HTTP以及HTTPS通信协议的原理分析

1、什么是HTTP

1.1 HTTP与OSI模型

HTTP全称：HyperText Transfer Protocol （超文本传输协议），是一个基于请求与响应，无状态的，应用层的协议，常基于TCP/IP协议传输数据，互联网上应用最为广泛的一种网络协议,所有的WWW文件都必须遵守这个标准。设计HTTP的初衷是为了提供一种发布和接收HTML页面的方法。想了解http，就需要了解TCP，IP协议。因为http是基于TCP，IP层上面的。如下图*OSI*（Open System Interconnection）七层模型所示：

以前总是记不住这七层模型，但其实理解之后自然就记住了。
整张图应该从下边向上看，假设别人想发一条信息给我的电脑。首先是通过网络把信息传导到的电脑，但不能说我的电脑对于所有的信息都接受，我会判断信息，之后在处理信息，这些就是七层模型做的事情。一步一步分开来看 ，并用打电话这一创建的例子来对应：

1. 物理层 包括物理连网媒介，实际上就是布线、光纤、网卡和其它用来把两台网络通信设备连接在一起的东西。它规定了激活、维持、关闭通信端点之间的机械特性、电气特性、功能特性以及过程特性。 **（这就相当于电信公司的信号发射塔，接收信号而已）**
2. 数据链路层 数据链路层主要作用是控制网络层与物理层之间的通信。它保证了数据在不可靠的物理线路上进行可靠的传递。它把从网络层接收到的数据分割成特定的可被物理层传输的帧，保证了传输的可靠性。 **（相当于发射塔让接受的信号更稳定，方便下一层的解读）**
3. 网络层 很多人经常混淆2层和3层的相关问题，简单来说，如果你在谈论一个与IP地址、路由协议或地址解析协议（ARP）相关的问题，那么这就是第三层的问题。网络层负责对子网间的数据包进行路由选择，它通过综合考虑发送优先权、网络拥塞程度、服务质量以及可选路由的花费来决定从一个网络中两个节点的最佳路径。另外，它还可以实现拥塞控制、网际互连等功能 **（相当于我们**

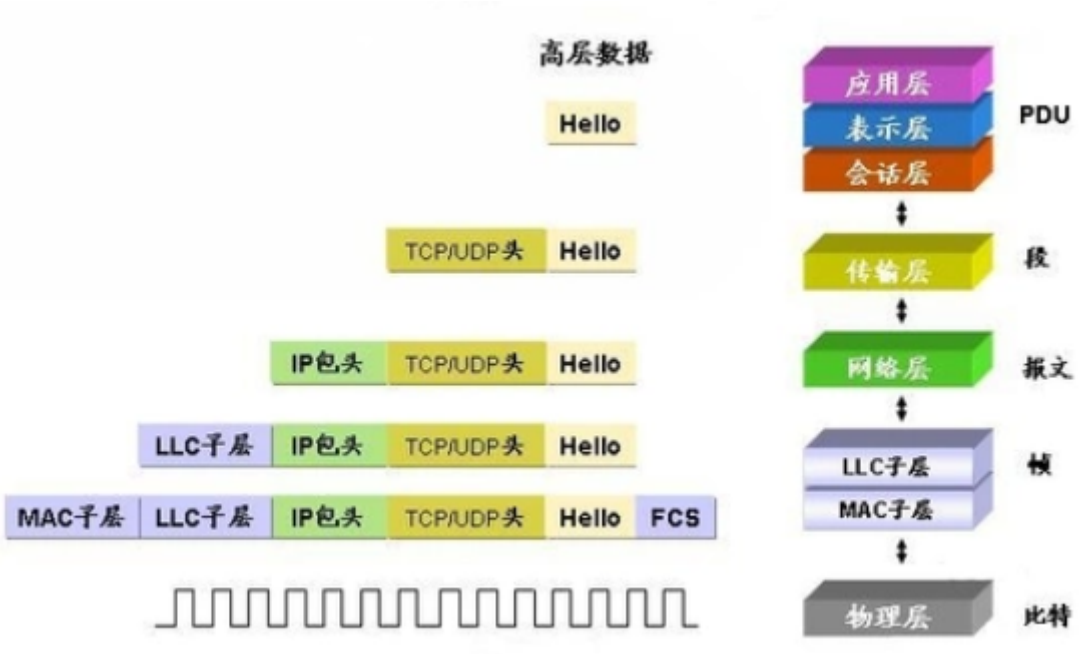
打一个长途电话，当前连接的发射站不能直接传递给另一用户，需要先传递给附近的发射站）
到这里为止，其实信息还没有传递到我们的电脑，可以理解为信息在路由网络间传递。

