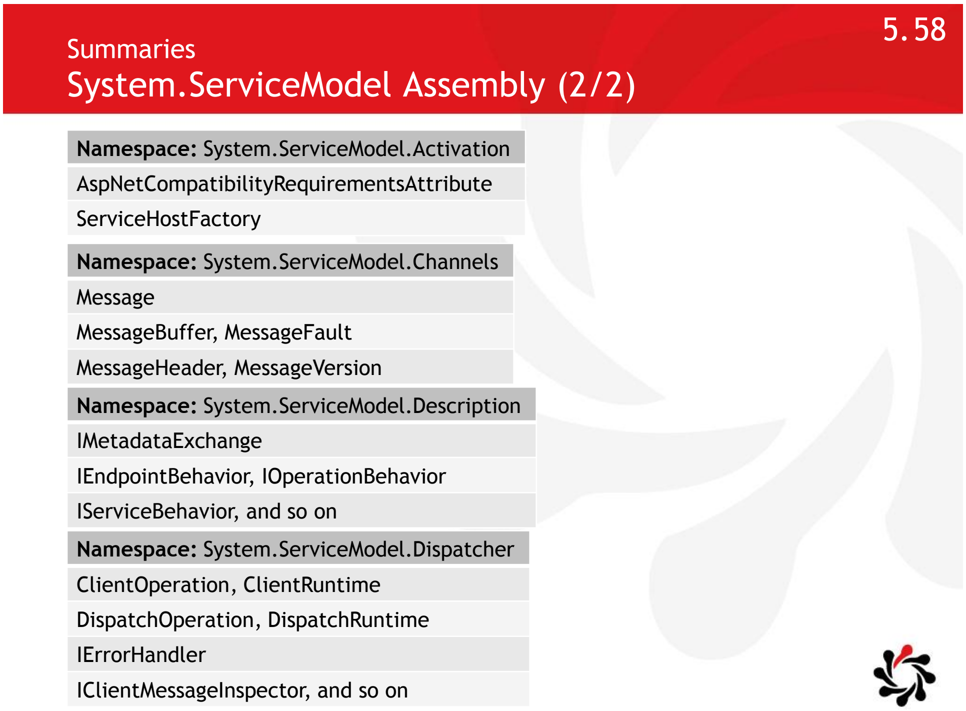


5.57

Summaries

System.ServiceModel Assembly (1/2)

Namespace: System.ServiceModel
[MessageContract], [MessageHeader], [MessageBodyMember]
[ServiceContract], [ServiceKnownType], [OperationContract], [FaultContract]
[ServiceBehavior], [OperationBehavior]
BasicHttpBinding, WSHttpBinding, NetTcpBinding, NetMsmqBinding, and so on
ChannelFactory<T>, DuplexChannelFactory<T>
ClientBase<T>, DuplexClientBase<T>
EndpointAddress, EndpointAddress10 (serializable)
FaultException<T>, ExceptionDetail, FaultReason, FaultReasonText
ICommunicationObject
InstanceContext, OperationContext, SecurityContext
ServiceHost, ServiceHostBase



5.58

Summaries

System.ServiceModel Assembly (2/2)

Namespace: System.ServiceModel.Activation
AspNetCompatibilityRequirementsAttribute
ServiceHostFactory

Namespace: System.ServiceModel.Channels
Message
MessageBuffer, MessageFault
MessageHeader, MessageVersion

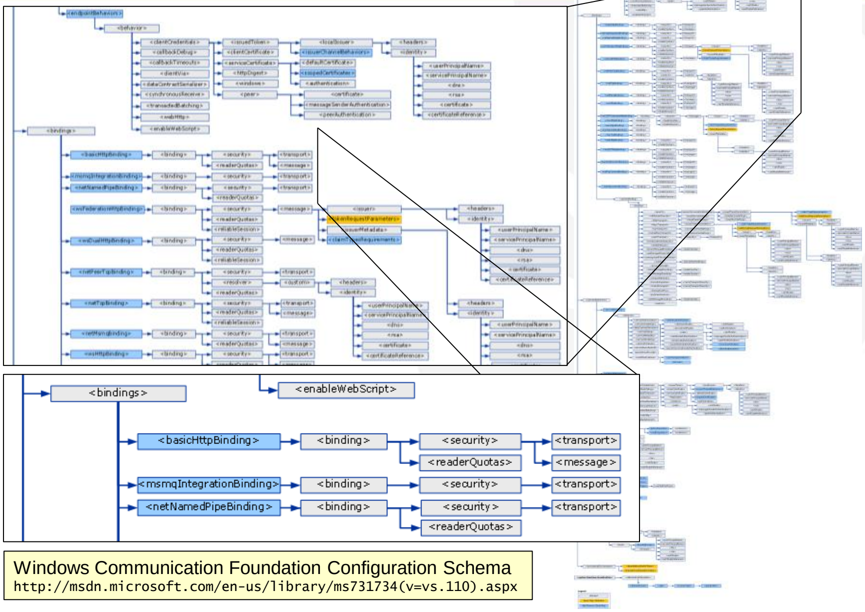
Namespace: System.ServiceModel.Description
IMetadataExchange
IEndpointBehavior, IOperationBehavior
IServiceBehavior, and so on

Namespace: System.ServiceModel.Dispatcher
ClientOperation, ClientRuntime
DispatchOperation, DispatchRuntime
IErrorHandler
IClientMessageInspector, and so on

Summaries

WCF Configuration Schema

5.59



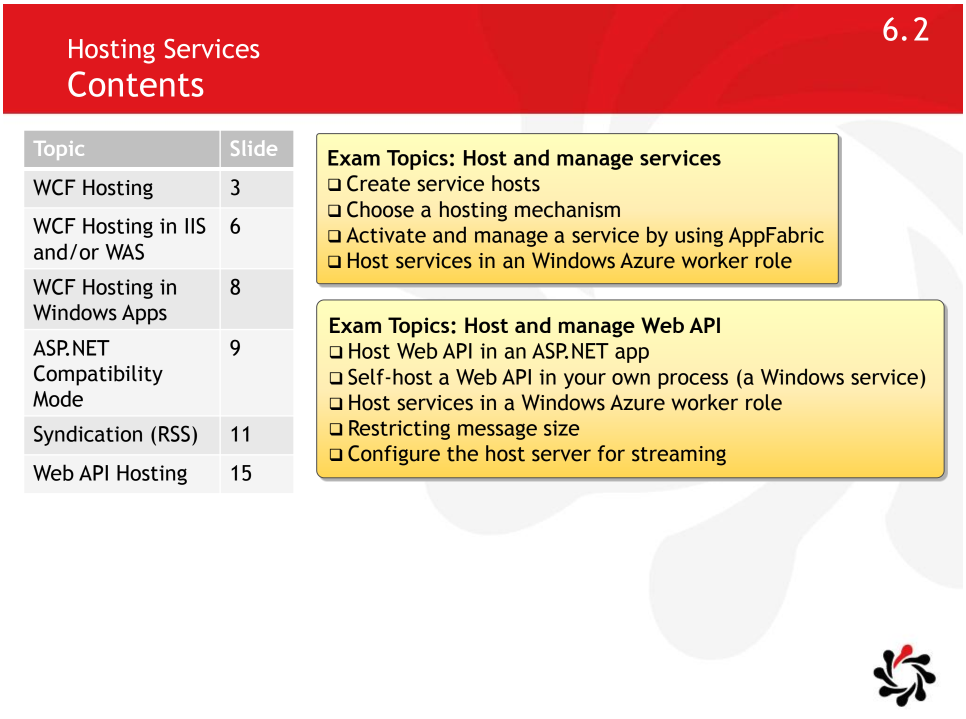
Windows Communication Foundation Configuration Schema
`http://msdn.microsoft.com/en-us/library/ms731734(v=vs.110).aspx`

6.1

Module 6
Hosting Services

Developing Windows Azure and Web Services

Updated 11th April 2014



6.2

Hosting Services

Contents

Topic	Slide
WCF Hosting	3
WCF Hosting in IIS and/or WAS	6
WCF Hosting in Windows Apps	8
ASP.NET Compatibility Mode	9
Syndication (RSS)	11
Web API Hosting	15

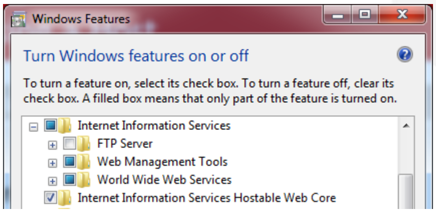
Exam Topics: Host and manage services
- ❑ Create service hosts
- ❑ Choose a hosting mechanism
- ❑ Activate and manage a service by using AppFabric
- ❑ Host services in an Windows Azure worker role

Exam Topics: Host and manage Web API
- ❑ Host Web API in an ASP.NET app
- ❑ Self-host a Web API in your own process (a Windows service)
- ❑ Host services in a Windows Azure worker role
- ❑ Restricting message size
- ❑ Configure the host server for streaming

WCF Hosting
Choosing a Host

6.3

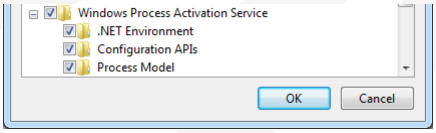
Host/Transport	HTTP	Others
IIS 6	✓	✗
IIS 7+ only	✓	✗
Windows Process Activation Service (WAS) only	✗	✓
IIS 7+ and WAS	✓	✓
Windows Service, Console, Windows application, and so on	✓	✓


Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ■ Internet Information Services
  - □ FTP Server
  - ■ Web Management Tools
  - ■ World Wide Web Services
- ☑ Internet Information Services Hostable Web Core


- ☑ Windows Process Activation Service
  - ☑ .NET Environment
  - ☑ Configuration APIs
  - ☑ Process Model

WCF Hosting
How to Instantiate a ServiceHost

6.4

- If you self-host a service, pass the type of *class name* of the service implementation into the constructor of ServiceHost

```
host = new ServiceHost(typeof(CalcService));
```

- For PerSession (default) and PerCall instance context modes the host will then automatically instantiate an instance whenever required

ServiceHost Constructor (Type, Uri[])
http://msdn.microsoft.com/en-us/library/ms585489.aspx

6.5

WCF Hosting

How to Instantiate a Singleton ServiceHost

- If you only want a single instance (aka singleton) you must use Single instancing mode

```
[ServiceBehavior(InstanceContextMode=InstanceContextMode.Single)]
public class CalcService
```

- You can leave the host to create the instance automatically on first client call or you can passing an instance into the constructor of ServiceHost

```
var calc = new CalcService();
host = new ServiceHost(calc);
```

  - An exception will be thrown if you do this without setting Single instance mode

ServiceHost Constructor (Object, Uri[])
http://msdn.microsoft.com/en-us/library/ms585487.aspx

6.6

WCF Hosting in IIS and/or WAS

Activating with .svc file

```
using System.ServiceModel;
using System.ServiceModel.Activation;
```

- Represented by .svc files

```
<%@ ServiceHost
    Service="Firebrand.CalcService"
    Factory="Firebrand.CalcHostFactory" %>
```

  - To dynamically customize the service host create a class that inherits from ServiceHostFactory and specify it as the factory

```
public class CalcHostFactory : ServiceHostFactory
```

```
protected override ServiceHost CreateServiceHost
    (Type serviceType, Uri[] baseAddresses)
```

- You do not specify a base address in .config because IIS uses the URL of the .svc file as the base address
- Deploy the service implementation DLL to bin folder

6.7

WCF Hosting in IIS and/or WAS

Activating with Configuration

- You can activate services *without* a physical file

```
<serviceHostingEnvironment>
  <serviceActivations>
    <add relativeAddress="Fake.svc"
         service="Firebrand.CalcService"
         factory="optional factory class" />
```

- File extensions supported include .svc, .xoml
  - To use others you can register them (but you don’t need to know how)

<serviceActivations>
http://msdn.microsoft.com/en-us/library/ee816902.aspx

6.8

WCF Hosting in Windows Applications

UI Thread Problems

- When using a Windows application to host a service you must ensure that the UI thread does not block the service

```
[ServiceBehavior(UseSynchronizationContext = false)]
```

- Default is true which causes deadlock problems if the ServiceHost was created on the main thread and you then call the service from the same thread

Synchronization Contexts in WCF
http://msdn.microsoft.com/en-us/magazine/cc163321.aspx

6.9

ASP.NET Compatibility Mode

What Is It?

- In compatibility mode, WCF services use the HTTP pipeline through an IHttpHandler implementation so WCF behaves identically to ASMX with respect to the following ASP.NET features (therefore must use HTTP)
  - HttpContext: can access Current and its associated state
  - File-based authorization: can be secure by attaching file system access control lists (ACLs) to the service’s .svc file
  - Configurable URL authorization: ASP.NET’s URL authorization rules are enforced for WCF requests
  - HttpModuleCollection extensibility: any HTTP module configured in the HTTP pipeline is able to operate on WCF requests both before and after service invocation
  - ASP.NET Impersonation: WCF services run using the current identity of the ASP.NET impersonated thread

6.10

ASP.NET Compatibility Mode

How to Use It

- Enable compatibility mode in .config

```
<system.serviceModel>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true" />
```

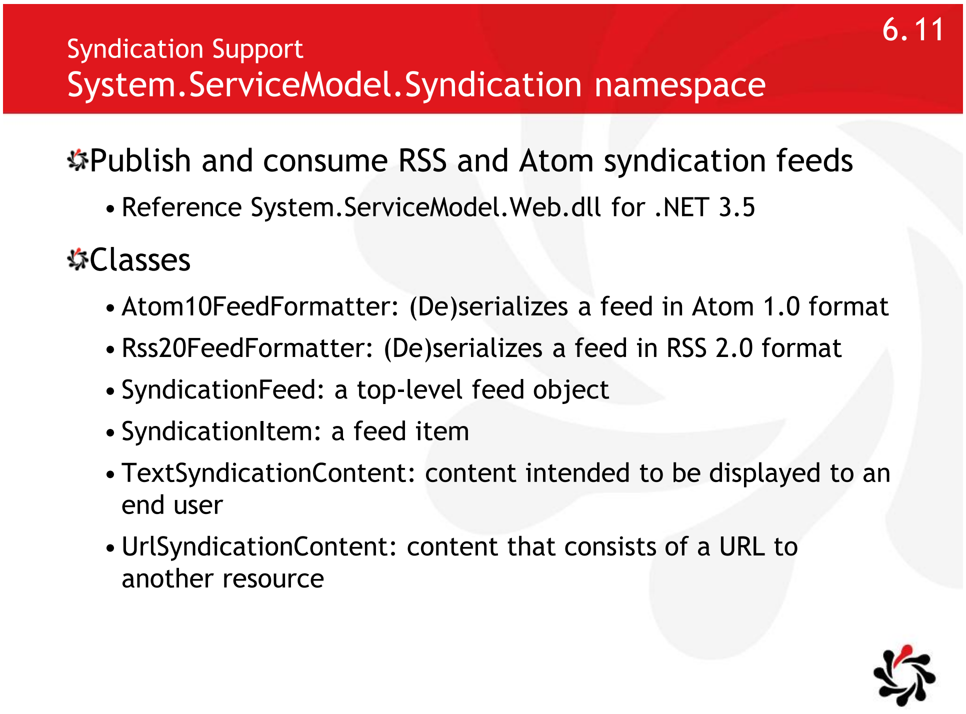
- Service implementations can control if they run if compatibility mode is enabled

```
using System.ServiceModel.Activation;
```

```
[AspNetCompatibilityRequirements(RequirementsMode =
  AspNetCompatibilityRequirementsMode.Required)]
public class CalculatorService : ICalculator
```

  - Required, Allowed, NotAllowed

- An activation error occurs when the service receives a message if the options conflict
  - Causes exception: true+NotAllowed or false+Required

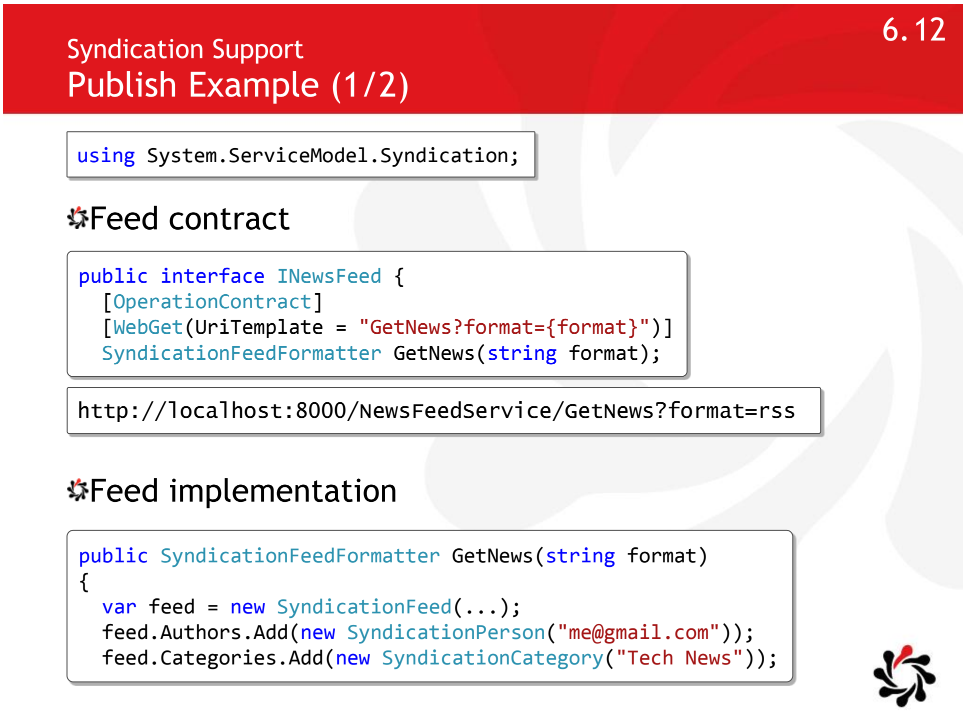


6.11

Syndication Support

System.ServiceModel.Syndication namespace

- Publish and consume RSS and Atom syndication feeds
  - Reference System.ServiceModel.Web.dll for .NET 3.5
- Classes
  - Atom10FeedFormatter: (De)serializes a feed in Atom 1.0 format
  - Rss20FeedFormatter: (De)serializes a feed in RSS 2.0 format
  - SyndicationFeed: a top-level feed object
  - SyndicationItem: a feed item
  - TextSyndicationContent: content intended to be displayed to an end user
  - UrlSyndicationContent: content that consists of a URL to another resource



6.12

Syndication Support

Publish Example (1/2)

```
using System.ServiceModel.Syndication;
```

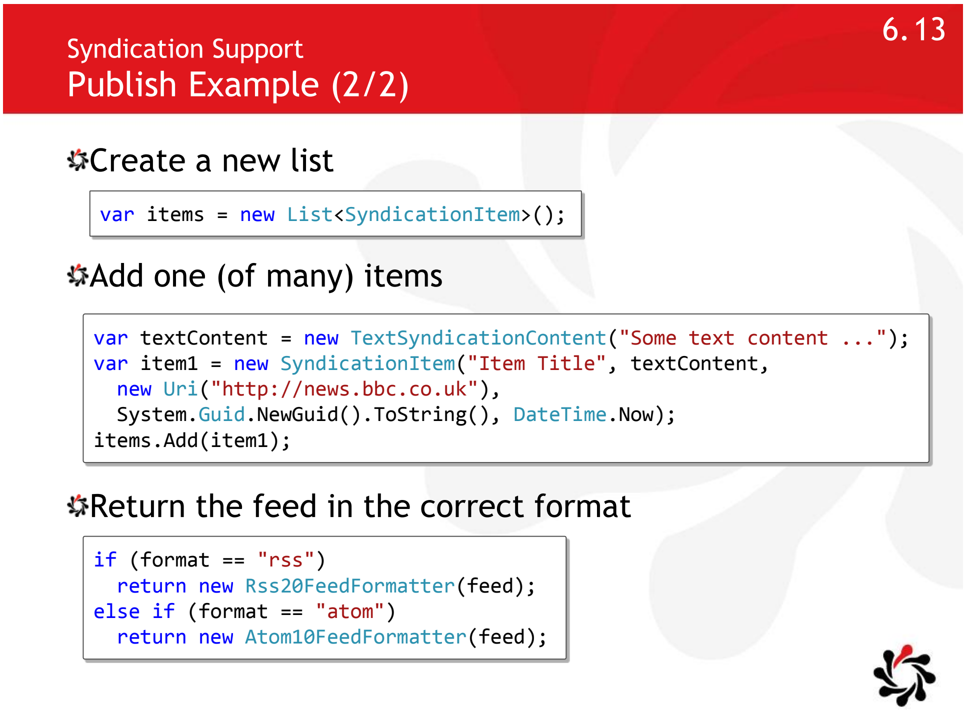
- Feed contract

```
public interface INewsFeed {
  [OperationContract]
  [WebGet(UriTemplate = "GetNews?format={format}")]
  SyndicationFeedFormatter GetNews(string format);
```

```
http://localhost:8000/NewsFeedService/GetNews?format=rss
```

- Feed implementation

```
public SyndicationFeedFormatter GetNews(string format)
{
  var feed = new SyndicationFeed(...);
  feed.Authors.Add(new SyndicationPerson("me@gmail.com"));
  feed.Categories.Add(new SyndicationCategory("Tech News"));
```



6.13

Syndication Support

Publish Example (2/2)

- Create a new list

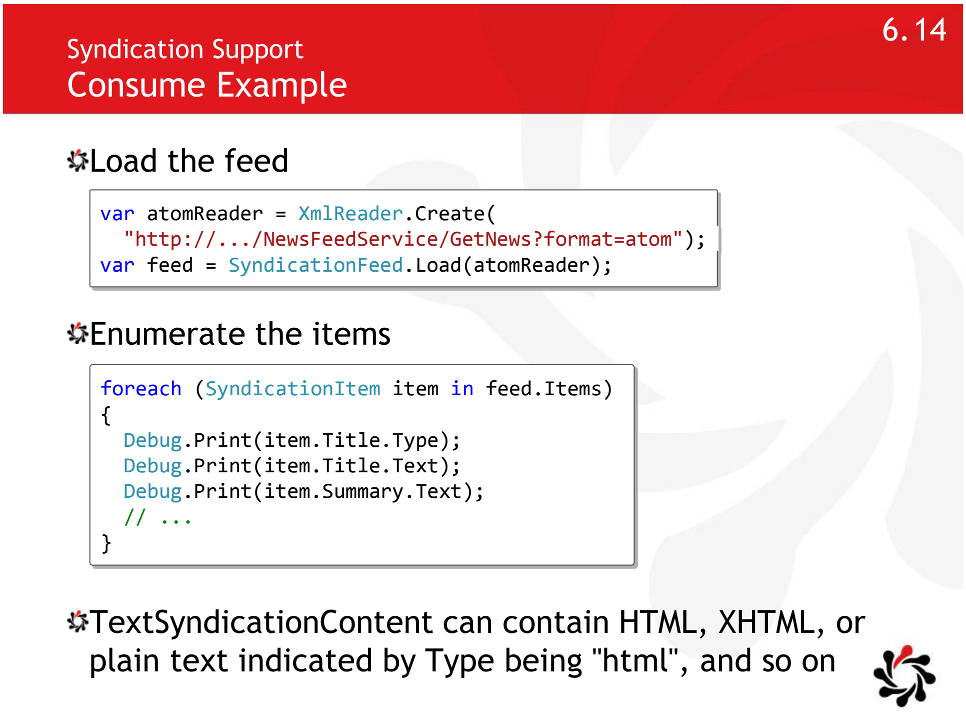
```
var items = new List<SyndicationItem>();
```

- Add one (of many) items

```
var textContent = new TextSyndicationContent("Some text content ...");
var item1 = new SyndicationItem("Item Title", textContent,
  new Uri("http://news.bbc.co.uk"),
  System.Guid.NewGuid().ToString(), DateTime.Now);
items.Add(item1);
```

- Return the feed in the correct format

```
if (format == "rss")
  return new Rss20FeedFormatter(feed);
else if (format == "atom")
  return new Atom10FeedFormatter(feed);
```



6.14

Syndication Support

Consume Example

- Load the feed

```
var atomReader = XmlReader.Create(
  "http://.../NewsFeedService/GetNews?format=atom");
var feed = SyndicationFeed.Load(atomReader);
```

- Enumerate the items

```
foreach (SyndicationItem item in feed.Items)
{
  Debug.Print(item.Title.Type);
  Debug.Print(item.Title.Text);
  Debug.Print(item.Summary.Text);
  // ...
}
```

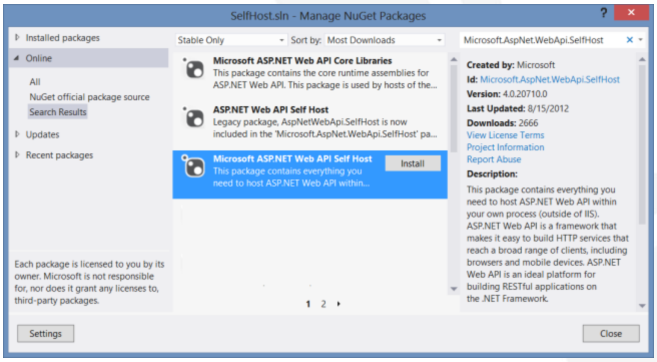
- TextSyndicationContent can contain HTML, XHTML, or plain text indicated by Type being "html", and so on

6.15

Web API Hosting

Self-Host Web API

- ASP.NET Web API does not require IIS
- You can self-host a web API in your own host process



Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Microsoft ASP.NET Web API Core Libraries
This package contains the core runtime assemblies for ASP.NET Web API. This package is used by hosts of the...

ASP.NET Web API Self Host
Legacy package, AspNetWebApi.SelfHost is now included in the 'Microsoft.AspNet.WebApi.SelfHost' pa...

Microsoft ASP.NET Web API Self Host
This package contains everything you need to host ASP.NET Web API within...

Created by: Microsoft
Id: Microsoft.AspNet.WebApi.SelfHost
Version: 4.0.20710.0
Last Updated: 8/15/2012
Downloads: 2666
View License Terms
Project Information
Report Abuse
Description:
This package contains everything you need to host ASP.NET Web API within your own process (outside of IIS). ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API is an ideal platform for building RESTful applications on the .NET Framework.

Self-Host a Web API
`http://www.asp.net/web-api/overview/hosting-aspnet-web-api/self-host-a-web-api`

6.16

Web API Hosting

Host Web API in Windows Azure

- Open Web Interface for .NET defines an abstraction between .NET web servers and web applications
  - OWIN decouples the web application from the server, which makes OWIN ideal for self-hosting a web application in your own process, outside of IIS-for example, inside an Azure worker role

Host ASP.NET Web API in an Azure Worker Role
`http://www.asp.net/web-api/overview/hosting-aspnet-web-api/host-aspnet-web-api-in-an-azure-worker-role`

7.1

Module 7
Windows Azure Service Bus

Developing Windows Azure and Web Services

Updated 11th April 2014

7.2

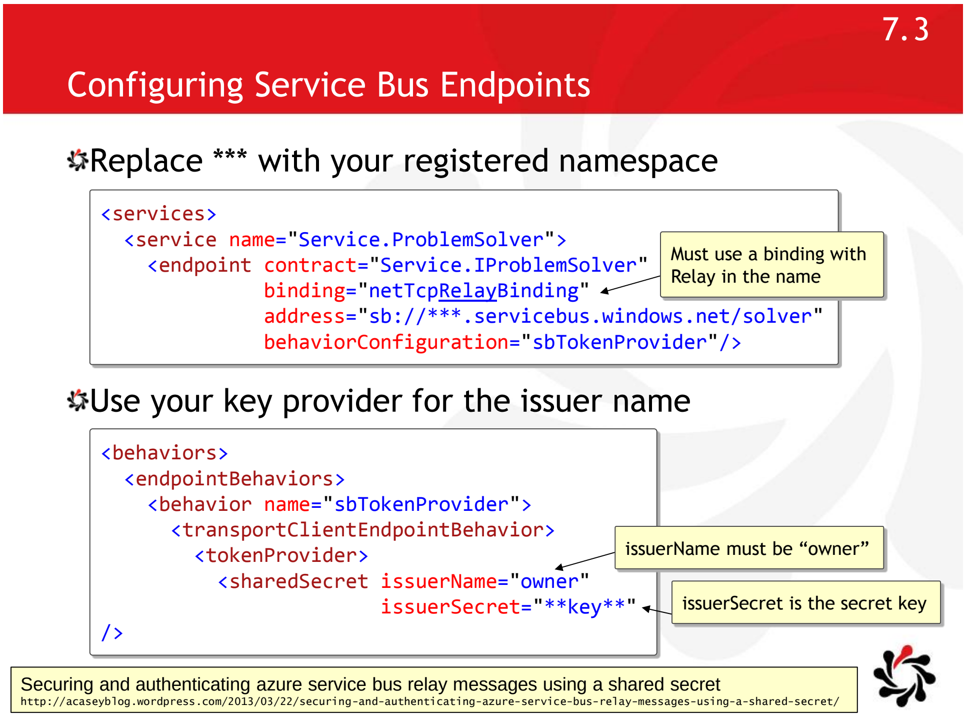
Windows Azure Service Bus

Contents

Exam Topics: Create and configure a WCF service on Windows Azure
❑ Create and configure bindings for WCF services (Azure SDK extensions to WCF)
❑ Relay bindings to Azure using service bus endpoints
❑ Integrate with the Azure service bus relay

Exam Topics: Implement messaging patterns
❑ Implement Windows Azure Service Bus and Windows Azure Queues

How to Use the Service Bus Relay Service
`http://www.windowsazure.com/en-us/develop/net/how-to-guides/service-bus-relay/`



7.3

Configuring Service Bus Endpoints

- Replace *** with your registered namespace

```
<services>
  <service name="Service.ProblemSolver">
    <endpoint contract="Service.IProblemSolver"
              binding="netTcpRelayBinding"
              address="sb://***.servicebus.windows.net/solver"
              behaviorConfiguration="sbTokenProvider"/>
```

Must use a binding with Relay in the name

- Use your key provider for the issuer name

```
<behaviors>
  <endpointBehaviors>
    <behavior name="sbTokenProvider">
      <transportClientEndpointBehavior>
        <tokenProvider>
          <sharedSecret issuerName="owner"
                        issuerSecret="**key**"
/>
```

issuerName must be “owner”

issuerSecret is the secret key

Securing and authenticating azure service bus relay messages using a shared secret
http://acaseyblog.wordpress.com/2013/03/22/securing-and-authenticating-azure-service-bus-relay-messages-using-a-shared-secret/

8.1

Module 8
Deploying Services

Developing Windows Azure and Web Services

Updated 11th April 2014

8.2

Deploying Services

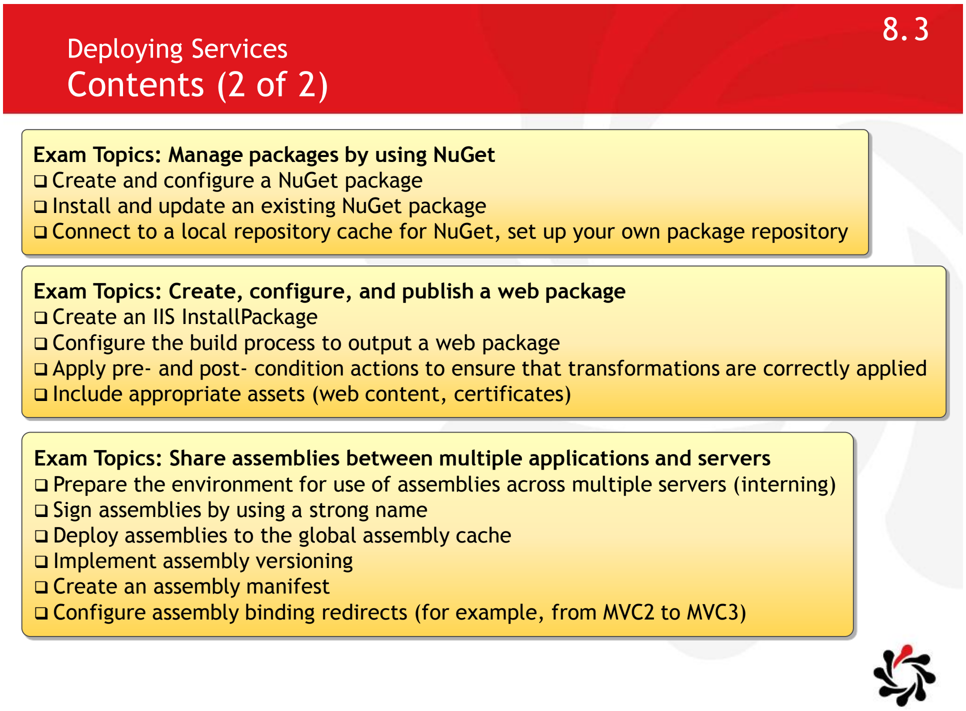
Contents (1 of 2)

Topic	Slide
Designing for Deployment	4
Deployment Tools	5
Windows Azure	6
NuGet	8

Exam Topics: Design a deployment strategy
- □ Create an IIS install package
- □ Deploy to web farms
- □ Deploy a web application by using Xcopy
- □ Automate a deployment from TFS or Build Server

Exam Topics: Configure a web application for deployment
- □ Switch from production/release mode to debug mode
- □ Use SetParameters to set up an IIS app pool, set permissions and passwords)
- □ Configure WCF endpoints, bindings, and behaviors
- □ Transform web.config by using XSLT (for example, across development, test, and production/release environments)
- □ Configure Azure configuration settings

Exam Topics: Choose a deployment strategy for a Windows Azure web application
- □ Perform an in-place upgrade and VIP swap
- □ Configure an upgrade domain
- □ Create and configure input and internal endpoints
- □ Specify operating system configuration



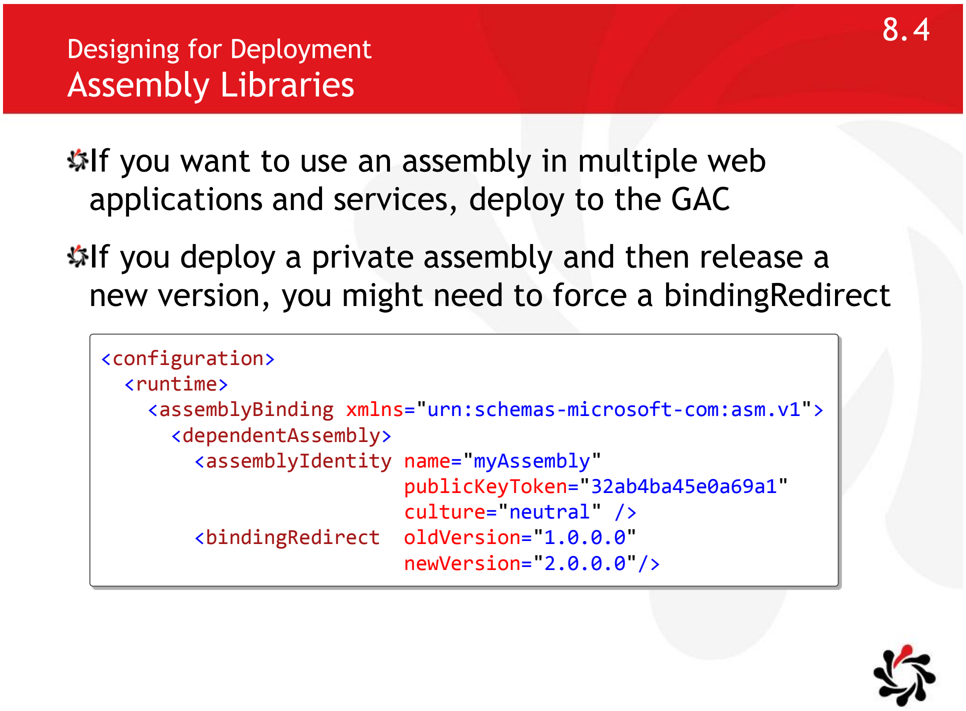
8.3

Deploying Services
Contents (2 of 2)

Exam Topics: Manage packages by using NuGet
□ Create and configure a NuGet package
□ Install and update an existing NuGet package
□ Connect to a local repository cache for NuGet, set up your own package repository

Exam Topics: Create, configure, and publish a web package
□ Create an IIS InstallPackage
□ Configure the build process to output a web package
□ Apply pre- and post- condition actions to ensure that transformations are correctly applied
□ Include appropriate assets (web content, certificates)

Exam Topics: Share assemblies between multiple applications and servers
□ Prepare the environment for use of assemblies across multiple servers (interning)
□ Sign assemblies by using a strong name
□ Deploy assemblies to the global assembly cache
□ Implement assembly versioning
□ Create an assembly manifest
□ Configure assembly binding redirects (for example, from MVC2 to MVC3)



8.4

Designing for Deployment
Assembly Libraries

- If you want to use an assembly in multiple web applications and services, deploy to the GAC
- If you deploy a private assembly and then release a new version, you might need to force a bindingRedirect

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="myAssembly"
                          publicKeyToken="32ab4ba45e0a69a1"
                          culture="neutral" />
        <bindingRedirect  oldVersion="1.0.0.0"
                          newVersion="2.0.0.0"/>
```

8.5

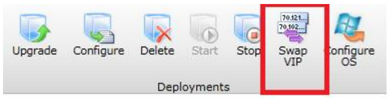
Deployment Tools
Web Deploy

- For any question about deployment tools, the answer is almost always use Web Deploy because
  - It works securely
  - It is powerful and flexible by changing the web publish pipeline
  - You can install SSL certificates using a custom target
- Only choose to use FTP, XCopy, VPN, SSH, and so on if you have a very good reason

8.6

Windows Azure
Swapping Staging and Production

- A packaged application can be deployed to the staging environment in Windows Azure to be tested before you move it to the production environment in which the application is accessible on the Internet
- The staging environment is exactly like the production environment, except that you can only access the staged application with an obfuscated (GUID-based) URL that is generated by Windows Azure
- After you have verified that your application is working correctly, it can be deployed to the production environment by performing a Virtual IP (VIP) swap



8.7

Windows Azure
Endpoints in Service Definition Files

The Windows Azure service definition file

- Contains the definitions for the roles available to a service, specifies the service endpoints, and establishes configuration

```
<ServiceDefinition ...>
  <WebRole name="web-role-name" ...
    <Endpoints>
      <InputEndpoint name="endpoint-name" protocol="HTTP" port="80" ...
      <InternalEndpoint ...
      <InstanceInputEndpoint ...
```

- InputEndpoint: an endpoint to a role from the external world
- InternalEndpoint: available only to other role instances running within the service
- InstanceInputEndpoint: associated with a specific role instance by using port forwarding in the load balancer

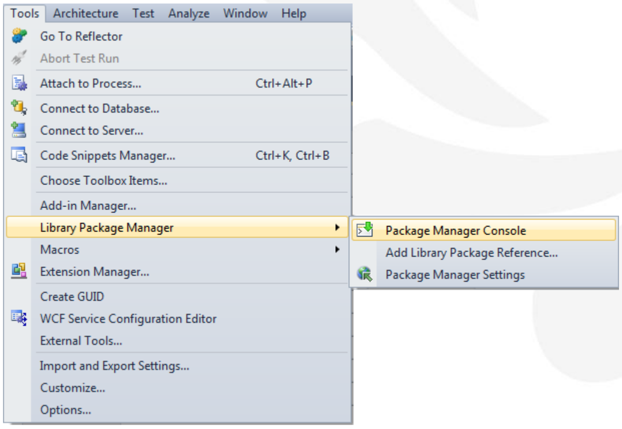
WebRole Schema
http://msdn.microsoft.com/en-us/library/windowsazure/gg557553.aspx

8.8

NuGet
What Is It?

NuGet is a Visual Studio extension that makes it easy to install and update third-party libraries and tools

- Choose “Library Package Manager” from the Tools menu

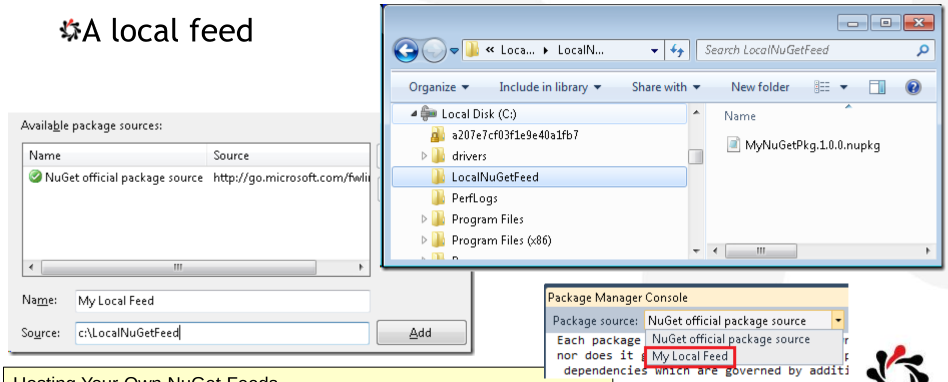


8.9

NuGet

Hosting Your Own NuGet Feeds

- You can configure Visual Studio to offer that feed instead of or in addition to the official feed
  - A feed can be local (a folder on the local machine or a network folder) or remote (an internet URL)
- A local feed



Available package sources:

Name	Source
NuGet official package source	http://go.microsoft.com/fwli

Name: My Local Feed

Source: c:\LocalNuGetFeed

Add

Package Manager Console

Package source: NuGet official package source

NuGet official package source

My Local Feed

```
Each package
nor does it
dependencies which are governed by additi
```

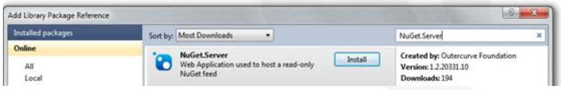
Hosting Your Own NuGet Feeds
http://docs.nuget.org/docs/creating-packages/hosting-your-own-nuget-feeds

8.10

NuGet

Creating Remote Feeds

- Install the NuGet.Server Package into a new ASP.NET Empty Application project



Sort by: Most Downloads

NuGet.Server

NuGet.Server
Web Application used to host a read-only NuGet feed

Created by: Outercurve Foundation
Version: 1.2.20331.10
Downloads: 194

  - The NuGet.Server package just converted your empty website into a site that’s ready to serve up the OData package feed
- Configure the Packages folder

```
<appSettings>
    <!-- Set the value here to specify your custom packages folder. -->
    <add key="packagesPath" value="C:\MyPackages" />
```

  - Just add packages into the Packages folder and they’ll show up

Module 9
Windows Azure Storage

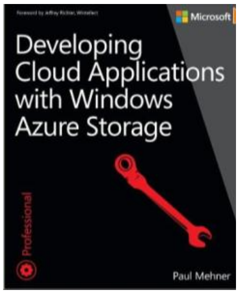
Developing Windows Azure and Web Services

Updated 11th April 2014

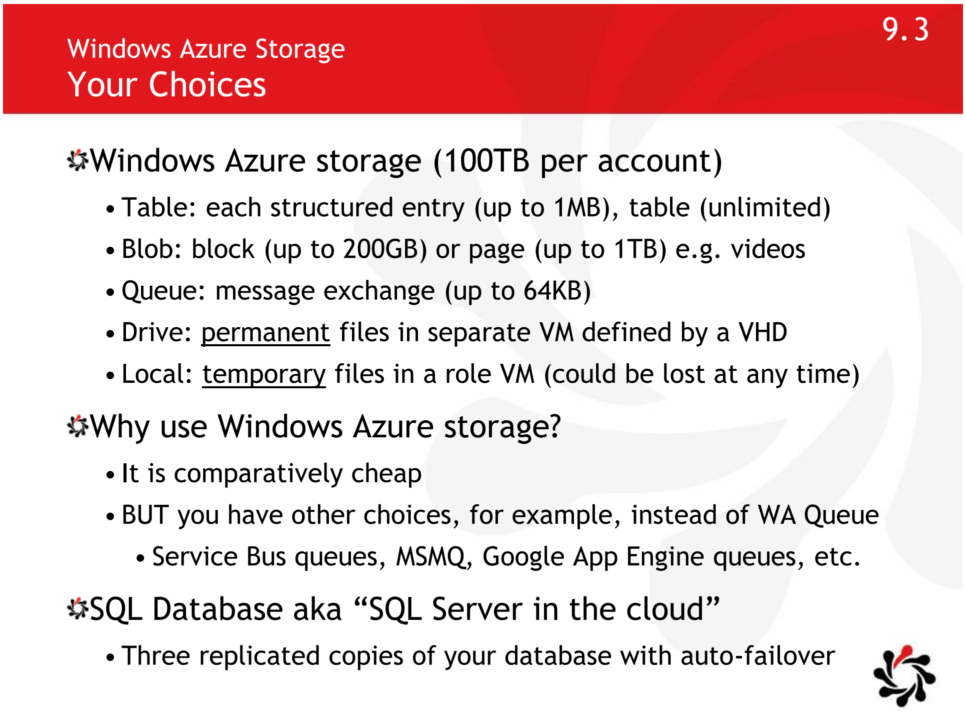
Windows Azure Storage

Contents

Exam Topic: Implement data storage in Windows Azure
□ Access data storage in Windows Azure
□ Choose data storage mechanism in Windows Azure (blobs, tables, queues, SQL Database)
□ Distribute data by using the Content delivery network (CDN)
□ Handle exceptions by using retries (SQL Database)
□ Manage Windows Azure Caching



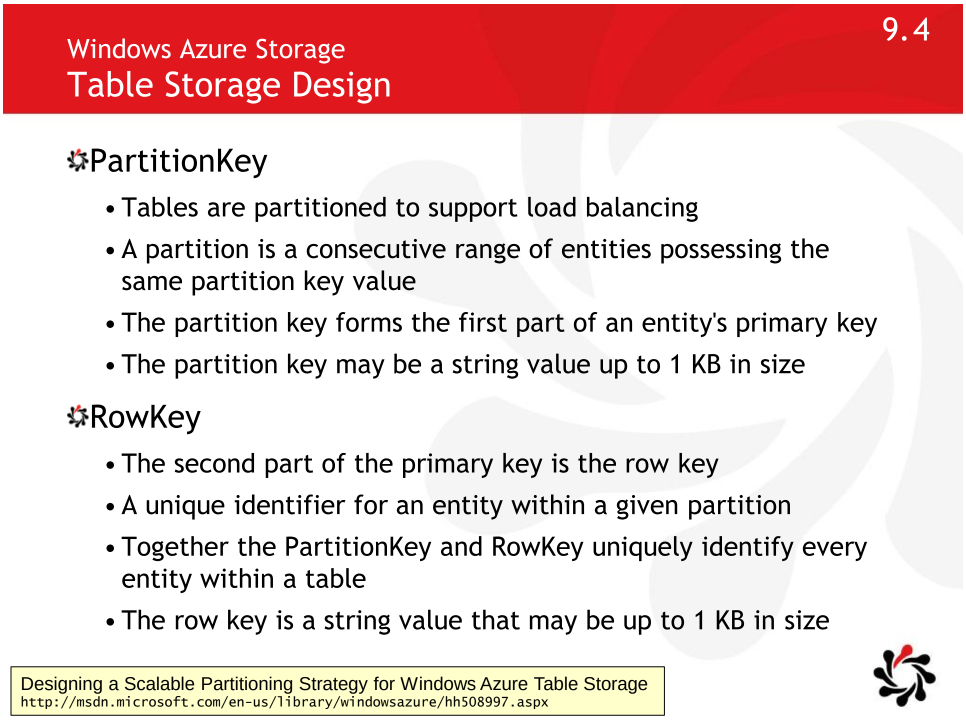
Developing Cloud Applications With Windows Azure Storage
http://www.amazon.co.uk/Developing-Cloud-Applications-Windows-Storage/dp/0735667985/



9.3

Windows Azure Storage
Your Choices

- Windows Azure storage (100TB per account)
  - Table: each structured entry (up to 1MB), table (unlimited)
  - Blob: block (up to 200GB) or page (up to 1TB) e.g. videos
  - Queue: message exchange (up to 64KB)
  - Drive: permanent files in separate VM defined by a VHD
  - Local: temporary files in a role VM (could be lost at any time)
- Why use Windows Azure storage?
  - It is comparatively cheap
  - BUT you have other choices, for example, instead of WA Queue
    - Service Bus queues, MSMQ, Google App Engine queues, etc.
- SQL Database aka “SQL Server in the cloud”
  - Three replicated copies of your database with auto-failover

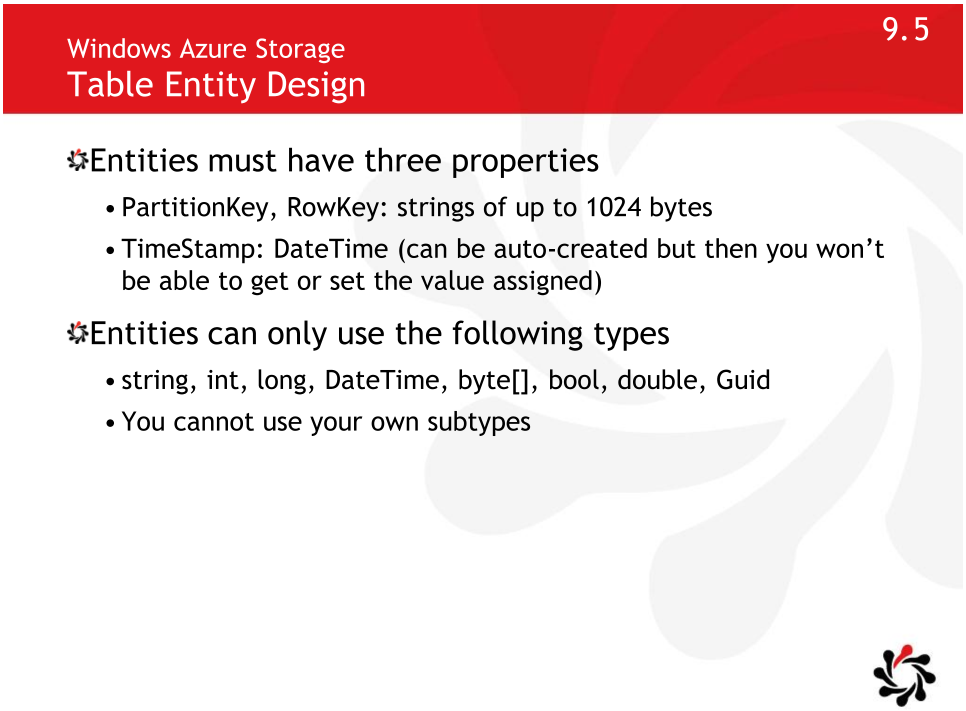


9.4

Windows Azure Storage
Table Storage Design

- PartitionKey
  - Tables are partitioned to support load balancing
  - A partition is a consecutive range of entities possessing the same partition key value
  - The partition key forms the first part of an entity's primary key
  - The partition key may be a string value up to 1 KB in size
- RowKey
  - The second part of the primary key is the row key
  - A unique identifier for an entity within a given partition
  - Together the PartitionKey and RowKey uniquely identify every entity within a table
  - The row key is a string value that may be up to 1 KB in size

Designing a Scalable Partitioning Strategy for Windows Azure Table Storage
http://msdn.microsoft.com/en-us/library/windowsazure/hh508997.aspx

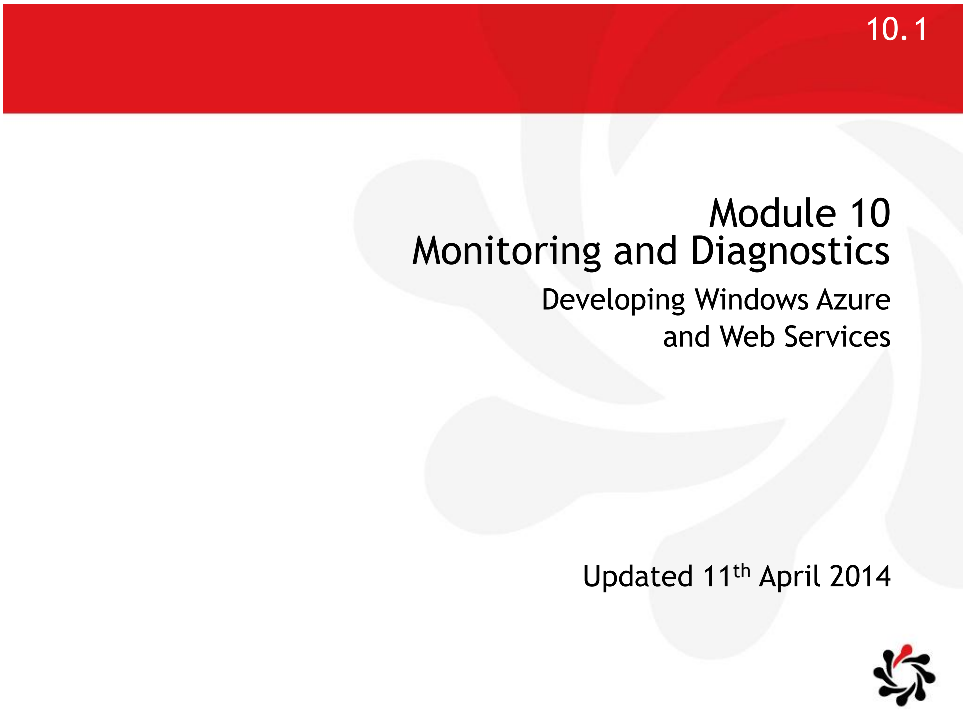


Windows Azure Storage

Table Entity Design

9.5

- Entities must have three properties
  - PartitionKey, RowKey: strings of up to 1024 bytes
  - TimeStamp: DateTime (can be auto-created but then you won't be able to get or set the value assigned)
- Entities can only use the following types
  - string, int, long, DateTime, byte[], bool, double, Guid
  - You cannot use your own subtypes

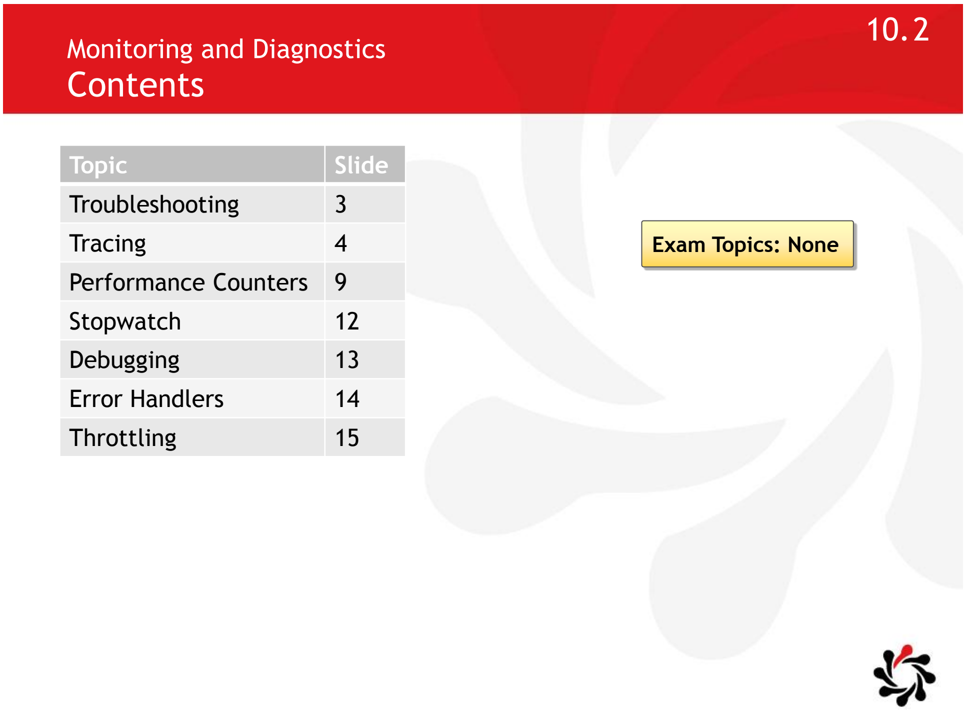


10.1

Module 10
Monitoring and Diagnostics

Developing Windows Azure
and Web Services

Updated 11th April 2014



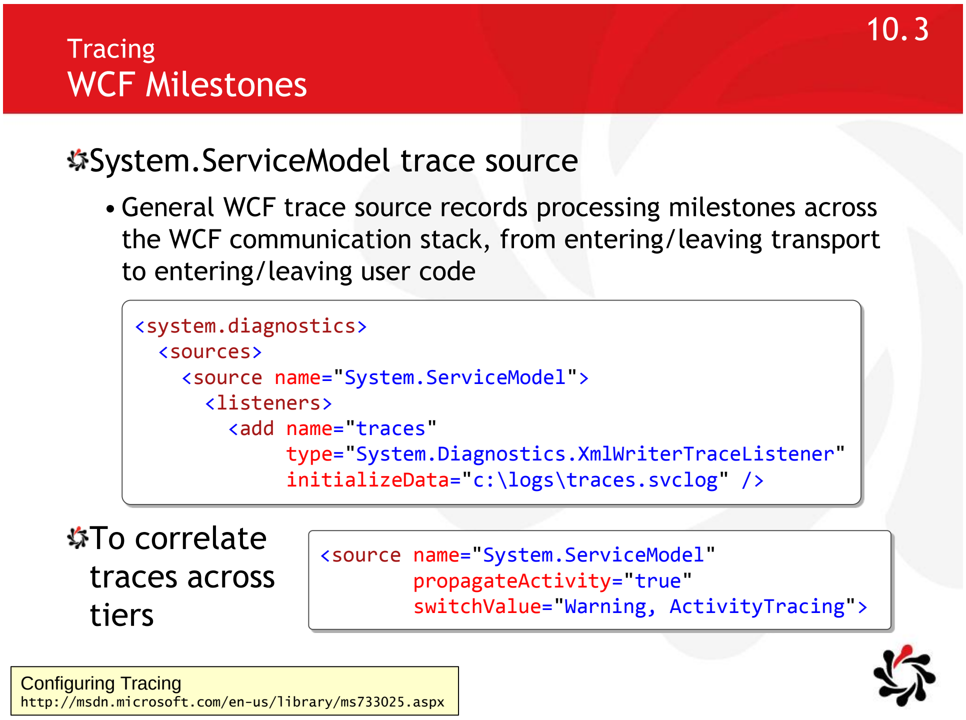
10.2

Monitoring and Diagnostics

Contents

Topic	Slide
Troubleshooting	3
Tracing	4
Performance Counters	9
Stopwatch	12
Debugging	13
Error Handlers	14
Throttling	15

Exam Topics: None



10.3

Tracing

WCF Milestones

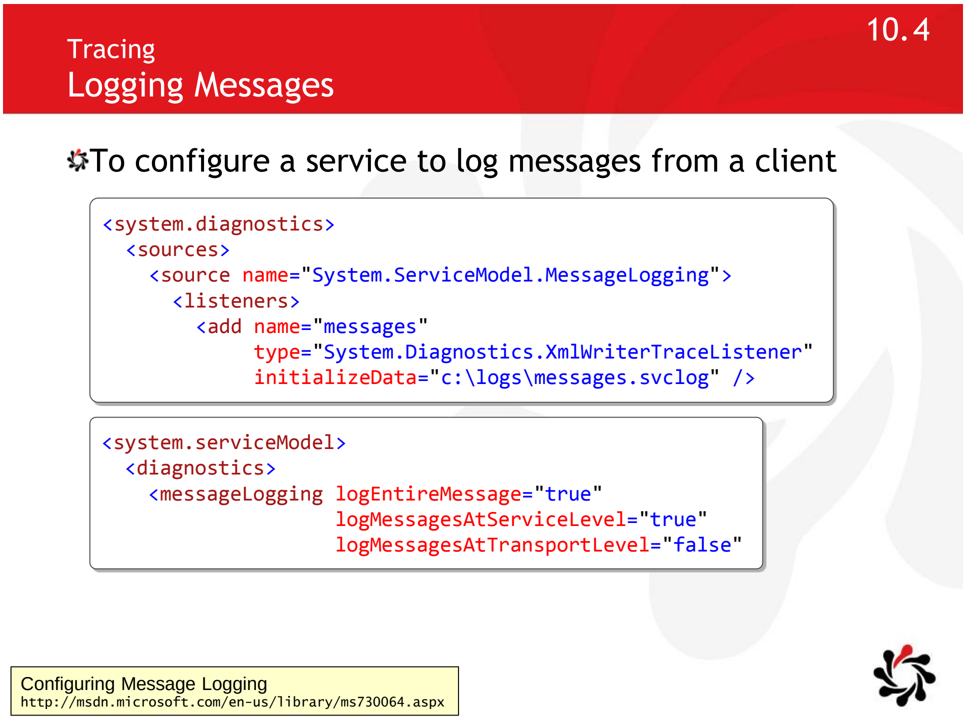
- System.ServiceModel trace source
  - General WCF trace source records processing milestones across the WCF communication stack, from entering/leaving transport to entering/leaving user code

