## 生成Service代理

使用SvcUtiil生成wcf服务的代理文件和配置文件

SvcUtil 服务地址?wsdl /out:代理文件名 /config:配置文件名



将config文件中的<system.serviceModel>拷贝到你的程序配置文件中

将代理类添加到你的程序中，添加程序集引用System.ServiceModel和System.Runtime.Serialization。

适当修改生成的类名以适合你的需要。然后就可以使用生成的ServiceClient调用你的服务了。

/// Tips:

/// 1:Always create the service with interface->Implementation format, mention the contract in interface.

/// 2:Define the service in class library and refer the class library in host project. Don't use service class in host project.

3:when we throw exception from service,it can't reach the client side.You must use throw new FaultException();

# WCF

### Host

IIS Host仅支持HTTP协议。Self host、Window Service、WAS支持所有的协议。

### Binding Type

* BasicHttpBinding

Suit for communicating with asp.net web services based services

This binding uses http

Security is disabled by default

Weak in interoperability

* WSHttpBinding

Defines a secure, reliable, interoperable binding suitable for non-duplex service contracts

offers lot more functionality in the area of interoperability

reliable and secure sessions using SOAP security

* WSDualHttpBinding

This binding is same as that of WSHttpBinding, except it supports duplex service

* NetTcpBinding

This binding provides secure and reliable binding environment for .Net to .Net cross machine communication. Both endpoints need the WCF.

It uses TCP protocol and provides support for security, transaction and reliability.

* NetNamedPipeBinding

provides secure and reliable binding environment for on-machine cross process communication. Same machine different process.

* NetMsmqBinding

provides secure and reliable queued communication for cross-machine environment

Queuing is provided by using MSMQ as transport

enables for disconnected operations, failure isolation and load leveling

* NetPeerTcpBinding

provides secure binding for peer-to-peer environment and network applications

uses TCP protocol for communication

## 注意点：

* 不要在Service接口中重载
* WCF使用DataContractSerializer来序列化和反序列化接口的参数和返回值的。
* DataContract标志类可被序列化，但每个字段必须用DataMember标记，Serializable标志类可以被序列化，不用序列化的字段需[**NonSerialized**]标记
* 兼容性，当客户端或者服务器端在Data Contract中新增一个Data Member项时，When deserializing the type, DataContractSerializer will simply ignore the new members. 因此，客户端和服务器端can accept data with new data members that were not part of the original contract.
* By default，客户端和服务器端都可以从Data Contract中移除Data Member。这意味着客户端或者服务器端可以序列化缺少某个Data Member的类型，发送给需要该Data Member的服务器端或者客户端。On the receiving side, DataContactSerializer 没有找到message to deserialize those members, it will silently deserialize them to their default values. Null for reference types and zero for value type.
* Data contract 可以实现方法**OnDeserializing OnSerializing等方法。**
* **Enumeration总是可以Serializable，所以不需要应用DataContract属性。**

Enumeration的DataContract项用EnumMember而不是DataMember来标记。Any enum value not decorated with the EnumMember attribute will not be part of the data contract for that enum

## Ninject依赖注入：

### Self Hosted

1. 通过 Nuget添加 **Ninject.Web.Common.Selfhost**

### Configure wcf service library：

1、通过 Nuget添加 **Ninject.Web.Common.Selfhost**

2、创建如下类

/// <summary>

/// 创建serviceHostFactory类

/// </summary>

public class NinjectWcfSeviceLibraryServiceHostFactory:NinjectServiceHostFactory

{

/// <summary>

/// 构造函数

/// </summary>

public NinjectWcfSeviceLibraryServiceHostFactory()

{

var kernel = new StandardKernel();

//var kernel = new StandardKernel(new ServiceModule());

//绑定dbContext在requestscope中

//kernel.Bind<DbContext>().ToSelf().InRequestScope();

kernel.Bind<ServiceHost>().To<NinjectServiceHost>();

SetKernel(kernel);

}

}

3、配置web.config文件

<serviceHostingEnvironment>

<serviceActivations>

<add factory="WcfServiceLibrary.NinjectWcfSeviceLibraryServiceHostFactory" service="WcfServiceLibrary.ServiceLayer.TimeService" relativeAddress="WcfServiceLibrary.ServiceLayer.TimeService.svc"/>

</serviceActivations>

</serviceHostingEnvironment>

### Configure wcf services using svc file

1、通过 Nuget添加 **Ninject.Web.Common.Selfhost**

2、配置Global.asax.cs,

继承自public class Global : Ninject.Web.Common.NinjectHttpApplication

并覆盖方法提供kernel

/// <summary>

/// 重载 提供kernel

/// </summary>

/// <returns></returns>

protected override Ninject.IKernel CreateKernel()

{

var kernel = new StandardKernel();

//绑定disposable对象在requestscope中

kernel.Bind<DisposableModel>().ToSelf().InRequestScope();

return kernel;

}

3、配置svc文件， codebehind不可少，否则可以将svc文件的内容清空，使用Configure wcf service library

<%@ ServiceHost

Language="C#"

Service="WcfTimeService.TimeService.TimeService" CodeBehind="TimeService.svc.cs"

Factory="Ninject.Extensions.Wcf.NinjectServiceHostFactory" %>