

Http协议上网 都是明文传送 电信号一旦截获 就是明文 不安全

现在想把B和S之间传递的数据 加密以后传输 这时候 首先

首先服务器向浏览器出示证书【就是公钥】

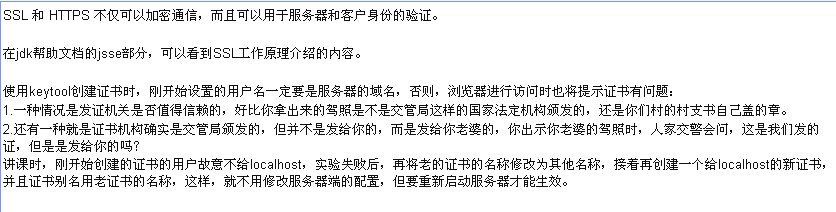
然后浏览器就把数据数据进行公钥加密 然后传递给服务器

----- 大家知道 非对称加密很慢 效率低

浏览器如何做？收到公钥后 产生一个密码 123457 然后使用服务器的公钥一加密 送给服务器 然后服务器知道密码是123457 --- 然后bs之间再传递数据 就是使用这个123457进行对称加密 也就是密码是非对称的形式进行加密 然后 双方使用这个相同的密码对传递的数据进行对称加密 这样效率高

