**优必选-Https协议接入**

**编制部门：软件部**

**编 制 人：贾亮亮**

**审 核：**

**会 签：**

**批 准：**

|  |  |  |  |
| --- | --- | --- | --- |
| **文件修订履行** | | | |
| 版本 | 修订内容简述 | 生效日期 | 修订部门/修订者 |
| 1 | 新制订 | 2017-09-07 | 软件部 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**说明**

1. 本文对Https规范进行介绍
2. 对不同容器添加Https协议添加说明文档

[1. HTTPS的协议规范 4](#_Toc19929)

[1.1 https协议详解 4](#_Toc9730)

[1.2 https和http的区别 4](#_Toc15145)

[1.3 https解决的问题 4](#_Toc22977)

[1.4 HTTPS 的繁琐性 5](#_Toc32515)

[2. SpringBoot工程增加Https支持 6](#_Toc31378)

[2.1 Springboot的application.properties 6](#_Toc14996)

[2.2 HTTP自动转向HTTPS 6](#_Toc17517)

[2.3 参考开发项目 7](#_Toc25456)

[3. Webapp工程增加Https支持 8](#_Toc16945)

[4. Nginx增加Https支持 9](#_Toc24136)

# HTTPS的协议规范

## https协议详解

HTTPS以保密为目标研发，简单讲是HTTP的安全版。其安全基础是SSL协议，因此加密的详细内容请看SSL。全称Hypertext Transfer Protocol over Secure Socket Layer。

它是一个URI scheme，句法类同http:体系。它使用了HTTP，但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）。这个协议的最初研发由网景公司进行，提供了身份验证与加密通讯方法，现在它被广泛用于互联网上安全敏感的通讯，例如交易支付方面。

  SSL极难窃听，对中间人攻击提供一定的合理保护。严格学术表述HTTPS是两个协议的结合，即传输层SS＋应用层HTTP。

HTTPS默认使用TCP端口443（HTTP默认则是TCP端口80），也可以指定其他TCP端口。

要使协议正常运作，至少服务器必需有PKI证书，而客户端则不一定。

它的加密强度依赖软件的正确实现，以及服务器客户端双方加密算法的支持。

## https和http的区别

https协议需要到ca申请证书，一般免费证书很少，需要交费，http是超文本传输协议，信息是明文传输，https 则是具有安全性的ssl加密传输协议

http和https使用的是完全不同的连接方式用的端口也不一样,前者是80,后者是443。

http的连接很简单,是无状态的

https协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议 要比http协议安全。

## https解决的问题

1 . 信任主机的问题. 采用https 的server 必须从CA 申请一个用于证明服务器用途类型的证书. 改证书只有用于对应的server 的时候,客户度才信任次主机. 所以目前所有的银行系统网站,关键部分应用都是https 的. 客户通过信任该证书,从而信任了该主机. 其实这样做效率很低,但是银行更侧重安全. 这一点对我们没有任何意义,我们的server ,采用的证书不管自己issue 还是从公众的地方issue, 客户端都是自己人,所以我们也就肯定信任该server.

2 . 通讯过程中的数据的泄密和被窜改

一般意义上的https, 就是 server 有一个证书.

a) 主要目的是保证server 就是他声称的server. 这个跟第一点一样.

b) 服务端和客户端之间的所有通讯,都是加密的.

i. 具体讲,是客户端产生一个对称的密钥,通过server 的证书来交换密钥. 一般意义上的握手过程.

ii. 加下来所有的信息往来就都是加密的. 第三方即使截获,也没有任何意义.因为他没有密钥. 当然窜改也就没有什么意义了.

3.少许对客户端有要求的情况下,会要求客户端也必须有一个证书.

a) 这里客户端证书,其实就类似表示个人信息的时候,除了用户名/密码, 还有一个CA 认证过的身份. 应为个人证书一般来说上别人无法模拟的,所有这样能够更深的确认自己的身份.

b) 目前少数个人银行的专业版是这种做法,具体证书可能是拿U盘作为一个备份的载体。

## HTTPS 的繁琐性

a) 本来简单的http协议,一个get一个response. 由于https 要还密钥和确认加密算法的需要.单握手就需要6/7 个往返.

i. 任何应用中,过多的round trip 肯定影响性能.

b) 接下来才是具体的http协议,每一次响应或者请求, 都要求客户端和服务端对会话的内容做加密/解密.

i. 尽管对称加密/解密效率比较高,可是仍然要消耗过多的CPU,为此有专门的SSL 芯片. 如果CPU 信能比较低的话,肯定会降低性能,从而不能serve 更多的请求.

ii. 加密后数据量的影响. 所以，才会出现那么多的安全认证提示

# SpringBoot工程增加Https支持

## Springboot的application.properties

(代码可以参考 ubtechinc-example-service，此工程已经支持了Https协议)

(其中keystore.p12 和 .keystore 请使用 ubtechinc-example-service项目文件夹下面的，已经上传了的，目录放到项目的根目录下--跟pom.xml文件同级)

