**武汉大学计算机学院**

**本科生课程设计报告**

**软件设计与体系结构课程设计**

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：软件设计与体系结构

指 导 教 师 ：王健 讲师

学 生 学 号 ：2018302110134 郑子杰

学 生 学 号 ：2018302110214 陈惟逸

学 生 学 号 ：2018302110158 梁志杰

二〇二〇 年 十一 月

**郑 重 声 明**

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

本人签名：郑子杰 陈惟逸 梁志杰 日期：2020.11.17

软件架构与体系设计

年轻人冲浪交友软件——*Surfinghub Application*

Contents

[实验一 4](#_Toc56672514)

[用例图 4](#_Toc56672515)

[类图 5](#_Toc56672516)

[时序图 6](#_Toc56672517)

[实验二 7](#_Toc56672518)

[简单工厂设计模式 7](#_Toc56672519)

[系统操作类 7](#_Toc56672520)

[增加好友类： 10](#_Toc56672521)

[删除好友类： 10](#_Toc56672522)

[简单工厂类： 12](#_Toc56672523)

[客户端调用代码： 12](#_Toc56672524)

[简单工厂类的优劣： 13](#_Toc56672525)

[设计思路： 13](#_Toc56672526)

[备忘录模式 14](#_Toc56672527)

[备忘录类： 14](#_Toc56672528)

[个人信息类： 14](#_Toc56672529)

[管理者类： 14](#_Toc56672530)

[备忘录模式优劣： 15](#_Toc56672531)

[设计思路： 15](#_Toc56672532)

[实验三 16](#_Toc56672533)

[SOAP、REST API的实现 16](#_Toc56672534)

[客户端调用 19](#_Toc56672535)

[参考文献 26](#_Toc56672536)

# 实验一

## 用例图



分为用户、管理员两个actor

用户：

1. 进入软件界面选择注册、登录、修改信息
2. 进入用户界面后，可自行选择体验冲浪中心功能（包括查看好友，发起聊天，进入博客，多媒体播放）
3. 其中每个功能还包括拓展功能：

* 查看好友：添加、删除好友
* 发起聊天：查看、删除聊天记录
* 进入博客：查看好友博客，编写自己的博客
* 多媒体播放：在线播放音乐及视频

管理员：

1. 修改用户信息
2. 发布公告

## 类图



登录与注册、操作多媒体、操作博客、发起聊天接口分别对应自己所提供服务的操作；

登录注册类包括用户ID、用户密码两个属性，可以进行注册、登录两种操作；

## 时序图



顺序图分为用户、登录系统、服务器、交互系统四个类；

可以处理用户登录界面、用户聊天界面的事件，它们是按时间顺序先后进行的。

# 实验二

## 简单工厂设计模式

系统操作类：其中对虚方法friendmanage由后续的操作子类继承后重构

class systemopera

**{**

**public** const string ConnectStr **=** "server = 127.0.0.1; port = 3306;user = root;password = winson; database = surfhub;"**;**

**public** static string CurUserName**;**

static **public** int initial**()**

**{**

**try**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

dbon**.**Open**();**

string createtable\_custom **=** "create table if not exists custom ( name varchar(15), id\_card char(18), psw varchar(25), gender varchar(4), phone\_number char(11), primary key(name));"**;**

MySqlCommand cmd **=** **new** MySqlCommand**(**createtable\_custom**,** dbon**);**

cmd**.**ExecuteNonQuery**();**

string createtable\_friend **=** "create table if not exists friends ( name varchar(15), friendsname char(18));"**;**

cmd **=** **new** MySqlCommand**(**createtable\_friend**,** dbon**);**

cmd**.**ExecuteNonQuery**();**

string createtable\_sign **=** "create table if not exists sign(name varchar(15), signs char(18))"**;**

cmd **=** **new** MySqlCommand**(**createtable\_sign**,** dbon**);**

cmd**.**ExecuteNonQuery**();**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**.**ToString**());**

**}**

**return** 0**;**

**}**

static **public** int register**(**string user\_name**,** string psw**,** string id**,** string phone\_number**,** string gender**)**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

int res **=** 0**;**

**try**

**{**

dbon**.**Open**();**

string insert\_custom **=** "insert into custom values ('" **+** user\_name **+** "', '" **+** id **+** "', '" **+** psw **+** "', '" **+** gender **+** "', '" **+** phone\_number **+** "');"**;**

