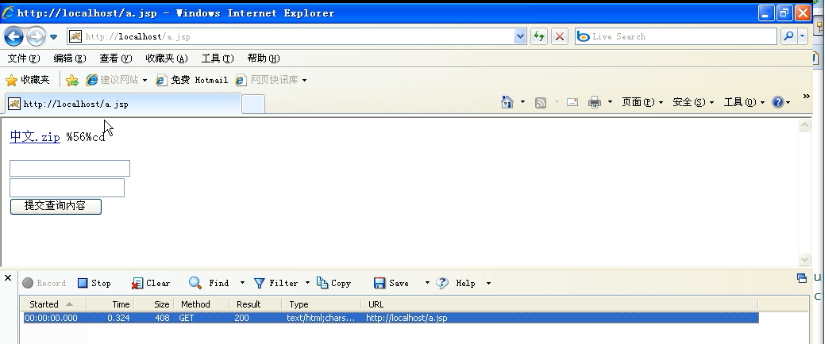
HttpClient---- HttpServer

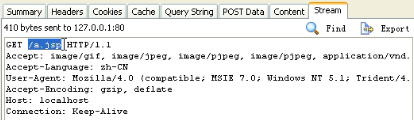
服务器和浏览器之间的交互

浏览器和服务器之间的对话要遵循一定的格式 这就是Httpxie协议

安装一个Http Watch



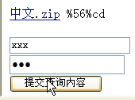
这个是浏览器发出去的请求的信息 也就是浏览器对服务器说的话





GET是获取资源的方式

/a.jsp是服务器的资源







用户名和密码

----- GET是获取 POST是提交数据 DELETE是删除资源



这个称谓请求行



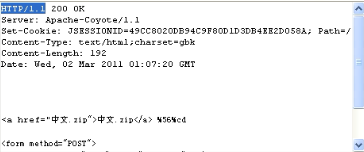
这个称为请求头 key-value ---->

这些都是事先规定好的 浏览器和服务器都能识别这个KV



浏览器和服务器从请求行开始说 一直说到最后一个空白行+一个回车 就算是结束了

------ 服务器回复的结果



--------------------------------

服务器回复的是协议也是Http/1.1

 这个是服务器响应信息的结果



这些是一些响应头 代表不同的意思

【



左边是request 右边是response

记忆办法 就是从左边念到右边 左边和右边通过使用的协议进行连接

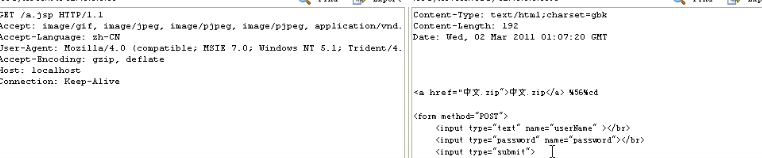
=🡺 左边request以http协议/版本 结束 就是 右边response响应以http协议/版本开始

左边是请求的内容 ---- 那么右边的响应一定要告诉你这个请求 我给的结果总体是怎么样的 也就是 左边是请求的资源 那么右边一定要根据这个资源给左边一个请求一个总体的答复】

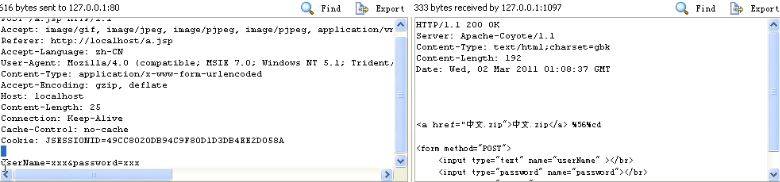
这个就是响应消息头



这样 根据这个响应信息的长度 浏览器就按照这个长度进行读取 多多余的字节不会继续读下去



------ 下面是post请求



左边是post请求 请求头结束之后 是一个空行 这个和get请求一样 但是

Post请求在这个空行之后 会跟上请求的数据

【post英语中是投递的意思

Post一定要有投递的信息 也就是投递的数据】

GET请求是不带内容的 POST请求方式是带内容的



注意 POST带有数据 这样 POST请求头有一个kv就是Content-length是25

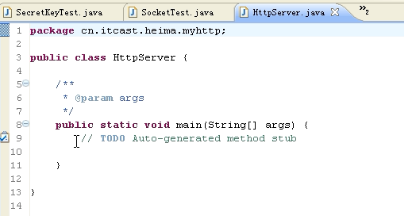
那么 你的后面的数据如果超过25 服务器也不会读下去了 他会认为是下一次请求的信息

