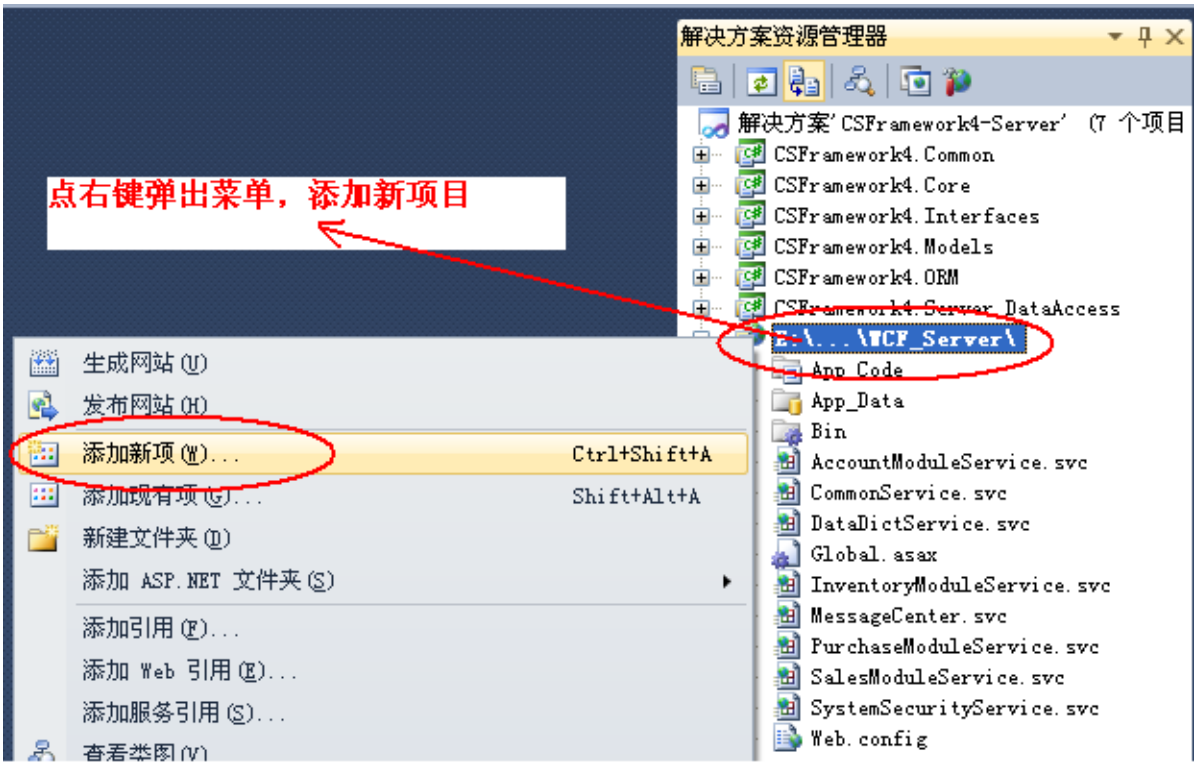
平台建立 WCF 服务操作指引

版本 v1.0
作者：JONNY
文档日期：2013-7-24

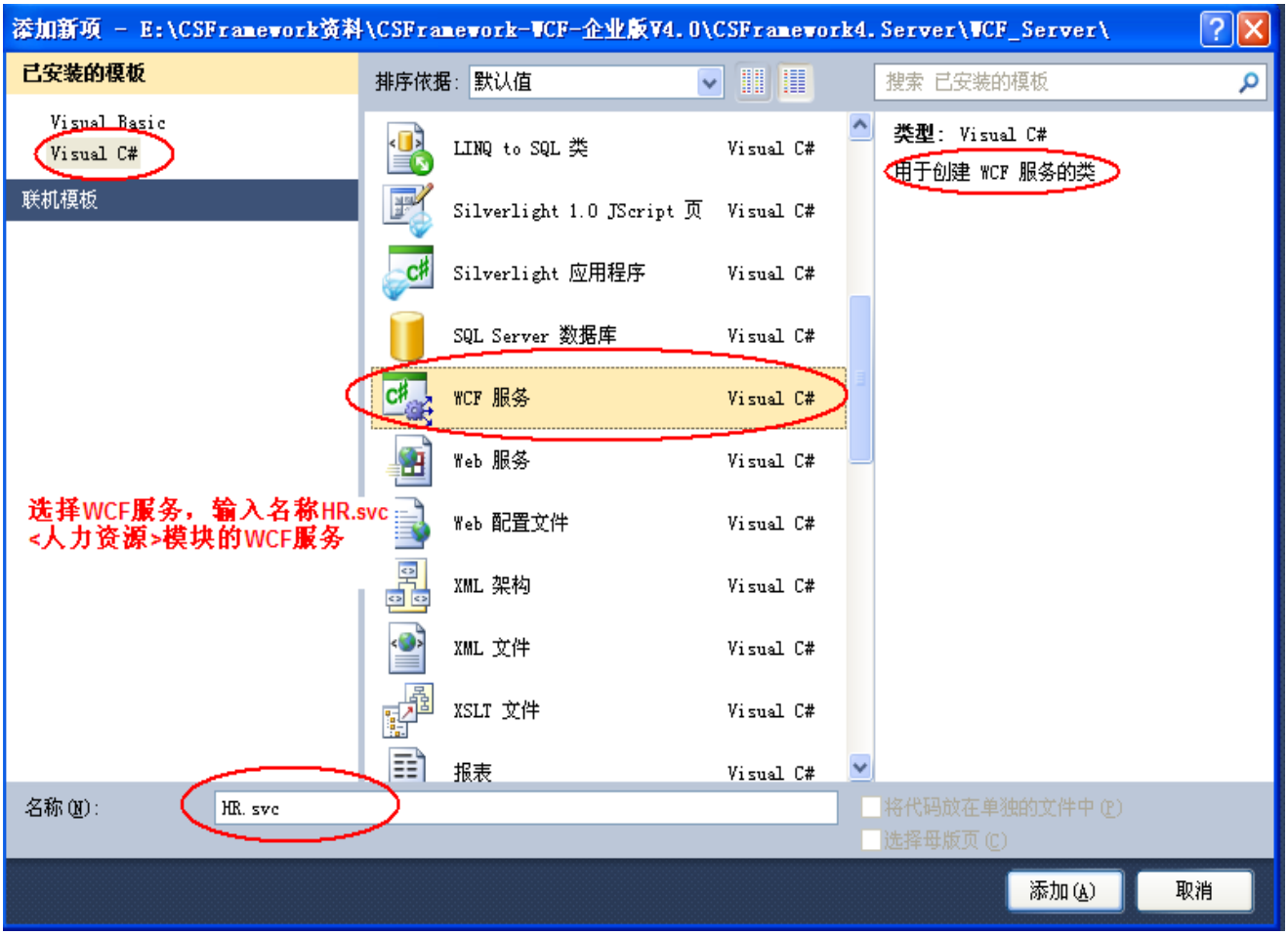
目 录

一. 在 WCF_Server 网站添加 WCF 服务

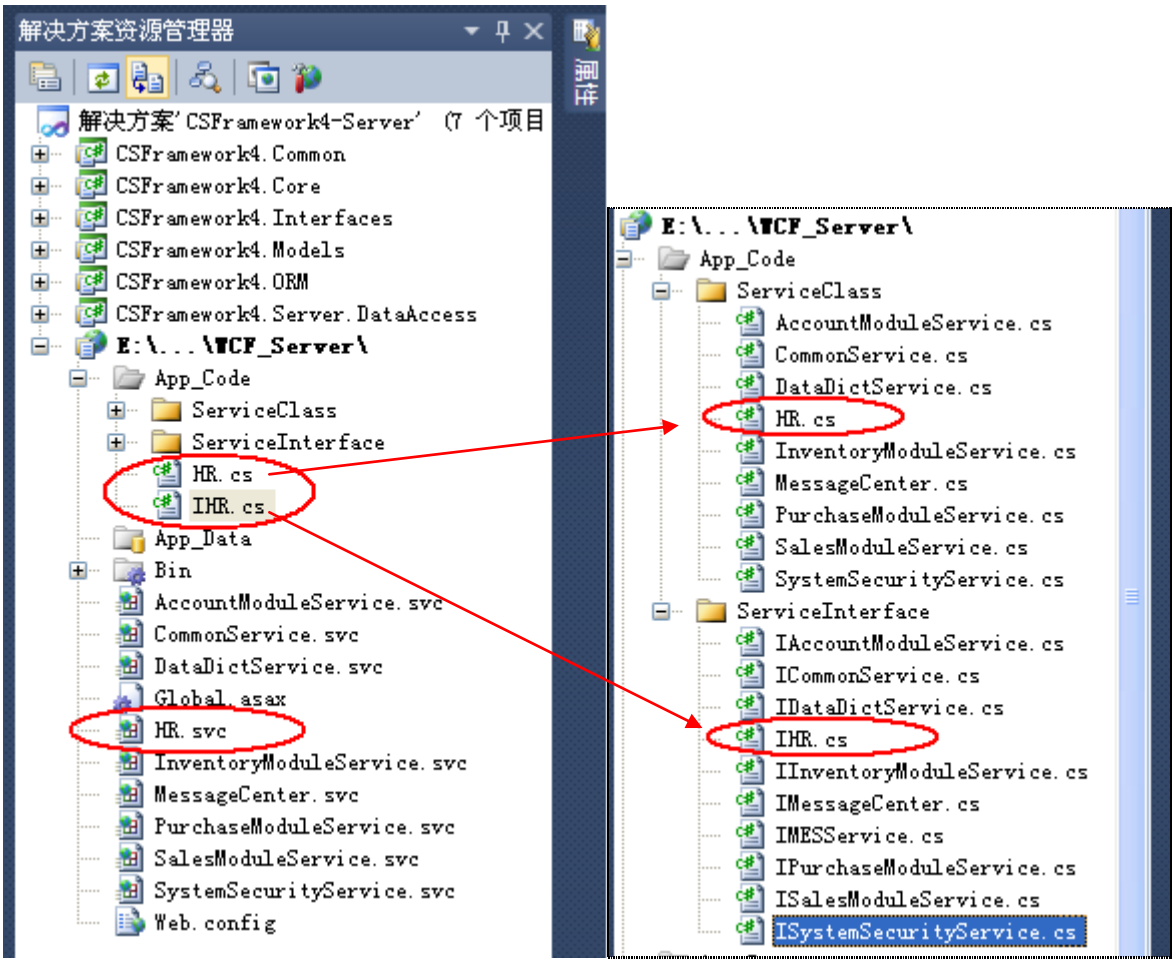
1.1 点右键弹出菜单，选择”添加新项”



1.2 选择 WCF 服务类型，输入名称 HR.svc，点添加按钮



1.3 将 WCF 服务接口文件及类文件复制到子目录统一管理



将接口文件 IHR.cs 复制到 ServiceInterface 子目录，HR.cs 类文件复制到 ServiceClass 子目录。

1.4 定义 WCF 服务的接口方法

```
using System.ServiceModel;
using System.Text;

// 注意：使用"重构"菜单上的"重命名"命令，可以同时更改代码和配置文件中的接口名"IHR"。
[ServiceContract]
public interface IHR
{
    [OperationContract]
    void DoWork();

    /// <summary>
    /// 获取员工表
    /// </summary>
    /// <param name="loginTicket">用户验证资料</param>
    /// <param name="employeeName">按员工姓名查询</param>
    /// <returns></returns>
    [OperationContract]
    byte[] GetEmplyoeeTable(byte[] loginTicket, string employeeName);

}
```

1.5 实现 WCF 服务的接口

```
// 注意：使用"重构"菜单上的"重命名"命令，可以同时更改代码、svc 和配置文件中的类名"HR"。
public class HR : IHR
{
    public void DoWork()
    {
    }

    public byte[] GetEmplyoeeTable(byte[] loginTicket, string employeeName)
    {
        try
        {
            Loginer loginer = WebServiceSecurity.ValidateLoginer(loginTicket);
            DataTable data = new dalBaseDataDict(loginer).GetDataDictByTableName("tb_EmpInfo");
            return ZipTools.CompressionDataSet(ServerLibrary.TableToDataSet(data));
        }
        catch (Exception ex)
        {
            throw new FaultException(ex.Message);
        }
    }
}
```