4. 传输层 是OSI模型中最重要的一层，它是两台计算机经过网络进行数据通信时,第一个端到端的层次，起到缓冲作用。当网络层的服务质量不能满足要求时，它将提高服务，以满足高层的要求；而当网络层服务质量较好时，它只需进行很少的工作。另外，它还要处理端到端的差错控制和流量控制等问题，最终为会话提供可靠的,无误的数据传输。 **（相当于我们找到了最后的信号站，可以直接发送信号给通话者，但是这层会做一些数据的整合等工作）**
5. 会话层 会话层负责在网络中的两节点之间建立和维持通信，并保持会话获得同步，它还决定通信是否被中断以及通信中断时决定从何处重新发送。 **（相当于两个信号塔建立了连接，等待信息的最终收发）**
6. 表示层 表示层的作用是管理数据的解密与加密，如常见的系统口令处理，当你的账户数据在发送前被加密，在网络的另一端，表示层将对接收到的数据解密。另外，表示层还需对图片和文件格式信息进行解码和编码。 **（相当于发射塔给手机发送一些压缩的信息，我们手机接收到后解码）**
7. 应用层　简单来说，应用层就是为操作系统或网络应用程序提供访问网络服务的接口，包括文件传输、文件管理以及电子邮件等的信息处理。应用层协议的代表包括：Telnet、FTP、HTTP、SNMP等。 **（这就是真正的收到信息，相当于我们听到了对方的声音，这次对话完成了）**

总结起来就是：
打电话信息发出(www） --->发射塔收到不稳定信息流（物理层） ---> 发射塔整理信息流（链路层）---> 发射塔转给离接收人更近的发射塔（网络层IP） ---> 发射塔向接收人发送信息（传输层TCP） ---> 通话建立（会话层） ---> 解码信息（表示层） ---> 发出声音给接收人（应用层HTTP）

以上是对支撑HTTP的网络七层的概括，我们需要理解为什么要在http层下面有那么多层，每一层的目的是什么。

明白了这些之后，问题来了，这七层是怎么知道自己应该怎么解读数据，而不是说TCP层解读HTTP层。这里面就是网络数据的封装和传递了：



从这个图中，我们可以看到每一层的数据都会被一个这层的头信息包裹。之后我们在接收到信息时再一层一层读取信息

1.2 TCP/IP协议

互联网的关键技术就是TCP/IP协议。两台计算机之间的通信是通过TCP/IP协议在因特网上进行的。实际上这个是两个协议：

TCP : Transmission Control Protocol 传输控制协议和IP：Internet Protocol 网际协议。

IP：计算机之间的通信

IP协议是计算机用来相互识别的通信的一种机制，每台计算机都有一个IP.用来在internet上标识这台计算机。 IP 负责在因特网上发送和接收数据包。通过 IP，消息（或者其他数据）被分割为小的独立的包，并通过因特网在计算机之间传送。IP 负责将每个包路由至它的目的地。

IP协议仅仅是允许计算机相互发消息，但它并不检查消息是否以发送的次序到达而且没有损坏（只检查关键的头数据）。为了提供消息检验功能，直接在IP协议上设计了传输控制协议TCP.

