CAS（Central Authentication Service，中央验证服务）

CAS由CAS Server和CAS Client两部分组成。CAS Server是一个单点的验证服务，CAS Client是共享

CAS Server登录态的客户端。

此外CAS还有三个重要的术语：

Ticket Granting Ticket（TGT）

Ticket Granted Cookie（TGC）

Service Ticket（ST）

◎ TGT是用户登录后生成的票根，包含用户的认证身份、有效期等，存储于CAS Server中，类似

于我们常见的服务器会话。

◎ TGC是存储在cookie中的一段数据，类似于会话ID，用户与CAS Server进行交互时，帮助用户

找到对应的TGT。

◎ ST是CAS Server使用TGT签发的一张一次性票据，CAS Client使用ST与CAS Server进行交互，

以获取用户的验证状态。

AS单点登录的完整步骤如下：

（1）用户通过浏览器访问CAS Client的某个页面，例如http://cas.client.com/me。

（2）当CAS Client判断用户需要进行身份认证时，携带service返回302状态码，指示浏览器重定向

到CAS Server，例如http://cas.server.com/?service=http://cas.client.com/me。service指用户原访问页面。

（3）浏览器携带service重定向到CAS Server。

（4）CAS Server获取并校验用户cookie中携带的TGC，如果成功，则身份认证完成；否则将用户

重定向到 CAS Server 提供的登录页，例如 http://cas.server.com/login?service=http://cas.client.com/me，由

用户输入用户名和密码，完成身份认证。

（5）如果用户已经登录过系统，那么CAS Server可以获取用户的TGC，并根据TGC找到TGT。如

果是首次登录，则CAS Server会首先生成TGT。每次验证时，CAS Server会根据TGT签发一个ST，把

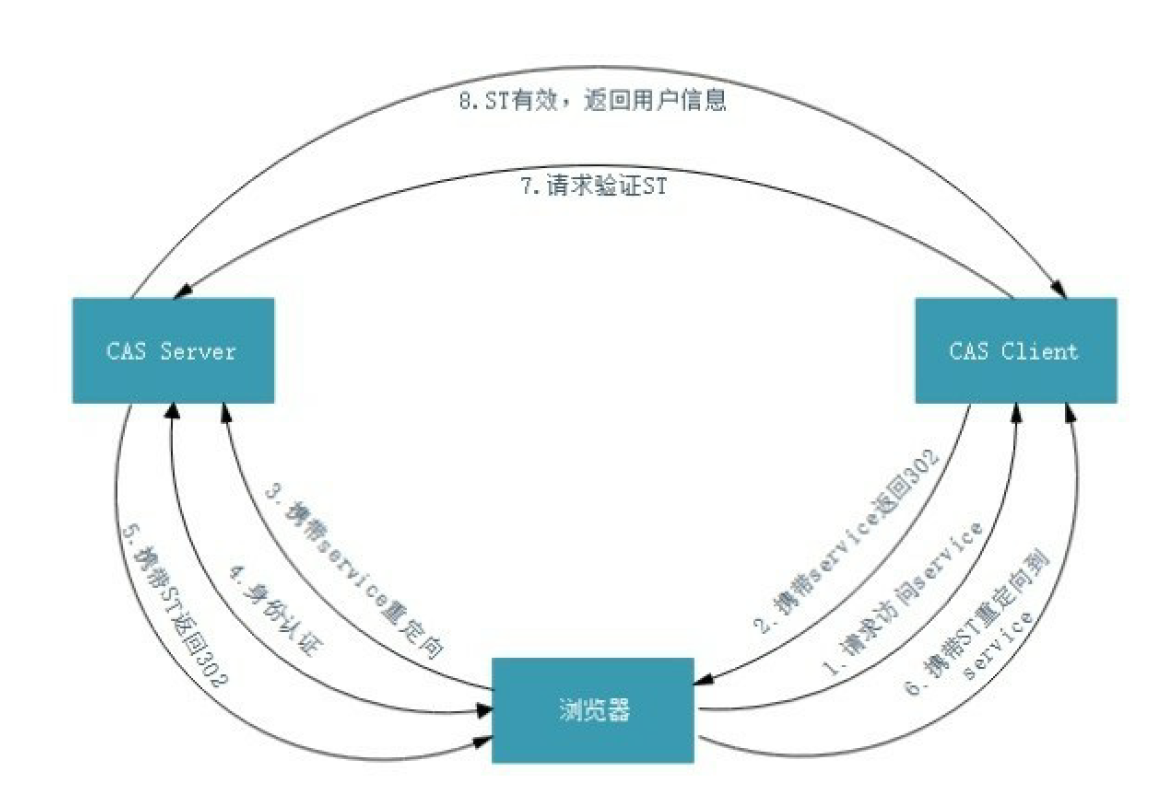
ST拼接在service中，同时将相应的TGC设置到用户的cookie中（域为CAS Server），并返回302状态