1.6 配置 web.config 文件

建立 WCF 服务后，web.config 文件内会自动生成预设的 Web 服务配置信息，需要优化参数配置。

1.6.1 配置 binding

修改参数 readerQuotas、maxStringContentLength、maxArrayLength 的值，如下图：

```
    <binding name="IMessageCenter" maxBufferPoolSize="524288" maxReceivedMessageSize="2147483647" messa
      <readerQuotas maxDepth="6553600" maxStringContentLength="2147483647" maxArrayLength="2147483647"
      <reliableSession ordered="true" inactivityTimeout="04:00:00" enabled="true"/>
      <security mode="None"/>
    </binding>
    <binding name="IHR" maxBufferPoolSize="524288" maxReceivedMessageSize="2147483647" messageEncoding=
      <readerQuotas maxDepth="6553600" maxStringContentLength="2147483647" maxArrayLength="2147483647"
      <reliableSession ordered="true" inactivityTimeout="04:00:00" enabled="true"/>
      <security mode="None"/>
    </binding>
```

1.6.2 配置 behavior

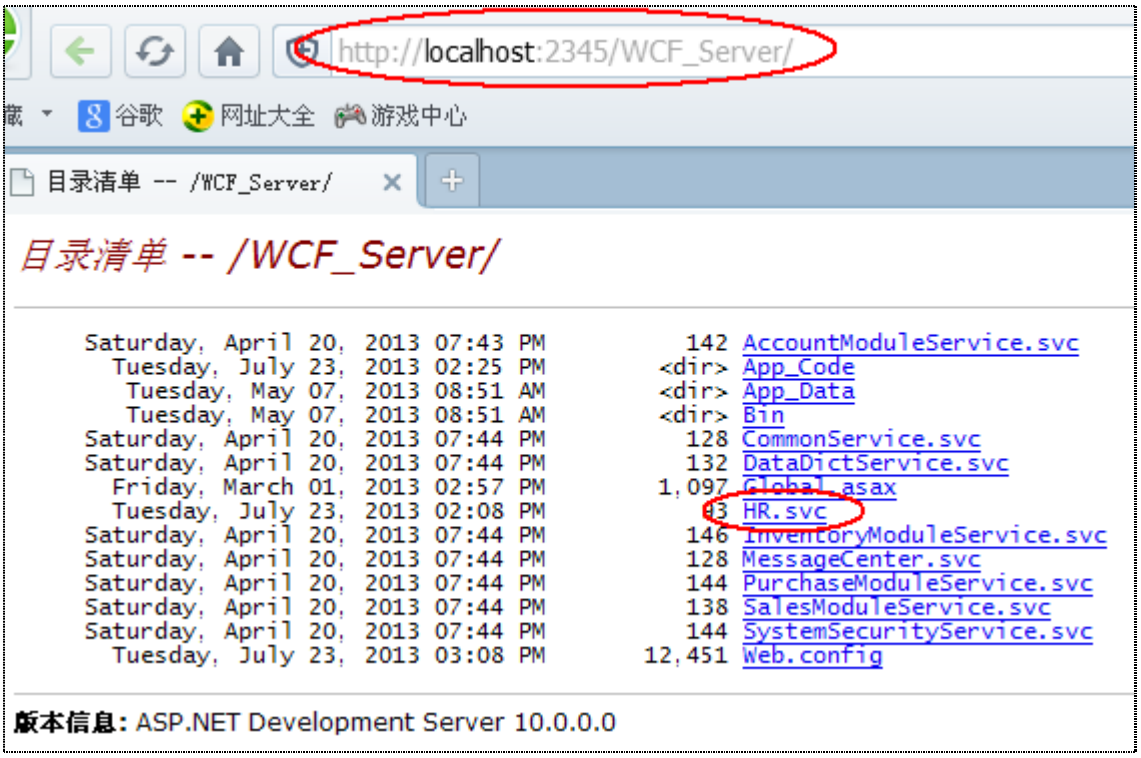
```
</behavior>
<behavior name="HRBehavior">
  <serviceMetadata httpGetEnabled="true" />
  <serviceDebug includeExceptionDetailInFaults="true" />
  <serviceThrottling maxConcurrentCalls="200" maxConcurrentSessions="200"
    maxConcurrentInstances="200" />
</behavior>
```