TCP : 应用程序之间的通信

TCP确保数据包以正确的次序到达，并且尝试确认数据包的内容没有改变。TCP在IP地址之上引端口（port），它允许计算机通过网络提供各种服务。一些端口号为不同的服务保留，而且这些端口号是众所周知。

服务或者守护进程：在提供服务的机器上，有程序监听特定端口上的通信流。例如大多数电子邮件通信流出现在端口25上，用于wwww的HTTP通信流出现在80端口上。

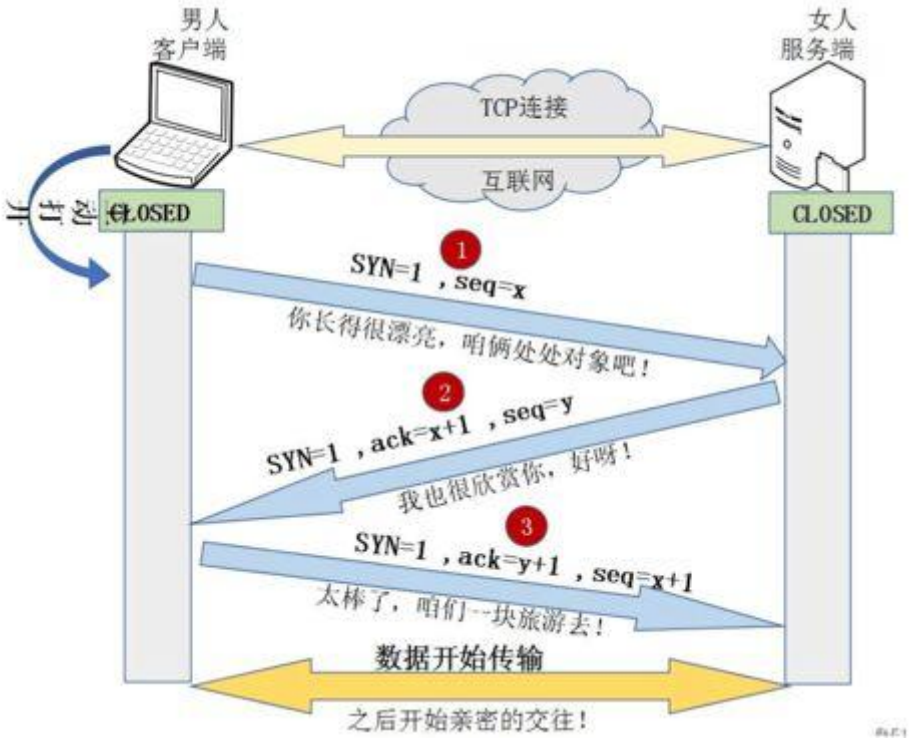
当应用程序希望通过 TCP 与另一个应用程序通信时，它会发送一个通信请求。这个请求必须被送到一个确切的地址。在双方"握手"之后，TCP 将在两个应用程序之间建立一个全双工 (full-duplex) 的通信，占用两个计算机之间整个的通信线路。TCP 用于从应用程序到网络的数据传输控制。TCP 负责在数据传送之前将它们分割为 IP 包，然后在它们到达的时候将它们重组。

TCP/IP 就是TCP 和 IP 两个协议在一起协同工作，有上下层次的关系。

TCP 负责应用软件（比如你的浏览器）和网络软件之间的通信。IP 负责计算机之间的通信。TCP 负责将数据分割并装入 IP 包，IP 负责将包发送至接受者，传输过程要经IP路由器负责根据通信量、网络中的错误或者其他参数来进行正确地寻址，然后在它们到达的时候重新组合它们。

PS：HTTP是基于TCP协议之上的，是应用层的协议，HTTP协议永远都是客户端发起请求，服务器回送响应

1.3 TCP三次握手（建立连接）



（1）男孩喜欢女孩，于是写了一封信告诉女孩：我爱你，请和我交往吧！;写完信之后，男孩焦急地等待，因为不知道信能否顺利传达给女孩。

（2）女孩收到男孩的情书后，心花怒放，原来我们是两情相悦呀！于是给男孩写了一封回信：我收到你的情书了，也明白了你的心意，其实，我也喜欢你！我愿意和你交往！；

写完信之后，女孩也焦急地等待，因为不知道回信能否能顺利传达给男孩。

（3）男孩收到回信之后很开心，因为发出的情书女孩收到了，并且从回信中知道了女孩喜欢自己，并且愿意和自己交往。然后男孩又写了一封信告诉女孩：你的心意和信我都收到了，谢谢你，还有我爱你！

