1. **回顾数据契约**

* [DataContract]
* 数据契约则是定义服务端和客户端之间要传送的自定义数据类型。那么该类型就可以被序列化在服务端和客户端之间传送。类只有声明为[DataContract]，该类型的对象才可以被传送，且只有类的属性会被传送，需要在属性前加[DataMember]声明，这样该属性就可以被序列化传送。

**[DataContract] //数据契约声明**

**class MyDataContract**

**{**

**[DataMember]//数据成员标记**

**public string Name**

**{**

**get; set;**

**}**

**[DataMember]//数据成员标记**

**public string Email**

**{**

**get; set;**

**}**

**}**

1. **WCF服务所支持的序列化器**

* 目前，WCF3.5支持Json以后，提供了一种新序列化器:DataContractJsonSerializer .
* 现在一共是4种：
  + 1、XmlSerializer
  + 2、DataContratSerializer
  + 3、NetDataContractSerializer
  + 4、DataContractJsonSerializer

1. **DataContractSerializer 序列化器**

* 创建DataContractSerializer实例的方式
  + 指定根类型进行创建，根类型是序列化或反序列化实例的类型。
  + DataContractSerializer dcs = new DataContractSerializer(typeof(Person));

DataContractSerializer序列化方法

* + WriteObject() //序列化
  + ReadObject() //反序列化

实例：

1. 添加一个控制台程序，在Program里添加一个类（当然可以另写一个类，此处为了演示方便）

添加using System.Runtime.Serialization;

using System.IO;

[DataContract]

class person

{

private string name;

[DataMember]

public string Name

{

get { return name; }

set { name = value; }

}

private int age;

[DataMember]

public int Age

{

get { return age; }

set { age = value; }

}

private string address;

[DataMember]

public string Address

{

get { return address; }

set { address = value; }

}

}

运行中程序：

class Program

{

static void Main(string[] args)

{

DataContractSerializer dcs = new DataContractSerializer(typeof(person));

MemoryStream ms = new MemoryStream();

person p = new person();

p.Name = "张明";

p.Age = 23;

p.Address = "上海市正大路88号";

dcs.WriteObject(ms,p);//把person对象序列化到流中

ms.Position = 0;

var sr = new StreamReader(ms);

string str = sr.ReadToEnd();

Console.WriteLine(str);

ms.Close();

sr.Close();

Console.WriteLine("反序列化------------------");

var buffer = System.Text.Encoding.UTF8.GetBytes(str);

MemoryStream ms2 = new MemoryStream(buffer);

person pp = dcs.ReadObject(ms2) as person; //反序化列

Console.WriteLine("姓名={0},年龄={1},地址={2}", pp.Name, pp.Age, pp.Address);

}

}

1. **大数据量传输设置4-1**

1、修改客户端接收的最大数据值。

默认情况下，其接收的最大数据量为65535字节，如超过这个范围，就必须修改默认设置。

<binding … maxReceivedMessageSize=“999999“>

bind.MaxReceivedMessageSize = 9999999

2、WCF没有限制服务端发送数据的大小，但可以设置serviceBehaviors的dataContractSerializer 的maxItemsInObjectGraph属性，用来指定序列化和反序列化对象的最大数目，默认值为65535

注：当序列化数组时，每个数组元素当作一个对象

<behaviors>

<serviceBehaviors>

<behavior name="WcfServiceLibrary1.Service1Behavior">

<!-- 为避免泄漏元数据信息，

请在部署前将以下值设置为 false 并删除上面的元数据终结点 -->

<serviceMetadata httpGetEnabled="True"/>

<!-- 要接收故障异常详细信息以进行调试，

请将下值设置为 true。在部署前

设置为 false 以避免泄漏异常信息-->

<serviceDebug includeExceptionDetailInFaults="False" />

<dataContractSerializer maxItemsInObjectGraph="99999" ignoreExtensionDataObject="false"/>

</behavior>

</serviceBehaviors>

</behaviors>

1. **大数据量传输设置4-2**

3、本文与二进制

在大多数情况下，二进制消息量要少于XML本文消息。

byte[] getFile();

4、编码

* + TextMessageEncodingBingingElement
  + MtomMessageEncodingBingingElement
  + BinaryMessageEncodingBindingElement

**大数据量传输设置4-3**

* 流模式
  + 当有大量的数据要传输时，WCF中的流模式是整体缓冲和处理内存中消息的默认行为的一个可行的替代方法。
  + 如果数据无法分段、消息必须以及时的方式传递或者当传输启动时数据尚未完全就绪，则应考虑启用流模式，且只能对大型消息（带文本或二进制内容）启用流模式。

<bindings>

<basicHttpBinding>

<binding name="" transferMode="streaming" />

</basicHttpBinding>

</bindings>