

Java Socket编程入门

原创 2014年03月31日 09:52:48

👁 3084

💬 0

❤ 13

编辑 (<http://write.blog.csdn.net/postedit/22647913>)

删除

Java Socket编程入门

摘要：对Java Socket（套接字）编程做入门的整理、学习的东西太散就老感觉既熟悉又陌生、记录一下、没事的时候可以翻翻看看。以后工作中用到的时候也不会两眼一抹黑、说听过、但是具体说个一二三却哑口无言。注：概念性的解释都是摘抄自网络！

一：简介

套接字编程、也是网络编程的基础。网络编程所围绕的核心就是两个或者多个网络上的主机之间信息的传递。所以要实现网络编程就要面临两个问题：

- 1、如何准确定位网络上的一台或者多台主机？
- 2、当定位成功之后、他们之间如何进行有效可靠的信息传递？

在TCP/IP协议中IP层主要负责网络主机的定位，数据传输的路由，由IP地址可以唯一地确定Internet上的一台主机。

而TCP层则提供面向应用的可靠（tcp）的或非可靠（UDP）的数据传输机制，这是网络编程的主要对象，一般不需要关心IP层是如何处理数据的。

3、目前较为流行的网络编程模型是客户机/服务器（C/S）结构。即通信双方一方作为服务器等待客户提出请求并予以响应。客户则在需要服务时向服务器提出申请。服务器一般作为守护进程始终运行，监听网络端口，一旦有客户请求，就会启动一个服务进程来响应该客户，同时自己继续监听服务端口，使后来的客户也能及时得到服务。

1、TCP/IP 概念

TCP/IP协议（传输控制协议）由网络层的IP协议和传输层的TCP协议组成。IP层负责网络主机的定位，数据传输的路由，由IP地址可以唯一的确定Internet上的一台主机。TCP层负责面向应用的可靠的或非可靠的数据传输机制，这是网络编程的主要对象。

2、TCP、UDP区别

TCP是一种面向连接的保证可靠传输的协议。通过TCP协议传输，得到的是一个顺序的无差错的数据流。发送方和接收方的成对的两个socket之间必须建立连接，以便在TCP协议的基础上进行通信，当一个socket（通常都是server socket）等待建立连接时，另一个socket可以要求进行连接，一旦这两个socket连接起来，它们就可以进行双向数据传输，双方都可以进行发送或接收操作。

UDP是一种面向无连接的协议，每个数据报都是一个独立的信息，包括完整的源地址或目的地址，它在网络上以任何可能的路径传往目的地，因此能否到达目的地，到达目的地的时间以及内容的正确性都是不能被保证的。

TCP与UDP区别：
TCP特点：

1、TCP是面向连接的协议，通过三次握手建立连接，通讯完成时要拆除连接，由于TCP是面向连接协议，所以只能用于点对点的通讯。而且建立连接也需要消耗时间和开销。



Oscar Chen (<http://blog.csdn.net/chenghuaying>)

+ 关注

(<http://blog.csdn.net/chenghuaying>)

原创	粉丝	喜欢
182	14	1

- > CentOS 集群机器之间ssh免密 (/crave_shy/article/details/72964997)
- > JVM-内存管理-运行时数据区域 (/crave_shy/article/details/56675052)
- > JVM-Blog目录 (/crave_shy/article/details/56675032)
- > JVM-为什么要学JVM (/crave_shy/article/details/56673439)

更多文章
(<http://blog.csdn.net/chenghuaying>)

在线课程



(http://edu.csdn.net/huiyiCourse/series_detail?utm_source=blog7)

【直播】机器学习&数据挖掘7周实训--韦玮

(http://edu.csdn.net/huiyiCourse/series_detail/54?utm_source=blog7)



(http://edu.csdn.net/combo/detail/471?utm_source=blog7)

【套餐】系统集成项目管理工程师顺利通关--徐朋

(http://edu.csdn.net/combo/detail/471?utm_source=blog7)