```
<system.diagnostics>
  <sources>
    <source name="System.ServiceModel">
      <listeners>
        <add name="traces"
             type="System.Diagnostics.XmlWriterTraceListener"
             initializeData="c:\logs\traces.svclog" />
```

- To correlate traces across tiers

```
<source name="System.ServiceModel"
        propagateActivity="true"
        switchValue="Warning, ActivityTracing">
```

Configuring Tracing
http://msdn.microsoft.com/en-us/library/ms733025.aspx



10.4

Tracing

Logging Messages

- To configure a service to log messages from a client

```
<system.diagnostics>
  <sources>
    <source name="System.ServiceModel.MessageLogging">
      <listeners>
        <add name="messages"
             type="System.Diagnostics.XmlWriterTraceListener"
             initializeData="c:\logs\messages.svclog" />
```

```
<system.serviceModel>
  <diagnostics>
    <messageLogging logEntireMessage="true"
                    logMessagesAtServiceLevel="true"
                    logMessagesAtTransportLevel="false"
```

Configuring Message Logging
http://msdn.microsoft.com/en-us/library/ms730064.aspx

10.5

Tracing

Messaging Logging Options

- logEntireMessage (default is false)
  - Specifies if the entire message (header and body) is logged
  - Affects service, transport, and malformed logging levels
- logMessagesAtServiceLevel (default is false)
  - Specifies whether messages are traced at the service level (before encryption- and transport-related transforms)
- logMessagesAtTransportLevel (default is false)
  - Specifies whether messages are traced at the transport level
  - Any filters specified in the config file are applied, and only messages that match the filters are traced
  - The filters element holds a collection of XPath filters

<messageLogging>
http://msdn.microsoft.com/en-us/library/ms731308.aspx

10.6

Tracing

Logging Security Information with Messages

- To include security tokens aka personally identifiable information (PII) in logged messages
  - Machine.config

```
<machineSettings enableLoggingKnownPii="true" ...
```

  - App.config or Web.config

```
<system.serviceModel>
  <diagnostics>
    <messageLogging logKnownPii="true" ...
```

10.7

Performance Counters
How to Enable WCF Performance Counters

- To enable all performance counters exposed by the ServiceModelService counter group

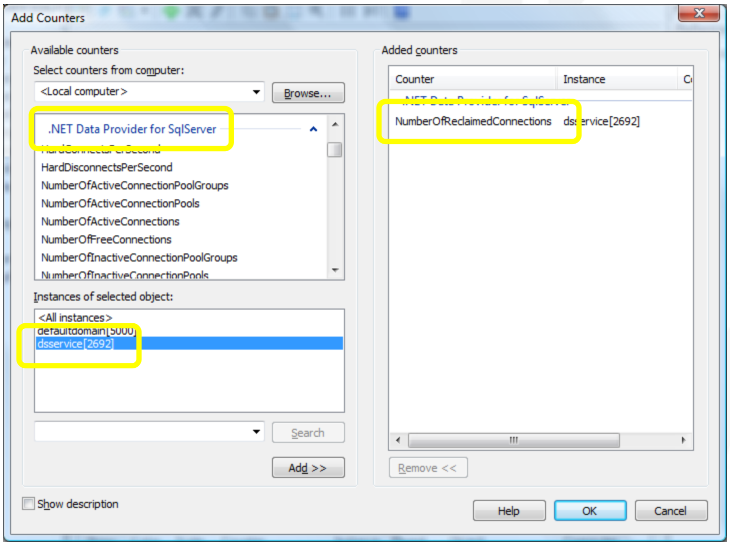
```
<diagnostics performanceCounters="ServiceOnly" ...
```

- Options
  - **All**: ServiceModelService, ServiceModelEndpoint and ServiceModelOperation
  - **ServiceOnly**: ServiceModelService
  - **None**
- To read the counters
  - ServiceModelX 4.0.0.0\CounterName e.g. Calls, Instances
  - Where X is either Service, Endpoint, or Operation

WCF Performance Counters
http://msdn.microsoft.com/en-us/library/ms735098.aspx

10.8

Performance Counters
Performance Counters in PerfMon.exe



Add Counters

Available counters

Select counters from computer: <Local computer>

.NET Data Provider for SqlServer

HardConnectsPerSecond
HardDisconnectsPerSecond
NumberOfActiveConnectionPoolGroups
NumberOfActiveConnectionPools
NumberOfActiveConnections
NumberOfFreeConnections
NumberOfInactiveConnectionPoolGroups
NumberOfInactiveConnectionPools

Instances of selected object:

<All instances>
defaultdomain[5000]
dsservice[2692]

Added counters

Counter	Instance
.NET Data Provider for SqlServer	
NumberOfReclaimedConnections	dsservice[2692]

□ Show description

Performance Counters

Performance Counters in Code

- To programmatically read a performance counter

```
using System.Diagnostics;
using System.Reflection;
```

```
string category = ".NET Data Provider for SqlServer";
string counter = "NumberOfReclaimedConnections";
string instance = string.Format("{0}[{1}]",
    Assembly.GetEntryAssembly().GetName().Name,
    Process.GetCurrentProcess().Id);
var pc = new PerformanceCounter(category, counter, instance);
```

Stopwatch

Using the Stopwatch class

- Two ways to create and use

```
var s = Stopwatch.StartNew();

// do some work

s.Stop();
TimeSpan ts = s.Elapsed;
long ms = s.ElapsedMilliseconds;
```

```
var s = new Stopwatch();
s.Start();
```

- Stop method
  - Stop and retain cumulative elapsed time
- Reset method
  - Stop and reset the elapsed time to zero
- Restart method is equivalent to: Reset then Start

10.11

Error Handlers

Processing Messages

- The body of a Message instance can only be accessed or written once
- If you have an operation that accepts the Message type and you need to process it multiple times you must
  - Call the CreateBufferedCopy method to load into memory
  - Call the CreateMessage method of the MessageBuffer

MessageBuffer Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.messagebuffer.aspx

10.12

Throttling

MaxReceivedMessageSize

- Gets and sets the maximum allowable message size, measured in bytes, that can be received
  - Default is 65536 bytes (64 KB)
  - For example, you might want to increase the maximum size of a message that a service can receive to 256 KB

```
<wsHttpBinding>
  <binding maxReceivedMessageSize="262144"
```

```
var binding = new WSHttpBinding();
binding.MaxReceivedMessageSize = 256 * 1024;
```

TransportBindingElement.MaxReceivedMessageSize - http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.transportbindingelement.maxreceivedmessagesize.aspx

12.1

Module 12
Scaling Services

Developing Windows Azure
and Web Services

Updated 11th April 2014

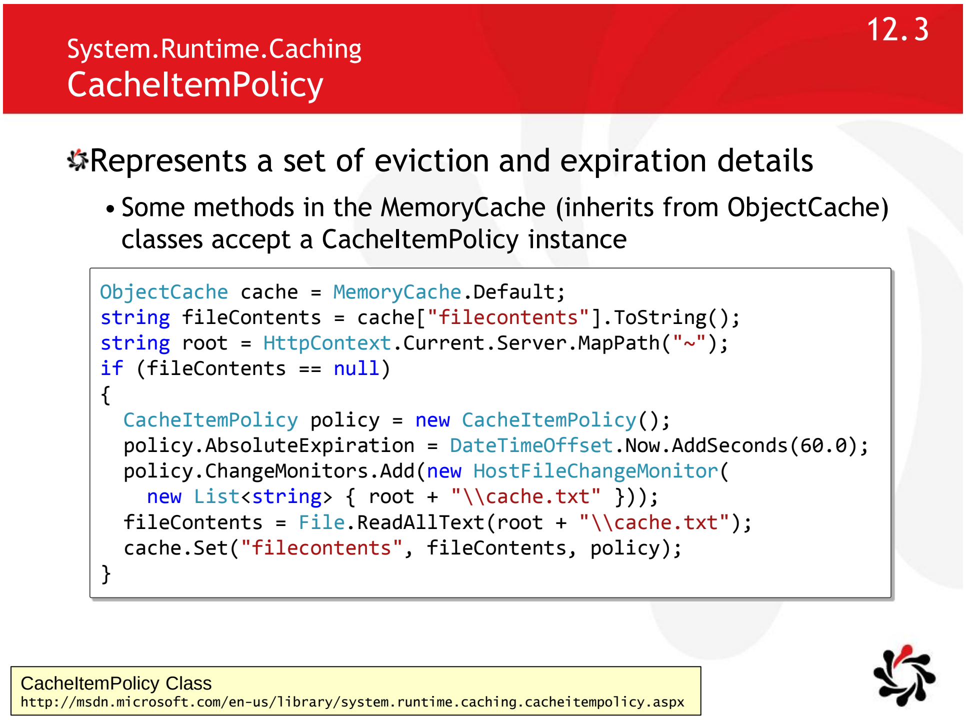
12.2

Scaling Services

Contents

Topic	Slide
System.Runtime.Caching	3
Real-Time Synchronization	4
Transient Fault Handling	7

Exam Topics: Implement caching
❑ Cache static data, apply cache policy (including expirations)
❑ Use CacheDependency to refresh cache data
❑ Query notifications



12.3

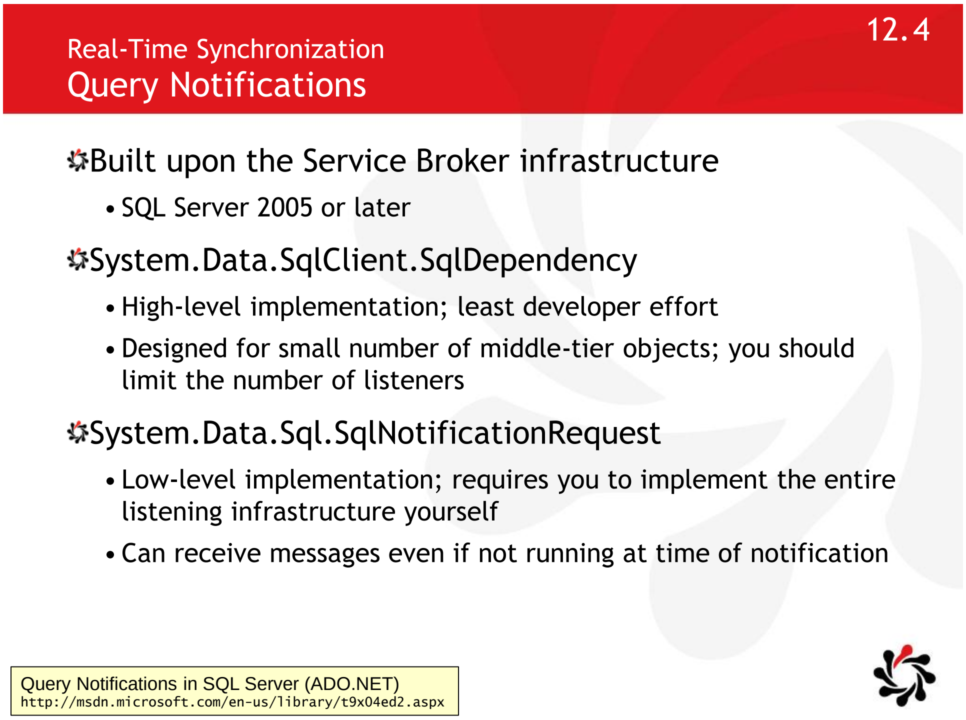
System.Runtime.Caching

CacheItemPolicy

- Represents a set of eviction and expiration details
  - Some methods in the MemoryCache (inherits from ObjectCache) classes accept a CacheItemPolicy instance

```
ObjectCache cache = MemoryCache.Default;
string fileContents = cache["filecontents"].ToString();
string root = HttpContext.Current.Server.MapPath("~");
if (fileContents == null)
{
  CacheItemPolicy policy = new CacheItemPolicy();
  policy.AbsoluteExpiration = DateTimeOffset.Now.AddSeconds(60.0);
  policy.ChangeMonitors.Add(new HostFileChangeMonitor(
    new List<string> { root + "\\cache.txt" }));
  fileContents = File.ReadAllText(root + "\\cache.txt");
  cache.Set("filecontents", fileContents, policy);
}
```

CacheItemPolicy Class
http://msdn.microsoft.com/en-us/library/system.runtime.caching.cacheitempolicy.aspx



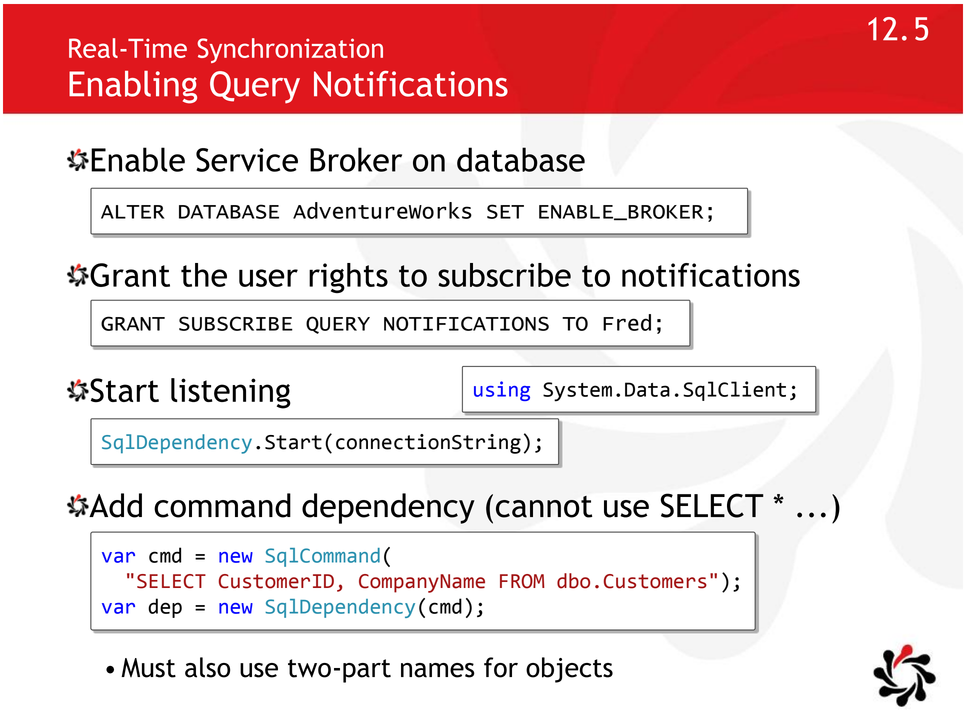
12.4

Real-Time Synchronization

Query Notifications

- Built upon the Service Broker infrastructure
  - SQL Server 2005 or later
- System.Data.SqlClient.SqlDependency
  - High-level implementation; least developer effort
  - Designed for small number of middle-tier objects; you should limit the number of listeners
- System.Data.Sql.SqlNotificationRequest
  - Low-level implementation; requires you to implement the entire listening infrastructure yourself
  - Can receive messages even if not running at time of notification

Query Notifications in SQL Server (ADO.NET)
http://msdn.microsoft.com/en-us/library/t9x04ed2.aspx



12.5

Real-Time Synchronization

Enabling Query Notifications

- Enable Service Broker on database

```
ALTER DATABASE AdventureWorks SET ENABLE_BROKER;
```

- Grant the user rights to subscribe to notifications

```
GRANT SUBSCRIBE QUERY NOTIFICATIONS TO Fred;
```

- Start listening

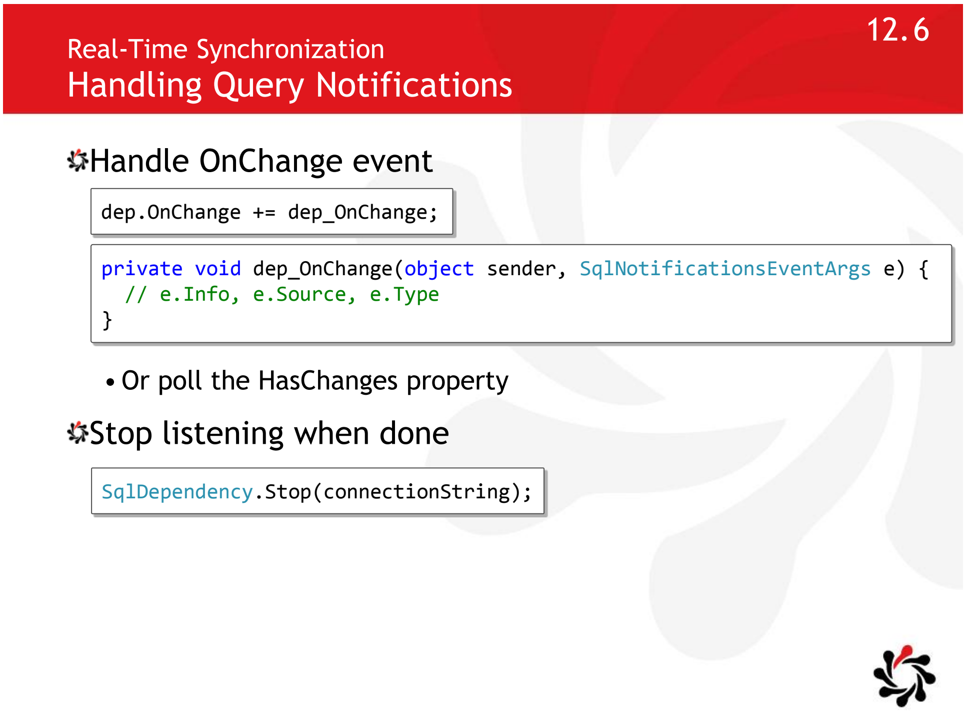
```
using System.Data.SqlClient;
```

```
SqlDependency.Start(connectionString);
```

- Add command dependency (cannot use SELECT * …)

```
var cmd = new SqlCommand(
  "SELECT CustomerID, CompanyName FROM dbo.Customers");
var dep = new SqlDependency(cmd);
```

  - Must also use two-part names for objects



12.6

Real-Time Synchronization

Handling Query Notifications

- Handle OnChange event

```
dep.OnChange += dep_OnChange;
```

```
private void dep_OnChange(object sender, SqlNotificationEventArgs e) {
  // e.Info, e.Source, e.Type
}
```

  - Or poll the HasChanges property

- Stop listening when done

```
SqlDependency.Stop(connectionString);
```

12.7

Transient Fault Handling

What Is It?

- When you’re designing a real world cloud app, one of the things you have to think about is how to handle temporary service interruptions
  - You can frequently get little glitches that are typically self-healing, and if you aren’t prepared to handle them intelligently, they’ll result in a bad experience for your customers
- Use smart retry/back-off logic to mitigate the effect of transient failures
  - Instead of throwing an exception and displaying a not available or error page to your customer, you can recognize errors that are typically transient, and automatically retry the operation that resulted in the error, in hopes that before long you’ll be successful

Transient Fault Handling
http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/transient-fault-handling

12.8

Transient Fault Handling

Implementing Smart Retry Logic

- Several ways you can implement smart retry logic
  - Microsoft Patterns & Practices group has a **Transient Fault Handling Application Block** that does everything for you if you’re using ADO.NET for SQL Database access (not through Entity Framework)

```
var policy = RetryPolicy.Create<SqlAzureTransientErrorDetectionStrategy>(
    retryCount: 3, retryInterval: TimeSpan.FromSeconds(5));
using (var conn = new ReliableSqlConnection(connStr, policy))
```

  - Entity Framework 6 builds this kind of retry logic right into the framework

```
public class EFConfiguration : DbConfiguration {
    public EFConfiguration() {
        AddExecutionStrategy(() => new SqlAzureExecutionStrategy());
    }
```

Transient Fault Handling Application Block
http://msdn.microsoft.com/en-us/library/dn440719(v=pandp.60).aspx

A.1

Appendix A
Designing and Extending WCF Services

Developing Windows Azure
and Web Services

Updated 11th April 2014

A.2

Designing and Extending WCF Services

Contents

Topic	Slide
MEP Contracts	3
Asynchronous Operations	4
Extending WCF	7
Transactions	8
Discovery	13
Routing	17

Exam Topics: Create a WCF service
❑ Implement message inspectors
❑ Implement asynchronous operations in the service

Exam Topics: Implement messaging patterns
❑ Implement one way, request/reply, streaming, and duplex communication

Exam Topics: Configure WCF services by using configuration settings or the API
❑ WCF routing and discovery features

Exam Topics: Host and manage services
❑ Implement transactional services

A.3

MEP Contracts
One-Way Considerations

- With a one-way operation the receiver does not send a reply message there is no response message to carry fault information back to the client
  - You may be able to detect error conditions by using features of the underlying binding, such as reliable sessions
- Most one-way operations return as soon as the outbound data is written to the network connection
  - Operation call can block if ConcurrencyMode is Single and the binding uses sessions
  - If the transport cannot find the endpoint a EndpointNotFoundException is thrown
  - If the SendTimeout period on the client transport binding is exceeded a TimeoutException is thrown

One-Way Services
`http://msdn.microsoft.com/en-us/library/vstudio/ms730149(v=vs.110).aspx`

A.4

Asynchronous Operations
Asynchronous Clients (WCF 3.0)

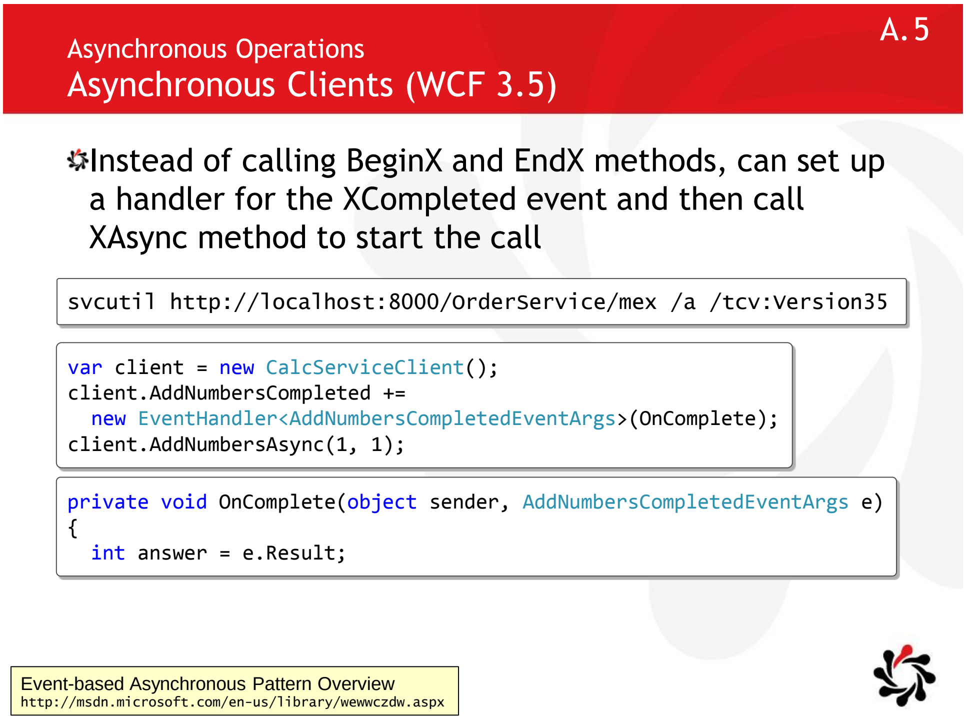
- To call a service asynchronously without affecting any other clients or changing the service, the proxy generated can implement the .NET APM

```
var client = new CalcServiceClient();
client.BeginAddNumbers(1, 1, OnComplete, client);
```

```
private void OnComplete(IAsyncResult result)
{
  var client = result.AsyncState as CalcServiceClient;
  int answer = client.EndAddNumbers(result);
```

- Use svcutil.exe with /a or /async switch to generate asynchronous methods like BeginX and EndX

ServiceModel Metadata Utility Tool (Svcutil.exe)
`http://msdn.microsoft.com/en-us/library/aa347733.aspx`



A.5

Asynchronous Operations

Asynchronous Clients (WCF 3.5)

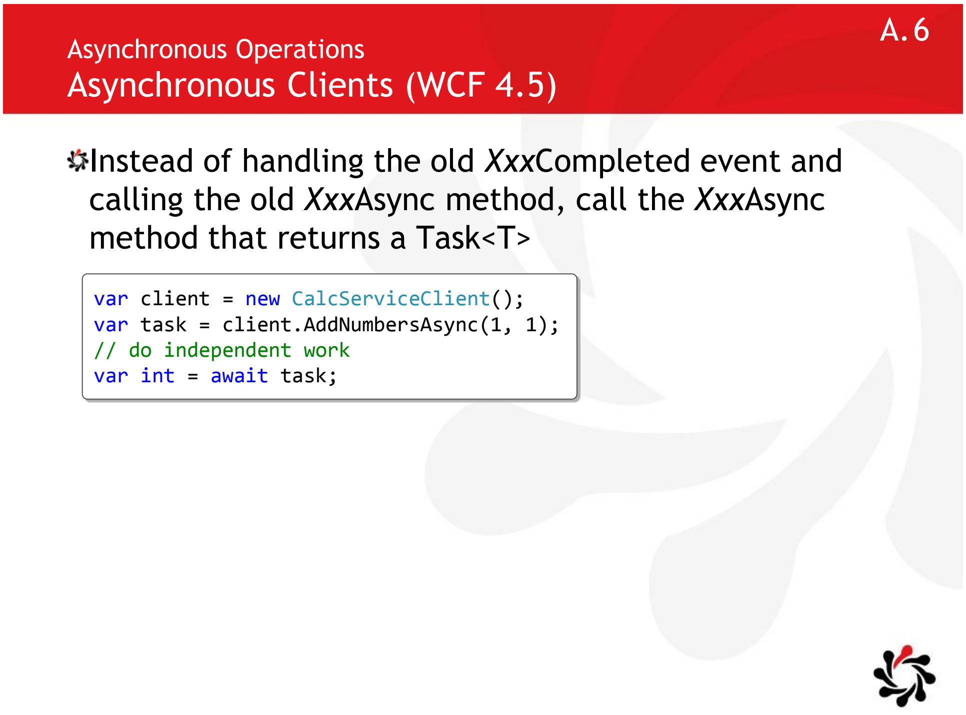
- Instead of calling BeginX and EndX methods, can set up a handler for the XCompleted event and then call XAsync method to start the call

```
svcutil http://localhost:8000/OrderService/mex /a /tcv:Version35
```

```
var client = new CalcServiceClient();
client.AddNumbersCompleted +=
  new EventHandler<AddNumbersCompletedEventArgs>(OnComplete);
client.AddNumbersAsync(1, 1);
```

```
private void OnComplete(object sender, AddNumbersCompletedEventArgs e)
{
  int answer = e.Result;
```

Event-based Asynchronous Pattern Overview
http://msdn.microsoft.com/en-us/library/wewwczdw.aspx



A.6

Asynchronous Operations

Asynchronous Clients (WCF 4.5)

- Instead of handling the old *Xxx*Completed event and calling the old *Xxx*Async method, call the *Xxx*Async method that returns a Task<T>

```
var client = new CalcServiceClient();
var task = client.AddNumbersAsync(1, 1);
// do independent work
var int = await task;
```

A.7

Extending WCF

- “Dispatch” = server-side
  - Implement these interfaces: IDispatchMessageFormatter, IDispatchMessageInspector, IDispatchOperationSelector
  - Attach to these objects:
    - DispatchOperation
      - ParameterInspectors, Formatter
    - DispatchRuntime
      - MessageInspectors

- “Client” = client-side
  - Implement these interfaces: IClientMessageFormatter, IClientMessageInspector, IClientOperationSelector
  - Attach to these objects:
    - ClientOperation
      - ParameterInspectors, Formatter
    - ClientRuntime
      - MessageInspectors

Extending WCF’s Client and Dispatcher Runtimes
`http://msdn.microsoft.com/en-us/library/ff183867.aspx`

A.8

Transactions

Enabling Transactions

- For an operation to automatically participate in an existing transaction or to create a new transaction if one does not exist (default is false)

```
[OperationBehavior(TransactionScopeRequired=true)]
public bool ValidateCredit(string cardNumber)
```

- For a service to support transactions it must also require sessions

```
[ServiceContract(SessionMode=SessionMode.Required)]
public interface IDataUpdate
```

A.9

Transactions

Service Behaviors for Transactions

- To automatically destroy the service instance when the transaction completes (default is false)

```
[ServiceBehavior(ReleaseServiceInstanceOnTransactionComplete=true)]
public class Calculator
```

  - You must set at least one operation to require a transaction
  - You must NOT set concurrency mode to multiple

- To automatically complete the transaction when the session ends i.e. client closes proxy (default is false)

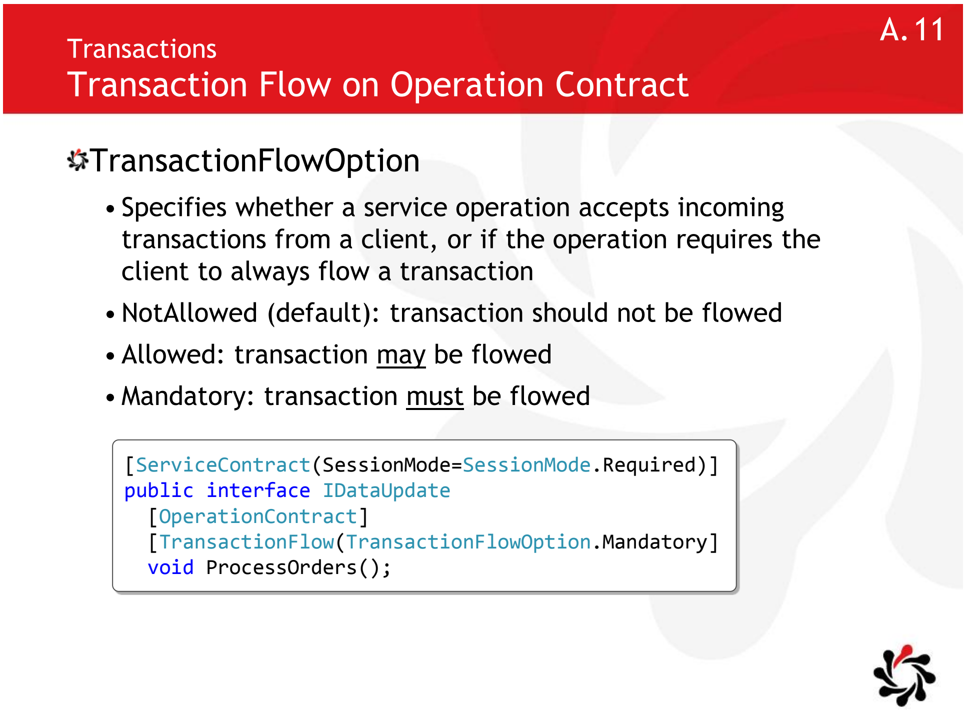
```
[ServiceBehavior(TransactionAutoCompleteOnSessionClose=true)]
public class Calculator
```

A.10

Transactions

Operation Behaviors for Transactions

- TransactionScopeRequired (default is false)
  - Indicates whether the method requires a transaction scope for its execution
  - Must be true on at least one operation if ReleaseServiceInstanceOnTransactionComplete is true
- TransactionAutoComplete (default is true)
  - Indicates whether to automatically complete the current transaction scope if no unhandled exceptions occur
  - Set to false to complete or abort transactions directly in the code for the operation
  - Handled exceptions can be thrown in the course of the operation without automatically aborting the transaction
  - Unhandled exceptions trigger an automatic abort



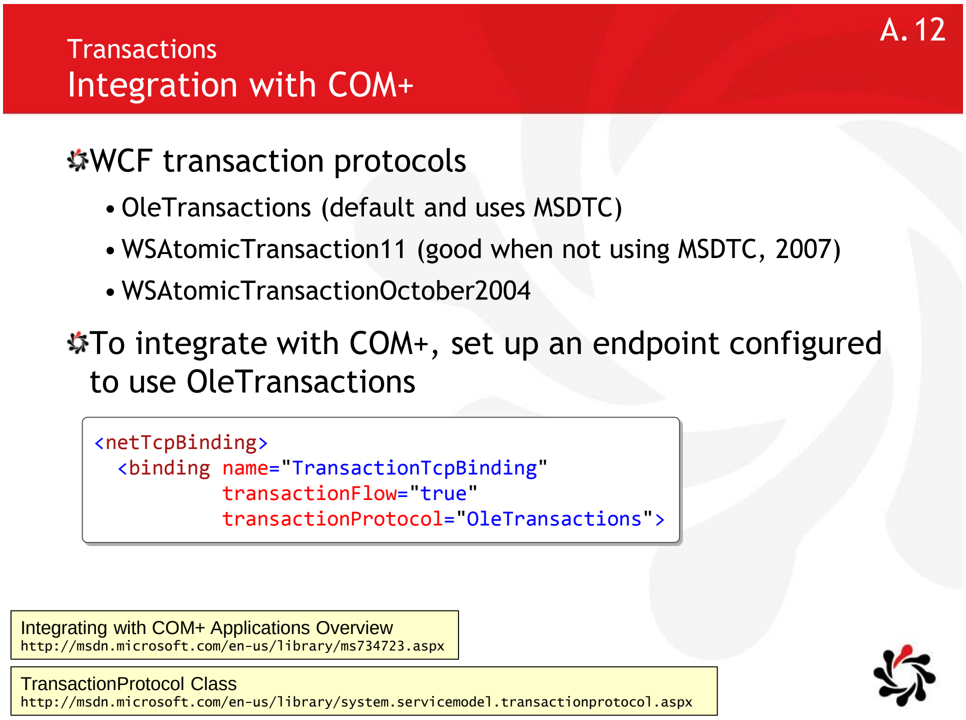
A. 11

Transactions
Transaction Flow on Operation Contract

TransactionFlowOption

- Specifies whether a service operation accepts incoming transactions from a client, or if the operation requires the client to always flow a transaction
- NotAllowed (default): transaction should not be flowed
- Allowed: transaction may be flowed
- Mandatory: transaction must be flowed

```
[ServiceContract(SessionMode=SessionMode.Required)]
public interface IDataUpdate
  [OperationContract]
  [TransactionFlow(TransactionFlowOption.Mandatory]
  void ProcessOrders();
```



A. 12

Transactions
Integration with COM+

WCF transaction protocols

- OleTransactions (default and uses MSDTC)
- WSAtomicTransaction11 (good when not using MSDTC, 2007)
- WSAtomicTransactionOctober2004

To integrate with COM+, set up an endpoint configured to use OleTransactions

```
<netTcpBinding>
  <binding name="TransactionTcpBinding"
           transactionFlow="true"
           transactionProtocol="OleTransactions">
```

Integrating with COM+ Applications Overview
http://msdn.microsoft.com/en-us/library/ms734723.aspx

TransactionProtocol Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.transactionprotocol.aspx

A. 13

Discovery
Announcements

AnnouncementService

- Used by clients to listen for and act on incoming messages on a standard announcement endpoint (AnnouncementEndpoint)
- Provides event notification when Hello or Bye announcement messages arrive

AnnouncementClient

- Used by services to send discovery announcement messages
- An announcement message contains information about the service such as its fully-qualified contract name, any scopes that the service is operating in as well as any custom metadata the service wants to send
- You do not need an AnnouncementClient if you want to service to make announcements automatically when the host opens and closes

A. 14

Discovery
Configuring Announcements and Discovery

You must enable discovery behavior for the service…

- …and discovery and announcement endpoints with code…

```
var disc = new ServiceDiscoveryBehavior();
disc.AnnouncementEndpoints.Add(new UdpAnnouncementEndpoint());
host.Description.Behaviors.Add(disc);
host.AddServiceEndpoint(new UdpDiscoveryEndpoint());
```

…or with configuration

```
<services>
  <service name="MyService">
    <endpoint kind="udpDiscoveryEndpoint" />
```

```
<behaviors>
  <serviceBehaviors>
    <behavior>
      <serviceDiscovery>
        <announcementEndpoints>
          <endpoint kind="udpAnnouncementEndpoint" />
```

A.15

Discovery

Find and FindAsync with FindCriteria

- Both methods pass in a FindCriteria
  - Has several properties, which can be grouped into search criteria, which specify what services you are looking for, and find termination criteria (how long the search should last)
- Search criteria include:
  - ContractTypeNames: if more than one contract name is specified, only service endpoints matching ALL contracts reply
  - Scopes: absolute URIs that are used to categorize individual service endpoints
  - ScopeMatchBy: ScopeMatchByExact, ScopeMatchByPrefix, ScopeMatchByLdap, and so on
- Termination criteria include Duration and MaxResults

Discovery Find and FindCriteria
http://msdn.microsoft.com/en-us/library/ee816866.aspx

A.16

Discovery

FindResponse

- FindResponse
  - Represents the response from a find request
- FindResponse.Endpoints Property
  - Gets a collection of EndpointDiscoveryMetadata for the discoverable services that matched the find request
- EndpointDiscoveryMetadata
  - Address: Gets or sets the endpoint address
  - ContractTypeNames: Gets a collection of contract type names implemented by the service
  - ListenUris: Gets the listen URIs for the service
  - Extensions, Scopes, and Version: other properties

EndpointDiscoveryMetadata Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.discovery.endpointdiscoverymetadata

A. 17

Routing

Routing Service Client Contracts

```
<client> <!-- router configuration -->
  <endpoint name="Calc"
            address="http://localhost:8080/CalcService"
            binding="basicHttpBinding"
            contract="*" />
```

- When configuring the router it acts as a client to many potential services so the contract element of an endpoint can use a wildcard

Mapping MEPs to routing interfaces

Message Exchange Pattern (MEP)	Interface
Request-Response	IRequestReplyRouter
One-Way	ISimplexDatagramRouter
One-Way (with sessions)	ISimplexSessionRouter
Duplex	IDuplexSessionRouter

Routing Service and Content Based Routing
http://www.prideparrot.com/blog/archive/2011/8/routing_service_and_content_based_routing

A. 18

Routing

Combining Filters and Setting Priorities

You may have multiple services that need to handle messages filtered by message content, for example

- Service1 must handle orders worth up to £100
- Service2 must handle orders worth more than £100

```
<filter name="LessThan100" filterType="XPath"
        filterData="//fb:OrderValue &lt; 100" />
<filter name="EverythingElse" filterType="MatchAll" />
```

```
<filterTable name="EventRoutingTable">
  <add filterName="LessThan100"
       endpointName="Service1" priority="2" />
  <add filterName="EverythingElse"
       endpointName="Service2" priority="1" />
```

http://seroter.wordpress.com/2011/01/13/wcf-routing-service-deep-dive-part-iiusing-filters/

Message Filters
http://msdn.microsoft.com/en-us/library/ee517424.aspx

Appendix B
Implementing Security in WCF Services

Developing Windows Azure and Web Services

Updated 11th April 2014

WCF Security
Contents

Topic	Slide
Certificates	3
Comparison	7
Authentication	9
Authorization	11
Impersonation	14
Auditing	16
Protection	17
WCF 4.5 Bindings	20

Microsoft® patterns & practices
proven practices for predictable results

Improving Web Services Security:
Scenarios and Implementation Guidance for WCF
`http://msdn.microsoft.com/en-us/library/ff650794.aspx`

Exam Topics: Secure a WCF service
- ❑ Implement message level security, transport level security
- ❑ Certificates

B.3

Certificates

What is X.509?

- In cryptography, X.509 is a standard for a public key infrastructure (PKI)
- X.509 specifies, amongst other things:
  - Formats for public key certificates
  - Certificate revocation lists
  - Certification path validation algorithm
- Assumes a strict hierarchical system of certificate authorities (CAs) for issuing the certificates
  - The “chain of trust”
  - If a certificate is revoked, all under it are revoked too

Working with Certificates
`http://msdn.microsoft.com/en-us/library/ms731899.aspx`

B.4

Certificates

What is a Certificate Authority (CA)?

- A certificate authority or certification authority (CA) is an entity that issues digital certificates
  - The digital certificate certifies the ownership of a public key by the named subject of the certificate
  - This allows others (relying parties) to rely upon signatures or assertions made by the private key that corresponds to the public key that is certified
- An organization’s trusted root certificates can be distributed to all employees so that they can use the company PKI system

B.5

Certificates
Sample

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
...
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
...
```

B.6

Certificates
What are X.509 Certificates Used For?

- Multiple reasons: Authenticate, Encrypt, Sign
  - X509CertificateInitiatorClientCredential.SetCertificate(): Used by a *client* to *identify itself*
  - X509CertificateInitiatorServiceCredential.SetCertificate(): Used by a *service* to *validate a client certificate*
  - X509CertificateRecipientServiceCredential.SetCertificate(): Used by a *service* to *identify itself*
- You can also set certificates by using configuration
  - If you are creating a service, credentials (including certificates) are specified under the *serviceBehaviors* section
  - When you are programming a client, credentials (including certificates) are specified under the *endpointBehaviors* section

X509CertificateRecipientServiceCredential Methods
http://msdn.microsoft.com/en-us/library/ms577349.aspx

B.7

Comparison
Security Modes

Security Mode	Encryption	Authentication
None	None	None
Transport	SSL (point-to-point)	Basic, Digest, Certificate, Windows, NTLM
TransportWith MessageCredential	SSL (point-to-point)	In SOAP Header (Certificate, UserName, IssuedToken, Windows*, Custom)
Message	Only if ProtectionLevel is set in service, operation, fault, or message contracts (end-to-end)	In SOAP Header (Certificate, UserName, IssuedToken, Windows*, Custom)
Transport CredentialOnly	None	Basic, Digest, Certificate, Windows*, NTLM

```
<basicHttpBinding>
  <binding name="kermit">
    <security mode="TransportCredentialOnly">
      <transport clientCredentialType="Windows" />
```

*Kerberos

B.8

Comparison
Security Types

	Transport (e.g. HTTPS)	Message (SOAP)
Speed	Faster (and can be hardware accelerated)	Slower
Extensible	No	Yes
Authentication Choices	Basic, Digest, NTLM, Windows, Certificate	Windows, UserName, Certificate, IssuedToken, Custom
Encryption	Point-to-point	End-to-end

Warning!
You cannot use Message security with a REST endpoint because there won’t be a SOAP message to use

WCF Security Guidelines
http://msdn.microsoft.com/en-us/library/ff648370.aspx

B.9

Authentication

Validating Identity with DNS

- A service’s endpoint identity is a value propagated to any client and used to authenticate the service
  - At design time, the client developer determines the service’s identity from the endpoint's metadata (exposed through WSDL)
  - At runtime, the client application checks the claims of the service’s security credentials before sending any messages to the service
- Identity types
  - Domain Name System (DNS), Certificate or Certificate Reference, RSA, User principal name, Service principal name
  - A DNS check enables you to use certificates reissued with a new RSA key but the same DNS or subject name, so the identity check is still valid

Service Identity and Authentication
http://msdn.microsoft.com/en-us/library/ms733130.aspx

B.10

Authentication

Caching the Security Context Token

- Support for security context tokens (SCT)
  - For clients that support SCT, key exchange and validation can be done once and cached for the session
  - For clients that do not, key exchange and validation must be done per call
- In environments that require maximum security you can force clients to authenticate on every call by disabling SCT support

```
<bindings>
  <wsHttpBinding>
    <binding ... >
      <security ... >
        <message establishSecurityContext="false" ...
```

http://stackoverflow.com/questions/1683724/what-are-the-impacts-of-setting-establishsecuritycontext-false-if-i-use-https

B.11

Authorization
Restricting Operations Calls By Role

- Use PrincipalPermission to ensure only members of a Windows Group or ASP.NET role can call an operation

```
using System.Security.Permissions;
```

```
[PrincipalPermission(SecurityAction.Demand, Role="Sales")]
public int AddNumbers(int a, int b)
```

  - Name="Alice,Bob", Role="Admin,Sales", Authenticated="true"
  - Note: by default, the role is a Windows group name
- SecurityAction
  - Demand: throw exception if security context does NOT match
  - Deny: throw exception if security context DOES match

PrincipalPermissionAttribute Class
http://msdn.microsoft.com/en-us/library/system.security.permissions.principalpermissionattribute.aspx

B.12

Authorization
Service Authorization

- principalPermissionMode
  - None, <u>UseWindowsGroups</u>, UseAspNetRoles, Custom
- roleProviderName can be any ASP.NET Role Provider
  - e.g. use "SqlProvider" to specify a SQL Server database for roles

```
<behaviors>
  <serviceBehaviors>
    <behavior name="kermit">
      <!-- below are the defaults -->
      <serviceAuthorization
          impersonateCallerForAllOperations="false"
          principalPermissionMode="UseWindowsGroups"
          roleProviderName=""
          serviceAuthorizationManagerType="" />
      <authorizationPolicies>
        <add policyType="" />
```

<serviceAuthorization> element
http://msdn.microsoft.com/en-us/library/ms731306.aspx

B.13

Authorization
Getting the Identity of the Caller

- Self-hosted service needs to retrieve a caller’s identity
  - ServiceSecurityContext represents the security context of a remote party
  - On the client, represents the service identity and, on the service, represents the client identity
- Correct (will always get name of caller)
  - ServiceSecurityContext.Current.PrimaryIdentity.Name
  - OperationContext.Current.
    ServiceSecurityContext.PrimaryIdentity.Name
- Wrong (won’t work if not using impersonation)
  - HttpContext.Current.User.Identity.Name
  - Thread.CurrentPrincipal.Identity.Name

ServiceSecurityContext Class
`http://msdn.microsoft.com/en-us/library/system.servicemodel.servicesecuritycontext.aspx`

B.14

Impersonation
Executing Operations Under Different Identities

- ImpersonateCallerForAllOperations (default is false)
  - Gets or sets a value that indicates whether the service performs impersonation for all the operations that it supports
  - A value of false indicates that impersonation is specified for each operation
- OperationBehavior.Impersonation (default NotAllowed)
  - ImpersonationOption.**Required**: impersonation is performed
  - ImpersonationOption.**Allowed**: impersonation is performed if ImpersonateCallerForAllOperations is true
  - ImpersonationOption.**NotAllowed**: throws exception if ImpersonateCallerForAllOperations is true

Delegation and Impersonation with WCF
`http://msdn.microsoft.com/en-us/library/ms730088.aspx`

B.15

Impersonation
Level

- To access out-of-process resources on behalf of a caller

```
using System.Security.Principal;
```

```
ServiceSecurityContext.Current.WindowsIdentity
  .ImpersonationLevel = TokenImpersonationLevel.Delegation;
```

- TokenImpersonationLevel
  - **None**, **Anonymous**: no impersonation
  - **Identification**: identify but not impersonate
  - **Impersonation**: impersonates the client’s security context on its local system, but not on remote systems
  - **Delegation**: impersonates the client’s security context on remote systems

TokenImpersonationLevel Enumeration
http://msdn.microsoft.com/en-us/library/system.security.principal.tokenimpersonationlevel.aspx

B.16

Auditing
How to Enable Logging

- To audit attempts to access a secure service

```
<serviceSecurityAudit
    auditLogLocation="Default"
    messageAuthenticationAuditLevel="None"
    serviceAuthorizationAuditLevel="None"
    suppressAuditFailure="true" />
```

- Audit levels
  - None (default), Success, Failure, SuccessAndFailure
- Audit log location
  - Default (default depends on OS), Application, Security
- Suppress audit failure
  - If we fail to write to the log, no exception is thrown by default

<serviceSecurityAudit>
http://msdn.microsoft.com/en-us/library/ms731694.aspx

B.17

Protection

Securing Streams

Scenario

- Need to stream sensitive BLOBs over a public network

A possible correct solution

- Use basicHttpBinding with transport security
  - Streaming is only available for: BasicHttpBinding, NetTcpBinding, NetNamedPipeBinding, WebHttpBinding

Wrong solutions include anything that uses SOAP

- Message security: digital signatures for the message body cannot be performed because they require computing a hash over the entire message content; with streaming the content is not available when the headers are constructed
- Reliable sessions: these must buffer the message so it can be re-sent and must hold a copy; cannot be used with streaming