MySqlCommand cmd\_insert **=** **new** MySqlCommand**(**insert\_custom**,** dbon**);**

cmd\_insert**.**ExecuteNonQuery**();**

string insert\_sign **=** "insert into sign values ('" **+** user\_name **+** "','Nothing');"**;**

cmd\_insert **=** **new** MySqlCommand**(**insert\_sign**,** dbon**);**

cmd\_insert**.**ExecuteNonQuery**();**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**.**ToString**());**

res **=** 1**;**

**}**

**finally** **{** dbon**.**Close**();** **}**

**return** res**;**

**}**

**public** static int Login**(**string user\_name**,** string user\_psw**)**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

int res **=** 0**;**

**try**

**{**

dbon**.**Open**();**

string text **=** "select \* from custom where name = '" **+** user\_name **+** "';"**;**

MySqlCommand cmd\_login **=** **new** MySqlCommand**(**text**,** dbon**);**

MySqlDataReader dr**;**

dr **=** cmd\_login**.**ExecuteReader**();**

bool null\_flag **=** **true;**

**while** **(**dr**.**Read**())**

**{**

null\_flag **=** **false;**

**if** **(**dr**[**"psw"**].**ToString**()** **==** user\_psw**)**

**{**

CurUserName **=** dr**[**"name"**].**ToString**();**

**}**

**else** res **=** 2**;**

**}**

dr**.**Close**();**

//用户不存在！

**if** **(**null\_flag**)** res **=** 1**;**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**.**ToString**());**

**}**

**finally**

**{**

dbon**.**Close**();**

**}**

**return** res**;**

**}**

**public** **virtual** void friendmanage**(**string friendsname**)**//使用简单工厂模式实现好友的添加与删除，后期还可增加其他管理功能

**{**

**}**

**public** static void signing**(**string signs**)**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

**try**

**{**

dbon**.**Open**();**

string updatesign **=** "update sign set signs = '" **+** signs **+** "' where name ='" **+** CurUserName **+** "';"**;**

MySqlCommand cmd\_update **=** **new** MySqlCommand**(**updatesign**,** dbon**);**

cmd\_update**.**ExecuteNonQuery**();**

**}**

**catch(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**.**ToString**());**

**}**

**finally**

**{**

dbon**.**Close**();**

**}**

**}**

**}**

增加好友类：对父类中的虚方法进行重构，调用时通过简单工厂类实例化

class operationaddfriends**:**systemopera//增加好友类

**{**

**public** **override** void friendmanage**(**string friendsname**)**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

**try**

**{**

dbon**.**Open**();**

string insert\_custom **=** "insert into friends(name,friendsname) select '" **+** CurUserName **+** "', '" **+** friendsname **+** "' from DUAL where not exists(select friendsname from friends where friendsname = '" **+** friendsname **+** "');"**;**

MySqlCommand cmd\_insert **=** **new** MySqlCommand**(**insert\_custom**,** dbon**);**

cmd\_insert**.**ExecuteNonQuery**();**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**.**ToString**());**

**}**

**finally**

**{**

dbon**.**Close**();**

**}**

**}**

**}**

删除好友类：对父类中的虚方法进行重构，调用时通过简单工厂类实例化

class operationdeletefriends **:** systemopera//删除好友类

**{**

**public** **override** void friendmanage**(**string friendsname**)**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

**try**

**{**

dbon**.**Open**();**

string insert\_custom **=** "delete from friends where friendsname = '" **+** friendsname **+** "';"**;**

MySqlCommand cmd\_insert **=** **new** MySqlCommand**(**insert\_custom**,** dbon**);**

cmd\_insert**.**ExecuteNonQuery**();**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**.**ToString**());**

