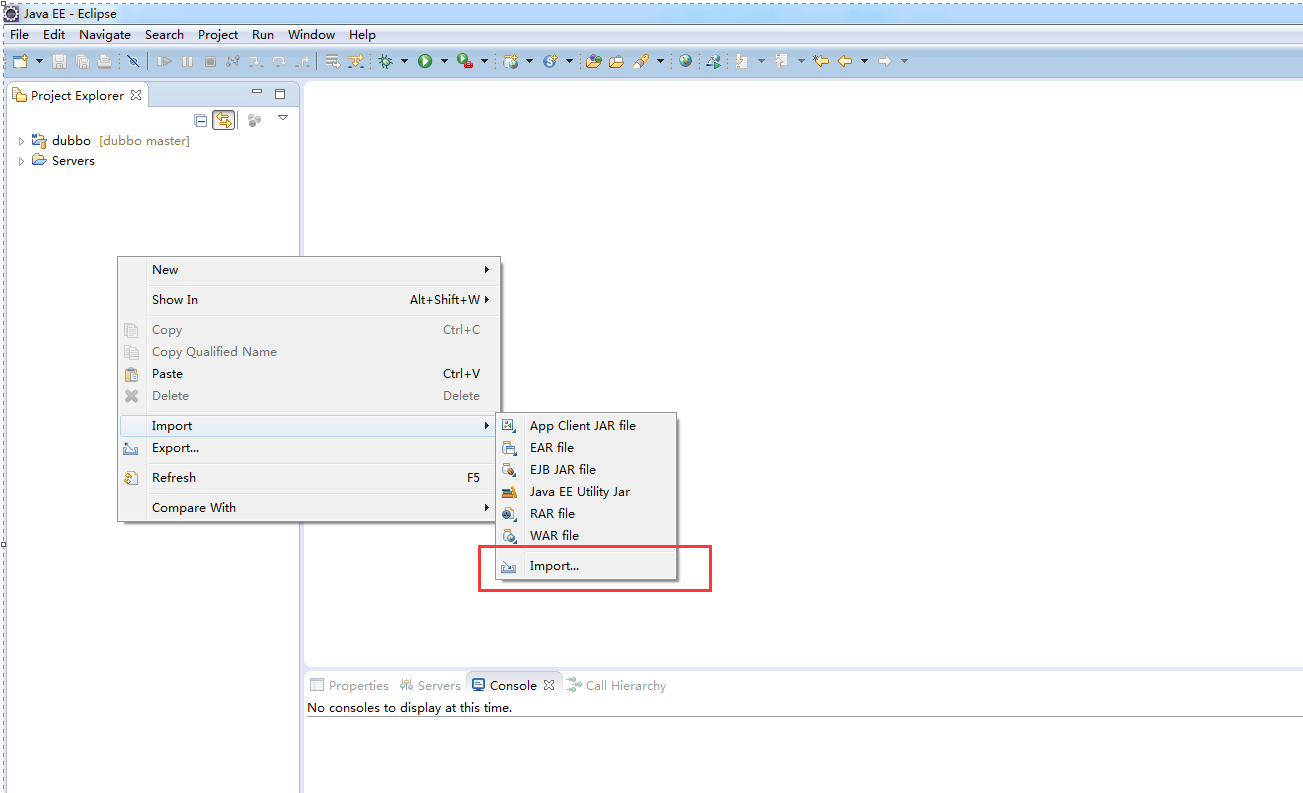
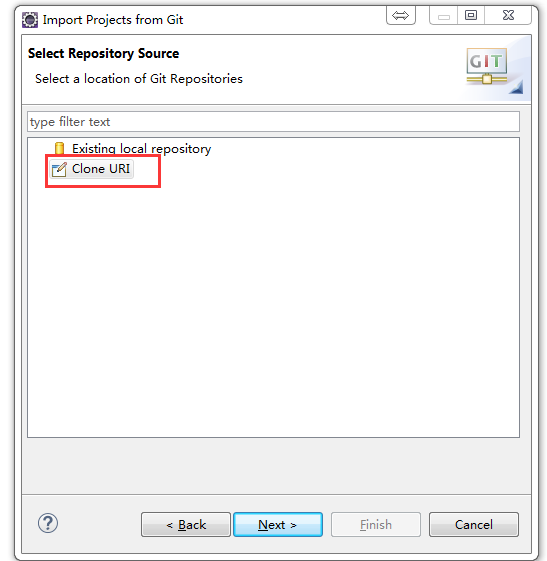
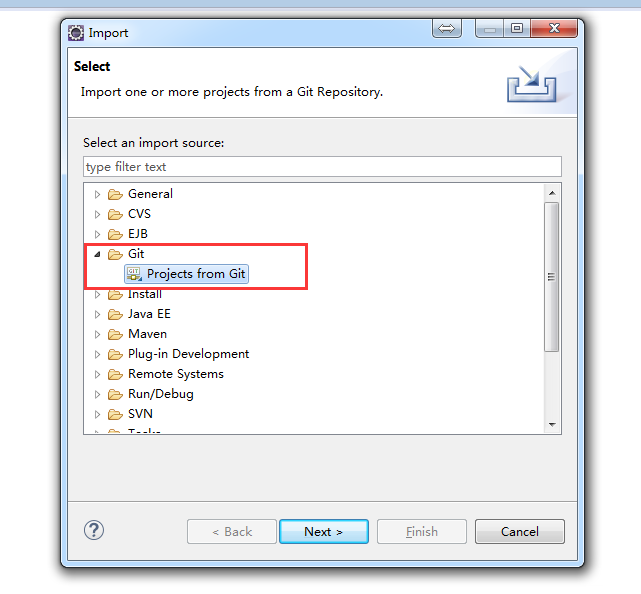
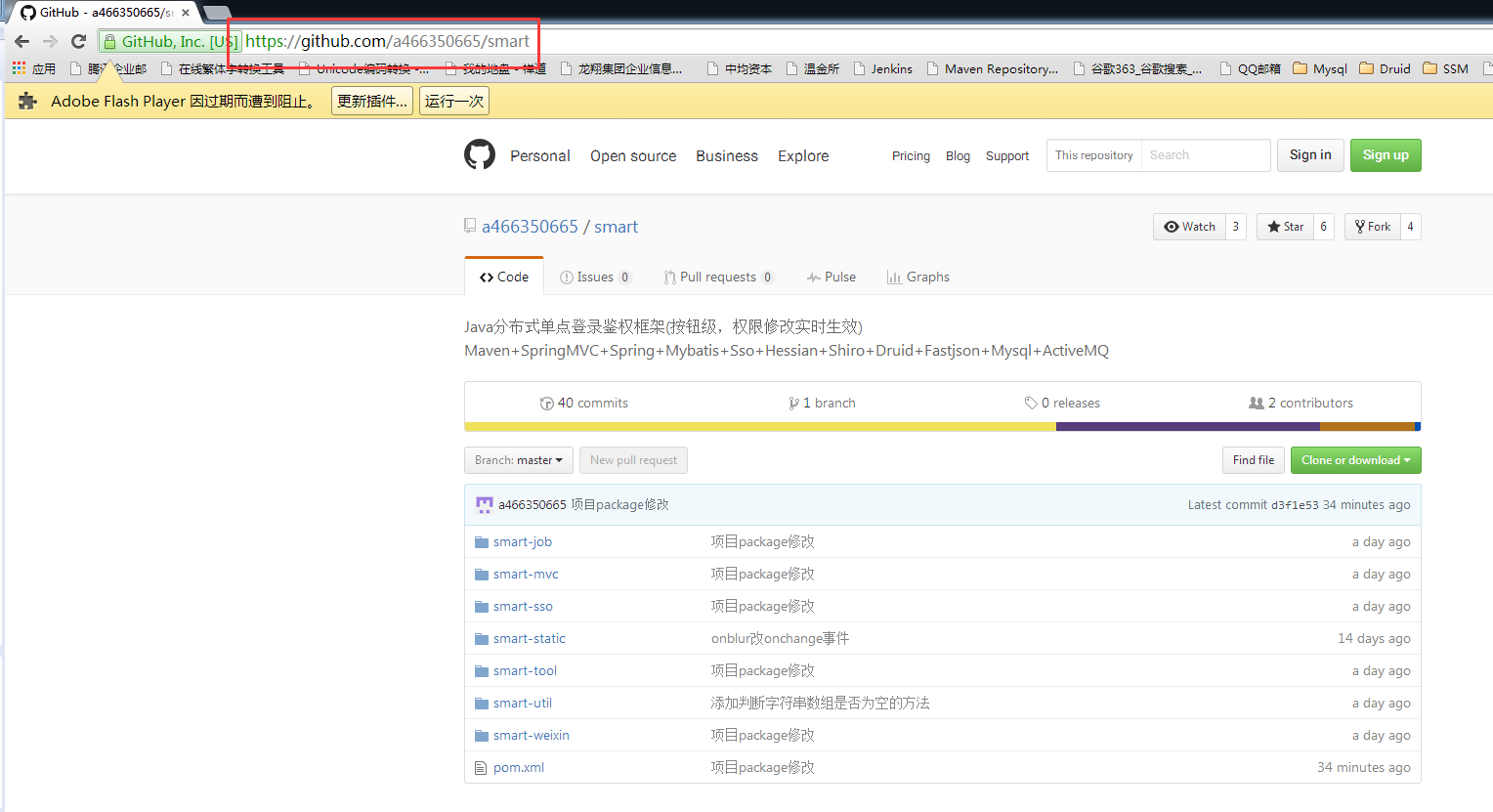
单点登录权限系统

