**接口**

* 定义一个接口是为了遵循同一种规范，便于程序的扩展。
* 关键字
  + Interface
  + public
  + abstract

**理解契约式编程**

* 契约合同能保障双方的利益，对客户来说，合同规定了供应者要做的工作；对供应者来说，合同说明了如果约定的条件不满足，供应者没有义务一定要完成规定的任务。该道理同样也适用于软件. 所以,契约式编程是编程的一种方法。
* 引入契约观念之后，这种Client 与 Server 关系被打破，大家都是平等的，你需要我正确提供服务，那么你必须满足我提出的条件，否则我没有义务“排除万难”地保证完成任务。

**服务契约**

* 服务契约描述了暴露给外部的类型（接口或类）、服务所支持的操作、使用的消息交换模式和消息的格式。每个WCF服务必须实现至少一个服务契约。使用服务契约必须要引用命名空间System.ServiceModel 。
* ServiceContractAttribute
* OperationContractAttribute

**数据契约**

* 大部分的系统都是以数据为中心的（Data Central），数据是信息的载体，一个分布式的互联系统关注于数据的交换，而数据正常交换的根本前提是参与数据交换的双方对于数据结构的一致性理解。这就为数据的表现提出了要求，为了保证处于不同平台、不同厂商的应用能够正常地进行数据交换，交换的数据必须采用一种大家都能够理解的展现方式。在这方面，XML无疑是最好的选择。所以WCF下的序列化（Serialization）解决的就是如何将数据从对象的表现形式转变成XML表现形式，以确保数据的正常交换。
* DataContractAttribute和DataMemberAttribute
* 对于应用了DataContractAttribute特性的类型，只有应用了DataMemberAttribute特性的字段或者属性成员才能成为数据契约的数据成员。
* DataContractSerializer能自动序列化任何实现了System.Runtime.Serialization.ISerializable接口的.NET类型。
* 通过使用DataContract和DataMember特性，可以非常简单地让WCF的DataContractSerializer将开发者定制的数据类型序列化。
* WCF对DataContractSerializer的调用被隐藏起来了.

例子：用控制台输出的方式进行讲解（该序列化过程是隐藏起来的）

namespace ConsoleApplication1

{

[DataContract]

class book

{

private int bookID;

[DataMember]

public int BookID

{

get;

set;

}

private string bookName;

[DataMember]

public string BookName

{

get;

set;

}

private double price;

[DataMember]

public double Price

{

get;

set;

}

}

class Program

{

static void Main(string[] args)

{

DataContractSerializer serializer = new DataContractSerializer(typeof(book));

MemoryStream ms = new MemoryStream();

book b=new book();

b.BookID=1001;

b.BookName="<wcf进阶>";

b.Price=50.5;

serializer.WriteObject(ms,b); //序列化

//serializer.ReadObject(); //反序列化

Console.WriteLine(UnicodeEncoding.UTF8.GetChars(ms.ToArray()));

Console.ReadKey();

}

}

}

**消息契约**消息契约用来控制消息头,和消息体; 当序列化一个对象并生成消息的时候，希望将部分数据成员作为SOAP的报头，部分作为消息的主体。

* MessageContractAttribute
* MessageHeaderAttribute
* MessageBodyMemberAttribute

[MessageContract]

public class uploadFile

{

[MessageHeader]

public string FileName

{ get;set;}

[MessageHeader]

public int length

{ get; set; }

[MessageBodyMember]

public byte[] getContent

{ get; set; }

}

复杂一些的消息契约：

[MessageContract]

public class order

{

[MessageHeader]

public string OrderID

{ get; set; }

[MessageHeader]

public DateTime Date

{ get; set; }

[MessageBodyMember]

public orderDetails Detail;

}

[CollectionDataContract]

public class orderDetails : List<orderDetails>

{ }

[DataContract]

public class orderDetail

{

[DataMember]

public string productID

{

get;

set;

}

[DataMember]

public int Quantity

{

get;

set;

}

}