B.18

Protection

Algorithm Suite for Message Security

Algorithm suites are described in the WS-SecurityPolicy specification

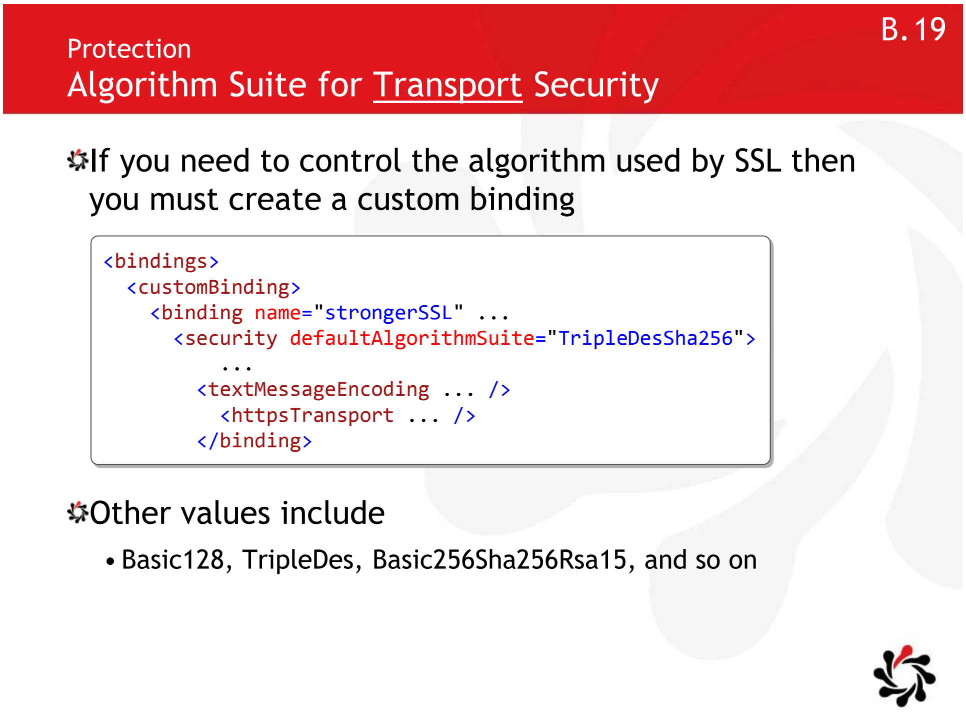
- Can be set in configuration of a binding for message-level security

```
<bindings>
  <wsHttpBinding>
    <binding name="strongerWSBinding">
      <security mode="Message">
        <message algorithmSuite="TripleDes"
                 clientCredentialType="UserName" />
```

Other options

- Basic256 (default), Basic128, Basic128Rsa15, Basic128Sha256, TripleDesSha256Rsa15, and so on

SecurityAlgorithmSuite Class
http://msdn.microsoft.com/en-us/library/system.servicemodel.security.securityalgorithmsuite(v=vs.110).aspx



B.19

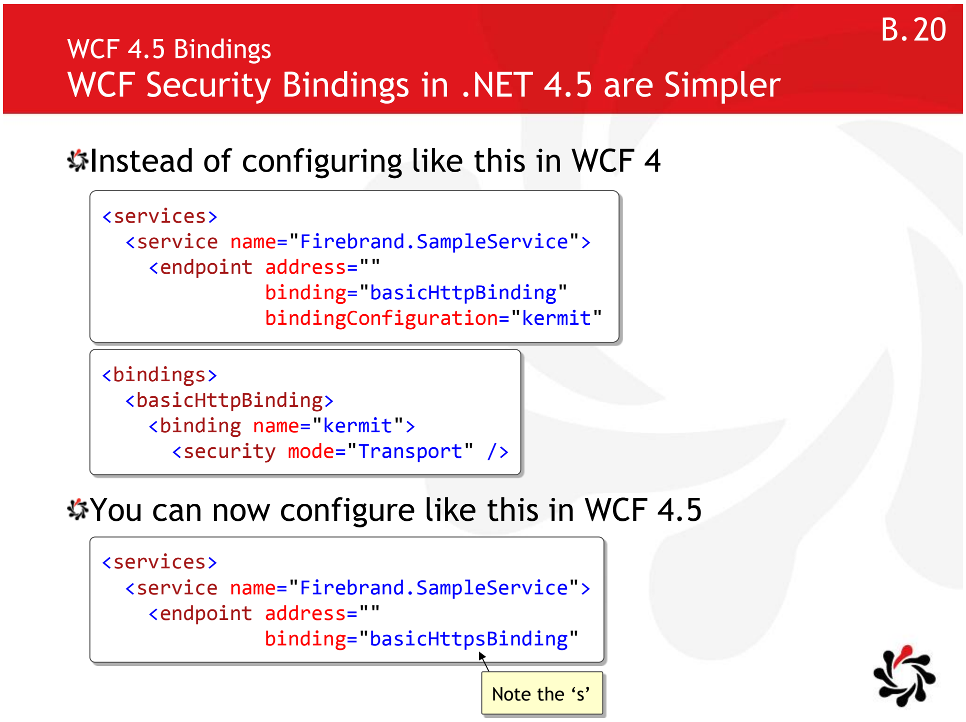
Protection

Algorithm Suite for Transport Security

- If you need to control the algorithm used by SSL then you must create a custom binding

```
<bindings>
  <customBinding>
    <binding name="strongerSSL" ...
      <security defaultAlgorithmSuite="TripleDesSha256">
        ...
        <textMessageEncoding ... />
          <httpsTransport ... />
        </binding>
```

- Other values include
  - Basic128, TripleDes, Basic256Sha256Rsa15, and so on



B.20

WCF 4.5 Bindings

WCF Security Bindings in .NET 4.5 are Simpler

- Instead of configuring like this in WCF 4

```
<services>
  <service name="Firebrand.SampleService">
    <endpoint address=""
              binding="basicHttpBinding"
              bindingConfiguration="kermit"
```

```
<bindings>
  <basicHttpBinding>
    <binding name="kermit">
      <security mode="Transport" />
```

- You can now configure like this in WCF 4.5

```
<services>
  <service name="Firebrand.SampleService">
    <endpoint address=""
              binding="basicHttpsBinding"
```

Note the ‘s’

C.1

Appendix C
“Classic” XML and ADO.NET

Developing Windows Azure
and Web Services

Updated 11th April 2014

C.2

“Classic” XML and ADO.NET

Contents

Topic	Slide
System.Xml	3
DataSets and XML	11
XPath	15
Connections	17
Protecting Data	28
Factories	34
Commands	36
DataReaders	40
DataAdapters	44
DataSets	48
SQL Server	57

Slides 3-27 are most likely to appear in the exam

Exam Topic: Manipulate XML data structures

❑ Read, filter, create, modify XML data structures

❑ Manipulate XML data by using XMLReader, XMLWriter, XMLDocument, XPath, transform XML by using XSLT transformations

Exam Topic: Query and manipulate data by using ADO.NET

❑ Query and manipulate data by using Connection, DataReader, Command, DataAdapter, DataSet

❑ Perform synchronous and asynchronous operations

❑ Manage transactions (API)

C.3

System.Xml (aka “Classic” XML)

XML Processing (.NET 1.0 and later)

For fast, memory-efficient processing of XML

- These use 8KB cache to load each node one at a time
- XmlReader (forward-only, read-only, fast, low-overhead)
- XmlWriter

For DOM-based XML

- These must load the entire file into memory
- XmlDocument: full functionality
- XmlDataDocument: an XmlDocument that can be sync-ed with DataSet
- XPathDocument: read-only but faster XPath queries and transformations

XML Documents and Data
http://msdn.microsoft.com/en-us/library/2bcctyt8.aspx

C.4

System.Xml

Reading XML Efficiently

Properties

- Name, NodeType, HasValue, Value, HasAttributes, AttributeCount, EOF

```
<book>
  <title>C# Rocks!</title>
</book>
```

```
<book>
  <title>C# Rocks!</title>
</book>
```

Methods

- Read, ReadContentAs, ReadElementContentAs, ReadInnerXml, ReadOuterXml, ReadToDescendant, ReadToNextSibling, etc.
- MoveToAttribute, MoveToContent, MoveToElement, Skip
- MoveToFirstAttribute, MoveToNextAttribute, GetAttribute
- IsStartElement

```
var myReader = XmlReader.Create("SampleXml.xml");
while (myReader.Read()) {
  if (myReader.NodeType == XmlNodeType.Element)
    Console.WriteLine(myReader.Name);
  else if (myReader.NodeType == XmlNodeType.Text)
    Console.WriteLine(myReader.Value);
```

C.5

System.Xml

Validating XML with XmlReader

- Use XmlReaderSettings class with an XmlReader to perform validation

```
var xrs = new XmlReaderSettings();
xrs.ValidationType = ValidationType.Schema;
xrs.Schemas.Add(null, "books.xsd");
xrs.ValidationEventHandler += xrs_VEH;
var xr = XmlReader.Create("books.xml", xrs);
```

```
public void xrs_VEH(object sender, ValidationEventArgs e)
{
  // e.Error, e.Message
}
```

- When adding a schema, pass null or empty string to specify the default XML namespace

Reading XML with the XmlReader
http://msdn.microsoft.com/en-us/library/9d83k261.aspx

C.6

System.Xml

XML Namespaces

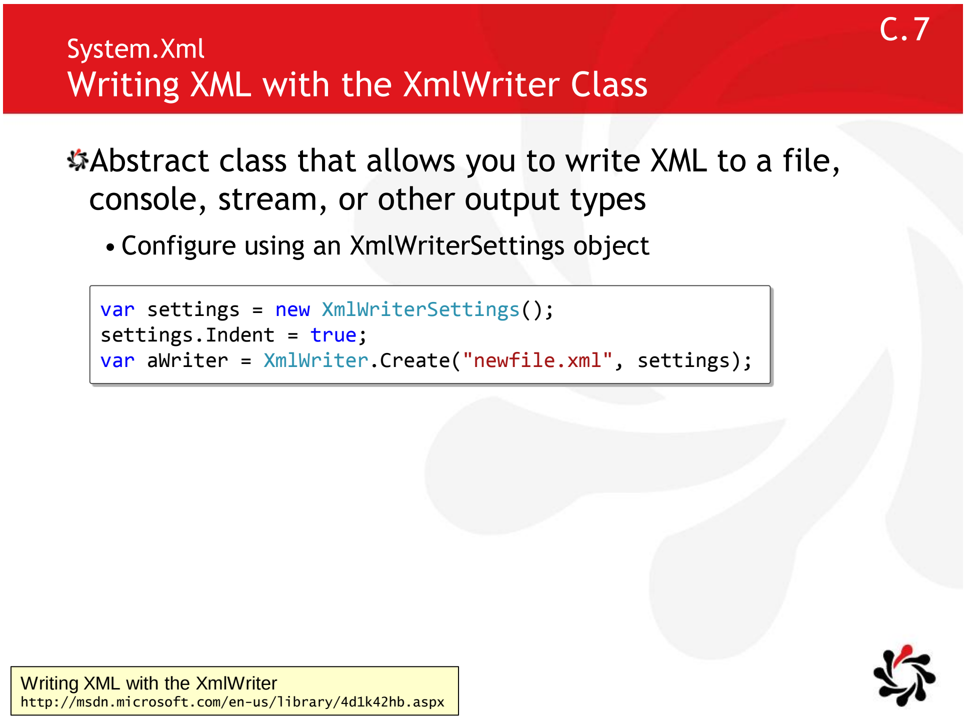
- What’s the difference between Name and LocalName?
  - XML can have namespaces with prefixes to differentiate elements and attributes defined by different schemas
  - Elements without prefixes belong to the default namespace

```
<item xmlns:media="http://schemas.microsoft.com/video/2006/04">
  <title>Article 1</title>
  <description><![CDATA[How to use StackOverflow.com]]></description>
  <link>http://youtube.com/?v=y6_-cLWwEU0</link>
  <media:player url="http://youtube.com/?v=y6_-cLWwEU0"    />
  <media:thumbnail url="http://img.youtube.com/vi/y6_-cLWwEU0/default.jpg"
      width="120" height="90" />
```

  - Use **Name** to retrieve the prefix:name (e.g. media:player) or **LocalName** to retrieve just the name without the prefix (player)

Understanding XML Namespaces
http://msdn.microsoft.com/en-us/magazine/cc302166.aspx

XmlNode.LocalName Property
http://msdn.microsoft.com/en-us/library/system.xml.xmlnode.localname(v=vs.110).aspx



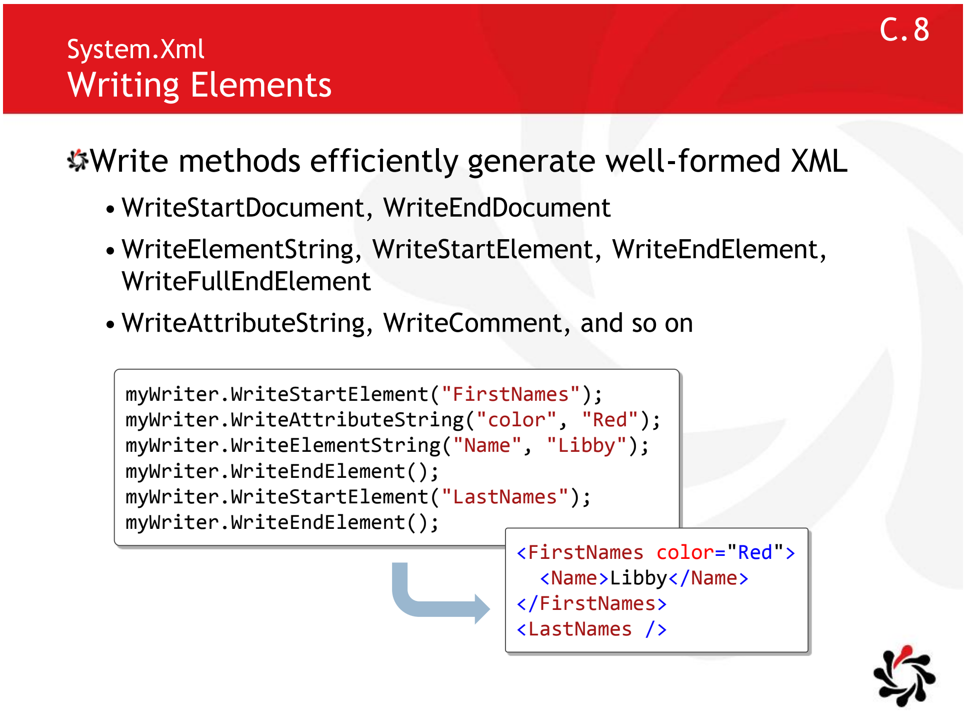
C.7

System.Xml
Writing XML with the XmlWriter Class

- Abstract class that allows you to write XML to a file, console, stream, or other output types
  - Configure using an XmlWriterSettings object

```
var settings = new XmlWriterSettings();
settings.Indent = true;
var aWriter = XmlWriter.Create("newfile.xml", settings);
```

Writing XML with the XmlWriter
http://msdn.microsoft.com/en-us/library/4d1k42hb.aspx



C.8

System.Xml
Writing Elements

- Write methods efficiently generate well-formed XML
  - WriteStartDocument, WriteEndDocument
  - WriteElementString, WriteStartElement, WriteEndElement, WriteFullEndElement
  - WriteAttributeString, WriteComment, and so on

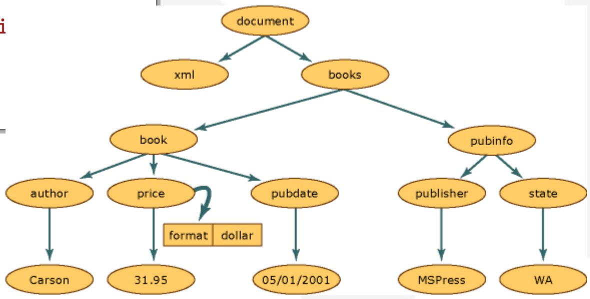
```
myWriter.WriteStartElement("FirstNames");
myWriter.WriteAttributeString("color", "Red");
myWriter.WriteElementString("Name", "Libby");
myWriter.WriteEndElement();
myWriter.WriteStartElement("LastNames");
myWriter.WriteEndElement();
```

```
<FirstNames color="Red">
  <Name>Libby</Name>
</FirstNames>
<LastNames />
```

C.9

System.Xml
XmlDocument

```
<?xml version="1.0"?>
<books>
  <book>
    <author>Carson</author>
    <price format="dollar">31.95</price>
    <pubdate>05/01/2001</pubdate>
  </book>
  <pubinfo>
    <publisher>MSPress</publi
    <state>WA</state>
  </pubinfo>
</books>
```



Process XML Data Using the DOM Model
http://msdn.microsoft.com/en-us/library/t058x2df.aspx

C.10

System.Xml
XSLT Transformations

- The Transform method accepts three input types for the source document: an object that implements the IXPathNavigable interface, an XmlReader object that reads the source document, or a string URI

```
using System.Xml;
using System.Xml.XPath;
using System.Xml.Xsl;
```

```
var doc = new XPathDocument("books.xml");
var writer = XmlWriter.Create("books.html");
var transform = new XslCompiledTransform();
var settings = new XsltSettings();
settings.EnableScript = true; // if you use scripting in the XSLT
transform.Load("transform.xsl", settings, null);
transform.Transform(doc, writer);
```

XSLT Transformations
http://msdn.microsoft.com/en-us/library/14689742.aspx

C.11

DataSets and XML

DataSet.WriteXml

- WriteXml method can be used to write the data to an XML file or a stream

```
aDataSet.WriteXml(HttpContext.Current.Server.MapPath(
  @"App_Data\employee.xml"));
```

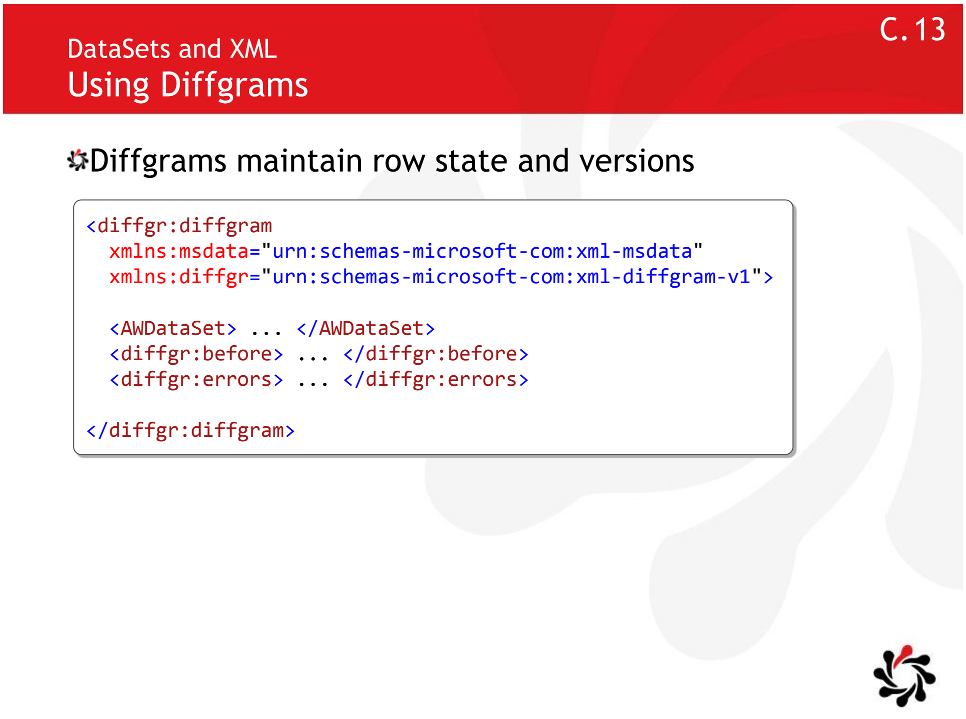
```
<?xml version="1.0" standalone="yes" ?>
<NewDataSet>
  <Employee>
    <Eid>123456789A</Eid>
    <FirstName>Nancy</FirstName>
    ...
  </Employee>
  <Employee>
...
```

C.12

DataSets and XML

Writing a DataSet as XML

- GetXml method
  - Returns XML as a string
- XmlWriteMode enumeration
  - IgnoreSchema (default): outputs data only
  - WriteSchema: outputs an inline schema as well as data
  - DiffGram: outputs original and changed data
- For multiple tables in a DataSet
  - The result of the WriteXml is XML that lists each table in the order they were defined in the DataSet’s table collection
  - To enforce nesting of nodes for related tables, set the Nested property of the DataRelation object to true



C.13

DataSets and XML

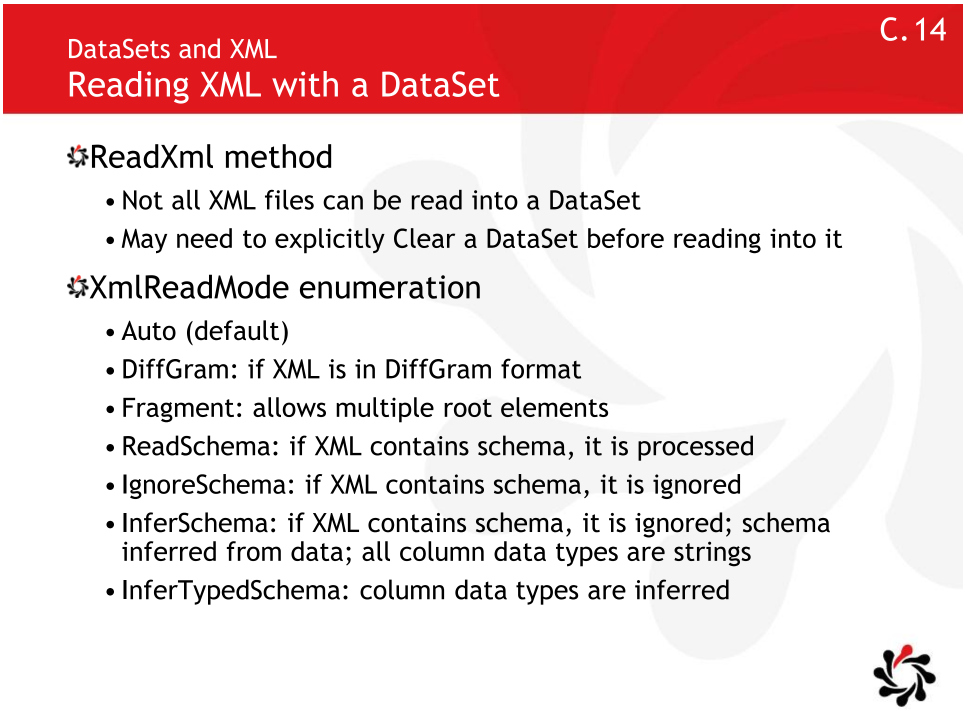
Using Diffgrams

- Diffgrams maintain row state and versions

```
<diffgr:diffgram
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
  xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">

  <AWDataSet> ... </AWDataSet>
  <diffgr:before> ... </diffgr:before>
  <diffgr:errors> ... </diffgr:errors>

</diffgr:diffgram>
```

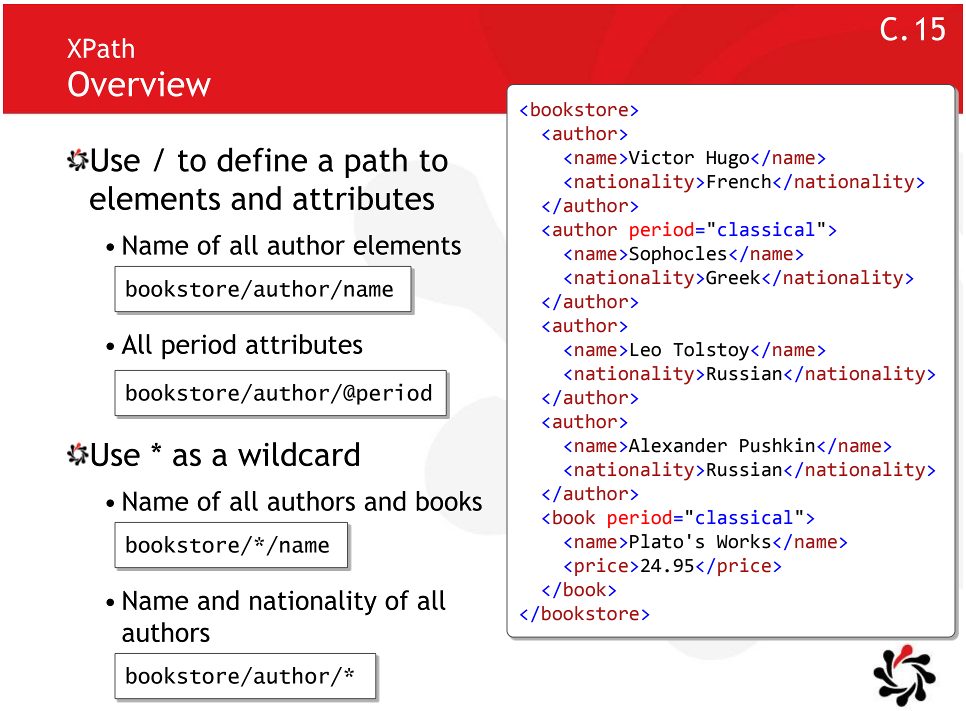


C.14

DataSets and XML

Reading XML with a DataSet

- ReadXml method
  - Not all XML files can be read into a DataSet
  - May need to explicitly Clear a DataSet before reading into it
- XmlReadMode enumeration
  - Auto (default)
  - DiffGram: if XML is in DiffGram format
  - Fragment: allows multiple root elements
  - ReadSchema: if XML contains schema, it is processed
  - IgnoreSchema: if XML contains schema, it is ignored
  - InferSchema: if XML contains schema, it is ignored; schema inferred from data; all column data types are strings
  - InferTypedSchema: column data types are inferred



C.15

XPath
Overview

- Use / to define a path to elements and attributes
  - Name of all author elements

    ```
    bookstore/author/name
    ```

  - All period attributes

    ```
    bookstore/author/@period
    ```

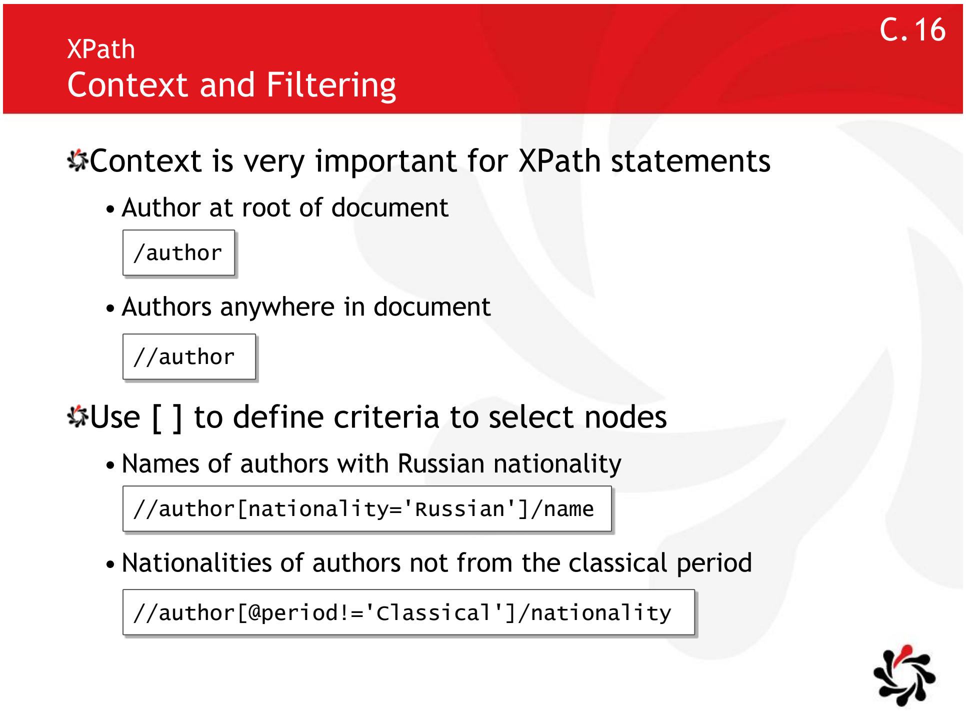
- Use * as a wildcard
  - Name of all authors and books

    ```
    bookstore/*/name
    ```

  - Name and nationality of all authors

    ```
    bookstore/author/*
    ```

```
<bookstore>
  <author>
    <name>Victor Hugo</name>
    <nationality>French</nationality>
  </author>
  <author period="classical">
    <name>Sophocles</name>
    <nationality>Greek</nationality>
  </author>
  <author>
    <name>Leo Tolstoy</name>
    <nationality>Russian</nationality>
  </author>
  <author>
    <name>Alexander Pushkin</name>
    <nationality>Russian</nationality>
  </author>
  <book period="classical">
    <name>Plato's Works</name>
    <price>24.95</price>
  </book>
</bookstore>
```



C.16

XPath
Context and Filtering

- Context is very important for XPath statements
  - Author at root of document

    ```
    /author
    ```

  - Authors anywhere in document

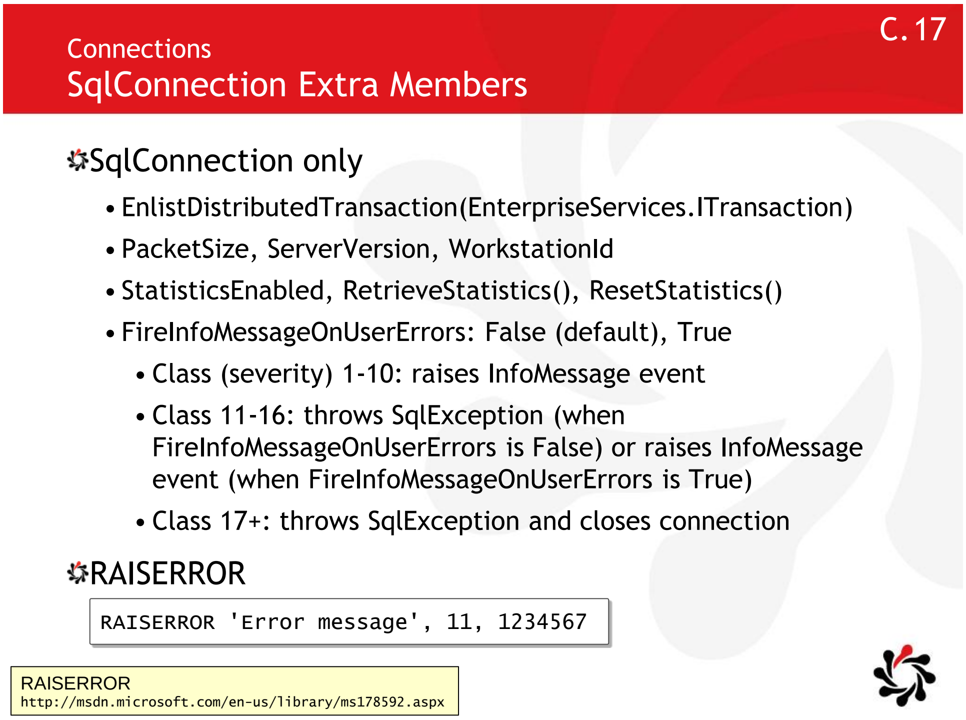
    ```
    //author
    ```

- Use [ ] to define criteria to select nodes
  - Names of authors with Russian nationality

    ```
    //author[nationality='Russian']/name
    ```

  - Nationalities of authors not from the classical period

    ```
    //author[@period!='Classical']/nationality
    ```



C.17

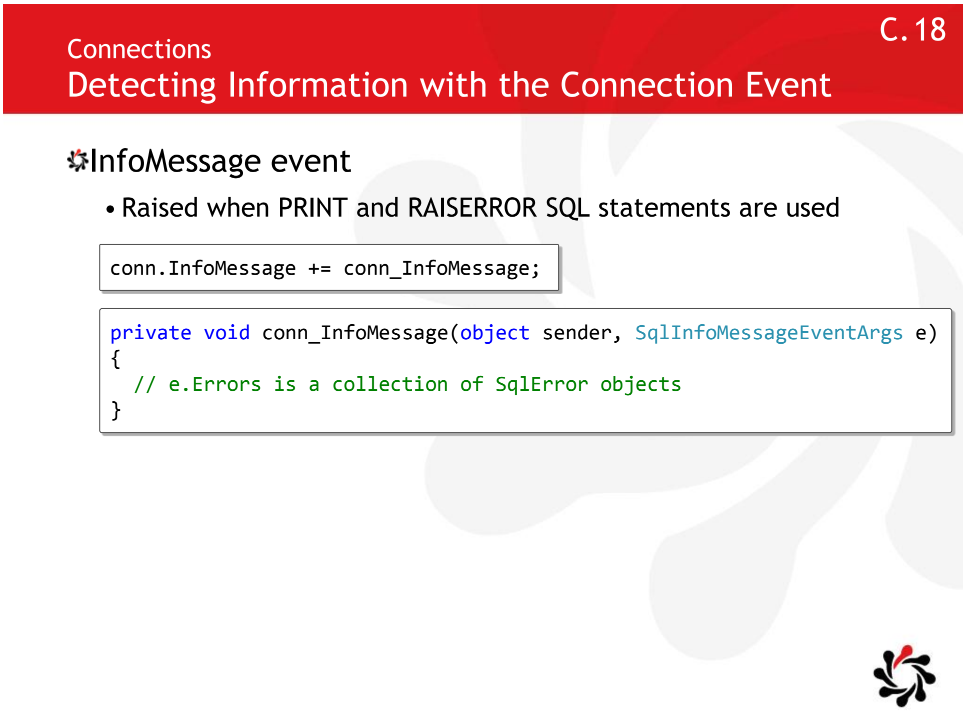
Connections

SqlConnection Extra Members

- SqlConnection only
  - EnlistDistributedTransaction(EnterpriseServices.ITransaction)
  - PacketSize, ServerVersion, WorkstationId
  - StatisticsEnabled, RetrieveStatistics(), ResetStatistics()
  - FireInfoMessageOnUserErrors: False (default), True
    - Class (severity) 1-10: raises InfoMessage event
    - Class 11-16: throws SqlException (when FireInfoMessageOnUserErrors is False) or raises InfoMessage event (when FireInfoMessageOnUserErrors is True)
    - Class 17+: throws SqlException and closes connection
- RAISERROR

```
RAISERROR 'Error message', 11, 1234567
```

RAISERROR
http://msdn.microsoft.com/en-us/library/ms178592.aspx



C.18

Connections

Detecting Information with the Connection Event

- InfoMessage event
  - Raised when PRINT and RAISERROR SQL statements are used

```
conn.InfoMessage += conn_InfoMessage;
```

```
private void conn_InfoMessage(object sender, SqlInfoMessageEventArgs e)
{
  // e.Errors is a collection of SqlError objects
}
```

C.19

Connections

Configuring Connection Strings

- Connection strings provide the necessary information for an application to connect a data source
  - Server, database, security information, connection pool size
- Many parameters have multiple possible keywords for backwards compatibility e.g. these all mean the same
  - Data Source, server, address, addr, Network Address
- SQL Server (servername\instancename)

```
Data Source=FLORIDA\MIAMI;Initial Catalog=Northwind;
Integrated Security=SSPI;
```

```
database=Northwind;
```

- OLEDB for Microsoft Access

```
Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=C:\MyDataFiles\MyDb.mdb;
```

C.20

Connections

Security

- Windows Authentication
  - Secure as no password is transmitted across the network
  - Not easy to configure across domains or firewalls
  - Use of service accounts is highly recommended to aid pooling

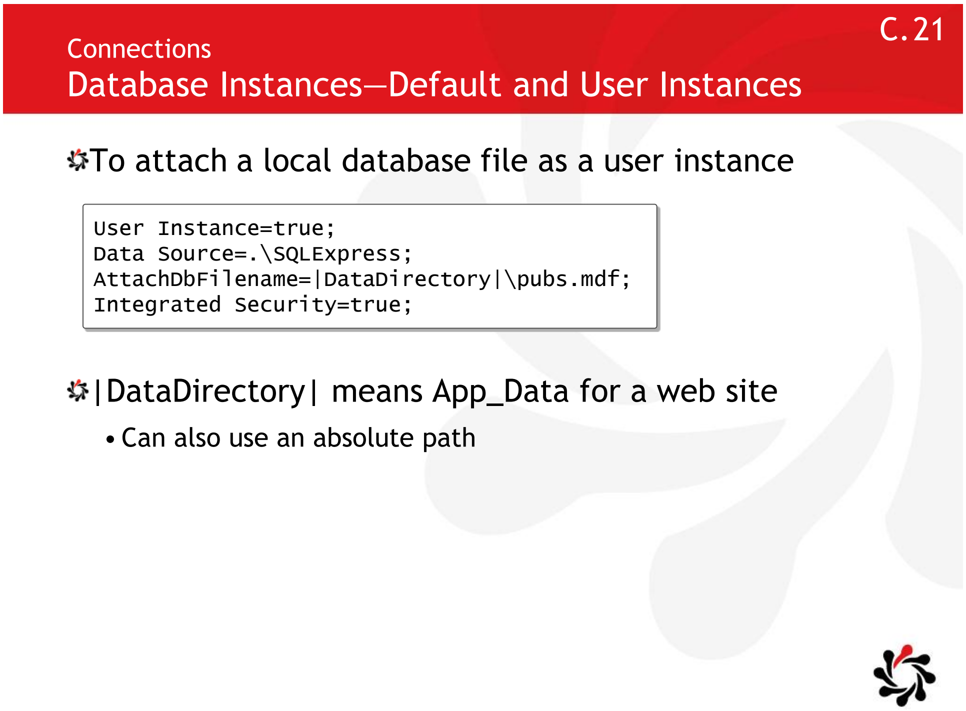
```
"...;Integrated Security=SSPI;"
```

```
"...;Trusted_Connection=true;"
```

- Mixed (aka SQL) Authentication
  - Not as secure because user id and password sent across network
  - Works across domains and through firewalls
  - Only option when using SQL Databases in Windows Azure

```
"...;User Id=Fred;Password=secret;Persist Security Info=false;"
```

```
"...;uid=Fred;pwd=secret;PersistSecurityInfo=false;"
```



C.21

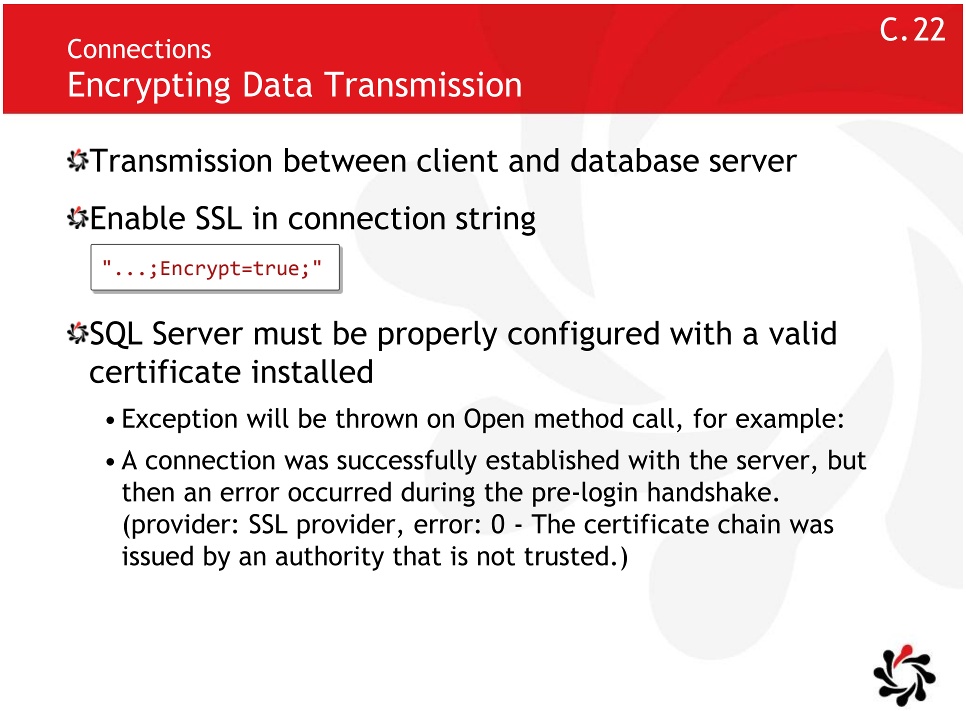
Connections

Database Instances—Default and User Instances

- To attach a local database file as a user instance

```
User Instance=true;
Data Source=.\SQLExpress;
AttachDbFilename=|DataDirectory|\pubs.mdf;
Integrated Security=true;
```

- |DataDirectory| means App_Data for a web site
  - Can also use an absolute path



C.22

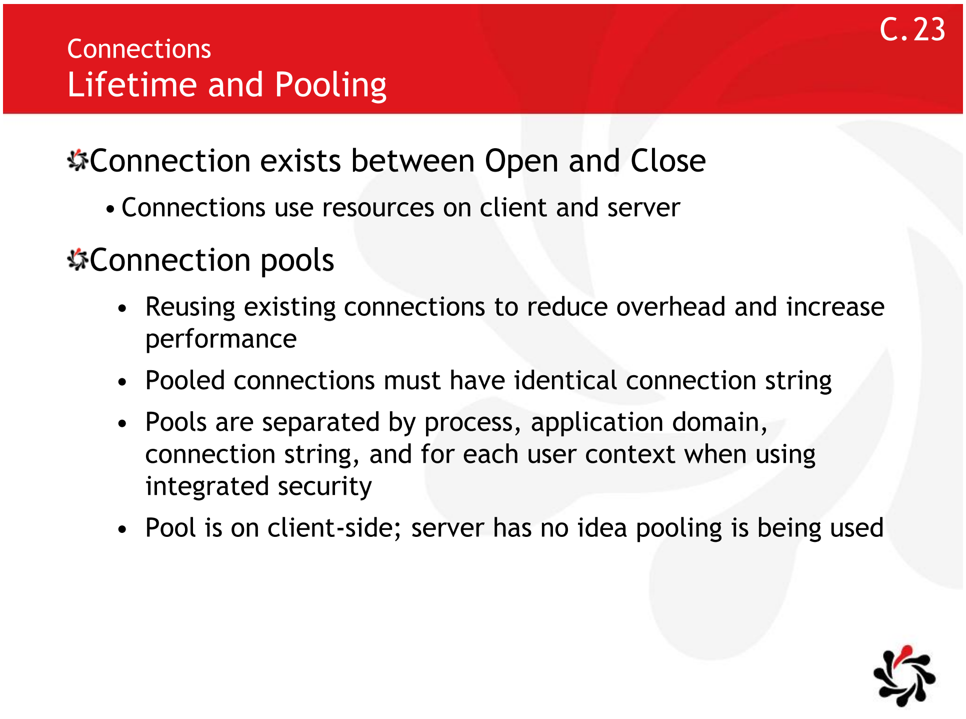
Connections

Encrypting Data Transmission

- Transmission between client and database server
- Enable SSL in connection string

```
"...;Encrypt=true;"
```

- SQL Server must be properly configured with a valid certificate installed
  - Exception will be thrown on Open method call, for example:
  - A connection was successfully established with the server, but then an error occurred during the pre-login handshake. (provider: SSL provider, error: 0 - The certificate chain was issued by an authority that is not trusted.)

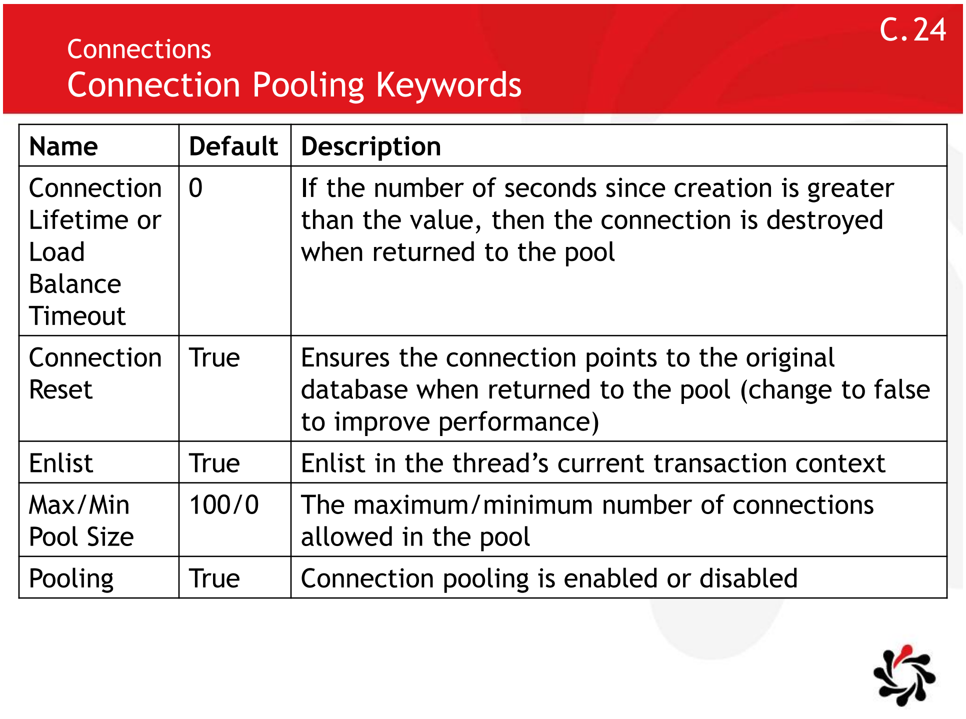


C.23

Connections

Lifetime and Pooling

- Connection exists between Open and Close
  - Connections use resources on client and server
- Connection pools
  - Reusing existing connections to reduce overhead and increase performance
  - Pooled connections must have identical connection string
  - Pools are separated by process, application domain, connection string, and for each user context when using integrated security
  - Pool is on client-side; server has no idea pooling is being used

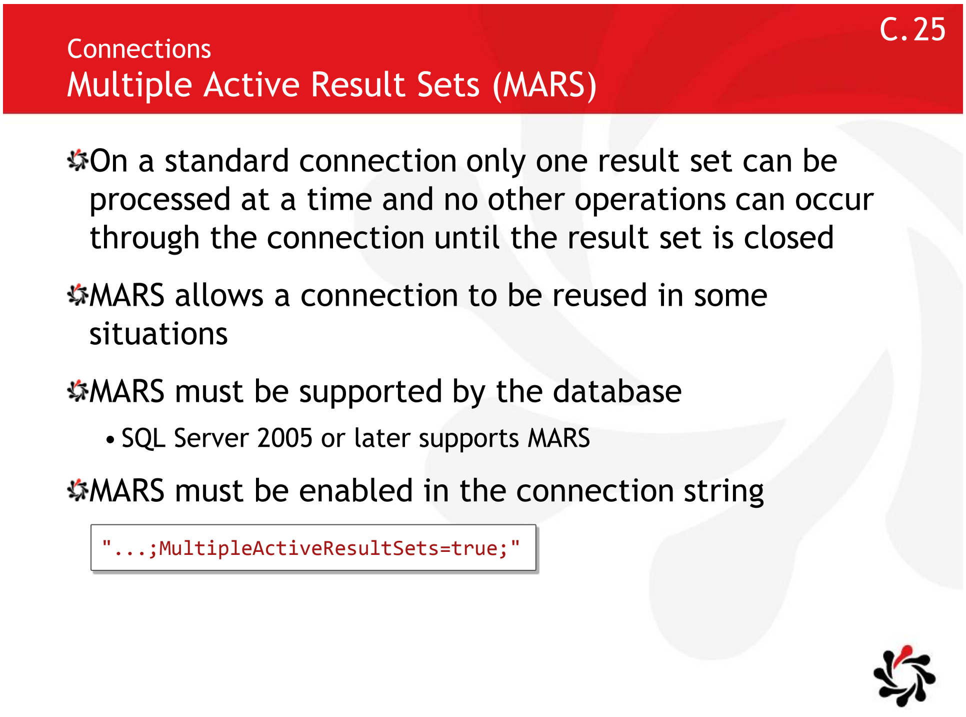


C.24

Connections

Connection Pooling Keywords

Name	Default	Description
Connection Lifetime or Load Balance Timeout	0	If the number of seconds since creation is greater than the value, then the connection is destroyed when returned to the pool
Connection Reset	True	Ensures the connection points to the original database when returned to the pool (change to false to improve performance)
Enlist	True	Enlist in the thread’s current transaction context
Max/Min Pool Size	100/0	The maximum/minimum number of connections allowed in the pool
Pooling	True	Connection pooling is enabled or disabled



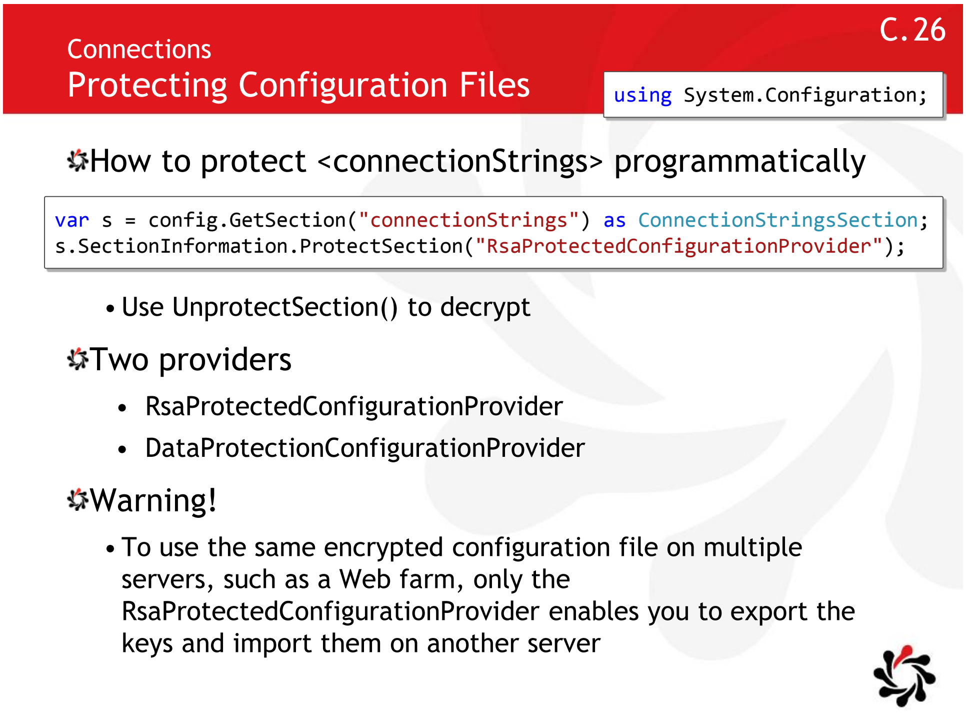
C.25

Connections

Multiple Active Result Sets (MARS)

- On a standard connection only one result set can be processed at a time and no other operations can occur through the connection until the result set is closed
- MARS allows a connection to be reused in some situations
- MARS must be supported by the database
  - SQL Server 2005 or later supports MARS
- MARS must be enabled in the connection string

```
"...;MultipleActiveResultSets=true;"
```



C.26

Connections

Protecting Configuration Files

```
using System.Configuration;
```

- How to protect <connectionStrings> programmatically

```
var s = config.GetSection("connectionStrings") as ConnectionStringsSection;
s.SectionInformation.ProtectSection("RsaProtectedConfigurationProvider");
```

  - Use UnprotectSection() to decrypt
- Two providers
  - RsaProtectedConfigurationProvider
  - DataProtectionConfigurationProvider
- Warning!
  - To use the same encrypted configuration file on multiple servers, such as a Web farm, only the RsaProtectedConfigurationProvider enables you to export the keys and import them on another server

C.27

Connections

ConnectionStringBuilder

- To dynamically but safely build a connection string

```
var sb = new SqlConnectionStringBuilder();
sb["Data Source"] = @".\SQLEXPRESS"; // or sb.DataSource
sb["database"] = "AdventureWorks";
// or sb["Initial Catalog"]
sb["Integrated Security"] = true;
// this connection string will use the modern keywords
string connectionString = sb.ConnectionString;
```

- Can also be used to test a connection string
  - An exception is thrown if any values are invalid

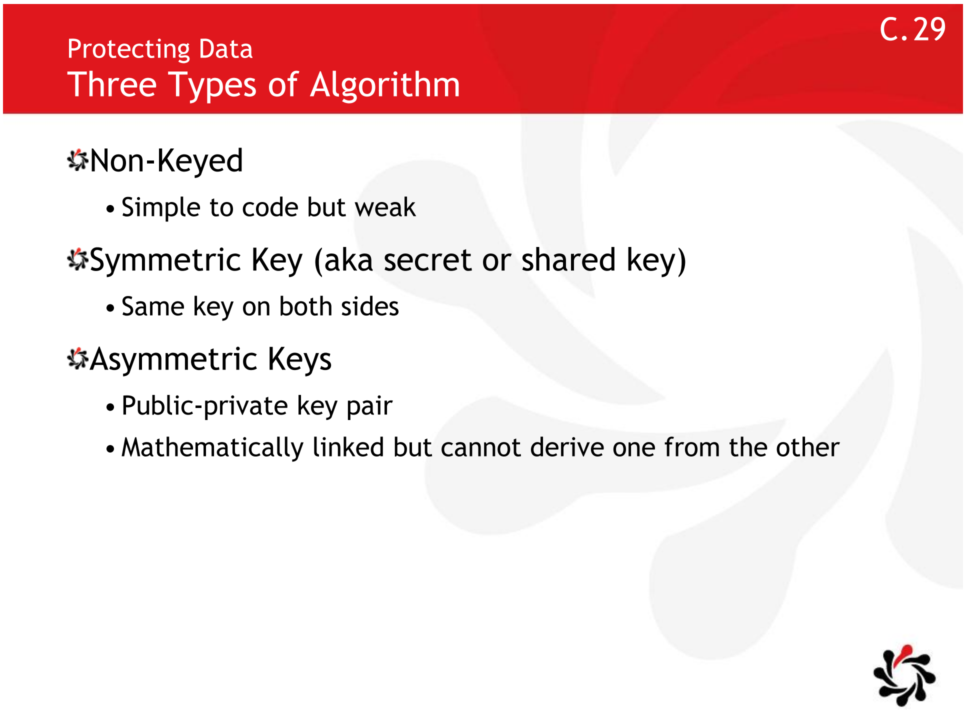
```
var sb = new SqlConnectionStringBuilder(conStrToTest);
// or set ConnectionString property
```

