《C#程序设计》

第九章 文本、XML与网络通信编程

武汉大学计算机学院 贾向阳

目录

Regex类

- using System.Text.RegularExpressions;
- 静态方法:
  - Regex.IsMatch(s, pattern) //返回布尔值
- Regex对象:
  - regex.IsMatch(s) //返回布尔值
  - regex.Match(s) //返回第一个匹配（Match对象）
  - regex.Matches(s) //返回所有匹配
  - regex.Replace(s, r) //将匹配子串替换为另一个字符串

正则表达式（Regular Expression)

- 字符
  - 普通字符：ab \t \[\x20
  - 特殊字符
    - . 回车之外的任何字符
    - \s 空格 \S 非空格
    - \d 数字 \D 非数字

正则表达式（Regular Expression)

- 字符集
  - [] 字符集
    - 例： [Bbw]、 [a-zA-Z0-9]
  - [^]排斥字符集
    - 例：[^0-9] 非数字（相当于\D）

正则表达式（Regular Expression)

- 量词
  - {n} 重复n次
    - A{3} 表示AAA， [0-9] {3} 表示三个连续数字
  - {n,m} 最少重复n次，最多重复m次
  - * 零次或任意次 相当于{0,}
  - + 一次或任意次 相当于{1,}
  - ？ 零次或一次 相当于{0,1}

正则表达式（Regular Expression)

- 位置
  - ^ 从首字符开始匹配
    - 例： ^A 表示以A开头
  - $ 字符串结尾
    - 例：^X[0-9]+$ 表示X开头接着一到多个数字的字符串

正则表达式（Regular Expression)

- 分组
  - () 子表达式进行分组
    - （good|bad）boy 匹配“goodboy”或者“badboy”
    - |表示或，常在分组内使用
  - (?<名称>xxxxxxxx) 表示对分组进行命名
    - 命名后，可以在匹配和查找时使用 ${名称}
    - 若不命名,则为$1，$2等等　　而$0 表示整个匹配

目录

- 文本及正则表达式
- XML
- 网络通信编程

XML文档的基本结构

```
< ? xml version="1.0" ? >
<book category="novel"  ISBN="1-8630-014">
   <title> The Handmaid's Tale</title>
   <price value=”19.95” />
</book>
```

XML指令:表明这是一个XML文档

属性：附加在开始标签上对信息，如：category=”novel” 和 ISBN=“1-8630-014”

元素：XML文档树中的节点.有开始标签和结束标签包围, 如:
<book> ...</book>, <title>...</titile> <price .... />

XML的基本处理方式

- DOM (Document Object Model)
  - 读取全部文档，形成一个树，对树进行操作。
  - 相关类：XmlDocument
- SAX(Simple API for XML)
  - 逐行读取文档，依次处理各个元素
  - 相关类：XmlTextReader、XmlTextWriter

DOM模式

- 元素、文本、属性都解析为XmlNode
- 按照层次关系嵌套形成树形结构

- XmlNode对象的属性
  - NodeType：节点类型
  - Attributes：节点的属性集合
  - ChildNodes: 子节点集合
  - Name：名称
  - Value：值

- XmlNode对象的方法
  - 查询(见Xpath)
  - 增加
    - AppendChild,PrependChild,
    - InsertBefore, InsertAfter
  - 删改
    - RemoveChild, ReplaceChild, RemoveAll

使用XmlTextReader及Writer

- XmlTextReader
  - 对 XML 数据进行快速、非缓存、只读访问的读取器
    - while (reader.Read()) {
    - switch (reader.NodeType):
    - 使用reader.Name及.Value
- XmlTextWriter
  - . WriteStartElement
  - . WriteAttributeString
  - . WriteEndElement 等

XPath的概念

- XPath 是对XML进行查询的表达式

```
XmlElement root = doc.DocumentElement;
//查询所有book元素下的title元素
XmlNodeList nodes =  root.SelectNodes(“/book/title”);
//查询第一个book元素下的isbn属性
XmlNode node = root.SelectSingleNode( “/book[1]/@isbn”);
```