- 2、TCP传输数据无大小限制，进行大数据传输。
- 3、TCP是一个可靠的协议，它能保证接收方能够完整正确地接收到发送方发送的全部数据。

UDP特点：

- 1、UDP是面向无连接的通讯协议，UDP数据包括目的端口号和源端口号信息，由于通讯不需要连接，所以可以实现广播发送。
- 2、UDP传输数据时有大小限制，每个被传输的数据报必须限定在64KB之内。
- 3、UDP是一个不可靠的协议，发送方所发送的数据报并不一定以相同的次序到达接收方。

TCP与UDP应用：

- 1、TCP在网络通信上有极强的生命力，例如远程连接（Telnet）和文件传输（FTP）都需要不定长度的数据被可靠地传输。但是可靠的传输是要付出代价的，对数据内容正确性的检验必然占用计算机的处理时间和网络的带宽，因此TCP传输的效率不如UDP高。
- 2、UDP操作简单，而且仅需要较少的监护，因此通常用于局域网高可靠性的分散系统中client/server应用程序。例如视频会议系统，并不要求音频视频数据绝对的正确，只要保证连贯性就可以了，这种情况下显然使用UDP会更合理一些。

3、Socket概念

Socket通常也称作"套接字"，用于描述IP地址和端口，是一个通信链的句柄。网络上的两个程序通过一个双向的通讯连接实现数据的交换，这个双向链路的一端称为一个Socket，一个Socket由一个IP地址和一个端口号唯一确定。应用程序通常通过"套接字"向网络发出请求或者应答网络请求。Socket是TCP/IP协议的一个十分流行的编程界面，但是，Socket所支持的协议种类也不光TCP/IP一种，因此两者之间是没有必然联系的。在Java环境下，Socket编程主要是指基于TCP/IP协议的网络编程。

Socket通讯过程：服务端监听某个端口是否有连接请求，客户端向服务端发送连接请求，服务端收到连接请求向客户端发出接收消息，这样一个连接就建立起来了。客户端和服务端都可以相互发送消息与对方进行通讯。

Socket的基本工作过程包含以下四个步骤：

- 1、创建Socket；
- 2、打开连接到Socket的输入输出流；
- 3、按照一定的协议对Socket进行读写操作；
- 4、关闭Socket。

4、Java Socket

网络上的两个程序通过一个双向的通讯连接实现数据的交换，这个双向链路的一端称为一个Socket。Socket通常用来实现客户方和服务方的连接。Socket是TCP/IP协议的一个十分流行的编程界面，一个Socket由一个IP地址和一个端口号唯一确定。但是，Socket所支持的协议种类也不光TCP/IP一种，因此两者之间是没有必然联系的。在Java环境下，Socket编程主要是指基于TCP/IP协议的网络编程。

Socket通讯的过程

Server端Listen(监听)某个端口是否有连接请求，Client端向Server 端发出Connect(连接)请求，Server端向Client端发回Accept（接受）消息。一个连接就建立起来了。Server端和Client 端都可以通过Send，Write等方法与对方通信。

对于一个功能齐全的Socket，都要包含以下基本结构，其工作过程包含以下四个基本的步骤：

- （1）创建Socket；
- （2）打开连接到Socket的输入/出流；
- （3）按照一定的协议对Socket进行读/写操作；
- （4）关闭Socket。

5、最基本的Server/Client程序

- 1、Server端

```

package com.chy.socket.server;

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;

public class SingleClientServer {
    private final int port = 7778;

    public SingleClientServer() {
        try {
            ServerSocket server = new ServerSocket(port);
            Socket client = server.accept();
            BufferedReader br = new BufferedReader(new InputStreamReader(client.getInpu
tStream()));

            System.out.println("client say : " + br.readLine());

            BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(client.getOut
putStream()));

            bw.write("Hello client, this is server...");
            bw.write("\r\n");
            bw.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        new SingleClientServer();
    }
}