码，指示浏览器重定向到 service，例如 http://cas.client.com/me? ticket=xxx。

（6）浏览器存储TGC，并携带ST重定向到service。

（7）CAS Client取得ST（即请求参数中的ticket）后，会向CAS Server请求验证该ST的有效性。

（8）若CAS Server验证该ST是有效的，就告知CAS Client该用户有效，并返回该用户的信息。



官方网址

<https://apereo.github.io/cas/6.1.x/index.html>

搭建CAS环境 以5.3版本搭建

<https://apereo.github.io/cas/5.3.x/>

**1.**环境依赖

◎ JDK1.8+，需要配置好环境变量，在搭建过程中将使用 Java 提供的 keytool 证书制作工具。

◎ Maven，需要配置好环境变量，在Windows的CMD或macOS X的Terminal中输入“mvn–v”进行验

证。

下载CAS源码 选择5.3分支， 下载zip

<https://github.com/apereo/cas-overlay-template>

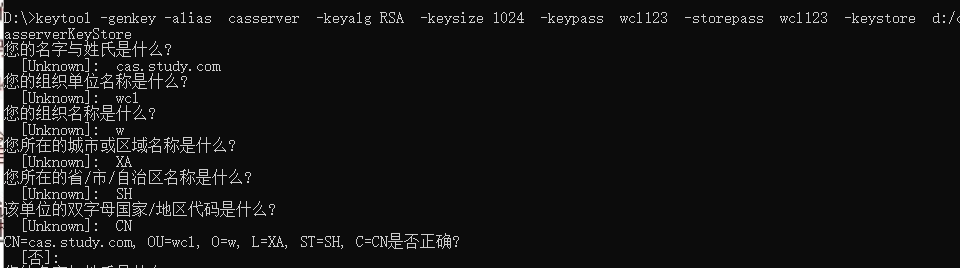
制作本地密钥库

CAS Server默认使用HTTPS进行访问，并要求我们提供一个密钥库。

可以使用 Java 自带的密钥和证书管理工具 keytool 制作本地密钥库，它位于{JAVA\_HOMR}/bin目

录下。在配置环境变量时，在控制台直接输入keytool命令就可以获取帮助，

keytool -genkey -alias casserver -keyalg RSA -keysize 1024 -keypass wcl123 -storepass wcl123 -keystore -storetype pkcs12 d:/casserverKeyStore



另外，在“您的名字与姓氏是什么？”一项，应当填写CAS Server的域名，否则在后续的单点登录

过程中会遇到问题

使用export子命令导出证书。

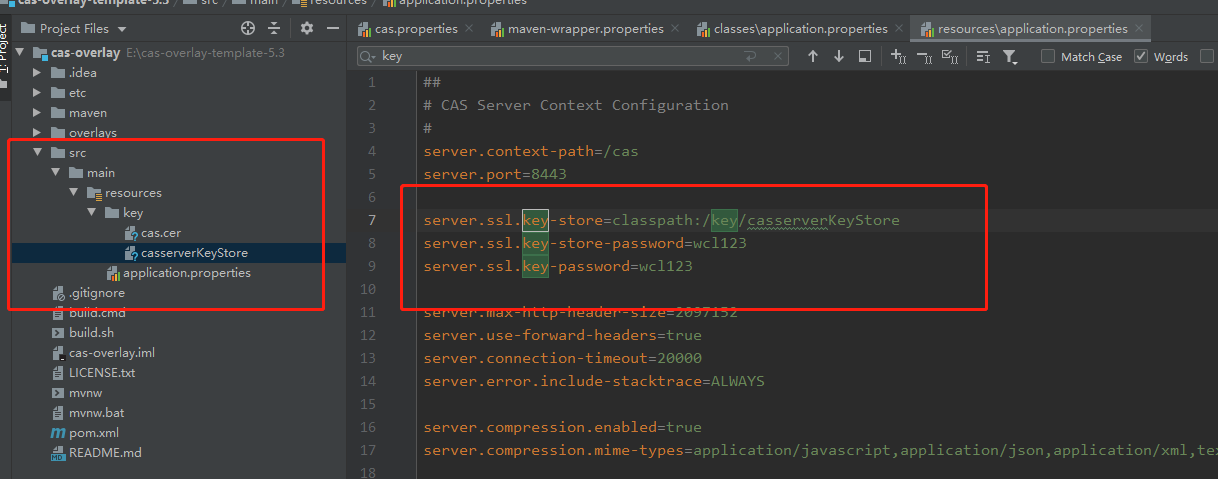
keytool -export -trustcacerts -alias casserver -file d:/cas.cer -keystore d:/casserverKeyStore