这个也就是socket编程上 不仅仅是http协议

只有网络通信上用加密用的更多



这个证书 不需要我们自己构建 底层API去构建

要用这个证书 就要Socket编程

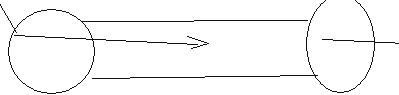


SSL就叫安全的Socket层

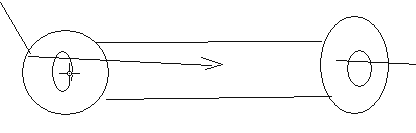
以前的Socket编程：

互联网上面的两端

通过Socket进行数据传递 但是这个数据是明文



现在的做法



Socket不直接发数据 把明文数据直接传递给外面包装的一层 这一层进行了加密

然后 传递数据出去到目的地 这个目的地接到数据之后 外面的SSL就会对数据进行解密 解密之后 再吧明文数据传递给接收方的Socket 这样对编程人员是透明的

这样 这两个SSL层的加密和解密就需要出示服务器的数字证书 这个数字证书就是包含了公钥

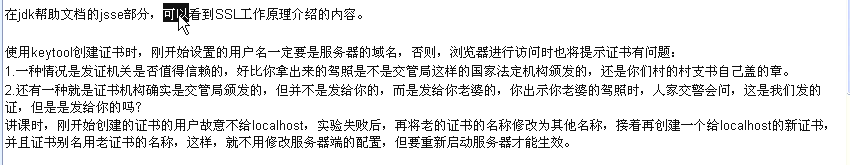
浏览器用这个公钥加密一个密码 服务器没有疑问 之后就是使用对称加密进行数据传递

后来大家都觉得这个SSL很好 国际组织就把这个进行了采纳 定了一个规范

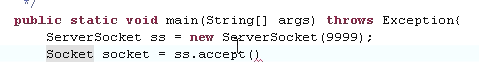


TLS传输层安全标准

传输层是安全的 应用层的编码没有变化

