**Socket**

1. 网络的基础知识
2. 两台计算机通信的基础

各自拥有IP地址（每台计算机的唯一标识）

通信协议（共同的语言）

相应的端口号区分不同的应用程序之间的通信

1. Tcp/IP协议

目前世界上应用最为广泛的协议，是以tcp和IP为基础的不同层次上的多个协议的集合

也称：TCP/IP协议族或TCP/IP协议栈

TCP:传输控制协议

IP:互联网协议

TCP/IP模型：

应用层（http:超文件传输协议 ftp:文件传输协议 smtp:简单邮件传送协议 telnet:远程登陆协议）

传输层（tcp/ip协议）

网络层

数据链路层

物理层

1. 端口号

用于区分不同应用程序

端口号的范围：0~65535，其中0~1023为系统所保留

IP地址和端口号组成了所谓的Socket,Socket是网络上运行的程序之间双向通信的链路终结点，是TCP和UDP的基础

常见端口号：

HTTP:80

ftp:21

telent:23

1. Java中的网络支持

针对网络通信的不同层次，java提供的网络功能有四大类：

InetAddress : 用于标识网络上的硬件资源

<URL:统一资源定位符> 通过URL可以直接读取或写入网络上的数据

Sockets:使用TCP协议实现网络通信的Socket相关的类

Datagrame:使用UDP协议，将数据保存在数据报中，通过网路进行通信

1. InetAddress类：用于标识网络上的硬件资源，表示互联网协议（IP）地址

Eg:

**public** **class** A {

**public** **static** **void** main(String[] args) **throws** UnknownHostException {

// **TODO** Auto-generated method stub

InetAddress address=InetAddress.*getLocalHost*();

System.***out***.println("计算机名："+address.getHostName());

System.***out***.println("IP地址："+address.getHostAddress());

**byte** [] b=address.getAddress();//获取字节数组形式的IP地址

System.***out***.println("字节数组形式的IP:"+Arrays.*toString*(b));

System.***out***.println(address);//直接输出InetAddress对象

System.***out***.println();

//根据机器名获取InetAddress实例

InetAddress address1=InetAddress.*getByName*("USER-20160608LK");

System.***out***.println("计算机名："+address1.getHostName());

System.***out***.println("IP地址："+address1.getHostAddress());

//根据机器IP地址获取InetAddress实例

System.***out***.println();

InetAddress address2=InetAddress.*getByName*("119.98.115.172");

System.***out***.println("计算机名："+address2.getHostName());

System.***out***.println("IP地址："+address2.getHostAddress());

}

}

1. URL

统一资源定位符，表示internet上某一资源的地址

两部分组成：协议名称和资源名称，中间用冒号隔开

在java.net中，提供了URL类来表示URL

URL常用方法：  
Eg:

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

//创建一个URL实例

**try** {

URL u = **new** URL ("http://www.jinyubao.com");

//根据已经存在的URL创建新的URL对象

URL u1=**new** URL(u,"/baobei.html?name=jinyubao#text");

//?后面表示参数，#后面表示参数

System.***out***.println("协议："+u1.getProtocol());

System.***out***.println("主机："+u1.getHost());

System.***out***.println("端口："+u1.getPort());

//如果未指定端口号，则使用默认的端口号，此时getport()返回值为-1

System.***out***.println("文件路径："+u1.getPath());

System.***out***.println("文件名："+u1.getFile());

System.***out***.println("相对路径："+u1.getRef());

System.***out***.println("查询字符串："+u1.getQuery());

} **catch** (MalformedURLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

URL读取网页内容

通过URL对象的openStream()方法可以得到指定资源的输入流

通过输入流可以读取，访问网络上的数据

Eg:

**public** **class** A {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**try** {

URL u=**new** URL("http://www.jikexueyuan.com");

//通过URL的openStream()方法获取URL对象所表示的资源的字节输入流

**try** {

InputStream is=u.openStream();

//将字节输入流转化为字符输入流

InputStreamReader in=**new** InputStreamReader(is,"utf-8");

//为字符输入流添加缓冲

BufferedReader b= **new** BufferedReader(in);