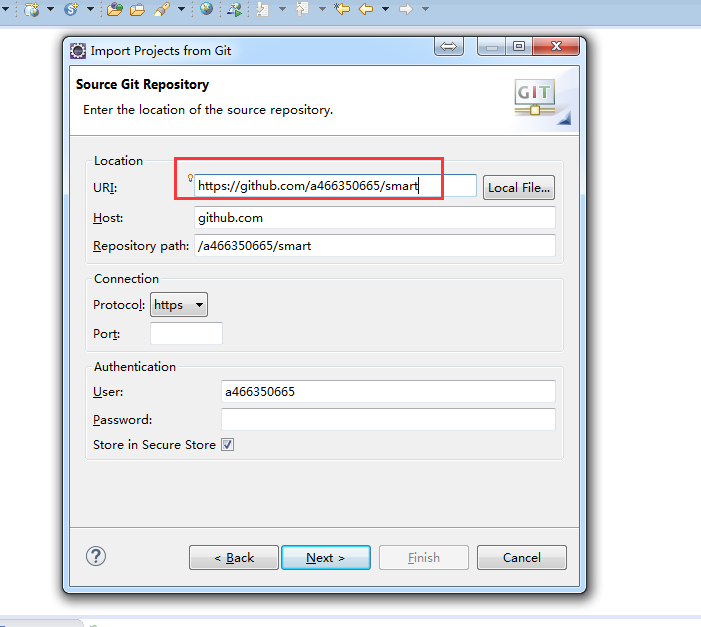
首先准备你的开发环境，笔者是Eclipse、Tomcat8、 Maven、 Mysql及其连接工具（Navicat for Mysql）。下面开始部署配置操作：

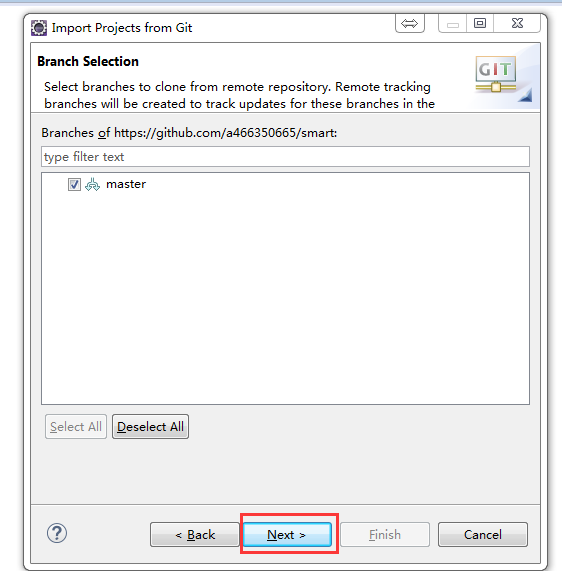
1. Smart项目的导入
2. 从Github下载Smart项目

右击“Import🡪 Import”，选择“Git->Projects from Git”，再 

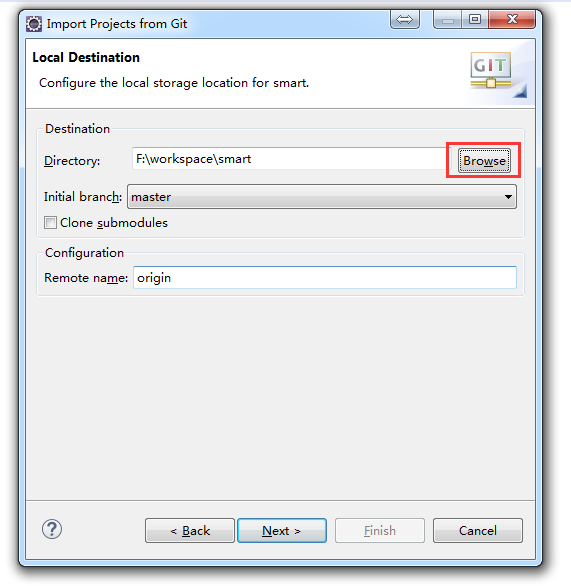
1. 选择“Clone URI”, 再  
2. 从github复制项目链接: https://github.com/a466350665/smart，并粘贴到“URI”上, 再



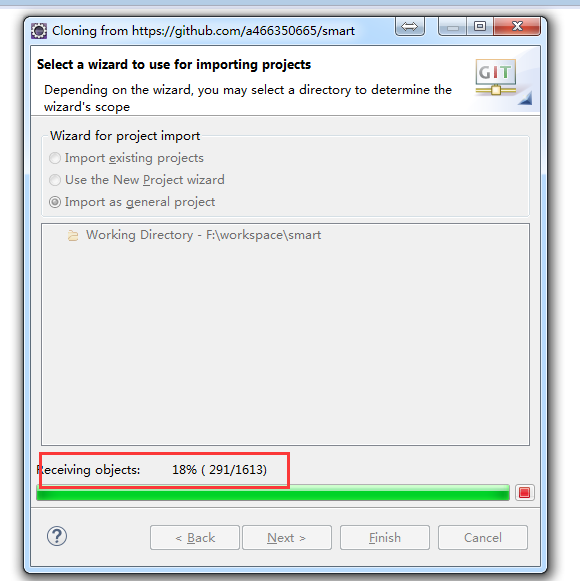


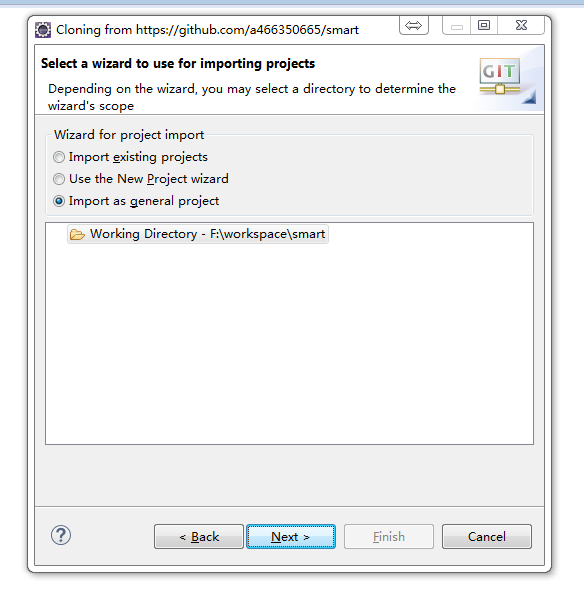


1. 选择本地存放smart项目的目录，再

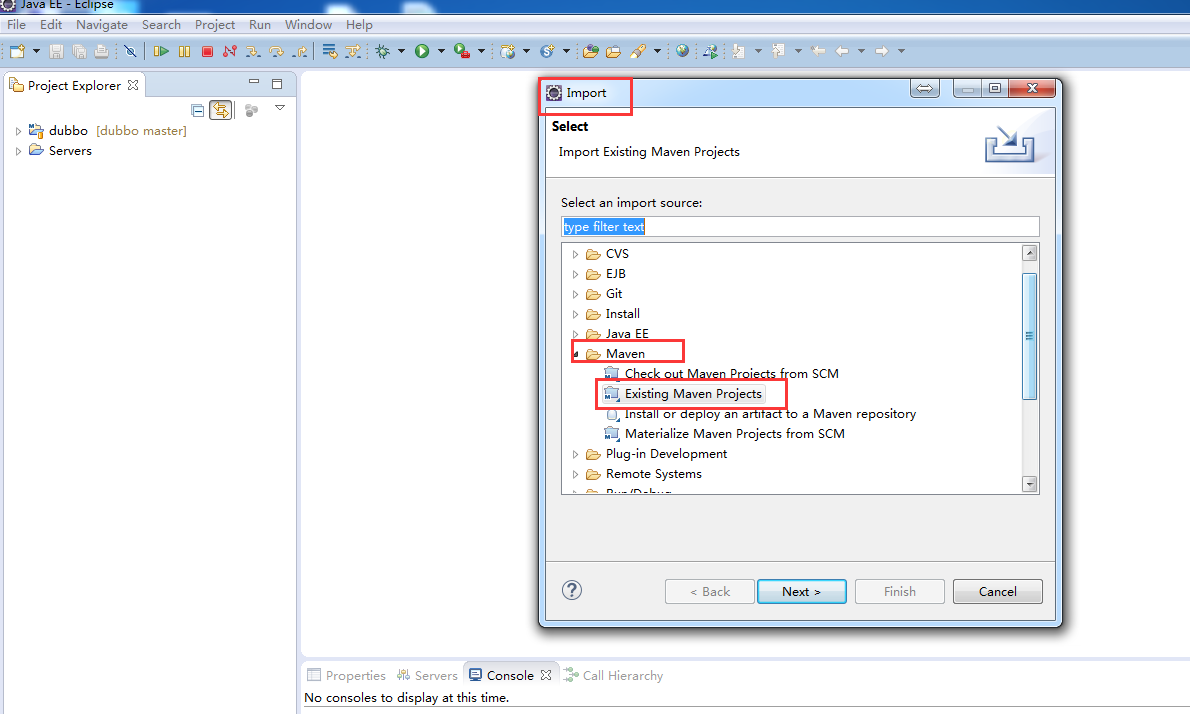


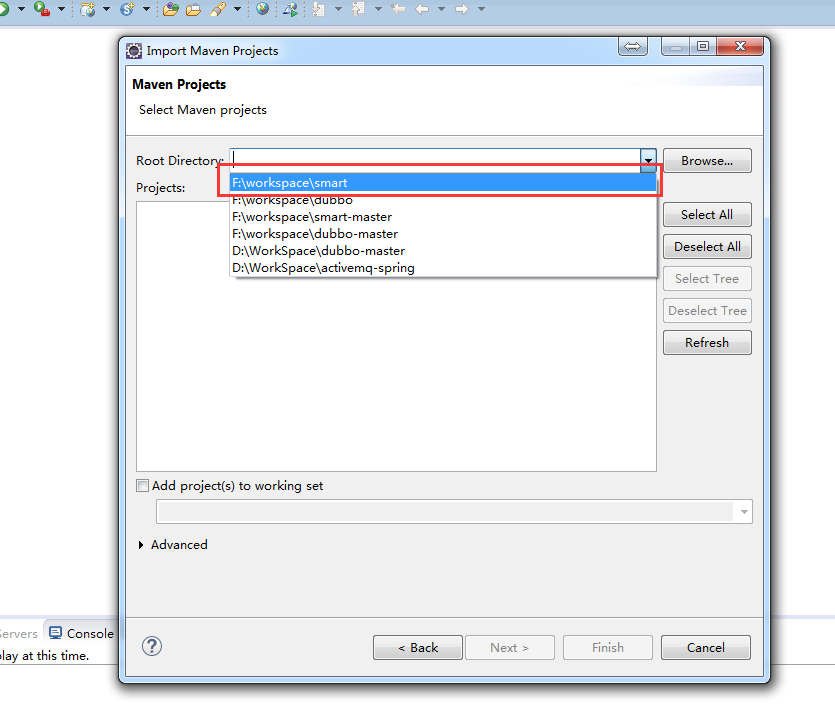
1. 看到如下第二张图就是导入成功了。 “Import as general project”提示让导入并生成项目,也可以直接掉窗口，因为项目已下载到本地。