-------- POST带有Content-length 但是GET没有这个请求头

============== 下面做Http服务器

Http服务器实际上就是Socket程序

新建一个工程是MyHttp



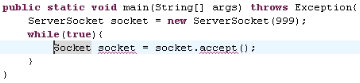
首先要在一个Socket上面进行监听

有一种可能就是 服务器监听的端口号起不来 这个端口

被别人占用了 所以 这个时候 要给用户一个错误提示

----- 这里面就节省了

这里面仅仅写核心代码



这样是不停的为客户服务 所以这个是循环

但是 如果有多个客户 此时 这个主线程为了保证更多的请求都能够被响应 不要纠缠对客户请求的服务 所以 这个时候 应该立刻回来

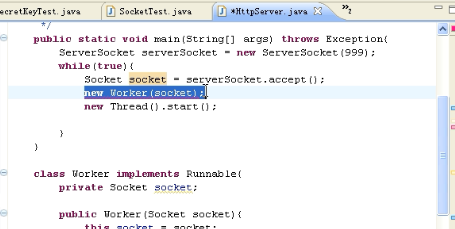
此时 另起一个单独的线程 与对方进行交流

拿着这个socket与对方交流

现在封装一个代码 是一个单独的线程

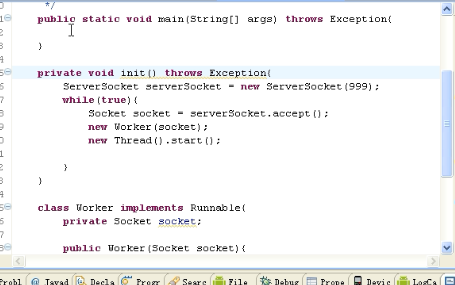


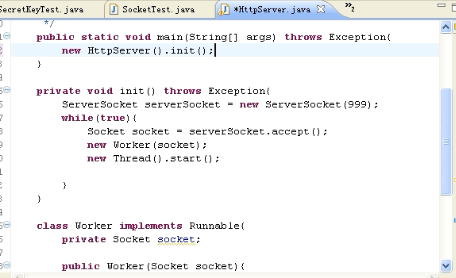
静态方法不能创建内部类的实例对象



修改如下

那就不让这个内部类的对象在main这个静态方法中被创建 放到一个普通的非静态方法中 然后main中进行实例化调用



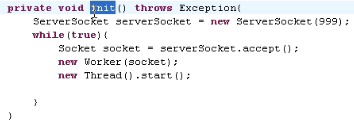


【内部类的对象是可以访问外部类的成员的【这个道理的原因就是内部类存在的最最根本的意思 就是 内部类可以不用倒包 就方便访问外部类的成员 从这个最最根本的地方往外面推】

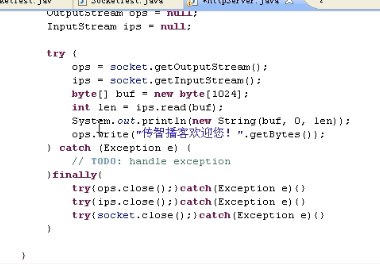
---- 隐含着外部类必须有this 才能访问他的成员变量

---- 这样 如果外部类没有this 那么内部类就不能访问外部类的成员

---- main是外部类的静态方法 ----- 这样 在main中是没有this的 这样 就不能实例化内部类的成员 因为这样实例化出来的内部类对象 是不能访问外部类的成员的】



现在构建这个Worker的代码



【Socket就相当于代表客户端和服务端 贮存在网络Connection上面的两个信号点

双方是有Session的

这样的话 服务端的socket首先要接受客户端的socket的信息 所以 要读 所以 有 socket.getInputStream ----- 获取到请求的信息之后 ------ 使用socket.getOutputStream把服务端自己的响应打出去 交给客户端 所以 使用socket.getOutputStream().write(xxxx)】

这个是昨天的内容 --- 这里面的内容是和服务端瞎说一气 没有什么规则（也就是没有遵循HTTP协议）

今天是采用http协议进行通话

首先解析浏览器的第一行



仅仅处理这里面的每一个部分

一直读到 \r\n

现在如何读取第一行？

是一个一个字节来读取么？

现在这种请求还好 没有二进制文件

传过来的这种 都是自负文本 可以利用读字符文本的类才处理这行

但是 有时候有二进制文本的时候

是不能使用readLine ---- readLine是转成字符

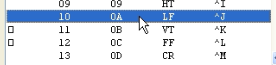
【二进制文件和纯文本 最大的区别就是 一个以\r\n做换行符 一个\r\n是没有换行这个意义的】