String data=b.readLine();

**while**(data!=**null**)//循环读取数据

{

System.***out***.println(data);//输出数据

data=b.readLine();

}

is.close();

b.close();

in.close();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

} **catch** (MalformedURLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

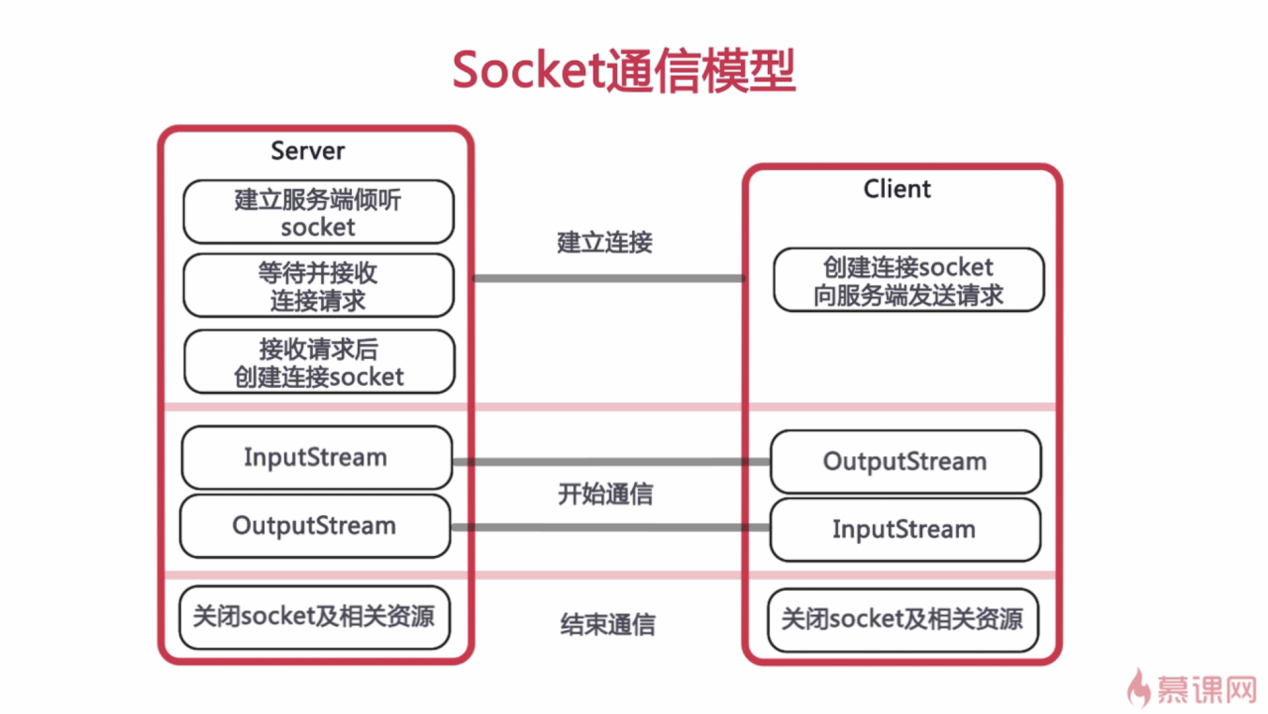
1. Socket通信

TCP协议是面向连接的，可靠的，有序的，以字节流的方式发送数据

基于TCP协议实现网络通信的类

客户端的Socket类

服务器端的ServerSocket类



Socket通信实现步骤：  
1.创建ServeSocket和Socket

2.打开连接到Socket的输入输出流

3.按照协议对Socket进行读写操作

4.关闭输入输出流，关闭Socket

1. 实现基于TCP的Socket通信

服务器端：

创建ServeSocket对象，绑定监听端口

创建accept()方法监听客户端的请求

建立连接后，通过输入流读取客户端发送的请求信息

通过输出流向客户端发送响应消息

关闭相关资源

客户端：

创建Socket对象，指明需要连接的服务器的地址和端口号

连接建立后，通过输出流向服务器端发送请求消息

通过输入流获取服务器响应的消息

关闭相关资源

服务器响应客户端