C.28

Protecting Data

Three Techniques

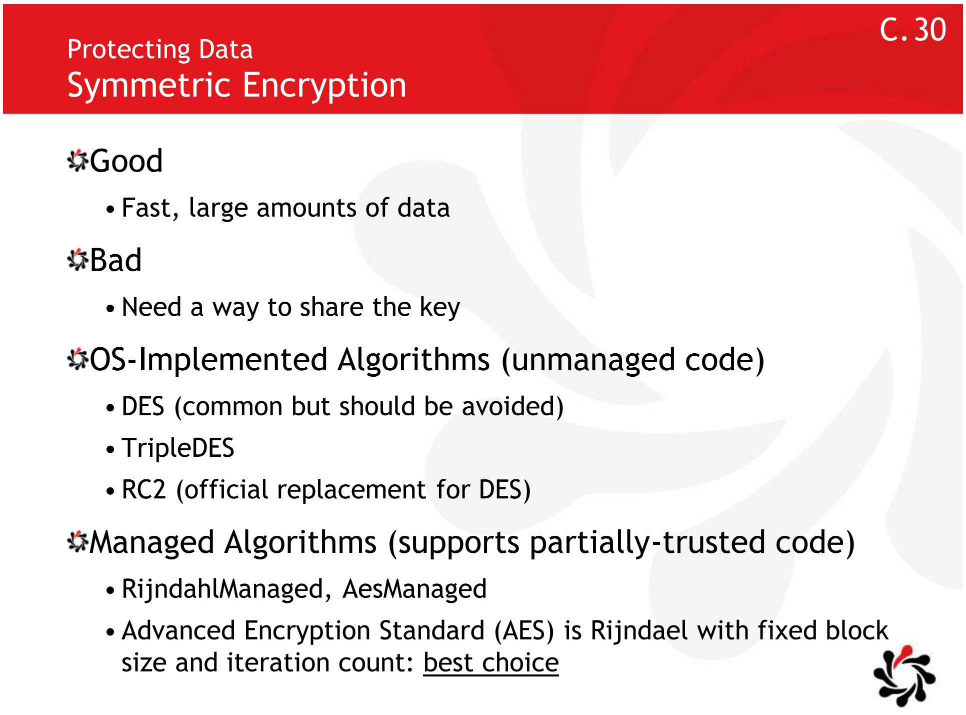
- Encrypt
  - Two-way operation (i.e. can be decrypted)
  - Best choice for data such as credit card numbers
- Hash (integrity check)
  - One-way operation (i.e. cannot create original data from hash)
  - A checksum that is unique to a piece of data to ensure no modification occurs
  - Best choice for data such as passwords
- Sign (authentication check)
  - A digital signature is a value that is appended to electronic data to prove it was created by someone who possesses a specific private key; the public key is used to verify the signature at the receiver's end



C.29

Protecting Data
Three Types of Algorithm

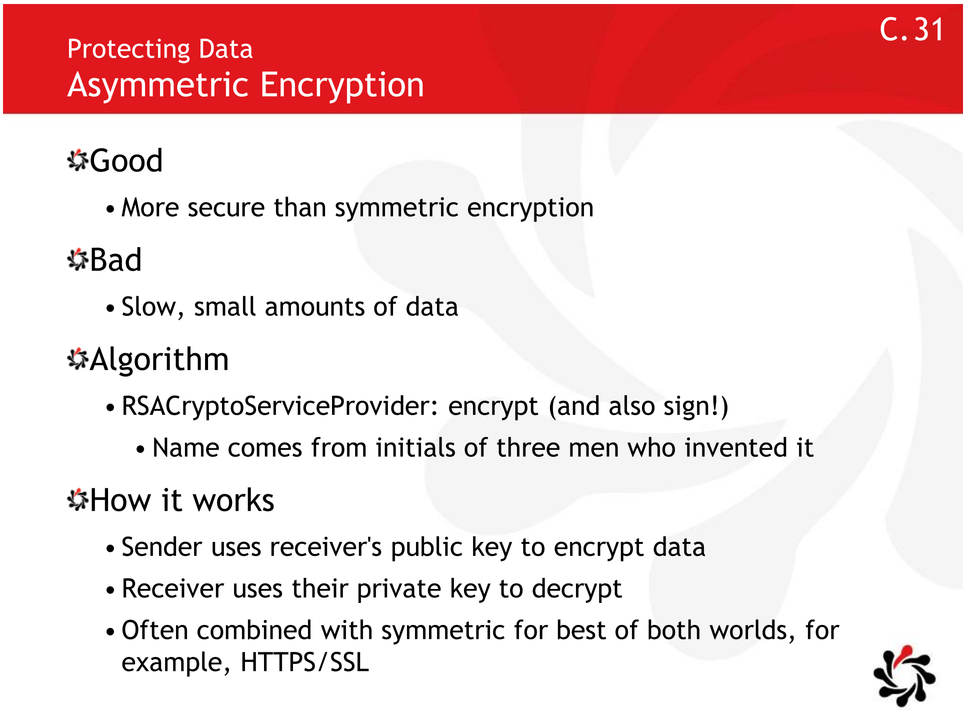
- Non-Keyed
  - Simple to code but weak
- Symmetric Key (aka secret or shared key)
  - Same key on both sides
- Asymmetric Keys
  - Public-private key pair
  - Mathematically linked but cannot derive one from the other



C.30

Protecting Data
Symmetric Encryption

- Good
  - Fast, large amounts of data
- Bad
  - Need a way to share the key
- OS-Implemented Algorithms (unmanaged code)
  - DES (common but should be avoided)
  - TripleDES
  - RC2 (official replacement for DES)
- Managed Algorithms (supports partially-trusted code)
  - RijndahlManaged, AesManaged
  - Advanced Encryption Standard (AES) is Rijndael with fixed block size and iteration count: best choice

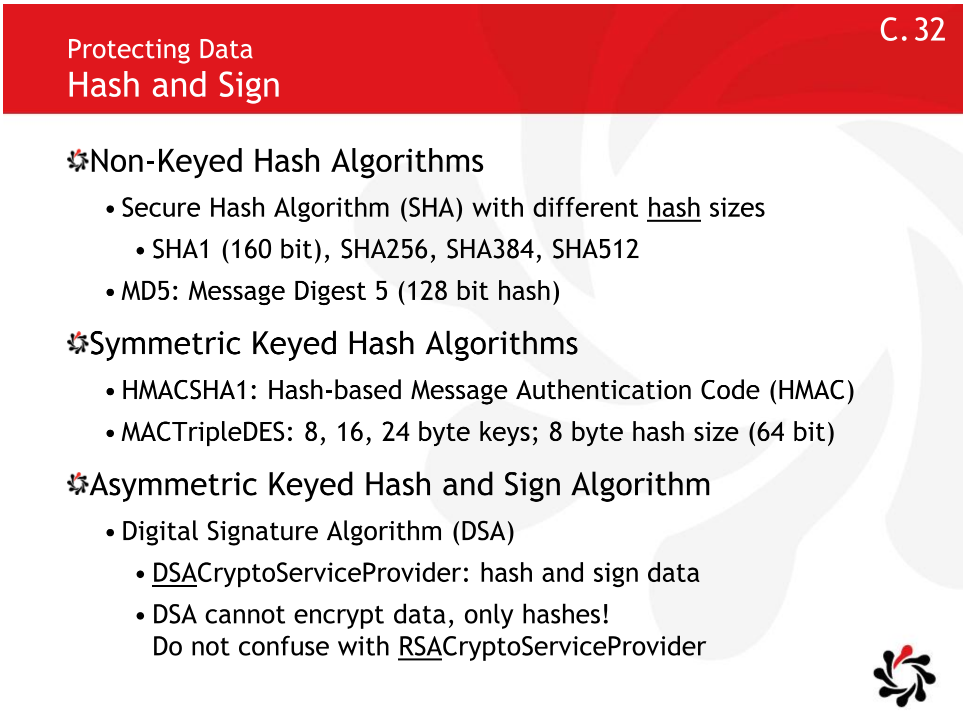


C.31

Protecting Data

Asymmetric Encryption

- Good
  - More secure than symmetric encryption
- Bad
  - Slow, small amounts of data
- Algorithm
  - RSACryptoServiceProvider: encrypt (and also sign!)
    - Name comes from initials of three men who invented it
- How it works
  - Sender uses receiver's public key to encrypt data
  - Receiver uses their private key to decrypt
  - Often combined with symmetric for best of both worlds, for example, HTTPS/SSL

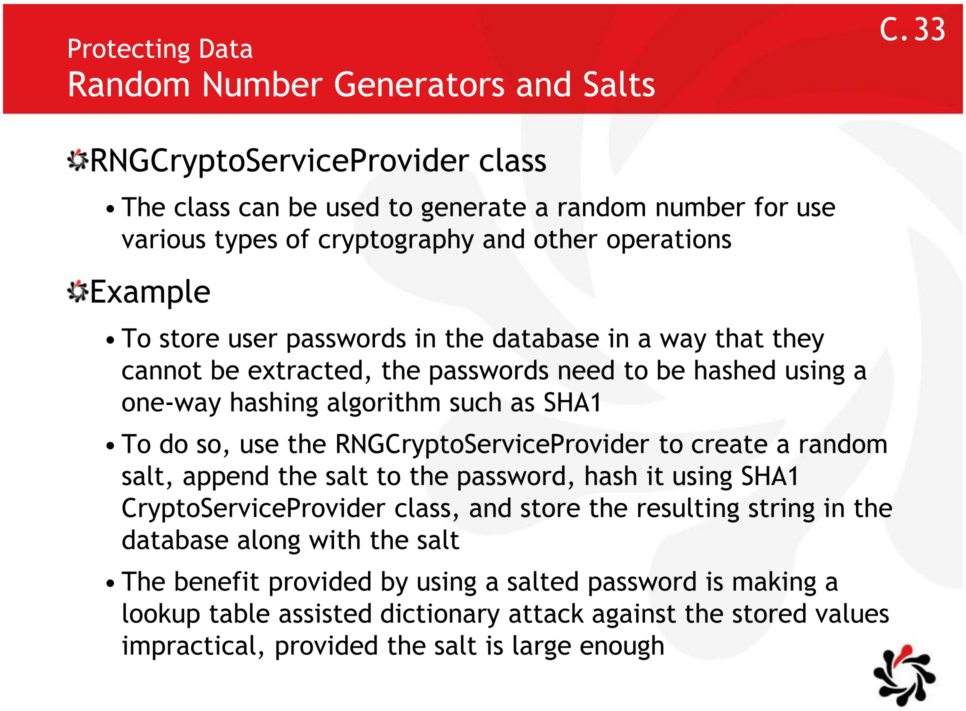


C.32

Protecting Data

Hash and Sign

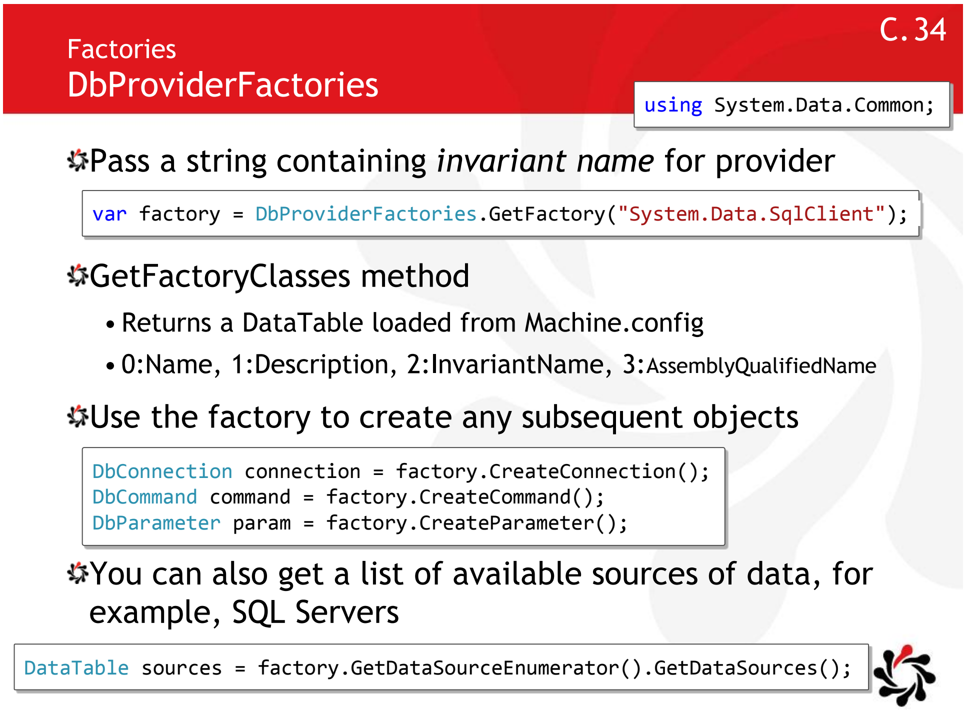
- Non-Keyed Hash Algorithms
  - Secure Hash Algorithm (SHA) with different hash sizes
    - SHA1 (160 bit), SHA256, SHA384, SHA512
  - MD5: Message Digest 5 (128 bit hash)
- Symmetric Keyed Hash Algorithms
  - HMACSHA1: Hash-based Message Authentication Code (HMAC)
  - MACTripleDES: 8, 16, 24 byte keys; 8 byte hash size (64 bit)
- Asymmetric Keyed Hash and Sign Algorithm
  - Digital Signature Algorithm (DSA)
    - DSACryptoServiceProvider: hash and sign data
    - DSA cannot encrypt data, only hashes!
      Do not confuse with RSACryptoServiceProvider



C.33

Protecting Data
Random Number Generators and Salts

- RNGCryptoServiceProvider class
  - The class can be used to generate a random number for use various types of cryptography and other operations
- Example
  - To store user passwords in the database in a way that they cannot be extracted, the passwords need to be hashed using a one-way hashing algorithm such as SHA1
  - To do so, use the RNGCryptoServiceProvider to create a random salt, append the salt to the password, hash it using SHA1 CryptoServiceProvider class, and store the resulting string in the database along with the salt
  - The benefit provided by using a salted password is making a lookup table assisted dictionary attack against the stored values impractical, provided the salt is large enough



C.34

Factories
DbProviderFactories

```
using System.Data.Common;
```

- Pass a string containing *invariant name* for provider

```
var factory = DbProviderFactories.GetFactory("System.Data.SqlClient");
```

- GetFactoryClasses method
  - Returns a DataTable loaded from Machine.config
  - 0:Name, 1:Description, 2:InvariantName, 3:AssemblyQualifiedName
- Use the factory to create any subsequent objects

```
DbConnection connection = factory.CreateConnection();
DbCommand command = factory.CreateCommand();
DbParameter param = factory.CreateParameter();
```

- You can also get a list of available sources of data, for example, SQL Servers

```
DataTable sources = factory.GetDataSourceEnumerator().GetDataSources();
```

C.35

Commands

DbCommand Members

Methods

- ExecuteNonQuery: returns Int32
- ExecuteReader: returns DbDataReader-derived object
- ExecuteScalar: returns Object
- Prepare, Cancel

Properties

- CommandType: TableDirect, StoredProcedure, Text (default)
- CommandText: name of table, stored procedure, or SQL
- CommandTimeout: default 30 seconds
- Connection, Transaction: objects associated with this command
- Parameters: collection of DbParameters

C.36

Commands

SqlCommand.ExecuteXmlReader

SQL Server 2000 and later supports FOR XML

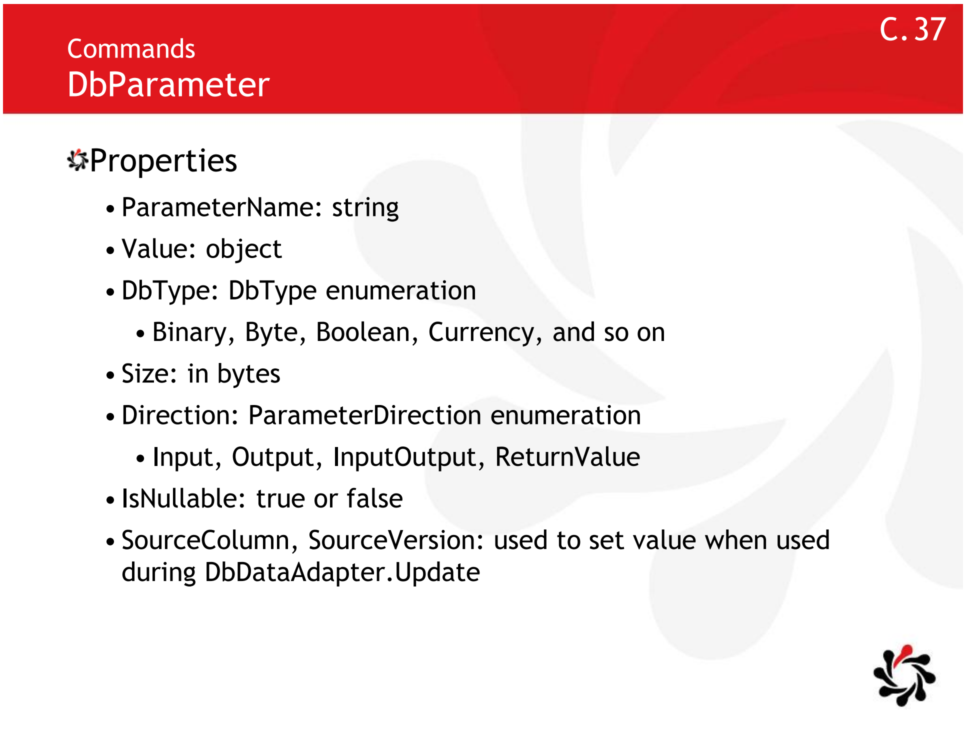
- AUTO means use attributes for column names and values

```
cmd.CommandText = "SELECT CustomerID, CompanyName, " +
  "Country FROM Customers FOR XML AUTO";
```

```
<Customers CustomerID="ALFKI"
  CompanyName="Alfreds Futterkiste" Country="Germany" />
```

```
XmlReader xr = cmd.ExecuteXmlReader();
while (xr.Read())
{
  Console.WriteLine(xr.GetAttribute("CompanyName"));
}
```

- Warning! Use ExecuteScalar to return the result as a string if you just need to write the XML to a file or stream rather than immediately process it with a reader

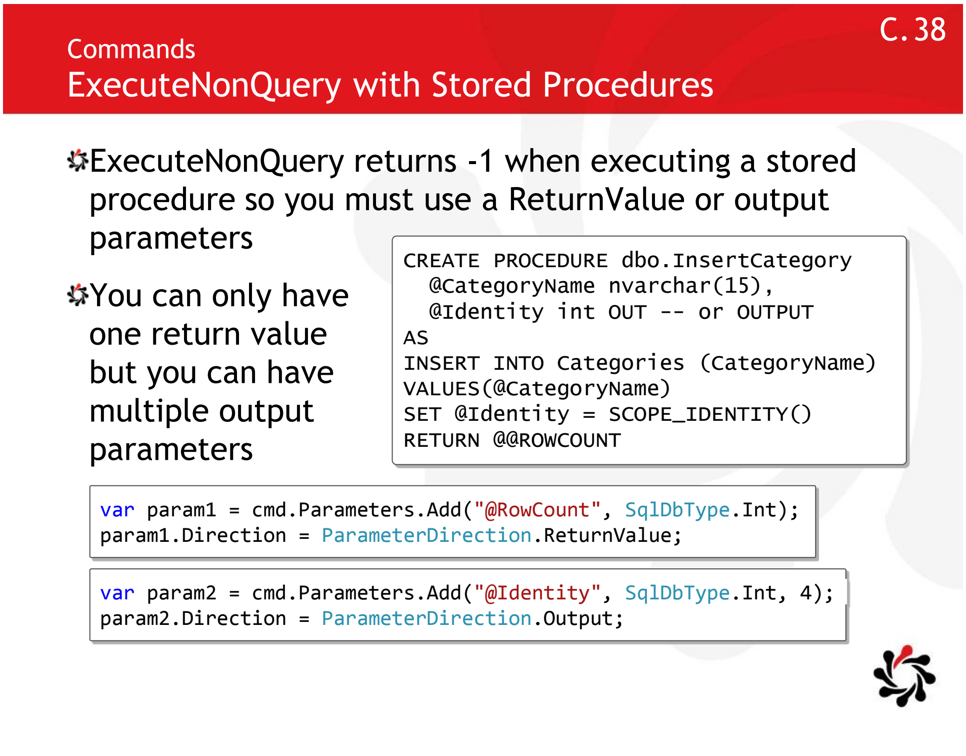


C.37

Commands

DbParameter

- Properties
  - ParameterName: string
  - Value: object
  - DbType: DbType enumeration
    - Binary, Byte, Boolean, Currency, and so on
  - Size: in bytes
  - Direction: ParameterDirection enumeration
    - Input, Output, InputOutput, ReturnValue
  - IsNullable: true or false
  - SourceColumn, SourceVersion: used to set value when used during DbDataAdapter.Update



C.38

Commands

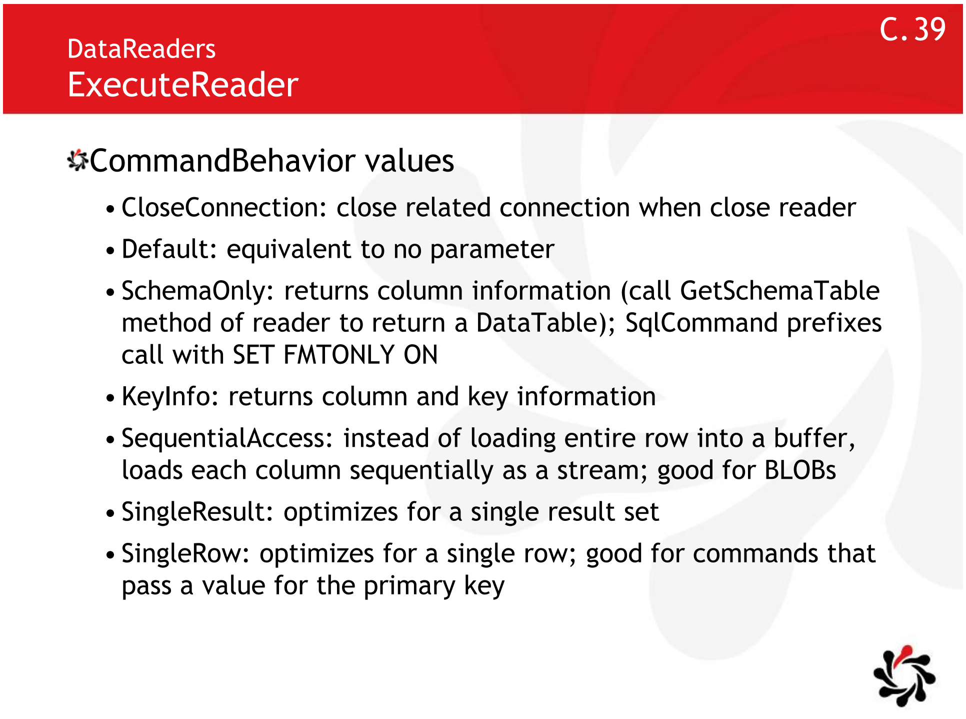
ExecuteNonQuery with Stored Procedures

- ExecuteNonQuery returns -1 when executing a stored procedure so you must use a ReturnValue or output parameters
- You can only have one return value but you can have multiple output parameters

```
CREATE PROCEDURE dbo.InsertCategory
  @CategoryName nvarchar(15),
  @Identity int OUT -- or OUTPUT
AS
INSERT INTO Categories (CategoryName)
VALUES(@CategoryName)
SET @Identity = SCOPE_IDENTITY()
RETURN @@ROWCOUNT
```

```
var param1 = cmd.Parameters.Add("@RowCount", SqlDbType.Int);
param1.Direction = ParameterDirection.ReturnValue;
```

```
var param2 = cmd.Parameters.Add("@Identity", SqlDbType.Int, 4);
param2.Direction = ParameterDirection.Output;
```

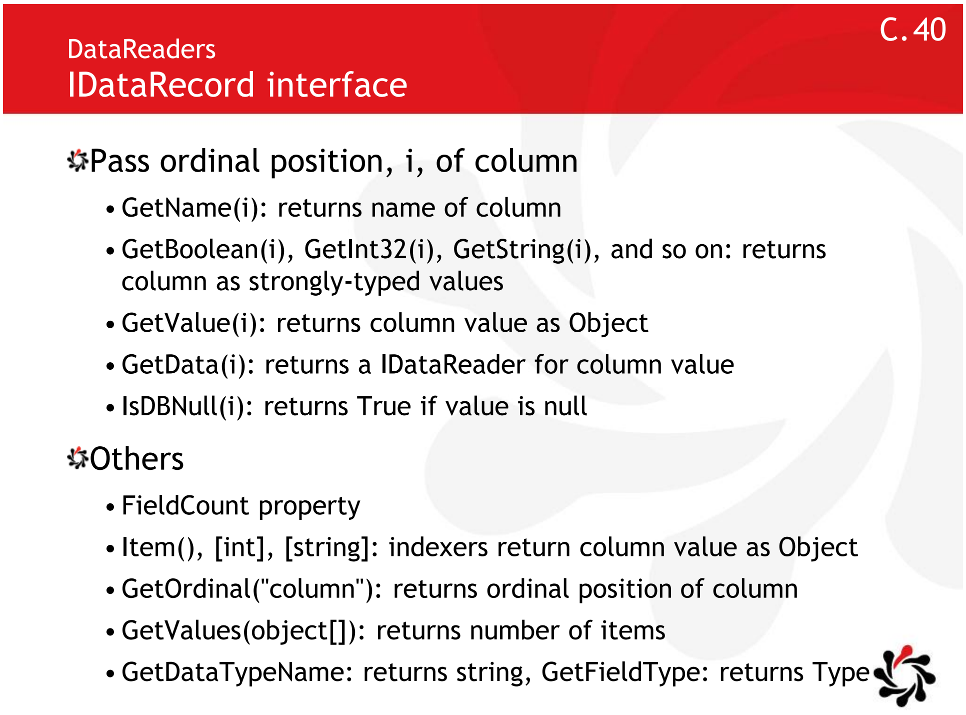


C.39

DataReaders

ExecuteReader

- CommandBehavior values
  - CloseConnection: close related connection when close reader
  - Default: equivalent to no parameter
  - SchemaOnly: returns column information (call GetSchemaTable method of reader to return a DataTable); SqlCommand prefixes call with SET FMTONLY ON
  - KeyInfo: returns column and key information
  - SequentialAccess: instead of loading entire row into a buffer, loads each column sequentially as a stream; good for BLOBs
  - SingleResult: optimizes for a single result set
  - SingleRow: optimizes for a single row; good for commands that pass a value for the primary key



C.40

DataReaders

IDataRecord interface

- Pass ordinal position, i, of column
  - GetName(i): returns name of column
  - GetBoolean(i), GetInt32(i), GetString(i), and so on: returns column as strongly-typed values
  - GetValue(i): returns column value as Object
  - GetData(i): returns a IDataReader for column value
  - IsDBNull(i): returns True if value is null
- Others
  - FieldCount property
  - Item(), [int], [string]: indexers return column value as Object
  - GetOrdinal("column"): returns ordinal position of column
  - GetValues(object[]): returns number of items
  - GetDataTypeName: returns string, GetFieldType: returns Type

C.41

DataReaders

IDataReader and DbDataReader members

IDataReader members

- RecordsAffected: rows inserted, updated, or deleted
- Read(): if true, moves to next row
- NextResult(): if true, moves to next resultset
- IsClosed, Close(): must close IDataReader before reading any output parameters or return values

DbDataReader members

- HasRows
- GetSchemaTable(): returns DataTable with schema info

SqlDataReader only: GetSqlXxx(int)

- GetSqlMoney, GetSqlXml, GetSqlGuid, etc.

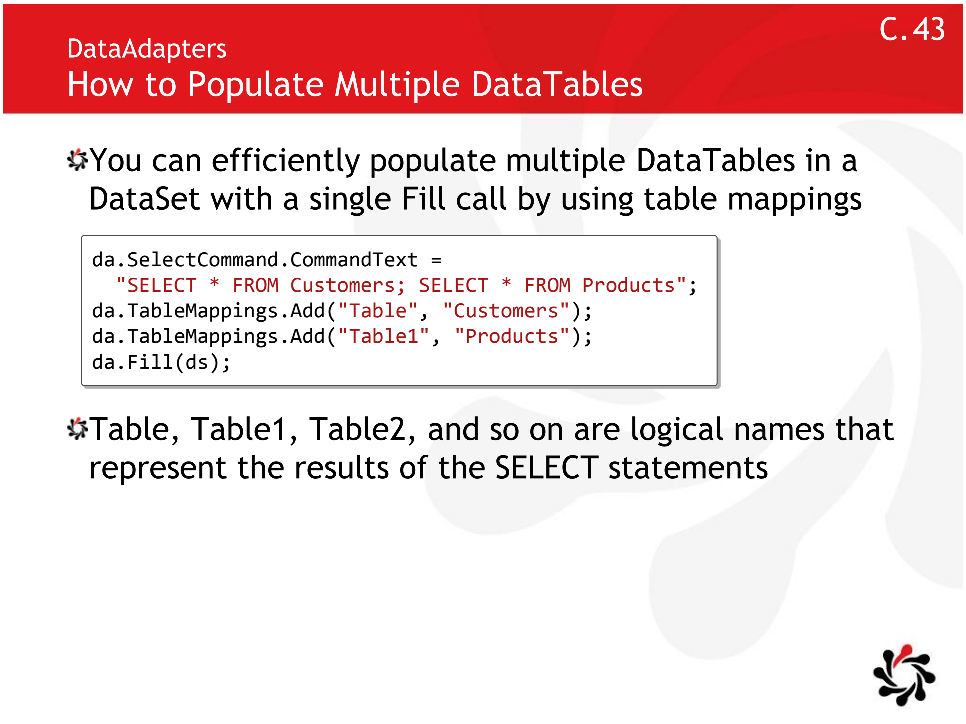
C.42

DataReaders

How to Process Multiple SELECTs

Does NOT need MARS

```
cmd.CommandText = "SELECT * FROM Customers; SELECT * FROM Products";
SqlDataReader reader = cmd.ExecuteReader();
// do NOT call NextResult() yet!
do
{
  while (reader.Read())
  {
    Console.WriteLine(reader[0].ToString());
  }
} while (reader.NextResult());
```



C.43

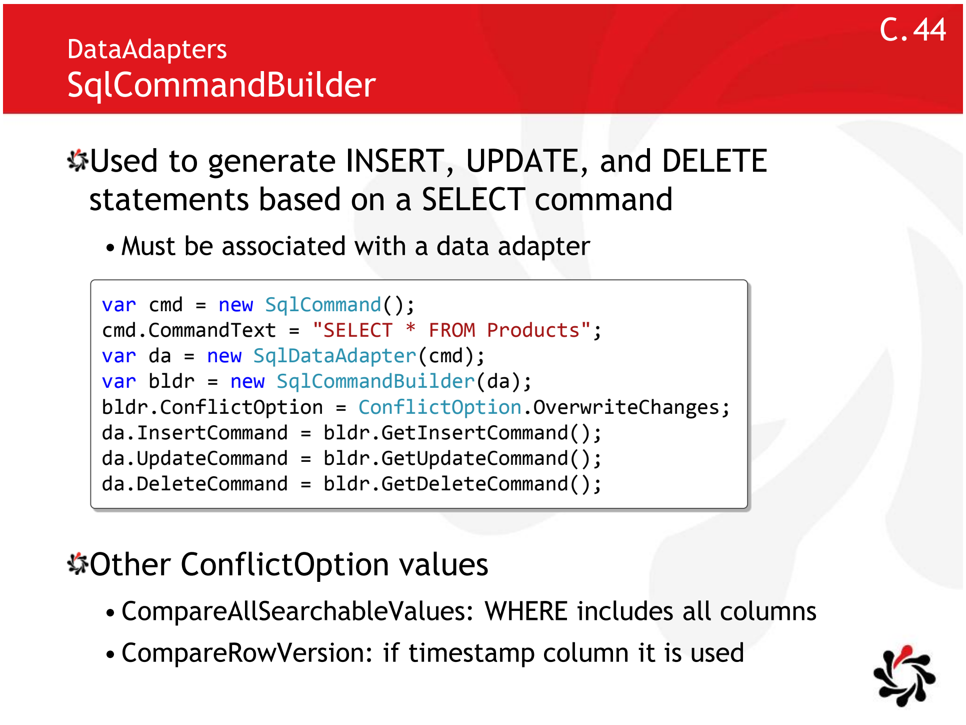
DataAdapters

How to Populate Multiple DataTables

- You can efficiently populate multiple DataTables in a DataSet with a single Fill call by using table mappings

```
da.SelectCommand.CommandText =
  "SELECT * FROM Customers; SELECT * FROM Products";
da.TableMappings.Add("Table", "Customers");
da.TableMappings.Add("Table1", "Products");
da.Fill(ds);
```

- Table, Table1, Table2, and so on are logical names that represent the results of the SELECT statements



C.44

DataAdapters

SqlCommandBuilder

- Used to generate INSERT, UPDATE, and DELETE statements based on a SELECT command
  - Must be associated with a data adapter

```
var cmd = new SqlCommand();
cmd.CommandText = "SELECT * FROM Products";
var da = new SqlDataAdapter(cmd);
var bldr = new SqlCommandBuilder(da);
bldr.ConflictOption = ConflictOption.OverwriteChanges;
da.InsertCommand = bldr.GetInsertCommand();
da.UpdateCommand = bldr.GetUpdateCommand();
da.DeleteCommand = bldr.GetDeleteCommand();
```

- Other ConflictOption values
  - CompareAllSearchableValues: WHERE includes all columns
  - CompareRowVersion: if timestamp column it is used

C.45

DataAdapters

Performing Bulk Updates

- The UpdateBatchSize property of the the DbDataAdapter object’s allows you to send updates to the DB in batches rather than record by record
- The UpdateBatchSize property value can be set to
  - 1 (default), which indicates one row at a time
  - 0, all updated rows in one batch
  - Any other value to indicate number of rows to include in the batch

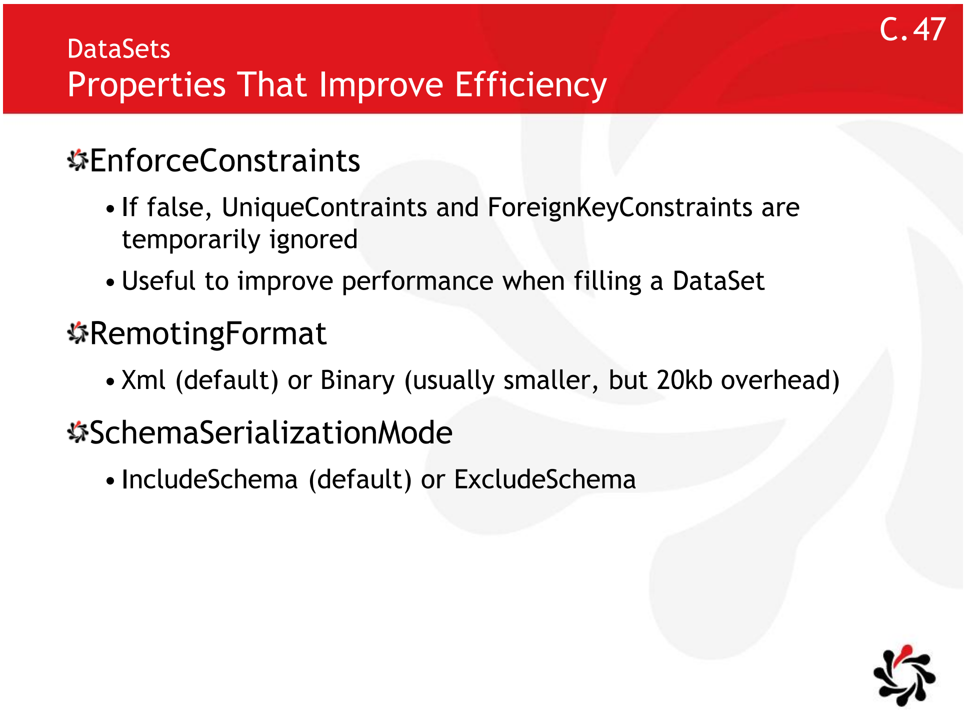
```
da.UpdateBatchSize = 3;
da.Update(pubsDataSet, "publishers");
```

C.46

DataAdapters

TableAdapterManager

- Enables hierarchical updates (ADO.NET 3.5+)
  - Created at design-time with a typed DataSet so not in BCL
  - Uses the foreign-key constraints in data tables to determine the correct order to send the Inserts, Updates, and Deletes from a dataset to the database
- Members
  - BackupDataSetBeforeUpdate: use for AutoIncrements that must not miss values
  - UpdateOrder: InsertUpdateDelete (default)
  - UpdateAll(): implicit transaction

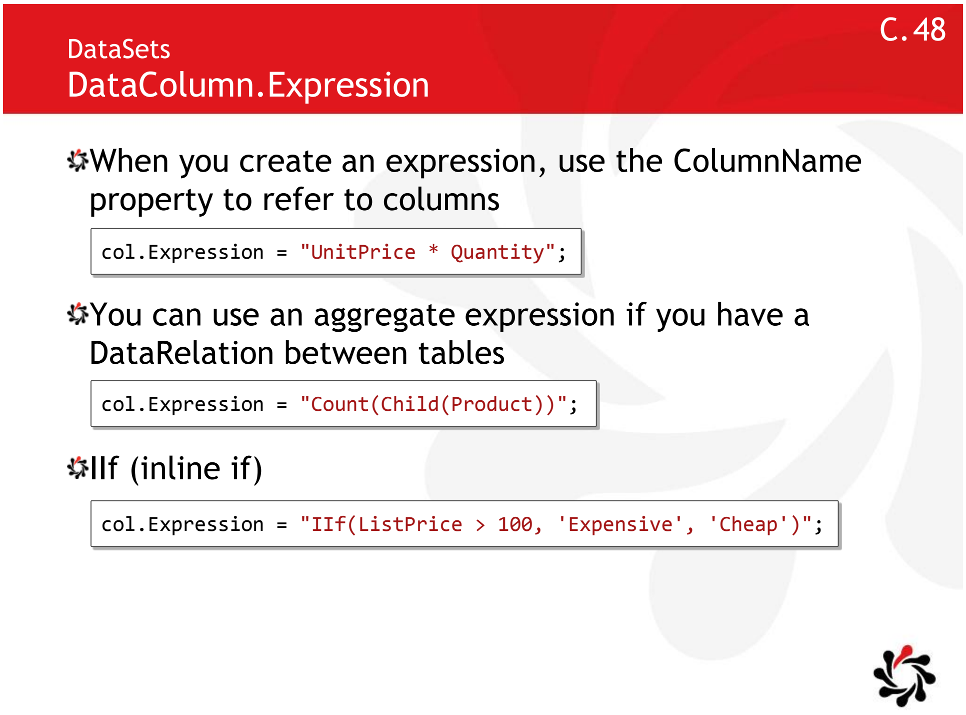


C.47

DataSets

Properties That Improve Efficiency

- EnforceConstraints
  - If false, UniqueContraints and ForeignKeyConstraints are temporarily ignored
  - Useful to improve performance when filling a DataSet
- RemotingFormat
  - Xml (default) or Binary (usually smaller, but 20kb overhead)
- SchemaSerializationMode
  - IncludeSchema (default) or ExcludeSchema



C.48

DataSets

DataColumn.Expression

- When you create an expression, use the ColumnName property to refer to columns

```
col.Expression = "UnitPrice * Quantity";
```

- You can use an aggregate expression if you have a DataRelation between tables

```
col.Expression = "Count(Child(Product))";
```

- IIf (inline if)

```
col.Expression = "IIf(ListPrice > 100, 'Expensive', 'Cheap')";
```

C.49

DataSets

DataRow.DataRowState

- Detached
  - Newly created row not yet added to Rows collection
- Added
  - Newly created row now member of Rows collection
- Unchanged
  - Row that has not been modified; only has current version
- Modified
  - Row that has been modified; has current and original versions
- Deleted
  - Row that has been marked for deletion; only has original version

C.50

DataSets

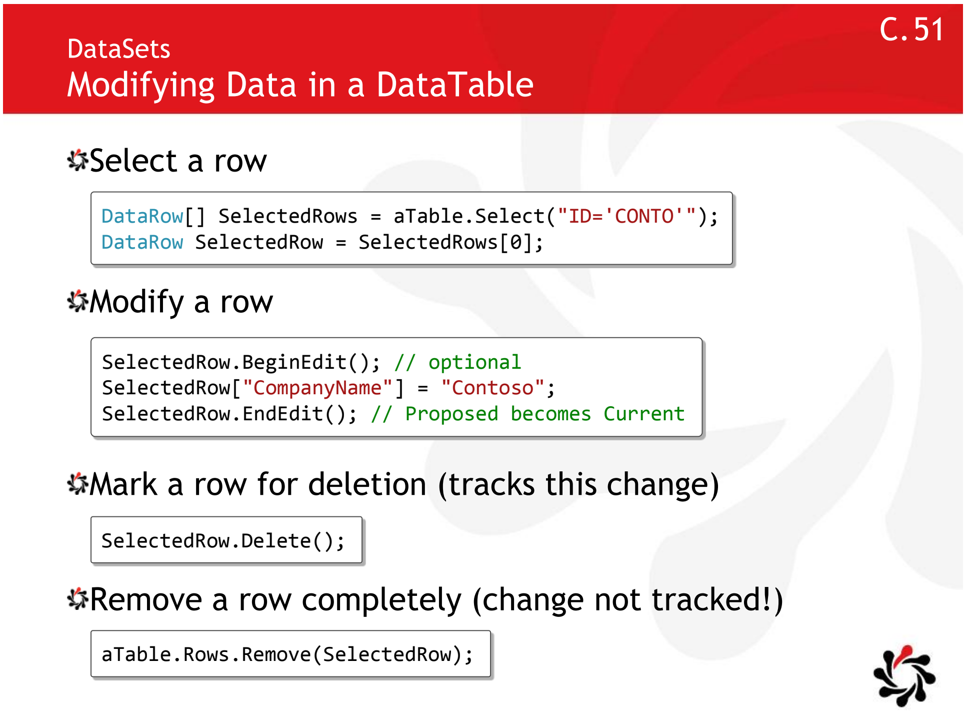
DataRowVersion

- DataRows can have up to four versions
  - Current
  - Original
  - Proposed: only exists until EndEdit is called on an edited row
  - Default: only one copy of this version exists for each table
- Read versions with overloaded indexer

```
s = row["FirstName", DataRowVersion.Original];
```

- VersionNotFoundException is raised if you attempt to retrieve a version that does not exist
  - Use HasVersion method to test for existence of a version

```
if(row.HasVersion(DataRowVersion.Original))
```



C.51

DataSets

Modifying Data in a DataTable

- Select a row

```
DataRow[] SelectedRows = aTable.Select("ID='CONTO'");
DataRow SelectedRow = SelectedRows[0];
```

- Modify a row

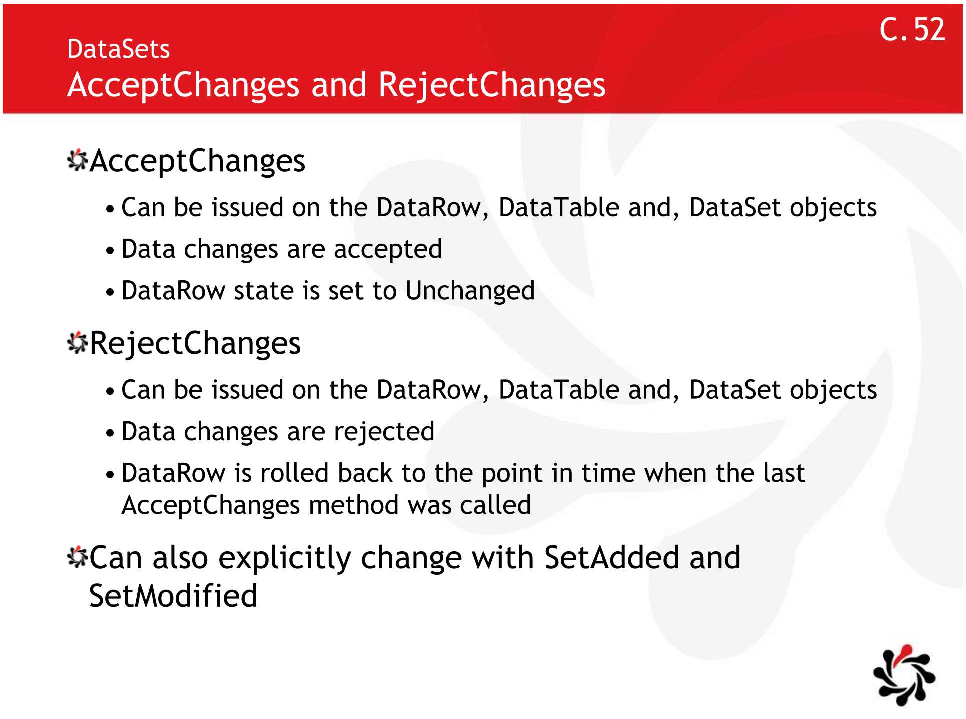
```
SelectedRow.BeginEdit(); // optional
SelectedRow["CompanyName"] = "Contoso";
SelectedRow.EndEdit(); // Proposed becomes Current
```

- Mark a row for deletion (tracks this change)

```
SelectedRow.Delete();
```

- Remove a row completely (change not tracked!)

```
aTable.Rows.Remove(SelectedRow);
```

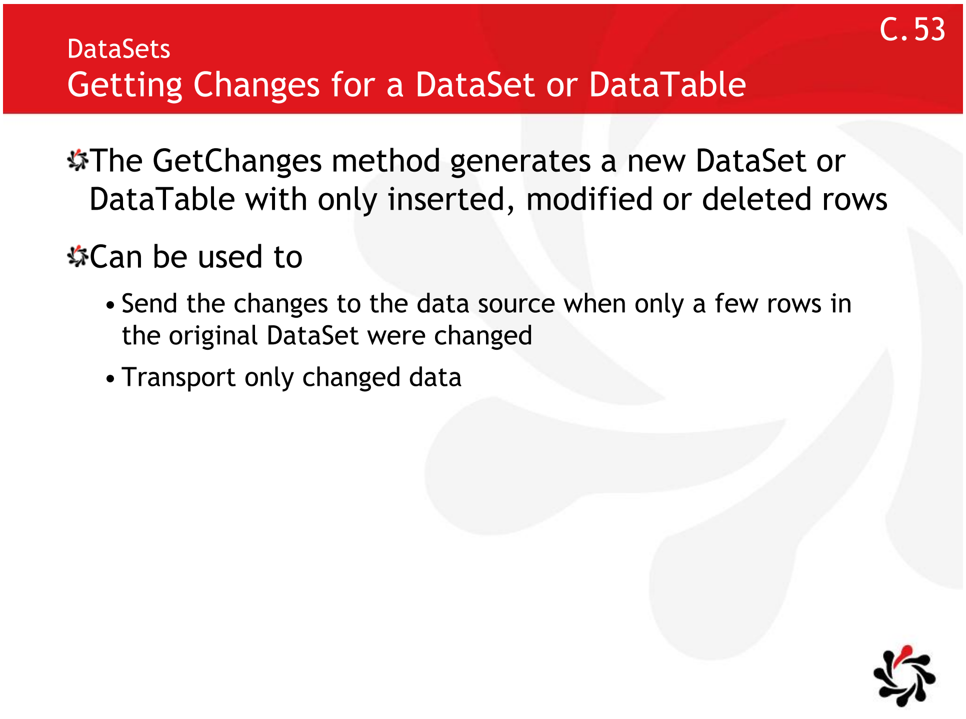


C.52

DataSets

AcceptChanges and RejectChanges

- AcceptChanges
  - Can be issued on the DataRow, DataTable and, DataSet objects
  - Data changes are accepted
  - DataRow state is set to Unchanged
- RejectChanges
  - Can be issued on the DataRow, DataTable and, DataSet objects
  - Data changes are rejected
  - DataRow is rolled back to the point in time when the last AcceptChanges method was called
- Can also explicitly change with SetAdded and SetModified

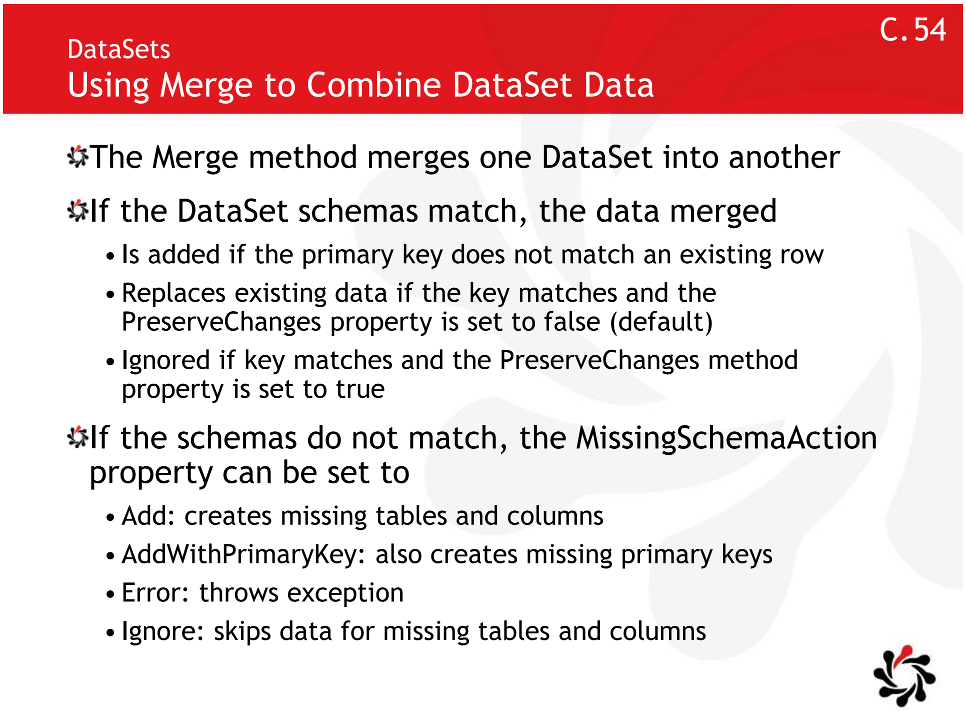


C.53

DataSets

Getting Changes for a DataSet or DataTable

- The GetChanges method generates a new DataSet or DataTable with only inserted, modified or deleted rows
- Can be used to
  - Send the changes to the data source when only a few rows in the original DataSet were changed
  - Transport only changed data



C.54

DataSets

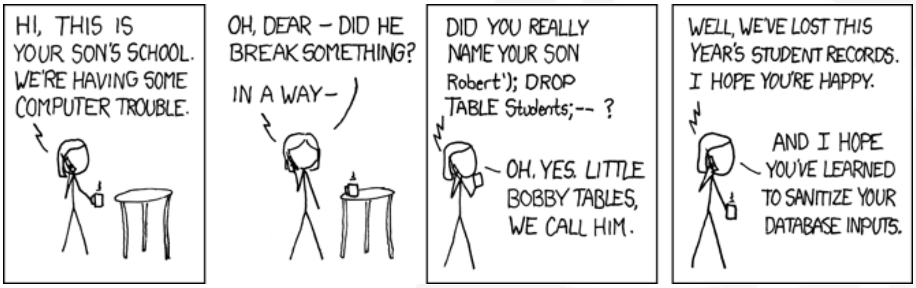
Using Merge to Combine DataSet Data

- The Merge method merges one DataSet into another
- If the DataSet schemas match, the data merged
  - Is added if the primary key does not match an existing row
  - Replaces existing data if the key matches and the PreserveChanges property is set to false (default)
  - Ignored if key matches and the PreserveChanges method property is set to true
- If the schemas do not match, the MissingSchemaAction property can be set to
  - Add: creates missing tables and columns
  - AddWithPrimaryKey: also creates missing primary keys
  - Error: throws exception
  - Ignore: skips data for missing tables and columns

DataSets

Data Integrity

- Add a ForeignKeyConstraint to Constraints collection of the foreign table
- Properties of ForeignKeyConstraint
  - Table: Orders, Columns: CustomerID
  - RelatedTable: Customer, RelatedColumns: ID
- DeleteRule, UpdateRule
  - Cascade (default): child rows deleted or updated (if necessary)
  - None (throw exception), SetDefault, SetNull
- AcceptRejectRule
  - None (default)
  - Cascade: AcceptChanges or RejectChanges on related child rows


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Exploits of a Mom
http://xkcd.com/327/

C.57

SQL Server

Avoiding SQL Injection Attacks

- SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution
  - For example, this bad code reads a value posted from a web form and concatenates it into a SQL statement

```
var ShipCity = Request.Form("ShipCity");
var sql = "select * from OrdersTable where ShipCity = '" + ShipCity + "'";
```

  - A malicious user could enter the following for ShipCity

```
Redmond'; drop table OrdersTable--
```

- Reject the following characters: '   ;   --   /*   */   xp_
  - Better: use paramterized statements or stored procedures

SQL Injection
http://msdn.microsoft.com/en-us/library/ms161953.aspx

C.58

SQL Server

Stored Procedures

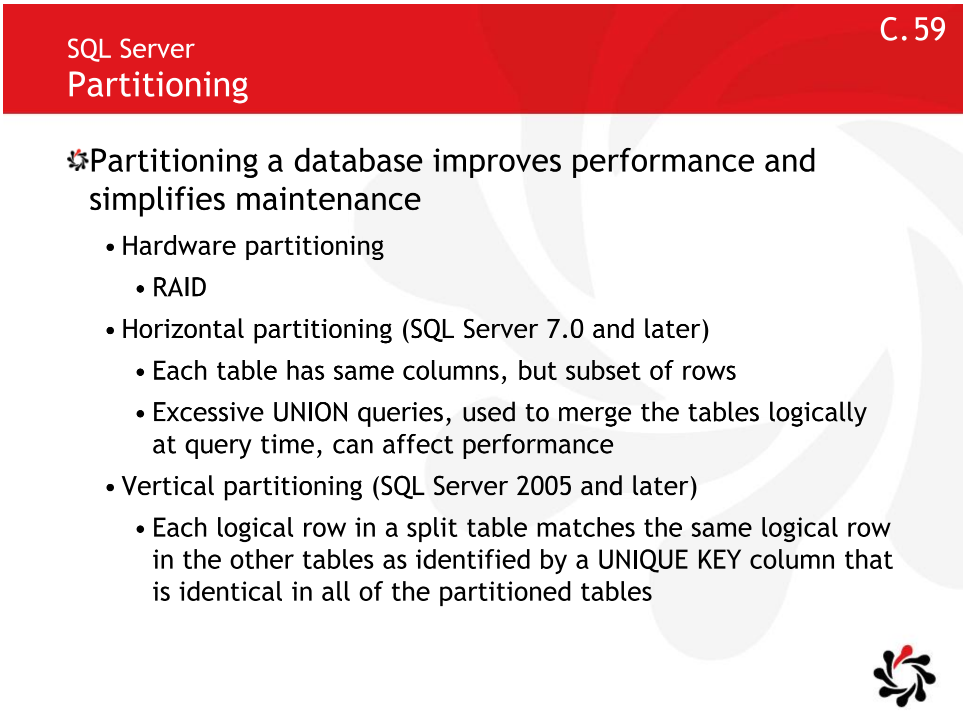
- Stored procedures should use parameters as their input

```
CREATE PROCEDURE HumanResources.uspGetEmployees
    @LastName nvarchar(50) = NULL, --default value
    @FirstName nvarchar(50),
    @HowMany int OUTPUT /* or OUT */
AS
    SET NOCOUNT ON;
    SET @HowMany = SELECT COUNT(*) FROM ...;
    SELECT FirstName, LastName, Department
    FROM HumanResources.vEmployeeDepartmentHistory
    WHERE FirstName = @FirstName AND LastName = @LastName;
```

