# 1.网络编程

## 1.1网络编程概述

计算机网络：概述

网络编程的目的：传播交流信息，数据的交换！

JDK提供的API：InetAdress和Socket

想要达到这个效果需要什么：

1. 如何准确定位网络上的一台主机：IP地址，端口，定位到该计算机上的某个资源。
2. 找到了这个主机，如何传输数据?

javaweb 网页编程B/S架构

网络编程 C/S架构

## 1.2网络通信的要素协议模型

通信的必要条件，通信双方地址：

1. ip地址
2. 端口号

规则：网络通信协议

http ,TCP, UDP, smtp等等

TCP/IP参考模型

如图：OSI七层网络模型是理想化的模型，并未实际应用，真正应用广泛的是TCP/IP四层模型



总结：

网络编程的两个主要问题：

如何定位网络上的一台主机：ping 域名

找到主机以后如何通信

网络编程的要素

ip地址和端口号

网络通信协议：TCP和UDP

针对于JAVA,万物皆对象

## 1.3IP

ip地址：jdk提供的API，InetAdress

唯一定位一台网络上的主机

127.0.0.1代表本机，localhost本机

IP地址的分类：

1. IPV4/IPV6

IPV4：4个字节组成

IPV6：16个字节组成

1. 公网(互联网)和私网(局域网)

* 192.168.\*.\*:专门给局域网使用的地址
* A/B/C/D类网址：

域名：解决记忆IP问题

例如：[www.abc.com](http://www.abc.com)

## 1.4端口

端口代表计算机一个程序的进程

* 不同的进程有不同的端口号,用来区分软件
* 端口号被规定为0-65535，不同协议下均如此
* TCP，UDP 65535\*2,，单个协议下端口号不能冲突。不通协议下端口号可以重复
* 端口分类
  + 共有端口：0-1023
    - HTTP:80
    - HTTPS:443
    - FTP:21
    - Telent:23
  + 程序注册端口：1024-49151
    - 用于分配给用户或者程序
    - Tomcat:8080
    - Mysql:3306
    - Oracle:1521
  + 动态或者私有端口：49152-65535
    - 尽量少使用
    - DOS命令:列举部分
      * netstat
      * ping
      * tasklist|findstr 端口号：查看指定端口的进程

## 1.5通信协议

协议:约定

网络通信协议：速率，传输码率，代码结构，传输控制…

大事化小：分层模型

TCP/IP协议族：实际上是一组协议。比较重要的协议，TCP和UDP

* TCP:传输控制协议
* UDP:用户数据报协议

TCP和UDP对比：

* TCP:类似于打电话
  + 连接稳定
  + 三次握手，四次挥手，以A和B举例
    - A：连接
    - B：连接
    - A：连接

三次握手完成，可以连接。三次握手没有完成，无法连接

四次挥手举例：

* A：断开连接
* B：收到，等我一下
* B：结束，确定断开连接
* A：收到，断开连接

四次挥手完成，确定断开。

* + 传输完成，释放连接，效率低
* UDP:类似于发短信
  + 不需要连接不稳定
  + 客户端服务端没有明确的界限
  + 只管发送，不管结果
  + DDOS：饱和攻击

## 1.6TCP

客户端：

* 连接服务器Socket
* 发送消息

服务端

* 建立服务的端口ServerSocket
* 等待用户连接accept
* 接受消息

客户端和服务端连接建立过程。

客户端创建socket套接字连接目标IP和端口

服务端创建ServerSocket套接字指定服务端口，服务端本机IP就是客户端的连接

服务端通过accept方法侦听需要连接的客户端

服务端获取客户端socket的输入输出流。

### 文件上传

com/study/sdm/net/TcpTestDemo2 客户端

package com.study.sdm.net;  
  
import java.io.\*;  
import java.net.InetAddress;  
import java.net.Socket;  
import java.net.UnknownHostException;  
  
public class TcpTestDemo2 {  
 public static void main(String[] args) throws Exception {  
 Socket socket = new Socket(InetAddress.*getByName*("127.0.0.1"),9000);  
 OutputStream os = socket.getOutputStream();  
 FileInputStream fis = new FileInputStream(new File("newvisual\_02.jpg"));  
 byte [] buffer = new byte[1024];  
 int len;  
 while((len=fis.read(buffer))!=-1){  
 os.write(buffer,0,len);  
 }  
 socket.shutdownOutput();  
 InputStream inputStream = socket.getInputStream();  
 ByteArrayOutputStream baos = new ByteArrayOutputStream();  
 byte[] bytes = new byte[1024];  
 int c;  
 while ((c = inputStream.read(bytes))!=-1){  
 baos.write(bytes,0,c);  
 }  
 System.*out*.println(baos.toString());  
 baos.close();  
 inputStream.close();  
 fis.close();  
 os.close();  
 socket.close();  
 }  
}

com/study/sdm/net/TcpTestDemon2 服务端

package com.study.sdm.net;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class TcpTestDemon2 {  
 public static void main(String[] args) throws Exception {  
 ServerSocket serverSocket = new ServerSocket(9000);//服务端创建  
 Socket socket = serverSocket.accept();//监听客户端连接  
 InputStream is = socket.getInputStream();  
 FileOutputStream fos = new FileOutputStream(new File("receive.jpg"));  
 byte [] buffer = new byte[1024];  
 int len;  
 while((len=is.read(buffer))!=-1){  
 fos.write(buffer,0,len);  
 }  
 OutputStream os = socket.getOutputStream();  
 os.write("接受完毕".getBytes());  
 fos.close();  
 is.close();  
 socket.close();  
 serverSocket.close();  
 }  
}

注意：服务端在接受客户端传输过来的数据时，客户端需要给定标识来表示已经传送完毕，比如：socket.shutdownOutput();//客户端关闭输出流，这时服务端就知道客户端传输完毕，也可以通过客户端的输出流输出“\n”符来表示输出完。

### Tomcat

服务端：

* 自定义Server
* Tomcat服务器：javaweb开发

客户端：

* 自定义Client
* 浏览器Browser

## 1.7UDP

不需要连接，用户只需要发送就可以

JDK 提供的API：DatagramSocket 和DatagramPacket

API简介：

DatagrameSocket用于构建发送和接受数据报的套接字。

DatagramPacket用于构建发送到指定主机(IP参数)指定端口(Port参数)的数据包(字节数组参数和偏移量和长度参数)；

## 1.8URL

URL: uniform Resourfe Locator,同一资源定位符

JDK提供的API: URL，HttpURLConnection,HttpsURLConnection

规范写法：

协议：//ip地址：端口/项目名称/资源

建立连接代码如下：

URL url = new URL("https://game.doaxvv.com/en/index.html");  
HttpsURLConnection connection = (HttpsURLConnection)url.openConnection();