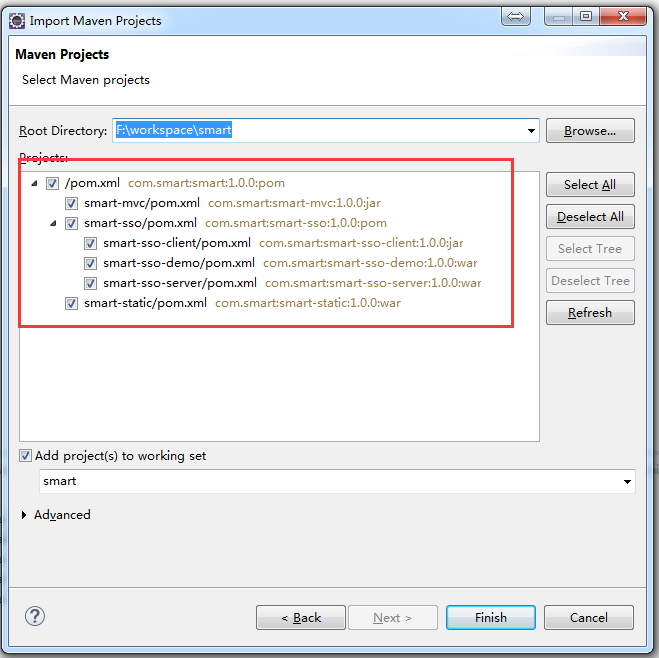




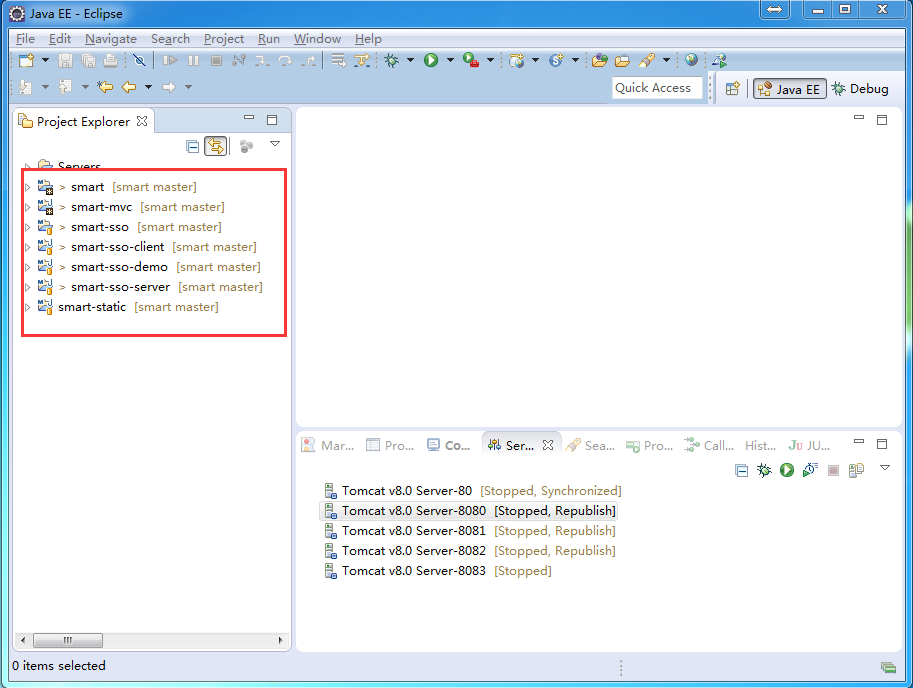
1. 接着就是以引入maven项目的方式，把smart导入到eclipse. 右击“Import🡪 Import”，选择“Maven->Existing Maven Projects”, 再



