

知识点列表

编号	名称	描述	级别
1	Timer (定时器)	掌握 Timer 类的使用	**
2	Socket 编程_TCP 服务器端	理解并掌握 Java Socket 编程	**
3	Socket 编程_TCP 客户端	理解并掌握 Java Socket 编程	**

注： **"理解级别 ***"掌握级别 ****"应用级别

目录

1. Timer (定时器) **	3
2. Java Socket 编程 TCP 协议编程	5
2.1. Socket 编程_TCP 服务器端 **	5
2.2. Socket 编程_TCP 客户端 **	8

1. Timer (定时器) **

java.util.Timer 功能

- 1) 可以安排自动的计划任务的类, 每个任务都是一个线程
- 2) 创建 Timer 实例
- 3) 为 Timer 实例增加计划任务, 计划任务是一个接口
- 4) 使用 cancel() 清除 timer 对象上所有计划任务

【案例 1】Timer (定时器) 演示

```

1 package corejava.day11.ch08;
2 import java.util.Calendar;
3
4 /** 定时器演示: 周末爬香山 */
5 public class TimerDemo {
6     public static void main(String[] args) {
7         //1. 创建定时器对象
8         Timer timer = new Timer();
9         //2. 设置日期为本周的周六
10        Calendar cal = new GregorianCalendar();
11        cal.set(Calendar.DAY_OF_WEEK, Calendar.SATURDAY);
12        Date date = cal.getTime();
13        //3. 设置任务
14        //在指定时间执行计划任务的run()方法
15        timer.schedule(new TimerTask() {
16            public void run() {
17                System.out.println("去爬香山!");
18            }
19        }, date);
20    }
21 }

```

【案例 2】Timer (定时器) 演示

```

1 package corejava.day11.ch08;
2 import java.util.Timer;
3
4 /** 定时器演示: 3秒后爬香山 */
5 public class TimerDemo {
6     public static void main(String[] args) {
7         //1. 创建定时器对象
8         Timer timer = new Timer();
9         //2. 设置任务
10        //在指定时间执行计划任务的run()方法

```

```

11 timer.schedule(new TimerTask() {
12     public void run() {
13         System.out.println("去爬香山!");
14     }
15 }, 3000);
16 }
17 }

```

【案例 3】Timer (定时器) 演示

Timer 定时器任务 (schedule) 会一直执行, 如果想取消, 需要设置 timer.cancel()

```

TimerDemo2.java
1 package corejava.day11.ch08;
2 import java.util.Timer;
3 import java.util.TimerTask;
4
5 /** 定时器演示:
6  * 5秒时输出"爆炸", 3秒时取消
7  */
8 public class TimerDemo2 {
9     public static void main(String[] args) {
10         final Timer timer = new Timer();
11         //在指定时间执行计划任务的run() 方法
12         timer.schedule(new TimerTask() {
13             public void run() {
14                 System.out.println("爆炸!");
15             }
16         }, 5000);
17         timer.schedule(new TimerTask() {
18             public void run() {
19                 timer.cancel(); //取消定时器
20             }
21         }, 3000);
22     }
23 }

```

【案例 4】倒计时

```

1 package corejava.day12.ch02;
2 import java.util.Date;
3
4
5
6 /** 定时器演示：倒计时 */
7 public class TimerDemo {
8     public static void main(String[] args) {
9         //min 5分钟
10        int min = 5;
11        long start = System.currentTimeMillis();
12        //end 计算结束时间
13        final long end = start + min*60*1000;
14
15        final Timer timer = new Timer();
16        //延迟0毫秒(即立即执行)开始，每隔1000毫秒执行一次
17        timer.schedule(new TimerTask() {
18            public void run() {
19                //show 是剩余时间，即要显示的时间
20                long show = end-System.currentTimeMillis();
21                long h=show/1000/60/60; //时
22                long m=show/1000/60%60; //分
23                long s=show/1000%60;    //秒
24                System.out.println(h+"":"+m+": "+"s");
25            }
26        }, 0, 1000);
27        //计时结束时候，停止全部timer计时任务
28        timer.schedule(new TimerTask() {
29            public void run() {
30                timer.cancel();
31            }
32        }, new Date(end));
33    }
34 }

```

2. Java Socket 编程 TCP 协议编程

2.1. Socket 编程 TCP 服务器端 **

TCP 服务器端编程需要如下包 (package) 中的类

- ✓ java.io.*
- ✓ java.net.*
- ✓ Java.lang.*

编程步骤：

- 1) 创建 ServerSocket 实例绑定一个服务端口 (Socket/套接字 端口号)
- 2) 开始 ServerSocket 实例的监听, 等待客户端的连接
- 3) 如果有客户连接进来, 就获得了客户的套接字(Socket)实例
客户的套接字(Socket)实例中包括与客户端建立的连接流
- 4) 为这个客户(Socket) 创建一个服务线程, 提供服务 (run()方法)
- 5) 继续等待下一个连接, 返回到 2)
- 6) 服务线程完成通讯服务过程
- 7) 端口号范围为: 0~65535, 但是 1K 以下 (0~1024) 是留给系统使用的