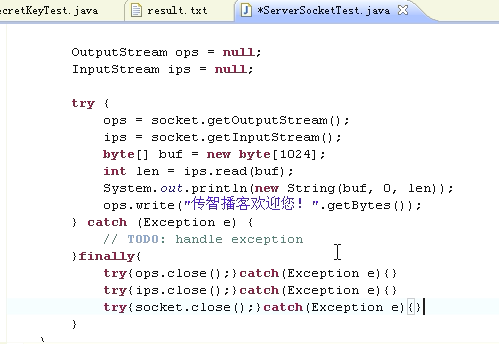


现在构建这样的一个服务端的程序



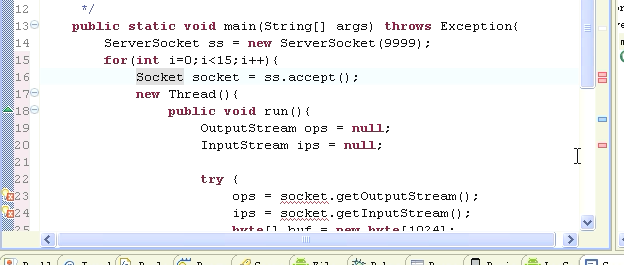
有了这个Socket 就可以和对方进行数据的传递了

这里面写点非专业的代码

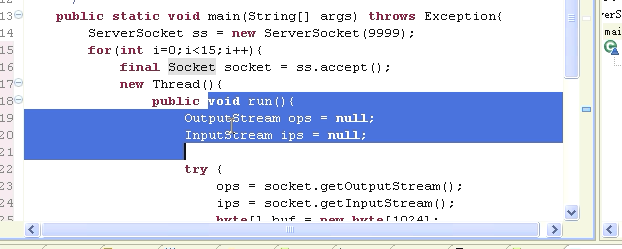


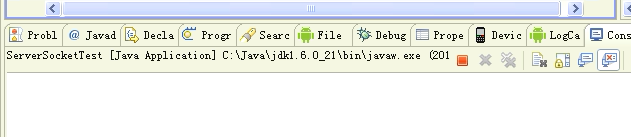
修改为不停的接收客户端的请求的代码

一旦收到了 我就启动线程 来单独执行响应代码



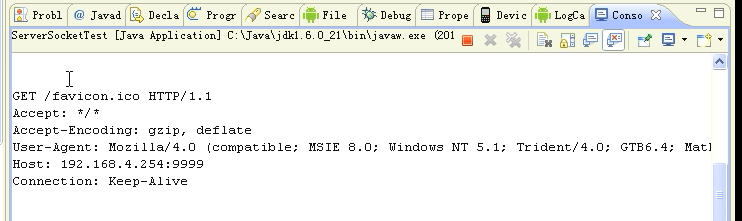
修改





现在浏览器来进行访问 9999这个端口

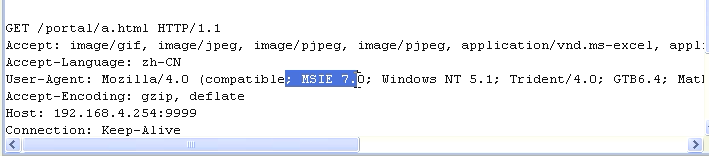


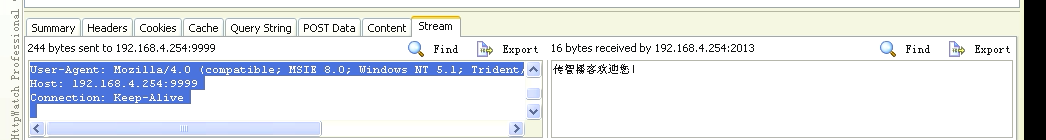


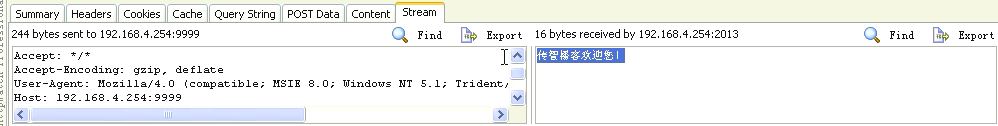


用的协议是HTTP1.1 可以接受的格式 是下面打印的



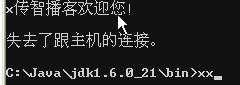






客户端懒得写 直接使用浏览器替代就可以

也可以不用浏览器 用telnet工具

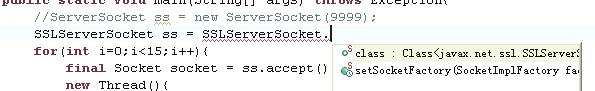


