## WebServices –web服务

## WCF

## Remoting

## WSE

## Socket

UDDI 提供了一组基于标准的规范用于描述和发现服务

统一描述、发现和集成（Universal Description, Discovery, and Integration，UDDI）

[Web Service](http://www.cnblogs.com/meil/archive/2006/08/22/483788.html)

由两部分组成:

(1)SOAP--Web Service之间的基本通信协议。 本质上 是通过xml来传输的

(2)WSDL--Web Service描述语言，它定义了Web Service做什么，怎么做和查询的信息。

## 什么是SOA

SOA全称是**[面向服务的体系结构](http://zh.wikipedia.org/wiki/SOA)**（Service-oriented architecture）是构造[分布式计算](http://zh.wikipedia.org/wiki/åå¸å¼è¨ç®" \o "分布式计算)的应用程序的方法。它将应用程序功能作为[服务](http://zh.wikipedia.org/wiki/æå¡)发送给[最终用户](http://zh.wikipedia.org/wiki/æç)或者其他[服务](http://zh.wikipedia.org/wiki/æå¡)。

它是一种组件架构模型。

## SOA的原则

* **服务**[**封装**](http://zh.wikipedia.org/w/index.php?title=%E5%B0%81%E8%A3%9D&action=edit&redlink=1)
* **服务[松耦合(Loosely coupled)](http://zh.wikipedia.org/w/index.php?title=%E9%AC%86%E8%80%A6%E5%90%88(Loosely_coupled)&action=edit&redlink=1" \o "松耦合(Loosely coupled)（页面不存在）)** - 服务之间的关系最小化，只是互相知道。
* **服务契约** - 服务按照服务描述文档所定义的服务契约行事。
* **服务抽象** - 除了服务契约中所描述的内容，服务将对外部隐藏逻辑。
* **服务的重用性** - 将逻辑分布在不同的服务中，以提高服务的重用性。
* **服务的可组合性** - 一组服务可以协调工作并组合起来形成一个组合服务。
* **服务自治** – 服务对所封装的逻辑具有控制权
* **服务无状态** – 服务将一个活动所需保存的资讯最小化。
* **服务的可被发现性** – 服务需要对外部提供描述资讯，这样可以通过现有的发现机制发现并访问这些服务。

## 什么是WCF

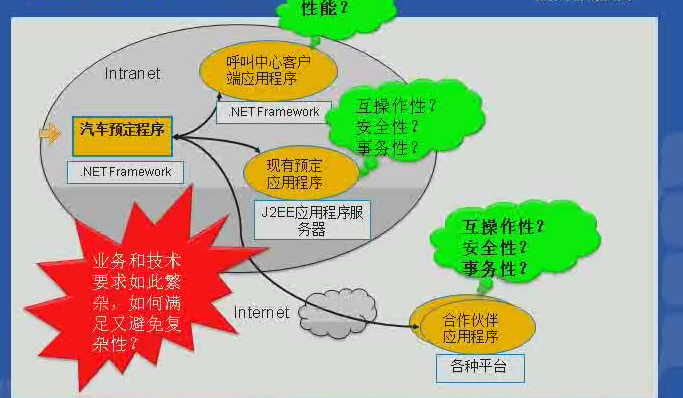
WCF的全称是：Windows Communication Foundation。从本质上来说，它是一套软件开发包，是微软公司推出的符合SOA思想的技术框架。

WCF为程序员提供了丰富的功能，其中包括：托管、服务实例管理、异步、安全、事务管理、离线队列等。并且WCF对产业中的标准协议进行了封装和定义，它把程序员从繁琐的通信、格式编码中解放出来，使得程序员能够专注于业务逻辑的实现。同时，WCF统一了微软公司之前推出的多种分布式技术，其中包括：