```

2、Client端

```

package com.chy.socket.client;

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 com.chy.socket.utils.ArgumentsUtil;

public class Client_1 {

    public Client_1() {
        try {
            Socket client = new Socket(ArgumentsUtil.host, ArgumentsUtil.port);
            BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(client.getOut
putStream()));

            bw.write("Hello server, this is " + this.getClass().getName() + " clien
t..." + "\r\n");

            bw.write("\r\n");
            bw.flush();

            BufferedReader br = new BufferedReader(new InputStreamReader(client.getInpu
tStream()));

            System.out.println(br.readLine() );
            client.close();
            bw.close();
            br.close();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        new Client_1();
    }
}

```

6、多客户端连接同一服务端

1、Server端

```
package com.chy.socket.server;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

import com.chy.socket.thread.ServerThread;
import com.chy.socket.utils.ArgumentsUtil;

public class ManyClientServer {
    private Socket client = null;

    public ManyClientServer() {
        try {
            ServerSocket server = new ServerSocket(ArgumentsUtil.port);
            while(true){
                client = server.accept();
                //print the context that is sent by client, and say hello to every
client.
                new ServerThread(client).start();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    public static void main(String[] args) {
        new ManyClientServer();
    }
}
```

2、Server端的Thread

```
package com.chy.socket.thread;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class ServerThread extends Thread {

    private InputStream is = null;
    private OutputStream os = null;
    private Socket client = null;

    public ServerThread(Socket client) {
        try {
            this.is = client.getInputStream();
            this.os = client.getOutputStream();
            this.client = client;
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(is));
            BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(os));

            System.out.println(getName() + " say : " + br.readLine());

            bw.write("Hello " + Thread.currentThread().getName() + " , this is serve
r..." + "\r\n");

            bw.flush();

            client.close();
            br.close();
            bw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

3、Client端——Client_2 Client_3 Client_4 基本类似

```
package com.chy.socket.client;

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 com.chy.socket.utils.ArgumentsUtil;

public class Client_2 {
    public Client_2() {
        try {
            Socket client = new Socket(ArgumentsUtil.host, ArgumentsUtil.port);
            BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(client.getOut
putStream()));
            bw.write("Hello server, this is " + this.getClass().getName() + " clien
t..." + "\r\n");
            bw.flush();

            BufferedReader br = new BufferedReader(new InputStreamReader( client.getInp
utStream()));
            System.out.println(br.readLine());

            client.close();
            br.close();
            bw.close();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        new Client_2();
    }
}
```

4、Client_3 :

```

package com.chy.socket.client;

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 com.chy.socket.utils.ArgumentsUtil;

public class Client_3 {

    public Client_3() {
        try {
            Socket client = new Socket(ArgumentsUtil.host, ArgumentsUtil.port);
            BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(client.getOutputStream()));

            bw.write("Hello server, this is " + this.getClass().getName() + " client..." + "\r\n");

            bw.flush();

            BufferedReader br = new BufferedReader(new InputStreamReader(client.getInputStream()));

            System.out.println(br.readLine());

            client.close();
            br.close();
            bw.close();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        new Client_3();
    }
}

```

7、文件传输

1、Server端：

```

package com.chy.socket.server;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

import com.chy.socket.utils.ArgumentsUtil;