【案例】TCP 服务器端编程演示

```

ServerDemo.java
1 package corejava.day14.ch01;
2 import java.io.IOException;
3 /** TCP 服务器演示:
4  * 客户端命令: telnet localhost 8000
5  */
6 public class ServerDemo {
7     public static void main(String[] args)
8         throws IOException{
9         ServerDemo server = new ServerDemo();
10        server.start();
11    }
12
13    public void start() throws IOException{
14        //1. 在服务器上绑定8000服务端口号, 不能重复绑定
15        ServerSocket ss = new ServerSocket(8000);
16        while(true){
17            System.out.println("等待客户的连接...");
18            //2. 开始ServerSocket实例的监听, 等待客户端的连接
19            //3. 如果有客户连接, 就获得了客户的套接字(Socket)实例
20            Socket s = ss.accept();
21            System.out.println("客户连接成功:"+s.getInetAddress());
22            //4. 为这个客户(Socket) 创建一个服务线程
23            new Service(s).start(); //派发服务线程, 处理客户的服务
24        }
25    }
26 }

```

```

29     }
30 }
31
32 /*成员内部类*/
33 class Service extends Thread{
34     Socket s;
35     public Service(Socket s) { this.s = s; }
36
37     public void run() {
38         try{
39             //in 代表客户端送来的信息
40             InputStream in = s.getInputStream();
41             //out 代表向客户端传送到信息
42             OutputStream out = s.getOutputStream();
43
44             //1. 服务器向客户端写信息
45             out.write("你吃点啥?\n".getBytes());
46             out.flush();
47
48             //2. 循环扫描客户端写回到服务器的信息
49             // 服务器端判断并回复
50             Scanner s = new Scanner(in);
51             while(true){
52                 String str = s.nextLine().trim();
53                 if(str.equals("粗面")){
54                     out.write("没有!\n".getBytes());
55                     out.flush();
56                 }else if(str.equals("包子")){
57                     out.write("有!给你\n".getBytes());
58                     out.flush();
59                     break;
60                 }else{
61                     out.write("你说啥?\n".getBytes());
62                     out.flush();
63                 }
64             }
65             s.close();
66         }catch(Exception e){
67             e.printStackTrace();
68         }
69     }
70 }

```

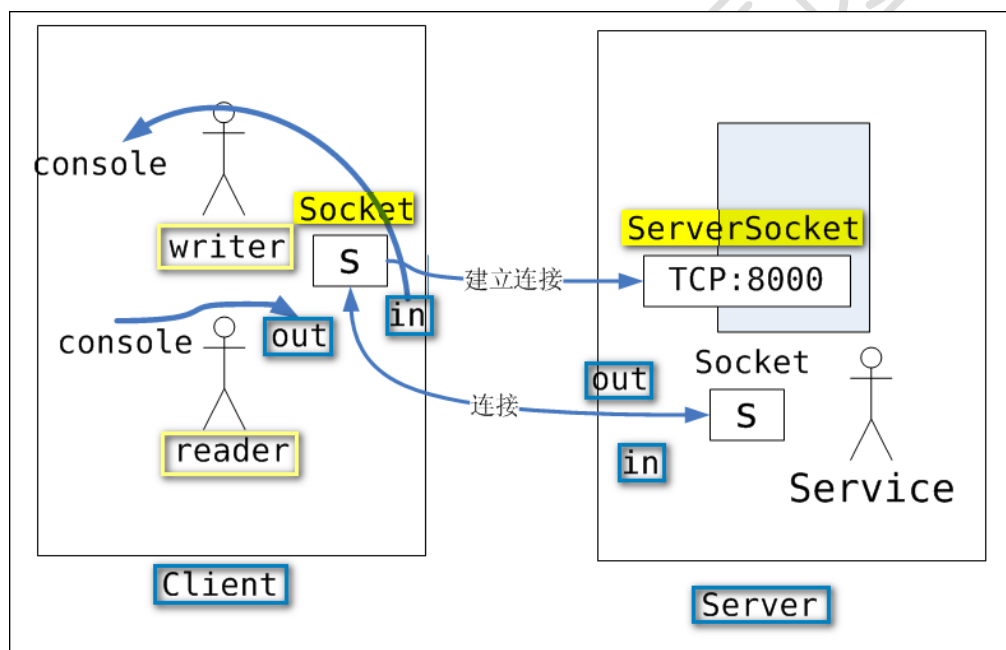
注：

- ✓ **ss.accept()方法**是 IO Block 方法，调用该方法会挂起当前线程，直到等到客户连接（IO 过程）的完成，一旦有客户连接进来就返回
- ✓ **Socket 实例 s** 代表客户端
- ✓ **s.getInetAddress()**，获得客户的 IP 地址