**}**

**finally**

**{**

dbon**.**Close**();**

**}**

**}**

**}**

简单工厂类：通过客户端代码调用实例化对应的类

class friendsmangefactory//简单工厂类

**{**

**public** static systemopera createfriendsmanage**(**string manage**)**

**{**

systemopera oper **=** **null;**

**switch** **(**manage**)**

**{**

**case** "+"**:**

oper **=** **new** operationaddfriends**();**

**break;**

**case** "-"**:**

oper **=** **new** operationdeletefriends**();**

**break;**

**default:**

**break;**

**}**

**return** oper**;**

**}**

**}**

### 客户端调用代码：

**private** void button1\_Click**(object** sender**,** EventArgs e**)**//简单工厂模式实现好友的增加与删除

**{**

string friendsname **=** **this.**textBox1**.**Text**.**ToString**();**

systemopera oper**;**

oper **=** friendsmangefactory**.**createfriendsmanage**(**"+"**);**

oper**.**friendmanage**(**friendsname**);**

showinfo**();**

**}**

**private** void button1\_Click\_1**(object** sender**,** EventArgs e**)**

**{**

string friendsname **=** **this.**textBox1**.**Text**.**ToString**();**

systemopera oper**;**

oper **=** friendsmangefactory**.**createfriendsmanage**(**"-"**);**

oper**.**friendmanage**(**friendsname**);**

showinfo**();**

**}**

### 简单工厂类的优劣：

优：在本案例中，通过简单工厂类可以使客户端免除直接创作产品对象的责任，同时，客户端无须知道所创建的具体产品类的类名，只需要知道具体产品类所对应的参数如本案例中只需输入参数+与-即可，对于一些复杂的类名，通过简单工厂模式可以减少使用者的记忆量。

劣：工厂类集中了所有产品创建逻辑，一旦不能正常工作，整个系统都要受到影响。同时，使用简单工厂类会使系统中的类的个数变多，增加了系统的复杂程度。系统的可扩展性也被下降，一旦要添加新的功能就必须修改工厂逻辑，破坏了软件的开闭原则。

### 设计思路：

在设计管理好友的功能初期，由于聊天社交软件对好友的管理仅限于添加与删除好友以及给好友增加备注名称。故思考使用工厂模式来实现以上功能，由于需要实现的功能个数少，可拓展功能不多，故挑选了简单工厂类进行设计。

## 备忘录模式

### 备忘录类：

class Memento//备忘录类

**{**

**private** string signs**;**

**public** Memento**(**string signs**)**//导入要保存的数据

**{** **this.**signs **=** signs**;** **}**

**public** string Signs//需要保存的数据，本例中为个人的个性签名，后期可适当拓展属性

**{**

**get** **{** **return** signs**;** **}**

**}**

**}**

### 个人信息类：

class originator//个人信息类

**{**

**private** string signs**;**//需要保存的属性

**public** string Signs

**{**

**get** **{** **return** signs**;** **}**

**set** **{** signs **=** **value;** **}**

**}**

**public** Memento CreateMemento**()**

**{**

**return** **(new** Memento**(**signs**));**

**}**

**public** void SetMento**(**Memento memento**)**

**{**

signs **=** memento**.**Signs**;**

**}**

**public** string show**()**

**{**

**return** signs**;**

**}**

**}**

### 管理者类：

class Caretaker

**{**

**private** Memento memento**;**

**public** Memento Memento//设置备忘录属性

**{**

**get** **{** **return** memento**;** **}**

**set** **{** memento **=** **value;** **}**

**}**

**}**

### 备忘录模式优劣：

优点：有时一些发起人对象的内部信息必须保存在发起人对象以外的地方，但是必须要由发起人对象自己读取，这时，使用备忘录模式可以把复杂的发起人内部信息对其他的对象屏蔽起来，从而可以恰当地保持封装的边界。在本例中，用户个人的个性签名则是只能由用户自己读取的。

本模式简化了发起人类。发起人不再需要管理和保存其内部状态的一个个版本，客户端可以自行管理他们所需要的这些状态的版本。

缺点：

发起人角色的状态如果需要完整地存储到备忘录对象中，那么在资源消耗上面备忘录对象空间会很昂贵，所占用的内存空间十分庞大。

当负责人角色存储一个发起者的备忘录时，负责人并不知道当前备忘录状态需要多大的空间，从而无法体现用户当前操作的代价是否很昂贵。

### 设计思路：

本案例中实现的是对用户自身个性签名的修改撤销，用户常常会在修改了自己的上一条个性状态之后反悔想要取消，一时间可能忘了上一条状态的内容。此时备忘录模式则能很好的实现撤销功能。同时，备忘录模式常常用于游戏中的存档功能，在本实例中可以把用户发布个性签名当作一个游戏，为每一次修改发布个性签名存一个档，当用户返回时则可以回档。同时个性签名所占用的内存空间通常不会很大，代价很低，可以很好的避免备忘录模式的缺点。