- Some organizations require use of stored procedures, others such as Facebook don’t use them at all

http://highscalability.com/blog/2010/11/9/facebook-uses-non-stored-procedures-to-update-social-graphs.html

How to: Create a Stored Procedure (SQL Server Management Studio)
http://msdn.microsoft.com/en-us/library/ms345415.aspx

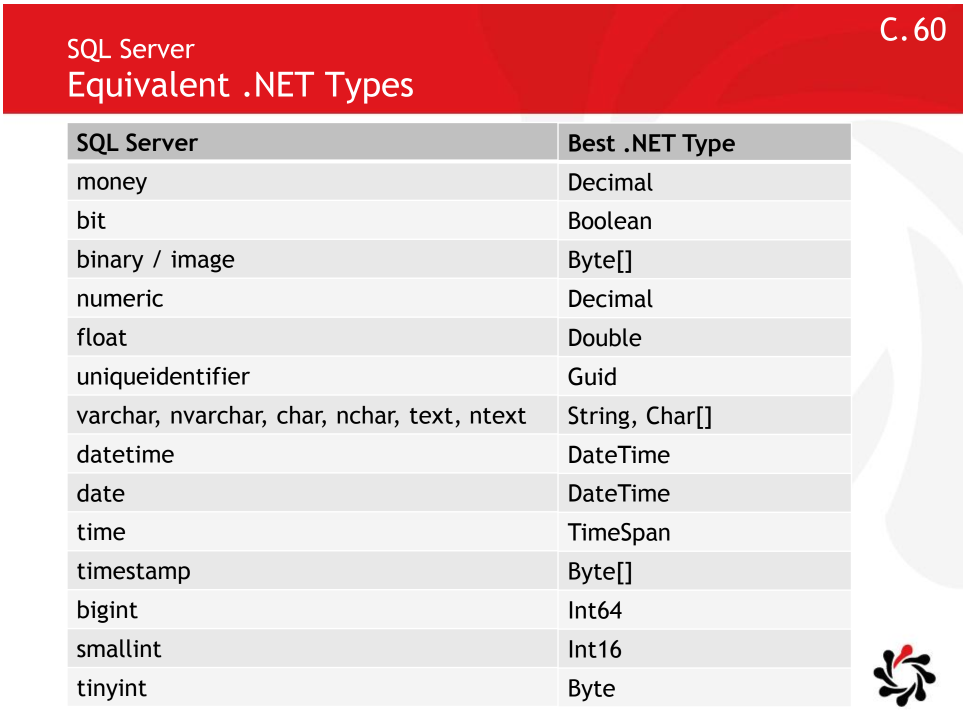


C.59

SQL Server

Partitioning

- Partitioning a database improves performance and simplifies maintenance
  - Hardware partitioning
    - RAID
  - Horizontal partitioning (SQL Server 7.0 and later)
    - Each table has same columns, but subset of rows
    - Excessive UNION queries, used to merge the tables logically at query time, can affect performance
  - Vertical partitioning (SQL Server 2005 and later)
    - Each logical row in a split table matches the same logical row in the other tables as identified by a UNIQUE KEY column that is identical in all of the partitioned tables



C.60

SQL Server

Equivalent .NET Types

SQL Server	Best .NET Type
money	Decimal
bit	Boolean
binary / image	Byte[]
numeric	Decimal
float	Double
uniqueidentifier	Guid
varchar, nvarchar, char, nchar, text, ntext	String, Char[]
datetime	DateTime
date	DateTime
time	TimeSpan
timestamp	Byte[]
bigint	Int64
smallint	Int16
tinyint	Byte

C.61

SQL Server

Custom CLR Types

- SQL Server 2005 and later supports the CLR and so can use any .NET type as a column or parameter type
  - Must use SqlCommand
  - Must set SqlParameter.SqlDbType to SqlDbType.Udt
  - Must set SqlParameter.UdtTypeName to the type name

```
paramLocation.SqlDbType = SqlDbType.Udt;
paramLocation.UdtTypeName = "Point";
```

  - Type name can be fully-qualified (as registered in database, NOT the .NET fully-qualified name)

```
paramLocation.UdtTypeName = "AdventureWorks.dbo.Point";
```

C.62

SQL Server

Spatial Datatypes

- SQL Server 2008 and later
  - geometry: planar (Euclidean) data
    - SqlDbType.Udt
    - UdtTypeName: "GEOMETRY"
  - geography: ellipsoidal data such as GPS coordinates
    - SqlDbType.Udt
    - UdtTypeName: "GEOGRAPHY"

C.63

SQL Server

Table Value Parameters

- SQL Server 2008 and later
  - A table type can be used as parameter to a stored procedure

```
CREATE TYPE dbo.Reviews AS TABLE
  ( ReviewID int, ReviewText nvarchar(50) )
```

```
CREATE PROCEDURE AddReviews
  (@reviews dbo.Reviews READONLY)
```

  - In .NET, pass a DataTable, IEnumerable<SqlDataRecord>, or DbDataReader

```
var cmd = new SqlCommand("AddReviews");
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.Add("reviews", SqlDbType.Structured);
cmd.Parameters["reviews"].Value = dataTableOfReviews;
```

Structured means table type

C.64

SQL Server

FILESTREAM in SQL Server 2008

```
CREATE TABLE employees ( ...,
  Photo VARBINARY(MAX) FILESTREAM  NULL, ...
```

```
command.CommandText = "select Photo.PathName(), " +
  "GET_FILESTREAM_TRANSACTION_CONTEXT() from employees";
using (SqlDataReader reader = command.ExecuteReader()) {
  while (reader.Read()) {
    string path = reader.GetString(0);
    SqlFileStream fileStream = new SqlFileStream(path,
      (byte[])reader.GetValue(1), FileAccess.Read,
      FileOptions.SequentialScan, 0);
    for (long index = 0; index < fileStream.Length; index++){
      Console.Write(fileStream.ReadByte());
    }
    fileStream.Close();
```

- Can also use GetSqlBytes() to store image in SqlBytes variable instead of reading from SqlFileStream

C.65

SQL Server

How to read SQL Server xml columns

- If you want to work with the content of the column as XML you must use an XmlReader

```
while (salesReaderData.Read()) // SqlDataReader
{
    SqlXml salesXML = salesReaderData.GetSqlXml(0);
    XmlReader salesReaderXml = salesXML.CreateReader();
    salesReaderXml.MoveToContent();
    while (salesReaderXml.Read())
    {
        if (salesReaderXml.NodeType == XmlNodeType.Element)
        {
            string elementLocalName = salesReaderXml.LocalName;
            salesReaderXml.Read();
            Console.WriteLine(elementLocalName + ": " +
                salesReaderXml.Value);
        }
    }
}
```

```
using System.Data.SqlTypes;
```

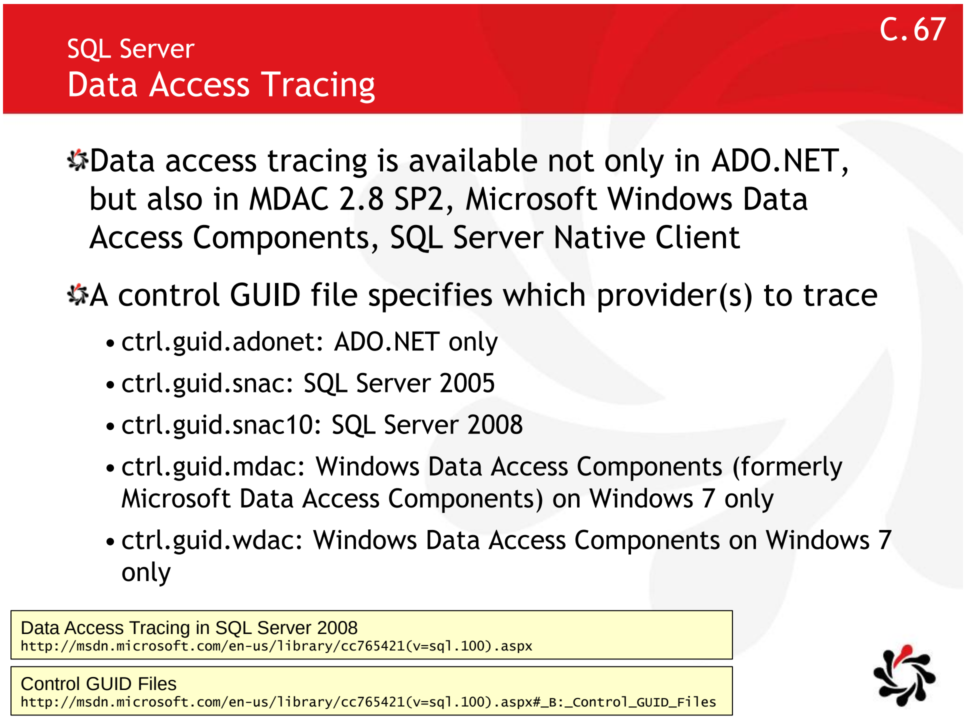
C.66

SQL Server

What Is Native Client?

- SQL Server Native Client
  - Is a stand-alone data access API used for both OLE DB and ODBC, that was introduced in SQL Server 2005
  - Provides new functionality above and beyond that supplied by the Windows Data Access Components (Windows DAC, formerly Microsoft Data Access Components, or MDAC)
  - Can be used to create new COM-based applications or enhance existing COM-based applications that need to take advantage of features introduced in SQL Server 2005, such as multiple active result sets (MARS), user-defined data types (UDT), query notifications, snapshot isolation, and XML data type support
  - If you are developing a managed application, you should use the .NET Data Provider for SQL Server, not the native client

SQL Server Native Client Programming
http://msdn.microsoft.com/en-us/library/ms130892



C.67

SQL Server

Data Access Tracing

- Data access tracing is available not only in ADO.NET, but also in MDAC 2.8 SP2, Microsoft Windows Data Access Components, SQL Server Native Client
- A control GUID file specifies which provider(s) to trace
  - ctrl.guid.adonet: ADO.NET only
  - ctrl.guid.snac: SQL Server 2005
  - ctrl.guid.snac10: SQL Server 2008
  - ctrl.guid.mdac: Windows Data Access Components (formerly Microsoft Data Access Components) on Windows 7 only
  - ctrl.guid.wdac: Windows Data Access Components on Windows 7 only

Data Access Tracing in SQL Server 2008
http://msdn.microsoft.com/en-us/library/cc765421(v=sql.100).aspx

Control GUID Files
http://msdn.microsoft.com/en-us/library/cc765421(v=sql.100).aspx#_B:_Control_GUID_Files

D. 1

Appendix D
LINQ

Developing Windows Azure and Web Services

Updated 11th April 2014

D. 2

LINQ

Contents

Topic	Slide
var	3
Generics	4
Initializers	6
Delegates	9
Lambda Expressions	11
LINQ	15
Extension Methods	22
Projection	33
Joining and Grouping	39
LINQ to XML	43
Tools for Learning	48

Exam Topic: Query data by using LINQ to Entities

❑ Query data by using LINQ operators (for example, project, skip, aggregate, filter, and join)

❑ Log queries

❑ Implement query boundaries (IQueryable vs. IEnumerable)

Exam Topic: Manipulate XML data structures

❑ LINQ to XML

D.3

var

Inferred and Anonymous Types

- var infers type of local variables *at compile time*

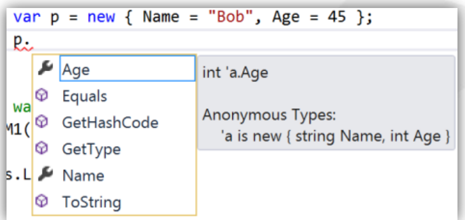
```
var name = "Mark";
```

```
var cost = 2.99M;
```

```
var width = GetWidth();
```

  - The compiler must be able to infer the type so you must assign an initial value, which can be returned from a method call

- Anonymous types can be inferred from an object initializer statement



```
var p = new { Name = "Bob", Age = 45 };
p.
```

  - Warning! Instances of anonymous types are immutable in C#

D.4

Generics

What Are They?

- Define a template for a strongly-typed class
  - Actual type is created at compile time
  - Improves performance and reduces runtime errors
  - Commonly used with collections

- Example with constraints

```
class Person : IComparable
```

```
class Employee : Person
```

```
public class Gen<TKey, TValue>
  where TKey : IComparable
  where TValue : Person, IComparable, new()
{
  public TKey Key;
  public TValue Value;
}
```

```
var ga = new Gen<int, Employee>();
```

D.5

Generics

Constraints

- When you define a generic class or method, you can apply restrictions to the kinds of types that can be used

Constraint	Description
where T : struct	The type argument must be a value type
where T : class	The type argument must be a reference type
where T : new()	The type argument must have a public parameterless constructor; ***must come last***
where T : <base class name>	The type argument must be or derive from the specified base class
where T : <interface name>	The type argument must be or implement the specified interface; multiple can be specified; can also be generic
where T : U	The type argument supplied for T must be or derive from the argument supplied for U

Constraints on Type Parameters (C# Programming Guide)
http://msdn.microsoft.com/en-us/library/d5x73970.aspx

D.6

Generics

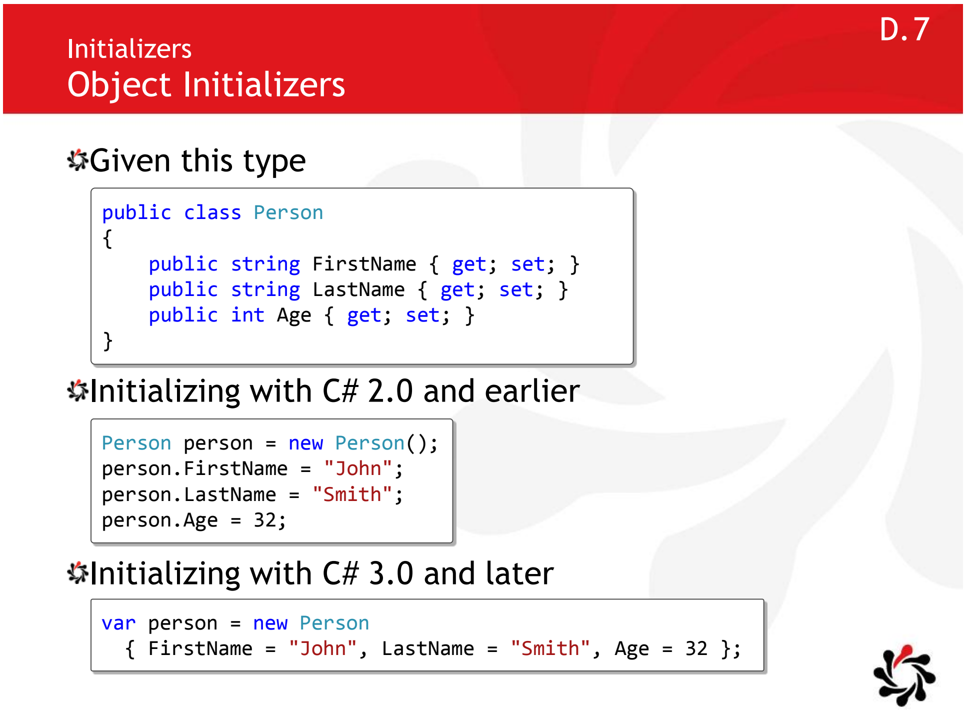
Generic Methods

- Any type (including non-generic types) can have generic methods
- The generic applies to the type in the method signature

```
public class NonGen
{
  public void M1<T>(T Value)
  {
    // use Value
  }
}
```

  - Specify the type when you call the method

```
var n = new NonGen();
n.M1<int>(23);
n.M1<string>("Fred");
```



D.7

Initializers

Object Initializers

- Given this type

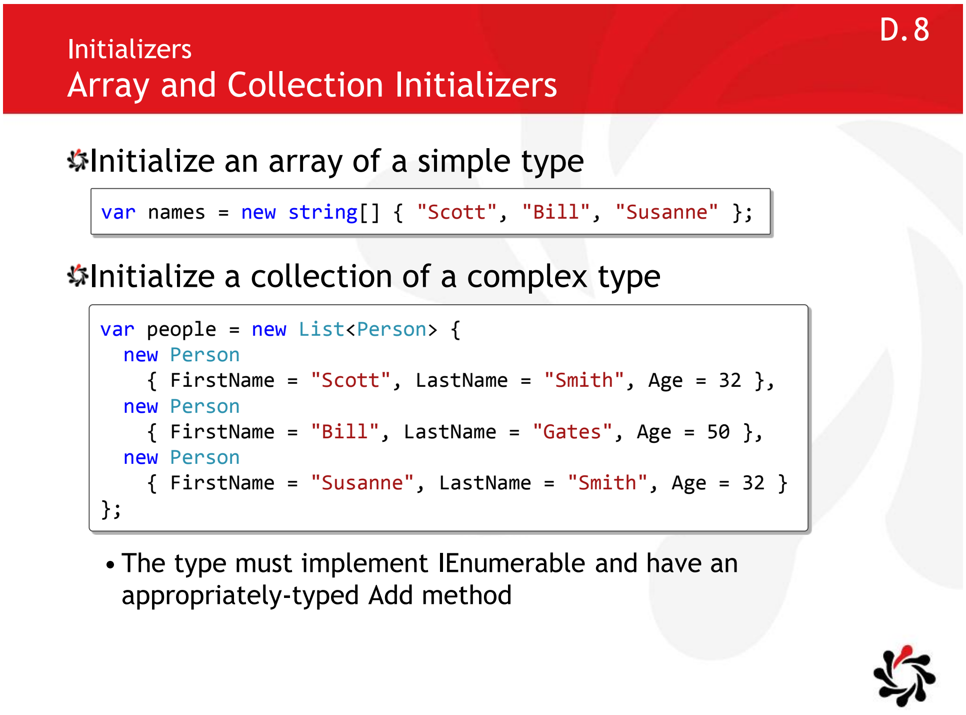
```
public class Person
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public int Age { get; set; }
}
```

- Initializing with C# 2.0 and earlier

```
Person person = new Person();
person.FirstName = "John";
person.LastName = "Smith";
person.Age = 32;
```

- Initializing with C# 3.0 and later

```
var person = new Person
  { FirstName = "John", LastName = "Smith", Age = 32 };
```



D.8

Initializers

Array and Collection Initializers

- Initialize an array of a simple type

```
var names = new string[] { "Scott", "Bill", "Susanne" };
```

- Initialize a collection of a complex type

```
var people = new List<Person> {
  new Person
    { FirstName = "Scott", LastName = "Smith", Age = 32 },
  new Person
    { FirstName = "Bill", LastName = "Gates", Age = 50 },
  new Person
    { FirstName = "Susanne", LastName = "Smith", Age = 32 }
};
```

  - The type must implement IEnumerable and have an appropriately-typed Add method

D.9

Delegates
What Is A Delegate?

- A delegate is a “type-safe function pointer”
  - Like a reference it contains the memory address of something, but it points to a method rather than a data structure
- A delegate must match the signature of the method you want to call

```
// method I want to call indirectly
int M1(string s)
{
    return s.Length;
}
```

```
// define with same signature
delegate int Del(string t);
```

```
// call method directly
int x = M1("Hello");
// call method indirectly
var d = new Del(M1);
int y = d("Goodbye");
```

MSDN Magazine Topics - Delegates
http://msdn.microsoft.com/en-us/magazine/ee532094.aspx?sdmr=delegates&sdmi=topics

D.10

Delegates
Why are Delegates Useful?

- Treat methods as data
  - For example, create a queue of methods to call
- Anonymous delegates
  - Simplify code by removing need for defining a private method

```
Button1.Click += delegate { Debug.Write("Clicked"); };
```

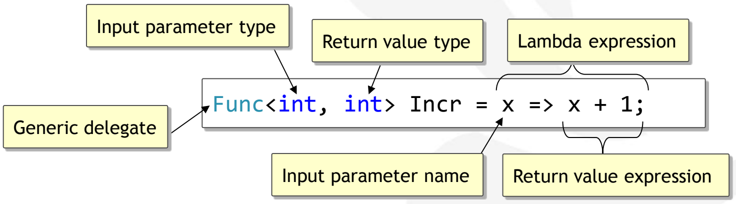
- Can be invoked asynchronously using BeginInvoke
- Lambda expressions (used in LINQ)
  - Lambda expressions can be used in place of a delegate instance
- Loose-binding of types; cleaner type design
  - Foundation of events

D.11

Lambda Expressions

What Are They?

- A lambda expression is simply a *nameless* function
  - Can be used wherever a delegate is valid



```
Func<int, int> Incr = x => x + 1;
```

- Note: Func is a generic delegate defined by Microsoft

D.12

Lambda Expressions

Syntax

- A lambda expression syntax

```
(parameters) => expression
```

```
parameter => expression
```

- Inferring input types from a delegate

```
delegate bool MyDelegate(int a, string b);
```

```
MyDelegate d = (x, y) => x == y.Length; // cannot use var with lambdas
```

- Explicitly defining input types

```
MyDelegate d = (int x, string y) => x == y.Length;
```

- Must use parentheses with zero input parameters; parentheses are only optional with one parameter

```
AnotherDelegate d2 = () => SomeMethod();
```

D.13

Lambda Expressions

Generic Delegates Often Used With Lambdas

- Func(TResult)
  - For lambda expressions with no inputs
- Func(T, TResult)
  - For lambda expressions with one input parameter

```
Func<int, bool> myFunc = x => x == 5;
bool result1 = myFunc(4); // returns false
bool result2 = myFunc(5); // returns true
```

- Func(T1, T2, TResult)
- Func(T1, T2, T3, TResult) and so on
- Predicate(T): one input and always returns a Boolean

```
Predicate<int> myFunc = x => x == 5;
```

D.14

Lambda Expressions

Lambda Statements and Multi-Line Expressions

- Lambda statements are nameless methods that return void
- Statement lambda syntax

```
(parameters) => { statements; }
```

  - Statement lambdas cannot be used in expression trees and therefore *cannot be used in LINQ queries*
- Generic delegates for use with lambda statements
  - Action(T), Action(T1, T2), and so on
- Lambda expressions can also have multiple statements, but must explicitly return a value

```
(input parameters) => { statements; return value; }
```

D.15

LINQ

What Is It?

- Most databases understand SQL...
  - ...but to C# 2.0 and VB 8.0, an SQL statement is just a string
  - LINQ integrates query syntax to a .NET language
- LINQ is made up of three parts
  - Providers for data sources (required)
    - LINQ to Objects, LINQ to SQL, LINQ to Entities, LINQ to XML, LINQ to SharePoint, LINQ to Amazon, and so on
  - Extensions to the base class libraries (required)
    - System.Linq.Enumerable and System.Linq.Queryable classes in System.Core.dll assembly
  - Extensions to the languages and compilers (optional)
    - C# keywords: from, select, orderby, and so on
    - VB keywords: From, Select, Order By, and so on

D.16

LINQ

Provider Limitations

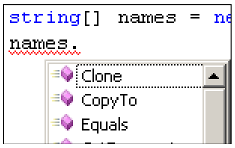
- Theoretically, once you learn LINQ, you can query any LINQ provider...
  - ...but some LINQ providers have limitations
- LINQ to Entities limitations
  - This LINQ provider must eventually convert the expression tree created by your LINQ statements into Transact-SQL statements, so not all LINQ statements are fully supported, or might be implemented in ways that you do not like
  - Aggregate, SkipWhile, ToString, Zip, and others not supported
- LINQ to Objects supports all features
  - So use To*Something*<T> methods to get data from LINQ to Entities and then use LINQ to Objects on the result

LINQ
Enumerable and Queryable classes (System.Linq)

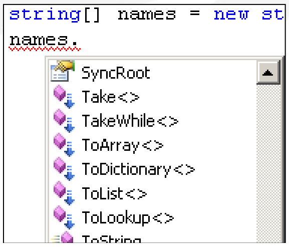
D.17

- LINQ requires types to implement interfaces to support it's features: IEnumerable(T) or IQueryable(T)
  - If a type does not, IEnumerable has some extension methods: OfType(T), Cast(T) that can convert to the generic versions
- LINQ uses extension methods defined by Enumerable and Queryable classes in the System.Linq namespace
  - Importing the namespace allows the extension methods to be used on any type that implements IEnumerable(T) or IQueryable(T)

```
using System.Linq;
```



```
string[] names = ne
names.
```



```
string[] names = new st
names.
```

LINQ
IEnumerable and IQueryable interfaces

D.18

- IEnumerable means LINQ to Objects
  - All data must be materialized locally before extension methods are applied
- IQueryable means LINQ to Entities (or LINQ to SQL)
  - Data retrieval is deferred
  - An expression tree is created and only when the query is enumerated (with foreach) or one of the ToXxx methods is called will the expression tree be converted into an SQL statement and executed to retrieve the appropriate data
  - Use ToString (with DbContext) or ToTraceString (with ObjectContext) to see the T-SQL that will be executed
  - Use a To*Something* method to retrieve data, then you can use LINQ to Objects without limitations

D.19

LINQ

How to Create Extension Methods

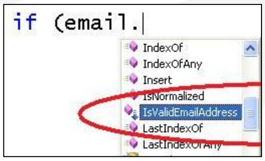
- Extension methods allow you to add methods to a type without inheritance or recompilation
  - Create a static class with a static method that uses the `this` keyword before a parameter to specify the type to extend

```
namespace MyExtensions {
  public static class StringExtensions {
    public static bool IsValidEmailAddress(this string s) {
```

  - Import the namespace containing the class

```
using MyExtensions;
```

  - Use the extension method on the type you extended



```
if (email.
```

D.20

LINQ

To create extension methods for LINQ

- Create a static class that extends IEnumerable<T>
  - Returns a scalar (double)

```
namespace System.Linq
{
  public static class MyLinqExtensions
  {
    public static double MyAggregate<T>(this IEnumerable<T> input)
```

  - Returns a sequence (IEnumerable<T>) and therefore can be chained

```
public static IEnumerable<T> MyProcessor<T>(this IEnumerable<T> input)
```

D.21

LINQ
Language-Level Support (Syntactic Sugar)

- Instead of using extension methods and lambda expressions, C# and Visual Basic provide simplified syntax for queries like this

```
var query = from name in names
            where name.StartsWith("A")
            orderby name
            select name; // select is required
```

- The compiler turns the syntactic sugar above into calls to the extension methods with lambdas

Lambda expression

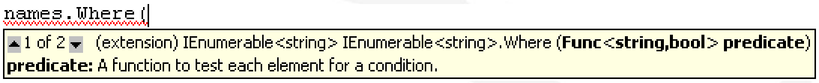
```
var query = names.Where(name => name.StartsWith("A"))
                 .OrderBy(name => name);
```

```
// Select is optional
.Select(name => name)
```

D.22

LINQ Extension Methods
Where

- Where takes a Func<TInput,TReturn> generic delegate as input (so we can use a lambda expression instead)
  - Expression must have a single input parameter (of whatever type T in IEnumerable(T) is) and must return a boolean



```
names.Where(
```

▲1 of 2▼ (extension) IEnumerable<string> IEnumerable<string>.Where (**Func<string,bool> predicate**)
predicate: A function to test each element for a condition.

  - n would be a string

```
string[] names = ...;
... names.Where(n => n.StartsWith("A"));
```

  - p would be a Person

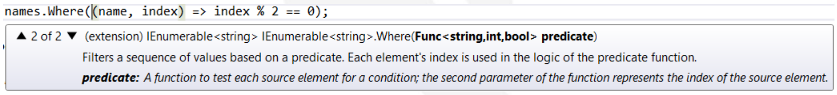
```
Person[] people = ...;
... people.Where(p => p.Age > 18);
```

LINQ Extension Methods

Where and Select Method Index Values

D.23

- Where (and Select) take Func<TInput,int,TReturn> delegate as input so you can filter based on index



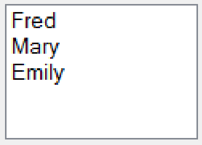
```
names.Where((name, index) => index % 2 == 0);
```

▲ 2 of 2 ▼ (extension) IEnumerable<string> IEnumerable<string>.Where(**Func<string,int,bool> predicate**)

Filters a sequence of values based on a predicate. Each element's index is used in the logic of the predicate function.

predicate: *A function to test each source element for a condition; the second parameter of the function represents the index of the source element.*

```
var names = new string[]
  { "Fred", "George", "Mary", "Sally", "Emily", "Harry" };
var query = names.Where((name, index) => index % 2 == 0);
var results = query.ToArray();
list.ItemsSource = results; // WPF ListBox
```



```
Fred
Mary
Emily
```

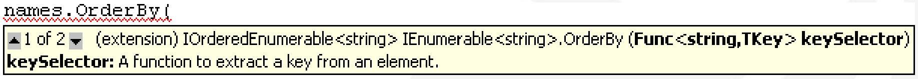
- Syntactic sugar does not seem to have an equivalent

LINQ Extension Methods

OrderBy

D.24

- OrderBy takes a Func<TInput,TKeySelector> generic delegate as input
  - Lambda expression must have a single input parameter (of whatever type T in IEnumerable(T) is) and can return any type
  - For names string array, we might want to order by the number of characters in each string entry



```
names.OrderBy(
```

▲1 of 2 ▼ (extension) IOrderedEnumerable<string> IEnumerable<string>.OrderBy (**Func<string,TKey> keySelector**)
keySelector: A function to extract a key from an element.

```
... names.OrderBy(name => name.Length);
```

  - For names string array, we might want to order by the entire entry (which looks strange but is necessary due to the syntax!)

```
... names.OrderBy(name => name);
```

D.25

LINQ Extension Methods

OrderBy and GroupBy

- Usually you want to order first, then group

```
var names = new string[]
  { "Fred", "George", "Gary", "Emily" };
var query = names.OrderBy(name => name)
                 .GroupBy(name => name[0]);
```

```
E
  Emily
F
  Fred
G
  Gary
  George
```

- If you group first, you need to specify how to order the *group's key*, not the original items

```
var names = new string[]
  { "Fred", "George", "Gary", "Emily" };
var query = names.GroupBy(name => name[0])
                 .OrderBy(group => group.Key);
```

```
E
  Emily
F
  Fred
G
  George
  Gary
```

D.26

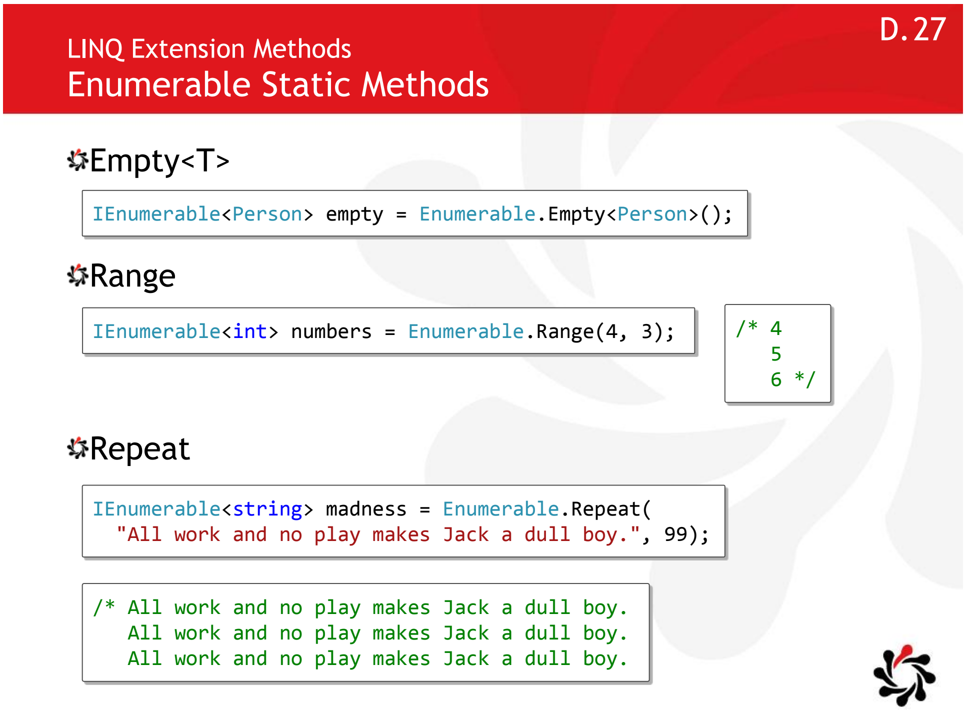
LINQ Extension Methods

Chaining, Deferred Execution, and Materialization

- Most extension methods return IEnumerable(T) so that they can be chained

```
... names.Where(name => name.StartsWith("A"))
         .OrderBy(name => name);
```

  - The extension methods will be processed in order
  - Methods that return a sequence of values do not consume the target data until the query is enumerated (deferred execution)
  - The query is not executed until enumerated over and will re-execute over the original data each time so will detect changes
  - Materialize a copy with ToArray, ToList, and so on
  - Methods that return a singleton value execute and consume the target data immediately
  - Do not assume the entries in a sequence are ordered unless you explicitly specify



D.27

LINQ Extension Methods

Enumerable Static Methods

- Empty<T>

```
IEnumerable<Person> empty = Enumerable.Empty<Person>();
```

- Range

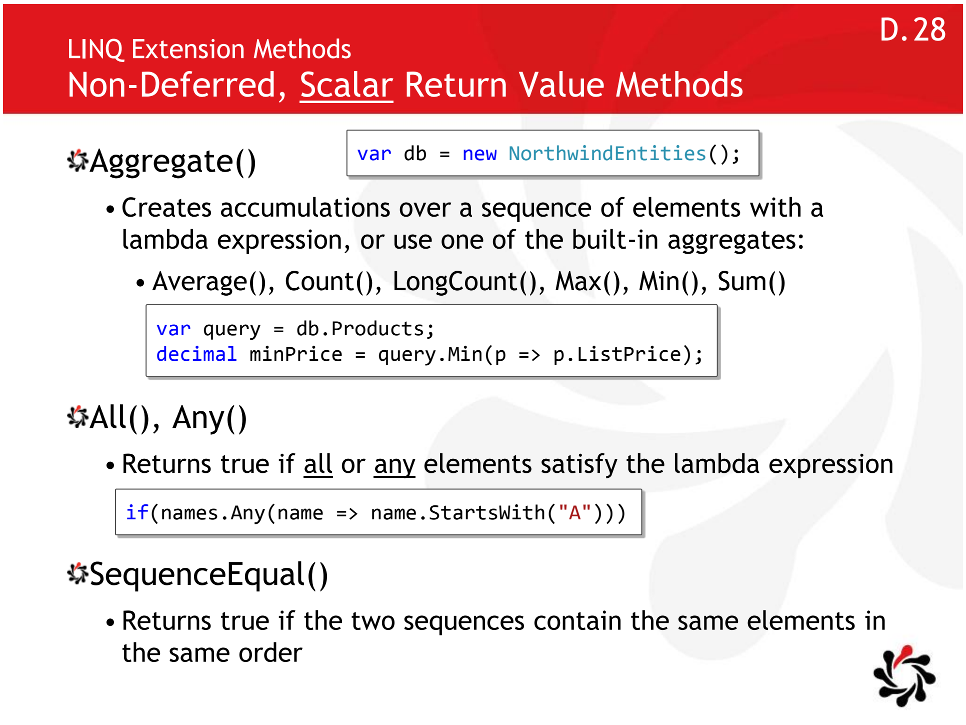
```
IEnumerable<int> numbers = Enumerable.Range(4, 3);
```

```
/* 4
   5
   6 */
```

- Repeat

```
IEnumerable<string> madness = Enumerable.Repeat(
  "All work and no play makes Jack a dull boy.", 99);
```

```
/* All work and no play makes Jack a dull boy.
   All work and no play makes Jack a dull boy.
   All work and no play makes Jack a dull boy.
```



D.28

LINQ Extension Methods

Non-Deferred, Scalar Return Value Methods

- Aggregate()

```
var db = new NorthwindEntities();
```

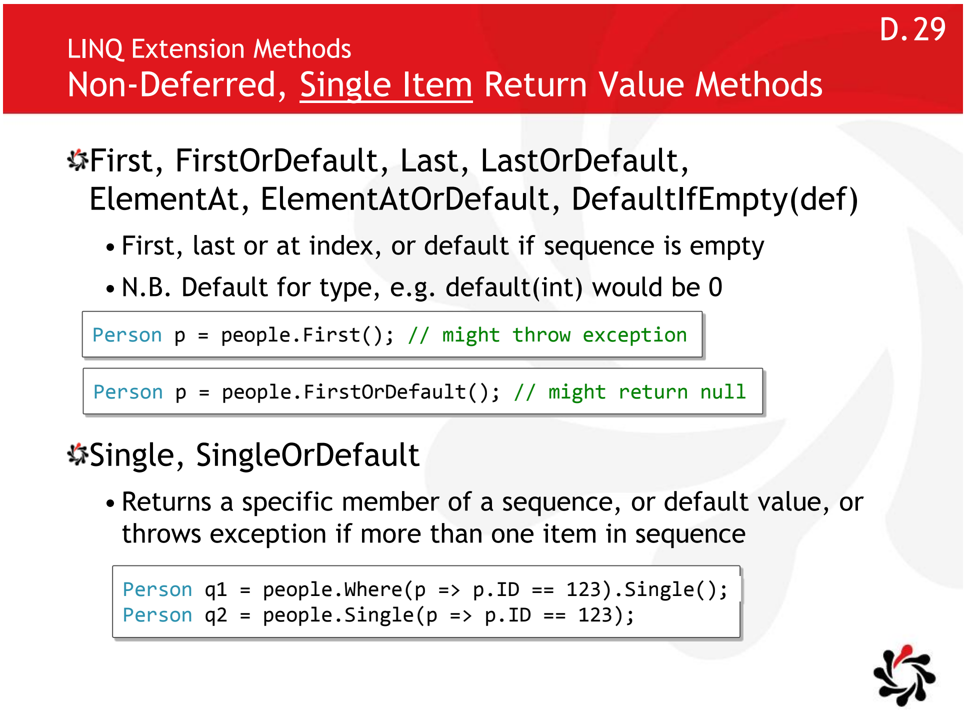
  - Creates accumulations over a sequence of elements with a lambda expression, or use one of the built-in aggregates:
    - Average(), Count(), LongCount(), Max(), Min(), Sum()

```
var query = db.Products;
decimal minPrice = query.Min(p => p.ListPrice);
```

- All(), Any()
  - Returns true if all or any elements satisfy the lambda expression

```
if(names.Any(name => name.StartsWith("A")))
```

- SequenceEqual()
  - Returns true if the two sequences contain the same elements in the same order



D.29

LINQ Extension Methods

Non-Deferred, Single Item Return Value Methods

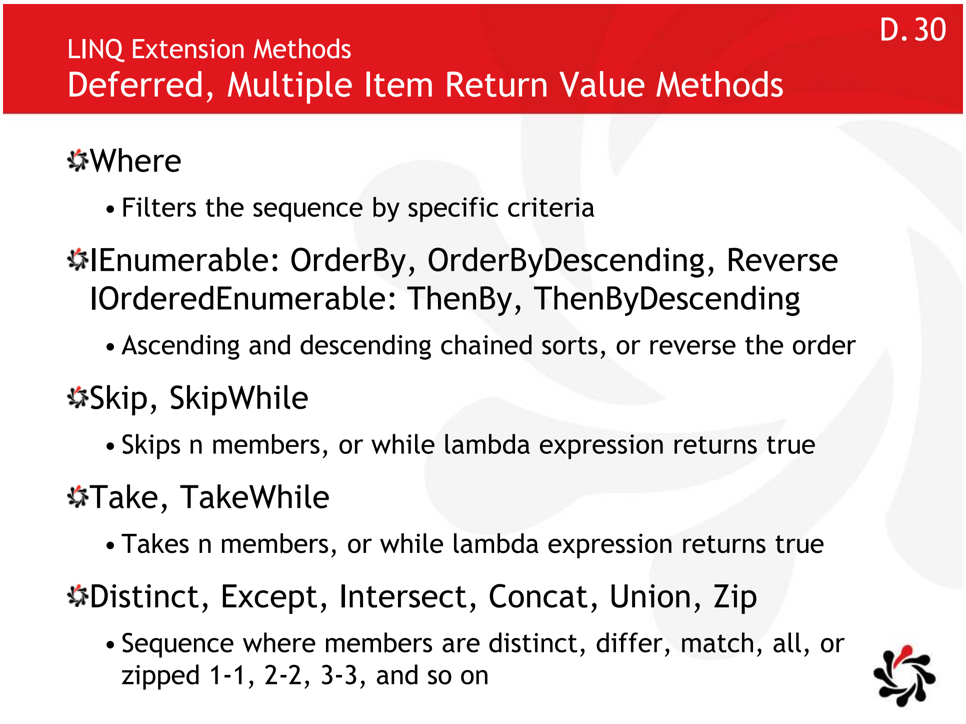
- First, FirstOrDefault, Last, LastOrDefault, ElementAt, ElementAtOrDefault, DefaultIfEmpty(def)
  - First, last or at index, or default if sequence is empty
  - N.B. Default for type, e.g. default(int) would be 0

```
Person p = people.First(); // might throw exception
```

```
Person p = people.FirstOrDefault(); // might return null
```

- Single, SingleOrDefault
  - Returns a specific member of a sequence, or default value, or throws exception if more than one item in sequence

```
Person q1 = people.Where(p => p.ID == 123).Single();
Person q2 = people.Single(p => p.ID == 123);
```

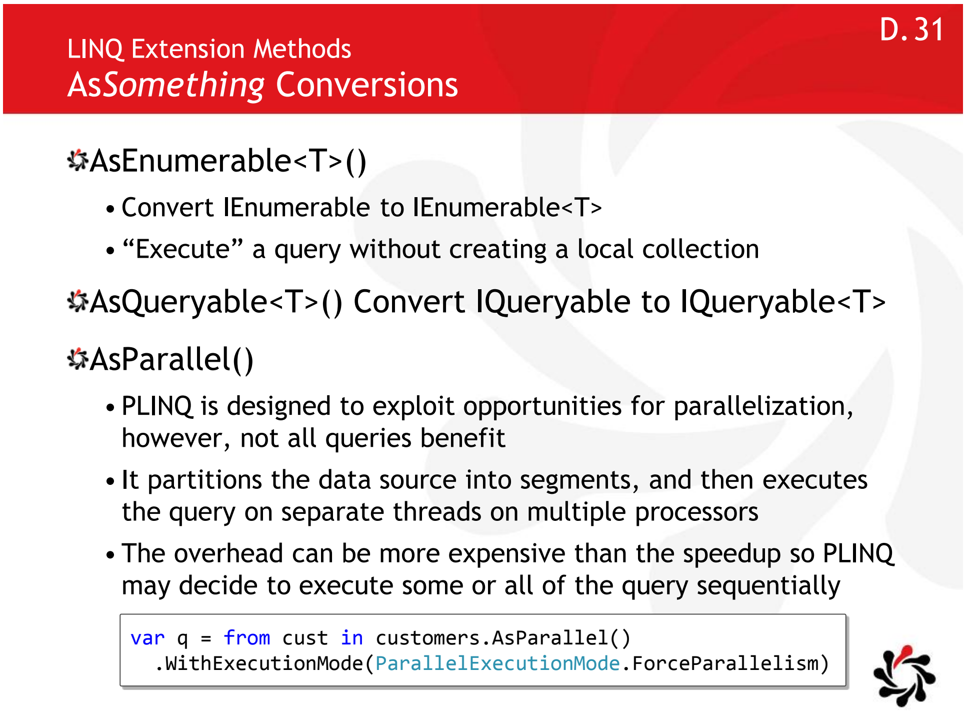


D.30

LINQ Extension Methods

Deferred, Multiple Item Return Value Methods

- Where
  - Filters the sequence by specific criteria
- IEnumerable: OrderBy, OrderByDescending, Reverse IOrderedEnumerable: ThenBy, ThenByDescending
  - Ascending and descending chained sorts, or reverse the order
- Skip, SkipWhile
  - Skips n members, or while lambda expression returns true
- Take, TakeWhile
  - Takes n members, or while lambda expression returns true
- Distinct, Except, Intersect, Concat, Union, Zip
  - Sequence where members are distinct, differ, match, all, or zipped 1-1, 2-2, 3-3, and so on



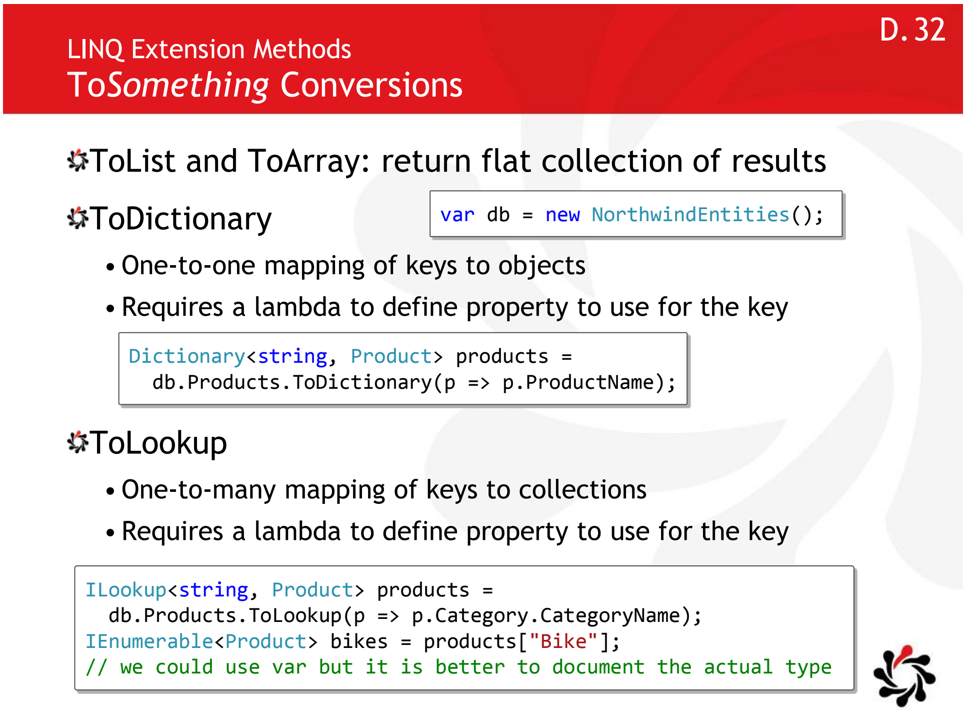
D.31

LINQ Extension Methods

As*Something* Conversions

- AsEnumerable<T>()
  - Convert IEnumerable to IEnumerable<T>
  - “Execute” a query without creating a local collection
- AsQueryable<T>() Convert IQueryable to IQueryable<T>
- AsParallel()
  - PLINQ is designed to exploit opportunities for parallelization, however, not all queries benefit
  - It partitions the data source into segments, and then executes the query on separate threads on multiple processors
  - The overhead can be more expensive than the speedup so PLINQ may decide to execute some or all of the query sequentially

```
var q = from cust in customers.AsParallel()
  .WithExecutionMode(ParallelExecutionMode.ForceParallelism)
```



D.32

LINQ Extension Methods

To*Something* Conversions

- ToList and ToArray: return flat collection of results
- ToDictionary

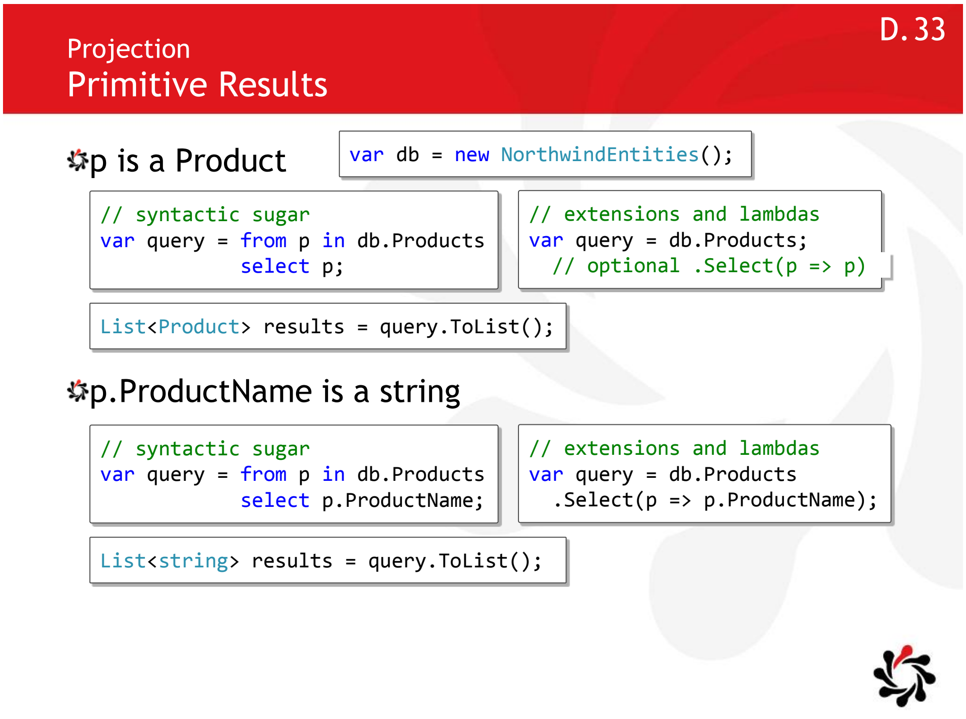
```
var db = new NorthwindEntities();
```

  - One-to-one mapping of keys to objects
  - Requires a lambda to define property to use for the key

```
Dictionary<string, Product> products =
  db.Products.ToDictionary(p => p.ProductName);
```

- ToLookup
  - One-to-many mapping of keys to collections
  - Requires a lambda to define property to use for the key

```
ILookup<string, Product> products =
  db.Products.ToLookup(p => p.Category.CategoryName);
IEnumerable<Product> bikes = products["Bike"];
// we could use var but it is better to document the actual type
```



D.33

Projection
Primitive Results

- p is a Product

```
var db = new NorthwindEntities();
```

```
// syntactic sugar
var query = from p in db.Products
            select p;
```

```
// extensions and lambdas
var query = db.Products;
  // optional .Select(p => p)
```

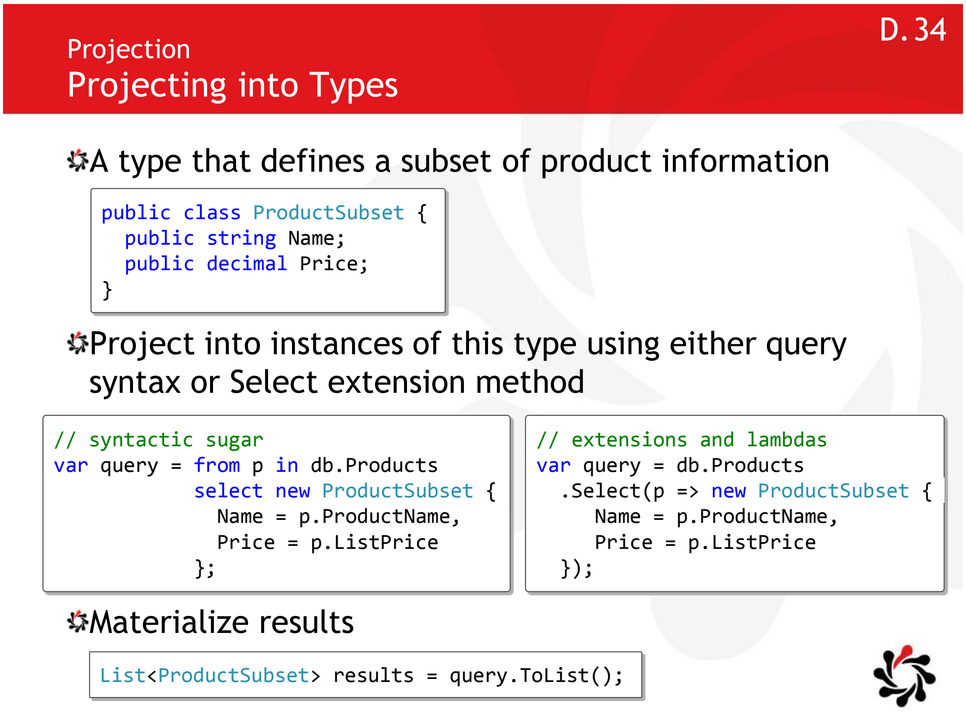
```
List<Product> results = query.ToList();
```

- p.ProductName is a string

```
// syntactic sugar
var query = from p in db.Products
            select p.ProductName;
```

```
// extensions and lambdas
var query = db.Products
  .Select(p => p.ProductName);
```

```
List<string> results = query.ToList();
```



D.34

Projection
Projecting into Types

- A type that defines a subset of product information

```
public class ProductSubset {
  public string Name;
  public decimal Price;
}
```

- Project into instances of this type using either query syntax or Select extension method

```
// syntactic sugar
var query = from p in db.Products
            select new ProductSubset {
              Name = p.ProductName,
              Price = p.ListPrice
            };
```

```
// extensions and lambdas
var query = db.Products
  .Select(p => new ProductSubset {
     Name = p.ProductName,
     Price = p.ListPrice
  });
```

- Materialize results

```
List<ProductSubset> results = query.ToList();
```

D.35

Projection

Projecting into Anonymous Types

- Project into instances of an anonymous type using either query syntax or Select extension method
  - Note: you can assign a new name for the property e.g. Name or reuse its existing name e.g. ListPrice

```
// syntactic sugar
var query = from p in db.Products
            select new {
              Name = p.ProductName,
              p.ListPrice
            };
```

```
// extensions and lambdas
var query = db.Products
        .Select(p => new {
          Name = p.ProductName,
          p.ListPrice
        });
```

- Materialize results and store in inferred variable

```
// must use var
var results = query.ToList();
```

D.36

Projection

SelectMany Example 1

- SelectMany projects each element of a sequence to an IEnumerable<T> and flattens the resulting sequences into one sequence

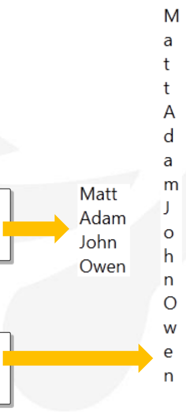
```
var nameList = new List<string> {
  "Matt", "Adam", "John", "Peter",
  "Owen", "Steve", "Richard", "Chris" };
```