XPath的要素

- / 路径　　// 任意路径
- [n] 第n个子结点（从1开始）
- 属性　@
- 条件　[]
- 例如
  - /books/book/@title　（/books/book路径下的title属性）
  - //price　（文档中任意路径下的price元素）
  - /para[@type="warning"][5]

使用XSLT进行转换

```
< ? xml version="1.0" ? >
<book category="novel"  ISBN="1-8630-014">
  <title> The Handmaid's Tale</title>
  <price value=”19.95” />
</book>
<book category="novel"  ISBN="1-8630-014
  <title>Harry Potter</title>
  <price value=”39.90” />
</book>
```

XML

```
<html>
  …
  <ul>
    <li> The Handmaid's Tale</li>
    <li> Harry Potter</li>
  </ul>
  …
</html>
```

其他格式，如HTML

```
<html>
  …
  <ul>
    <xsl:for-each select="book">
      <li><xsl:value-of select="title" /></li>
    </xsl:for-each>
  </ul>
  ….
</html>
```

XSLT

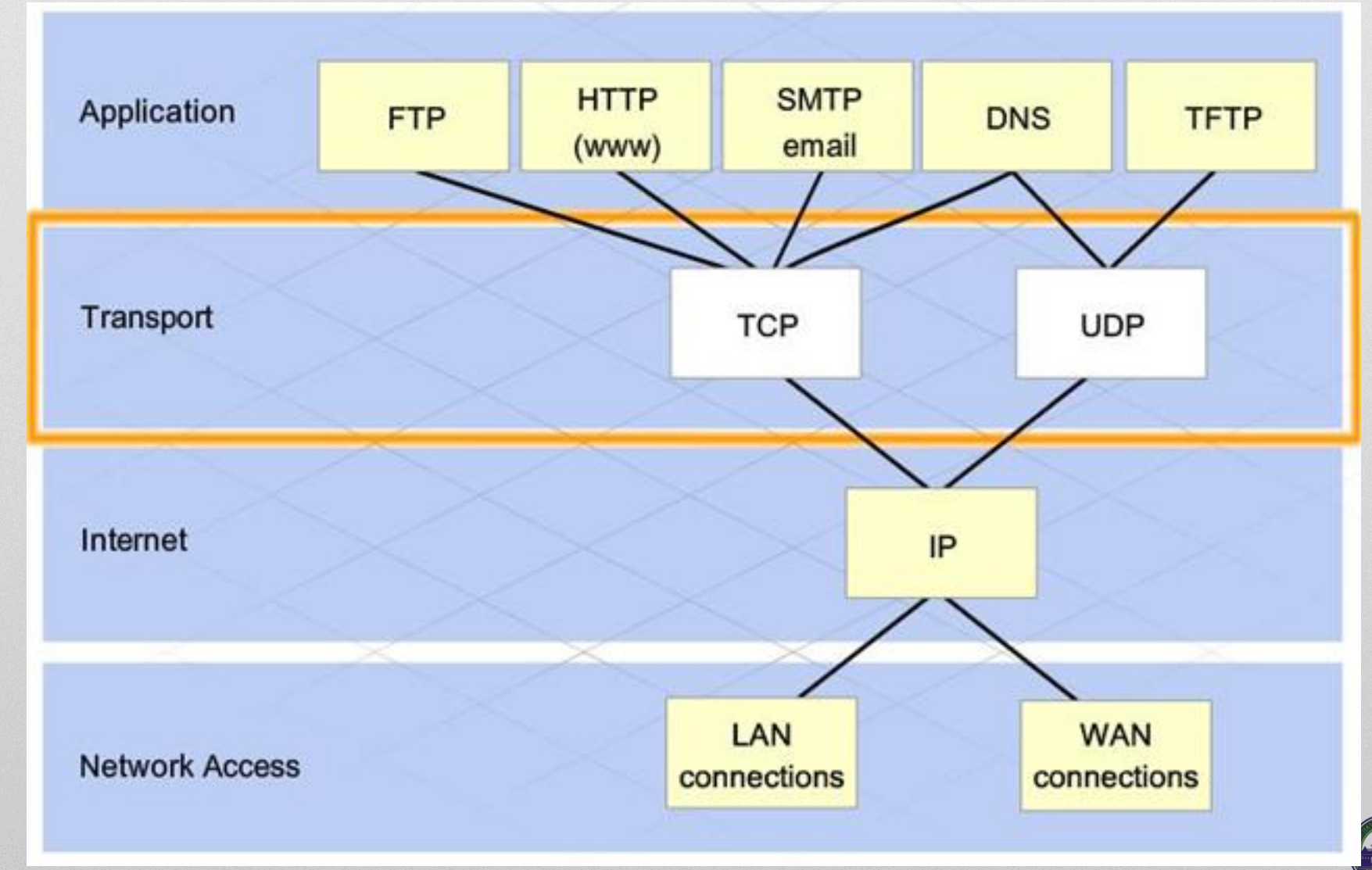
使用XSLT进行转换

- XmlDocument doc = new XmlDocument();
- doc.Load(@".\BookList.xml");
- XPathNavigator nav =doc.CreateNavigator();
- nav.MoveToRoot();
- XslCompiledTransform xt = new XslCompiledTransform();
- xt.Load(@".\BookList.xslt");
- XmlTextWriter writer = new XmlTextWriter(Console.Out);
- xt.Transform(nav,null, writer);