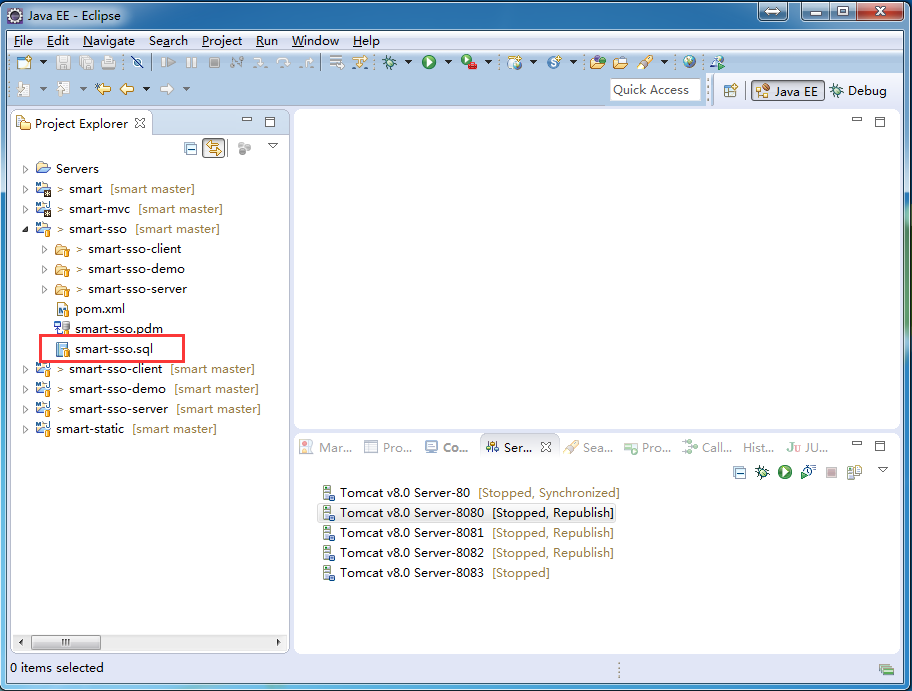
1. 选择下载好的Smart项目本地存放路径，点击

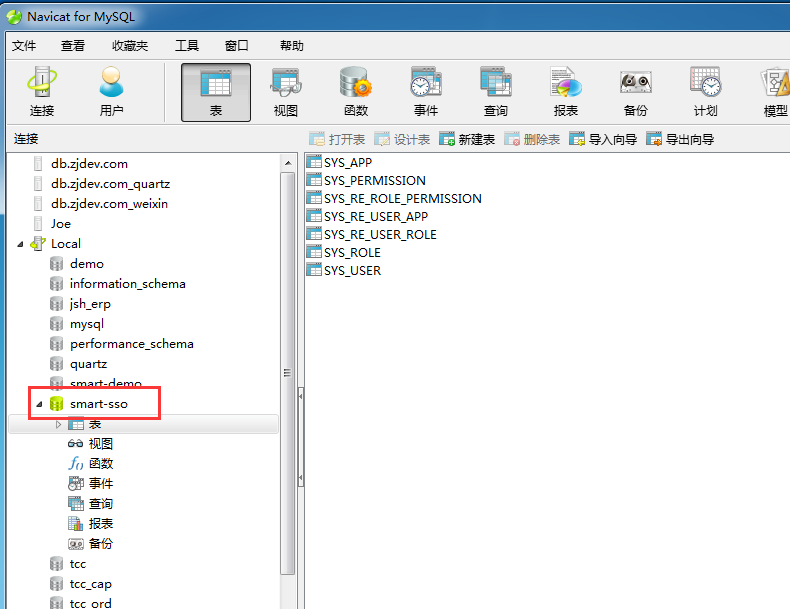


1. 第一次导入，maven需要下载smart所需要的Jar文件，会等待比较长的时间，下载完成后如下图所示：

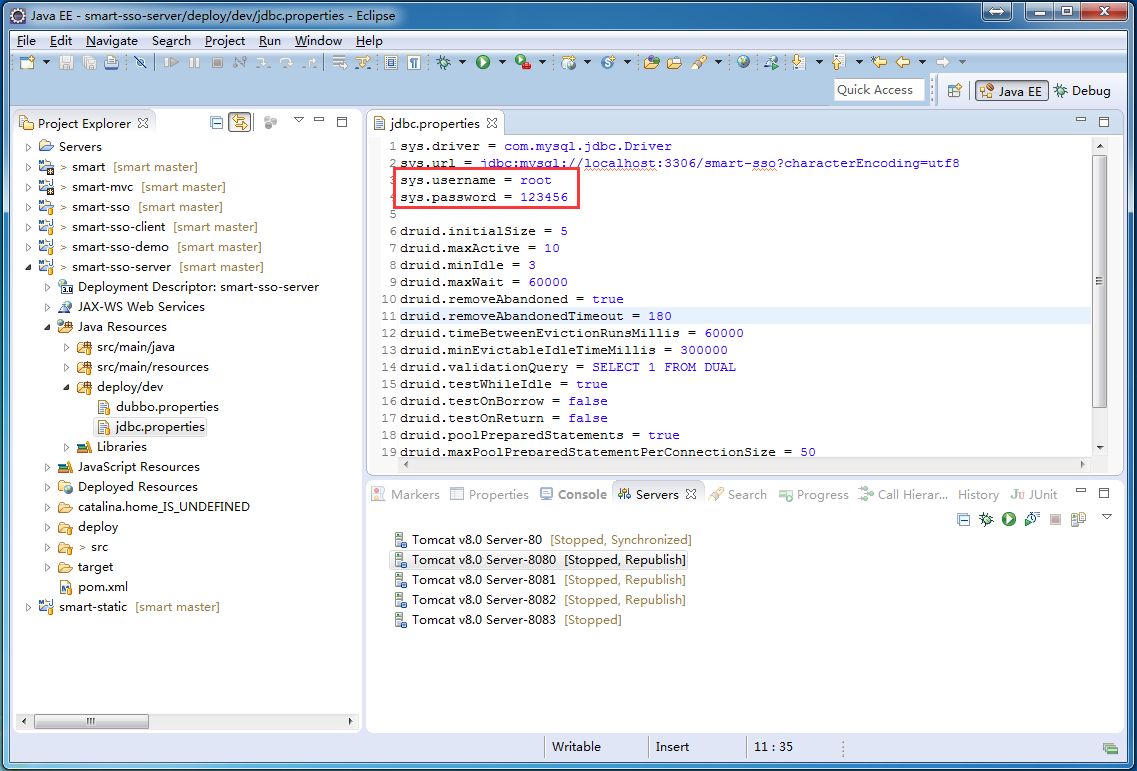


1. smart-sso-server服务端需要连接mysql数据库。如下图，找到对应的sql文件，并创建名称为smart-sso的数据库，并运行sql文件。





1. 打开smart-sso-server项目中jdbc.properties文件，修改数据库连接配置。如下图所示

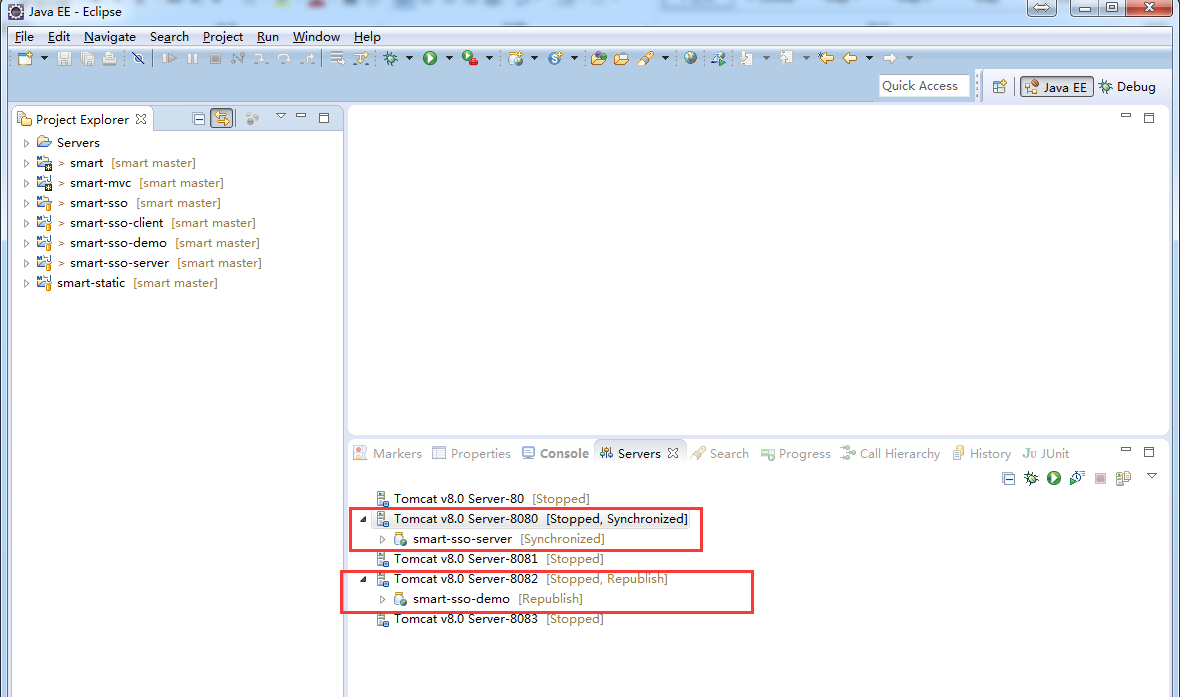


1. 下面需要启动两个tomcat实例。

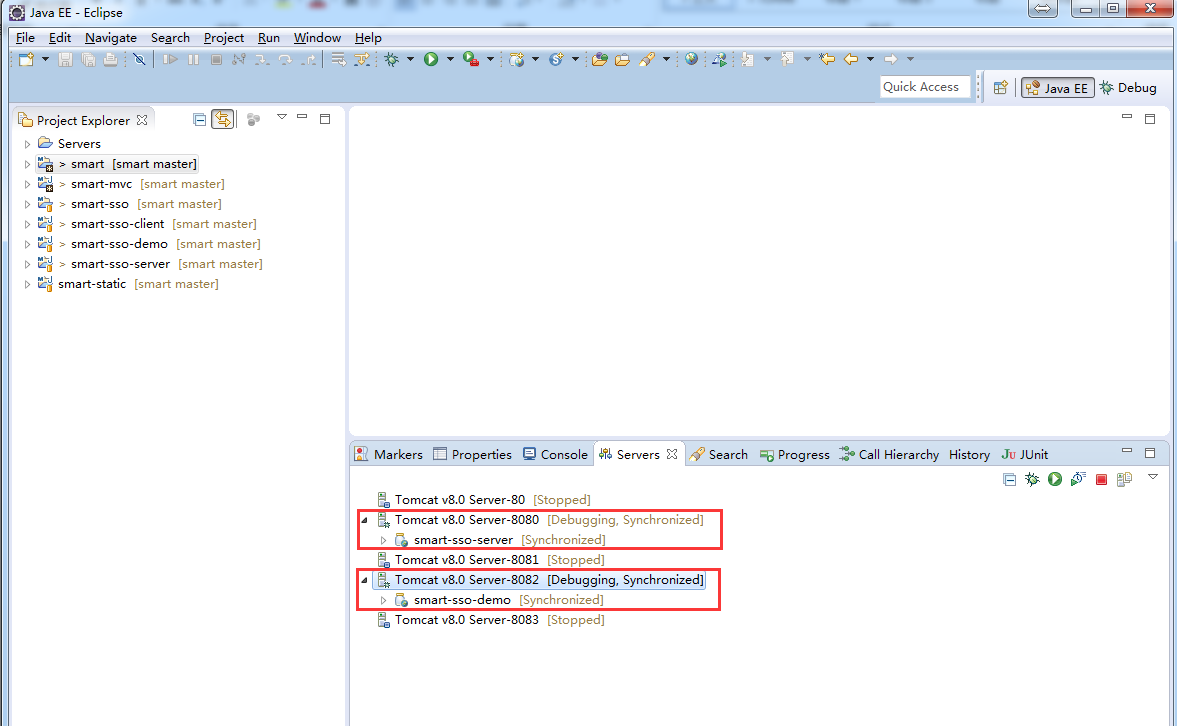
如果你是熟悉maven容器插件的方式，直接可以分别在项目上使用tomcat7:run或jetty:run。启动顺序必须是先smart-sso-server后smart-sso-demo。

* 1. 把smart-sso-server加入到8080端口的tomcat，并启动成功。
  2. 把smart-sso-demo加入到8082端口的tomcat，并启动成功。

如下图所示：



1. 启动成功后如下图所示：

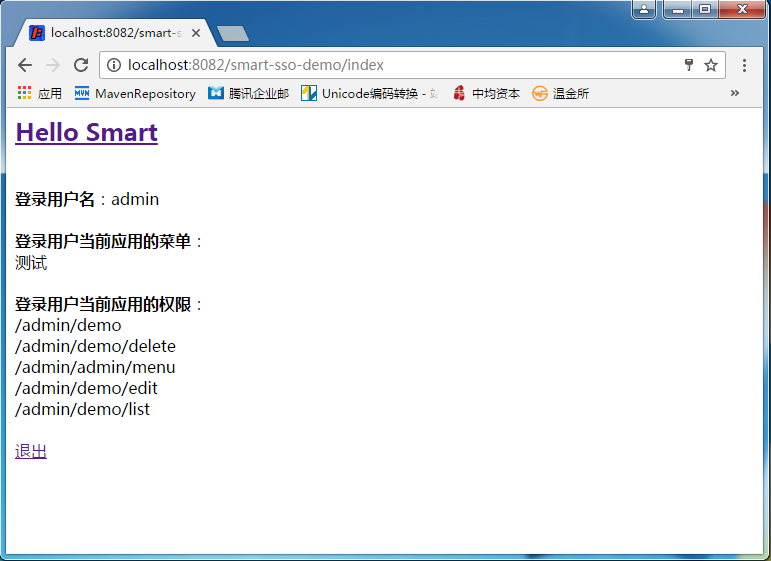


1. 服务启动正常，就可以在浏览器试访问smart-sso-server服务端了。在浏览器输入地址：http://localhost:8080/smart-sso-server，回车会跳转登录页，默认的账号密码是：admin 123456，登录成功后会进入到服务端首页。如下图所示：





1. 在smart中，服务端其实也是它自身的一个客户端，且上一步操作已登录成功了。接下来在浏览器输入地址：<http://localhost:8082/smart-sso-demo>，会直接进入到smart-sso-demo首页。看到下图，就表示配置成功了，并实现了单点登录及菜单和权限的获取。