/**
 * File upload server.
 */
public class Server extends ServerSocket{

    private ServerSocket server;
    private Socket client;
    private DataInputStream dis;
    private FileOutputStream fos;

    public Server()throws Exception{
        try {
            try {
                server =new ServerSocket(ArgumentsUtil.port);

                while(true){
                    client = server.accept();

                    dis =new DataInputStream(client.getInputStream());
                    //文件名和长度
                    String fileName = dis.readUTF();
                    long fileLength = dis.readLong();
                    fos =new FileOutputStream(new File("d:" + fileName));

                    byte[] sendBytes =new byte[1024];
                    int transLen =0;
                    System.out.println("----开始接收文件<" + fileName +">,文件大小为<" + fileLength
+ ">----");

                    while(true){
                        int read =0;
                        read = dis.read(sendBytes);
                        if(read == -1)
                            break;
                        transLen += read;
                        System.out.println("接收文件进度" +100 * transLen/fileLength +"%...");
                        fos.write(sendBytes,0, read);
                        fos.flush();
                    }
                    System.out.println("----接收文件<" + fileName +">成功-----");
                    client.close();
                }
            }catch (Exception e) {
                e.printStackTrace();
            }finally {
                if(dis !=null)
                    dis.close();
                if(fos !=null)
                    fos.close();
                server.close();
            }
        }catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args)throws Exception {
        new Server();
    }
}

```

2、Client端：

```

package com.chy.socket.client;

import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.net.Socket;

import com.chy.socket.utils.ArgumentsUtil;

/**
 * 客户端
 */
public class Client extends Socket{

    private Socket client;
    private FileInputStream fis;
    private DataOutputStream dos;

    public Client(){
        try {
            try {
                client =new Socket(ArgumentsUtil.host, ArgumentsUtil.port);
                //向服务端传送文件
                File file =new File("c:/test.doc");
                fis =new FileInputStream(file);
                dos =new DataOutputStream(client.getOutputStream());

                //文件名和长度
                dos.writeUTF(file.getName());
                dos.flush();
                dos.writeLong(file.length());
                dos.flush();

                //传输文件
                byte[] sendBytes =new byte[1024];
                int length =0;
                while((length = fis.read(sendBytes,0, sendBytes.length)) >0){
                    dos.write(sendBytes,0, length);
                    dos.flush();
                }
            }catch (Exception e) {
                e.printStackTrace();
            }finally{
                if(fis !=null)
                    fis.close();
                if(dos !=null)
                    dos.close();
                client.close();
            }
        }catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args)throws Exception {
        new Client();
    }
}

```

补充与总结

1、补充：

上面多次用到的、用于获取IP 和端口号的工具类ArgumentsUtil：

```

package com.chy.socket.utils;

