WCF中间件使用说明

一.配置文件说明：

<configSections>

<section name="WpwWcfSeting" type="Eltc.Base.FrameWork.Helper.Wcf.InitConfig,Eltc.Base.FrameWork" requirePermission="false"/>

</configSections>

<WpwWcfSeting>

<WcfServer Path=" ~/servercontract.config"/>

<WcfConstant Path=" ~/serverconst.config"/>

<WcfClent Path=" ~/clientcontract.config"/>

<WcfClentConstantPath Path=" ~/clientconst.config"/>

</WpwWcfSeting>

说明：

一.app.config或者web.config配置

文件路径可以使相对路径，也可以是绝对路径

1. WcfServer 服务器端契约配置文件，可以设置多个

2. WcfConstant服务器端常量配置文件，只允许一个

3. WcfClent客户端契约配置文件，可以设置多个

3. WcfClentConstantPath客户端常量配置文件，可以设置多个

二.各个 契约、常量 文件说明

1.服务端常量配置文件serverconst.config

<?xml version="1.0"?>

<configuration>

<NetTcpBinding>

<readerQuotas maxDepth="64"

maxStringContentLength="2147483647"

maxArrayLength="16384"

maxBytesPerRead="4096"

maxNameTableCharCount="16384" />

<!--定义可靠会话状态参数-->

<reliableSession enabled="false"

ordered="true"

inactivityTimeout="00:10:00"/>

<!--定义发布的ip和端口，以及协议-->

<host baseAddress="localhost:30005"

binding="netTcpBinding" />

<!--定义服务器端配置参数-->

<behaviors closeTimeout="00:10:00"

openTimeout="00:10:00"

receiveTimeout="00:10:00"

sendTimeout="00:10:00"

transactionFlow="false"

transferMode="Buffered"

transactionProtocol="OleTransactions"

hostNameComparisonMode="StrongWildcard"

listenBacklog="200"

maxBufferPoolSize="2147483647"

maxBufferSize="2147483647"

maxConnections="1000"

maxReceivedMessageSize="2147483647"

portSharingEnabled="false"

securitymode="None"

clientCredentialType="None"

enableBinaryFormatterBehavior="true">

</behaviors>

<!-- 要接收故障异常详细信息以进行调试，

请将以下值设置为 true。在部署前设置为 false

以避免泄漏异常信息-->

<serviceDebug includeExceptionDetailInFaults="true"/>

<!--定义会话行为-->

<serviceThrottling maxConcurrentCalls="1000"

maxConcurrentInstances="1000"

maxConcurrentSessions="1000"/>

</NetTcpBinding>

</configuration>

注：1.reliableSession节点enabled设置为false 将加大服务器并发属性

2. enableBinaryFormatterBehavior 是否开启自定义二进制序列化器（暂时不支持，等待后续开发）

2.服务端契约配置文件servercontract.config

<?xml version="1.0"?>

<configuration>

<Services>

<!--maxItemsInObjectGraph定义序列化数据大小(大数据量使用)-->

<service name="WPW.Demo.Wcf.Imp.UserBll,WPW.Demo.Wcf.Imp"

contract="WPW.Demo.Wcf.Interface.IUserBll,WPW.Demo.Wcf.Interface"

address="WPW/Demo/Wcf/Interface/IUserBll"

maxItemsInObjectGraph="2147483647">

</service>

</Services>

</configuration>

注：1.契约和实现都必须是全限定名

2. maxItemsInObjectGraph属性配置大数据传输时使用，否则报异常

3.客户端常量配置文件clientcontract.config

<?xml version="1.0"?>

<configuration>

<Servers>

<Server name="Demo">

<Binding Binding="netTcpBinding"

MaxBufferPoolSize="524288"

MaxBufferSize="2147483647"

MaxReceivedMessageSize="2147483647"

MaxConnections="1000"

ListenBacklog="200"

SendTimeout="00:10:00"

OpenTimeout="00:10:00"

ReceiveTimeout="00:10:00"

TransferMode="Buffered"

SecurityMode="None"

Address="localhost:30005"

ReaderQuotasMaxDepth="64"

ReaderQuotasMaxStringContentLength="2147483647"

ReaderQuotasMaxArrayLength="16384"

ReaderQuotasMaxBytesPerRead="4096"

ReaderQuotasMaxNameTableCharCount="16384"

ReliableSessionOrdered="true"

ReliableSessionEnabled="false"

ReliableSessionInactivityTimeout="00:10:00"

EnableBinaryFormatterBehavior="true"

IsUseWcfPool="true"