女孩收到男孩的回信之后，也很开心，因为发出的情书男孩收到了。由此男孩女孩双方都知道了彼此的心意，之后就快乐地交流起来了~~

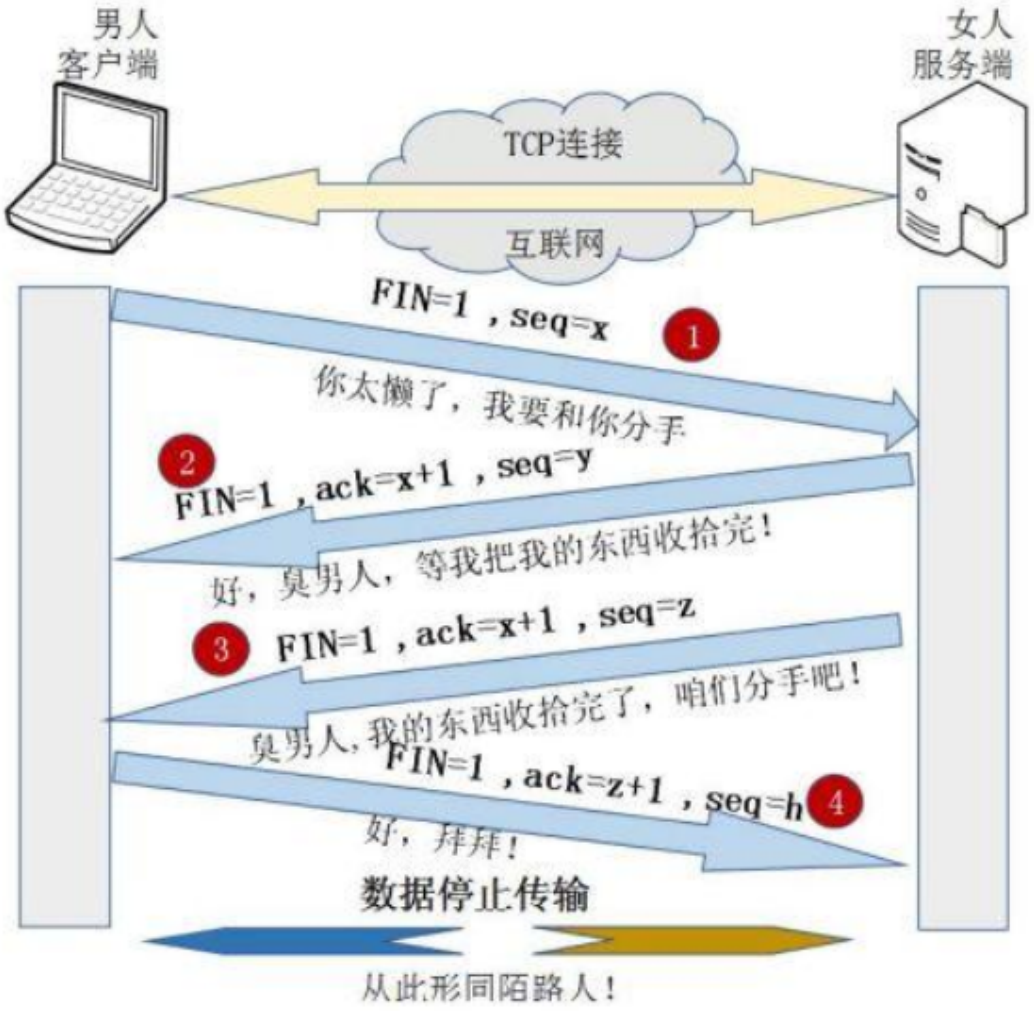
这就是通俗版的“三次握手”，期间一共往来了三封信也就是“三次握手”，以此确认两个方向上的数据传输通道是否正常。