新建目录src/main/resources，将overlays/{cas-server}/WEB-INF/classes/application.properties复制到

此目录下，后续将用这个新的配置文件覆盖CAS Server的配置。

在CAS Server配置中，关于SSL证书的三个主要配置如下。

也可以把keystore复制到resources目录下，把key-store修改为下面的形式。



**5.**启动**CAS Server**

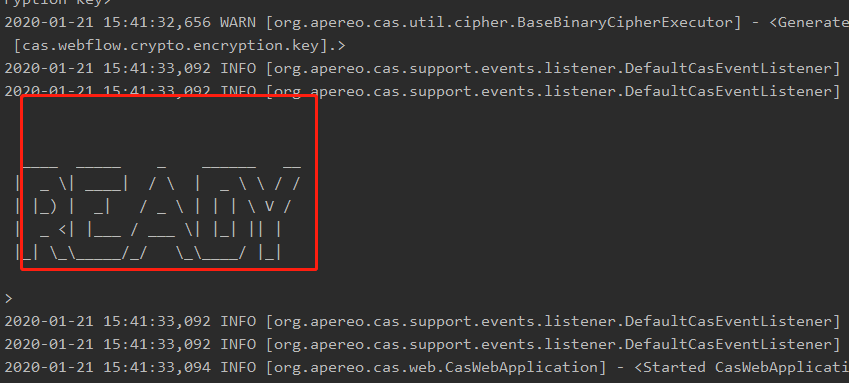
既可以将CAS Server打包为war包的形式运行，也可以使用Spring Boot的方式运行，