现在 请求修改成SSL的 客户端请求的时候 就给用户出示一个证书 等等 这个客户端代码很麻烦

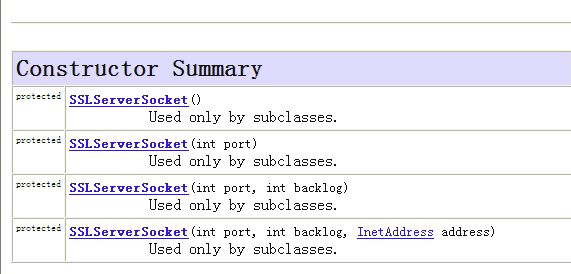
现在 修改服务器端



SSLServerSocket是ServerSocket的子类

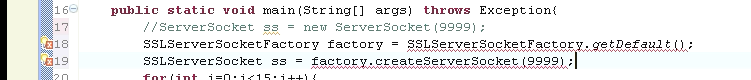


无法产生对象 继续读文档

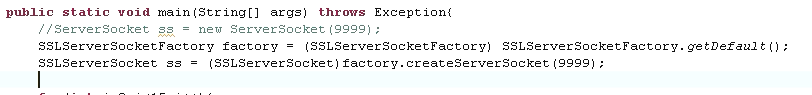


全部是保护性的

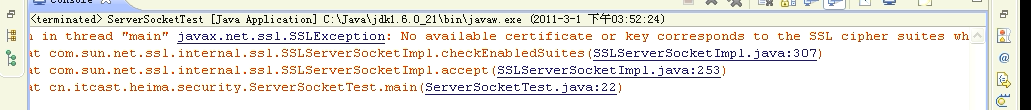
那就看看有没有工厂Factory或者Generator



进行强转



这时候 运行一下



给出异常 没有可获得的证书 或者key

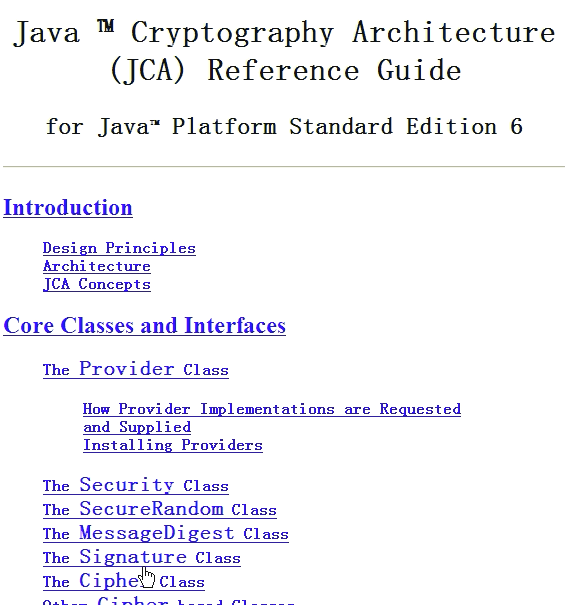


要给出KeyStore的位置 哪一个证书

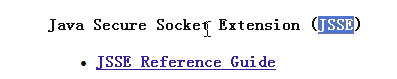
安全分成了两块



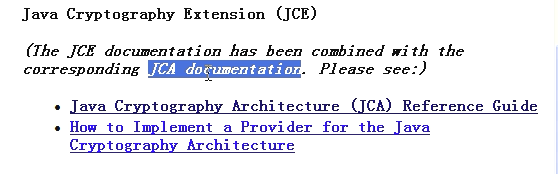
安全刚才的技术 是Java的加密术的参考指南---JCA



刚才学的叫 JSSE Java安全Socket扩展

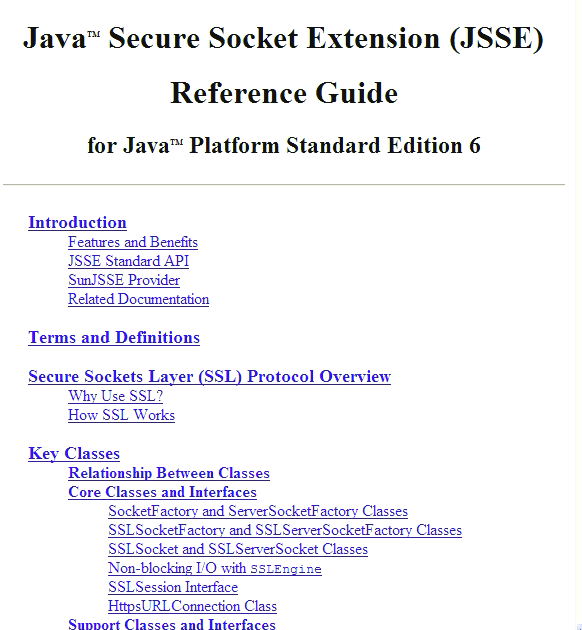


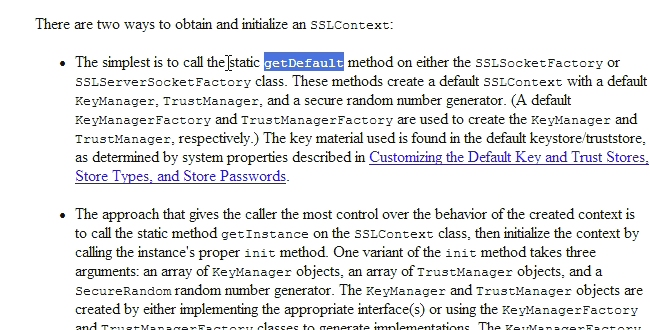