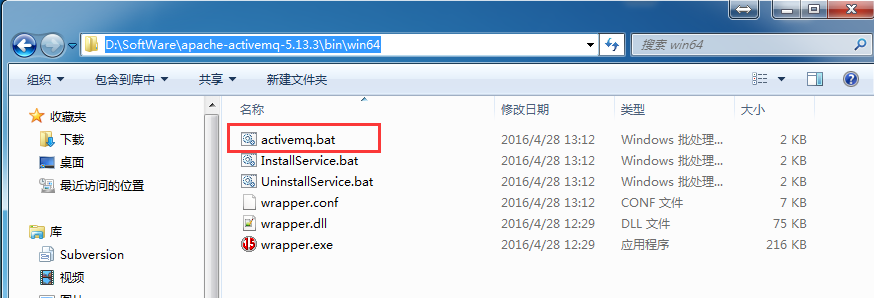


**选配软件配置**：

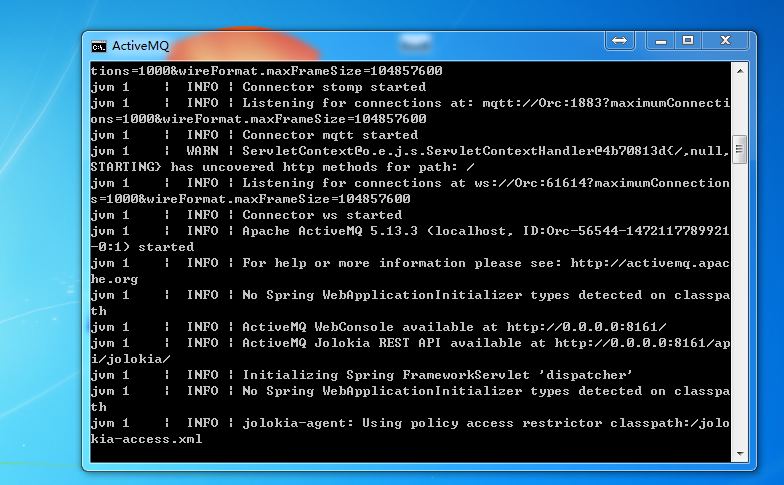
1. ActiveMQ

注：activeMQ在smart中主要用于服务端权限发生修改变动时，异步通知各个子应用更新权限，在子应用不需要重启的情况下，实现权限修改的实时生效。

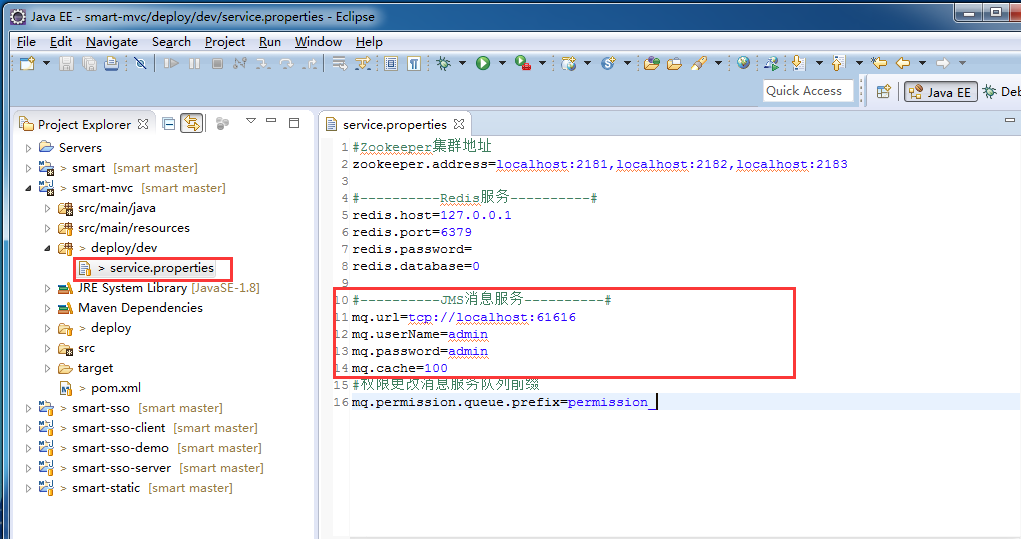
1. 官网下载地址：<http://activemq.apache.org/download-archives.html>，也可以在QQ群文件里下载笔者使用的apache-activemq-5.13.3版本。下载完成后解压，并找到bin路径，双击activemq.bat启动,如下图所示：



1. 双击后，保持如下窗口，证明启动成功了（如果出现闪退等情况，请检查activeMQ默认的：8161端口是否被占用）：

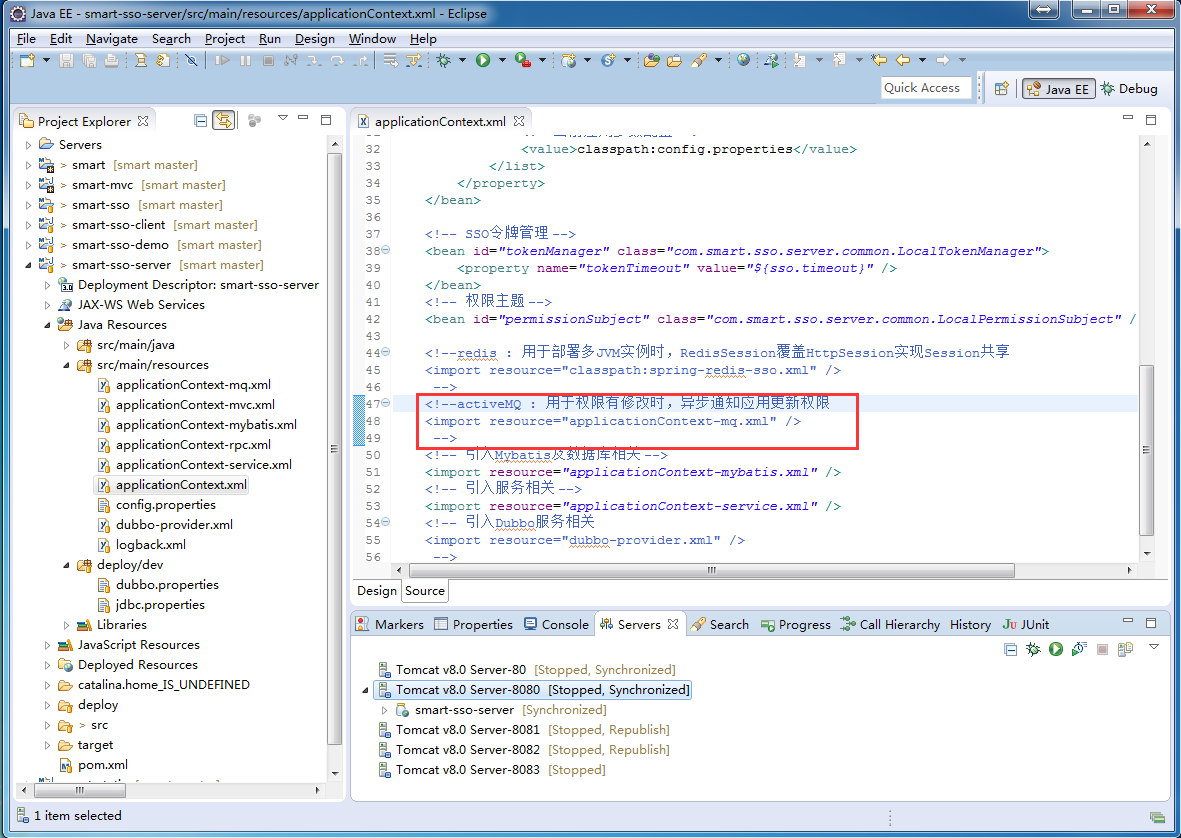


1. 打开smart-mvc中service.properties，修改activeMQ连接信息

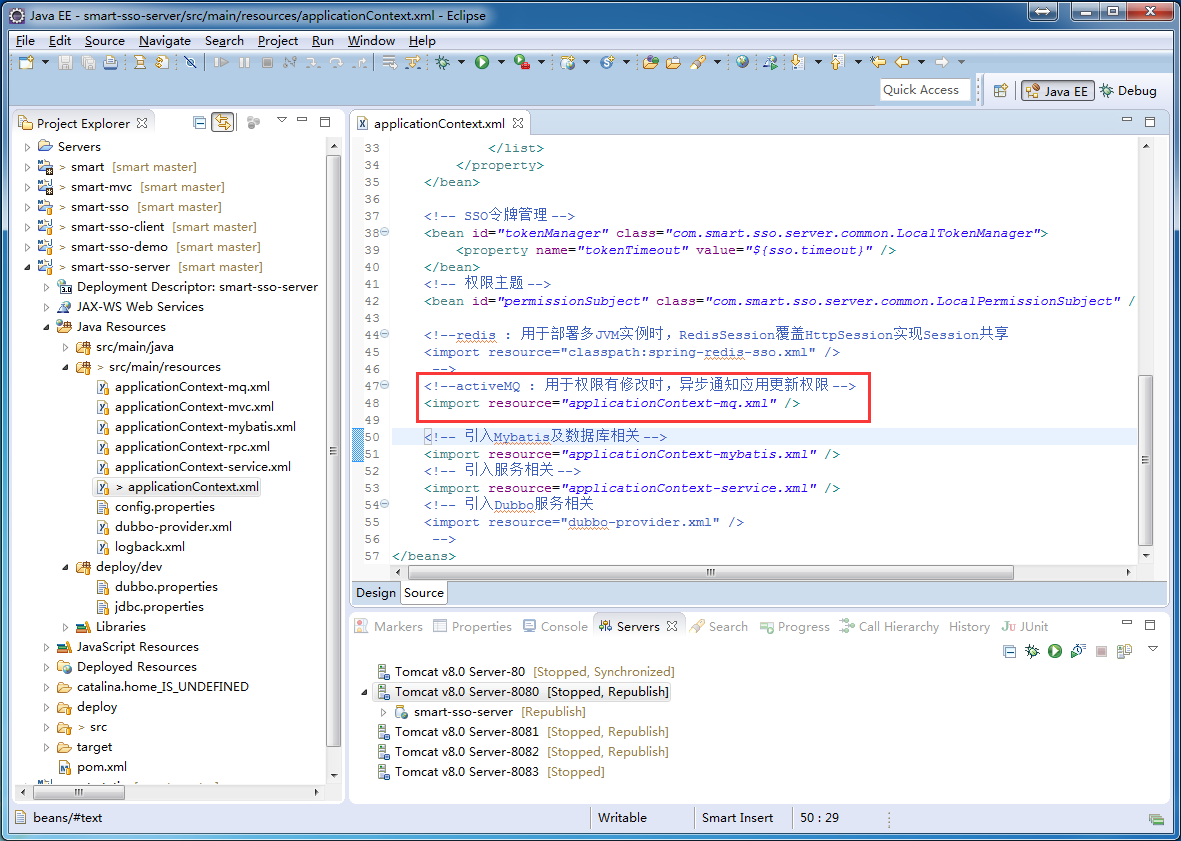


1. 打开smart-sso-server中applicationContext.xml配置文件, 去掉<import resource="applicationContext-mq.xml" />注释，如下图所示：