PS：

seq--->表示的是我方（发送方）这边，这个packet的数据部分的第一位应该在整个data stream中所在的位置。

ack--->表示的是期望的对方（接收方）的下一次sequence number是多少

1.4 TCP四次挥手（释放连接）



"第一次挥手"：日久见人心，男孩发现女孩变成了自己讨厌的样子，忍无可忍，于是决定分手，随即写了一封信告诉女孩。

“第二次挥手”：女孩收到信之后，知道了男孩要和自己分手，怒火中烧，心中暗骂：你算什么东西，当初你可不是这个样子的！于是立马给男孩写了一封回信：分手就分手，给我点时间，我要把你的东西整理好，全部还给你！男孩收到女孩的第一封信之后，明白了女孩知道自己要和她分手。随后等待女孩把自己的东西收拾好。

“第三次挥手”：过了几天，女孩把男孩送的东西都整理好了，于是再次写信给男孩：你的东西我整理好了，快把它们拿走，从此你我恩断义绝！

“第四次挥手”：男孩收到女孩第二封信之后，知道了女孩收拾好东西了，可以正式分手了，于是再次写信告诉女孩：我知道了，这就去拿回来！

这里双方都有各自的坚持。女孩自发出第二封信开始，限定一天内收不到男孩回信，就会再发一封信催促男孩来取东西！男孩自发出第二封信开始，限定两天内没有再次收到女孩的信就认为，女孩收到了自己的第二封信；若两天内再次收到女孩的来信，就认为自己的第二封信女孩没收到，需要再写一封信，再等两天.....

倘若双方信都能正常收到，最少只用四封信就能彻底分手！这就是“四次挥手”。

1.5 为什么“握手”是三次，“挥手”却要四次?

TCP建立连接时之所以只需要"三次握手"，是因为在第二次"握手"过程中，服务器端发送给客户端的TCP报文是以SYN与ACK作为标志位的。SYN是请求连接标志，表示服务器端同意建立连接；ACK是确认报文，表示告诉客户端，服务器端收到了它的请求报文。

即SYN建立连接报文与ACK确认接收报文是在同一次"握手"当中传输的，所以"三次握手"不多也不少，正好让双方明确彼此信息互通。

TCP释放连接时之所以需要“四次挥手”,是因为FIN释放连接报文与ACK确认接收报文是分别由第二次和第三次"握手"传输的。为何建立连接时一起传输，释放连接时却要分开传输?

建立连接时，被动方服务器端结束CLOSED阶段进入“握手”阶段并不需要任何准备，可以直接返回SYN和ACK报文，开始建立连接。释放连接时，被动方服务器，突然收到主动方客户端释放连接的请求时并不能立即释放连接，因为还有必要的数据需要处理，所以服务器先返回ACK确认收到报文，经过CLOSE-WAIT阶段准备好释放连接之后，才能返回FIN释放连接报文。

所以是“三次握手”，“四次挥手”。

2、什么是HTTPS

2.1 https协议概念

讲完http，我们会发现，我们传递的信息在网络上经过那么多物理层的传输，保不准会被别人截获，而我们却一点也不知道。于是我们想到要加密我们传输的数据。因为只有http层信息是我们要的，所以在这一层下面加入一层来加密信息。这一层就是SSL层，同时我们如果想传给SSL层，我们用端口TCP-443，这个协议就叫HTTPS协议

HTTPS （全称：Hyper Text Transfer Protocol over SecureSocket Layer）超文本传输安全协议，是以安全为目标的 HTTP 通道，在HTTP的基础上通过传输加密和身份认证保证了传输过程的安全性 [1] 。HTTPS 在HTTP 的基础下加入SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。HTTPS 存在不同于 HTTP 的默认端口及一个加密/身份验证层（在 HTTP与 TCP之间）。这个系统提供了身份验证与加密通讯方法。它被广泛用于万维网上安全敏感的通讯，例如交易支付等方面

