**网络编程模型**

课程目标

* 了解网络相关概念
* 理解 TCP协议
* 使用TCP协议收发数据 掌握

**基于scoket**

1. 网络
2. 什么是网络

将独立的计算连接起来构成一个计算机网络.

1. 网络好处

连接多台计算机、共享资源、数据交互

1. IP地址(Internet protocal):唯 一标识网络上的每一台计算机

组成:32位，由4个8位进制数组成.

1-255.1-255.1-255.1-255

Ip地址=网络地址+主机地址.

网络地址:标识计算机或网络设备

主机地址:标识特定主机或网络设备

1. 常用的网络命令

4.1 查看ip地址:

ipconfig /all

* 1. 查看网络是否通畅

ping 目标地址

例如:ping [www.baidu.com](http://www.baidu.com) 、ping ip地址

1. 网络协议

**5.1网络通信协议：为在网络中不同的计算机之间进行通信而建立的规则、标准或约定的集合**

* + **TCP协议（Transmission Control Protocol 传输控制协议）**

**面向连接的、可靠的、基于字节流的传输通信协议**

* + **UDP协议（User Datagram Protocol 用户数据报协议）**

**无连接的协议，在传输数据之前，客户端和服务器并不建立和维护连接**

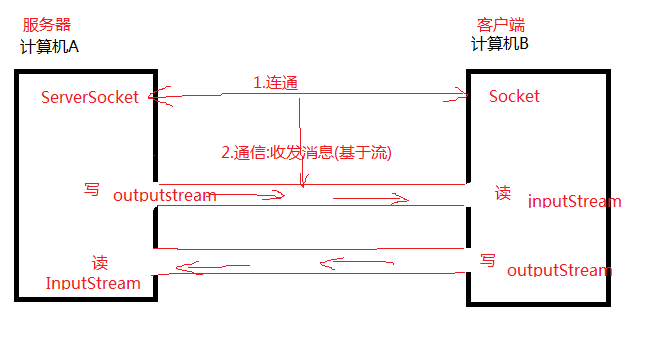
1. socket编程(实现网络通信)
2. 什么是socket:

**通信链路的端点就被称为“套接字”（英文名Socket）**

**注意:socket是提供给程序操作的接口.**

1. socket编程的工作原理

注意:建立在TCP协议基础之上



示例：服务端代码server.java

|  |
| --- |
| **public static void** main(String[] args) **throws** InterruptedException {  **try** {  *//1.创建ServerSocket对象* ServerSocket serverSocket=**new** ServerSocket(6699); *//指定端口  //2.等待客户端来连* System.***out***.println(**"客户端快来连我。。。我等你"**);  Socket socket=serverSocket.accept();  System.***out***.println(**"接通啦...."**);   *//3.获取流对象进行收发信息  //3.1获取流对象* InputStream inputStream=socket.getInputStream();  OutputStream os=socket.getOutputStream();  *//3.2接收消息* **int** len=0; *//确定流中有没有数据* **while**(len==0){ *//死循环* len=inputStream.available();  }  **byte** [] bs=**new byte**[inputStream.available()];  inputStream.read(bs); *//读* String str=**new** String(bs); *//将字节数组转化为字符串* System.***out***.println(**"服务器收到客户端的消息:"**+str);   *//3.3发送消息* String msg=**"恭喜你！登入成功"**;  os.write(msg.getBytes()); *//发消息    //关闭资源* os.flush();  os.close();  inputStream.close();  socket.close();  serverSocket.close();  } **catch** (IOException e) {  e.printStackTrace();  } } |

示例:客户端代码:client.java

|  |
| --- |
| **public class** Client {  **public static void** main(String[] args) {  **try** {  *//1.创建socket对象  //Socket socket=new Socket("服务器ip地址",服务器端口);  //本机:localhost、127.0.0.1* Socket socket=**new** Socket(**"localhost"**,6699);  System.***out***.println(**"我去连接服务器啦,已接通"**);   *//2.利用socket对象收发消息  //2.1基于socket获取流对象* InputStream is=socket.getInputStream(); *//输入流 收信息* OutputStream os=socket.getOutputStream(); *//输出流 写信息   //2.2写(发)信息* String str=**"用户名:wangjianbing,密码:123"**;  os.write(str.getBytes());  System.***out***.println(**"发送成功..."**);   *//2.3收信息* **int** len=0; *//确定流中有没有数据* **while**(len==0){ *//死循环* len=is.available();  }  **byte** [] bs=**new byte**[is.available()];  is.read(bs); *//读* String msg=**new** String(bs); *//将字节数组转化为字符串* System.***out***.println(**"客户端收到服务器的消息:"**+msg);   *//关闭资源* os.flush();  os.close();  is.close();  socket.close();  } **catch** (IOException e) {  e.printStackTrace();  }  } } |

扩展:使服务器实现多端支持(利用多线程):修改服务器类

|  |
| --- |
| **public static void** main(String[] args) {  *//服务端* **try** {  *//1.创建ServerSocker对象 并开放端口* ServerSocket serverSocket=**new** ServerSocket(7788);  System.***out***.println(**"我是服务器端...等着客户端来连"**);  *//2.等待接收客户端连接* **while**(**true**) {  Socket socket = serverSocket.accept();  *//开启线程 :创建线程支持多端连接* Task task=**new** Task(socket);  Thread t1=**new** Thread(task);  t1.start();  }   } **catch** (IOException e) {  e.printStackTrace();  } }  **public class** Task **implements** Runnable{  **private** Socket **socket**;   **public** Task(){}  **public** Task(Socket socket) {  **this**.**socket** = socket;  }  @Override  **public void** run() {  **try** {  *//3.利用socket接收消息  //3.1 获取输入流* InputStream is = **socket**.getInputStream();  *//3.2读取信息  //循环保存，流中没有没有数据* **int** len = 0;  **while** (len == 0) {  len = is.available();  }  **byte** bs[] = **new byte**[len];  is.read(bs);  *//将字节数组转化为字符串* String str = **new** String(bs);  System.***out***.println(**"收到客户端发送的消息:"** + str);   *//4.获取输出流发送消息* OutputStream os = **socket**.getOutputStream();  String msg = **"恭喜你，登入成功"**;  os.write(msg.getBytes());  System.***out***.println(**"发送消息成功"**);  *//3.3关闭流* os.close();  is.close();  }**catch** (Exception e){  e.printStackTrace();  }  } } |