public class ArgumentsUtil {
    public static int port = 7778;
    public static String host = "127.0.0.1";
}

```

2、总结：

Java Socket入门很简单、难的是如何处理好在多线程的环境下对各种信息的处理、还有流的正确使用、不会因为对流的工作原理不理解而导致各种各样的奇葩的问题。建议在写多线程的环境下程序的时候、写一点调试一点、如果只顾着闷着头写、到最后出问题再回来调是非常困难的！

标签：socket (<http://so.csdn.net/so/search/s.do?q=socket&t=blog>) /
多线程 (<http://so.csdn.net/so/search/s.do?q=多线程&t=blog>) /
clientserver (<http://so.csdn.net/so/search/s.do?q=clientserver&t=blog>) /
网络编程 (<http://so.csdn.net/so/search/s.do?q=网络编程&t=blog>) /
通信 (<http://so.csdn.net/so/search/s.do?q=通信&t=blog>) /

0条评论



qq_36596145 (http://my.csdn.net/qq_36596145)

(http://my.csdn.net/qq_36596145)



发表评论

暂无评论

相关文章推荐

深入浅出 **RPC** - 深入篇 (</mindfloating/article/details/39474123>)

《深入篇》我们主要围绕 RPC 的功能目标和实现考量去展开，一个基本的 RPC 框架应该提供什么功能，满足什么要求以及如何去实现它？RPC 功能目标 RPC 的主要功能目标是让构建分布式计算（应用）...



mindfloating 2014-09-22 11:25 183639

JAVA中几种常用的**RPC**框架介绍 (</hj419460467/article/details/72674655>)

JAVA中几种常用的RPC框架介绍



hj419460467 2017-05-24 11:19 249

基于**Netty**的高性能**JAVA**的**RPC**框架 (</zhujunxxxxx/article/details/48742529>)

前言今年7月份左右报名参加了阿里巴巴组织的高性能中间件挑战赛，这次比赛不像以往的比赛，是从一个工程的视角来比赛的。这个比赛有两个赛题，第一题是实现一个RPC框架，第二道题是实现一个Mom消息中间件...



zhujunxxxxx 2015-09-26 00:11 7791

实现**java** **RPC**框架 (</sunmenggmail/article/details/8545701>)

<http://javatar.iteye.com/blog/1123915> 主要利用socket通信，反射，代理实现类似RMI的RPC框架 首先是框架的代码 package fr...



sunmenggmail 2013-01-27 00:25 10175

JAVA中几种常用的**RPC**框架介绍 (</volts/article/details/54672525>)

RPC是远程过程调用的简称，广泛应用在大规模分布式应用中，作用是有助于系统的垂直拆分，使系统更易拓展。Java中的RPC框架比较多，各有特色，广泛使用的有RMI、Hessian、Dubbo等。




volts 2017-01-22 22:36 1551

JAVA中几种常用的**RPC**框架介绍 (</zhaowen25/article/details/45443951>)


RPC是远程过程调用的简称，广泛应用在大规模分布式应用中，作用是有助于系统的垂直拆分，使系统更易拓展。

Java中的RPC框架比较多，各有特色，广泛使用的有RMI、Hessian、Dubbo等。1、R...




zhaowen25

2015-05-02 23:17

 51720


Zookeeper的集群配置和Java测试程序 (/catoop/article/details/50848555)

概述Zookeeper是Apache下的项目之一，倾向于对大型应用的协同维护管理工作。IBM则给出了IBM对ZooKeeper的认知：Zookeeper 分布式服务框架是 Apache Hadoop...




catoop

2016-03-10 17:28

 12882


Redis 缓存 + Spring 的集成示例 (/defonds/article/details/48716161)

《整合 spring 4(包括mvc、context、orm) + mybatis 3 示例》一文简要介绍了 Spring MVC、IOC、MyBatis ORM 三者的集成以及声明式事务处理。本文将...




defonds

2015-09-24 19:53

 108291


RPC框架与Dubbo完整使用 (/u010297957/article/details/51702076)

这并不是原理性的解释文章。只是快速入门，还有一个完整的Java例子。 一篇我觉得不错的文章推荐：深入浅出RPC - 浅出篇 一、RPC 什么是RPC ? RPC (Remote Procedure Ca...




u010297957

2016-06-21 19:28

 13571


Java_io体系之File、FileInputStream、FileOutputStream简介、走进源码及示例——05 (/crave_shy/article/details/16922777)

Java_io体系之File、FileInputStream、FileOutputStream简介、走进源码及示例——05 一：File 1、File类简介： 它既可以表示...




chenghuaying

2013-11-24 18:50

 2612


Java Socket编程入门 (/manzhizhen/article/details/52606712)

Java Socket编程入门 1.必备知识 TCP是Tranfer Control Protocol的简称，即传输控制协议，基于TCP协议，可以进行有顺序的，无差错的数据流传输...



manzhizhen

2016-09-21 13:04

 124