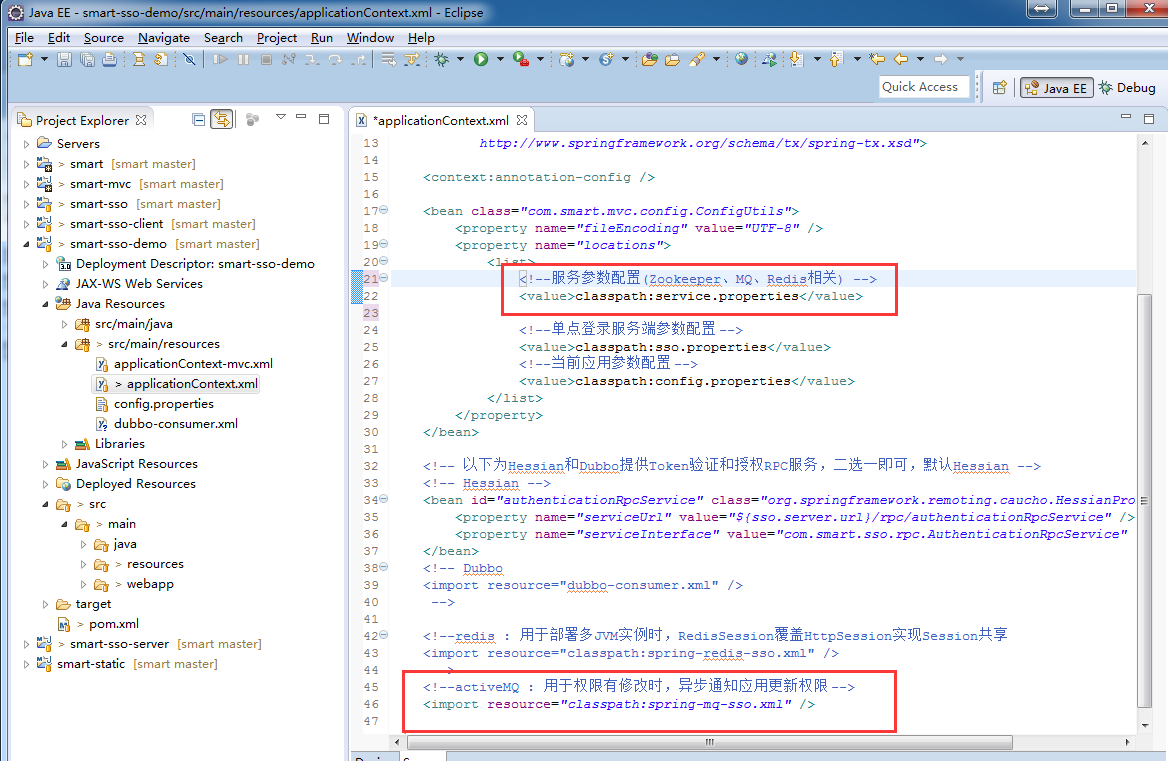
修改前：



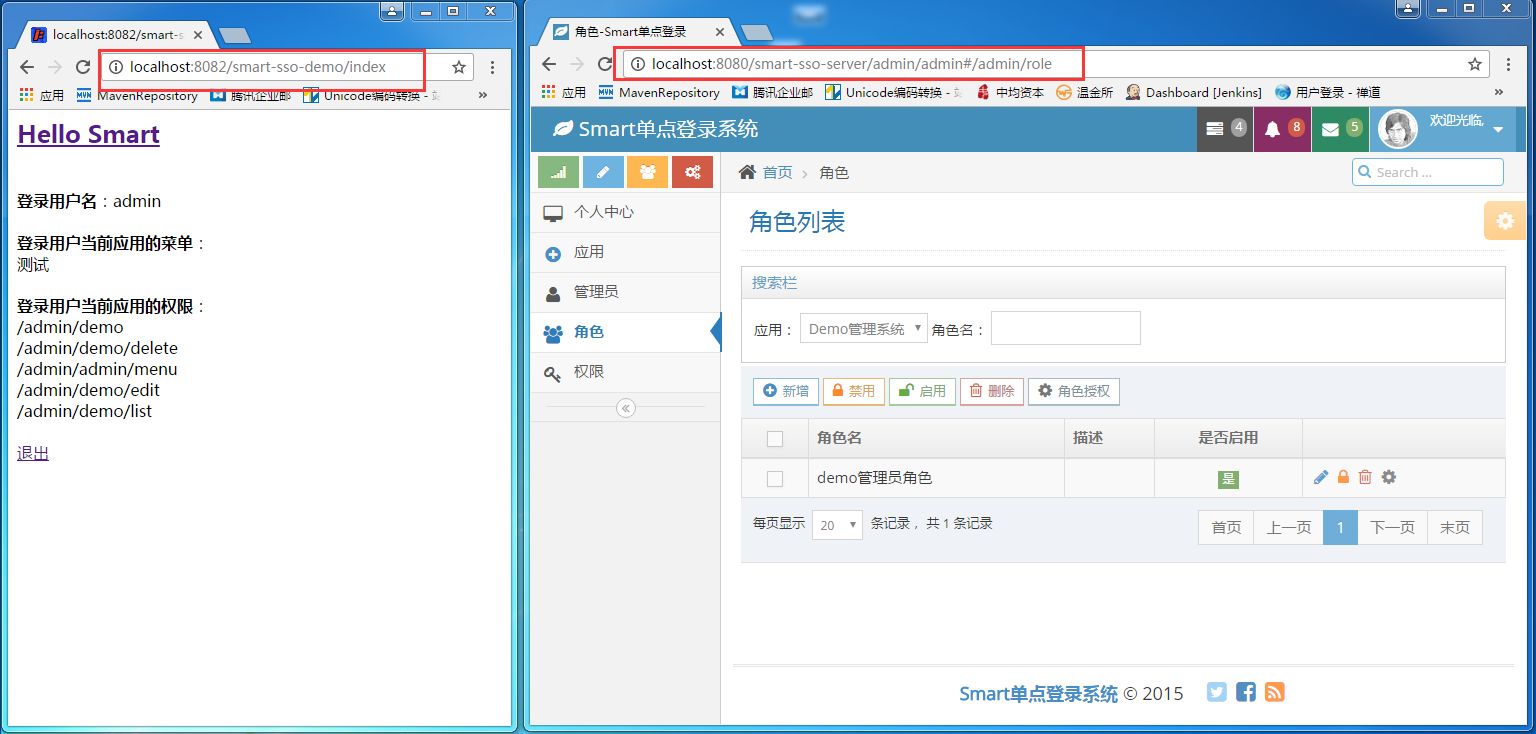
修改后：



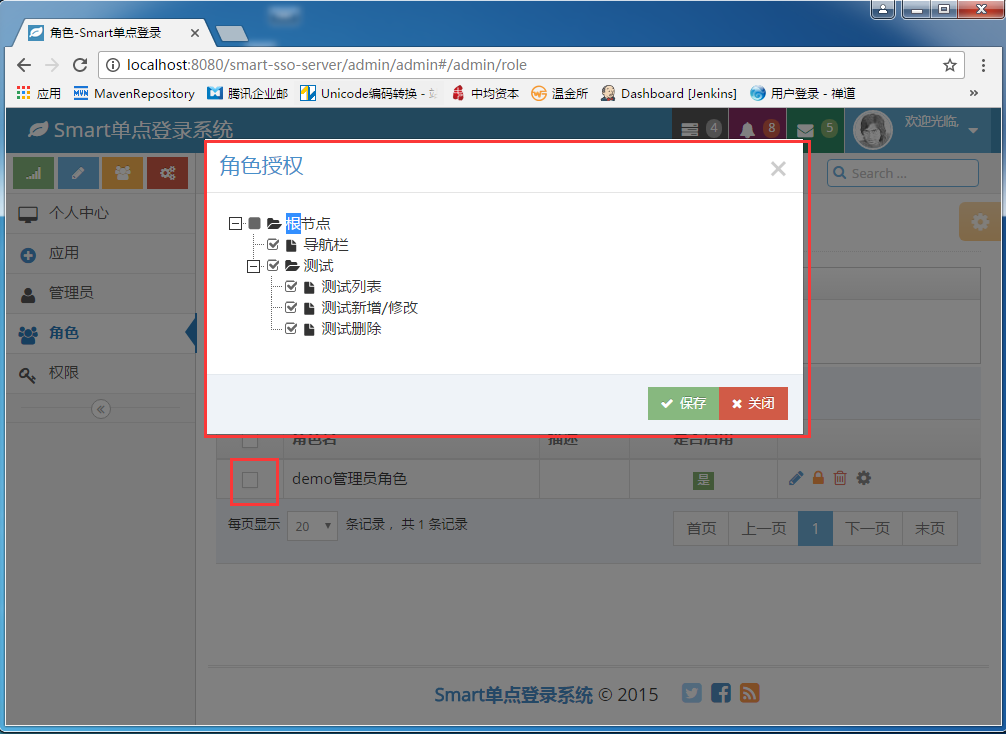
1. 重启项目，没有报错就表示配置成功。
2. 同样，在客户端smart-sso-demo中applicationContext.xml文件去掉下图红色框两处注释，并重启。