还有一个是JSSE+JCA = JCE ---- Java加密扩展

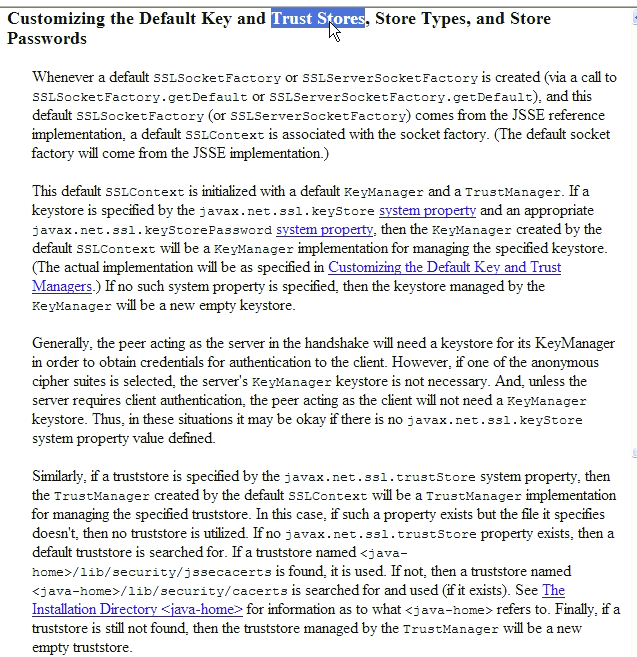


Java研究过两块：熟练掌握Java安全体系中的JCA和JSSE部分【编写简历的时候】

其他技术暂时不研究了

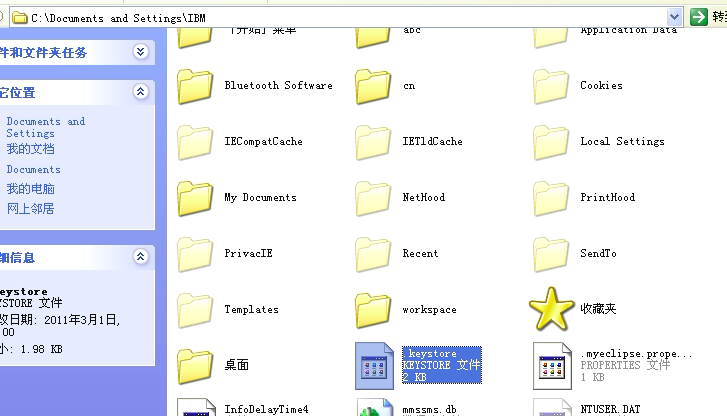






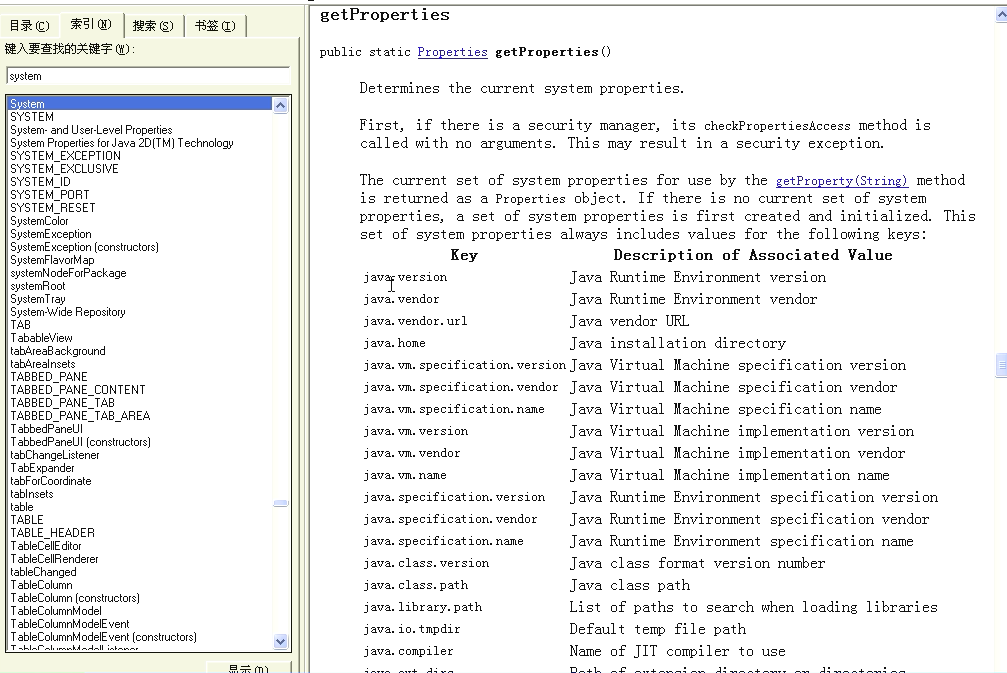
文档慢慢去读

要成功运行 要知道系统的keystore的位置



介绍一下Java系统变量

这里面通过系统变量来取代硬编码



Java虚拟机上面有很多属性 也就是很多变量

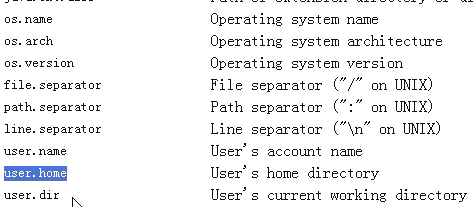
系统属性 就是操作系统上面的变量

任何一个程序都会获取操作系统上面的变量

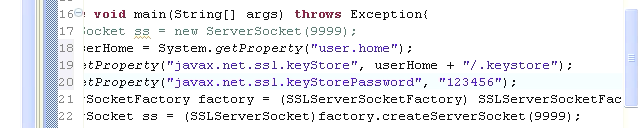
Windows上面也有系统属性

设置系统属性 就是多个程序都可以设置这个变量

系统的值一改 就都改了



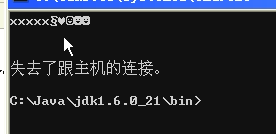




设置好这个环境变量 就可以运行了

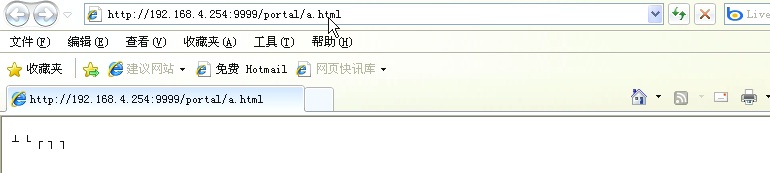
再次去访问





不可以