2.2. Socket 编程_TCP 客户端 **

实现步骤

- 1) 创建 Socket 实例, 连接到服务器端, 成功创建 s 就表示连接到了服务器
`Socket s = new Socket("host", port)`
- 2) 客户端 Socket 与服务器端 Socket 对应, 都包含输入、输出流
 客户端的 `s.getInputStream()` 连接于服务器 `s.getOutputStream()`
 客户端的 `s.getOutputStream()` 连接于服务器 `s.getInputStream()`
- 3) 使用线程处理网络流



【案例】TCP 客户端编程演示


```
ClientDemo.java X
1 package corejava.day14.ch01;
2 import java.io.IOException;
3
4
5 public class ClientDemo {
6     public static void main(String[] args)
7         throws IOException{
8         ClientDemo client = new ClientDemo();
9         client.open();
10    }
11
12    public void open() throws IOException{
13        //1. new Socket()会发起向localhost:8000的连接
14        //如果连接成功就会创建Socket实例s, 不成功会抛出异常
15        Socket s = new Socket("localhost", 8000);
16
17        //in代表服务器到客户端的流
18        InputStream in = s.getInputStream();
19        //代表客户端到服务器的流
20        OutputStream out = s.getOutputStream();
21
22        //2. Reader线程
23        // 负责将从控制台读取的信息写到服务器端(out)
24        new Reader(out).start();
25        //3. Writer线程
26        // 负责将从服务器读取的信息(in)写到客户端控制台
27        new Writer(in).start();
28    }
29
30    /* 成员内部类: Reader线程
31     * 负责将读取控制台的信息写到服务器端(out)
32     */
33    class Reader extends Thread{
34        OutputStream out;
35        public Reader(OutputStream out) {
36            this.out = out;
37            setDaemon(true);
38        }
39        public void run(){
40            Scanner s = new Scanner(System.in);
41            try{
42                while(true){
43                    String str = s.nextLine();//读取控制台
44                    out.write(str.getBytes());//发送到服务器
45                    out.write('\n');
46                    out.flush();
47                }
48            }catch(IOException e){
49                e.printStackTrace();
50            }
51        }
52    }
```

```

53     }
54 }
55 }
56 /* 成员内部类: Writer线程
57  * 负责将从服务器读取的信息(in)写到客户端控制台
58  */
59 class Writer extends Thread{
60     InputStream in;
61     public Writer(InputStream in) { this.in = in; }
62
63     public void run() {
64         try{
65             int b;
66             while((b = in.read())!=-1){
67                 System.out.write(b); //控制台会处理编码问题
68             }
69         } catch (Exception e) {
70             e.printStackTrace();
71         }
72     }
73 }
74 }

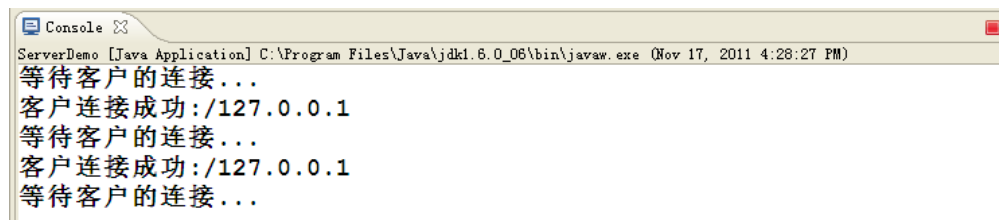
```

注：

- ✓ "localhost" 是服务器的域名/IP, localhost 在 IP 协议中是当前计算机自身的域名, 会映射到 127.0.0.1
- ✓ 第 40 行 因为服务器会主动地关闭连接, 将客户端 Reader 设置为客户端后台线程, 前台线程 Writer 结束后, Reader 即被杀掉
- ✓ 第 48 行 out.write('\n')代码的作用
是因为服务器端 (ServerDemo 第 52 行), String str = s.nextLine().trim()以回车符判断是否接收一行
- ✓ 第 49 行 不要忘记 flush()
- ✓ 第 67 行 控制台会处理编码问题

执行结果 (可同时开多个客户端)

● 服务器端获得两个连接



```

ServerDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Nov 17, 2011 4:28:27 PM)
等待客户的连接...
客户连接成功:/127.0.0.1
等待客户的连接...
客户连接成功:/127.0.0.1
等待客户的连接...

```

- 客户端 1 (连接服务器端命令 : telnet localhost 8000)

```
C:\> Telnet localhost
你吃点啥?
粗面
没有!
```

- 客户端 2 (连接服务器端命令 : telnet localhost 8000)
程序执行结束即断开连接

```
C:\WINDOWS\system32\cmd.exe
你吃点啥?
粗面
没有!
粗面
没有!
大饼
你说啥?
大饼
你说啥?
粗面
没有!
包子
有!给你

失去了跟主机的连接。
D:\>
```