1. Web服务
2. .Net Remoting。
3. 微软消息队列（MSMQ）。
4. 其他一些服务

WCF对这些技术的集成包括两个方面：

1. WCF的架构本身吸取了这些技术的精华。
2. WCF开发的服务/客户端可以和现有的Web服务、MSMQ程序进行交互。



1. 契约

从SOA的概念上来看，契约属于一个服务公开接口的一部分。一个服务的契约，定义了服务端公开的服务方法、使用的传输协议、可访问的地址、传输的消息格式等内容。

基本上，契约的定义描述了该服务的功能和作用，它告诉SOA系统中的其它节点这个服务是“干什么”的。

1. 服务运行时

服务运行时定义了服务在运行时的具体行为。如果说契约描述了服务是“干什么”的，那么服务运行时就在一定程度上描述了服务是“怎么干”的。

1. 消息

消息方面包含了消息的传输方式、消息的编码与解码。消息方面的内容基本属于服务边界以内的具体实现。具体的传递时限，必须符合在契约中定义的绑定协议。

1. 激活和宿主

激活和宿主属于WCF程序的部署方式。一个WCF服务需要在一个可运行的程序中寄宿，我们可以把宿主理解为WCF运行的容器。常用的寄宿方式包括自寄宿、IIS寄宿、Windows激活服务、Windows服务、Com+组件等。根据SOA的原则，激活和宿主类型的变化不会影响服务本身的特性和外部对该服务的访问，而WCF在这一方面也确实做的非常出色。

Demo1-WCF的构成以及基本的通讯

WCF契约定义

[ServiceContract]

public interface IService1

{

[OperationContract]

string GetData(int value,ref string strRef,out string strOut);

[OperationContract]

CompositeType GetDataUsingDataContract(CompositeType composite);

// TODO: Add your service operations here

[OperationContract(IsOneWay=true)]

void TestMethod(string strInput);

}

实现契约

public class Service1 : IService1

{

public string GetData (int value,ref string strRef,out string strOut)

{

//System.Threading.Thread.Sleep(10000);

strRef += "---changed by wcf method";

strOut = "from wcf method";

return string.Format("You entered: {0}", value);

}

public CompositeType GetDataUsingDataContract(CompositeType composite)

{

if (composite.BoolValue)

{

composite.StringValue += "Suffix";

}

return composite;

}

public void TestMethod(string strInput)

{

System.Threading.Thread.Sleep(10000);

}

**WCF的基本组成**

1. 地址（Address）

在WCF框架中，每个服务都具有唯一的地址。在SOA系统中，其它服务和客户端通过服务的地址来对服务进行访问。一个服务的地址由一个统一资源标示符（URI）来表示。下面是几个地址示例：

http://localhost:80/Service

net.tcp://dc3web1:9023/MyService

net.msmq://localhost/MyMsMqService

实际上地址的形式不止这些，它们的构成形式如下所示：

http://[Hostname]:[Port]/[ServiceAddress]

https://[Hostname]:[Port]/[ServiceAddress]

net.tcp://[Hostname]:[Port]/[ServiceAddress]

net.pipe://[Hostname]:[Port]/[ServiceAddress]

net.msmq://[Hostname]/public(private)/[QueueName]

msmq.formatname://{msmq format name}

1. 绑定（Binging）

绑定定义了服务与外部通信的方式。它由一组称为绑定元素的元素而构成，这些元素组合在一起形成通信基础结构。一个绑定可以包含以下内容。

1. 通信所使用的协议，如HTTP、TCP、P2P等。
2. 消息编码方式，如纯文本、二进制编码、MTOM等。
3. 消息安全保障策略。
4. 通信堆栈的其它任何要素。
5. 契约（Contract）

在WCF中一共包含了4种契约，分别是服务契约、数据契约、错误契约和消息契约。

1. 服务契约[ServiceContract]

服务契约将多个相关的操作联系在一起，组成单个功能单元。

1. 数据契约[DataContract]

数据类型的说明称为数据契约。服务使用的数据类型必须在元数据中进行描述，以使其它各方面可以与该服务进行交互操作。

1. 错误契约[FaultContract]

错误类型的说明称为错误契约。

1. 消息契约[MessageContract]

消息契约描述消息的格式。

1. 终节点(EndPoint)

终结点是用来发送或接收消息（或同时执行这两种操作）的构造。一个终节点由三个要素组成，分别是:地址、绑定和契约。以SOA的思想来看，一个终节点就相当于服务的公共接口。

1. 元数据

服务的元数据描述服务的特征，外部实体需要了解这些特征以便与该服务进行通信。服务所公开的元数据包括XML架构文档（用于定义服务的数据协定）和WSDL文档（用于描述服务的方法）。启用元数据后，WCF通过检查服务及其终节点自动生成服务的元数据。

1. 宿主

服务必须承载于某个进程中。宿主是控制服务的生存期的应用程序。

## 创建服务契约的两种方式

WCF 服务端编程

WCF全解析

1. 从类上定义契约

[ServiceContract]

public class Service1

{

[OperationContract]

public string GetData(int value,ref string strRef,out string strOut)

{

//System.Threading.Thread.Sleep(10000);

strRef += "---changed by wcf method";

strOut = "from wcf method";

return string.Format("You entered: {0}", value);