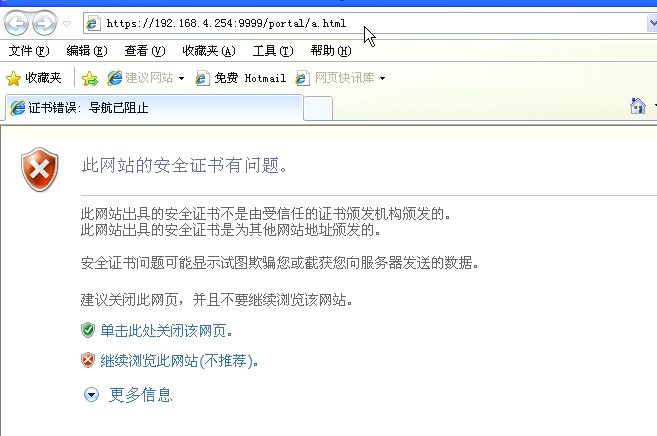
我们用浏览器来替代



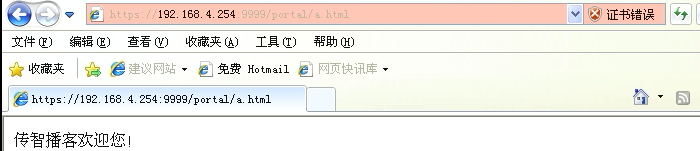
错误的原因 必须使用https来访问

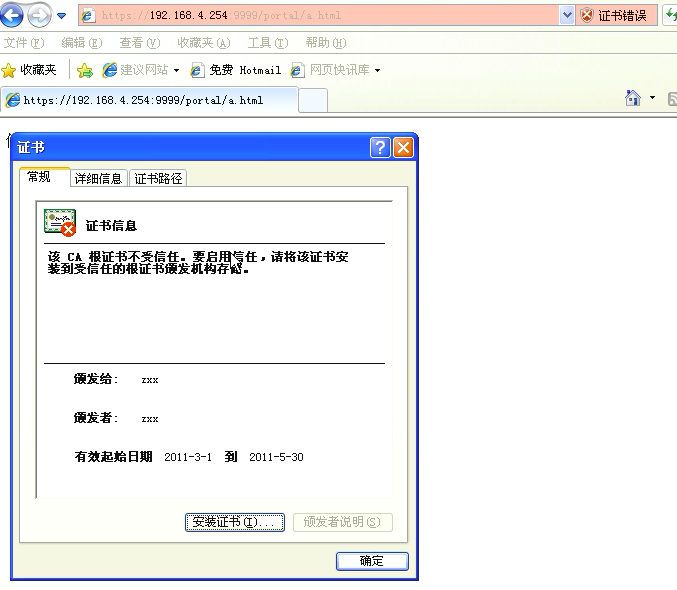
这种情况才会出示证书 表示用加密的方式

用http情况 没有加密的情况



出示了一个证书

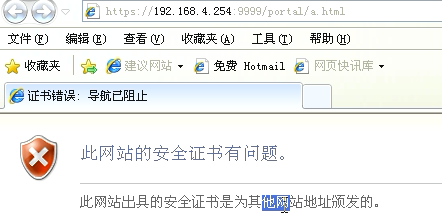
点进去：



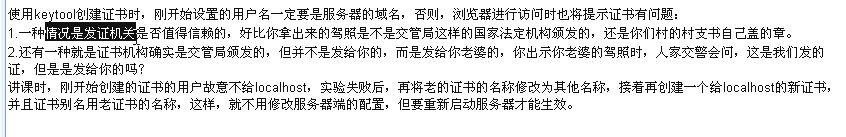
点击安装



【可以看到 这个证书的颁发者是xxx 颁发给的也是zxx】

再访问  


两种错误

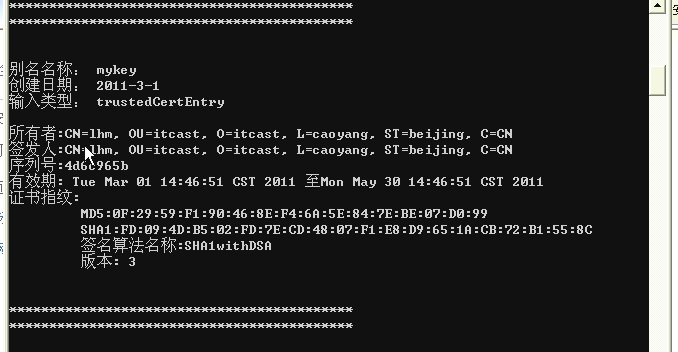


第一种是没有盖章的证书

第二种 盖章的 但是 不是给你发的证书

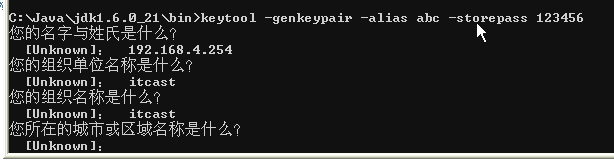
刚才服务器的地址 是的 但是 刚才的证书是办法给zxx的





所有者的CN是lhm

再做一个证书【因为B和S同一台机器 所以 我在本地做一个证书 就相当于服务器做了一个新的证书】

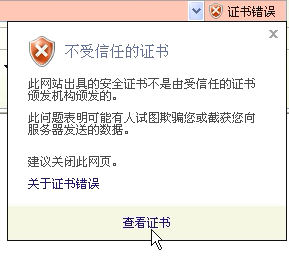
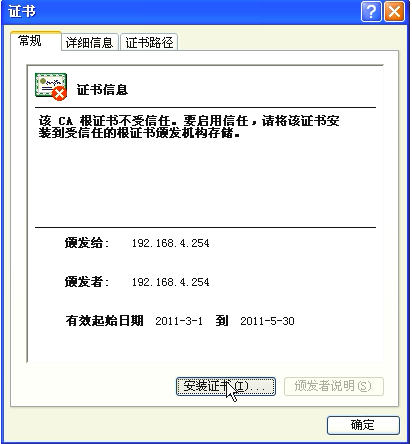