Java网络编程从入门到精通（29）： 服务端Socket的选项 (http://haierboos.iteye.com/blog/1440383)

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <meta name="ProgId" content="Word.Document"> <meta name="Generator" content="Microsoft Word 11"> <meta name="Originator" content="Microsoft Word 11"> <link rel="File-List" href="file:///C:%5CDOCUME%7E1%5CA




haierboos

2009-08-12 14:47

 52


Java socket编程入门[1] (/ronbi/article/details/2215859)

导读： Java socket编程入门[1] [原创] Robinh00d 2004-02-19 -----...




Ronbi

2008-03-25 10:24

 1206


Java网络编程从入门到精通（17）： Socket类的getter和setter方法（1） (http://seara.iteye.com/blog/893226)

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <meta name="ProgId" content="Word.Document"> <meta name="Generator" content="Microsoft Word 11"> <meta name="Originator" content="Microsoft Word 11"> <link rel="File-List" href="file:///C:%5CDOCUME%7E1%5CA

seara2009-05-29 09:15👁83


Java网络编程从入门到精通（13）：使用Socket类接收和发送数据
(http://seara.iteye.com/blog/893575)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（12）：使用isReachable方法探测主机是否可以连通 网络应用分为客户端和服务端两部分，而Sock

seara2009-05-14 10:16👁78

Java网络编程从入门到精通（18）： Socket类的getter和setter方法（2）
(/nokiaguy/article/details/4684662)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（17）：Socket类的getter和setter方法（1）二、用于获得和设置Socket选项的getter和set...

nokiaguy2009-06-01 17:15👁833


Java网络编程从入门到精通（16）： 客户端套接字（Socket）的超时
(http://haierboos.iteye.com/blog/1442199)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（15）：为什么要使用SocketAddress来管理网络地址 客户端套接字的超时

haierboos2009-05-26 08:15👁215


java socket一对多通信编程 (/uglikewater/article/details/50488556)

先介绍一些基本常识。socket是基于TCP/IP协议的。通过IP协议找到目的主机，在通过TCP协议完成数据链路的联通。然后开始收发数据。（当然，个人理解。若有不对，敬请指正）socket的输入流...

UGLikeWater2016-01-09 17:30👁354

Java socket编程入门[1] (http://daimajishu.iteye.com/blog/1079195)

导读：Java socket编程入门[1] [原创] Robinh00d 2004-02-19 -----
----- 本教程由IBM DevelopWorks提供，版权归IBM所有 原作：
Roy Miller 翻译：Robinh00d[CSTC] 翻译本文的目的仅仅是为了练习，本教程版权归IBM所有，本人不对其
拥有版权 第一章：学习提示 我适合学习这份教程吗

isiqi2008-03-25 10:24👁322

Java网络编程从入门到精通（17）： Socket类的getter和setter方法（1）
(/nokiaguy/article/details/4684666)

Normal07.8 磅02falsefalsefalseMicrosoftInternetExplorer4st1".*{behavior:url(#ieooui) }

Java网络编程从入门到精通（17）： Socket类的getter和setter方法（1）
(http://seara.iteye.com/blog/892886)

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <meta name="ProgId" content="Word.Document"> <meta name="Generator" content="Microsoft Word 11"> <meta name="Originator" content="Microsoft Word 11"> <link rel="File-List" href="file:///C:%5CDOCUME%7E1%5CA



seara 2009-05-29 09:15 85

java的socket网络编程 (/anitak/article/details/50133489)

当时毕设的时候有想过做这个 结果还是做了asp.net 读研了，计算机网络的实验课大作业就是这个，看了《java编程那些事》《socket网络编程》《极客学院网络编程的课》，大致也有了入门的了解。最...



anitak 2015-12-01 18:45 170

Java网络编程从入门到精通（13）：使用Socket类接收和发送数据 (http://haierboos.iteye.com/blog/1440478)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（12）：使用isReachable方法探测主机是否可以连通 网络应用分为客户端和服务端两部分，而Sock