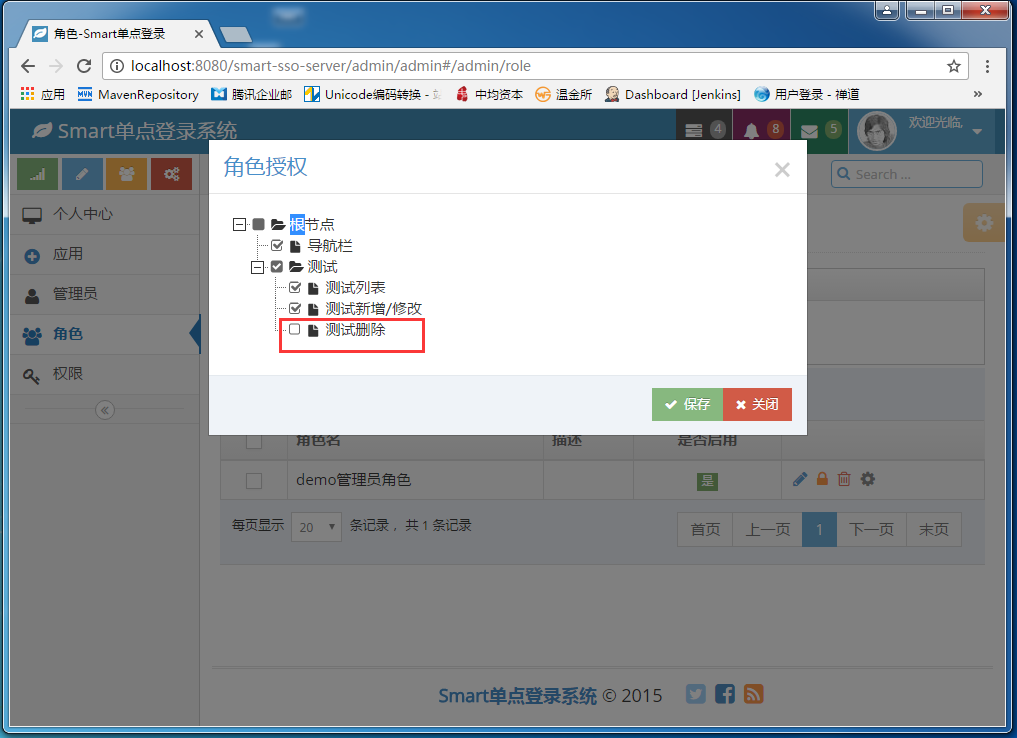
1. 接下来可以做如下测试，开两个窗口用admin 123456登录，分别访问客户端和服务端首页，如下图所示：