这个名字就是ip地址

重启程序

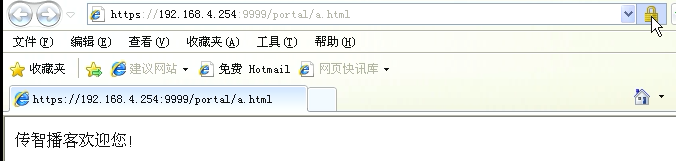


第一次是不受信任的错误

这样这个证书的颁发者 和 颁发给都是192.168.4.254这个ip

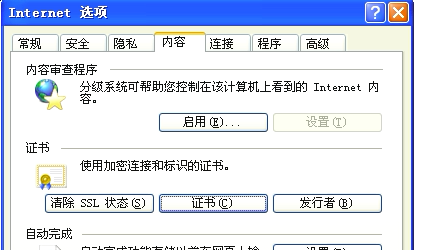
重启访问



这儿证书确实是发给这个IP地址的

这个是安全套接字编程

浏览器每次都要从自己安装的证书中看

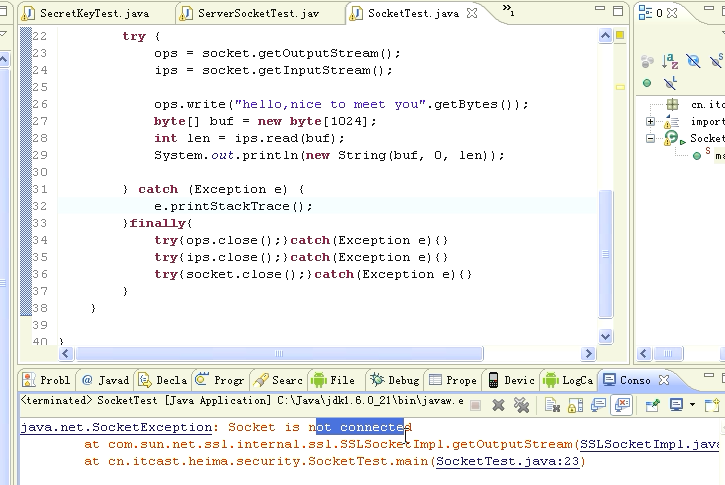


所以 Java的客户端也有一个安装证书的地方

现在构建一个Java的客户端 对服务端的证书进行校验

先给服务器发信息

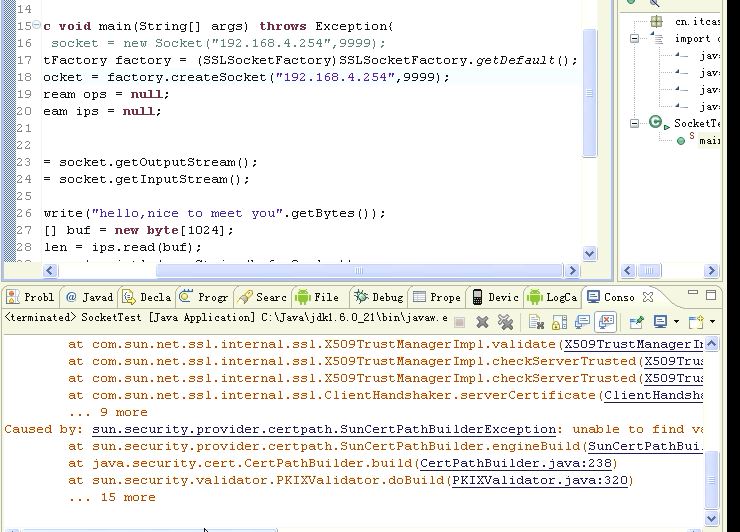
要使用和服务端一致的安全套接字

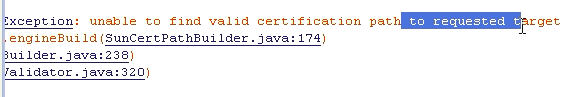


给出的信息是Socket没有连接上

报的信息不太好

IP和端口号没有写



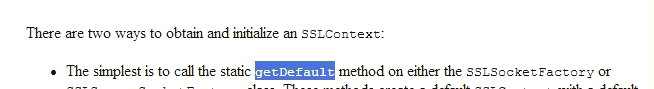


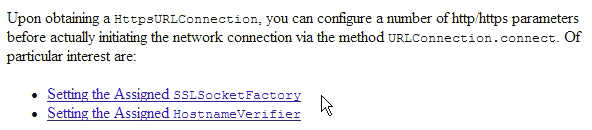
这个上面给出的是不能够找到一个证书 没有法去比对

所以 要有你的证书

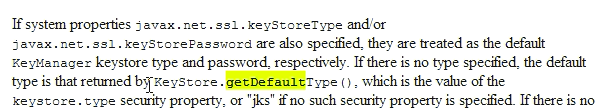
不知道去哪里获取证书 不知道到哪里获取

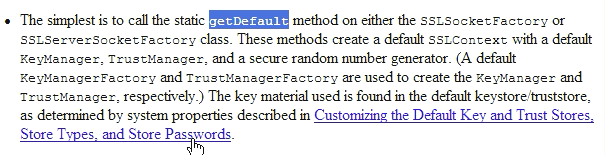
刚才读文档的时候

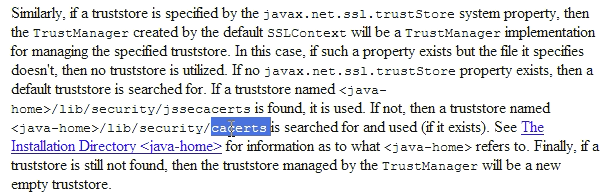


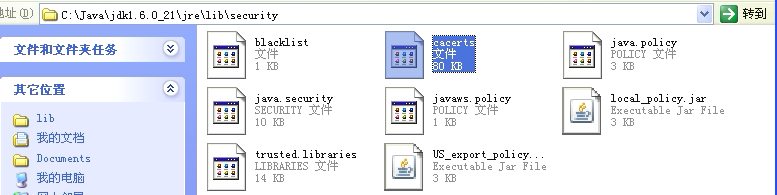


默认的方式 要去哪个地方读东西 设置getDefault()