1.6.3 配置 service

Service 配置不需要优化配置，按系统生成的配置参数就行。

```
  </service>
  <service behaviorConfiguration="HRBehavior" name="HR">
    <endpoint address="" binding="wsHttpBinding" bindingConfiguration="IHR"
      contract="IHR">
      <identity>
        <dns value="localhost" />
      </identity>
    </endpoint>
    <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
  </service>
</services>
```

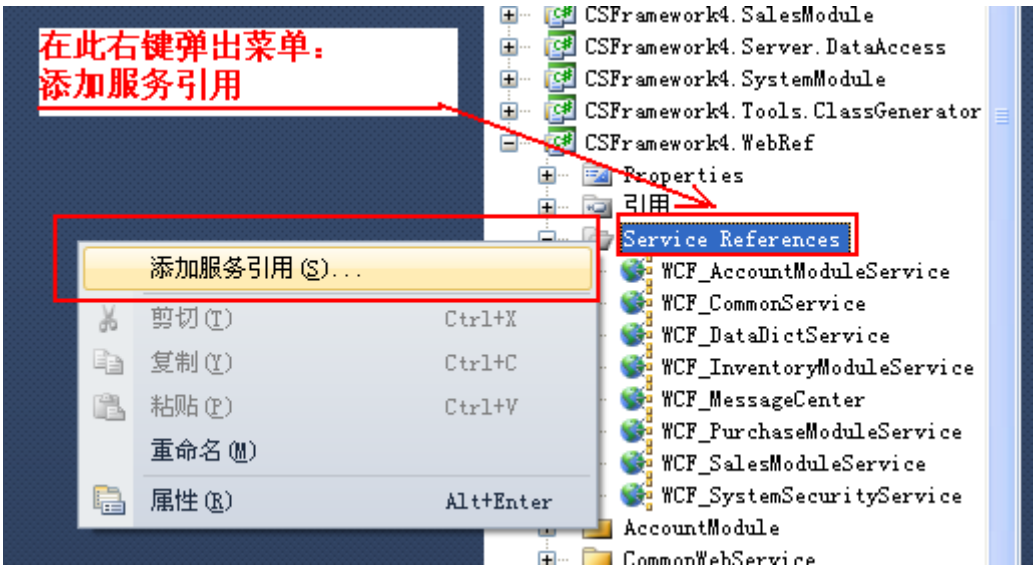
1.7 运行 WCF 服务



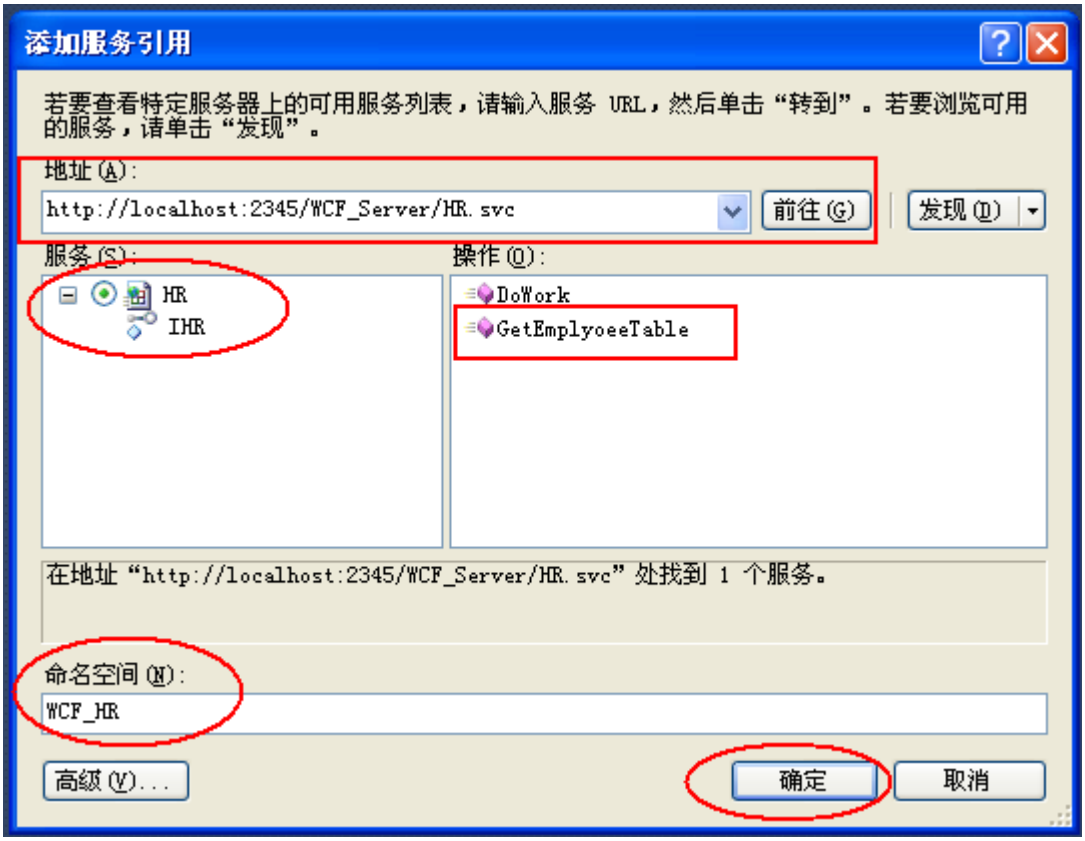
如上图所示，WCF 列表中 HR.svc 运行成功。

二．客户端引用(调用)WCF 服务

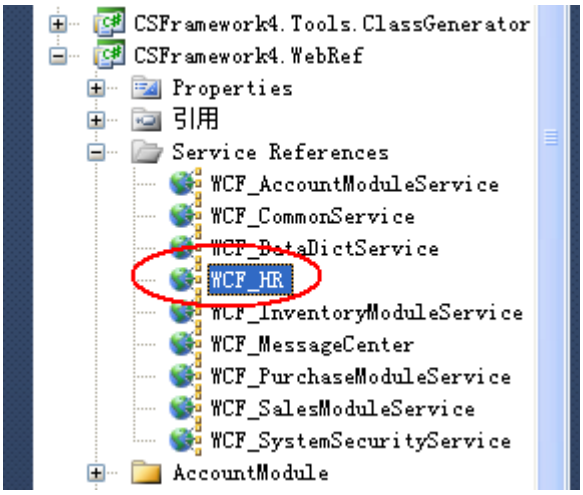
2.1 添加服务引用



复制 WCF 服务地址点“前往”按钮，系统会获取到 WCF 服务信息，如下图所示：



输入命名空间 WCF_HR，平台内所有 WCF 服务引用均以 WCF_打头，点确定按钮。



图示 WCF_HR 服务添加成功。

2.2 配置 app.config 文件

添加 WCF 引用后，app.config 文件生成预设的 binding 配置，请参考 app.config 文件其它 binding 例子，复制过来替换，然后配置 endpoint。

```
</binding>
<binding name="WSHttpBinding_IHR" closeTimeout="00:01:00" openTimeout="00:01:00"
  receiveTimeout="00:10:00" sendTimeout="00:01:00" bypassProxyOnLocal="false"
  transactionFlow="false" hostNameComparisonMode="StrongWildcard"
  maxBufferPoolSize="524288" maxReceivedMessageSize="65536" messageEncoding="Text"
  textEncoding="utf-8" useDefaultWebProxy="true" allowCookies="false">