haierboos 2009-05-14 10:16 240

Java网络编程入门 (/liuyijie5566/article/details/7976265)

Java网络编程入门 服务器与客户程序只需关心发送什么样的数据给对方，而不必考虑如何把这些数据传输给对方，传输数据的任务由计算机网络完成。 两个进程顺利通信的前提条件是它们所在的主机都连接...



liuyijie5566 2012-09-13 19:00 325

Java网络编程从入门到精通（13）：使用Socket类接收和发送数据 (http://seara.iteye.com/blog/892554)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（12）：使用isReachable方法探测主机是否可以连通 网络应用分为客户端和服务端两部分，而Sock



seara 2009-05-14 10:16 219

linux网络编程入门——基于socket的proxy (/clearriver/article/details/4725205)

学习Linux 网络编程有一段时间了，作为一个总结，写一个基于socket的proxy，算是复习前面的所学。一直以来就有一个写proxy的想法，实验室项目中我所负责的模块就是一个Http proxy，...



clearriver 2009-10-25 10:40 1659

Java网络编程从入门到精通（16）：客户端套接字（Socket）的超时 (http://haierboos.iteye.com/blog/1440468)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（15）：为什么要使用SocketAddress来管理网络地址 客户端套接字的超时



haierboos 2009-05-26 08:15 68

Java网络编程入门 (/zhangguoliang521/article/details/7975060)

Java网络编程入门 服务器与客户程序只需关心发送什么样的数据给对方，而不必考虑如何把这些数据传输给对方，传输数据的任务由计算机网络完成。 两个进程顺利通信的前提条件是它们所在的主机都连接到了计算...




zhangguoliang521 2012-09-13 14:59 1153

Java网络编程从入门到精通（29）：服务端Socket的选项 (http://seara.iteye.com/blog/892499)


<meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <meta name="ProgId" content="Word.Document"> <meta name="Generator" content="Microsoft Word 11">

<meta name="Originator" content="Microsoft Word 11"> <link rel="File-List" href="file:///C:%5CDOCUME%7E1%5CA

seara2009-08-12 14:47👁236

java网络编程入门到精通 (/zpf644792799/article/details/6738566)


Java网络编程从入门到精通（1）：Internet地址概述 . Java网络编程从入门到精通（2）：创建InetAddress对象的四个静态方法 . Ja...


zpf6447927992011-09-01 11:13👁704



Java网络编程从入门到精通（13）：使用Socket类接收和发送数据 (http://haierboos.iteye.com/blog/1442209)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（12）：使用isReachable方法探测主机是否可以连通 网络应用分为客户端和服务端两部分，而Sock


haierboos2009-05-14 10:16👁219

喜欢收藏




Java网络编程从入门到精通（13）：使用Socket类接收和发送数据 (/nokiaguy/article/details/4684677)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（12）：使用isReachable方法探测主机是否可以连通 网络应用分为客户端和服务端两部分，而Socket类...

nokiaguy2009-05-14 10:16👁1191


Java网络编程从入门到精通（16）：客户端套接字（Socket）的超时 (http://seara.iteye.com/blog/893229)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（15）：为什么要使用SocketAddress来管理网络地址 客户端套接字的超时

seara2009-05-26 08:15👁102


Java socket编程入门[1] (/java169/article/details/2476090)

google_ad_client = "pub-8800625213955058";/* 336x280, 创建于 07-11-21 */google_ad_slot = "0989131976";g...

java1692008-05-24 04:26👁250


Java网络编程从入门到精通（16）：客户端套接字（Socket）的超时 (http://seara.iteye.com/blog/894463)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（15）：为什么要使用SocketAddress来管理网络地址 客户端套接字的超时

seara2009-05-26 08:15👁204

Socket编程入门 (/xiangliling/article/details/8097661)

在网上看了半天，看得迷迷糊糊、不知所云，索性打开eclipse小小的试了一下。说是试一下，其实就是把别人的例子敲了一遍，不过思路总算比之前清晰多了例子是这样的：用户在客户端输入一句话，就在服务器端打印...

xiangliling2012-10-22 11:00👁124