./build.cmd run

或者

./build.cmd bootrun

看到ready 图案即起动成功



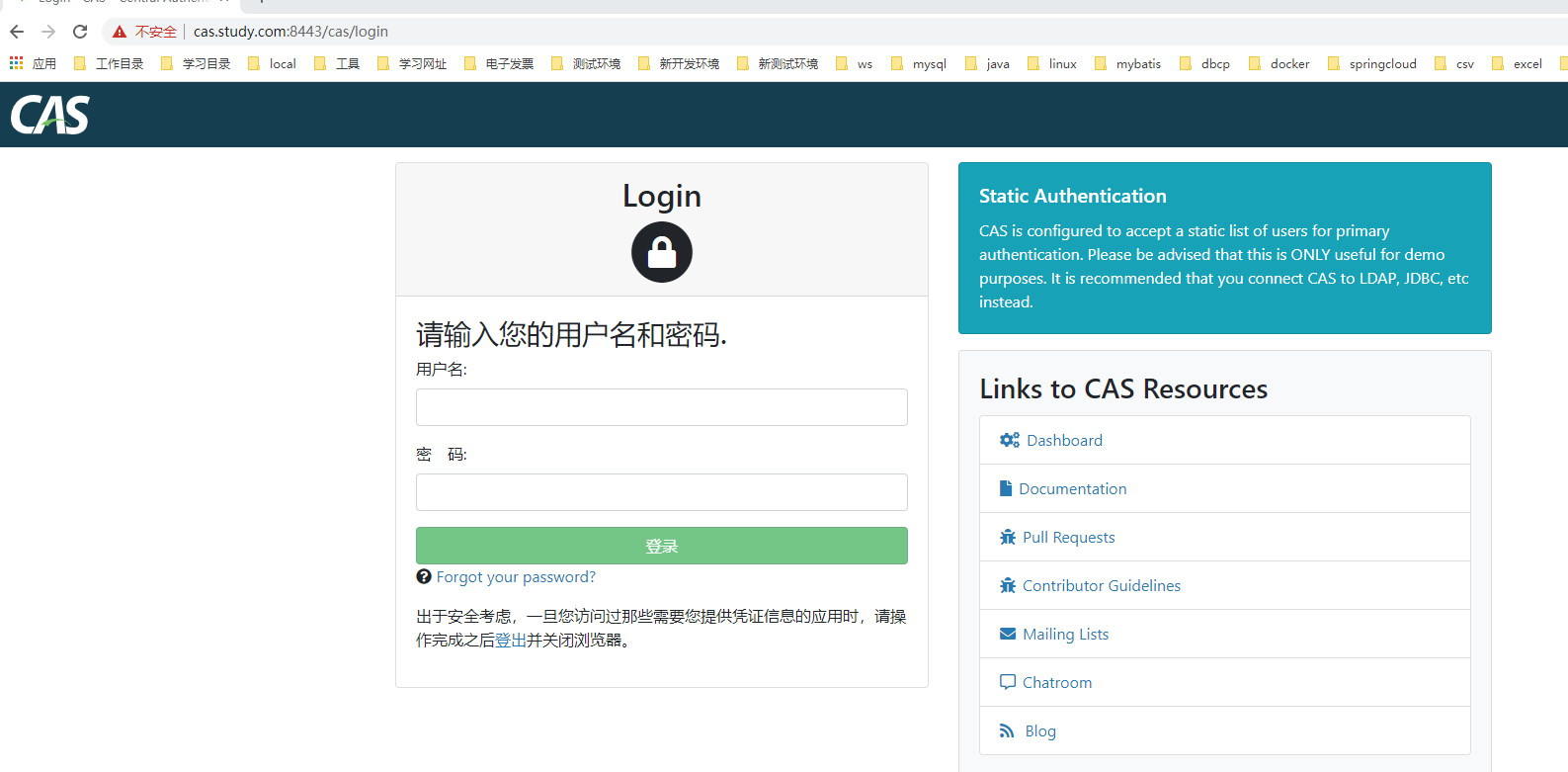
Hosts文件中添加

127.0.0.1 cas.study.com



保存，重生打开浏览器 输入

<https://cas.study.com:8443/cas/>



默认情况下使用用户名 casuser 和密码 Mellon 即可登录，如果需要修改，则可以在

application.properties文件的最后一行中变更cas.authn.accept.users的值，重启后即可生效。

**7.Services**配置

如果CAS Client想要接入某个CAS Server，则必须定义服务记录，否则客户端在访问时将提示“未

验证授权的服务”。CAS Server本身支持多种注册服务方式，每个服务对应一个允许接入的客户端。

在测试阶段，只使用JSON 声明的方式注册服务，每个JSON文件对应一个服务，建议用下面的规

则创建JSON文件名。

{

"@class":"org.apereo.cas.service.RegexRegisteredService",

"serviceId":"^https://client.cas.study.com",

"name":"cas client",

"id":10000,

"description":"cas client based on spring security",

"evaluationOrder":1

}



其中，@class 必须指定为org.apereo.cas.services.RegexgisteredService 的实现类，serviceId则通过正则

表达式的方式匹配来自 CAS Client 的 URL，每个 id 都是全局唯一编号，evaluationOrder指明了服务的

执行顺序。

把该JSON文件命名为cas\_client-10000.json，并放置到/src/main/resources/services下，但暂时不会生

效，还需要为CAS Server添加一些相关配置。

# Json资源加载路径

cas.serviceRegistry.json.location=calsspath:/services

#是否开启json文件识别，默认为false

cas.serviceRegistry.initFromJson=true

#是否自动扫描服务配，默认开启（在CAS Server启动时通过控制台定时输出的信息也能观察到这一点）

#默认每120s扫描一次服务配置

#cas.serviceRegistry.repeatInterval=120000

#默认在CAS Server 启动后延迟15s开启自动扫描

#cas.serviceRegistry.startDelay=15000