(网上可以参考：<http://blog.csdn.net/u012702547/article/details/53790722)>

添加HTTPS支持。在application.properties中添加如下代码：

server.ssl.key-store=springboot.p12

server.ssl.key-store-password=111111

server.ssl.keyStoreType=PKCS12

server.ssl.keyAlias:1

第一行指定签名文件，第二行指定签名密码，第三行指定密钥仓库类型，第四个是别名。OK，这样配置完成之后我们就可以通过HTTPS来访问我们的Web了。

## HTTP自动转向HTTPS

光有HTTPS肯定还不够，很多用户可能并不知道，用户有可能继续使用HTTP来访问你的网站，这个时候我们需要添加HTTP自动转向HTTPS的功能，当用户使用HTTP来进行访问的时候自动转为HTTPS的方式。这个配置很简单，在入口类中添加相应的转向Bean就行了，如下：

@Bean

Public EmbeddedServletContainerFactory servletContainer() {

TomcatEmbeddedServletContainerFactory tomcat=new TomcatEmbeddedServletContainerFactory() {

@Override

protected void postProcessContext(Context context) {

SecurityConstraint constraint = new SecurityConstraint();

constraint.setUserConstraint("CONFIDENTIAL");

SecurityCollection collection = new SecurityCollection();

collection.addPattern("/\*");

constraint.addCollection(collection);

context.addConstraint(constraint);

}

};

tomcat.addAdditionalTomcatConnectors(httpConnector());

return tomcat;

}

@Bean

public Connector httpConnector() {

Connector connector = new Connector("org.apache.coyote.http11.Http11NioProtocol");

connector.setScheme("http"); //Connector监听的http的端口号

connector.setPort(8080);

connector.setSecure(false);

//监听到http的端口号后转向到的https的端口号

connector.setRedirectPort(8443);

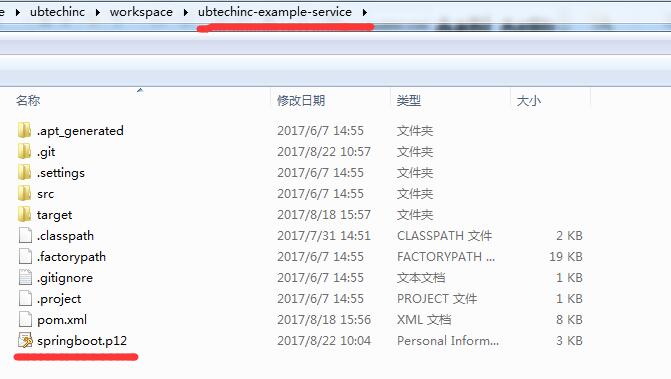
return connector;

}

## 参考开发项目

代码可以参考 ubtechinc-example-service，此工程已经支持了Https协议。

其中springboot.p12已经在该工程下面了，请大家直接可以拷贝使用，具体配置可以参考2.1和2.2的配置。



# Tomcat工程增加Https支持

tomcat.keystore 生成的私有证书，请到服务器:

(ubt-temp(\\10.10.1.1)(Z:\软件部门\开发私有证书\_数据平台\Tomcat\windows\)) 目录下：

并且将 tomcat.keystore 放到apache-tomcat 的bin目录下面

在server.xml中添加如下配置信息：

<Connector port="443" protocol="org.apache.coyote.http11.Http11Protocol" SSLEnabled="true" maxThreads="150" scheme="https" secure="true"

keystoreFile="bin/tomcat.keystore" keystorePass="111111" clientAuth="false" sslProtocol="TLS" />

启动Tomcat后，便可以通过https访问，例如：<https://127.0.0.1/userInfo?id=1>

# Nginx增加Https支持

私有证书Lee.crt和lee.key，请放到Nginx服务器下面的conf文件夹下:

(ubt-temp(\\10.10.1.1)(Z:\软件部门\开发私有证书\_数据平台\Nginx\windows\)) 目录下：

并且将 lee.crt和lee.key 放到Nginx 的conf目录下面。

Nginx的配置文件信息需要添加 conf/lee.crt; #(证书公钥) 和 conf/lee.key; #(证书私钥)

upstream tomcat-client {

server 127.0.0.1:8081 weight=1; #服务器imageserver01

}

upstream cas.account.ubtrobot.com {

server account.ubtrobot.com:8060 weight=1; #服务器imageserver01

}

# HTTPS

server {

listen 443;

server\_name www.lee.com; # 项目域名

ssl on;

ssl\_certificate conf/lee.crt; #(证书公钥)

ssl\_certificate\_key conf/lee.key; #(证书私钥)

ssl\_session\_timeout 5m;

ssl\_protocols SSLv2 SSLv3 TLSv1;

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

location ^~ /cas/ {

#proxy\_redirect off;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_pass https://cas.account.ubtrobot.com;

client\_max\_body\_size 1000m; ##文件上传大小限制

}

location / {

proxy\_pass http://blog.csdn.net; # tomcat服务器地址

index index.html index.htm;

}

}