网络编程

网络编程在开发之中使用很少，应该说在之前的桌面开发时代会用到，现在已经逐步淡出历史舞台，之所以讲解，是因为有很多概念需要掌握（J2EE和Android都会用到）

基本概念

1. 计算机网络

   使用光纤、网线等连接设备将处于不同位置的计算机 连接起来组成的网络。网络最大的优势在于可以共享设备和数据

2. **IP**地址

   为计网络中的每一台计算机分配一个唯一的地址，来唯一标识网络中的计算机，这个地址就是IP地址

3. 域名

   由于IP地址不方便记忆，所以为了方便记忆，又引入了另外一个概念——域名(Domain Name)，例如baidu.com等。一个IP地址可以对应多个域名，一个域名只能对应一个IP地址

4. **DNS**服务器

   在网络中传输的数据，全部是以IP地址作为地址标识，所以在实际传输数据以前需要将域名转换为IP地址，实现这种功能的服务器称之为DNS服务器，也就是通俗的说法叫做域名解析

5. 端口

   IP地址和域名很好的解决了在网络中找到一个计算机的问题，但是为了让一个计算机可以同时运行多个网络程序，就引入了另外一个概念——端口(port)。在同一个计算机中每个程序对应唯一的端口，这样一个计算机上就可以通过端口区分发送给每个端口的数据了，换句话说，也就是一个计算机上可以并发运行多个网络程序，而不会在互相之间产生干扰。

6. 请求和响应

7. 客户端和服务端

   在网络通讯中，第一次主动发起通讯的程序被称作客户端(Client)程序，简称客户端，而在第一次通讯中等待连接的程序被称作服务器端(Server)程序，简称服务器。一旦通讯建立，则客户端和服务器端完全一样，没有本质的区别。

8. **P2P(Point to Point)**结构

   P2P程序是一种特殊的程序，一个P2P程序中既包含客户端程序，也包含服务器端程序。P2P既可以为其他安装P2P程序的计算机提供资源，又可以从其他安装P2P程序的计算机获取共享的数据。

C/S结构和B/S结构

1. **C/S**结构：网络编程中的两种程序分别是客户端和服务器端，例如QQ程序，每个QQ用户安装的都是QQ客户端程序，而QQ服务器端程序则运行在腾讯公司的机房中，为大量的QQ用户提供服务。这种网络编程的结构被称作客户端/服务器结构，也叫做Client/Server结构，简称C/S结构。
2. **B/S**结构：在运行很多程序时，没有必要使用专用的客户端，而需要使用通用的客户端，例如浏览器，使用浏览器作为客户端的结构被称作浏览器/服务器结构，也叫做Browser/Server结构，简称为B/S结构。

	C/S结构	**B/S**结构
开发	单独开发客户端和服务端，工作量大	只需要开发服务器端
维护	需要单独对客户端和服务器进行维护，成本高	只需要维护服务端
用户体验	客户端表现力丰富	表现力相较于C/S差距很大
应用范围	安装客户端才可以使用	面向整个万维网
对计算机的要求	有一定的配置要求	浏览器，联网
安全性	安全性相对较高	安全性相对较低

协议

网络编程就是运行在不同计算机中两个程序之间的数据交换。在实际进行数据交换时，为了让接收端理解该数据，那么就需要规定该数据的格式，这个数据的格式就是协议。

常见的协议

- IP：Internet Protocol（网络之间互联的协议）的缩写，也就是为计算机在网络中相互连接进行通信而设计的协议
- TCP：Transmission Control Protocol 传输控制协议，TCP是一种面向连接的、可靠的、基于字节流的传输层通信协议
- UDP：User Datagram Protocol （用户数据包协议）的简称，一种无连接的传输层协议，提供面向事务的简单不可靠的信息的传送服务
- HTTP：Hyper Text Transport Protocol（超文本传输协议），一种详细规定了浏览器和万维网服务器之间相互通信的规则，通过因特网传送万维网文档的数据传送协议
- DNS：Domain name System 或 Domain name Service （计算机域名系统），由解析器和域名服务器组成。
- FTP：File Transfer Protocol，是TCP/IP网络上两台计算机传送文件的协议

TCP和UDP

	TCP	UDP
建立虚拟连接	需要	不需要
释放确认连接	是	否
发送失败	重写发送	不重新发送
适合传输的数据	重要数据	非核心数据
性能开销	因需要专门建立虚拟连接，所以速度稍慢，且在传输时产生的数据量比UDP大	不需要建立连接，且没有超时重发机制，故传输速度快

TCP编程

服务端

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

/**
 * 服务器
 * @author Kevin
 */
public class MyServer {

    public static void main(String[] args) {
        try {
            // 在9999端口启动服务器
            ServerSocket server = new ServerSocket(9999);
            System.out.println("服务器启动成功，等待客户端连接。。。");
            // 当有客户端连接的时候，此方法会返回客户端对象
            Socket client = server.accept();
            // 获取客户端主机地址
            String address = client.getInetAddress().getHostAddress();
            System.out.println(address + "上线了！！！");
            // 获取字节输入流并转换为字符输入流
            BufferedReader br = new BufferedReader(new
InputStreamReader(client.getInputStream()));
            // 获取字节输出流并转换为字符输出流
            BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(client.getOutputStream()));
            Scanner scanner = new Scanner(System.in);
            while (true) {
                String str = br.readLine();
                if (str.equalsIgnoreCase("bye")) {// 如果客户端发送'bye'，则退出
                    break;
                } else {
                    System.out.println("From Client:" + str);
                    System.out.print("To Client:");
                    String toClient = scanner.nextLine();
                    bw.write(toClient);// 想客户端发送数据
                    bw.newLine();// 插入换行符
                    bw.flush();// 开始发送
                }
            }
            // 释放资源
            scanner.close();
            bw.close();
            br.close();
            client.close();
            server.close();
        } catch (IOException e) {
            e.printStackTrace();
```

```
        }
    }

}
```

客户端

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

/**
 * 客户端
 * @author Kevin
 */
public class MyClient {

    public static void main(String[] args) {
        try {
            // 连接服务器
            Socket client = new Socket("172.0.3.243", 9999);
            // 获取字节输出流并转换为字符输出流
            BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(client.getOutputStream()));
            // 获取字节输入流并转换为字符输入流
            BufferedReader br = new BufferedReader(new
InputStreamReader(client.getInputStream()));
            Scanner scanner = new Scanner(System.in);
            while (true) {
                System.out.print("To Server:");
                String toServer = scanner.nextLine();
                bw.write(toServer);
                bw.newLine();// 必须加
                bw.flush();// 必须加
                if (toServer.equalsIgnoreCase("bye")) {
                    break;
                } else {
                    String fromServer = br.readLine();
                    System.out.println("From Server:" + fromServer);
                }
            }
            // 释放资源
            scanner.close();
            br.close();
            bw.close();
            client.close();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```