**服务的三要素(ABC)**

**A:Address 意味着在哪里(也含有传输方式信息)**

**B:Binding 意味着怎么做(与地址的传输方式要匹配)**

**C:Contract意味着做什么(服务契约)**

配置文件：

**<system.ServiceModel>**

**<services>**

**<service>**

**<endpoint/> /\***服务和终结点**\*/**

**</service>**

**</services>**

**<bindings>**  /\*绑定（可选）\*/

**<binding>**

**</binding>**

**</bindings>**

**<behaviors> /\***行为（可选） **\*/**

**<behavior>**

**</behavior>**

**</behaviors>**

**</system.ServiceModel>**

* **endpoint元素**
* 每个终结点都需要以下属性表示的地址、绑定和协定：

– address。指定服务的统一资源标识符(URI)，它可以是一个绝对

地址，或是一个相对于服务基址给定的地址。如果设置为空字符

串，则指示在创建服务的ServiceHost 时，终结点在指定的基址

上可用。

– binding。通常，指定一个类似WsHttpBinding 的系统提供的绑定，

但也可以指定一个用户定义的绑定。指定的绑定确定传输协议类

型、安全和使用的编码，以及是否支持或启用可靠会话、事务或

流。

– bindingConfiguration。如果必须修改绑定的默认值，则可通过在

bindings 元素中配置相应的binding 元素来执行此操作。此属性

应赋予与用于更改默认值的binding 元素的name 属性相同的值。

– contract。指定定义协定的接口。这是在由service 元素的name

属性指定的公共语言运行库(CLR) 类型中实现的接口。

**绑定(Binding)**

**绑定：**

**就是预先配置好的信道栈，它代表了客户端和服务器之间线路级别上的约定。每个绑定都会指定通信所用的传输、编码及协议。通过绑定，WCF封装了各种不同通信场所下所使用的配置。**





**性能比较：**



**Get256Bytes50000次 平均响应时间(单位：ms) 值小者为佳**

**端点地址的定义**

* 终结点的地址由EndpointAddress 类表示，该类包含一个表示服务地址的统一资源定位符(URI)，大多数传输的地址URI 包含四个部分。
* 例如，

“http://www.sina.com.cn:3200/mathservice”这个URI 具有以下四个部分：

– 方案：http:

– 计算机：www.sina.com.cn

– （可选）端口：3200

– 路径：/mathservice

**端点地址定义**

* •在配置文件中定义终结点地址

**<system.serviceModel>**

**<services>**

**<service**

**name="CalculatorService"**

**behaviorConfiguration="CalculatorServiceBehavior">**

**<host>**

**<baseAddresses>**

**<add baseAddress="http://localhost:8000/service"/>**

**</baseAddresses>**

**</host>**

**<endpoint address="" binding="wsHttpBinding" contract="Samples.ICalculator" />**

**<endpoint address="/test" binding="wsHttpBinding"**

**contract="Samples.ICalculator" />**

**<endpoint address="net.tcp://localhost:9000/service"**

**binding="netTcpBinding"**

**contract="Samples.ICalculator" />**

**</service>**

**</services>**

下面举例讲解一下WCF服务的调用及简单配置:

首先创建WCF服务应用程序WcfServiceLibrary2

Iservice接口：

using System.Data;

using System.Data.SqlClient;

namespace WcfServiceLibrary2

{

[ServiceContract]

public interface IService1

{

[OperationContract]

DataTable getLinks();

[OperationContract]

void addEmps( link e);

}

[DataContract]

public class link

{

private int id;

[DataMember]

public int Id

{

get { return id; }

set { id = value; }

}

private string site\_path;

[DataMember]

public string Site\_path

{

get { return site\_path; }

set { site\_path = value; }

}

private string user\_name;

[DataMember]

public string User\_name

{

get { return user\_name; }

set { user\_name = value; }

}

private string user\_tel;

[DataMember]

public string User\_tel

{

get { return user\_tel; }

set { user\_tel = value; }

}

}

}

对应的实现：

using System.Data;

using System.Data.SqlClient;

namespace WcfServiceLibrary2

{

// 注意: 如果更改此处的类名“IService1”，也必须更新 App.config 中对“IService1”的引用。

public class Service1 : IService1

{

#region IService1 成员

public System.Data.DataTable getLinks()

{

using (SqlConnection conn = new SqlConnection("server=.;uid=sa;pwd=123;database=GjjWebdb"))

{

string sql = " select \* from dt\_link";

SqlDataAdapter sda = new SqlDataAdapter(sql,conn);

DataTable dt = new DataTable("员工表");

sda.Fill(dt);

return dt;

}

}

public void addEmps(link e)

{

using (SqlConnection conn = new SqlConnection("server=.;uid=sa;pwd=123;database=GjjWebdb"))

{

string sql = "insert into dt\_link value(@id,@User\_name,@User\_tel,@Site\_path)";

SqlCommand cmd = new SqlCommand(sql,conn);

cmd.Parameters.AddWithValue("@id",e.Id);

cmd.Parameters.AddWithValue("@User\_name", e.User\_name);

cmd.Parameters.AddWithValue("@User\_tel", e.User\_tel);

cmd.Parameters.AddWithValue("@Site\_path", e.Site\_path);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

#endregion

}

}

然后再创建web窗体应用程序WCFServer

1. 先添加引用System.serviceModel,和刚建立的WCF服务WcfServiceLibrary2
2. Form1窗体中添加两个按钮，分别表示服务的开启以及停止，并创建ServiceHost对象

ServiceHost host = null;

//开启

private void button1\_Click(object sender, EventArgs e)

{

host = new ServiceHost(typeof(WcfServiceLibrary2.Service1));

host.Open();

label1.Text = "服务已开启";

}

//停止

private void button2\_Click(object sender, EventArgs e)

{

if (host.State != CommunicationState.Closed)

{

host.Close();

label1.Text = "服务已停止";

}

}

1. 添加应用程序配置文件App.config

<configuration>

<system.serviceModel>

<services>

<service name="WcfServiceLibrary2.Service1" behaviorConfiguration="testBehavior">

<host>

<baseAddresses>

<add baseAddress="http://localhost:3333"/>

</baseAddresses>

</host>

<endpoint address="" binding="wsHttpBinding" contract="WcfServiceLibrary2.IService1"></endpoint>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="testBehavior">

<serviceMetadata httpGetEnabled="true"/>

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

</configuration>

到此，服务即可启动以及停止，可以用浏览器打开<http://localhost:3333>查看；