- Select the names of length four

```
var names1 = nameList.Where(n => n.Length == 4)
                     .Select(n => n);
```

- SelectMany the names of length four

```
var names2 = nameList.Where(n => n.Length == 4)
                     .SelectMany(n => n);
```



```
Matt
Adam
John
Owen
```

```
M
a
t
t
A
d
a
m
J
o
h
n
O
w
e
n
```

Projection

SelectMany Example 2

D.37

- We want to create a single sequence of words from a sequence of sentences

```
var sentences = new List<string> {
  "Bob is quite excited.",
  "Jim is very upset."
};
```

- Using SelectMany

```
// extensions and lambdas
var words = sentences
  .SelectMany(s => s.TrimEnd('.').Split(' '));
```

- Using "from chaining"

```
// syntactic sugar
var words = from s in sentences
            from w in s.TrimEnd('.').Split(' ')
            select w;
```

```
Bob
is
quite
excited
Jim
is
very
upset
```

Projection

SelectMany Example 3

D.38

- We want to get a flat list of products from categories

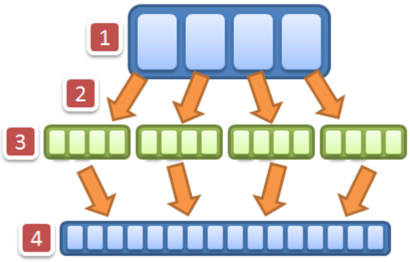
```
var db = new NorthwindEntities();
```

- Using SelectMany

```
var query = db.Categories.SelectMany(c => c.Products);
```

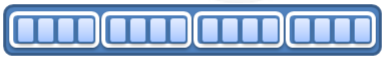
- Using LINQ 'from' chaining

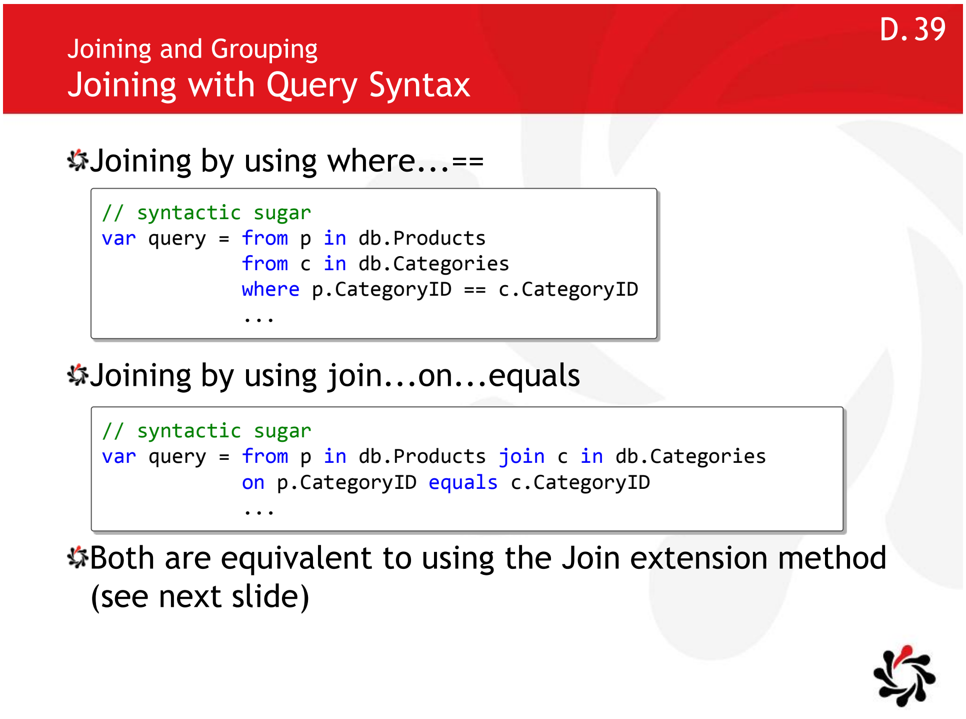
```
// syntactic sugar
var query = from c in db.Categories
            from p in c.Products
            select p;
```



```
... = from c in db.Categories
      select c.Products;
```

Select would give us this:





Joining and Grouping

Joining with Query Syntax

D.39

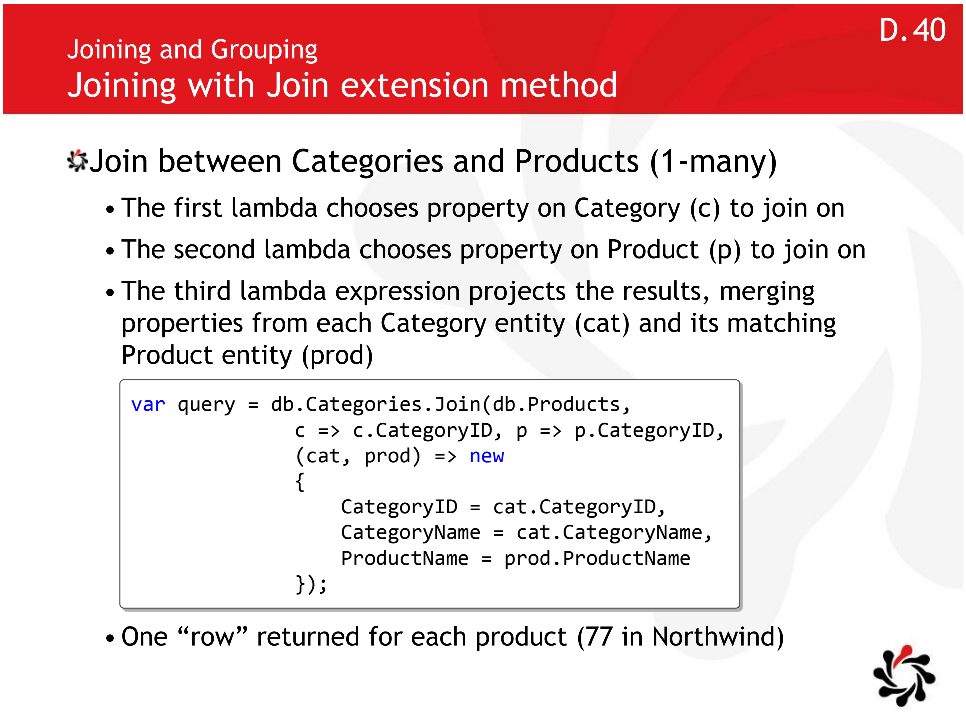
- Joining by using where...==

```
// syntactic sugar
var query = from p in db.Products
            from c in db.Categories
            where p.CategoryID == c.CategoryID
            ...
```

- Joining by using join...on...equals

```
// syntactic sugar
var query = from p in db.Products join c in db.Categories
            on p.CategoryID equals c.CategoryID
            ...
```

- Both are equivalent to using the Join extension method (see next slide)



Joining and Grouping

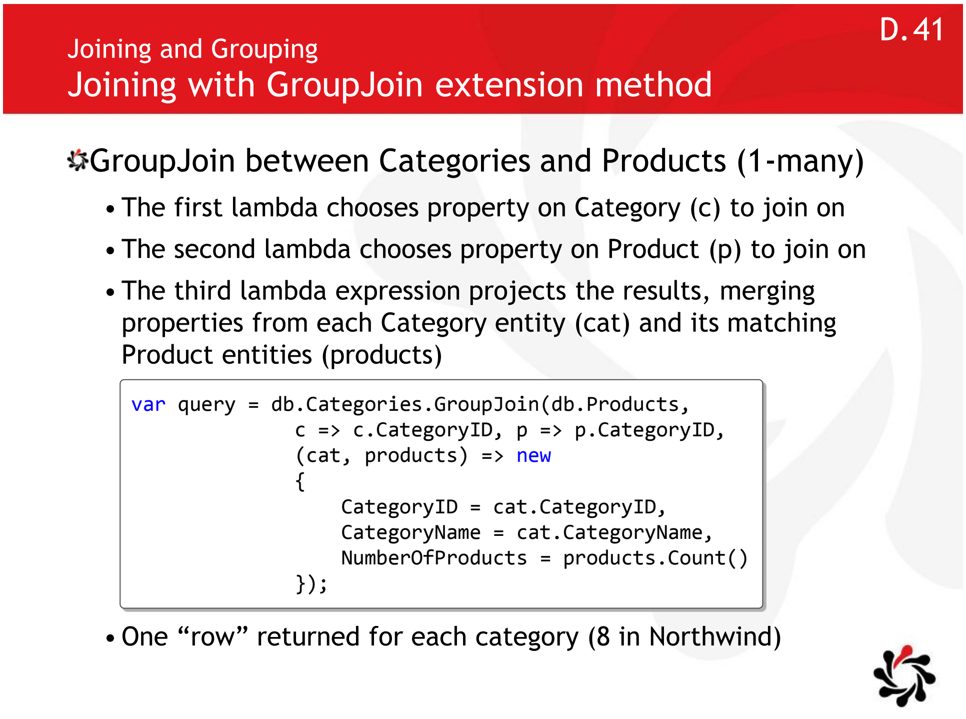
Joining with Join extension method

D.40

- Join between Categories and Products (1-many)
  - The first lambda chooses property on Category (c) to join on
  - The second lambda chooses property on Product (p) to join on
  - The third lambda expression projects the results, merging properties from each Category entity (cat) and its matching Product entity (prod)

```
var query = db.Categories.Join(db.Products,
              c => c.CategoryID, p => p.CategoryID,
              (cat, prod) => new
              {
                  CategoryID = cat.CategoryID,
                  CategoryName = cat.CategoryName,
                  ProductName = prod.ProductName
              });
```

  - One “row” returned for each product (77 in Northwind)



D.41

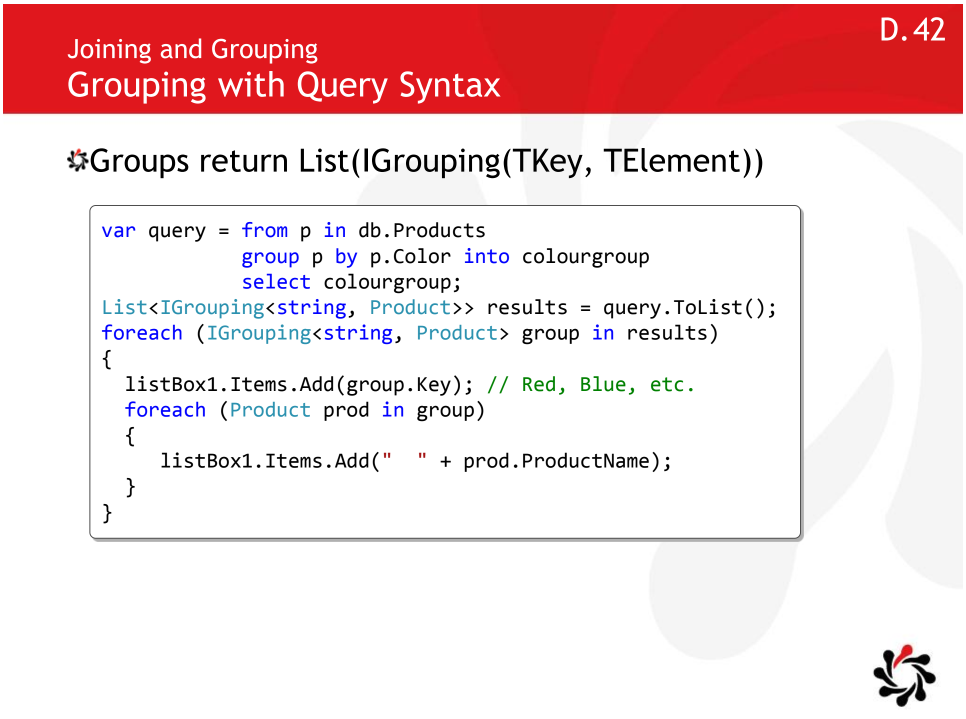
Joining and Grouping

Joining with GroupJoin extension method

- GroupJoin between Categories and Products (1-many)
  - The first lambda chooses property on Category (c) to join on
  - The second lambda chooses property on Product (p) to join on
  - The third lambda expression projects the results, merging properties from each Category entity (cat) and its matching Product entities (products)

```
var query = db.Categories.GroupJoin(db.Products,
              c => c.CategoryID, p => p.CategoryID,
              (cat, products) => new
              {
                  CategoryID = cat.CategoryID,
                  CategoryName = cat.CategoryName,
                  NumberOfProducts = products.Count()
              });
```

  - One “row” returned for each category (8 in Northwind)



D.42

Joining and Grouping

Grouping with Query Syntax

- Groups return List(IGrouping(TKey, TElement))

```
var query = from p in db.Products
            group p by p.Color into colourgroup
            select colourgroup;
List<IGrouping<string, Product>> results = query.ToList();
foreach (IGrouping<string, Product> group in results)
{
  listBox1.Items.Add(group.Key); // Red, Blue, etc.
  foreach (Product prod in group)
  {
     listBox1.Items.Add("   " + prod.ProductName);
  }
}
```

D.43

LINQ to XML

Generating an XML File from LINQ-able Entities

- “products” could be an entity set or collection

```
var xml = new XElement("Products",
                from p in products
                select new XElement("Product",
                    new XElement("ProductID", p.ProductID),
                    new XElement("Name", p.Name),
                    new XElement("Color", p.Color),
                    new XElement("ListPrice", p.ListPrice),
                    new XElement("Size", p.Size)));
xml.Save(productFileName);
```

```
using System.Xml.Linq;
```

```
<Products>
  <Product>
    <ProductID>1</ProductID>
    <Name>Chai</Name>
    ...
  </Product>
  <Product
    ...
  </Product>
</Products>
```

- Use XAttribute for attributes

D.44

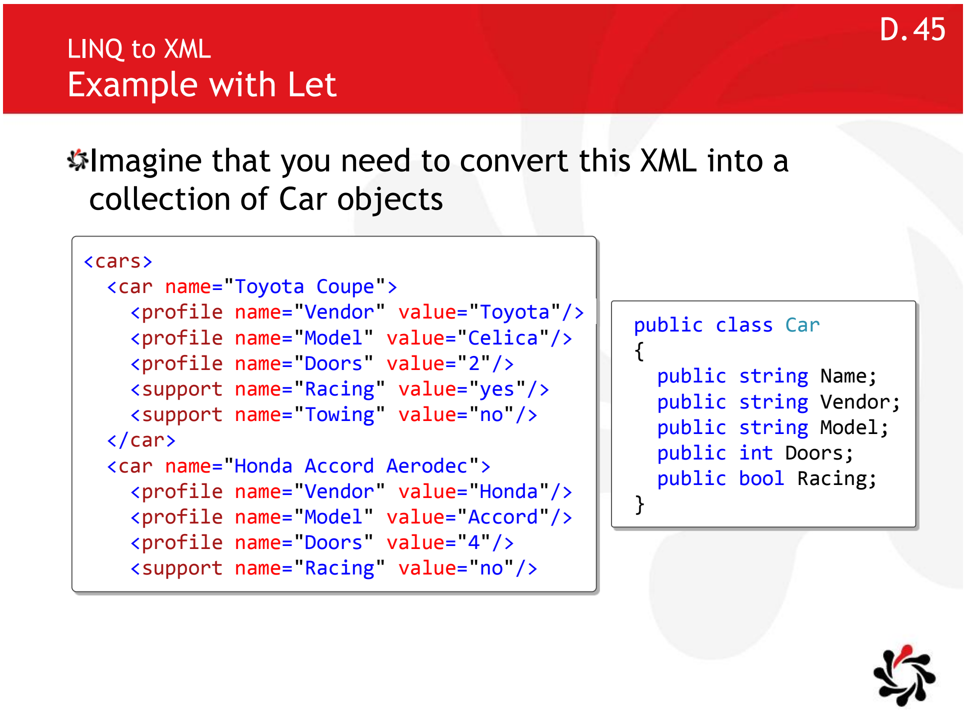
LINQ to XML

Generating a Collection from an XML File

- Convert each child XML element into an entity

```
using System.Xml.Linq;
```

```
var doc = XDocument.Load(productFileName);
var query = from product in doc.Descendants("Product")
            select new Product
            {
              ProductID = (int)product.Element("ProductID").Value,
              Name = product.Element("Name").Value,
              Color = product.Element("Color").Value,
              ListPrice = product.Element("ListPrice").Value,
              Size = product.Element("Size").Value
            };
List<Product> products = query.ToList<Product>();
```



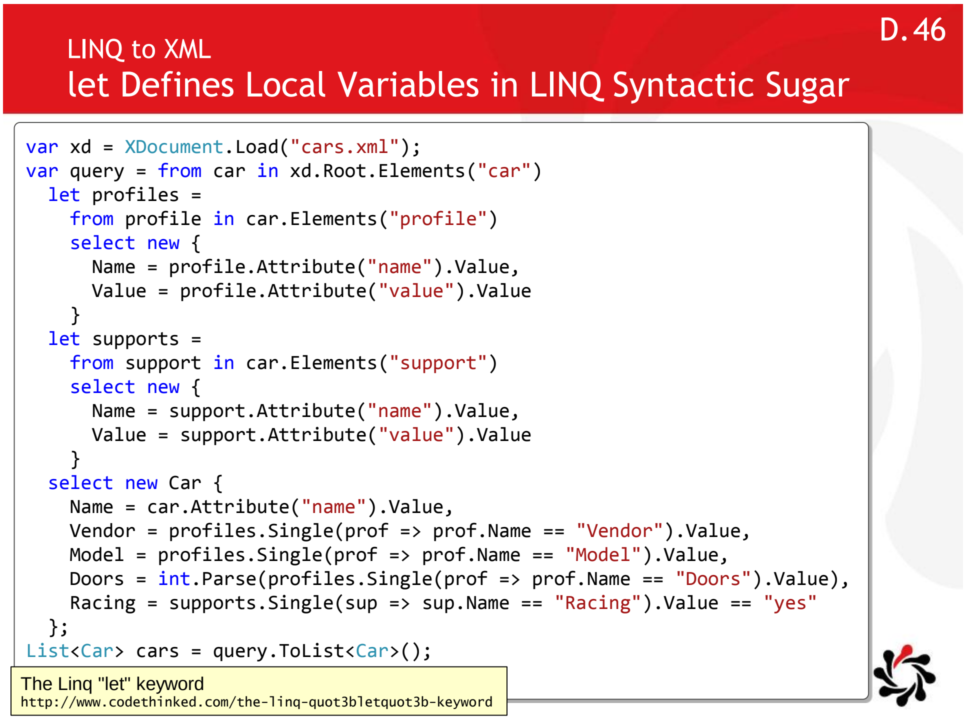
D.45

LINQ to XML
Example with Let

- Imagine that you need to convert this XML into a collection of Car objects

```
<cars>
  <car name="Toyota Coupe">
    <profile name="Vendor" value="Toyota"/>
    <profile name="Model" value="Celica"/>
    <profile name="Doors" value="2"/>
    <support name="Racing" value="yes"/>
    <support name="Towing" value="no"/>
  </car>
  <car name="Honda Accord Aerodec">
    <profile name="Vendor" value="Honda"/>
    <profile name="Model" value="Accord"/>
    <profile name="Doors" value="4"/>
    <support name="Racing" value="no"/>
```

```
public class Car
{
  public string Name;
  public string Vendor;
  public string Model;
  public int Doors;
  public bool Racing;
}
```



D.46

LINQ to XML
let Defines Local Variables in LINQ Syntactic Sugar

```
var xd = XDocument.Load("cars.xml");
var query = from car in xd.Root.Elements("car")
  let profiles =
    from profile in car.Elements("profile")
    select new {
      Name = profile.Attribute("name").Value,
      Value = profile.Attribute("value").Value
    }
  let supports =
    from support in car.Elements("support")
    select new {
      Name = support.Attribute("name").Value,
      Value = support.Attribute("value").Value
    }
  select new Car {
    Name = car.Attribute("name").Value,
    Vendor = profiles.Single(prof => prof.Name == "Vendor").Value,
    Model = profiles.Single(prof => prof.Name == "Model").Value,
    Doors = int.Parse(profiles.Single(prof => prof.Name == "Doors").Value),
    Racing = supports.Single(sup => sup.Name == "Racing").Value == "yes"
  };
List<Car> cars = query.ToList<Car>();
```

The Linq "let" keyword
http://www.codethinked.com/the-linq-quot3bletquot3b-keyword

Tools for Learning
LINQPad 4 and 101 LINQ Samples

- LINQPad
  - Interactively query databases using LINQ
  - 500 examples

LINQPad
`http://www.linqpad.net`

101 LINQ Samples – C#
`http://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b`

Lambda Expressions (C#)
`http://msdn.microsoft.com/en-us/library/bb397687.aspx`

LINQ Query Expressions (C#)
`http://msdn.microsoft.com/en-us/library/bb397676.aspx`

Appendix E
Entity Framework 4

Developing Windows Azure and Web Services

Updated 11th April 2014

Entity Framework 4

Contents

Topic	Slide
Entity Data Models	3
Designer Generated Code	7
POCO	9
Retrieving Data	12
Modifying Data	16
Performance	22

You are unlikely to get questions about features of EF4 and ObjectContext in the exam although Visual Studio can generate the old ObjectContext and EntityObject classes or the inner ObjectContext can be accessed by casting a DbContext as an IObjectContextAdapter like this:

```
using System.Data.Entity.Infrastructure;
```

```
var db = new NorthwindEntities(); // DbContext
var oc = ((IObjectContextAdapter)db).ObjectContext;
```

E.3

Entity Data Models

Vocabulary Overview

	AWModel.edmx	AWModel.edmx.cs
“Database”	<**EntityContainer** Name="AdventureWorksEntities"	class AdventureWorksEntities : **ObjectContext** (or **DbContext**)
“Table”	<**EntitySet** Name="Contacts"	**ObjectSet<Contact**> Contacts (or **DbSet<Contact>** Contacts)
“Row”	<**EntityType** Name="Contact"	class Contact : **EntityObject** (or class Contact)
“Column”	<**Property** Name="ContactID"	[EdmScalarProperty(…)] Int32 ContactID …
“Relationship”	<**NavigationProperty** Name="Orders" … <**NavigationProperty** Name="Contact" … <**Association** Name= "FK_Orders_Contact_ContactID"	[EdmRelationshipNavigationProperty(…)] EntityCollection<Order> Orders [EdmRelationshipNavigationProperty(…)] Contact Contact

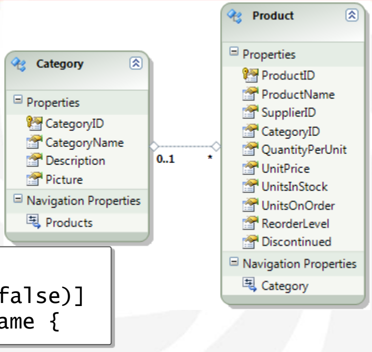
VS2010 .edmx designer-generated code creates classes that use **ObjectContext**
VS2012 .edmx designer-generated code creates classes that use **DbContext**

E.4

Entity Data Models

Entity Properties

```
<EntityType Name="Category">
  ...
  <Property Name="CategoryName" ...
  <NavigationProperty
    Name="Products" ...
```

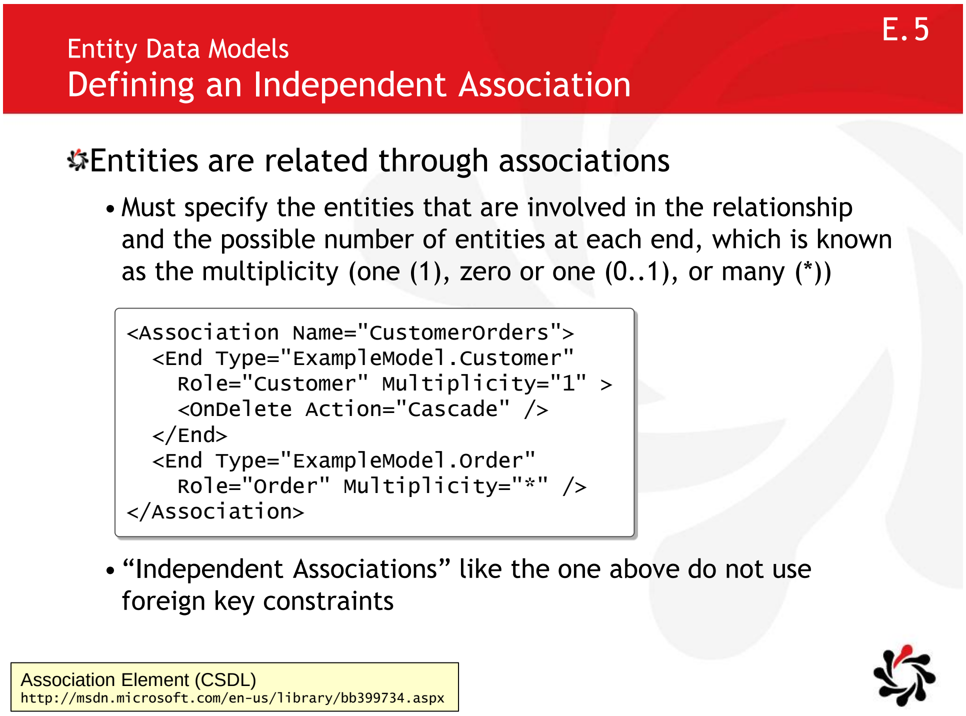


Scalar property in class

```
[EdmScalarPropertyAttribute(
  EntityKeyProperty=false, IsNullable=false)]
public global::System.String CategoryName {
```

Navigation property in EntityObject class

```
[EdmRelationshipNavigationPropertyAttribute(
  "NorthwindModel", "FK_Products_Categories", "Products")]
public EntityCollection<Product> Products {
```



E.5

Entity Data Models

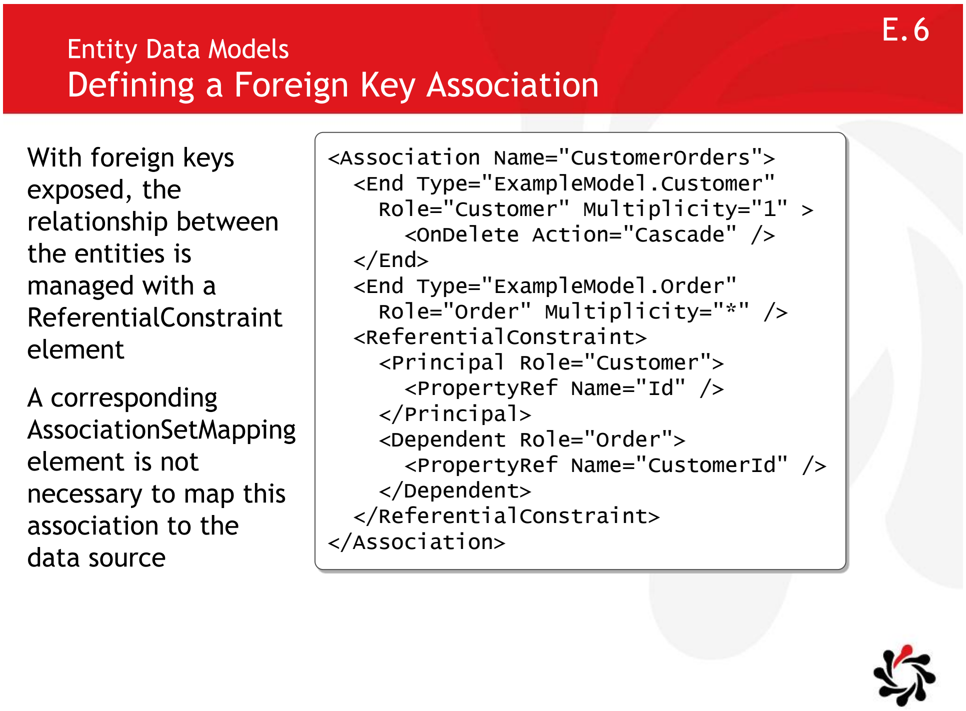
Defining an Independent Association

- Entities are related through associations
  - Must specify the entities that are involved in the relationship and the possible number of entities at each end, which is known as the multiplicity (one (1), zero or one (0..1), or many (*))

```
<Association Name="CustomerOrders">
  <End Type="ExampleModel.Customer"
    Role="Customer" Multiplicity="1" >
    <OnDelete Action="Cascade" />
  </End>
  <End Type="ExampleModel.Order"
    Role="Order" Multiplicity="*" />
</Association>
```

  - “Independent Associations” like the one above do not use foreign key constraints

Association Element (CSDL)
http://msdn.microsoft.com/en-us/library/bb399734.aspx



E.6

Entity Data Models

Defining a Foreign Key Association

With foreign keys exposed, the relationship between the entities is managed with a ReferentialConstraint element

A corresponding AssociationSetMapping element is not necessary to map this association to the data source

```
<Association Name="CustomerOrders">
  <End Type="ExampleModel.Customer"
    Role="Customer" Multiplicity="1" >
      <OnDelete Action="Cascade" />
  </End>
  <End Type="ExampleModel.Order"
    Role="Order" Multiplicity="*" />
  <ReferentialConstraint>
    <Principal Role="Customer">
      <PropertyRef Name="Id" />
    </Principal>
    <Dependent Role="Order">
      <PropertyRef Name="CustomerId" />
    </Dependent>
  </ReferentialConstraint>
</Association>
```

E.7

Designer-Generated Code (VS2010)
ObjectSet(T), ObjectQuery(T), and IQueryable(T)

- ObjectContext uses ObjectSet(T) for the entity sets

```
public class AWContext : ObjectContext { ...
  public ObjectSet<Product> Products { ...
```

- ObjectSet(T) inherits from ObjectQuery(T) and implements some common interfaces

```
public class ObjectSet<TEntity> : ObjectQuery<TEntity>,
  IObjectSet<TEntity>, IQueryable<TEntity>,
  IEnumerable<TEntity>, IQueryable, IEnumerable
  where TEntity : class
```

- Cast your LINQ query to access ObjectQuery features such as ToTraceString to see the T-SQL (or whatever)

```
string TSQL = ((ObjectQuery)query).ToTraceString();
```

ObjectQuery<T> Class
http://msdn.microsoft.com/en-us/library/bb345303.aspx

Profiling Database Activity in the Entity Framework
http://msdn.microsoft.com/en-gb/magazine/gg490349.aspx

E.8

Designer-Generated Code (VS2010)
EntityCollection(T), EntityReference(T)

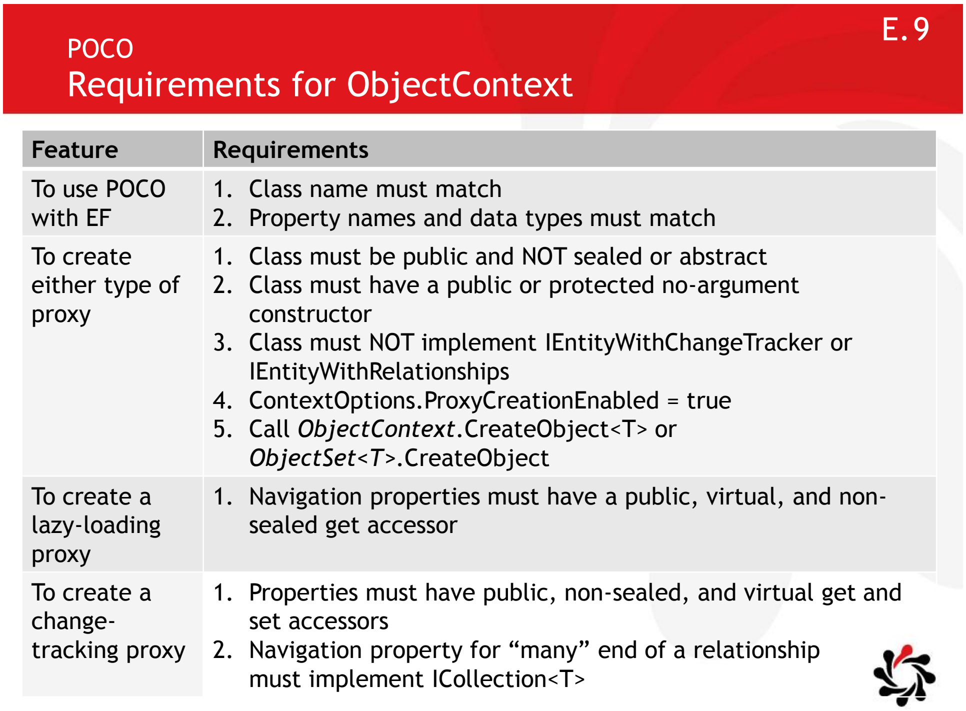
- Navigation properties in designer-generated code

```
public class Customer : EntityObject { ...
  public EntityCollection<Order> Orders { ...
```

```
public class Order : EntityObject { ...
  public EntityReference<Customer> CustomerReference { ...
```

- Useful members
  - IsLoaded: returns true if collection or reference is loaded
  - Load(MergeOption): Loads related object(s)
  - CreateSourceQuery(): Returns an object query that returns the same object(s) that exist in the current collection or reference
  - EntityCollection(T).Count: the number of entities in collection
  - EntityReference(T).Value: the actual entity

EntityCollection(TEntity) Class
http://msdn.microsoft.com/en-us/library/bb354106.aspx

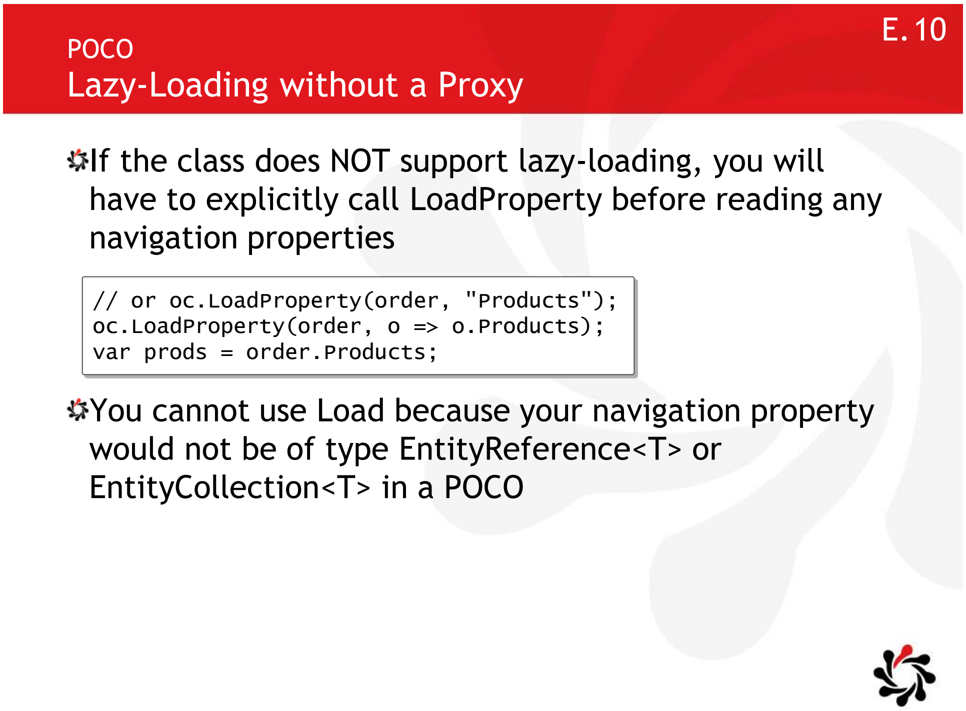


E.9

POCO

Requirements for ObjectContext

Feature	Requirements
To use POCO with EF	1. Class name must match 2. Property names and data types must match
To create either type of proxy	1. Class must be public and NOT sealed or abstract 2. Class must have a public or protected no-argument constructor 3. Class must NOT implement IEntityWithChangeTracker or IEntityWithRelationships 4. ContextOptions.ProxyCreationEnabled = true 5. Call *ObjectContext*.CreateObject<T> or *ObjectSet<T>*.CreateObject
To create a lazy-loading proxy	1. Navigation properties must have a public, virtual, and non-sealed get accessor
To create a change-tracking proxy	1. Properties must have public, non-sealed, and virtual get and set accessors 2. Navigation property for “many” end of a relationship must implement ICollection<T>



E.10

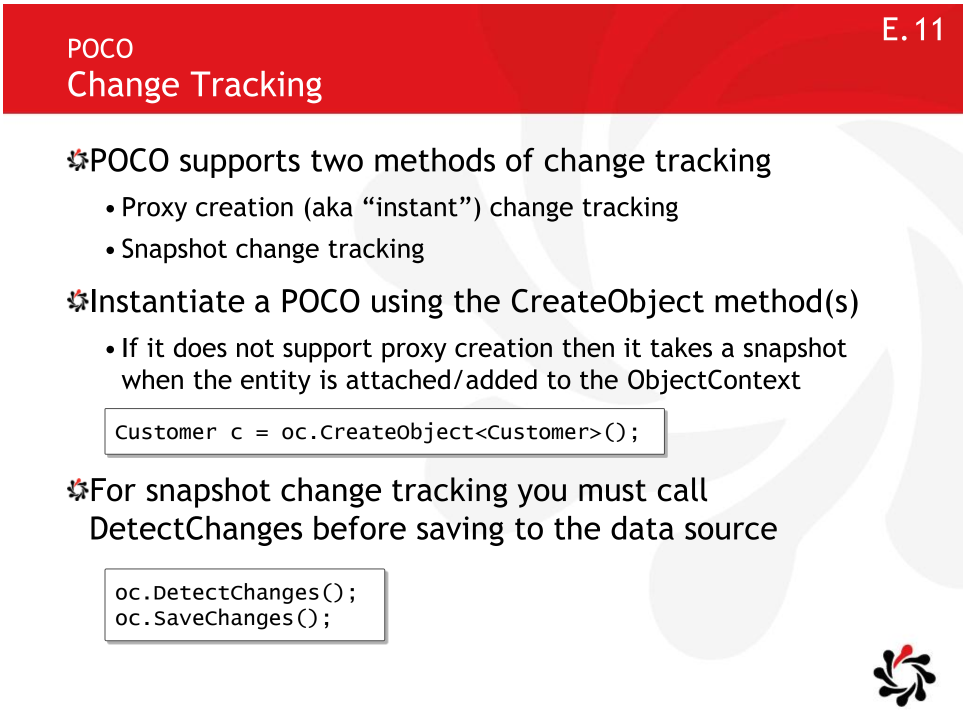
POCO

Lazy-Loading without a Proxy

- If the class does NOT support lazy-loading, you will have to explicitly call LoadProperty before reading any navigation properties

```
// or oc.LoadProperty(order, "Products");
oc.LoadProperty(order, o => o.Products);
var prods = order.Products;
```

- You cannot use Load because your navigation property would not be of type EntityReference<T> or EntityCollection<T> in a POCO



E.11

POCO

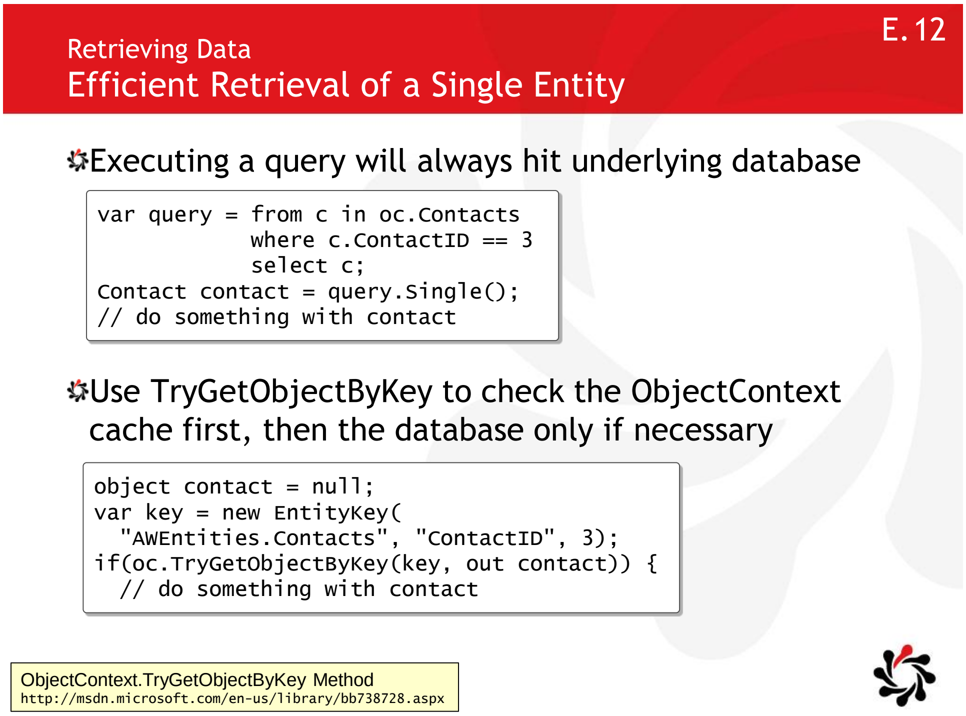
Change Tracking

- POCO supports two methods of change tracking
  - Proxy creation (aka “instant”) change tracking
  - Snapshot change tracking
- Instantiate a POCO using the CreateObject method(s)
  - If it does not support proxy creation then it takes a snapshot when the entity is attached/added to the ObjectContext

```
Customer c = oc.CreateObject<Customer>();
```

- For snapshot change tracking you must call DetectChanges before saving to the data source

```
oc.DetectChanges();
oc.SaveChanges();
```



E.12

Retrieving Data

Efficient Retrieval of a Single Entity

- Executing a query will always hit underlying database

```
var query = from c in oc.Contacts
            where c.ContactID == 3
            select c;
Contact contact = query.Single();
// do something with contact
```

- Use TryGetObjectByKey to check the ObjectContext cache first, then the database only if necessary

```
object contact = null;
var key = new EntityKey(
  "AWEntities.Contacts", "ContactID", 3);
if(oc.TryGetObjectByKey(key, out contact)) {
  // do something with contact
```

ObjectContext.TryGetObjectByKey Method
http://msdn.microsoft.com/en-us/library/bb738728.aspx

Retrieving Data

Entity Keys

E.13

- You can create an instance of EntityKey by using the class constructor

```
var key = new EntityKey(
  "AWEntities.Contacts", "ContactID", 3);
```

- Or you can use the CreateEntityKey method of ObjectContext to generate a key for a specific object

```
Contact c = ...;
var key = oc.CreateEntityKey("Contacts", c);
```

How to: Create an EntityKey
http://msdn.microsoft.com/en-us/library/dd283138.aspx

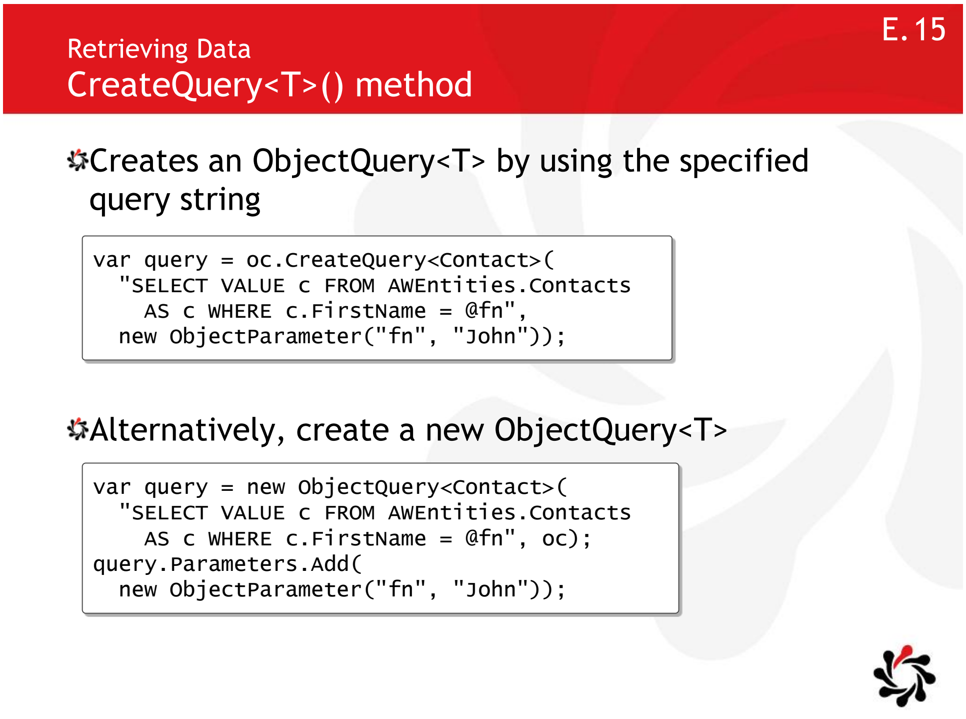
Retrieving Data

ObjectContext execute methods

E.14

- ExecuteFunction
  - Executes a stored procedure or function that is mapped in the conceptual model
- ExecuteStoreCommand
  - Executes an arbitrary command directly against the data source using the existing connection
- ExecuteStoreQuery
  - Executes a query directly against the data source that returns a sequence of typed results
- Translate
  - Translates a DbDataReader that contains rows of entity data to objects of the requested entity type

ObjectContext Class
http://msdn.microsoft.com/en-us/library/bb156104



E.15

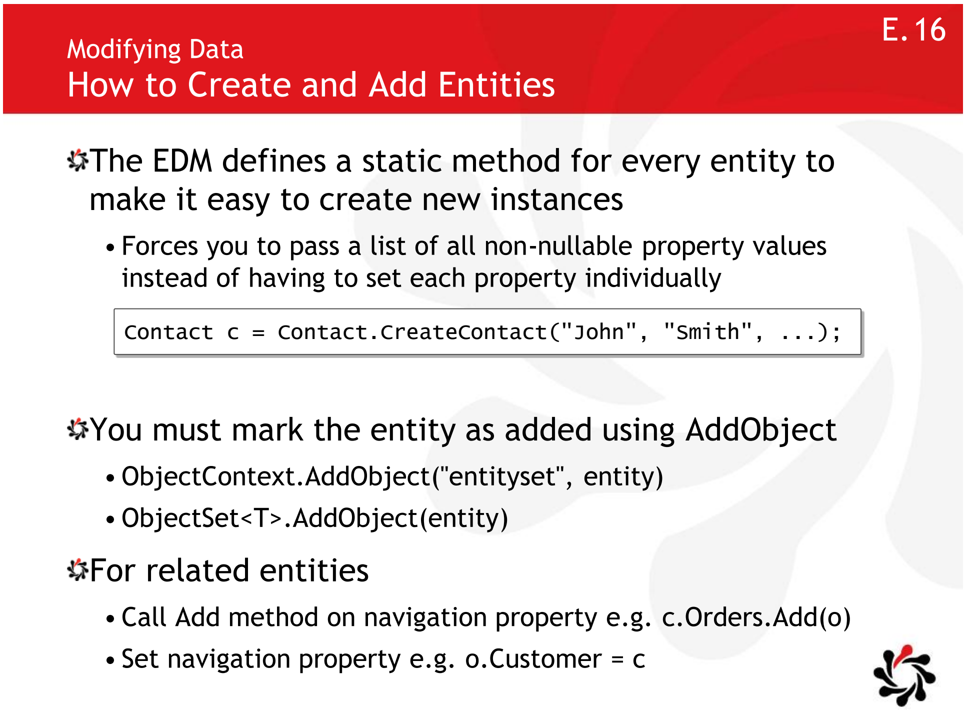
Retrieving Data
CreateQuery<T>() method

- Creates an ObjectQuery<T> by using the specified query string

```
var query = oc.CreateQuery<Contact>(
  "SELECT VALUE c FROM AWEntities.Contacts
    AS c WHERE c.FirstName = @fn",
  new ObjectParameter("fn", "John"));
```

- Alternatively, create a new ObjectQuery<T>

```
var query = new ObjectQuery<Contact>(
  "SELECT VALUE c FROM AWEntities.Contacts
    AS c WHERE c.FirstName = @fn", oc);
query.Parameters.Add(
  new ObjectParameter("fn", "John"));
```



E.16

Modifying Data
How to Create and Add Entities

- The EDM defines a static method for every entity to make it easy to create new instances
  - Forces you to pass a list of all non-nullable property values instead of having to set each property individually

```
Contact c = Contact.CreateContact("John", "Smith", ...);
```

- You must mark the entity as added using AddObject
  - ObjectContext.AddObject("entityset", entity)
  - ObjectSet<T>.AddObject(entity)
- For related entities
  - Call Add method on navigation property e.g. c.Orders.Add(o)
  - Set navigation property e.g. o.Customer = c

E. 17

Modifying Data

MergeOption (only applies when loading)

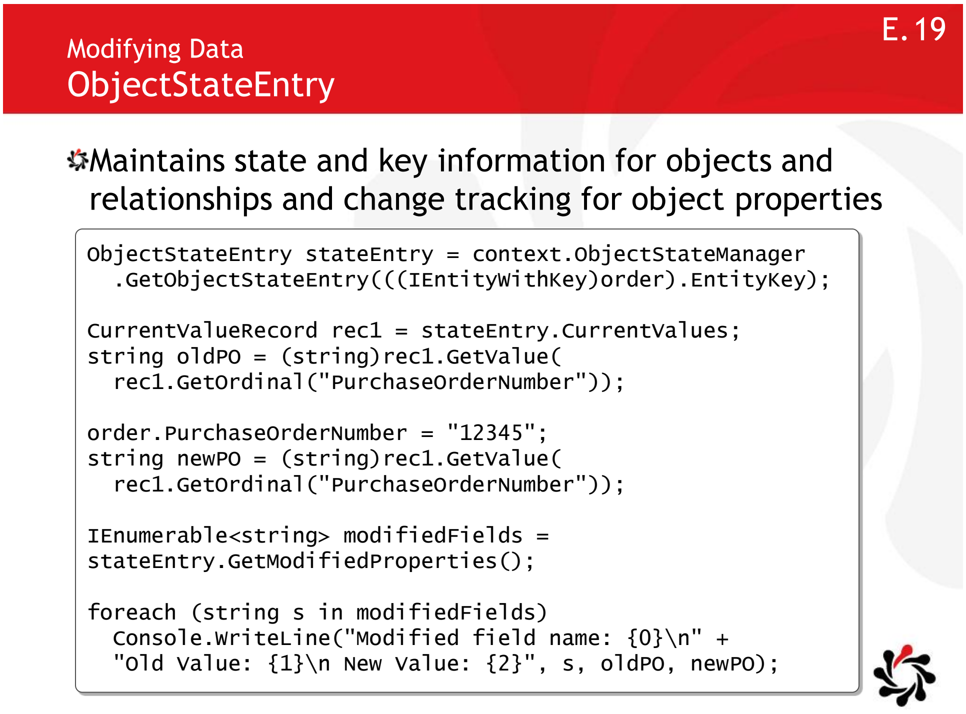
- AppendOnly (default)
  - Objects that do not exist in the object context are attached
  - Existing objects are *unaffected*
- OverwriteChanges
  - Objects that do not exist in the object context are attached
  - Existing objects are *overwritten*
- PreserveChanges
  - Objects that do not exist in the object context are attached
  - Existing objects that are Unchanged are *overwritten*
  - Existing objects that are Modified are *preserved*
- NoTracking: entities are Detached

E. 18

Modifying Data

ObjectContext methods

- AddObject(string entitySet, object)
  - Adds an object to the object context (marked as Added)
- Attach(IEntityWithKey) and Detach(object)
  - Attaches an object or object graph to the object context when the object has an entity key
- AttachTo(string entitySet, object)
  - Call AttachTo on the ObjectContext to attach the object to a specific entity set in the object context or if the object has a null (Nothing in Visual Basic) EntityKey value
- ApplyCurrentValues(string entitySet, object) and ApplyOriginalValues(string entitySet, object)
  - Copies the scalar values from the supplied object into the object in the ObjectContext that has the same key



E.19

Modifying Data

ObjectStateEntry

- Maintains state and key information for objects and relationships and change tracking for object properties

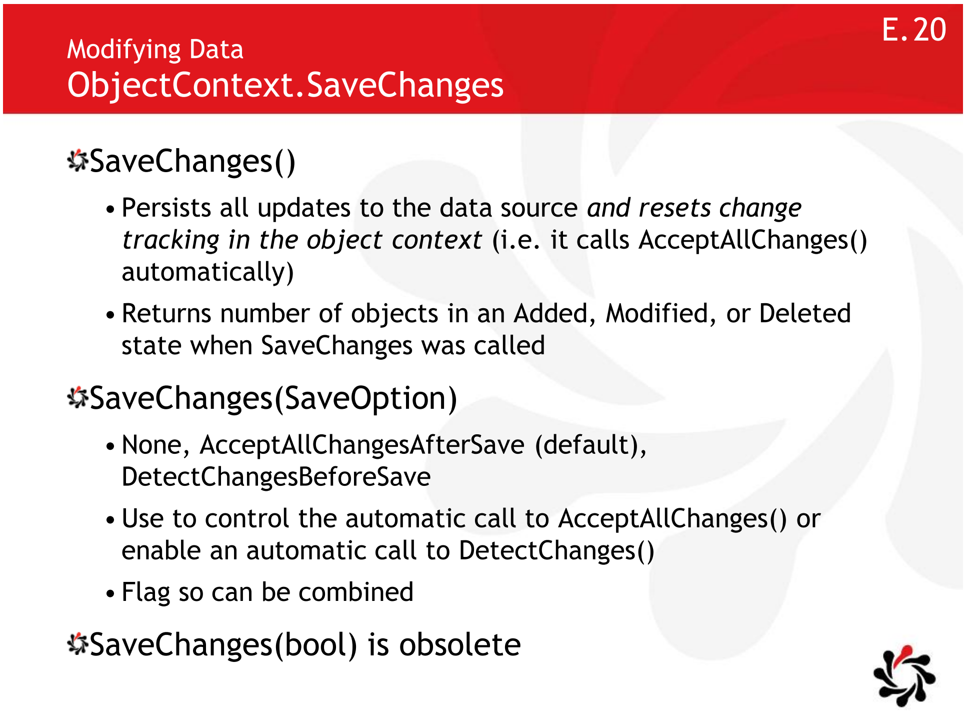
```
ObjectStateEntry stateEntry = context.ObjectStateManager
  .GetObjectStateEntry(((IEntityWithKey)order).EntityKey);

CurrentValueRecord rec1 = stateEntry.CurrentValues;
string oldPO = (string)rec1.GetValue(
  rec1.GetOrdinal("PurchaseOrderNumber"));

order.PurchaseOrderNumber = "12345";
string newPO = (string)rec1.GetValue(
  rec1.GetOrdinal("PurchaseOrderNumber"));

IEnumerable<string> modifiedFields =
stateEntry.GetModifiedProperties();

foreach (string s in modifiedFields)
  Console.WriteLine("Modified field name: {0}\n" +
  "Old Value: {1}\n New Value: {2}", s, oldPO, newPO);
```



E.20

Modifying Data

ObjectContext.SaveChanges

- SaveChanges()
  - Persists all updates to the data source *and resets change tracking in the object context* (i.e. it calls AcceptAllChanges() automatically)
  - Returns number of objects in an Added, Modified, or Deleted state when SaveChanges was called
- SaveChanges(SaveOption)
  - None, AcceptAllChangesAfterSave (default), DetectChangesBeforeSave
  - Use to control the automatic call to AcceptAllChanges() or enable an automatic call to DetectChanges()
  - Flag so can be combined
- SaveChanges(bool) is obsolete

E.21

Modifying Data

OptimisticConcurrencyException

- When invoking SaveChanges, catch this exception
- Invoke Refresh on context to resolve conflict
  - RefreshMode: StoreWins, ClientWins
  - Pass entity or IEnumerable
- Invoke SaveChanges again

```
try
{
  context.SaveChanges();
}
catch (OptimisticConcurrencyException ex)
{
  context.Refresh(RefreshMode.ClientWins, orders);
  context.SaveChanges();
}
```

E.22

Performance

Considerations

Operation	Frequency	Comments
Loading metadata	Once per AppDomain	
Opening connection	As needed	You can manually open and close connections
Generating views	Once per AppDomain	Can pre-generate
Preparing the query	Once for each unique query	Can pre-compile
Executing the query	Once for each query	
Loading and validating types	Once for each ObjectContext instance	Types are validated against the model
Tracking	Once for each object returned*	MergeOption.NoTracking disables this step
Materializing	Once for each object returned*	

*Not required when using EntityDataReader

E.23

Performance

Compiling Queries for LINQ Providers

- Both LINQ to Entities and LINQ to SQL have a CompiledQuery class with a Compile method
  - … but the concept of compilation doesn't really make sense for LINQ to Object queries
  - In LINQ to Objects, a Where method takes a **Func<T, bool>** as opposed to an **Expression<Func<T, bool>>**
  - In LINQ to Objects, the C# compiler compiles the lambda expression down to an anonymous method and generates the IL at compile time and passes a delegate to that Where method
  - In other LINQ flavors, like LINQ to Entities, the lambda is not compiled to IL; Instead, the compiler builds up an expression tree object out of the lambda expression and passes the expression tree to LINQ methods

Is it possible to compile a query for LINQ to Objects?
`http://stackoverflow.com/questions/2649874/is-it-possible-to-compile-a-query-for-linq-to-objects`

E.24

EF 4.1

What is Microsoft ADO.NET Entity Framework 4.1?

