1. **SQL中的事务处理**

* 无论是基于SQL Server的T-SQL，抑或是基于Oracle的PL-SQL都对事务提供了原生的支持。以T-SQL为例

我们可以通过如下三个SQL语句实现事务的启动、提交与回滚：

* + BEGIN TRANSACTION: 开始一个事务；
  + COMMIT TRANSACTION：提交事务
  + ROLLBACK TRANSACTION：回滚事务

T-SQL事务处理方式

create procedure TransferMoeny

@m money

as begin

begin transaction

begin try

update account set balance=balance-@m where ID='A'

update account set balance=balance+@m where ID='B'

commit transaction

end try

begin catch

rollback transaction

end catch

end

exec TransferMoeny 200

1. **ADO.NET中事务处理**

对于.NET开发人员，我们可以直接利用ADO.NET将基于单个数据库连接的多个操作纳入同一个事务之中。

using (DbTransaction transaction = connection.BeginTransaction())

{

}

例子：

class Class1

{

SqlConnection conn;//连接对象

SqlTransaction tran;//事务对象

public Class1()

{

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

}

//转出

private void OutMoney(int m)

{

SqlCommand cmd = new SqlCommand("update account set balance=balance-"+m+" where ID='A'", conn);

cmd.Transaction = tran;

cmd.ExecuteNonQuery();

}

//转入

private void IntoMoney(int m)

{

SqlCommand cmd = new SqlCommand("update account set balance=balance+" + m + " where ID='A'", conn);

cmd.Transaction = tran;

cmd.ExecuteNonQuery();

}

//公开转账

public void TransferMoney(int m)

{

conn.Open();

tran = conn.BeginTransaction();//开启事务

try

{

OutMoney(m);

IntoMoney(m);

tran.Commit(); //提交事务

}

catch (Exception err)

{

tran.Rollback(); //回滚事务

}

finally

{

conn.Close();

}

}

}

1. **实例上下文模式——单调PerCall**

由于WCF的并发是针对某个封装了服务实例的InstanceContext而言的，但是对单调的实例上下文模式，WCF服务端运行时总是创建一个全新的InstanceContext来处理每一个请求，不管该请求是否来自相同的客户端。

[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerCall)]

public class ServiceClass : IService1

{

}

1. **会话（PerSession）**

* 在基于会话的实例上下文提供机制下，被创建出来封装服务实例的InstanceContext与会话（客户端或者服务代理）绑定在一起。也就是说，InstanceContext和服务代理是具有一一对应的关系。

[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession)]

public class ServiceClass : IService1

{

}

1. **单例（Single）**

* 单例上下文实例模式，是指所有的客户端共享同一个服务实例的InstanceContext。

[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]

public class ServiceClass : IService1

{

}

例子：

1. 新建WCF服务库，契约Iservice1实现类（可以通过更换InstanceContextMode来测试三种的区别
2. ）

[ServiceBehavior(InstanceContextMode=InstanceContextMode.PerCall)]

public class Service1 : IService1

{

public Service1()

{

Console.WriteLine("实例已创建");

}

public string GetData(int value)

{

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

}

}

1. 新建控制台程序，启动服务

Uri baseAddress = new Uri("http://localhost:3200/test");

WSHttpBinding bind = new WSHttpBinding();

ServiceHost host = new ServiceHost(typeof(WcfServiceLibrary3.Service1), baseAddress);

host.AddServiceEndpoint(typeof(WcfServiceLibrary3.IService1),bind,"");

host.Open();

Console.WriteLine("服务正在运行中，地址是"+baseAddress);

1. 新建控制台客户端程序，调用服务

class Program

{

static WcfServiceLibrary3.IService1 proxy;

static void Main(string[] args)

{

WSHttpBinding bind = new WSHttpBinding();

EndpointAddress address = new EndpointAddress("http://localhost:3200/test");

ChannelFactory<WcfServiceLibrary3.IService1> factory = new ChannelFactory<WcfServiceLibrary3.IService1>(bind, address);

//通过通道工厂创建代理类

proxy = factory.CreateChannel();

Console.Write("回车开始：");

Console.ReadLine();

for (int i = 1; i < 6; i++)

{

Thread t = new Thread(new ThreadStart((ThreadMethod)));

t.Start();

Console.WriteLine("线程"+i+"已启动");

}

}

static void ThreadMethod()

{

proxy.GetData(1);

}

}