# **实验三**

## SOAP、REST API的实现

本部分实现一个我们软件设计中的获取用户好友信息、修改与获取用户个性签名的webservice服务。

在该实验中，我们使用访问数据库的方式来获取用户的好友信息、个性签名以及修改用户个性签名。通过getfriends（），getsigns（），updatesigns（）三个方法实现服务。

Getfriends（）对应的关键代码如下：

**public** string GetFriends**(**string Name**)**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

dbon**.**Open**();**

string sqlstring **=** "select friendsname from friends where name = '" **+** Name **+** "'; "**;**

MySqlCommand cmd **=** **new** MySqlCommand**(**sqlstring**,** dbon**);**

DataSet ds **=** **new** DataSet**();**

**try**

**{**

MySqlDataAdapter da **=** **new** MySqlDataAdapter**(**sqlstring**,** dbon**);**

da**.**Fill**(**ds**,** "tb"**);**

**}**

**catch(**MySqlException E**)**

**{**

**throw** **new** Exception**(**E**.**Message**);**

**}**

dbon**.**Close**();**

XDocument xDoc **=** **new** XDocument**();**

XElement root **=** **new** XElement**(**"Root"**);**

xDoc**.**Add**(**root**);**

**try**

**{**

**for(**int i**=**0**;**ds**.**Tables**.**Count**>**0**&&**i**<**ds**.**Tables**[**0**].**Rows**.**Count**;**i**++)**

**{**

XElement xele **=** **new** XElement**(**"Data" **+** **(**i **+** 1**).**ToString**());**

root**.**Add**(**xele**);**

**for(**int j**=**0**;**j**<**e**.**Length**;**j**++)**

**{**

xele**.**SetElementValue**(**e**[**j**],** ds**.**Tables**[**0**].**Rows**[**i**].**ItemArray**[**j**]);**

**}**

LogWriteLock**.**EnterWriteLock**();**

xDoc**.**Save**(**CurrPath **+** "friends.xml"**);**

LogWriteLock**.**ExitWriteLock**();**

**}**

**}**

**catch** **(**MySqlException E**)**

**{**

**throw** **new** Exception**(**E**.**Message**);**

**}**

**return** xDoc**.**ToString**();**

**}**

代码中先是对服务器本机的数据库进行访问，获取对应列的信息，接着是将获取到的信息写成XML文档的形式。Getsigns（）与updatesigns（）的设计实现思路同上，仅是数据库访问操作上有所差别，getsigns（）实现代码如下：

**public** string GetSigns**(**string Name**)**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

dbon**.**Open**();**

string sqlstring **=** "select signs from sign where name = '" **+** Name **+** "'; "**;**

MySqlCommand cmd **=** **new** MySqlCommand**(**sqlstring**,** dbon**);**

DataSet ds **=** **new** DataSet**();**

**try**

**{**

MySqlDataAdapter da **=** **new** MySqlDataAdapter**(**sqlstring**,** dbon**);**

da**.**Fill**(**ds**,** "tb"**);**

**}**

**catch** **(**MySqlException E**)**

**{**

**throw** **new** Exception**(**E**.**Message**);**

**}**

dbon**.**Close**();**

XDocument xDoc **=** **new** XDocument**();**

XElement root **=** **new** XElement**(**"Root"**);**

xDoc**.**Add**(**root**);**

**try**

**{**

**for** **(**int i **=** 0**;** ds**.**Tables**.**Count **>** 0 **&&** i **<** ds**.**Tables**[**0**].**Rows**.**Count**;** i**++)**

**{**

XElement xele **=** **new** XElement**(**"Data" **+** **(**i **+** 1**).**ToString**());**

root**.**Add**(**xele**);**

**for** **(**int j **=** 0**;** j **<** f**.**Length**;** j**++)**

**{**

xele**.**SetElementValue**(**f**[**j**],** ds**.**Tables**[**0**].**Rows**[**i**].**ItemArray**[**j**]);**

**}**

LogWriteLock**.**EnterWriteLock**();**

xDoc**.**Save**(**CurrPath **+** "signs.xml"**);**

LogWriteLock**.**ExitWriteLock**();**

**}**

**}**

**catch** **(**MySqlException E**)**

**{**

**throw** **new** Exception**(**E**.**Message**);**

**}**

**return** xDoc**.**ToString**();**

**}**

Updatesigns（）实现代码如下：

**public** string Updatesigns**(**string Name**,**string signs**)**

**{**

MySqlConnection dbon **=** **new** MySqlConnection**(**ConnectStr**);**

dbon**.**Open**();**

string sqlstring **=** "update sign set signs = '" **+** signs **+** "' where name ='" **+** Name **+** "';"**;**