}

}

2 利用接口定义契约

[ServiceContract]

public interface IService1

{

[OperationContract]

string GetData(int value,ref string strRef,out string strOut);

[OperationContract]

CompositeType GetDataUsingDataContract(CompositeType composite);

// TODO: Add your service operations here

[OperationContract(IsOneWay=true)]

void TestMethod(string strInput);

}

## 创建服务契约的消息模式

**请求响应模式**

VS2010建立WCF时默认生成的消息模式

**缺点：**客户端需要等待服务端执行完毕，并相应消息后才能继续往下执行

**优点：**响应消息中可以返回Soap错误，这表明可能在通信或处理过程中发生了一些与服务有关的错误，客户端可以得到此错误信息反馈给用户

**请求报文模式**isoneway=true

[OperationContract(IsOneWay=true)]

void TestMethod(string strInput);

**双工通讯模式**

public interface ICallBack

{

[OperationContract(IsOneWay = true)]

void UpdateInterval(int seconds);

}

[ServiceContract(CallbackContract=typeof(ICallBack))]

public interface IService1

{

[OperationContract]

void Heartbeat();

}

**典型的双工通讯要素：**

1. **服务端需要定义客户端请求的契约**
2. **服务端需要定义双工通讯中回调的接口**
3. **客户端实现WCF服务端定义的回调接口中的函数**

**4、在服务端契约上面要开启sessionMode=** Required并且要利用CallbackContract

指定当前服务契约回调的接口是什么

5、在服务契约实现类中根据当前操作上下文实现回调接口的具体实例，以及在方法里面调用

## 错误契约

Fault的子节点

|  |  |
| --- | --- |
| **子节点** | **描述** |
| <Code> | 供识别故障的代码，可包含两个子节点：Code和Subcode |
| <Reason> | 可供人阅读的有关故障的说明 |
| <Node> | 可选，指向SOAP消息中引发错误的节点 |
| <Role> | 可选，发生错误时节点所处的角色 |
| <Detail> | 可选，包含了详细的错误信息 |

说明：无论是使用哪种错误方式：WCF系统最终会把错误信息放入Fault节点中进行传输

## 消息契约

[MessageContract]

public class MyMessage

{

[MessageHeader]

public string Operation

{

get { return operation; }

set { operation = value; }

}

[MessageBodyMember]

public double Result

{

get { return result; }

set { result = value; }

}

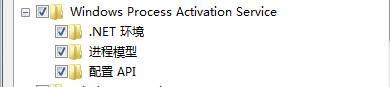
}

# WCF的net.tcp绑定在IIS中承载

### 安装WAS

IIS原本是不支持非HTTP协议的服务，为了让IIS支持net.tcp，必须先安装WAS(Windows Process Activation Service)，即windows进程激活服务。

打开控制面板--程序和功能--打开或关闭windows功能，安装WAS，如图：



安装完毕后在Services窗口中可以到到如下服务:Windows Process Activation Service;Net.Msmq Listener Adapter;Net.Pipe Listener Adapter;Net.Tcp Listener Adapter;Net.Tcp Port Sharing Service.这几个服务。确定Net.Tcp Listener Adapter 与Net.Tcp Port Sharing Service是否已经启动。

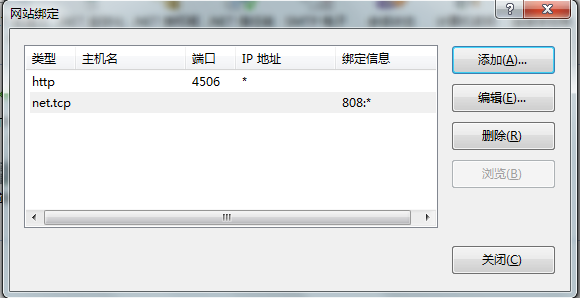
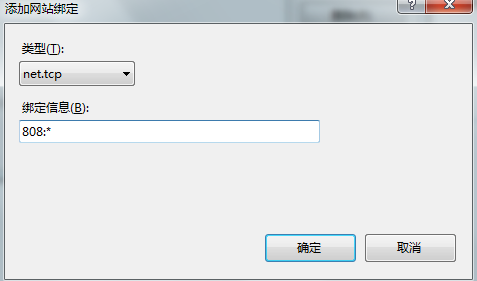
### 确定WCF是否启用Non-Http支持