  <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
    maxBytesPerRead="4096" maxNameTableCharCount="16384" />
  <reliableSession ordered="true" inactivityTimeout="04:00:00"
    enabled="true" />
  <security mode="None">
    <transport clientCredentialType="Windows" proxyCredentialType="None"
      realm="" />
    <message clientCredentialType="Windows" negotiateServiceCredential="true" />
  </security>
</binding>
```

加载WCF引用后自动生成的binging, 需要替换！！！

```
<binding name="WSHttpBinding_IHR" closeTimeout="00:01:00"
  openTimeout="00:01:00" receiveTimeout="04:00:00" sendTimeout="00:03:00"
  bypassProxyOnLocal="false" transactionFlow="false" hostNameComparisonMode="StrongWildcard"
  maxBufferPoolSize="524288" maxReceivedMessageSize="2147483647"
  messageEncoding="Text" textEncoding="utf-8" useDefaultWebProxy="true"
  allowCookies="false">
  <readerQuotas maxDepth="6553600" maxStringContentLength="2147483647"
    maxArrayLength="2147483647" maxBytesPerRead="6553600" maxNameTableCharCount="6553600" />
  <reliableSession ordered="true" inactivityTimeout="04:00:00"
    enabled="true" />
  <security mode="None">
    <transport realm="" />
  </security>
</binding>
```