MySqlCommand cmd **=** **new** MySqlCommand**(**sqlstring**,** dbon**);**

DataSet ds **=** **new** DataSet**();**

**try**

**{**

MySqlDataAdapter da **=** **new** MySqlDataAdapter**(**sqlstring**,** dbon**);**

da**.**Fill**(**ds**,** "tb"**);**

**}**

**catch** **(**MySqlException E**)**

**{**

**throw** **new** Exception**(**E**.**Message**);**

**}**

dbon**.**Close**();**

XDocument xDoc **=** **new** XDocument**();**

XElement root **=** **new** XElement**(**"Root"**);**

xDoc**.**Add**(**root**);**

**try**

**{**

XElement xele **=** **new** XElement**(**"修改成功"**);**

root**.**Add**(**xele**);**

LogWriteLock**.**EnterWriteLock**();**

xDoc**.**Save**(**CurrPath **+** "friends.xml"**);**

LogWriteLock**.**ExitWriteLock**();**

**}**

**catch** **(**MySqlException E**)**

**{**

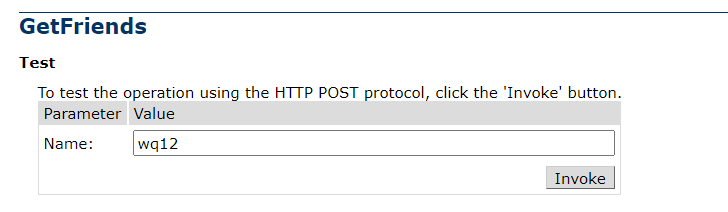
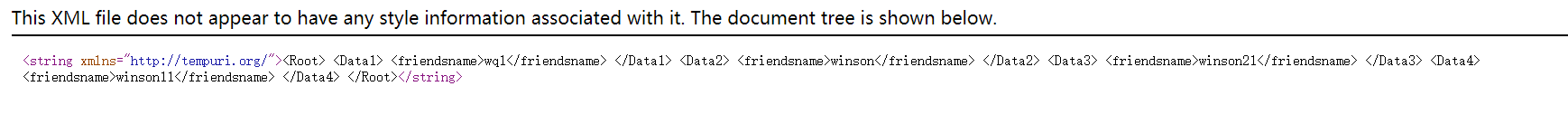
**throw** **new** Exception**(**E**.**Message**);**

**}**

**return** xDoc**.**ToString**();**

**}**

Webservice服务实现截图，仅展示getfriends（）实现效果，其他两种服务效果类似：



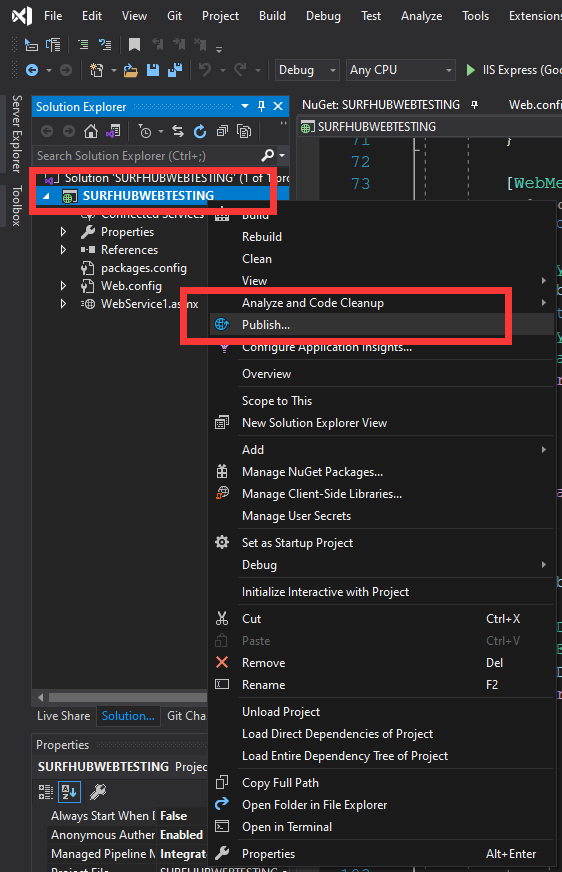
## 客户端调用

摘要：

为了实现客户端对API的调用，首先要对API进行“发布”操作，接着在客户端对发布好的API进行“引用”操作，最后通过代码调用它们。本部分将从如何发布及如何引用进行论述。