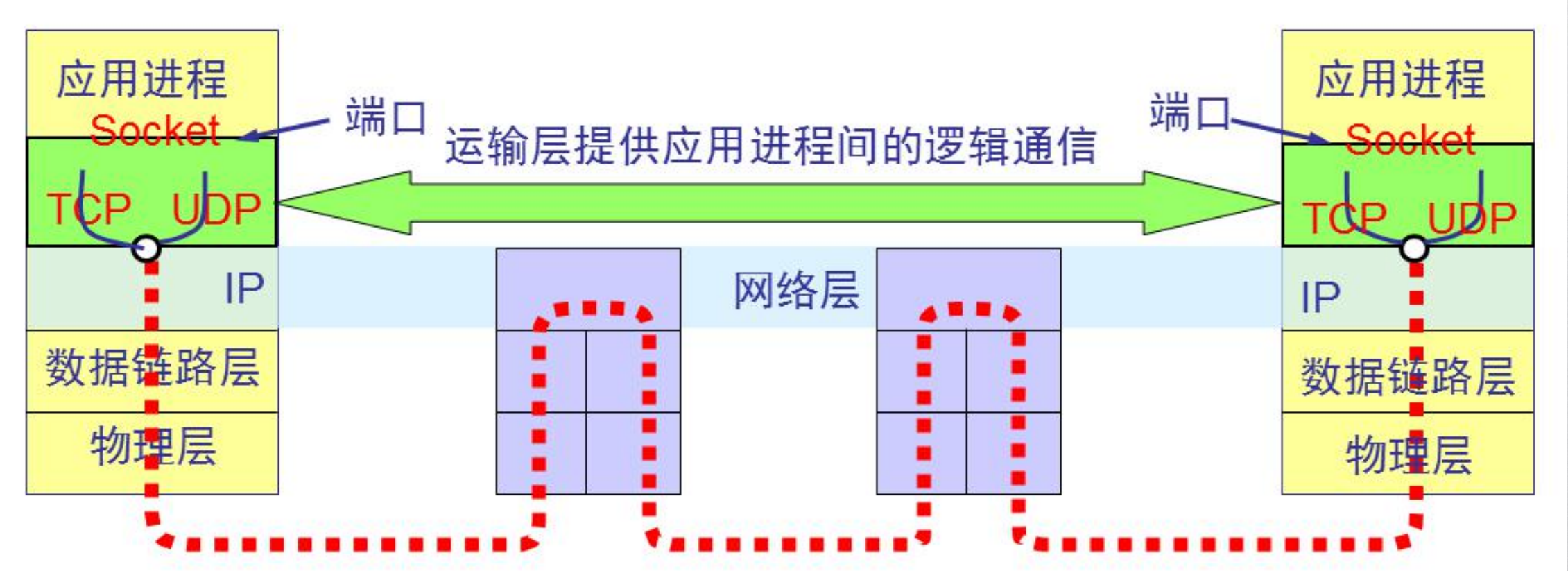
目录

- 文本及正则表达式
- XML
- 网络通信编程

1、网络协议栈

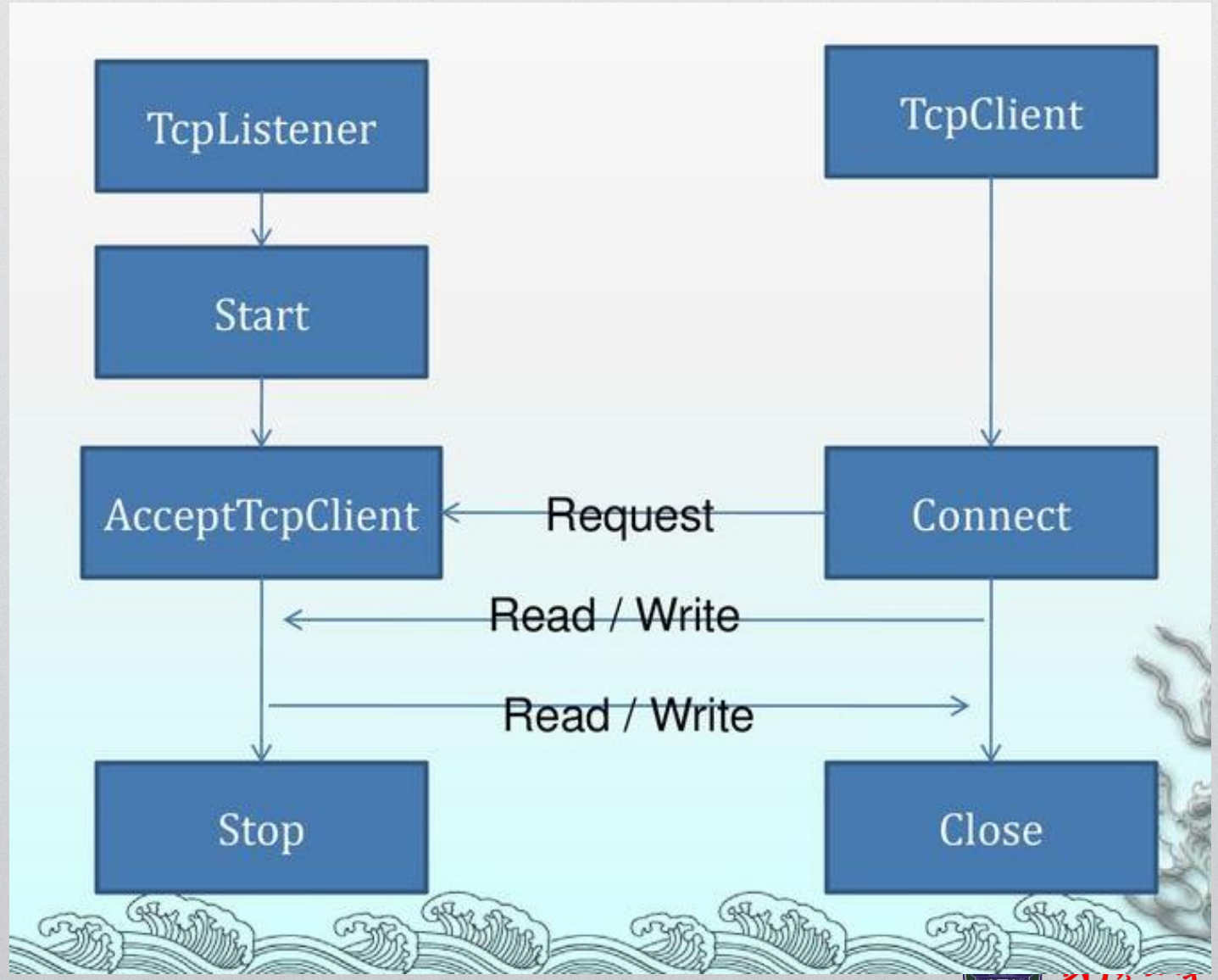


2、Socket编程



2、Socket编程

- 两个关键的类
  - TCPClient：TCP客户端类
  - TCPListener：TCP服务端类



TcpListener类的方法

方法	说明
AcceptSocket	从端口处接收一个连接并赋予它Socket对象
AcceptTcpClient	从端口处接收一个连接并赋予它TcpClient对象
Equals	判断两个TcpListener对象是否相等
GetHashCode	用作特定类型的哈希函数
GetType	得到当前实例类型
Pending	确定是否有挂起的连接请求
Start	开始侦听传入的连接请求
Stop	关闭侦听器
ToString	创建TcpListener对象的字符串表示