Java网络编程从入门到精通（17）：Socket类的getter和setter方法（1） (http://haierboos.iteye.com/blog/1442196)

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <meta name="ProgId" content="Word.Document"> <meta name="Generator" content="Microsoft Word 11"> <meta name="Originator" content="Microsoft Word 11"> <link rel="File-List" href="file:///C:%5CDOCUME%7E1%5CA

 haierboos 2009-05-29 09:15  206



基于Socket的低层次Java网络编程 (/xi_hong_shi/article/details/7876844)

8.3.1 Socket通讯 网络上的两个程序通过一个双向的通讯连接实现数据的交换，这个双向链路的一端称为一个Socket。Socket通常用来实现客户方和服务方的连接。Socket是TCP/I...

 xi_hong_shi 2012-08-17 12:41  1773



Java网络编程从入门到精通（19）：套接字(Socket)的异常 (http://seara.iteye.com/blog/893222)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（18）：Socket类的getter和setter方法（2）<s

 seara 2009-06-02 08:15  83

Java中利用socket实现简单的服务端与客户端的通信（入门级） (/qq_25352981/article/details/49930913)

Java编程中，要想要使用网络通信，就离不开Socket编程，在此对socket进行简单的介绍。首先声明，这是一个入门级的介绍，仅仅简单的实现了客户端向服务端发送数据，服务端正常的接收数据，当接收到特...

 qq_25352981 2015-11-19 18:15  2481



Java网络编程从入门到精通（29）：服务端Socket的选项 (http://haierboos.iteye.com/blog/1442122)

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <meta name="ProgId" content="Word.Document"> <meta name="Generator" content="Microsoft Word 11"> <meta name="Originator" content="Microsoft Word 11"> <link rel="File-List" href="file:///C:%5CDOCUME%7E1%5CA

 haierboos 2009-08-12 14:47  53



JAVA Socket编程进阶(一) (/mochi_chen/article/details/7784632)

上一篇文章里面介绍了JAVA Socket编程入门简介，在示例中介绍了服务端和客户端之间的通信和交互，这个示例中服务端一次只能接受一个客户的连接，这显然不能满足我们对Socket编程的探索。今天我们将...

 MOCHI_CHEN 2012-07-25 14:29  162

Java网络编程从入门到精通（13）：使用Socket类接收和发送数据 (http://seara.iteye.com/blog/892897)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（12）：使用isReachable方法探测主机是否可以连通 网络应用分为客户端和服务端两部分，而Sock

 seara 2009-05-14 10:16  105

Java网络编程之Socket入门 (/chen1678940/article/details/25671329)

套接字 (Socket)



chen1678940 2014-05-13 07:04 188

Java网络编程从入门到精通（17）：Socket类的getter和setter方法（1） (<http://haierboos.iteye.com/blog/1440465>)

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" > <meta name="ProgId" content="Word.Document" > <meta name="Generator" content="Microsoft Word 11" > <meta name="Originator" content="Microsoft Word 11" > <link rel="File-List" href="file:///C:%5CDOCUME%7E1%5CA



haierboos 2009-05-29 09:15 58

【Socket编程一】Java Socket编程入门介绍 (/noaman_wgs/article/details/55218310)

Java Socket编程入门介绍 1 计算机之间通信条件 在介绍Socket之前先简单介绍下计算机之间通讯所需要的条件：IP地址，协议，端口号。IP地址：为实现网络中不同计算机之间的通信，每台计算...



noaman_wgs 2017-02-16 13:32 109

Linux socket 网络编程入门 (</u012707739/article/details/75944204>)

Linux下的网络编程一般即是指socket套接字编程，入门比较矮简单，网上也有很多入门的例程。不过每次看过用过以后过段时间又忘了具体的操作了，又得去查，所以在这里总结整理一下，也省了以后查别人教程的...



u012707739 2017-07-23 18:11 1

Java网络编程从入门到精通（13）：使用Socket类接收和发送数据 (<http://seara.iteye.com/blog/894470>)

本文为原创，如需转载，请注明作者和出处，谢谢！ 上一篇：Java网络编程从入门到精通（12）：使用isReachable方法探测主机是否可以连通 网络应用分为客户端和服务端两部分，而Sock



seara 2009-05-14 10:16 230