一、发布

0.对编写好的webservice进行操作——右键后选择发布

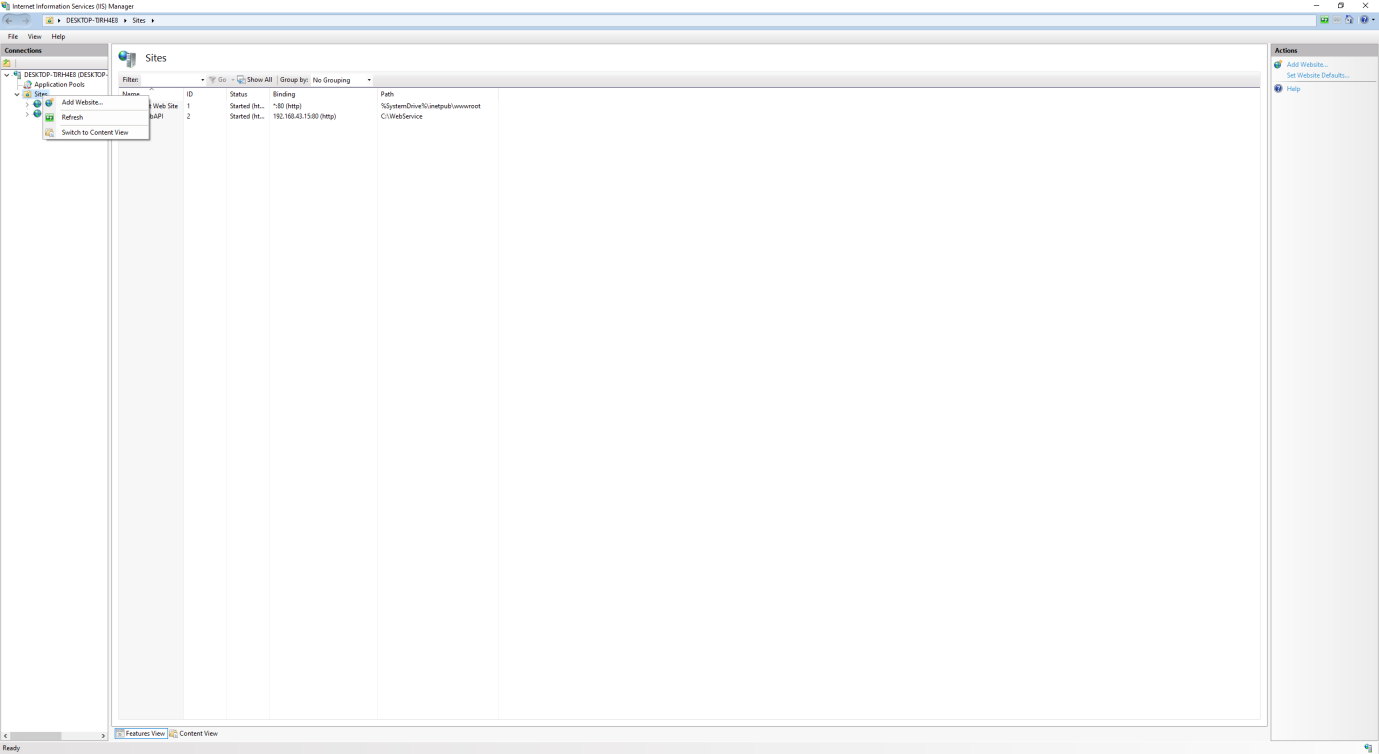


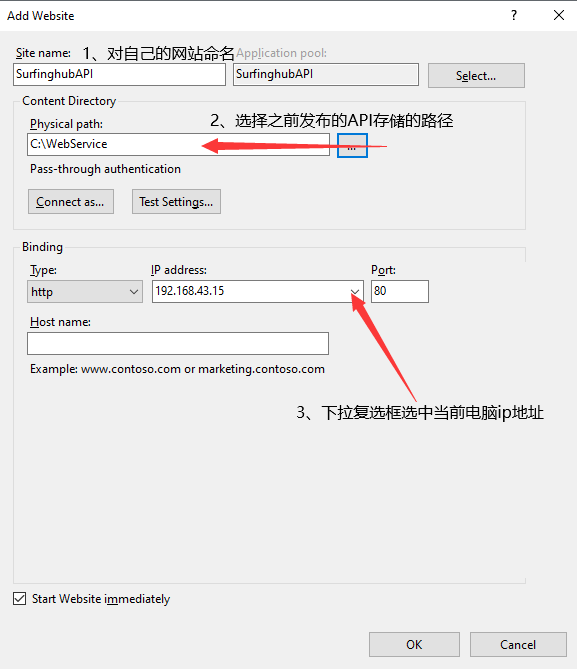
1.本实验中，将服务部署到本地，故选择本地文件夹进行服务的存储

之后选择浏览，选择存放的路径，点击finish完成发布的基本操作。

|  |  |
| --- | --- |
|  |  |

2.接着来到llS Manager，右击击网站，选择添加网站，并按下图操作填写相关信息。





（因发布在本机，故IP地址选择为当前电脑的IP地址）

3.右击新建的网站、管理网站、浏览；这时会跳转到浏览器，复制地址栏中的url（客户端调用时需要）

|  |
| --- |
|  |
|  |

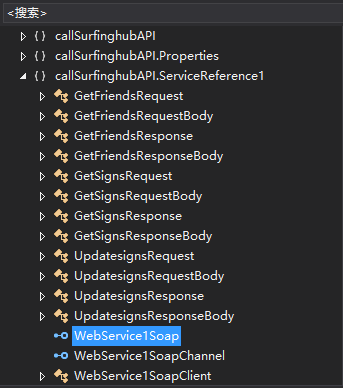
截至当前步骤，API以及WebService的发布已经全部完成。

二、调用

1. 在引用中添加服务引用；
2. 在地址栏中粘贴浏览器显示的 url，点击转到可以验证此网址有效性（若出现有图下部红框内容，则表示url有效）；
3. 输入自定义服务命名空间，点击确定。

|  |  |
| --- | --- |
|  |  |

查看对象浏览器可以看到webservice中对RESTAPI的封装，消息机制基于soap协议。



RESTAPI简介

REST是Representational State Transfer（表现层状态转移）的缩写，它是由罗伊·菲尔丁（Roy Fielding）提出的，是用来描述创建HTTP API的标准方法的，他发现这四种常用的行为（查看（view），创建（create），编辑（edit）和删除（delete））都可以直接映射到HTTP 中已实现的GET,POST,PUT和DELETE方法

（附：HTTP 中的8中不同的方法：

）

SOAP 简介

SOAP（Simple Object Accrss Protocol，简单对象访问协议）是一种简单的基于XML的协议，可以使应用程序在分散或分布式的环境中通过HTTP来交换信息。

SOAP消息基本上是从发送端到接收端的单向传输，常常结合起来执行类似于请求/应答的模式。不需要把SOAP消息绑定到特定的协议，SOAP可以运行在任何其他传输协议（HTTP、SMTP、FTP等）上。另外，SOAP提供了标准的RPC方法来调用Web Service以请求/响应模式运行。

SOAP是Web Service的通信协议。当用户通过UDDI找到你的WSDL描述文档后，他通过可以SOAP调用你建立的Web服务中的一个或多个操作。SOAP是XML文档形式的调用方法的规范，可以支持不同的底层接口，像HTTP(S)或者SMTP。

