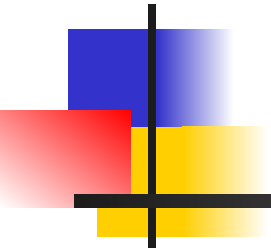

Java程序设计

第14章　网络程序设计

第14章 网络程序设计

14.1 基础知识

14.2 IP地址封装

14.3 套接字

14.4 数据报

14.1 基础知识

要开发网络应用程序，就必须对网络的基础知识有一定的了解，Java的网络通信可以使用TCP、IP、UDP等协议，在学习Java网络程序设计之前，先简单了解一下有关协议的基础知识。

14.1.1 TCP

TCP的全称是Transmission Control Protocol，也就是传输控制协议，主要负责数据的分组和重组。它与IP协议组合使用，称为TCP/IP。

TCP适合于对可靠性比较高的运行环境，因为TCP是严格的、安全的。它以固定连接为基础，提供计算机之间可靠的数据传输，计算机之间可以凭借连接交换数据，并且传送的数据能够正确抵达目标，传送到目标后的数据仍然保持数据送出时的顺序。

14.1.2 UDP

UDP的全称是User Datagram Protocol，也就是用户数据报协议，和TCP不同，UDP是一种非持续连接的通信协议，它不保证数据能够正确抵达目标。

虽然UDP可能会因网络连接等各种原因，无法保证数据的安全传送，而且多个数据包抵达目标的顺序可能和发送时的顺序不同，但是它比TCP更轻量一些，TCP的认证会耗费额外的资源，可能导致传输速度的下降。在正常的网络环境中，数据都可以安全的抵达目标计算机中，所以使用UDP会更加适合一些对可靠性要求不高的环境，例如在线影视、聊天室等。

14.2 IP地址封装

IP地址是每个计算机在网络中的唯一标识，它是32位或128位的无符号数字，使用4组数字表示一个固定的编号，例如“192.168.128.255”就是局域网络中的编号。

IP 地址，它是一种低级协议，UDP和TCP都是在它的基础上构建的。

Java提供了IP地址的封装类InetAddress。它封装了IP地址，并提供了相关的常用方法，例如解析IP地址的主机名称、获取本机IP地址的封装、测试IP地址是否可达等

IP地址封装

InetAddress类的常用方法如表14-1所示。

方法名称	方法说明	返回类型
getLocalHost()	返回本地主机的InetAddress对象	InetAddress
getByName(String host)	获取指定主机名称的IP地址	InetAddress
getHostName()	获取此主机名	String
getHostAddress()	获取主机IP地址	String
isReachable(int timeout)	在timeout指定的毫秒时间内，测试IP地址是否可达	boolean

14.3 套 接 字

套接字（Socket）是代表计算机之间网络连接的对象，用于建立计算机之间的TCP连接。

它提供了多种方法，使计算机之间可以建立连接并实现网络通信。

14.3.1 服务器端套接字

服务器端的套接字是ServerSocket类的实例对象，用于实现服务器程序，ServerSocket类将监视指定的端口，并建立客户端到服务器端套接字的连接，也就是客户负责呼叫任务。

服务器端套接字（创建服务器端套接字）

创建服务器端套接字可以使用4种构造方法。

（1）ServerSocket()

默认构造方法，可以创建未绑定端口号的服务器套接字。服务器套接字的所有构造方法都需要处理IOException异常。

一般格式为：

```
try {
        ServerSocket server=new ServerSocket() ;
} catch (IOException e) {
        e.printStackTrace() ;
}
```

服务器端套接字
（创建服务器端套接字 ）

（2）ServerSocket(int port)

该构造方法将创建绑定到port参数指定端口的服务器套接字对象，默认的最大连接队列长度为50，也就是说如果连接数量超出50个，将不会再接收新的连接请求。

一般格式为：

```
try {
        ServerSocket server=new ServerSocket(9527);
} catch (IOException e) {
        e.printStackTrace();
}
```

服务器端套接字
（创建服务器端套接字 ）

（3）ServerSocket(int port, int backlog)

使用port参数指定的端口号和backlog参数指定的最大连接队列长度创建服务器端套接字对象，这个构造方法可以指定超出50的连接数量，例如300。

一般格式为：

```
try {
        ServerSocket server=new ServerSocket(9527, 300);
} catch (IOException e) {
        e.printStackTrace();
}
```