WcfMaxPoolSize="1000"

WcfOutTime="5"

WcfFailureTime="60"

WcfPoolMonitorReapTime="5"

WcfAppName="abc">

</Binding>

<!--负载均衡信息-->

<LoadBalance IsUsed="true"

BalanceAlgorithm="ConsistentHashing"

IsUseWcfPool="true"

WcfOutTime="10"

WcfFailureTime="60"

WcfPoolMonitorReapTime="10">

<Address WcfMaxPoolSize="1000" HeatBeatTime="00:00:00.500">localhost:30005</Address>

<Address WcfMaxPoolSize="1000" HeatBeatTime="00:00:00.500">localhost:30006</Address>

</LoadBalance>

</Server>

<Server name="Demo1">

<Binding Binding="netTcpBinding"

MaxBufferPoolSize="524288"

MaxBufferSize="2147483647"

MaxReceivedMessageSize="2147483647"

MaxConnections="1000"

ListenBacklog="200"

SendTimeout="00:10:00"

OpenTimeout="00:10:00"

ReceiveTimeout="00:10:00"

TransferMode="Buffered"

SecurityMode="None"

Address="localhost:30005"

ReaderQuotasMaxDepth="64"

ReaderQuotasMaxStringContentLength="2147483647"

ReaderQuotasMaxArrayLength="16384"

ReaderQuotasMaxBytesPerRead="4096"

ReaderQuotasMaxNameTableCharCount="16384"

ReliableSessionOrdered="true"

ReliableSessionEnabled="false"

ReliableSessionInactivityTimeout="00:10:00"

EnableBinaryFormatterBehavior="true"

IsUseWcfPool="true"

WcfMaxPoolSize="1000"

WcfOutTime="5"

WcfFailureTime="60"

WcfPoolMonitorReapTime="5"

WcfAppName="abc">

<!--负载均衡信息-->

<LoadBalance IsUsed="true"

BalanceAlgorithm="ConsistentHashing"

IsUseWcfPool="true"

WcfOutTime="10"

WcfFailureTime="60"

WcfPoolMonitorReapTime="10">

<Address WcfMaxPoolSize="1000" HeatBeatTime="00:00:00.500">localhost:30005</Address>

<Address WcfMaxPoolSize="1000" HeatBeatTime="00:00:00.500">localhost:30006</Address>

</LoadBalance>

</Binding>

</Server>

</Servers>

</configuration>

注：

1. MaxBufferSize MaxReceivedMessageSize 两个属性同时影响接收数据大小限制，至于小粒度的接收数据的控制通过契约中的maxItemsInObjectGraph属性来控制

2.IsUseWcfPool="true" WcfMaxPoolSize="1000" WcfOutTime="5" WcfFailureTime="60"

其中这几项是配置wcf连接池使用

（1）. IsUseWcfPool true标识使用连接池

（2）. WcfMaxPoolSize 连接池大小，大并发量是可以增大此值大小,但是防止服务器被压死，此数据不易调整太大，建议1000-1500

（3）. WcfOutTime 获取连接超时时间(秒为单位)，如果规定时间内未获取连接，则抛出连接池满，获取连接超时异常

（4）. WcfFailureTime 连接回收时间(秒为单位)，超过设定大小，则销毁连接

（5）. WcfPoolMonitorReapTime 连接池监控线程轮询时间

（5）. Server name="Demo1" 此节点为所配置的服务器别名，供契约找到指定服务器使用

（6）. ReliableSessionEnabled="false" 与服务器端设置一直，否则报不匹配异常

（7）. WcfPoolMonitorReapTime Wcf连接池轮询时间（以秒为单位），大并发量时建议轮询时间设为5秒

（8）. EnableBinaryFormatterBehavior 是否启用自定义二进制序列化器（暂不支持，等后续开发）

（9）. WcfAppName wcf应用程序名称(上下文传递使用)，可选

（10）. LoadBalance节点配置负载均衡信息，参数说明：

a. IsUsed 是否启用负载均衡，注：如果启用负载均衡，则Binding节点的Address，IsUseWcfPool，WcfMaxPoolSize，WcfOutTime，WcfFailureTime，WcfPoolMonitorReapTime属性无效。