TcpClient类的方法

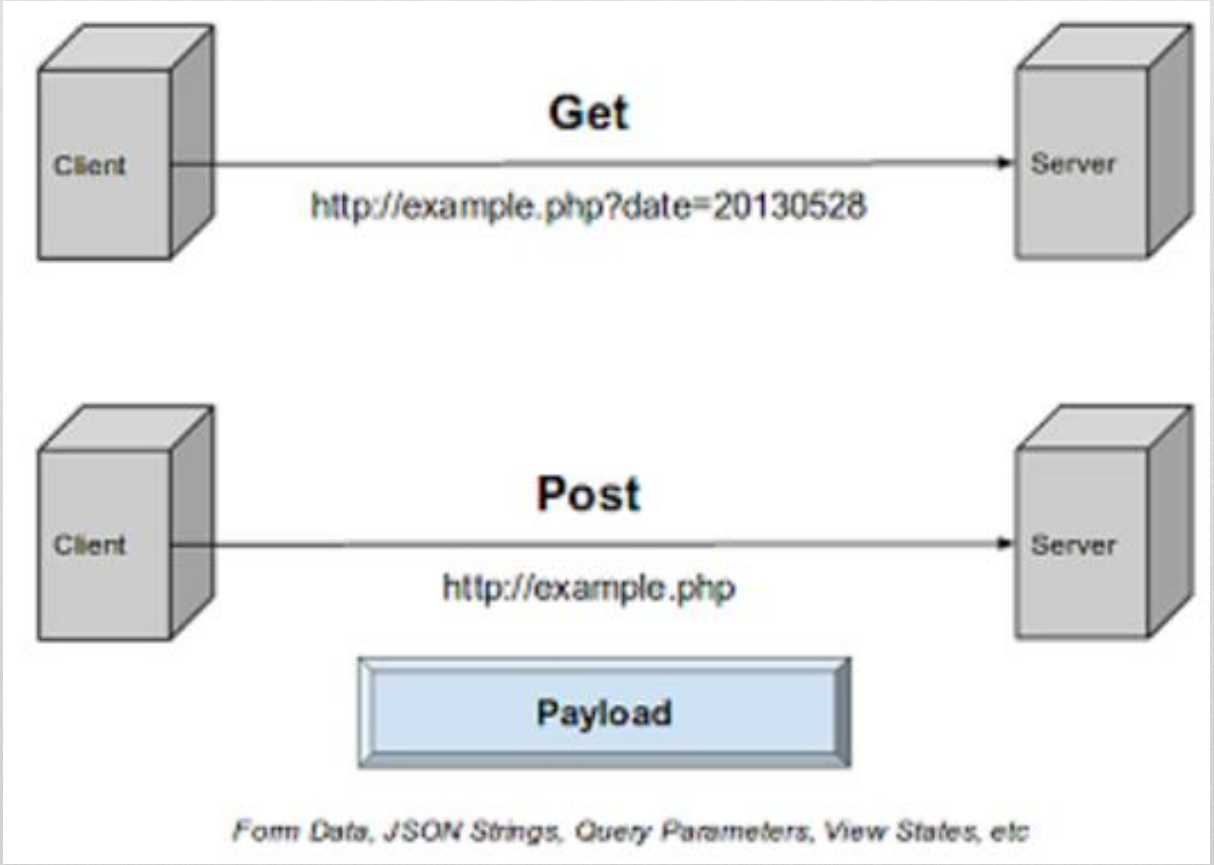
方法	含义
Close	释放TcpClient实例，而不关闭基础连接
Connect	用指定的主机名和端口号将客户端连接到TCP主机
BeginConnect	开始一个对远程主机连接的异步请求
EndConnect	异步接受传入的连接尝试
GetStream	获取能够发送和接收数据的NetworkStream对象