服务器端套接字（创建服务器端套接字）

(4) public ServerSocket(int port, int backlog, InetAddress bindAddr)

使用port参数指定的端口号和backlog参数指定的最大连接队列长度创建服务器端套接字对象，如果服务器有多个IP地址，可以使用bindAddr参数指定创建服务器套接字的IP地址。

一般格式为：

```
try {
    InetAddress address= InetAddress.getByName("192.168.1.128") ;
    ServerSocket server=new ServerSocket(9527,300,address) ;
} catch (IOException e) {
    e.printStackTrace() ;
}
```

服务器端套接字（接受套接字连接）

当服务器建立ServerSocket套接字对象以后，就可以使用该对象的accept()方法接受客户端请求的套接字连接。

语法格式为：

```
serverSocket.accept()
```

该方法被调用之后，将等待客户的连接请求，在接收到客户端的套接字连接请求以后，该方法将返回Socket对象，这个Socket对象是已经和客户端建立好连接的套接字，可以通过这个Socket对象获取客户端的输入输出流来实现数据发送与接收。

服务器端套接字（接受套接字连接）

该方法可能会产生IOException异常，所以在调用accept()方法时必须捕获并处理该异常。

一般格式为：

```
try {
    server.accept();
} catch (IOException e) {
    e.printStackTrace();
}
```

accept()方法将阻塞当前线程，直到接收到客户端的连接请求为止，该方法之后的任何语句都不会被执行，必须有客户端发送连接请求；accept()方法返回Socket套接字以后，当前线程才会继续运行，accept()方法之后的程序代码才会被执行。

14.3.2 客户端套接字

Socket类是实现客户端套接字的基础。它采用TCP建立计算机之间的连接，并包含了Java语言所有对TCP有关的操作方法，例如建立连接、传输数据、断开连接等。

客户端套接字
（创建客户端套接字）

Socket类定义了多个构造方法，它们可以根据InetAddress对象或者字符串指定的IP地址和端口号创建实例。下面介绍一下Socket常用的4个构造方法。

客户端套接字
（创建客户端套接字）

(1) Socket(InetAddress address, int port)

使用address参数传递的IP封装对象和port参数指定的端口号创建套接字实例对象。Socket类的构造方法可能会产生UnknownHostException和IOException异常，在使用该构造方法创建Socket对象时必须捕获和处理这两个异常。

一般格式为：

```
try {
    InetAddress address=InetAddress.getByName("LZW") ; // 创建IP封装类
    int port=33;                                        // 定义端口号
    Socket socket=new Socket(address,port) ;            // 创建套接字
} catch (UnknownHostException e) {
    e.printStackTrace() ;
} catch (IOException e) {
    e.printStackTrace() ;
}
```

客户端套接字
（创建客户端套接字）

（2）Socket(String host, int port)

使用host参数指定的IP地址字符串和port参数指定的整数类型端口号创建套接字实例对象。

一般格式为：

```
try {
        Socket socket=new Socket("192.168.1.1",33);
} catch (UnknownHostException e) {
        e.printStackTrace();
} catch (IOException e) {
        e.printStackTrace();
}
```

客户端套接字
（创建客户端套接字 ）

（3）Socket(InetAddress address, int port, InetAddress localAddr, int localPort)

创建一个套接字并将其连接到指定远程地址的指定远程端口。

一般格式为：

```
try {
      InetAddress localHost = InetAddress.getLocalHost() ;
      InetAddress address = InetAddress.getByName("192.168.1.1") ;
      Socket socket=new Socket(address,33,localHost,44) ;
} catch (UnknownHostException e)  {
      e.printStackTrace() ;
} catch (IOException e)  {
      e.printStackTrace() ;
}
```

客户端套接字
（创建客户端套接字 ）

（4）Socket(String host, int port, InetAddress localAddr, int localPort)

创建套接字并将其连接到指定远程主机上的指定远程端口

一般格式为：

```
try {
      InetAddress localHost = InetAddress.getLocalHost() ;
      Socket socket=new Socket("192.168.1.1",33,localHost,44) ;
} catch (UnknownHostException e)  {
      e.printStackTrace() ;
} catch (IOException e)  {
      e.printStackTrace() ;
}
```

【例3】

客户端套接字（发送和接收数据）

Socket对象创建成功以后，代表和对方的主机已经建立了连接，可以接收和发送数据了。Socket提供了两个方法分别获取套接字的输入流和输出流，可以将要发送的数据写入输出流，实现发送功能，或者从输入流读取对方发送的数据，实现接收功能。