在app.config文件内找个binding例子，复制过来替换

2.3 配置 endpoint

```
</endpoint>
<endpoint address="http://localhost:2345/WCF_Server/HR.svc"
  binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_IHR"
  contract="WCF_HR.IHR"
  name="WSHttpBinding_IHR">
  <identity>
    <dns value="localhost" />
  </identity>
</endpoint>
```

三．开发客户端演示功能

3.1 建立 WCF 客户端代理类(HRClient)的工厂方法

打开SoapClientFactory.cs文件，添加一个工厂方法用于创建WCF代理类(HRClient)的实例，如下图所示：

```
/// <summary>
/// 创建人力资源模块的WCF客户端实例
/// </summary>
/// <returns></returns>
public static HRClient CreateHRClient()
{
    WSHttpBinding BINDING = new WSHttpBinding();
    SoapClientConfig.ReadBindingConfig(BINDING, "WSHttpBinding_IHR");//从配置文件(app.config)读取配置。
    string endpoint = SoapClientConfig.GetSoapRemoteAddress("WSHttpBinding_IHR");
    return new HRClient(BINDING, new EndpointAddress(endpoint));//构建WCF客户端实例
}
```

3.2 建立桥接(策略)接口< IBridge_Employee>

```
namespace CSFramework4.Interfaces.Interfaces_Bridge
{
    /// <summary>
    /// 职员资料数据层桥接接口
    /// </summary>
    public interface IBridge_Employee
    {
        /// <summary>
        /// 查询职员数据，模糊查询
        /// </summary>
        /// <param name="employeeName">姓名，模糊查询</param>
        /// <returns></returns>
        DataTable GetEmplyoeeTable(string employeeName);
    }
```

3.3 建立 DAL 层

创建 dalEmployee，继承 dalBaseDataDict 数据字典基类，并实现< IBridge_Employee>接口，
请参考平台内完整的 dalEmployee 类定义。

```
/// <summary>
/// 职员管理的数据访问层
/// </summary>
[DefaultORM_UpdateMode(typeof(tb_Employee), true)]
public class dalEmployee : dalBaseDataDict, IBridge_Employee
{
    /// <summary>
    /// 构造器
    /// </summary>
    /// <param name="loginer">当前登录用户</param>
    public dalEmployee(Loginer loginer)
        : base(loginer)
    {
        _KeyName = tb_Employee.__KeyName; //主键字段
        _TableName = tb_Employee.__TableName;//表名
        _ModelType = typeof(tb_Employee);
    }
```

3.4 建立 BLL 层

创建 bllEmployee，继承 bllBaseDataDict 数据字典基类，请参考平台内完整的 bllEmployee 类定义。

```
/// <summary>
/// 职员资料业务逻辑层
/// </summary>
public class bllEmployee : bllBaseDataDict
{
    private IBridge_Employee _MyBridge; //桥接/策略接口
    public bllEmployee()
    {
        _KeyFieldName = tb_Employee.__KeyName; //主键字段
        _SummaryTableName = tb_Employee.__TableName;//表名
        _DataDictBridge = BridgeFactory.CreateDataDictBridge(typeof(tb_Employee));
        _MyBridge = this.CreateBridge();
    }
```

3.5 建立测试窗体

查询按钮源码：

```
namespace CSFramework4.DataDictionary
{
    public partial class frmEmployee : CSFramework4.Library.frmBaseChild
    {
        public frmEmployee()
        {
            InitializeComponent();
        }

        private void frmEmployee_Load(object sender, EventArgs e)
        {
            this.InitButtons();
        }

        private void btnSearch_Click(object sender, EventArgs e)
        {
            //调用BLL层，取数据
            DataTable dt = new bllEmployee().GetEmplyoeeTable(textEdit1.Text);
            gridControl1.DataSource = dt;
        }
    }
}
```

*** 文档存在不完整的地方，请 QQ 通知小二，谢谢~ ***

=== 文档完 ===