同样是在控件面板中打开这个功能，如图：



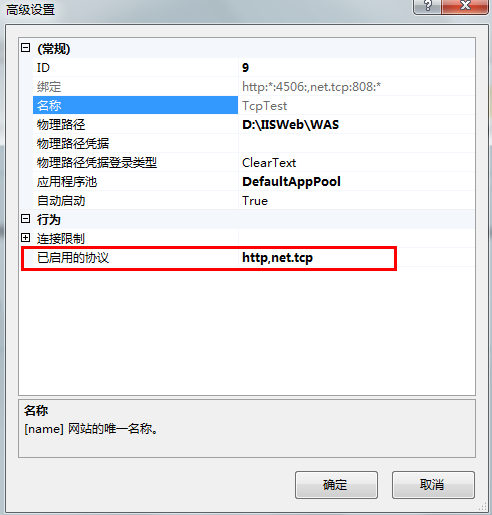
### 给站点添加net.tcp绑定

在IIS中，选中你的网站，然后在右边的操作菜单栏中单击绑定，会弹出一个“网站绑定”窗口，点击添加，类型选择net.tcp



### 启用net.tcp协议

 选择你的网站，点击“高级设置”，弹出的的窗体中，在“已启用的协议”一栏中手动添加：net.tcp



# 2 测试服务

### 2.1 新建服务

用VS2010新建一个WCF服务，为了简单，我就直接用VS默认生成的作测试了。只有一个GetData方法

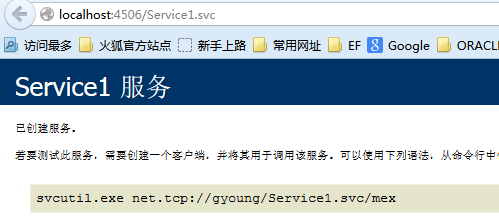
下面是配置的Config:

View Code



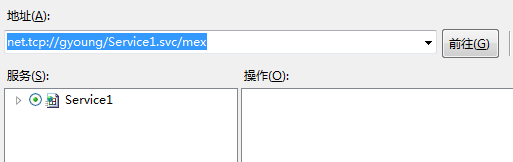
### 2.2 发布服务

将服务发布到IIS，在浏览器中访问服务，如果访问正常就说明服务部署成功，如图：



### 2.3 测试服务

 新建一个控制台项目，测试服务。添加服务



测试服务正常。

**如果由请求响应改成数据包模式，那么方法必须定义为void 并且不能**

**配置文件三个模块**

**服务终结点作用是，**

<services>

<service> </service>

</services>

**暴露wcf服务的接口（address），**

<endpoint address=""

**元数据**

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

**，绑定方式，**

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

**以及行为配置**

<endpoint address="" binding="wsHttpBinding" **behaviorConfiguration=""**

**指定当前endpoint address指定的服务用的是哪个契约**

<endpoint address="" binding="wsHttpBinding" behaviorConfiguration="" bindingConfiguration="transactionalBinding"

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

**绑定模块**

**定义当前的Services模块中绑定类型（**binding属性**）的绑定**

bindings

**下的节点必须与**endpoint中的binding类型一致

例如

<**wsHttpBinding**>

<binding name="**transactionalBinding**" transactionFlow="true"/>

</**wsHttpBinding**>

**行为模块注意点**

<endpoint address="" binding="wsHttpBinding" behaviorConfiguration="CalculatorServiceBehavior" bindingConfiguration="transactionalBinding"

contract="Microsoft.ServiceModel.Samples.ICalculator"/>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

<endpoint address="" binding="wsHttpBinding" behaviorConfiguration=" CalculatorServiceBehavior1" bindingConfiguration="transactionalBinding"

contract="Microsoft.ServiceModel.Samples.ICalculator"/>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

<behaviors>

<serviceBehaviors>

<behavior name="CalculatorServiceBehavior">

<serviceMetadata httpGetEnabled="true"/>

<serviceDebug includeExceptionDetailInFaults="false"/>

</behavior>

<behavior name="CalculatorServiceBehavior1">

<serviceMetadata httpGetEnabled="true"/>

<serviceDebug includeExceptionDetailInFaults="false"/>

</behavior>

</serviceBehaviors>

</behaviors>

**契约**

**四种契约**

**服务契约的通讯模型**

**请求-响应 –同步处理**

**数据报模式 –服务方法需要长时间处理才能完成的并且不需要实时返回给客户端的处理**

**双工—服务端在处理完毕或过程中需要调用客户端的方法**

**IIS托管WCF服务**

**在客户端用SvcUtil.exe工具生成代理类文件**

**相关DEMO**