6、单点登录 service配置

# 1、解释;

cas客户端接入称之为service，必须经过cas的允许才能进行登录，当然不同的客户端可以做不同的事情，其中包括：

* 自定义主题（各客户端登录页自定义）
* 自定义属性（服务属性（固定）与用户属性（动态））
* 自定义协议
* 自定义登录后跳转方式，跳转路径
* 授权策略（拒绝属性、可登录时间范围限制、等等）
* 拒绝授权模式

接下来来一个FAQ

**A:** 什么是service，根cas有什么关系？   
**B:** service是使用型cas是服务型，cas好比游乐园，service好比来游乐园的游客   
**A:** 那service如何对接cas？service如何知道是否被允许接入？   
**B:** 好比游客需要进入游乐园，那么游客需要门票，获取门票有多种方式，可以用手机校验码，或者身份证进行获取。当然如果是犯罪分子门票都买不了，更何况进去游乐园   
**A:** 具体service如何作为客户端使用？   
**B:** 好比买门票，必须填写身份证号、手机号、付款等流程，当然也可以通过不同渠道购买，cas也有不同的客户端实现，cas client、pac4j等   
**A:** service接入有何好处？   
**B:** 被接入的service无需进行输入密码即可进入系统，好比A-service（OA系统）登录了，B-service（账单系统），C-service（CRM系统）无需再次登录，传统的方式有10个八个系统进行登录，用户会疯掉，开发员也需要管理多个系统的帐号数据

说了一大推，不如来点正事，例如怎么配置，如何管理，马上马上~~

# 实战

持久化策略：

* InMemory XML(通过spring bean进行内存存储)
* JSON(通过json文件存储) **推荐** ❤❤❤
* YAML(通过yml文件存储)
* Mongo(文档数据库持久化)**推荐** ❤❤❤❤❤
* JPA(关系型数据库持久化)
* DynameDb
* LDAP
* Cochbase

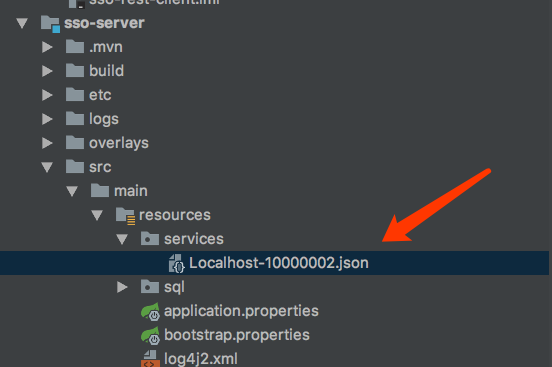
在sso初步上线时推荐采用json文件存储，后面逐步多服务注入时推荐采用Mongo进行存储，采用cas-management进行采用UI进行管理我们的数据，目前阶段，持久化策略必须和cas进行配置一致才能生效

**本章进行service的json配置及介绍，下一章进行cas-management配置**

# **1、**JSON 配置 （默认json/yml资源加载路径为resources/services）

## 需求：对所有http://localhost开头请求的service进行允许认证

## 1、sso-server 中的resource中添加services文件夹



{  
 "@class": "org.apereo.cas.services.RegexRegisteredService",  
 "serviceId": "^(http|https)://localhost.\*",  
 "name": "Local Services",  
 "id": 10000002,  
 "description": "this is a localhost service,all of can login",  
 "evaluationOrder": 10000000  
}

### 解释

#### 1、@class：必须为org.apereo.cas.services.RegisteredService的实现类，对其他属性进行一个json反射对象，常用的有RegexRegisteredService，匹配策略为id的正则表达式

#### 2、serviceId：唯一的服务id

#### 3、name： 服务名称，会显示在默认登录页

#### 4、id：全局唯一标志

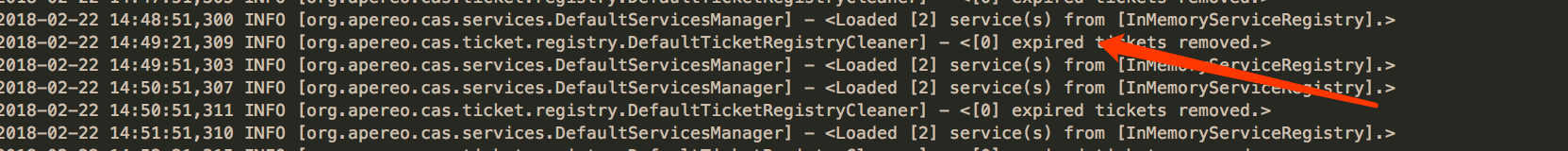
#### 5、description：服务描述，会显示在默认登录页

#### 6、evaluationOrder： 匹配争取时的执行循序

# 2、sso-config中添加配置启用识别json

# 6、 service  
  
#开启识别json文件，默认false  
cas.serviceRegistry.initFromJson=true  
#自动扫描服务配置，默认开启  
#cas.serviceRegistry.watcherEnabled=true  
#120秒扫描一遍  
#cas.serviceRegistry.repeatInterval=120000  
#延迟15秒开启  
#cas.serviceRegistry.startDelay=15000  
#默认json/yml资源加载路径为resources/services  
#cas.serviceRegistry.config.location=classpath:/services

# 3、测试启动，接上5中代码，启动 sso-rest-client sso-config sso-server (观察到下面，说明成功)



# 补充：cas提供了默认的service，如果不需要可以设置在tomcat的插件中 ，这样上面的2就会显示成1 ，写成下面的时候表示用到的时候才会进行json的初始化

<plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-war-plugin</artifactId>  
 <version>3.1.0</version>  
 <configuration>  
 <warName>cas</warName>  
 <failOnMissingWebXml>false</failOnMissingWebXml>  
 <recompressZippedFiles>false</recompressZippedFiles>  
 <archive>  
 <compress>false</compress>  
 <manifestFile>${project.build.directory}/war/work/org.apereo.cas/cas-server-webapp-tomcat/META-INF/MANIFEST.MF</manifestFile>  
 </archive>  
 <overlays>  
 <overlay>  
 <groupId>org.apereo.cas</groupId>  
 <artifactId>cas-server-webapp-tomcat</artifactId>  
 </overlay>  
 </overlays>  
 <dependentWarExcludes>  
 <!--war包下的服务不进行初始化-->  
 \*\*/services/\*.json  
 </dependentWarExcludes>  
 </configuration>  
</plugin>