- aka “Magic Unicorn Edition” for VS2010 and later
- EF 4.1 introduces two new features
  - The **DbContext API** is a simplified abstraction over ObjectContext and a number of other types
  - **Code First** is a new development pattern that provides an alternative to the Database First and Model First patterns
- Code First is focused around defining your model using .NET classes
  - These classes can then be mapped to an existing database or be used to generate a database schema
  - Additional configuration can be supplied using Data Annotations or via a fluent API

EF 4.1 Released
`http://blogs.msdn.com/b/adonet/archive/2011/04/11/ef-4-1-released.aspx`

F.1

Appendix F
Firebrand Sample Services

Developing Windows Azure and Web Services

Updated 11th April 2014

F.2

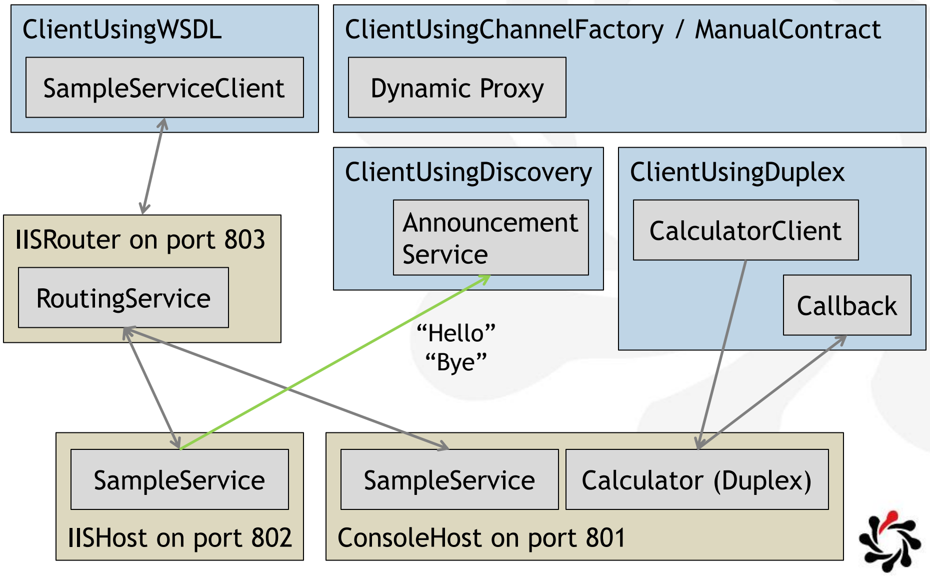
Firebrand Sample Services

Contents

Topic	Slide
Overview	3
The Service	5
The Hosts - Console	8
The Clients	12
Exposing Data	21
The Hosts – IIS	24
Faults	27
Duplex	30
REST	34
Discovery	35
Router	40
Sessions	46

F. 3

Overview
Clients and Services



ClientUsingWSDL

SampleServiceClient

ClientUsingChannelFactory / ManualContract

Dynamic Proxy

ClientUsingDiscovery

Announcement Service

ClientUsingDuplex

CalculatorClient

Callback

IISRouter on port 803

RoutingService

“Hello”
“Bye”

SampleService

IISHost on port 802

SampleService

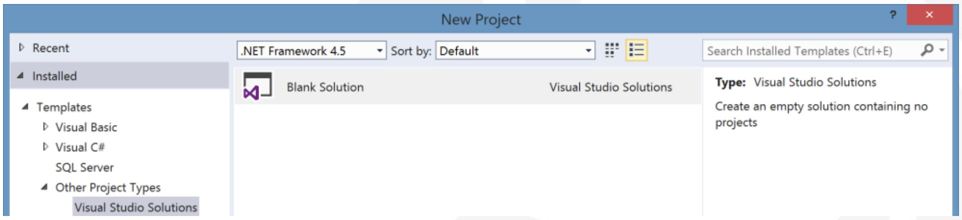
Calculator (Duplex)

ConsoleHost on port 801

F. 4

Overview
Start with a Blank Solution

- Add a new Blank Solution named Firebrand



New Project

.NET Framework 4.5

Sort by: Default

Blank Solution — Visual Studio Solutions

Type: Visual Studio Solutions

Create an empty solution containing no projects

- This solution will eventually contain these projects
  - Services: S1.DataContracts, S2.ServiceContracts, S3.Implementation
  - Hosts: H1.ConsoleHost, H2.IISHost, H3.IISRouter
  - Clients: C1.ClientUsingWSDL, C2.ClientUsingChannelFactory, C3.ClientUsingManualContract, C4.ClientUsingDuplex, C5.ClientUsingDiscovery

Download complete solution
`http://bit.ly/fbfiles`

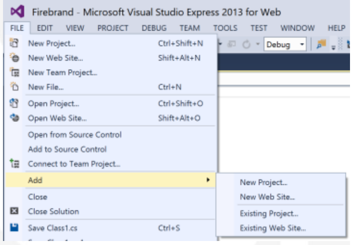
F. 5

The Service

Define the Data Contract(s)

Add a new Class Library

- Name: **S1.DataContracts**
- Add a reference to **System.Runtime.Serialization**
- Rename Class1.cs to **Widget.cs**
- Change the namespace to **Firebrand**



```
namespace Firebrand {
    using System.Runtime.Serialization;
    [DataContract]
    public class Widget
    {
        [DataMember]
        public string Name { get; set; }
        [DataMember]
        public bool Active { get; set; }
    }
}
```

F. 6

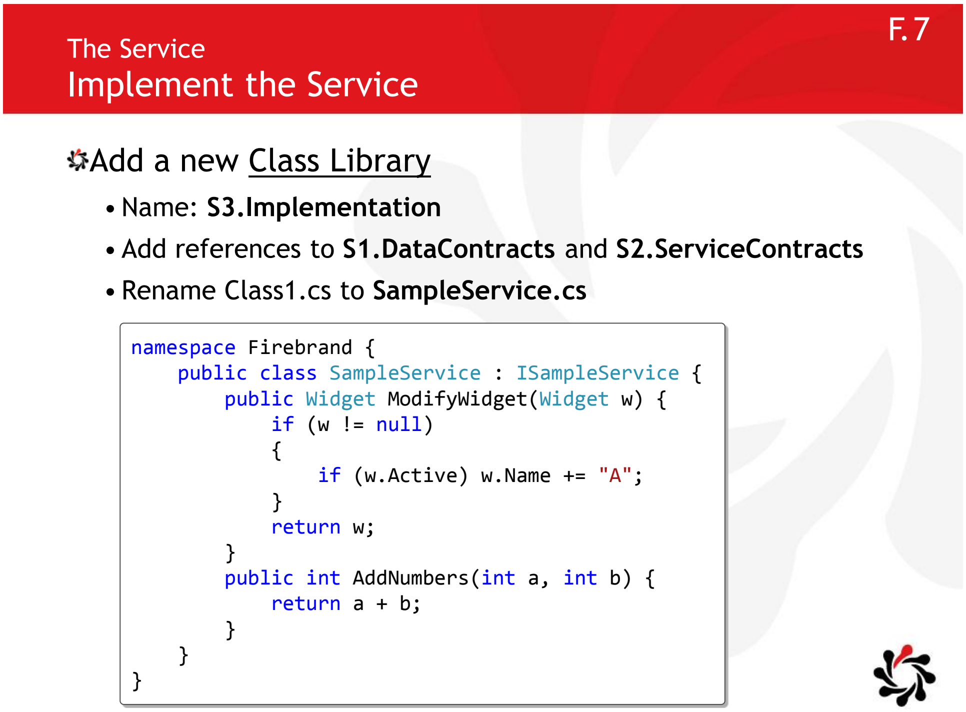
The Service

Define the Service Contract(s)

Add a new Class Library

- Name: **S2.ServiceContracts**
- Add references to the **S1.DataContracts** project and **System.ServiceModel** assembly
- Rename Class1.cs to **ISampleService.cs**

```
namespace Firebrand
{
    using System.ServiceModel;
    [ServiceContract]
    public interface ISampleService
    {
        [OperationContract]
        Widget ModifyWidget(Widget w);
        [OperationContract]
        int AddNumbers(int a, int b);
    }
}
```



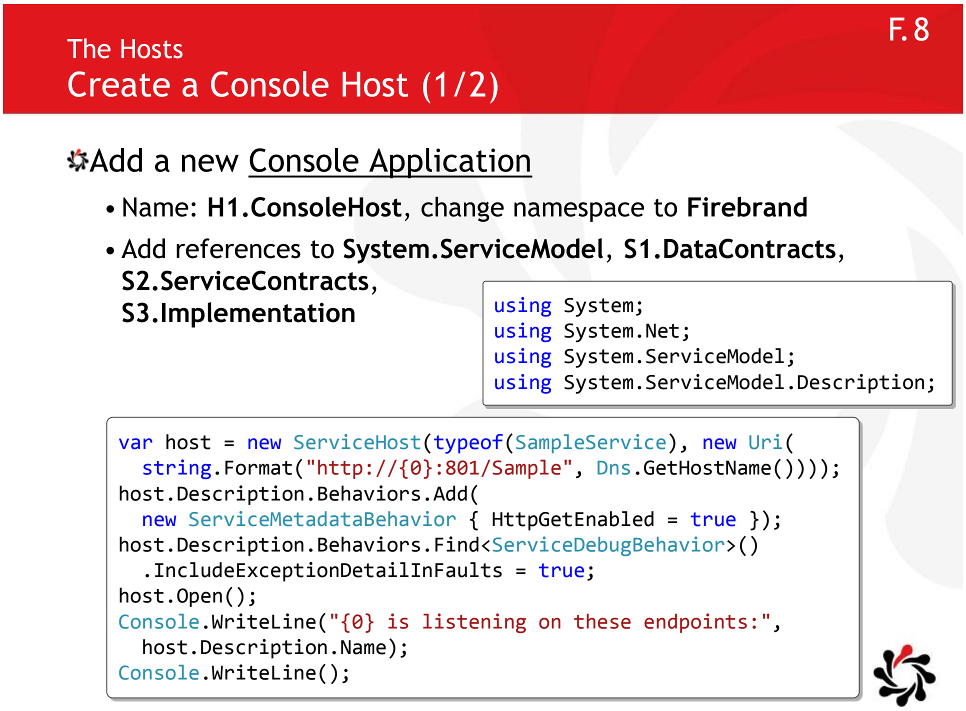
F.7

The Service
Implement the Service

Add a new Class Library

- Name: **S3.Implementation**
- Add references to **S1.DataContracts** and **S2.ServiceContracts**
- Rename Class1.cs to **SampleService.cs**

```
namespace Firebrand {
    public class SampleService : ISampleService {
        public Widget ModifyWidget(Widget w) {
            if (w != null)
            {
                if (w.Active) w.Name += "A";
            }
            return w;
        }
        public int AddNumbers(int a, int b) {
            return a + b;
        }
    }
}
```



F.8

The Hosts
Create a Console Host (1/2)

Add a new Console Application

- Name: **H1.ConsoleHost**, change namespace to **Firebrand**
- Add references to **System.ServiceModel**, **S1.DataContracts**, **S2.ServiceContracts**, **S3.Implementation**

```
using System;
using System.Net;
using System.ServiceModel;
using System.ServiceModel.Description;
```

```
var host = new ServiceHost(typeof(SampleService), new Uri(
  string.Format("http://{0}:801/Sample", Dns.GetHostName())));
host.Description.Behaviors.Add(
  new ServiceMetadataBehavior { HttpGetEnabled = true });
host.Description.Behaviors.Find<ServiceDebugBehavior>()
  .IncludeExceptionDetailInFaults = true;
host.Open();
Console.WriteLine("{0} is listening on these endpoints:",
  host.Description.Name);
Console.WriteLine();
```

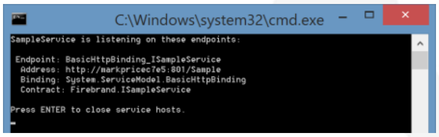
The Hosts
Create a Console Host (2/2)

- Enumerate the endpoints that have been configured, including automatic (aka default) endpoints
  - Wait for the user to press ENTER, then close the host

```
foreach (var ep in host.Description.Endpoints)
{
    Console.WriteLine(" Endpoint: " + ep.Name);
    Console.WriteLine("  Address: " + ep.Address);
    Console.WriteLine("  Binding: " + ep.Binding);
    Console.WriteLine("  Contract: " + ep.Contract.ContractType);
    Console.WriteLine();
}
Console.WriteLine("Press ENTER to close service hosts.");
Console.ReadLine();
Console.WriteLine("Closing hosts... please wait.");
host.Close();
Console.WriteLine("Closed.");
```

The Hosts
Start the Console Host

- **Ctrl+F5** to start without attaching the debugger



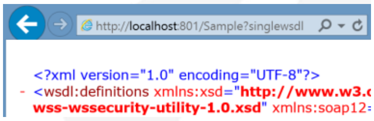
```
SampleService is listening on these endpoints:

 Endpoint: BasicHttpBinding_ISampleService
  Address: http://markpricec7e5:801/Sample
  Binding: System.ServiceModel.BasicHttpBinding
  Contract: Firebrand.ISampleService

Press ENTER to close service hosts.
_
```

  - If you see this exception, restart Visual Studio as Administrator

```
Unhandled Exception: System.ServiceModel.AddressAccessDeniedException: HTTP coul
d not register URL http://+:801/Sample/. Your process does not have access right
s to this namespace (see http://go.microsoft.com/fwlink/?LinkId=70353 for detail
s). ---> System.Net.HttpListenerException: Access is denied
```



```
  <?xml version="1.0" encoding="UTF-8"?>
- <wsdl:definitions xmlns:xsd="http://www.w3.c
  wss-wssecurity-utility-1.0.xsd" xmlns:soap12=
```

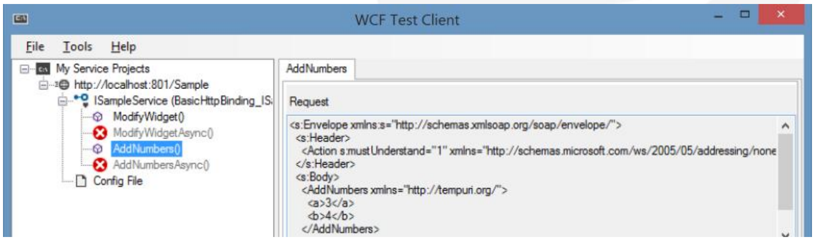
- Run Internet Explorer and enter
  - http://localhost:801/Sample?singlewsdl

The Hosts
Test the Console Host

Run the WCF Test Client

```
C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\WcfTestClient.exe
```

- File-Add Service...
- Enter URL: **http://localhost:801/Sample**
- Double-click **Config File** and review it (or right-click to edit it)
- Double-click **AddNumbers**, fill parameters, click **Invoke**
- Click **XML** tab to see SOAP messages



```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none
  </s:Header>
  <s:Body>
    <AddNumbers xmlns="http://tempuri.org/">
      <a>3</a>
      <b>4</b>
    </AddNumbers>
```

The Hosts
Create Explicit WS-* and Custom Binding Endpoints

In H1.ConsoleHost, in App.config

- Define a pair of binding configurations and endpoints to use them

```
<system.serviceModel>
  <bindings>
    <wsHttpBinding>
      <binding name="FBWSNoSecurity">
        <security mode="None"/>
```

```
<customBinding>
  <binding name="FBBinaryOverHttp">
    <binaryMessageEncoding/>
    <httpTransport/>
```

```
<services>
  <service name="Firebrand.SampleService">
    <endpoint address=""
              binding="customBinding"
              bindingConfiguration="FBBinaryOverHttp"
              contract="Firebrand.ISampleService" />
    <endpoint address="ws"
              binding="wsHttpBinding"
              bindingConfiguration="FBWSNoSecurity"
              contract="Firebrand.ISampleService" />
```

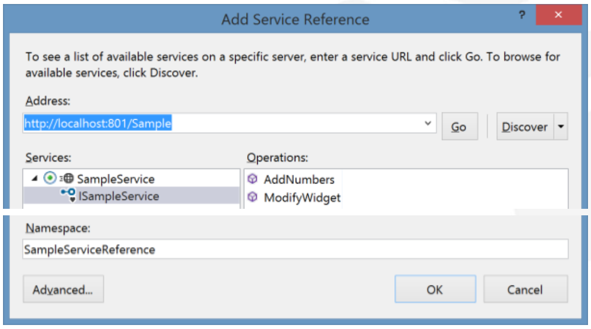
F. 13

The Clients

Create a Client with a Service Reference

Add a new WPF Application (clients can be anything)

- Name: **C1.ClientUsingWSDL**
- Ensure **H1.ConsoleHost** is running
- **Add Service Reference** and type **http://localhost:801/Sample**
  - Expand the service to download the WSDL, click the interface
  - Namespace: **SampleServiceReference**, click OK



To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost:801/Sample

Services: SampleService, ISampleService

Operations: AddNumbers, ModifyWidget

Namespace: SampleServiceReference

F. 14

The Clients

Code the Client to Use the Proxy (Add Numbers)

MainWindow: add two text boxes named **A** and **B**, a label named **Answer**, and a button named **AddNumbers**

```
using C1.ClientUsingWSDL.SampleServiceReference;
```

```
try
{
    var proxy = new SampleServiceClient("WSHttpBinding_ISampleService");
    int a = int.Parse(A.Text);
    int b = int.Parse(B.Text);
    Answer.Content = proxy.AddNumbers(a, b); // or use async & await
}
catch (Exception ex)
{
    Answer.Content = ex.Message;
}
```

```
Answer.Content = await proxy.AddNumbersAsync(a, b);
```

If you use await you must also apply async to method

```
private async void AddNumbers_Click(object sender, RoutedEventArgs e)
```

F. 15

The Clients
Code the Client to Use the Proxy (Modify Widget)

- Add a check box named **Active**, a text box named **NameTextBox**, a label named **ModifiedName**, and a button named **ModifyWidget**

```
try
{
    var proxy = new SampleServiceClient(
        "CustomBinding_ISampleService");
    var w = new Widget
    {
        Name = NameTextBox.Text,
        Active = Active.IsChecked.Value
    };
    w = await proxy.ModifyWidgetAsync(w);
    ModifiedName.Content = "Modified name: " + w.Name;
    proxy.Close();
}
catch (Exception ex)
{
    ModifiedName.Content = "Exception: " + ex.Message;
}
```

F. 16

The Clients
Reuse Data Contracts

- The Widget class is currently re-created on the client by the service reference
  - Underneath **SampleServiceReference** review the **Reference.cs** file (you will need to **Show All Files**)
- To reuse the original data contract
  - Add a reference to **S1.DataContracts**
  - Right-click **SampleServiceReference** and choose **Update Service Reference**
- Now the Widget class is not re-created
  - Import the Firebrand namespace to use it

```
using Firebrand;
```

F. 17

The Clients
Read from <client> endpoints

- When you have multiple client endpoints the proxy constructor must have the name of the endpoint set
  - Add a reference to System.Configuration
  - Add a ComboBox named Endpoints
  - Handle the Window_Loaded event

```
using System.ServiceModel.Configuration;
using System.Configuration;
```

```
var client = (ClientSection)ConfigurationManager.GetSection(
  "system.serviceModel/client");
Endpoints.Items.Clear();
foreach (ChannelEndpointElement ep in client.Endpoints)
    Endpoints.Items.Add(ep.Name);
if (Endpoints.Items.Count > 0)
    Endpoints.SelectedIndex = 0;
```

```
var proxy = new SampleServiceClient(Endpoints.Text);
```

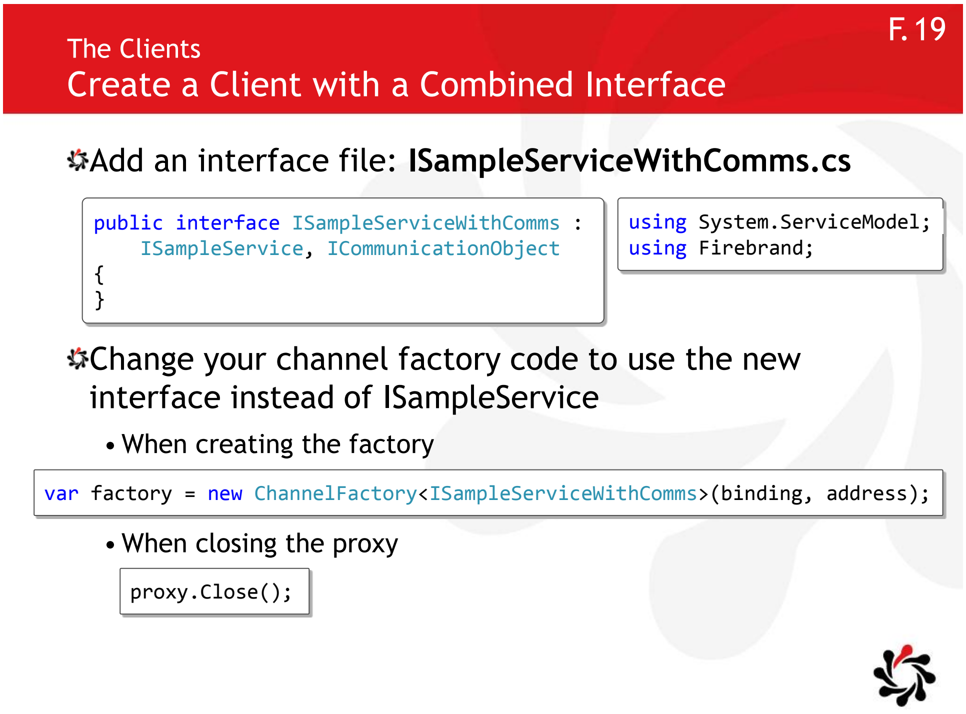
F. 18

The Clients
Create a Client with a Channel Factory

- Add a new WPF Application
  - Name: **C2.ClientUsingChannelFactory**
  - Add references to **System.ServiceModel**, **S1.DataContracts** and **S2.ServiceContracts** projects
  - Draw a check box, text box, label, and button

```
var address = new EndpointAddress("http://localhost:801/Sample/ws");
var binding = new WSHttpBinding
    { Security = new WSHttpSecurity { Mode = SecurityMode.None } };
var factory = new ChannelFactory<ISampleService>(binding, address);
var proxy = factory.CreateChannel();
var w = new Widget
{
    Name = NameTextBox.Text,
    Active = Active.IsChecked.Value
};
w = proxy.ModifyWidget(w);
ModifiedName.Content = "Modified: " + w.Name;
((ICommunicationObject)proxy).Close();
```

```
using System.ServiceModel;
using Firebrand;
```



F. 19

The Clients

Create a Client with a Combined Interface

- Add an interface file: **ISampleServiceWithComms.cs**

```
public interface ISampleServiceWithComms :
    ISampleService, ICommunicationObject
{
}
```

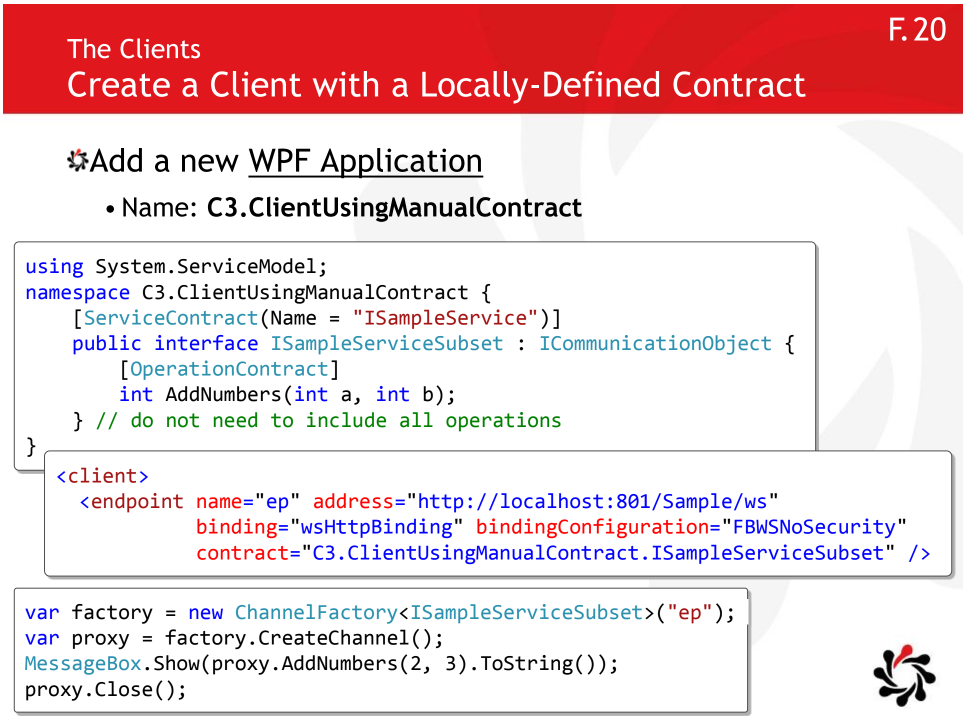
```
using System.ServiceModel;
using Firebrand;
```

- Change your channel factory code to use the new interface instead of ISampleService
  - When creating the factory

```
var factory = new ChannelFactory<ISampleServiceWithComms>(binding, address);
```

  - When closing the proxy

```
proxy.Close();
```



F. 20

The Clients

Create a Client with a Locally-Defined Contract

- Add a new WPF Application
  - Name: **C3.ClientUsingManualContract**

```
using System.ServiceModel;
namespace C3.ClientUsingManualContract {
    [ServiceContract(Name = "ISampleService")]
    public interface ISampleServiceSubset : ICommunicationObject {
        [OperationContract]
        int AddNumbers(int a, int b);
    } // do not need to include all operations
}
```

```
<client>
  <endpoint name="ep" address="http://localhost:801/Sample/ws"
            binding="wsHttpBinding" bindingConfiguration="FBWSNoSecurity"
            contract="C3.ClientUsingManualContract.ISampleServiceSubset" />
```

```
var factory = new ChannelFactory<ISampleServiceSubset>("ep");
var proxy = factory.CreateChannel();
MessageBox.Show(proxy.AddNumbers(2, 3).ToString());
proxy.Close();
```

F. 21

Exposing Data
Using Entity Framework (Data Contracts)

S1.DataContracts

- Add **ADO.NET Entity Data Model**, “Generate from Database”
- Connect to **(localdb)\v11.0** and **Northwind** database
- Select all tables and click **Finish**
- Review NorthwindModel.Context.cs and Category.cs

Optional but strongly recommended

- Click background of EDMX design window
- In Properties, change “Lazy Loading Enabled” to False

F. 22

Exposing Data
Using Entity Framework (Service Contract and Impl.)

S2.ServiceContracts, ISampleService.cs

```
[OperationContract]
Product[] GetProducts();
[OperationContract]
Product GetProduct(int ProductID);
```

S3.Implementation, SampleService.cs

```
public Product[] GetProducts() {
    using (var db = new NorthwindEntities()) {
        db.Configuration.LazyLoadingEnabled = false;
        return db.Products.ToArray();
    }
}
```

```
using System.Linq;
```

```
public Product GetProduct(int ProductID) {
    using (var db = new NorthwindEntities()) {
        db.Configuration.LazyLoadingEnabled = false;
        return db.Products.Single(p => p.ProductID == ProductID);
    }
}
```

F. 23

Exposing Data
Using Entity Framework (Host)

H1.ConsoleHost, App.config

- Copy and paste the connectionStrings section from S1.DataContracts into the host's App.config file

```
<connectionStrings>
  <add name="NorthwindEntities"
connectionString="metadata=res://*/NorthwindModel.csdl|
res://*/NorthwindModel.ssdl|res://*/NorthwindModel.msl;
provider=System.Data.SqlClient;provider connection
string=&quot;data source=(localdb)\v11.0;initial
catalog=Northwind;integrated
security=True;MultipleActiveResultSets=True;App=EntityF
ramework&quot;" providerName="System.Data.EntityClient"
/>
```

S3.Implementation

- Add NuGet package for EntityFramework (best to use the same version that your S1.DataContracts project uses)

F. 24

Exposing Data
Using Entity Framework (Updating References)

C1.ClientUsingWSDL

- Right-click **SampleServiceReference**, and then click **Update Service Reference**
- Add a button "Get Products" and a DataGrid named grid

```
private void GetProducts_Click(object sender, RoutedEventArgs e) {
    try {
        var proxy = new SampleServiceClient(Endpoints.Text);
        var products = proxy.GetProducts();
        grid.ItemsSource = products;
        proxy.Close();
    }
    catch(FaultException ex) {
        MessageBox.Show(ex.Message);
    }
}
```

F. 25

Exposing Data

Using Entity Framework ()

C1.ClientUsingChannelFactory

- Add a button “Get Products” and a DataGrid named grid

```
private void GetProducts_Click(object sender, RoutedEventArgs e) {
    try {
        var address = new EndpointAddress("http://localhost:801/Sample/ws");
        var binding = new WSHttpBinding
            { Security = new WSHttpSecurity { Mode = SecurityMode.None } };
        var factory = new ChannelFactory<ISampleServiceWithComms>
            (binding, address);
        var proxy = factory.CreateChannel();
        var products = proxy.GetProducts();
        grid.ItemsSource = products;
        proxy.Close();
    }
    catch(FaultException ex) {
        MessageBox.Show(ex.Message);
    }
}
```

F. 26

The Hosts - IIS

Create an IIS Host

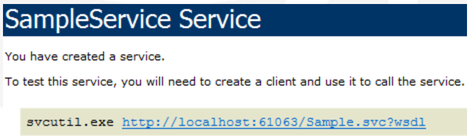
Add a new WCF Service Application

- Project Name: **H2.IISHost**
- Delete the IService1.cs and Service1.svc.cs files
- Add reference to **S3.Implementation** project
- Rename Service1.svc to **Sample.svc** and then edit it

```
<%@ ServiceHost Service="Firebrand.SampleService" %>
```

Build and test the service

- Right-click the .svc file, choose View in Browser, click WSDL link



SampleService Service

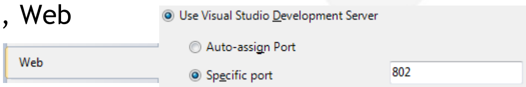
You have created a service.

To test this service, you will need to create a client and use it to call the service.

```
svcutil.exe http://localhost:61063/Sample.svc?wsdl
```

Change the port number assigned by Cassini to 802

- Project-Properties, Web



● Use Visual Studio Development Server

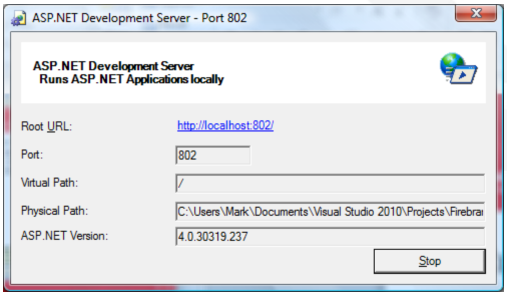
○ Auto-assign Port

● Specific port 802

F. 27

The Hosts - IIS
Test the IIS Host

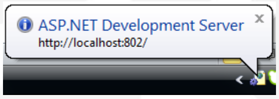
- Run the WCF Test Client
  - File-Add Service...
  - Enter URL: http://localhost:802/Sample.svc
  - Test as before



ASP.NET Development Server - Port 802

ASP.NET Development Server
Runs ASP.NET Applications locally

Root URL: http://localhost:802/
Port: 802
Virtual Path: /
Physical Path: C:\Users\Mark\Documents\Visual Studio 2010\Projects\Firebrai
ASP.NET Version: 4.0.30319.237



ASP.NET Development Server
http://localhost:802/

Warning! When testing a Web Project solution in Visual Studio 2010 you are using the ASP.NET Development Server (aka Cassini) which only supports HTTP. To test other bindings you must either deploy to IIS 7.x with WAS or create a non-Web host like the ConsoleHost project.

F. 28

The Hosts - IIS
Using Service Activations

- We do not need a physical .svc file to host a service

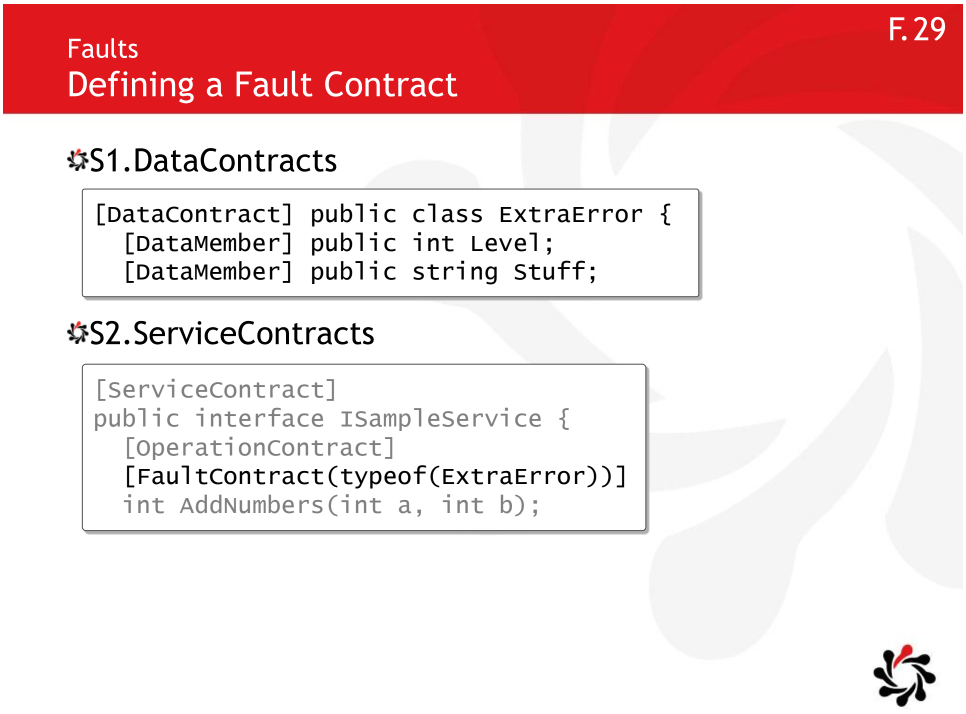
```
<serviceHostingEnvironment ...>
  <serviceActivations>
    <add relativeAddress="VirtualService.svc"
      service="Firebrand.SampleService" />
```

- You will now be able to use your “fake” service

```
http://localhost:802/VirtualService.svc
```

- You can also specify a factory to customize how the service host is created

```
<add relativeAddress="VirtualService.svc"
  service="Firebrand.SampleService"
  factory="Firebrand.SampleServiceFactory" />
```



F. 29

Faults

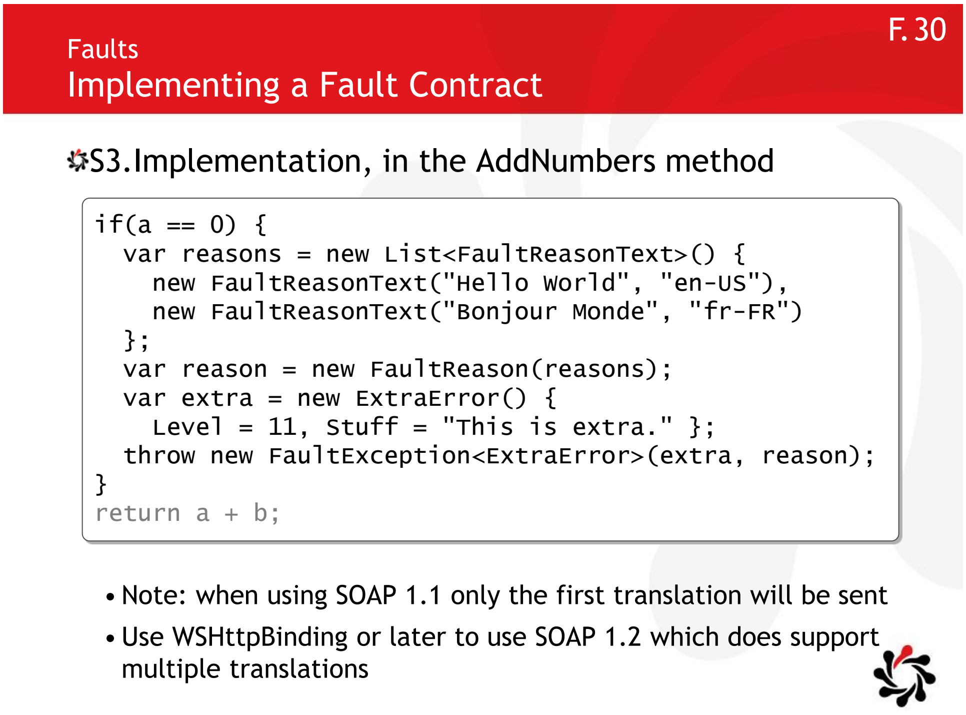
Defining a Fault Contract

S1. DataContracts

```
[DataContract] public class ExtraError {
  [DataMember] public int Level;
  [DataMember] public string Stuff;
```

S2. ServiceContracts

```
[ServiceContract]
public interface ISampleService {
  [OperationContract]
  [FaultContract(typeof(ExtraError))]
  int AddNumbers(int a, int b);
```



F. 30

Faults

Implementing a Fault Contract

S3. Implementation, in the AddNumbers method

```
if(a == 0) {
  var reasons = new List<FaultReasonText>() {
    new FaultReasonText("Hello World", "en-US"),
    new FaultReasonText("Bonjour Monde", "fr-FR")
  };
  var reason = new FaultReason(reasons);
  var extra = new ExtraError() {
    Level = 11, Stuff = "This is extra." };
  throw new FaultException<ExtraError>(extra, reason);
}
return a + b;
```

- Note: when using SOAP 1.1 only the first translation will be sent
- Use WSHttpBinding or later to use SOAP 1.2 which does support multiple translations

F. 31

Faults

Catching a Fault Contract

C1.ClientUsingServiceReference, AddNumbersButton

```
catch (FaultException<ExtraError> fex) {
  MessageBox.Show(string.Format(
    "Localized: {0}, Level: {1}, Stuff: {2}",
    // fex.Message is thread localized reason
    fex.Message, fex.Detail.Level, fex.Detail.Stuff),
    "FaultException<ExtraError>");
}
```

```
// all localized reasons
fex.Reason.Translations
```

```
catch (FaultException<ExceptionDetail> fed) {
  // if includeExceptionDetailsInFaults is true
  MessageBox.Show(fed.Detail.StackTrace,
    "FaultException<ExceptionDetail>");
}
catch (CommunicationException ce) {
  MessageBox.Show(ce.ToString(),
    "CommunicationException");
}
```

F. 32

Duplex

Service Contract

S2.ServiceContract, add two new interfaces

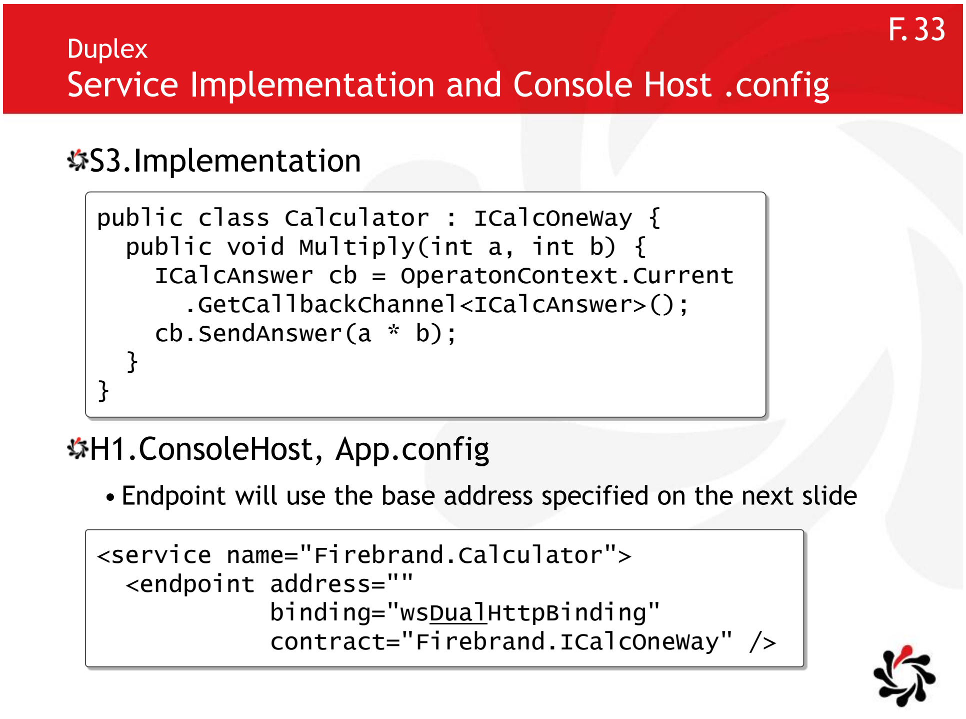
- Interface for service-side to implement

```
[ServiceContract(CallbackContract = typeof(ICalcAnswer))]
public interface ICalcOneWay {
  [OperationContract(IsOneWay = true)]
  void Multiply(int a, int b);
}
```

- Interface for client-side to implement

```
public interface ICalcAnswer {
  [OperationContract(IsOneWay = true)]
  void SendAnswer(int answer);
}
```

- No need for [ServiceContract] on this interface



F. 33

Duplex
Service Implementation and Console Host .config

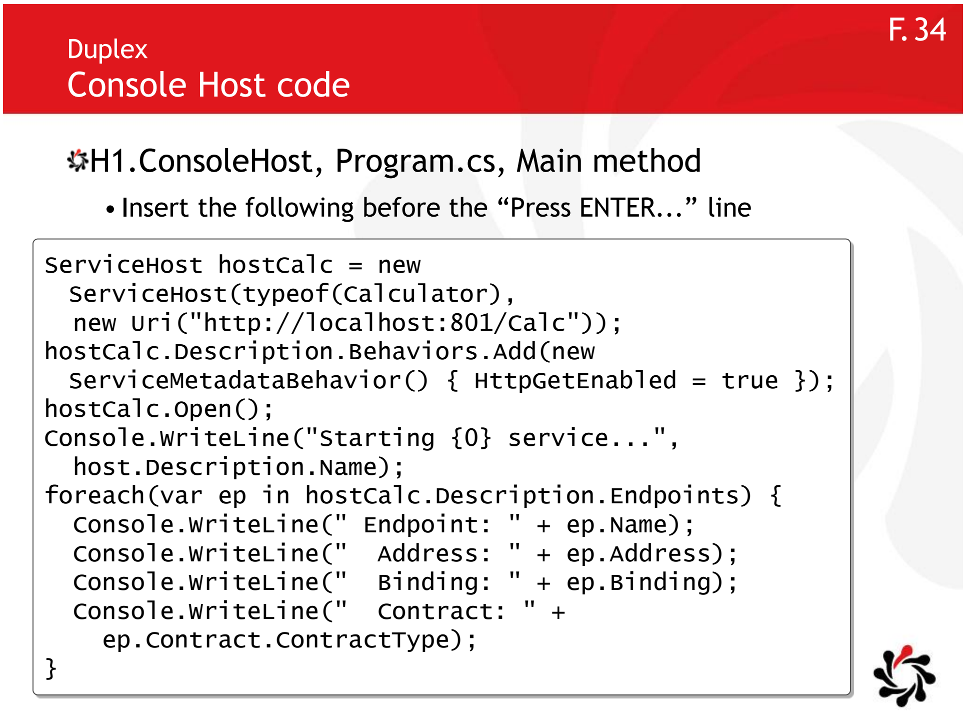
S3.Implementation

```
public class Calculator : ICalcOneWay {
  public void Multiply(int a, int b) {
    ICalcAnswer cb = OperatonContext.Current
      .GetCallbackChannel<ICalcAnswer>();
    cb.SendAnswer(a * b);
  }
}
```

H1.ConsoleHost, App.config

- Endpoint will use the base address specified on the next slide

```
<service name="Firebrand.Calculator">
  <endpoint address=""
            binding="wsDualHttpBinding"
            contract="Firebrand.ICalcOneWay" />
```



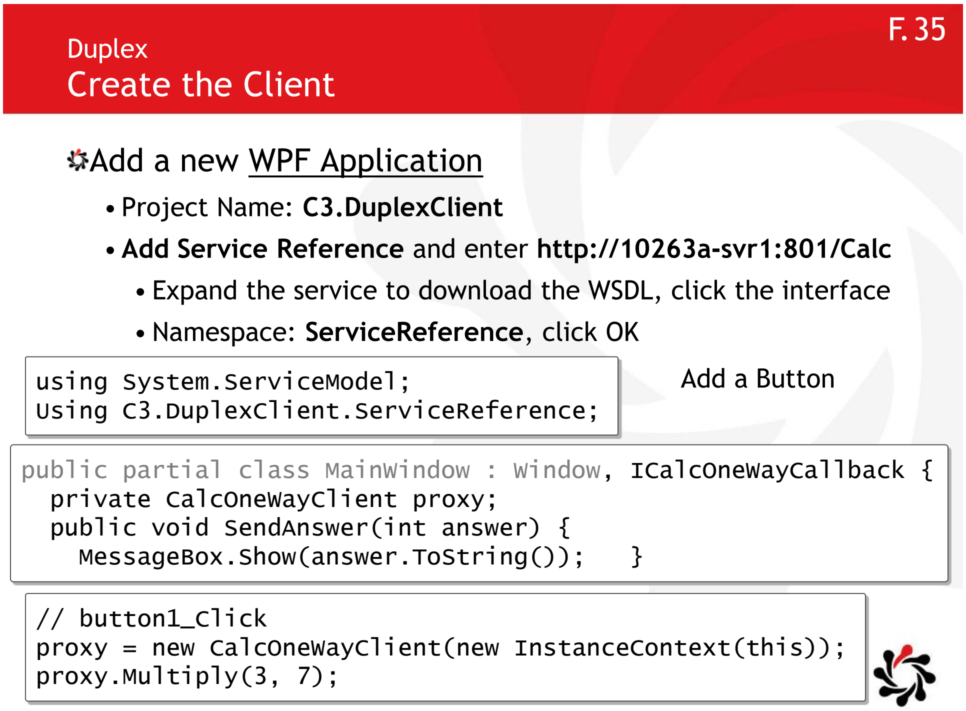
F. 34

Duplex
Console Host code

H1.ConsoleHost, Program.cs, Main method

- Insert the following before the “Press ENTER…” line

```
ServiceHost hostCalc = new
  ServiceHost(typeof(Calculator),
  new Uri("http://localhost:801/Calc"));
hostCalc.Description.Behaviors.Add(new
  ServiceMetadataBehavior() { HttpGetEnabled = true });
hostCalc.Open();
Console.WriteLine("Starting {0} service...",
  host.Description.Name);
foreach(var ep in hostCalc.Description.Endpoints) {
  Console.WriteLine(" Endpoint: " + ep.Name);
  Console.WriteLine("  Address: " + ep.Address);
  Console.WriteLine("  Binding: " + ep.Binding);
  Console.WriteLine("  Contract: " +
    ep.Contract.ContractType);
}
```



F. 35

Duplex
Create the Client

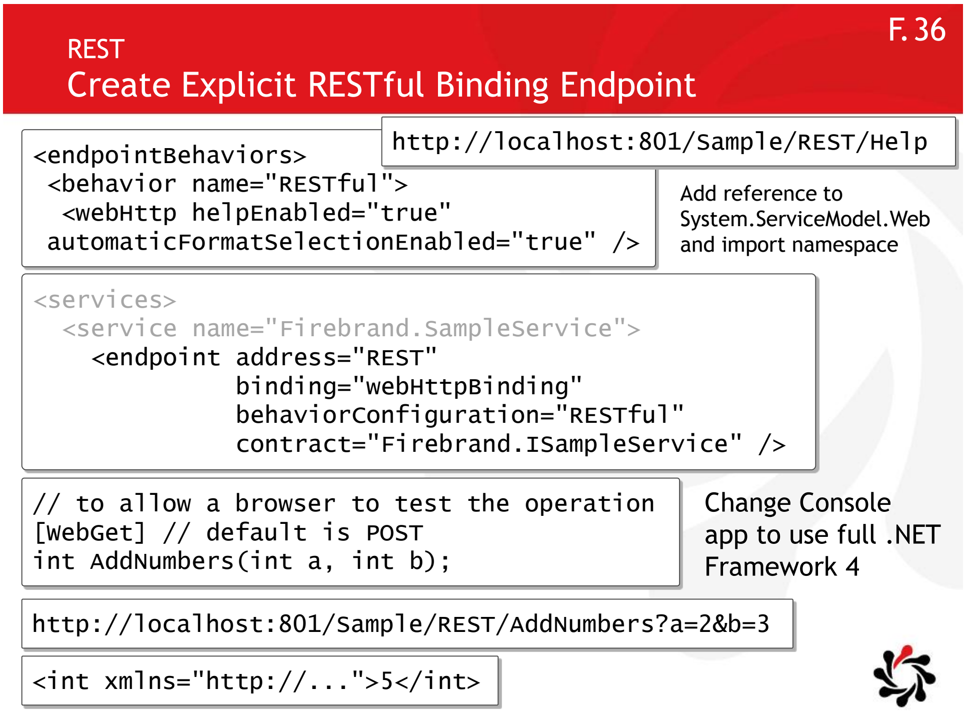
- Add a new WPF Application
  - Project Name: **C3.DuplexClient**
  - **Add Service Reference** and enter **http://10263a-svr1:801/Calc**
    - Expand the service to download the WSDL, click the interface
    - Namespace: **ServiceReference**, click OK

```
using System.ServiceModel;
Using C3.DuplexClient.ServiceReference;
```

Add a Button

```
public partial class MainWindow : Window, ICalcOneWayCallback {
  private CalcOneWayClient proxy;
  public void SendAnswer(int answer) {
    MessageBox.Show(answer.ToString());   }
```

```
// button1_Click
proxy = new CalcOneWayClient(new InstanceContext(this));
proxy.Multiply(3, 7);
```



F. 36

REST
Create Explicit RESTful Binding Endpoint

```
http://localhost:801/Sample/REST/Help
```

```
<endpointBehaviors>
 <behavior name="RESTful">
  <webHttp helpEnabled="true"
 automaticFormatSelectionEnabled="true" />
```

Add reference to System.ServiceModel.Web and import namespace

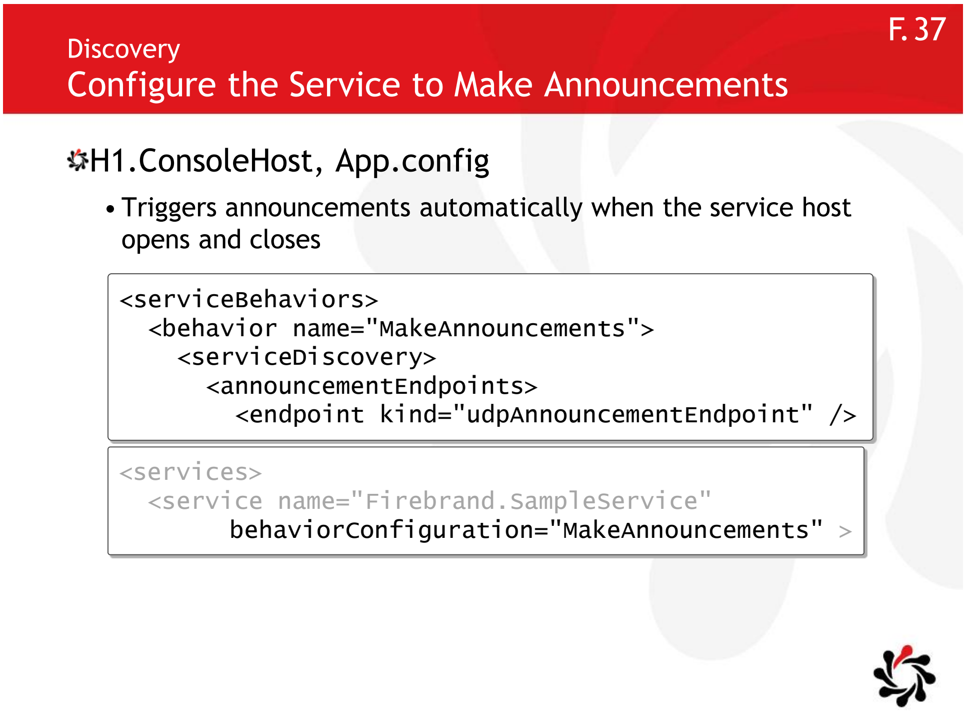
```
<services>
  <service name="Firebrand.SampleService">
    <endpoint address="REST"
              binding="webHttpBinding"
              behaviorConfiguration="RESTful"
              contract="Firebrand.ISampleService" />
```

```
// to allow a browser to test the operation
[WebGet] // default is POST
int AddNumbers(int a, int b);
```