SOAP 消息组成

所有的SOAP消息都使用XML编码，一条SOAP消息就是一个普通的XML文档，文档包括下列元素：

A、Envelope（信封）元素，必选，可把此XML文档标识为一条SOAP消息。

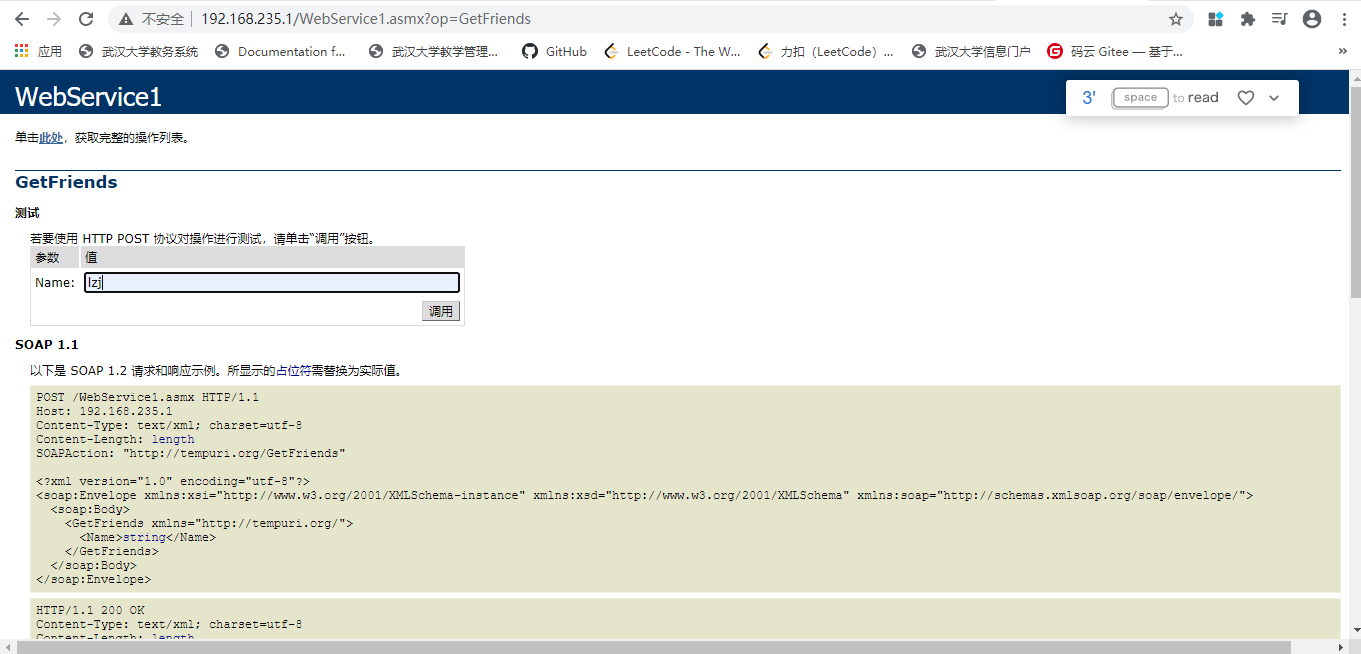
B、Header（报头）元素，可选，包含头部信息（包含了使消息在到达最终目的地之前，能够被路由到一个或多个中间节点的信息）。

C、Body(主体)元素，必选，包含所有的调用和响应信息。

D、Fault元素，位于Body内，可选，提供有关处理此消息所发生错误的信息。

E、Attachment（附件）元素，可选，可通过添加一个或多个附件扩展SOAP消息。

例：部署于本地的surfAPI的soap消息机制



1. 编写客户端各控件代码

API调用结果展示：

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

可以看到，当我们在客户端点击相应功能的按钮时，客户端的下端会同时打印出我们调用自己api后返回的soap格式信息，并且信息的正确性与数据库中的内容一致。

各按键对应事件的代码如下：

**private** void getFriendsBtn\_Click**(object** sender**,** EventArgs e**)**

**{**

WebService1SoapClient client **=** **new** WebService1SoapClient**();**

string name **=** inputNameBox**.**Text**;**

outputBox**.**Text **=** client**.**GetFriends**(**name**);**

**}**

**private** void editSignBtn\_Click**(object** sender**,** EventArgs e**)**

**{**

WebService1SoapClient client **=** **new** WebService1SoapClient**();**

string name **=** inputNameBox**.**Text**;**

string sign **=** inputSignBox**.**Text**;**

outputBox**.**Text **=** "新个性签名\r\n" **+** "----------------------------------------\r\n" **+**

client**.**GetSigns**(**name**)+** "\r\n\r\n\r\n" **+**

"修改反馈\r\n" **+** "----------------------------------------\r\n" **+**

client**.**Updatesigns**(**name**,** sign**);**

**}**

**private** void getSignBtn\_Click**(object** sender**,** EventArgs e**)**

**{**

WebService1SoapClient client **=** **new** WebService1SoapClient**();**

string name **=** inputNameBox**.**Text**;**

outputBox**.**Text **=** client**.**GetSigns**(**name**);**

**}**

# 参考文献

[1] 李必信，廖力，王璐璐，孔祥龙，周颖. 软件架构理论与实践. 2019：23-53.

[2] 李必信，廖力，王璐璐，孔祥龙，周颖. 软件架构理论与实践. 2019：154-174.