（1）接收数据

Socket对象从数据输入流中获取数据，该输入流中包含对方发送的数据，这些数据可能是文件、图片、音频或视频。所以，在实现接收数据之前，必须使用getInputStream()方法获取输入流。

语法格式为：

```
socket.getInputStream()
```

socket：套接字实例对象。

客户端套接字（发送和接收数据）

（2）发送数据

Socket对象使用输出流，向对方发送数据，所以，在实现数据发送之前，必须使用getOutputStream()方法获取套接字的输出流。

语法格式为：

```
socket.getOutputStream()
```

socket：套接字实例对象。

【例5】

14.4 数 据 报

Java语言可以使用TCP和UDP两种通信协议实现网络通信，其中TCP通信由Socket套接字实现，而UDP通信需要使用DatagramSocket类实现。

UDP传递信息的速度更快，但是没有TCP的高可靠性，当用户通过UDP发送信息之后，无法确定能否正确的传送到目的地。虽然UDP是一种不可靠的通信协议，但是大多数场合并不需要严格的、高可靠性的通信，它们需要的是快速的信息发送，并能容忍一些小的错误，那么使用UDP通信来实现会更合适一些。

UDP将数据打包，也就是通信中所传递的数据包，然后将数据包发送到指定目的地，对方会接收数据包，然后查看数据包中的数据。

14.4.1 DatagramPacket

该类是UDP所传递的数据包，即打包后的数据。数据包用来实现无连接包投递服务。每个数据包仅根据包中包含的信息从一台计算机传送到另一台计算机，传送的多个包可能选择不同的路由，也可能按不同的顺序到达。

DatagramPacket类提供了多个构造方法用于创建数据包的实例，下面介绍最常用的两个。

DatagramPacket

(1) DatagramPacket(byte[] buf, int length)

该构造方法用来创建数据包实例，这个数据包实例将接收长度为length的数据包。

语法格式为：

```
DatagramPacket(byte[] buf, int length)
```

buf：保存传入数据报的缓冲区。

len：要读取的字节数。

DatagramPacket

(2) DatagramPacket(byte[] buf, int length, InetAddress address, int port)

创建数据报包实例，用来将长度为length的数据包发送到address参数指定地址和port参数指定端口号的主机。length参数必须小于等于buf数组的长度。

语法格式为：

```
DatagramPacket(byte[] buf, int length, InetAddress address, int port)
```

buf：包数据。

length：包长度。

address：目的地址。

port：目的端口号。

14.4.2 DatagramSocket

该类是用于发送和接收数据的数据报套接字。数据报套接字是数据包传送服务的发送或接收点。要实现UDP通信的数据发送就必须创建数据报套接字。

DatagramSocket类提供了多个构造方法用于创建数据报套接字，下面介绍最常用的3个构造方法。

DatagramSocket

(1) DatagramSocket()

默认的构造方法，该构造方法将使用本机任何可用的端口创建数据报套接字实例。在创建DatagramSocket类的实例时，有可能会产生SocketException异常，所以在创建数据报套接字时，应该捕获并处理该异常。

一般格式为：

```
try {
    DatagramSocket dsocket=new DatagramSocket();
} catch (SocketException e) {
    e.printStackTrace();
}
```

DatagramSocket

(2) DatagramSocket(int port)

创建数据报套接字并将其绑定到port参数指定的本机端口，端口号取值必须在0～65535（包括两者）。

一般格式为：

```
try {
    DatagramSocket dsocket=new DatagramSocket(9527);
} catch (SocketException e) {
    e.printStackTrace();
}
```

DatagramSocket

(3) DatagramSocket(int port, InetAddress laddr)

创建数据报套接字，将其绑定到laddr参数指定的本机地址和port参数指定的本机端口号。本机端口号取值必须在0～65535之间（包括两者）。

一般格式为：

```
try {
    InetAddress localHost = InetAddress.getLocalHost();
    DatagramSocket dsocket=new DatagramSocket(9527,localHost);
} catch (SocketException e) {
    e.printStackTrace();
} catch (UnknownHostException e) {
    e.printStackTrace();
}
```

在了解了DatagramPacket类和DatagramSocket类之后，就可以使用这两个类实现UDP通信程序设计。