Change Console app to use full .NET Framework 4

```
http://localhost:801/Sample/REST/AddNumbers?a=2&b=3
```

```
<int xmlns="http://...">5</int>
```



F. 37

Discovery

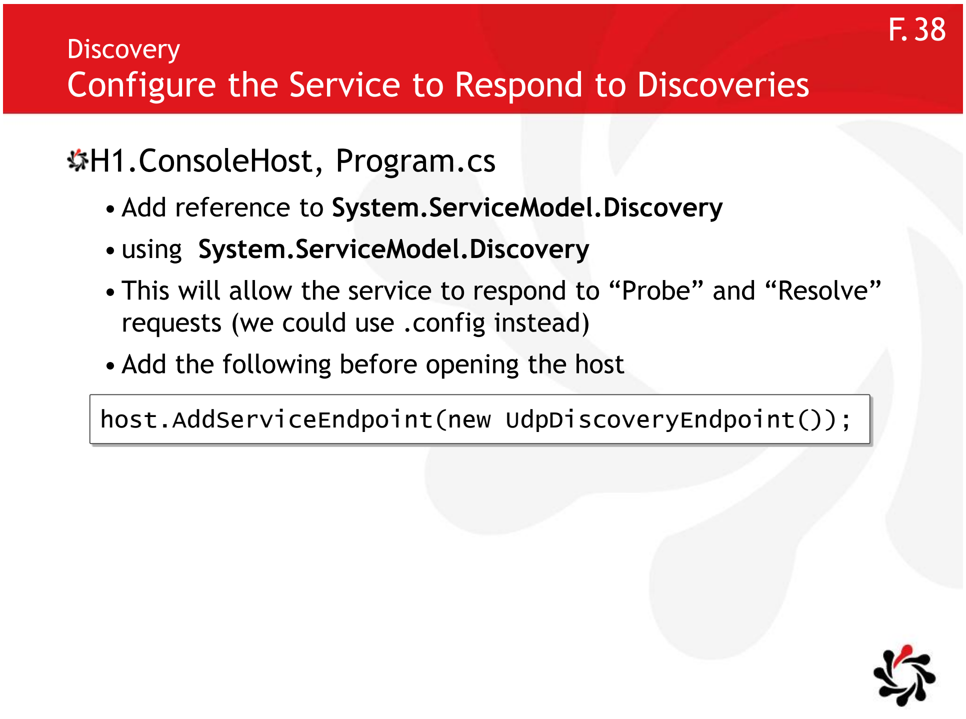
Configure the Service to Make Announcements

H1.ConsoleHost, App.config

- Triggers announcements automatically when the service host opens and closes

```
<serviceBehaviors>
  <behavior name="MakeAnnouncements">
    <serviceDiscovery>
      <announcementEndpoints>
        <endpoint kind="udpAnnouncementEndpoint" />
```

```
<services>
  <service name="Firebrand.SampleService"
        behaviorConfiguration="MakeAnnouncements" >
```



F. 38

Discovery

Configure the Service to Respond to Discoveries

H1.ConsoleHost, Program.cs

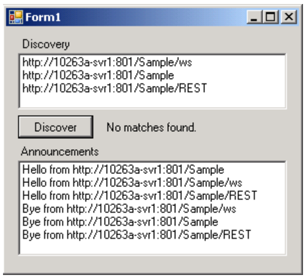
- Add reference to **System.ServiceModel.Discovery**
- using **System.ServiceModel.Discovery**
- This will allow the service to respond to “Probe” and “Resolve” requests (we could use .config instead)
- Add the following before opening the host

```
host.AddServiceEndpoint(new UdpDiscoveryEndpoint());
```

Discovery

Configure the Client

- Add a new Windows Forms Application
  - Project Name: **C4.ClientUsingDiscovery**
  - Add reference to **S2.ServiceContracts**, **System.ServiceModel**, **System.ServiceModel.Discovery**
  - Switch to target the full .NET Framework 4
  - Add two labels and list boxes, and one button and label



```
http://10263a-svr1:801/Sample/ws
http://10263a-svr1:801/Sample
http://10263a-svr1:801/Sample/REST
```

No matches found.

```
Hello from http://10263a-svr1:801/Sample
Hello from http://10263a-svr1:801/Sample/ws
Hello from http://10263a-svr1:801/Sample/REST
Bye from http://10263a-svr1:801/Sample/ws
Bye from http://10263a-svr1:801/Sample
Bye from http://10263a-svr1:801/Sample/REST
```

```
using System.ServiceModel;
using System.ServiceModel.Discovery;
using Firebrand;
using System.Collections.ObjectModel;
```

```
// fields in Form1 class
private AnnouncementService service
  = new AnnouncementService();
private ServiceHost host;
```

Discovery

Configure the Client (to Receive Announcements)

- Form1_Load (use Tab, Tab to generate handlers)

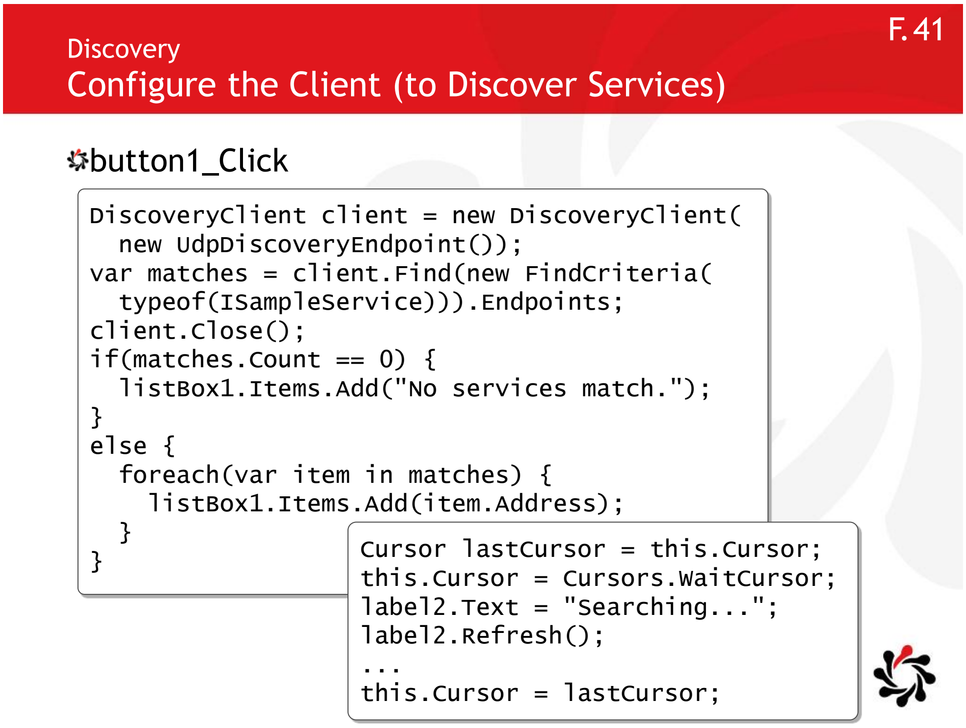
```
service.OnlineAnnouncementReceived += new ...
service.OfflineAnnouncementReceived += new ...
host = new ServiceHost(service);
host.AddServiceEndpoint(new UdpAnnouncementEndpoint());
host.Open();
```

- Online (and offline) event handlers

```
// service_OnlineAnnouncementReceived
listBox2.Items.Add("Hello from " +
  e.EndpointDiscoveryMetadata.Address);
```

```
// service_OfflineAnnouncementReceived
listBox2.Items.Add("Bye from " +
  e.EndpointDiscoveryMetadata.Address);
```

- Test by starting and stopping H1.ConsoleHost



Discovery

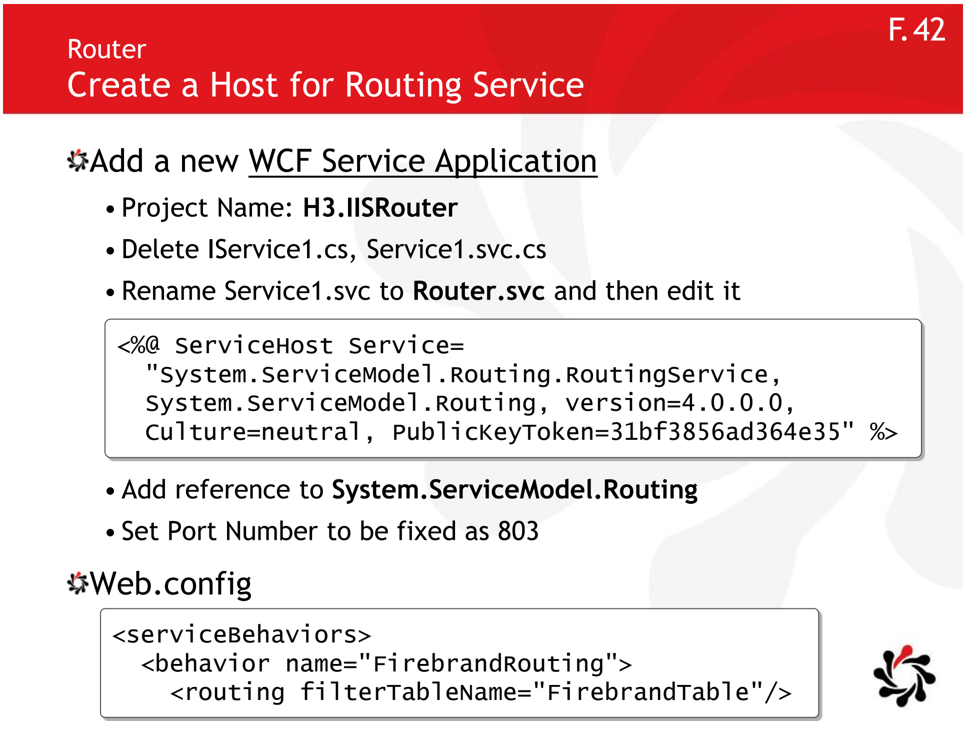
Configure the Client (to Discover Services)

F. 41

- button1_Click

```
DiscoveryClient client = new DiscoveryClient(
  new UdpDiscoveryEndpoint());
var matches = client.Find(new FindCriteria(
  typeof(ISampleService))).Endpoints;
client.Close();
if(matches.Count == 0) {
  listBox1.Items.Add("No services match.");
}
else {
  foreach(var item in matches) {
    listBox1.Items.Add(item.Address);
  }
}
```

```
Cursor lastCursor = this.Cursor;
this.Cursor = Cursors.WaitCursor;
label2.Text = "Searching...";
label2.Refresh();
...
this.Cursor = lastCursor;
```



Router

Create a Host for Routing Service

F. 42

- Add a new WCF Service Application
  - Project Name: **H3.IISRouter**
  - Delete IService1.cs, Service1.svc.cs
  - Rename Service1.svc to **Router.svc** and then edit it

```
<%@ ServiceHost Service=
  "System.ServiceModel.Routing.RoutingService,
  System.ServiceModel.Routing, version=4.0.0.0,
  Culture=neutral, PublicKeyToken=31bf3856ad364e35" %>
```

  - Add reference to **System.ServiceModel.Routing**
  - Set Port Number to be fixed as 803
- Web.config

```
<serviceBehaviors>
  <behavior name="FirebrandRouting">
    <routing filterTableName="FirebrandTable"/>
```

F. 43

Router

Configure the Filters and Backup Lists

```
<routing>
  <filters>
    <filter name="FirebrandFilter" filterType="MatchAll" />
  </filters>
  <filterTables>
    <filterTable name="FirebrandTable">
      <add filterName="FirebrandFilter"
           endpointName="Service801"
           backupList="FirebrandList" />
    </filterTable>
  </filterTables>
```

Visual Studio 2010 warns that the name attribute is not allowed but it does work

```
  <backupLists>
    <backupList name="FirebrandList">
      <add endpointName="Service802" />
    </backupList>
  </backupLists>
</routing>
```

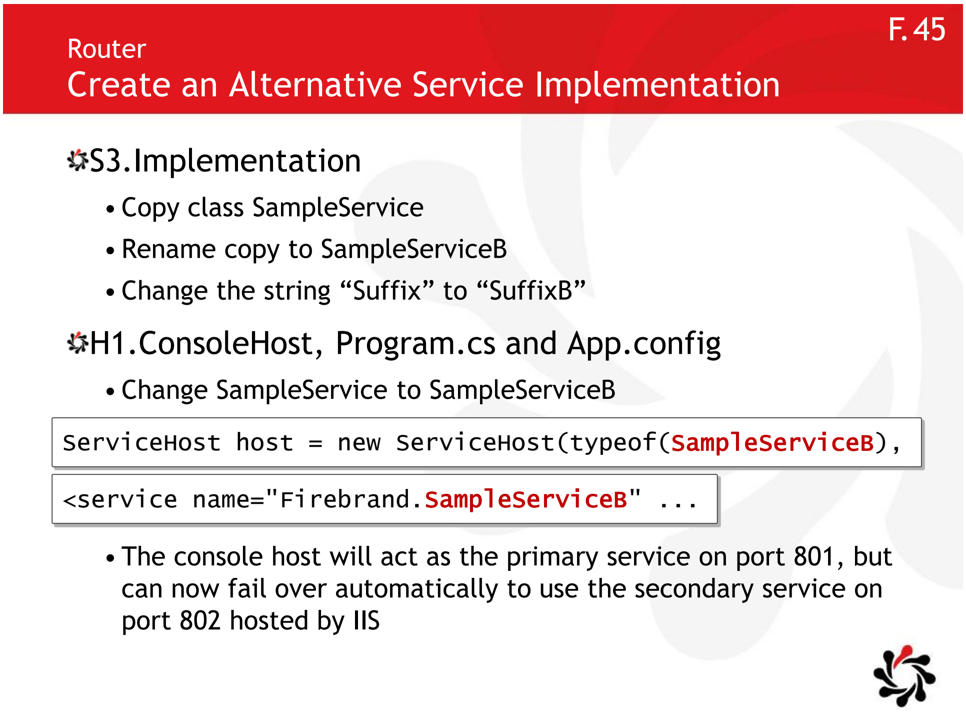
F. 44

Router

Configure the Endpoints

```
<services>
  <service name="System.ServiceModel.Routing.RoutingService"
    behaviorConfiguration="FirebrandRouting">
    <endpoint address="" name="requestReplyEndpoint"
      binding="basicHttpBinding" contract=
        "System.ServiceModel.Routing.IRequestReplyRouter" />
    <endpoint address="OneWay" name="oneWayEndpoint"
      binding="basicHttpBinding" contract=
    "System.ServiceModel.Routing.ISimplexDatagramRouter" />
  </service>
</services>
```

```
<client>
  <endpoint address="http://localhost:801/Sample/ws"
    binding="wsHttpBinding"
    contract="*" name="Service801" />
  <endpoint address="http://localhost:802/Sample.svc"
    binding="basicHttpBinding"
    contract="*" name="Service802" />
```



F. 45

Router

Create an Alternative Service Implementation

S3.Implementation

- Copy class SampleService
- Rename copy to SampleServiceB
- Change the string “Suffix” to “SuffixB”

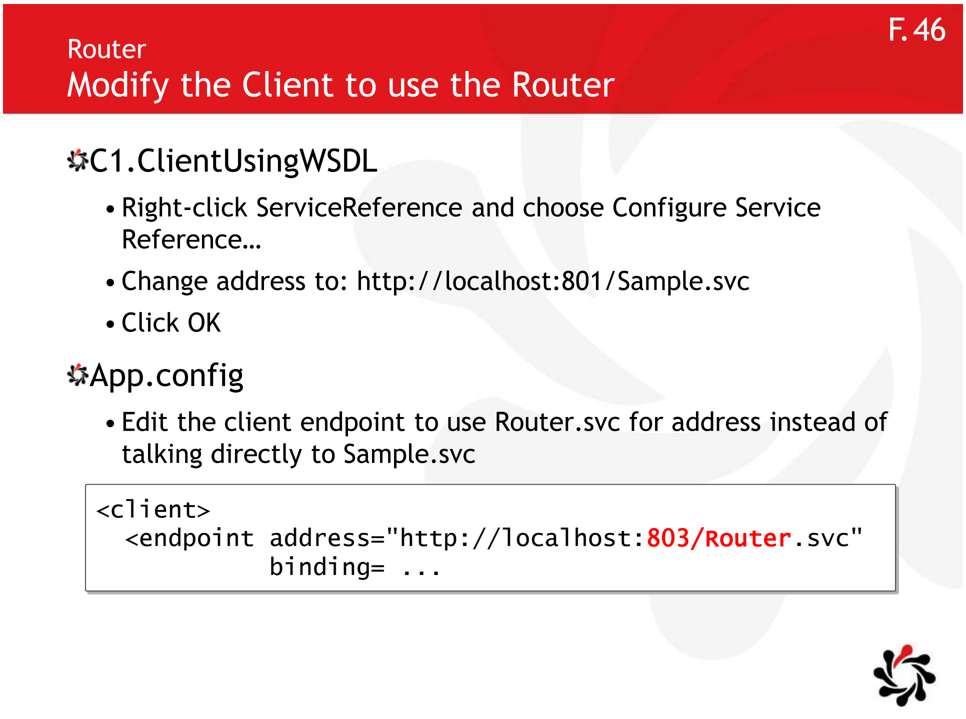
H1.ConsoleHost, Program.cs and App.config

- Change SampleService to SampleServiceB

```
ServiceHost host = new ServiceHost(typeof(SampleServiceB),
```

```
<service name="Firebrand.SampleServiceB" ...
```

- The console host will act as the primary service on port 801, but can now fail over automatically to use the secondary service on port 802 hosted by IIS



F. 46

Router

Modify the Client to use the Router

C1.ClientUsingWSDL

- Right-click ServiceReference and choose Configure Service Reference…
- Change address to: http://localhost:801/Sample.svc
- Click OK

App.config

- Edit the client endpoint to use Router.svc for address instead of talking directly to Sample.svc

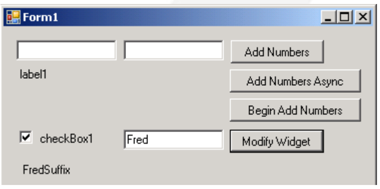
```
<client>
  <endpoint address="http://localhost:803/Router.svc"
            binding= ...
```

F. 47

Router

Test the Router

- Build the Solution
- Ensure the services are running
  - H1.ConsoleHost: start without debugger (**Ctrl+F5**)
  - H2.IISHost: right-click **Sample.svc**, View in Browser
  - H3.IISRouter: right-click **Router.svc**, View in Browser
- Start C1.ClientUsingWSDL
  - Click Modify Widget: name will be suffixed with “SuffixB”
  - Stop the H1.ConsoleHost
  - Click Modify Widget: name will be suffixed with “Suffix”



☑ checkBox1

Fred

FredSuffix

F. 48

Sessions

Define Operations That Use Sessions

- S2.ServiceContracts

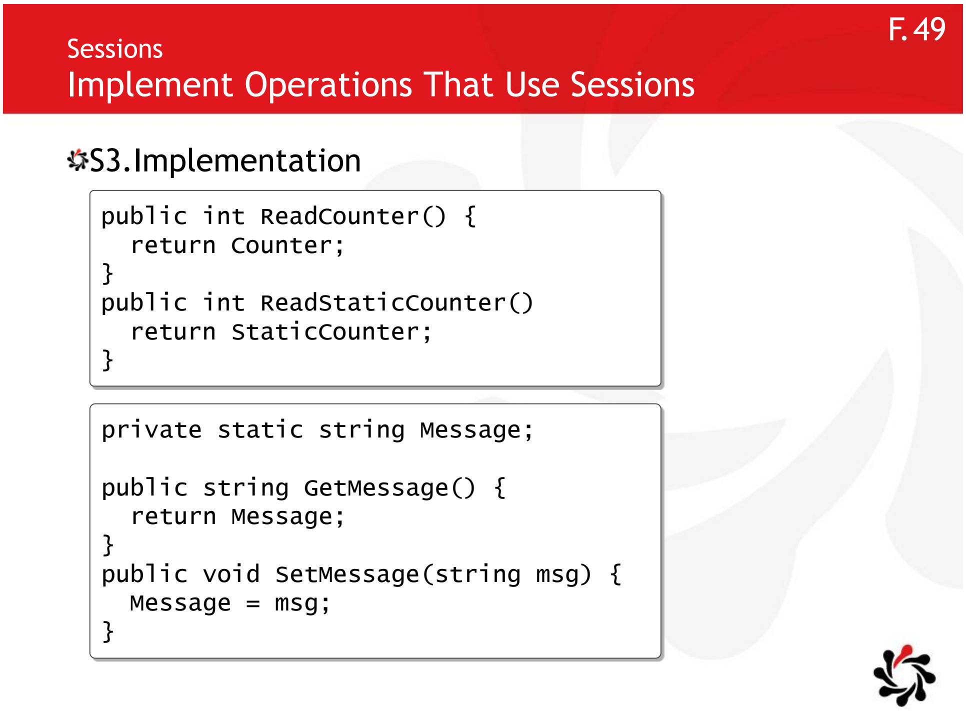
```
[OperationContract]
int ReadCounter();
[OperationContract]
int ReadStaticCounter();
[OperationContract]
string GetMessage();
[OperationContract]
void SetMessage(string message);
```

- S3.Implementation

```
private int Counter;
private static int StaticCounter;
```

  - Add the following to the AddNumbers method

```
Interlocked.Increment(ref Counter);
Interlocked.Increment(ref StaticCounter);
```



F. 49

Sessions

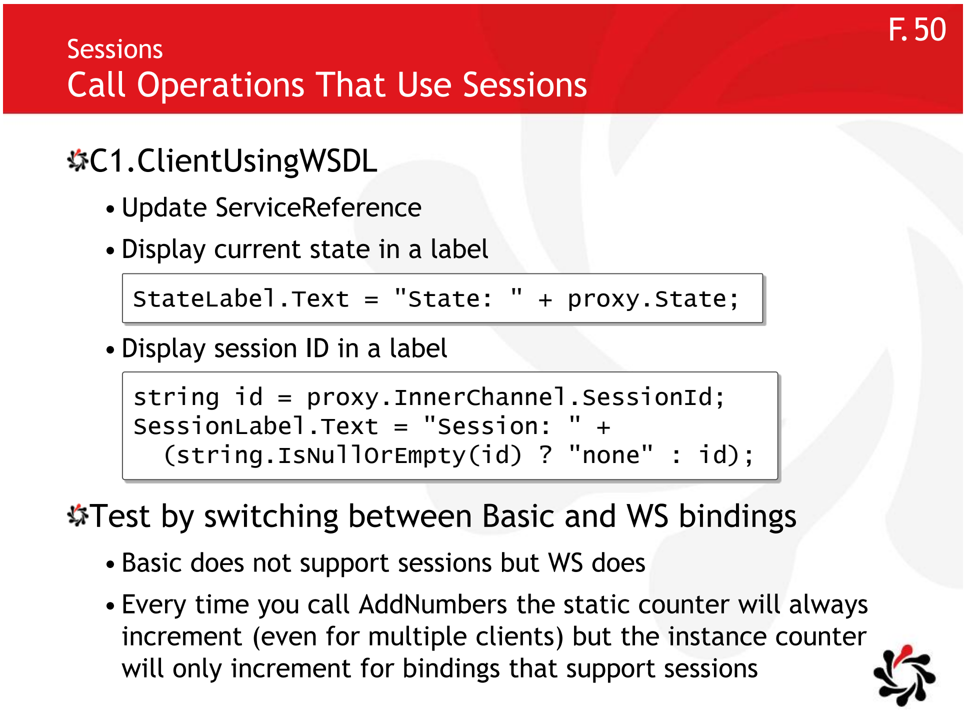
Implement Operations That Use Sessions

S3.Implementation

```
public int ReadCounter() {
  return Counter;
}
public int ReadStaticCounter()
  return StaticCounter;
}
```

```
private static string Message;

public string GetMessage() {
  return Message;
}
public void SetMessage(string msg) {
  Message = msg;
}
```



F. 50

Sessions

Call Operations That Use Sessions

C1.ClientUsingWSDL

- Update ServiceReference
- Display current state in a label

```
StateLabel.Text = "State: " + proxy.State;
```

- Display session ID in a label

```
string id = proxy.InnerChannel.SessionId;
SessionLabel.Text = "Session: " +
  (string.IsNullOrEmpty(id) ? "none" : id);
```

Test by switching between Basic and WS bindings

- Basic does not support sessions but WS does
- Every time you call AddNumbers the static counter will always increment (even for multiple clients) but the instance counter will only increment for bindings that support sessions

Appendix G
What's New in Visual Studio 2013 and Updated Exam

Developing Windows Azure and Web Services

Updated 11th April 2014

What's New in Visual Studio 2013 and Updated Exam

Changes for EF and WCF

Exam Topics: Query and manipulate data by using the Entity Framework
- □ Perform asynchronous operations using Entity Framework
- □ Map a stored procedure

Exam Topics: Query and manipulate data by using Data Provider for Entity Framework
- □ Programmatically configure a Data Provider

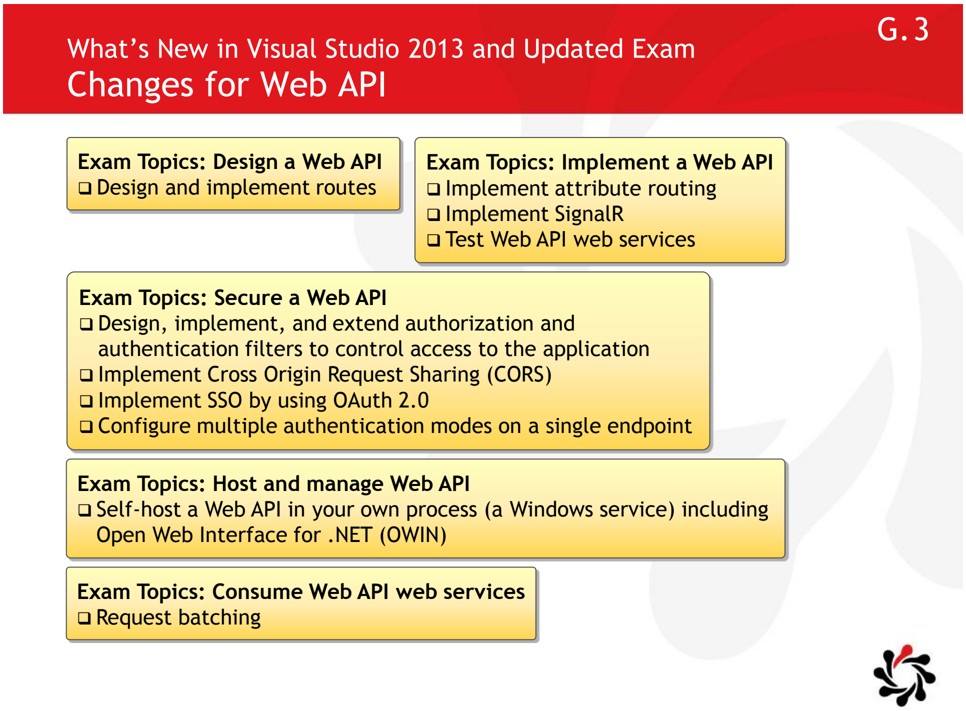
Exam Topics: Query data by using LINQ to Entities
- □ Log queries and database commands
- □ Implement async query

Most of these topics have already been covered in the main slides

Exam Topics: Create an Entity Framework data model
- □ Describe a data model by using Custom Code First Conventions

Exam Topics: Configure WCF services by using configuration settings
- □ Configure bindings (including WebSocket bindings)
- □ Configure message compressions and encoding

Exam Topics: Secure a WCF service
- □ Design and implement multiple authentication modes



G.3

What’s New in Visual Studio 2013 and Updated Exam

Changes for Web API

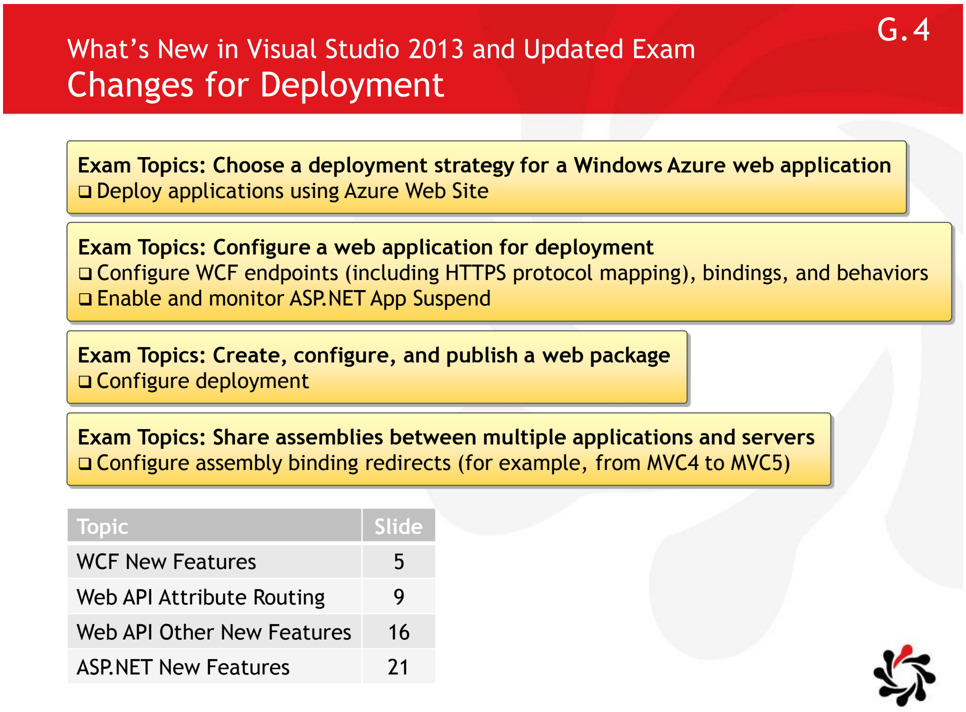
Exam Topics: Design a Web API
□ Design and implement routes

Exam Topics: Implement a Web API
□ Implement attribute routing
□ Implement SignalR
□ Test Web API web services

Exam Topics: Secure a Web API
□ Design, implement, and extend authorization and authentication filters to control access to the application
□ Implement Cross Origin Request Sharing (CORS)
□ Implement SSO by using OAuth 2.0
□ Configure multiple authentication modes on a single endpoint

Exam Topics: Host and manage Web API
□ Self-host a Web API in your own process (a Windows service) including Open Web Interface for .NET (OWIN)

Exam Topics: Consume Web API web services
□ Request batching



G.4

What’s New in Visual Studio 2013 and Updated Exam

Changes for Deployment

Exam Topics: Choose a deployment strategy for a Windows Azure web application
□ Deploy applications using Azure Web Site

Exam Topics: Configure a web application for deployment
□ Configure WCF endpoints (including HTTPS protocol mapping), bindings, and behaviors
□ Enable and monitor ASP.NET App Suspend

Exam Topics: Create, configure, and publish a web package
□ Configure deployment

Exam Topics: Share assemblies between multiple applications and servers
□ Configure assembly binding redirects (for example, from MVC4 to MVC5)

Topic	Slide
WCF New Features	5
Web API Attribute Routing	9
Web API Other New Features	16
ASP.NET New Features	21

WCF New Features

Miscellaneous Improvements

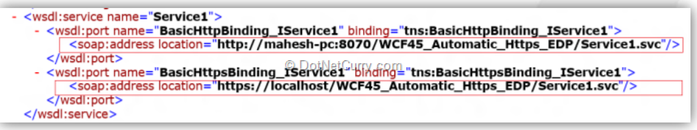
- Improved Intellisense Support
- Support for UDP Endpoint (udpBinding)
  - Typically TCP is used in non-time critical (may be non-real-time) applications but UDP is very useful for real-time applications
  - TCP performs data-packets checking for the order of the message to be send, it is slower than UDP
  - In UDP, packet delivery is not guaranteed and loss or out-of-arrival of packets is acceptable

What's New in WCF 4.5
`http://www.dotnetcurry.com/showarticle.aspx?ID=866`

WCF New Features

HTTPS Protocol Mapping

- In WCF 4.0
  - Default endpoint defines protocol mapping for basicHttpBinding
  - Although we had SSL support in previous versions of WCF, but it was necessary to declare an explicit endpoint
- In WCF 4.5
  - If IIS is enabled for SSL and if the service does not have any explicit endpoint defined for the specific binding, then WCF can be hosted on IIS with HTTPS enabled i.e. basicHttpsBinding
  - To experience the automatic Https protocol mapping on IIS, you need to create an Application Pool with Identity as LocalSystem



```
- <wsdl:service name="Service1">
  - <wsdl:port name="BasicHttpBinding_IService1" binding="tns:BasicHttpBinding_IService1">
      <soap:address location="http://mahesh-pc:8070/WCF45_Automatic_Https_EDP/Service1.svc"/>
    </wsdl:port>
  - <wsdl:port name="BasicHttpsBinding_IService1" binding="tns:BasicHttpsBinding_IService1">
      <soap:address location="https://localhost/WCF45_Automatic_Https_EDP/Service1.svc"/>
    </wsdl:port>
  </wsdl:service>
```

G.7

WCF New Features

Web Socket Binding

- The support of WebSocket in WCF 4.5 is achieved through NetHttpBinding and NetHttpsBinding
  - WebSocket is the new protocol for bidirectional communication over port 80 and 443
  - NetHttpBinding with WebSocket can replace the use of WsDualHttpBinding, since WebSocket provides a duplex channel which also supports sessions
  - This is better than WsDualHttpBinding which uses two channels, and require the use of WS-ReliableMessaging for session management

```
<endpoint address="http://localhost:8083"
          binding="netHttpBinding"
          contract="Contracts.IDuplexContract"/>
```

What's new in WCF 4.5? WebSocket support (Part 1 of 2)
http://www.codeproject.com/Articles/338789/What-s-new-in-WCF-WebSocket-support-Part-of

G.8

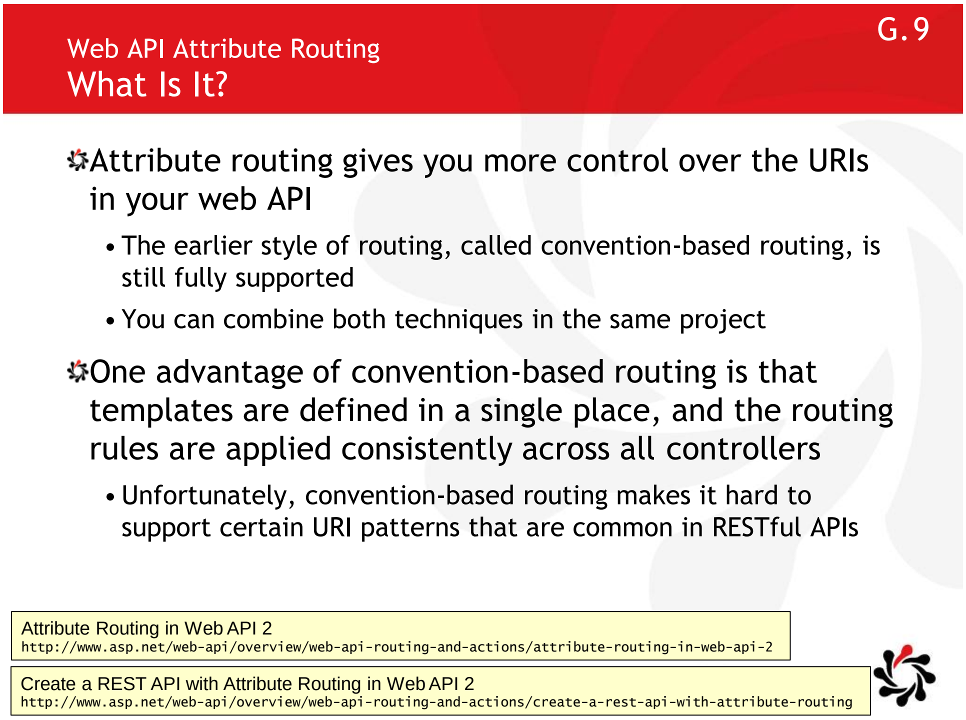
WCF New Features

Authentication

- WCF 4.5 enabled you to inherit the authentication types from IIS, so you only need to declare them once
  - Set the clientCredentialType attribute to InheritedFromHost

```
<services>
  <service name="MyService">
    <endpoint address="" binding="basicHttpBinding"
              contract="Mycontract" bindingConfiguration="secured"/>
  </service>
</services>
<bindings>
  <basicHttpBinding>
    <binding name="secured">
      <security mode="Transport">
        <transport clientCredentialType="InheritedFromHost" />
```

What's new in WCF 4.5? Multiple authentication support on a single endpoint in IIS
http://blogs.microsoft.co.il/idof/2011/09/25/whats-new-in-wcf-45-multiple-authentication-support-on-a-single-endpoint-in-iis/



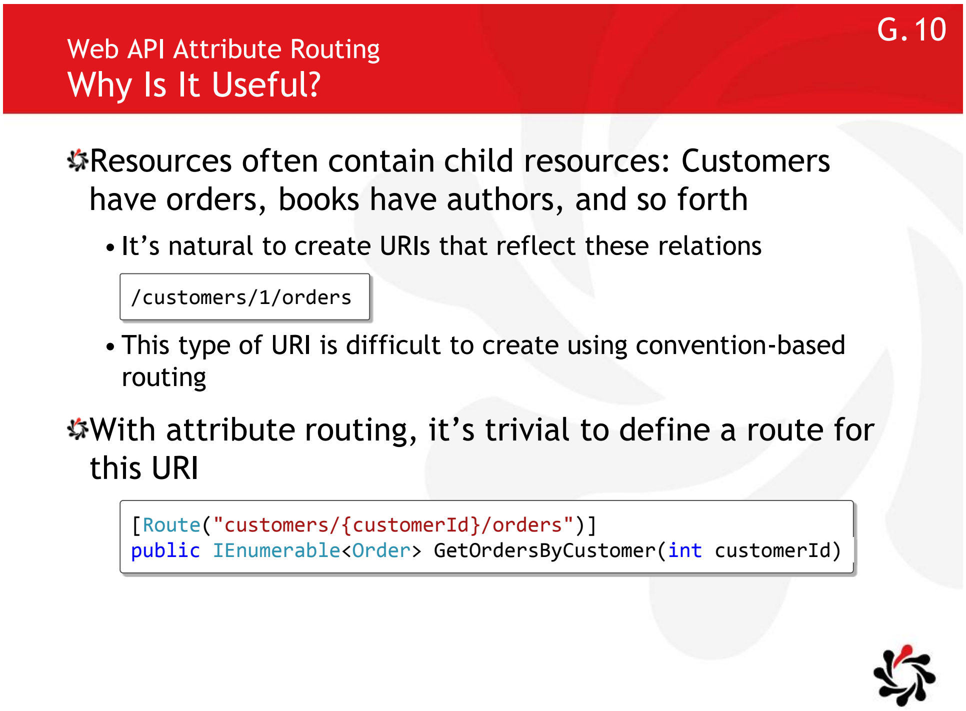
G.9

Web API Attribute Routing
What Is It?

- Attribute routing gives you more control over the URIs in your web API
  - The earlier style of routing, called convention-based routing, is still fully supported
  - You can combine both techniques in the same project
- One advantage of convention-based routing is that templates are defined in a single place, and the routing rules are applied consistently across all controllers
  - Unfortunately, convention-based routing makes it hard to support certain URI patterns that are common in RESTful APIs

Attribute Routing in Web API 2
http://www.asp.net/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2

Create a REST API with Attribute Routing in Web API 2
http://www.asp.net/web-api/overview/web-api-routing-and-actions/create-a-rest-api-with-attribute-routing



G.10

Web API Attribute Routing
Why Is It Useful?

- Resources often contain child resources: Customers have orders, books have authors, and so forth
  - It's natural to create URIs that reflect these relations

```
/customers/1/orders
```

  - This type of URI is difficult to create using convention-based routing
- With attribute routing, it's trivial to define a route for this URI

```
[Route("customers/{customerId}/orders")]
public IEnumerable<Order> GetOrdersByCustomer(int customerId)
```

G.11

Web API Attribute Routing
How to Enable Attribute Routing

- Call MapHttpAttributeRoutes during configuration

```
using System.Web.Http;
```

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        // Web API configuration and services

        // Web API routes
        config.MapHttpAttributeRoutes();

        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );
    }
}
```

G.12

Web API Attribute Routing
Route Prefixes

- You can set a common prefix for an entire controller

```
[RoutePrefix("api/books")]
public class BooksController : ApiController
{
    // GET api/books
    [Route("")]
    public IEnumerable<Book> Get() { ... }

    // GET api/books/5
    [Route("{id:int}")]
    public Book Get(int id) { ... }

    // POST api/books
    [Route("")]
    public HttpResponseMessage Post(Book book) { ... }

    // GET /api/authors/1/books
    [Route("~/api/authors/{authorId:int}/books")]
    public IEnumerable<Book> GetByAuthor(int authorId) { ... }
}
```

G.13

Web API Attribute Routing
Route Constraints

- The general syntax is "{parameter:constraint}“
  - Here, the first route will only be selected if the "id" segment of the URI is an integer
  - Otherwise, the second route will be chosen

```
[Route("users/{id:int}"]
public User GetUserById(int id)
```

```
[Route("users/{name}"]
public User GetUserByName(string name)
```

- Some, such as min, take arguments in parentheses

```
[Route("users/{id:int:min(1)}")]
public User GetUserById(int id)
```

Route Constraints
http://www.asp.net/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2#constraints

G.14

Web API Attribute Routing
Custom Route Constraints

```
using System.Web.Http.Routing;
```

- Implement the IHttpRouteConstraint interface

```
public class NonZeroConstraint : IHttpRouteConstraint
{
    public bool Match(HttpRequestMessage request, IHttpRoute route,
        string parameterName, IDictionary<string, object> values,
        HttpRouteDirection routeDirection)
```

- Register the route constraint

```
var constraintResolver = new DefaultInlineConstraintResolver();
constraintResolver.ConstraintMap.Add(
    "nonzero", typeof(NonZeroConstraint));
config.MapHttpAttributeRoutes(constraintResolver);
```

- Apply the route constraint

```
[Route("{id:nonzero}")]
public HttpResponseMessage GetNonZero(int id)
```

G.15

Web API Attribute Routing

Route Name and Order

- Setting a route name

```
[Route("api/books/{id}", Name="GetBookById")]
public BookDto GetBook(int id)
```

- Setting a route order (default is zero)

```
[RoutePrefix("orders")]
public class OrdersController : ApiController
{
    [Route("{id:int}")] // constrained parameter (2nd)
    public HttpResponseMessage Get(int id) { ... }
    [Route("details")]  // literal (1st)
    public HttpResponseMessage GetDetails() { ... }
    [Route("pending", RouteOrder = 1)] // (would be 2nd, but now 5th)
    public HttpResponseMessage GetPending() { ... }
    [Route("{customerName}")]  // unconstrained parameter (3rd)
    public HttpResponseMessage GetByCustomer(string customerName) { ... }
    [Route("{*date:datetime}")]  // wildcard (4th)
    public HttpResponseMessage Get(DateTime date) { ... }
}
```

G.16

Web API Other New Features

Request Batching

```
using System.Web.Http.Batch;
```

- Useful way of minimizing the number of messages that are passed between the client and the server
  - To enable batching in general, use custom message handlers (DefaultHttpBatchHandler, DefaultODataBatchHandler) which you can register per-route to handle the batch requests

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

- On the client side you can use the existing Web API client library to submit a batch request

Web API Batching
http://aspnetwebstack.codeplex.com/wikipage?title=Web+API+Request+Batching

G.17

Web API Other New Features

SSO with OAuth 2.0

- To authenticate a simple http request in a Web API service you have to send a token in the http authorization header
- When Microsoft released Web API it did not provide any tool or api to handle authentication
  - This gap was immediately filled by open source providers such as Dominic Baeir see Thinktecture.IdentityModel
  - Microsoft now has the JWT Token handler that should be used to validate JWT tokens on the web api application
  - You can include it by importing the NuGet Package: JSON Web Token Handler for Microsoft .Net framework 4.5

How to Authenticate Web API using OAuth 2.0
`http://blogs.microsoft.co.il/applisec/2013/10/15/how-to-authenticate-web-api-using-oauth-20/`

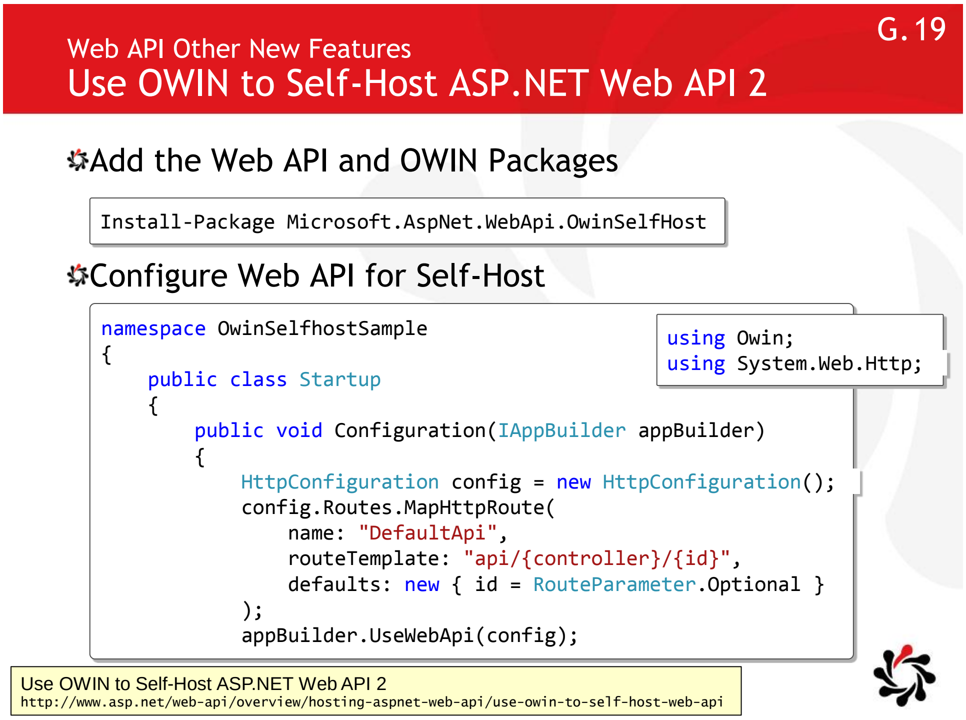
G.18

Web API Other New Features

The Open Web Interface for .NET (OWIN)

- OWIN defines a standard interface between .NET web servers and web applications
  - The goal of the OWIN interface is to decouple server and application
  - The Katana project represents the set of OWIN components that, while still open source, are built and released by Microsoft
- There are three hosting options for Katana
  - IIS/ASP.NET, Custom Host, OwinHost.exe
- Katana includes two server implementations
  - Microsoft.Owin.Host.SystemWeb
  - Microsoft.Owin.Host.HttpListener

An Overview of Project Katana
`http://www.asp.net/aspnet/overview/owin-and-katana/an-overview-of-project-katana`



G.19

Web API Other New Features

Use OWIN to Self-Host ASP.NET Web API 2

Add the Web API and OWIN Packages

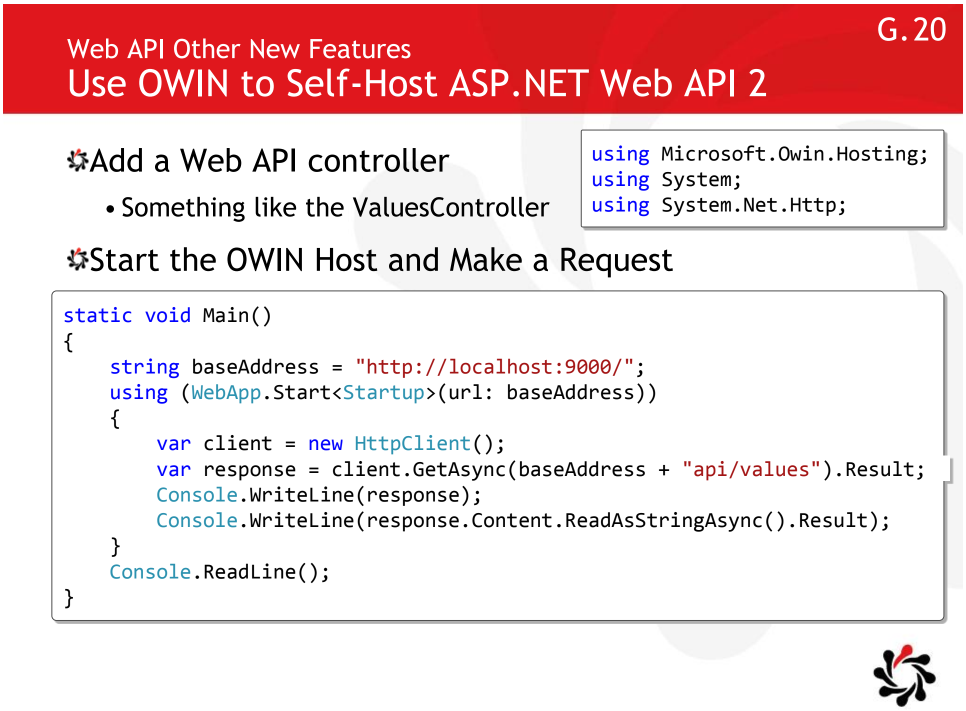
```
Install-Package Microsoft.AspNet.WebApi.OwinSelfHost
```

Configure Web API for Self-Host

```
using Owin;
using System.Web.Http;
```

```
namespace OwinSelfhostSample
{
    public class Startup
    {
        public void Configuration(IAppBuilder appBuilder)
        {
            HttpConfiguration config = new HttpConfiguration();
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
            appBuilder.UseWebApi(config);
```

Use OWIN to Self-Host ASP.NET Web API 2
http://www.asp.net/web-api/overview/hosting-aspnet-web-api/use-owin-to-self-host-web-api



G.20

Web API Other New Features

Use OWIN to Self-Host ASP.NET Web API 2

Add a Web API controller

- Something like the ValuesController

```
using Microsoft.Owin.Hosting;
using System;
using System.Net.Http;
```

Start the OWIN Host and Make a Request

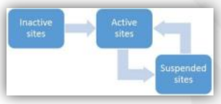
```
static void Main()
{
    string baseAddress = "http://localhost:9000/";
    using (WebApp.Start<Startup>(url: baseAddress))
    {
        var client = new HttpClient();
        var response = client.GetAsync(baseAddress + "api/values").Result;
        Console.WriteLine(response);
        Console.WriteLine(response.Content.ReadAsStringAsync().Result);
    }
    Console.ReadLine();
}
```

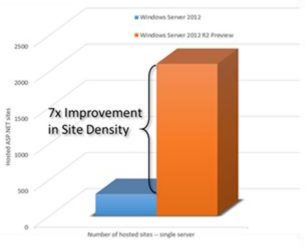
ASP.NET New Features

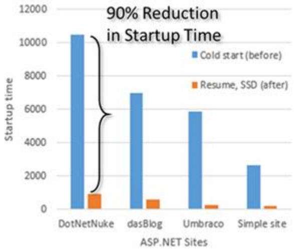
App Suspend

G.21

- Radically changes the economic model for hosting large numbers of ASP.NET sites on a server
  - Built on top of a new feature in Windows Server 2012 R2 called IIS Idle Worker Process Page-out
  - It's the cloud version of the app suspend (AKA tombstoning) that you see in Windows Phone and Windows Store apps







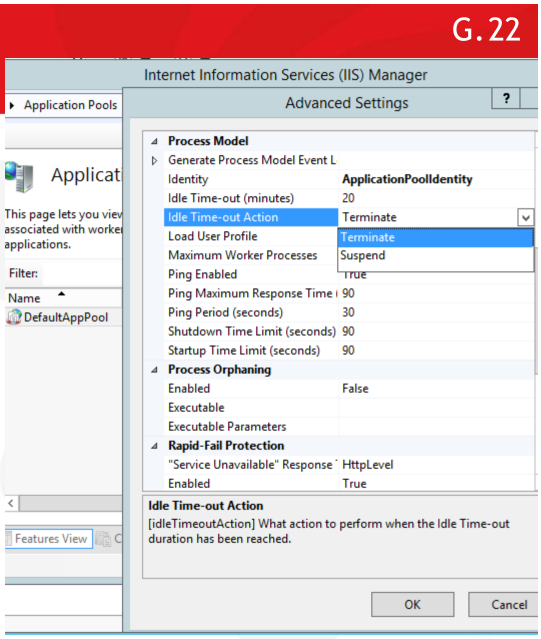
ASP.NET App Suspend – responsive shared .NET web hosting
http://blogs.msdn.com/b/dotnet/archive/2013/10/09/asp-net-app-suspend-responsive-shared-net-web-hosting.aspx

ASP.NET New Features

App Suspend

G.22

- App Suspend is a new setting in IIS configuration, available on each application pool
- You can validate that an app was suspended in the event viewer, in the Application event log
  - Search for event 2310



This page lets you view
associated with worke
applications.

DefaultAppPool

Process Model	
Generate Process Model Event L	
Identity	**ApplicationPoolIdentity**
Idle Time-out (minutes)	20
Idle Time-out Action	Terminate Terminate Suspend
Load User Profile	
Maximum Worker Processes	
Ping Enabled	True
Ping Maximum Response Time (	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90
Process Orphaning	
Enabled	False
Executable	
Executable Parameters	
Rapid-Fail Protection	
"Service Unavailable" Response '	HttpLevel
Enabled	True

Idle Time-out Action
[idleTimeoutAction] What action to perform when the Idle Time-out duration has been reached.

Enable and monitor ASP.NET App Suspend on Windows Server 2012 R2
http://blogs.msdn.com/b/webdev/archive/2013/10/09/enable-and-monitor-asp-net-app-suspend-on-windows-server-2012-r2.aspx