---- 所以 不能使用Reader 要使用InputStream

---- 使用 InputStream要读到\r\n

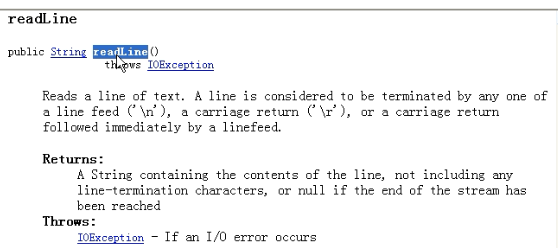
这里面都是纯文本

\r --- ascII是13 \n的ascii是10

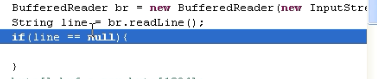


如果是图片 什么都有可能 这样写读取一行的代码很难

我们现在使用一个类BufferedReader



我们这里面使用BufferedReader来进行读取



为null的时候 是读到什么？我们看一下文档



此时 表示没有数据了 这样说明Socket关闭了

Socket开着 就说明 可能有数据

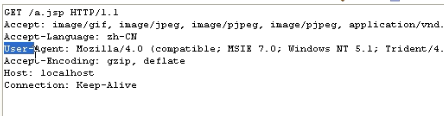
所以 返回null 说明客户端关闭了 流结束了

不是读完眼前的数据了

 读到””说明数据已经读取完毕了

再有 说明是下一次的请求的数据了

【】



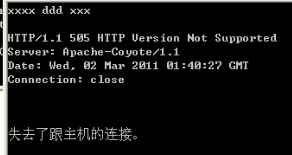
最下面是一个空行 表示请求信息全部读完了

模仿服务器 telnet localhost 80



第一次回车没有处理

再来一次回车 才开始处理

 ----  协议开头的一定是HTTP Response的响应信息

所以 我的代码不是在

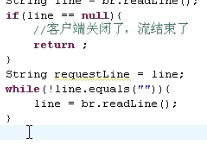
If(line == null){

Return;

}

String requestLine = line;之后处理

而是在while(){}读取完所有的 再作处理的



【因为我们处理的仅仅是针对请求行 也就是第一行

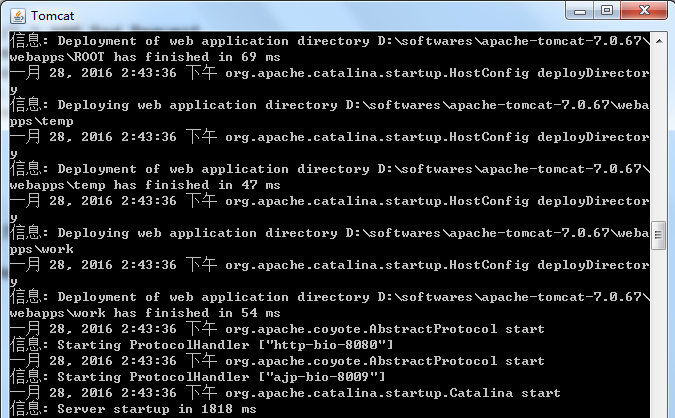
GET /a.jsp HTTP/1.1 这一行

而不是

】

【

我现在本地启动了一个Tomcat



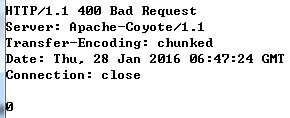
然后 我使用客户端工具 telnet localhost 8080

进入其中 输入了 GET /a.jsp HTTP/1.1 然后第一次回车



此时 还是停留在输入页面 还没有看到服务器的响应

再敲一次回车 就得到了服务器的响应

 ---- 这说明 第一次回车 表示服务器还在等待你继续输入其他的请求信息 但是 如果 你这一行不输入信息 然后 还是直接回车 就形成了刚才非常经典的request的请求画面



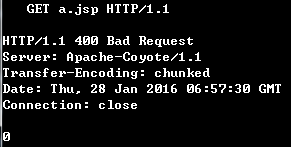
这一个 也就是 第一次回车 还不确定你回车之后 是否是还要继续输入信息

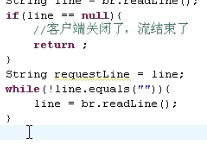
但是 你第一次回车 然后又第二次回车 表示 已经输入了一个空行 同时又是GET请求 所以 这个请求的所有信息就结束了

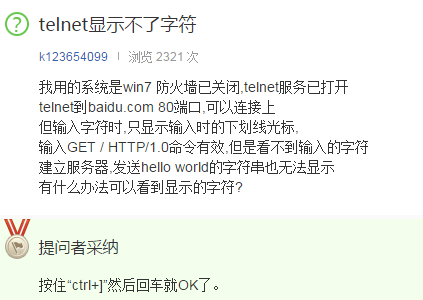


 如果第一次回车之后 还是输入了内容 再回车的时候 还是要求你继续输入内容

此时 如果连续输入两次回车 就会出现空行 这样 服务器就会进行应答

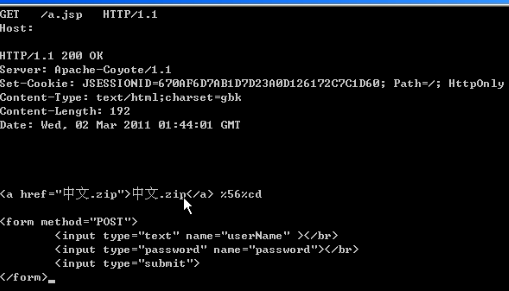
 ====>**这说明的是服务器要等待你把所有的信息都输入完成之后 才进行处理 而不是 输入一行处理一行**

 这样while循环中 就应该是中间请求消息头 while循环到line.equals(“”)的时候 退出了 也就是while条件不满足 也就是 真的读到这个请求中的最后一个空行 才开始做真正的处理

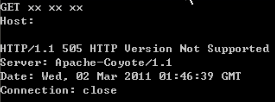
】

【null说明读不到东西了 所以推断出 客户端退出了】

再次模仿



测试了 之间是多个空格



可以看到响应之后

43:55