网络访问

中山大学　郑贵锋

通过这一节课，可以使学生了解网络访问的概念和具体实现方法。

大纲

1 HTTP请求

2 WCF数据服务

3 Socket通信

HTTP请求

HTTP请求

- HTTP请求是Windows 8 网络服务通信的第一步，通过HTTP请求可以访问网络中的各种资源，比如图像、文档以及网络服务。

- 本节重点介绍如何使用HttpClient和HttpWebRequest类来访问和获取网络资源

HttpClient类

HttpClient类包含在 System.Net.Http 命名空间中，
是向以URI标识的网络资源发送HTTP请求和接收HTTP响应的基类。

在HTTP请求中使用该类可以向Web服务发送Get、Post等异步请求，并接收服务器返回的响应数据。

采用异步方式获取网络数据，可以防止占用UI线程造成混乱，也会降低网络连接失去响应的几率，
尤其是当执行长时间运行或者高延迟的任务时采用异步方式的优势会很明显。

HttpClient类

Get、Post:

```
// Create a New HttpClient object.
HttpClient client = new HttpClient();
//get
HttpResponseMessage response =
      await client.GetAsync(“/*resourceUri*/”);
//will throw an exception if not successful
 response.EnsureSuccessStatusCode();
//post
response = await httpClient.PostAsync(
      “/*resourceUri*/”, new StringContent(data));
```

HttpWebRequest类

如果希望更好地控制HTTP请求，可以使用System.Net类库中的HttpWebRequest类。

该类与HttpClient类的使用方法很相似，但还是有一些区别。

下面列出HttpWebRequest类常用的方法和属性：

Create()	创建一个HttpWebRequest对象
GetResponseAsync()	异步返回响应的数据流
GetRequestStreamAsync()	异步获取用于向URI资源发送数据的Stream对象
Method	获取或设置请求的类型
ContentType	获取或设置HTTP标头的值

HttpWebRequest类

使用HttpWebRequest类与服务器进行通信：

1. 使用Create方法创建HttpWebRequest对象

2. 使用GetResponseAsync方法向指定的URI资源发出请求并接收响应数据

如果需要向URI资源发送数据，可以使用GetRequestStreamAsync方法。

WCF服务

WCF数据服务

- 与HTTP请求类似，WCF（Windows Communication Foundation）也是一种数据通信技术。

- 在WCF出现之前，当针对不同的通信协议开发分布式应用时，由于各个通信协议的底层设计方法不同，因此开发人员需要面对多种程序设计模型，为不同的通信协议都设计一套符合其标准的接口，增加了开发的复杂度。WCF的出现很好地解决了这一问题。

WCF服务

客户端和WCF服务器进行通信时需要借助代理，也就是添加服务引用，这样才能依据WCF契约实现数据的交换。

使用WCF通信服务需要宿主程序的支持。

WCF服务包含四种契约：

服务契约：定义了WCF提供的多个服务。
操作契约：定义了客户端的访问类型。
数据契约：定义了服务器与客户端沟通时的数据格式。
消息契约：定义了通信期间改写消息内容的规范。

创建WCF服务

使用WCF通信服务需要宿主程序的支持。

```
using System.Runtime.Serialization;
namespace HttpServer
{
        //定义数据契约
        [DataContract]
        public class DataClass
        {
                [DataMember]
                public string NewsTime{set; get;}
                [DataMember]
                public string NewsTitle{set; get;}
        }
}
```

[DataContract]：表示该类为可用的数据契约类型，使DataClass类能用于WCF接口的数据交换。
[DataMember]：表示契约类型中的属性。

创建WCF服务

```
namespace HttpServer
{
    public class NewsDataService : INewsDataService
    {
        public DataClass[] News
        {
            //新闻标题
            List<DataClass> NewsDataService = new List<DataClass>()
            {
                NewsDataService DataClass{ NewsTitle="新闻标题1", NewsTime="2013-08-
13"}
            };
            //返回新闻标题数组
            return newsData.ToArray();
        }
    }
}
```

客户端可以通过调用News方法获得新闻标题数据

创建WCF服务

```
using System.ServiceModel;
namespace HttpServer
{
    //定义服务契约
    [ServiceContract]
    public interface INewsDataService
    {
        //操作契约
        [OperationContract]
        DataClass[] News();
    }
}
```

在WCF服务中，因为允许客户端调用的方法需要声明[OperationContract]标签，所以在接口中添加类型为DataClass[]的News方法并为其添加特性标签[OperationContract]。

使用WCF服务

```
//获得对WCF服务的引用
WCFService.NewsDataServiceClient client = new
WCFService.NewsDataServiceClient();
//异步获取数据
ObservableCollection<WCFService.DataClass> data = await client.NewsAsync();
```

Socket通信

Socket通信

- Socket通信是一种点对点的通信技术，具有简单易用，连接稳定，数据传送能力强等特点。

Socket入门

HTTP是一种临时的、无状态的通信协议。临时性，指的是当应用程序发出HTTP请求时，服务端在做出一次回应后，HTTP就会处于断开状态，当应用程序开发人员需要获取服务端的最新数据时，只能每隔一段时间让应用程序发出一次HTTP请求。

与HTTP相比，Socket是以IP地址及端口号为连接对象的一个通信句柄，以端对端为通信模型的网络通信协议。Socket通信能够保持连接的持久性，Socket一旦与服务器成功连接，服务端的相应端口会处于开放状态。直到程序发出断开指令或者网络中断，Socket才会结束整个连接过程。

Socket通信能够支持两种工作模式：

TCP模式：要发起连接，服务端必须处于工作状态，通信是顺序进行的。

UDP模式：不依赖于目标是否处于工作状态，并且通信时无序进行的。

Socket连接

1. 服务器连接

示例：

```
using System.Net.Sockets

public StreamSocket socket;
//初始化
socket = new StreamSocket();
//与服务器连接
await socket.ConnectAsync(new HostName(HostName.Text), ServerPort.Text);
```

socket对象的ConnectAsync方法异步与服务器建立连接，参数是服务器的地址和端口号。

Socket连接

2．发送和接收数据

示例：

```
public DataWriter writer;
public DataReader reader;
//发送数据流
writer = new DataWriter(socket.OutputStream);
writer.WriteString(str);
//异步发送数据
await writer.StoreAsync();
//读取数据流
reader = new DataReader(socket.InputStream);
reader.InputStreamOptions = InputStreamOptions.Partial;
await reader.LoadAsync(1024);
string data = reader.ReadString(reader.UnconsumedBufferLength);
```

Socket连接

3．关闭通信

示例：

```
reader.Dispose();
writer.Dispose();
socket.Dispose();
```

总结

- HTTP请求：HttpClient类、HttpWebRequest类
- WCF服务：创建、使用WCF服务
- Socket通信： Socket入门、连接、发送接收数据、关闭连接

Q&A

Microsoft®