HTTPS是身披SSL外壳的HTTP。HTTPS是一种通过计算机网络进行安全通信的传输协议，经由HTTP进行通信，利用SSL/TLS建立全信道，加密数据包。HTTPS使用的主要目的是提供对网站服务器的身份认证，同时保护交换数据的隐私与完整性。

2.2 SSL安全套接字

SSL层简称安全套接字，主要就是加密解密。

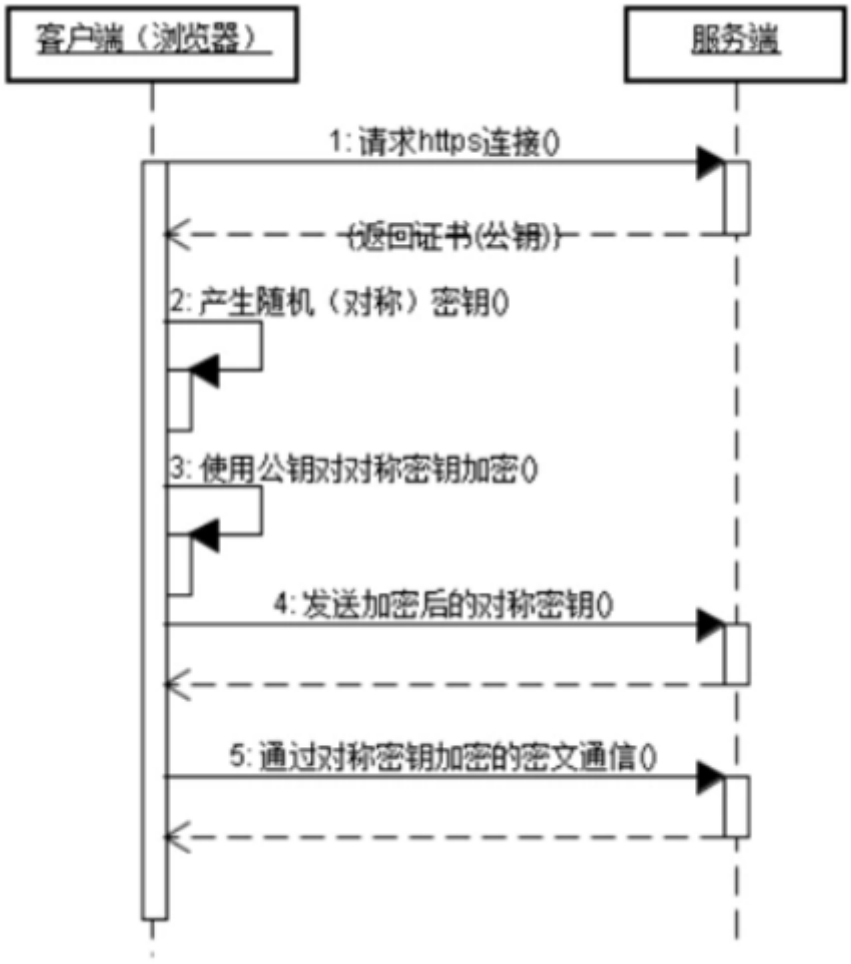
什么是加解密？比如我在群里面吼一句：这周末有空吗？@悠悠；这样会出大事情的！为什么，大家都看到我这句话了，很明显的就要那啥啊？

咋办？加密呗，改成这样：吗末有周这空？@悠悠，你们还看得懂吗？

一开始我们会说，我们就让服务器给我们一个公钥，每次用他来解密不就行了。但是这种办法不可以，因为别人同样可以截获你的公钥，毕竟公钥也是在网络上传递的。

于是我们就要想一种办法，让我们的公钥也被加密，并且这个加密方式是在服务器或者我们电脑上进行的，这样才安全

如图：



重点在这里：

1. 服务器端产生的是对称秘钥，所以只是把公钥发送出去，私钥一直在自己这里 （对称秘钥就是一个公钥，一个私钥，用公钥加密，用私钥直接能解码，但是比较慢，不能一直用这个通信）
2. 客户端首先向一个权威的服务器检查证书的合法性，如果证书合法，客户端产生一段随机数，这个随机数就作为通信的密钥，我们称之为对称密钥，用公钥加密这段随机数，然后发送到服务器
3. 服务器用密钥解密获取对称密钥，然后，双方就以对称密钥进行加密解密通信了