3、Web信息获取

- 协议：HTTP协议（超文本传输协议）
- 消息模式：请求/响应模式
- 资源地址：URL/URI

GET和POST方法

- GET方法
  - 没有HTTP消息体
  - 请求参数一般附加在URL上
- POST方法
  - 有HTTP的消息体
  - 请求数据一般放在消息体内，也可以放在URL上



使用System.Web

- System.Web提供支持浏览器/服务器通讯的类和接口。
- 此命名空间包括提供有关当前 HTTP 请求的大量信息的 Request 类、管理 HTTP 到客户端的输出的 Response 类，以及提供对服务器端实用工具和进程的访问的 HttpServerUtility 对象。

- System.Web 还包括用于 Cookie 操作、文件传输、异 常信息和输出缓存控制的类。

System.Net中的类

类	说　明
Cookie	提供对cookie(一种网络服务器传递给浏览器的信息)进行管理的一套方法和属性
Dns	提供简单的域名协议功能
EndPoint	表示网络地址的抽象类
FileWebRequest	与‘file://’开头的URI地址进行交互，以访问本地文件
FileWebResponse	通过‘file://’　URI地址提供对文件系统的只读访问
HttpWebRequest	授权客户向HTTP服务器发送请求
HttpWebResponse	授权客户接收HTTP服务器的回答信息
IPAddress	表示一个IP地址
IPEndPoint	表示一个IP终端(1P地址加端口号)
IPHostEntry	与带有一组别名和匹配IP地址的DNS登录建立连接
WebClient	提供向URL传送数据和从URI接收数据的通用方法
WebException	使用网络访问时产生的异常

WebClient类

主要方法：

- DownloadString() ：下载为字符串
- DownloadData() ：下载为Byte数组
- DownloadFile() ：下载到本地文件
- UploadData 及 UploadFile ：上传
- OpenRead 及 OpenWrite ：以Stream方式进行读写

示例：获取百度主页

```
string url = @"http://www.baidu.com";
WebClient client = new WebClient();
byte[] pageData = client.DownloadData(url);
string pageHtml = Encoding.Default.GetString(pageData);
Console.WriteLine(pageHtml);
```

4、E-Mail编程

- 邮件协议
  - 接收邮件：POP3协议、IMAP协议
  - 发送邮件：SMTP协议

163免费邮箱的服务器信息

服务器名称	服务器地址	SSL协议端口号	非SSL协议端口号
IMAP	imap.163.com	993	143
SMTP	smtp.163.com	465/994	25
POP3	pop.163.com	995	110

4、E-Mail编程

- 使用MailMessage类和SmtpClient发送Email

```
MailMessage message = new MailMessage();
SmtpClient client = new SmtpClient(txt_smtpserver.Text);
message.From = new MailAddress(txt_from.Text);
message.To.Add(txt_to.Text);
message.Subject = txt_subject.Text;
message.Body = richTextBox1.Text;
client.Port = 25;
client.Credentials = new System.Net.NetworkCredential
                        (txt_from.Text, txt_password.Text);
//SmtpServer.EnableSsl = true;
client.Send(message);
```

4、E-Mail编程

- 使用MailMessage类和SmtpClient发送Email

```
MailMessage message = new MailMessage();
SmtpClient client = new SmtpClient(txt_smtpserver.Text);
message.From = new MailAddress(txt_from.Text);
message.To.Add(txt_to.Text);
message.Subject = txt_subject.Text;
message.Body = richTextBox1.Text;
client.Port = 25;
client.Credentials = new System.Net.NetworkCredential
                        (txt_from.Text, txt_password.Text);
//SmtpServer.EnableSsl = true;
client.Send(message);
```

4、Web应用编程简介

Web应用

- 服务端
  - Web页面和资源
    - 包括Web网页（html）、JavaScript代码、图形、图像。
  - Web服务/Web API
    - 响应用户请求，提供XML格式/JSON格式的响应消息。是一种网络可以访问的API。
- 客户端
  - 执行环境：浏览器
  - 浏览器下载Web页面和JavaScript代码到本机，然后展示页面，运行JavaScript代码。
  - JavaScript代码可以直接与Web API 进行交互。