这个里面存储了信任别人的证书

我的Java程序默认到这里面读取

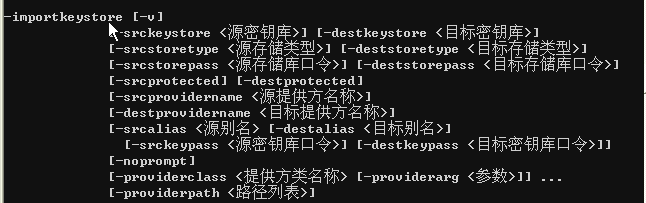
如果不想在这里面读

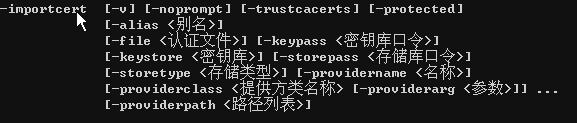


现在 把证书 导入到刚才默认的目录下面



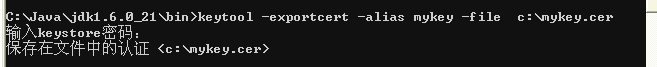
导进去





有导入证书和keyStore



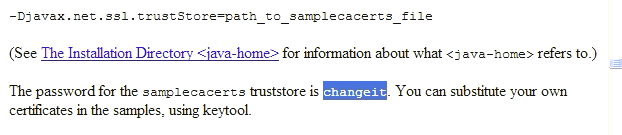


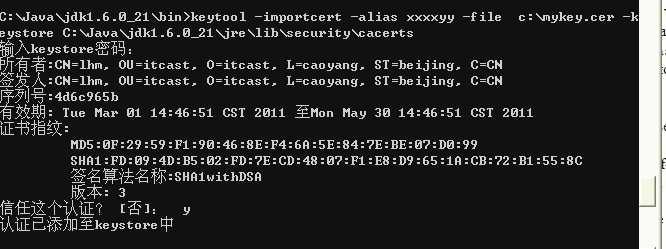
先导出证书

现在关联一个别名

再导入



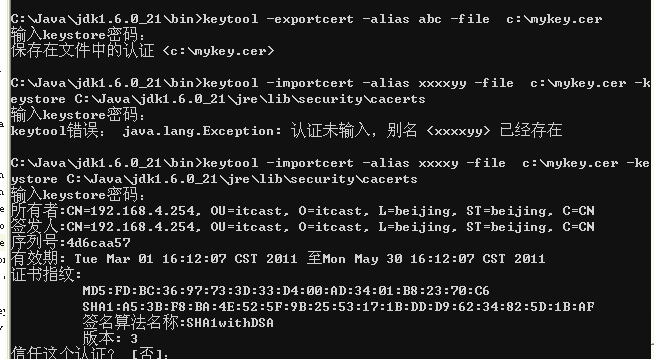




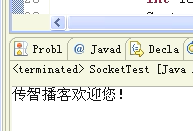
但是 这个mykey是给lhm的

不对 应该到处 abc这个证书



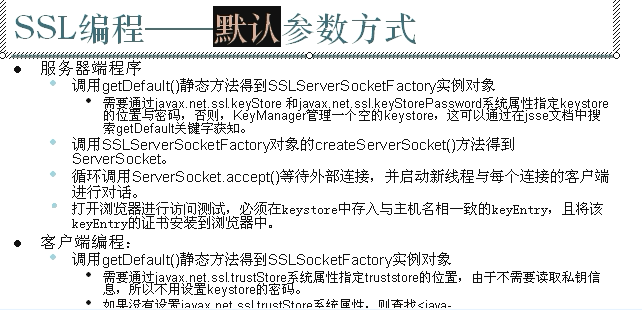


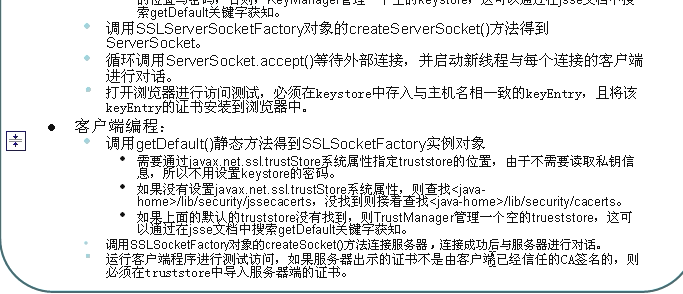
然后再运行这个程序



要和你通信 首先要获得你的证书

以上是SSL编程的默认参数方式



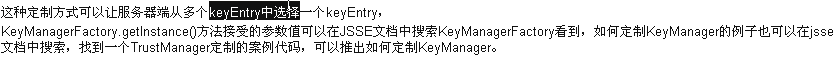




只有信任了 才能对话



定制SSL：所用到的文件



刚才KeyStore有三个key 要获取哪一个？

并且都是签发给你这个地址的 领哪一个证书？

19：27