除了对称密钥外，还有非对称密钥，他们的侧重点不同：

（1）对称加密（symmetrcic encryption）：密钥只有一个，加密解密为同一个密码，且加解密速度快，典型的对称加密算法有DES、AES，RC5，3DES等；对称加密主要问题是共享秘钥，除你的计算机（客户端）知道另外一台计算机（服务器）的私钥秘钥，否则无法对通信流进行加密解密。解决这个问题的方案非对称秘钥。

（2）非对称加密：使用两个秘钥：公共秘钥和私有秘钥。私有秘钥由一方密码保存（一般是服务器保存），另一方任何人都可以获得公共秘钥。这种密钥成对出现（且根据公钥无法推知私钥，根据私钥也无法推知公钥），加密解密使用不同密钥（公钥加密需要私钥解密，私钥加密需要公钥解密），相对对称加密速度较慢，典型的非对称加密算法有RSA、DSA等。

3、HTTP协议项目转HTTPS协议项目

3.1 Https证书准备

开发环境下，可直接用JDK自带的keytool工具生成一个证书，正式环境可购买一个，配置过程是一样的：

打开cmd命令行，输入以下命令:

```
keytool -genkey -alias mykeystore -keypass 123456 -keyalg RSA -keysize 1024 -
validity 365 -keystore D:/mykeystore.keystore -storepass 123456
```

命令解释:

1. -alias 证书别名
2. -keypass 证书密码
3. -keyalg 生证书的算法名称，RSA是一种非对称加密算法
4. -keysize 密钥长度
5. -validity 证书的有效期(单位:天)
6. -keystore 生成的证书文件的存储路径
7. -storepass 获取keystore信息的密码

根据提示输入相关信息即可:

3.2 Tomcat服务器配置

将8443端口配置注释取消，并添加第一步生成的证书路径及密码

```
<Connector port="8443"
        SSLEnabled="true" clientAuth="false"
        keystoreFile="D:\mykeystore.keystore"
        keystorePass="123456"
        maxThreads="150"
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        scheme="https" secure="true" sslProtocol="TLS"/>

<!--
    SSLEnabled:开启SSL安全套接字协议证书
    clientAuth:客户端是否需要认证
    keystoreFile：证书位置
    keystorePass:证书密码，跟生成的时候保持一致
    sslProtocol :协议类型
-->
```

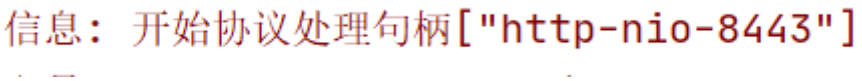
3.3 配置项目web.xml

打开项目下web.xml，添加如下配置

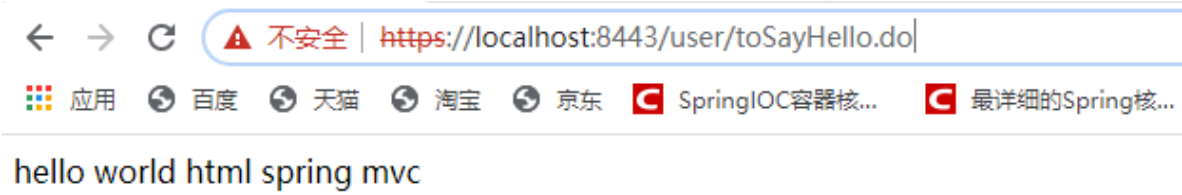
```
<security-constraint>
        <!-- Authorization setting for SSL -->
        <web-resource-collection >
            <web-resource-name >SSL</web-resource-name>
            <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <user-data-constraint>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
 </security-constraint>
```

3.4 测试

服务启动时会暴露8443安全证书接口：



接口访问测试：



当然，使用的是JDK自带的证书，所以谷歌浏览器会告诉你不是一个有效的证书：