Session 9

Prerequisite: Interfaces

Interface Types

- **Not an implementation, only a contract for others**
- **Cannot be instantiated**
- **Defines the signatures of methods**
- **Concrete types will “implement” an interface**
- **Can implement explicitly**
- **Explicit Interface:**
  - Qualify member with interface name
  - Cannot access interface member via class instance
  - Cast class to interface type to access explicit members

```
Truck t = new Truck();
t.Price = 0; //invalid!

IVehicle v = t as IVehicle;
v.Price = 0; //okay
```

Interface example

```
public interface IVehicle
  {
      string Make { get; set; }
      string Model { get; set; }
      ColorEnum Color { get; set; }
       int Year {get; set; }
       int Price {get; set; }
  }
```

```
6  namespace AutoSales.Library
7  {
8      public enum VehicleColor...
18
19     public interface IVehicle...
27
28     public class Truck : IVehicle
29     {
30         public string Make { get; set; }
31
32         public string Model { get; set; }
33
34         public VehicleColor Color { get; set; }
35
36         private int _year;
37         public int Year
38         {
39             get { return _year; }
40             set
41             {
42                 if (value < 1950 || value > DateTime.Now.Year)
43                     throw new ApplicationException("Year value out of range");
44                 _year = value;
45             }
46         }
47
48         private int _price;
49         public int Price...
59     }
60 }
```

Session 9

WCF – Windows Communication Foundation

Web Service

- **Part of WCF**
- **Enables communications across organizational and platform boundries**
- **Usually hosted by a web server**
- **Usually use HTTP/HTTPS**

WCF Service

- **System.ServiceModel namespace**
- **Collection of Endpoints**
- **Client applications access services through endpoints**
- **Endpoint has:**
  - Address – *where* it resides
  - Binding – *how* it communicates including:
    - Transport protocol (HTTP, TCP)
    - Encoding (text, binary)
    - Security requirements (SSL, SOAP message security)
  - Contract – what is communicated – collection of messages organized into operations.

WCF Service

- **Comprised of:**
  - Service Contract (WSDL)
  - Data Contract (XSD)
  - Message Contract (SOAP)
  - Fault Contract (exceptions)

Service Contract

- **Describe what operations the service can perform**
- **Aka list of all the callable methods**
- **Apply ServiceContractAttribute to the class or interface (preferred)**
- **There are two types of Service Contract**
  - ServiceContract – the class
  - OperationContract – the method

```
[ServiceContract]
public interface IVehicleService
{
    // Define the OperationContact here….
}
```

Operation Contract

- **Identifies a method as an operation of the service**
- **Apply OperationContractAttribute to the methods which are part of the service**
- **It is best to do this in the interface definition, not the class definition**

```
[ServiceContract]
public interface IVehicleService
{
    [OperationContract]
    bool Save(Vehicle vehicle);
}
```

DataContract

- **Defines which data types are passed to and from the service**
- **WCF defines implicit contracts for built-in (intrinsic) types**
- **Two types of Data Contract**
  - DataContract – the class
  - DataMember – the properties

```
[DataContract]
public class Vehicle
{
    …
}
```

DataMember

- **Identifies which class members are exposed**

```
[DataContract]
public class Vehicle
{
    string _make;

    [DataMember]
    public string Make { get; set; }

    [DataMember]
    public string Model { get; set; }
}
```

Fault Contract

- **Defines which errors are raised by the service**
- **Apply FaultContractAttribute to class**

Consuming Web Services

- **Add a Web Reference**
- **Visual Studio automatically creates proxy classes**
- **Proxy classes can be created manually using the WSDL tool (wsdl.exe)**
  - `wsdl /language:CS /out:WebProxy.cs WebService.wsdl`