b. BalanceAlgorithm 负载均衡算法，目前只支持一致性哈希（权重轮询等后续版本处理）

c. IsUseWcfPool 是否启用连接池

d. WcfOutTime 获取连接超时时间(秒为单位)，如果规定时间内未获取连接，则抛出连接池满，获取连接超时异常。

e. WcfFailureTime 连接回收时间(秒为单位)，超过设定大小，则销毁连接

f. WcfPoolMonitorReapTime 连接池监控线程轮询时间

g. HeatBeatTime 负载服务器状态监控的心跳间隔时间

4.客户端契约配置文件clientcontract.config

<?xml version="1.0"?>

<configuration>

<Clients>

<Server name="Demo">

<!--maxItemsInObjectGraph属性在需要传输大数据量数据时使用-->

<client contract="WPW.Demo.Wcf.Interface.IUserBll"

address="WPW/Demo/Wcf/Interface/IUserBll"

maxItemsInObjectGraph="2147483647">

</client>

</Server>

</Clients>

</configuration>

注：

（1）. Server name="Demo" 服务器别名对应客户端常量配置文件中指定的别名

（2）. contract 必须是契约接口的全名，而非全限定名

二.各项目间的调用关系

采用非双工，问答模式，服务消费者主动请求，服务提供者响应，服务消费者主动断开连接

服务器端

Model

契约的实现

契约(接口)

客户端

FrameWork类库

三.用例：

启动服务器端服务，可选择的宿主：1.控制台，2.winform 3.windows服务

using Eltc.Base.FrameWork.Helper;

WcfServer server = WcfServer.GetInstance();

server.WcfFaultedEvent += new WcfFaulted(server\_WcfFaultedEvent);

server.WcfClosedEvent += new WcfClosed(server\_WcfClosedEvent);

server.WcfBeforeCallEvent += new WcfBeforeCall(server\_WcfBeforeCallEvent);

server.WcfAfterCallEvent += new WcfAfterCall(server\_WcfAfterCallEvent);

server.Start();

Console.WriteLine("服务启动...");

Console.ReadKey();

server.Stop();

客户端调用：

using Eltc.Base.FrameWork.Helper;

using WPW.Demo.Wcf.Interface;

using WPW.Demo.Wcf.Model;

IUserBll bll = WcfClient.GetProxy<IUserBll>();

IList<User> list = bll.FindAll();

四.关于客户端传递消息上下文到服务器端跟踪具体的业务行为

客户端设定传递的上下文信息

//绑定列表

IList<TSource> list = null;

//创建传递的上下文信息

HeaderContext context = new HeaderContext();

context.AppName = "";

context.CorrelationState = Guid.NewGuid().ToString();

context.Ip = "";

HeaderOperater.SetClientWcfHeader(context);

ITSourceBll bll = WcfClient.GetProxy<ITSourceBll>();

//Stopwatch s = new Stopwatch();

//s.Start();

list = bll.FindByPage(0, 9999);

//s.Stop();

//Response.Write(s.ElapsedMilliseconds);

this.Repeater1.DataSource = list;

this.Repeater1.DataBind();

//清除传递的上下文信息，

//注：上下文信息必须在创建后清除

HeaderOperater.ClearClientWcfHeader();

服务器端获取客户端传递的上下文信息

//获取传递的自定义消息头

HeaderContext headercontext = HeaderOperater.GetServiceWcfHeader(OperationContext.Current);

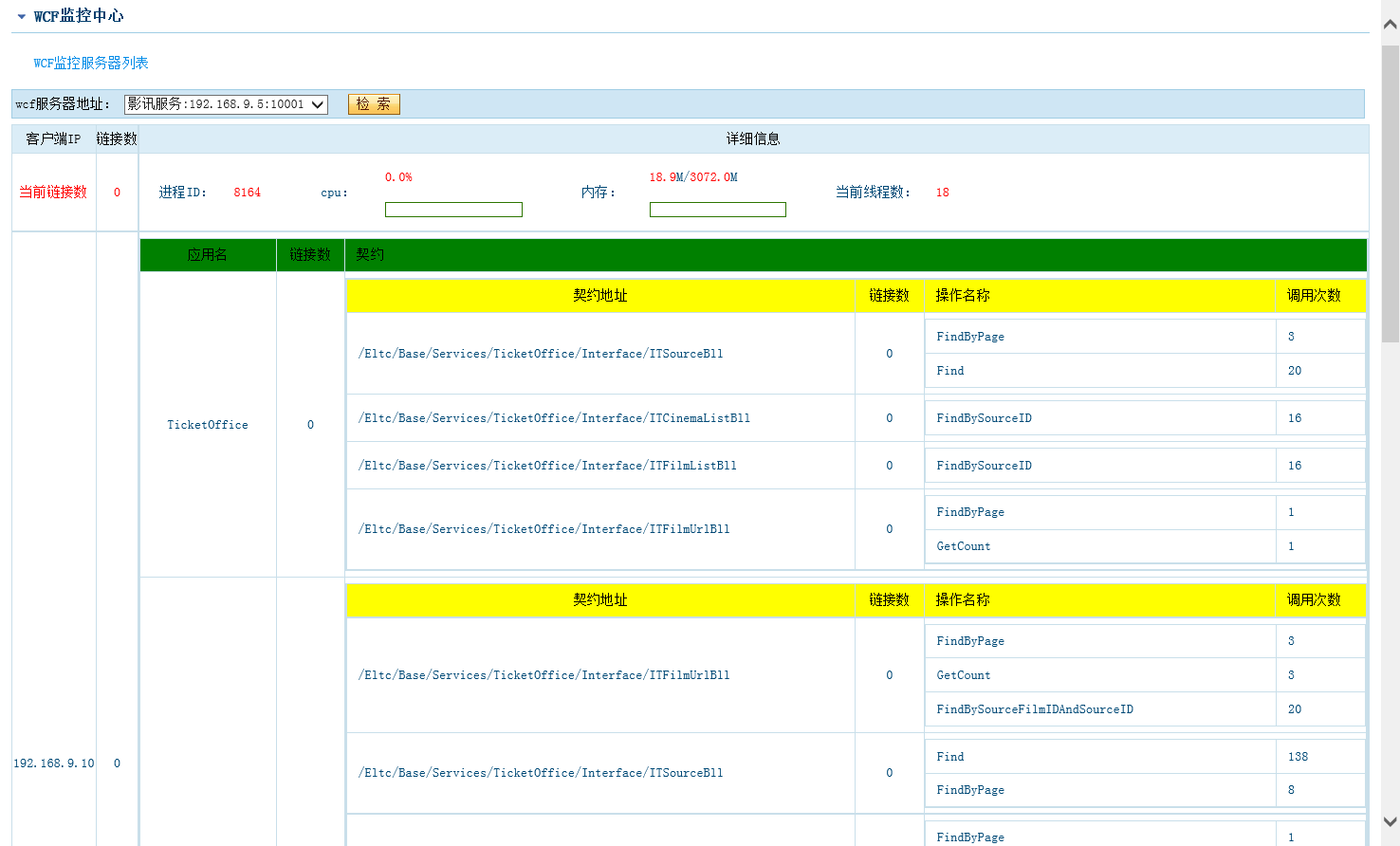
注：如果客户端没有传递上下文信息，服务器端将自动创建一个标识Guid，此标识只能在

WcfServer.WcfBeforeCallEvent和WcfServer.WcfAfterCallEvent两个事件中获取，但是不会创建上下文信息，如果要在服务器端创建上下文信息，可以自己定义。

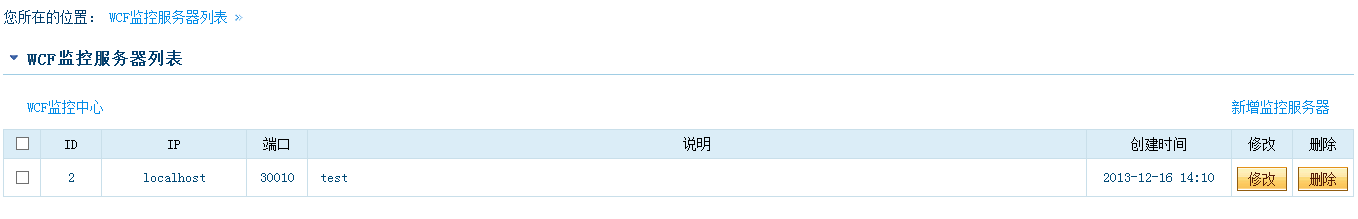
五．关于wcf监控中心的使用：

监控中心初步监控界面，后续需要持续开发

1.服务状态监控



2.需监控服务的列表



后续需要开发的功能：

1. 监控大盘：

(1).通过服务名称🡪负载的服务提供者🡪服务消费者(应用名称)

三级显示监控数据，图例：

服务名称

服务提供者

192.168.9.11

显示服务器性能数据

服务提供者

192.168.9.10

显示服务器性能数据

服务消费者(应用名称)

192.168.9.101

显示客户机连接数

服务消费者(应用名称)

192.168.9.102

显示客户机连接数

服务消费者(应用名称)

192.168.9.100

显示客户机连接数

1. 监控数据报表：

下面按照天，周，月，季度统计

(1).服务提供者cpu,内存,线程数,总连接数 折线图

(2).服务消费者总连接数,各个契约连接数 折线图

每分钟各个方法调用次数 柱状图