1. 定位到服务端“角色”菜单，选中表格中角色，并点击“角色授权”按钮。会显示当前admin对应的角色在客户端应用的所有权限。如下图所示：

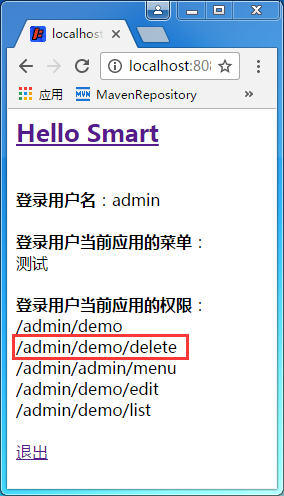


1. 去掉勾选其中的“测试删除”权限，并保存。



1. 客户端请求地址在浏览器中刷新一下，可以看到如下结果，就表示权限的修改实时生效了：

刷新前：



刷新后：

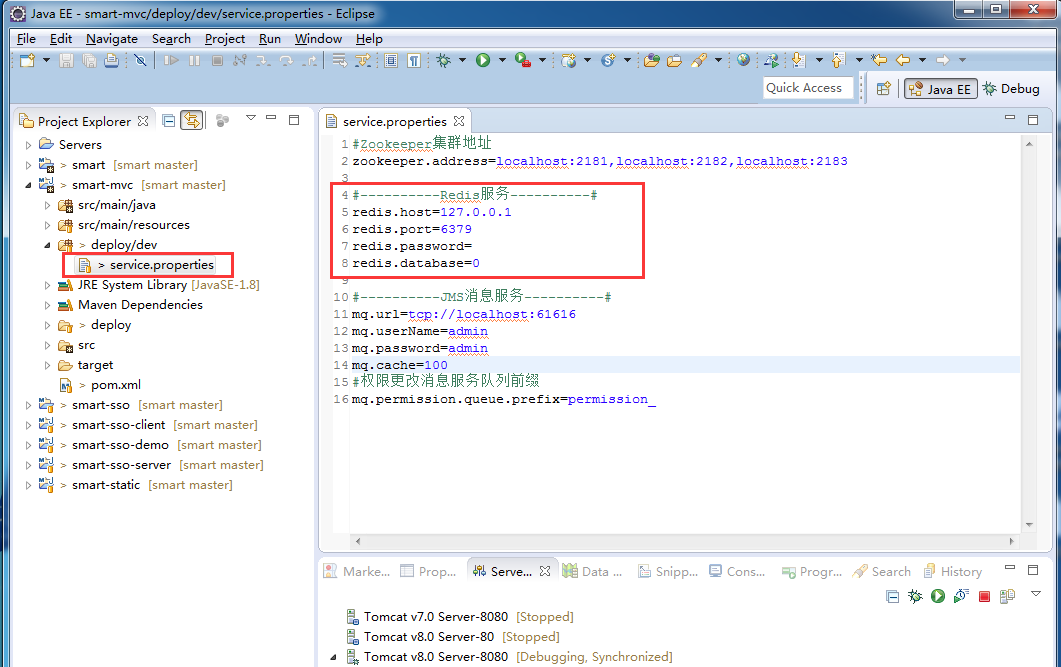


1. Redis

注：redis在smart中的作用主要是用于结合spring覆盖HttpSession,实现分布式session。

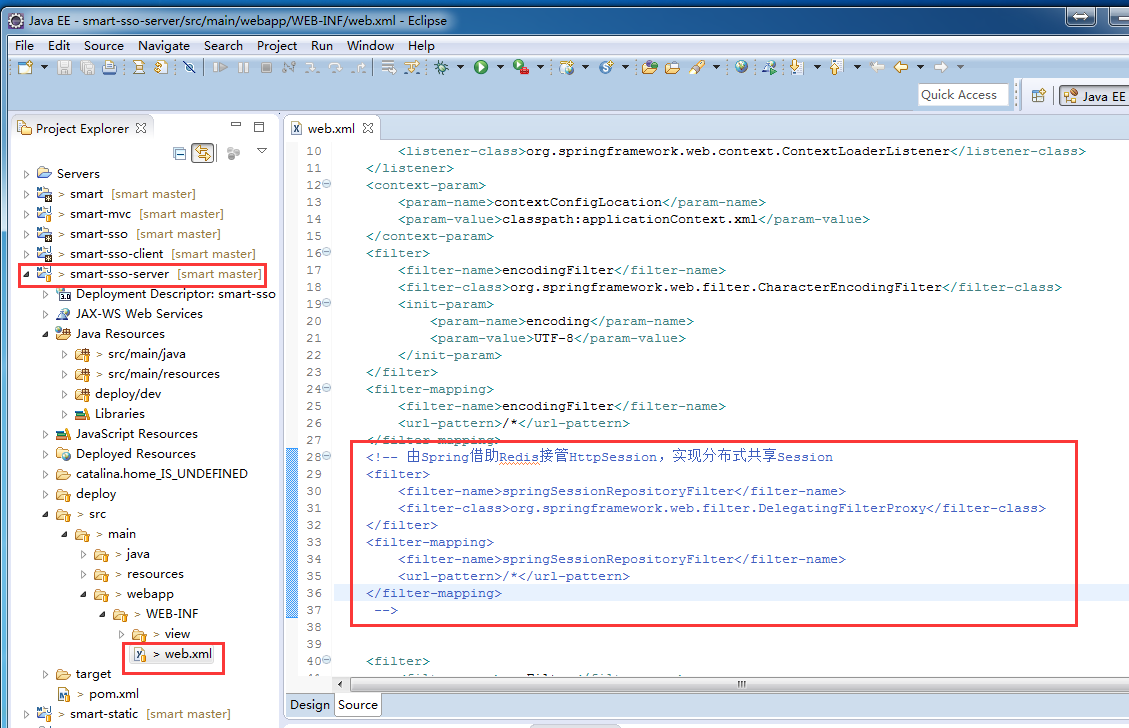
Redis的安装配置，这里就不描述。如下配置请保持redis服务已正常启动。

1. 打开smart-mvc中service.properties，修改redis连接信息

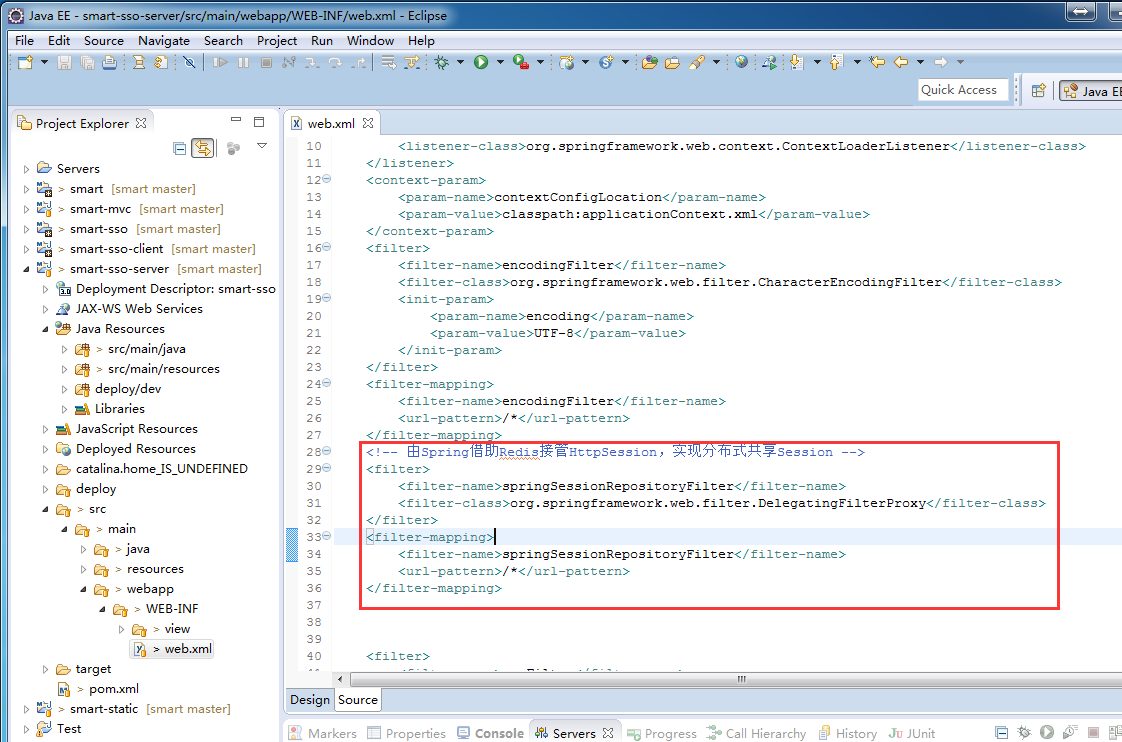


1. 打开smart-sso-server中web.xml,去掉Spring拦截session注释，如下图所示：

修改前：



修改后：

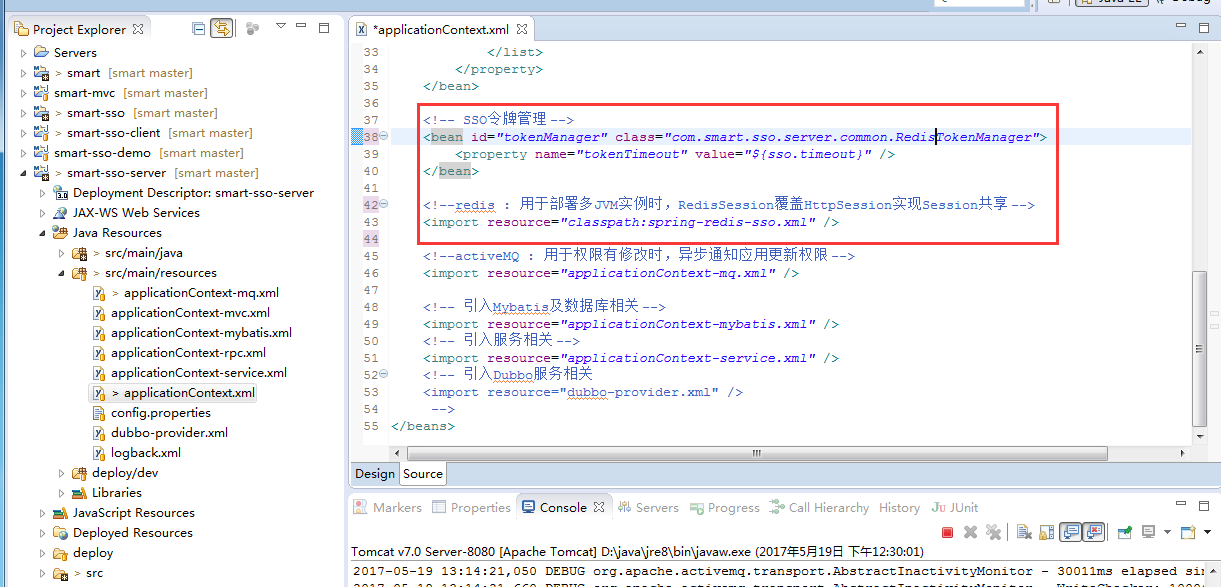


1. 打开smart-sso-server中applicationContext.xml配置文件,修改下图所示红色框住的两处。
2. 把LocalTokenManager改为RedisTokenManager
3. 去掉<import resource="classpath:spring-redis-sso.xml" />注释

修改前：



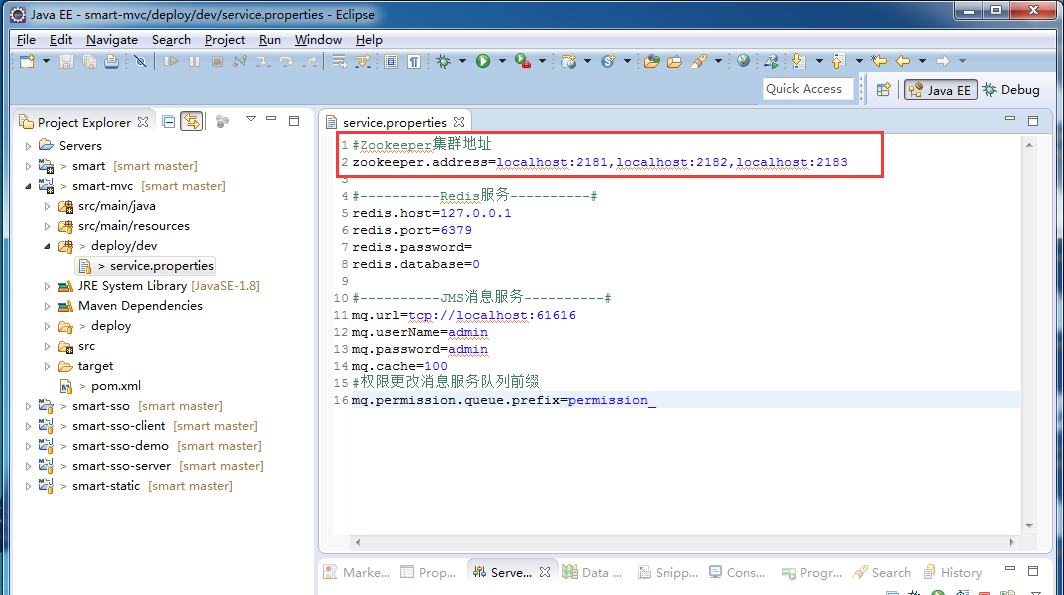
修改后：



1. 重启项目，没有报错就表示配置成功。也就是说现在支持启动多个smart-sso-server实例，借助nginx反向代理到这些实例，可实现高可用负载均衡。
2. Dubbo

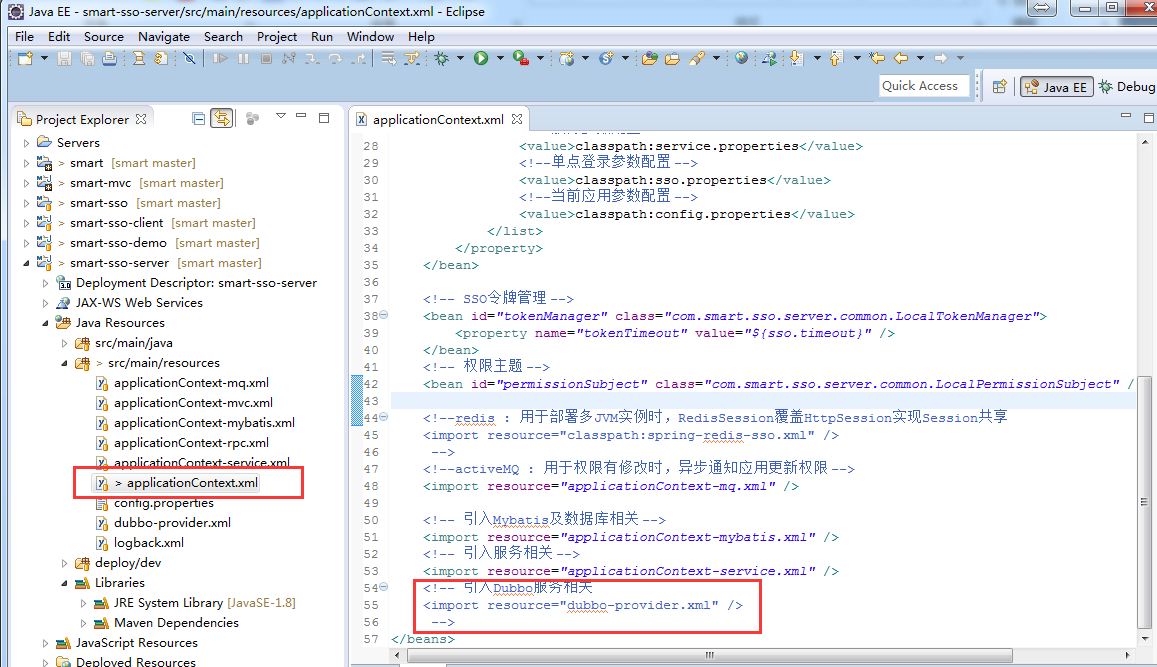
注：smart支持把默认的RPC服务Hessian改为Dubbo，配置以下的操作之前，必须保证zookeeper的服务已安装并启动成功。

1. 打开smart-mvc中service.properties，修改zookeeper连接信息

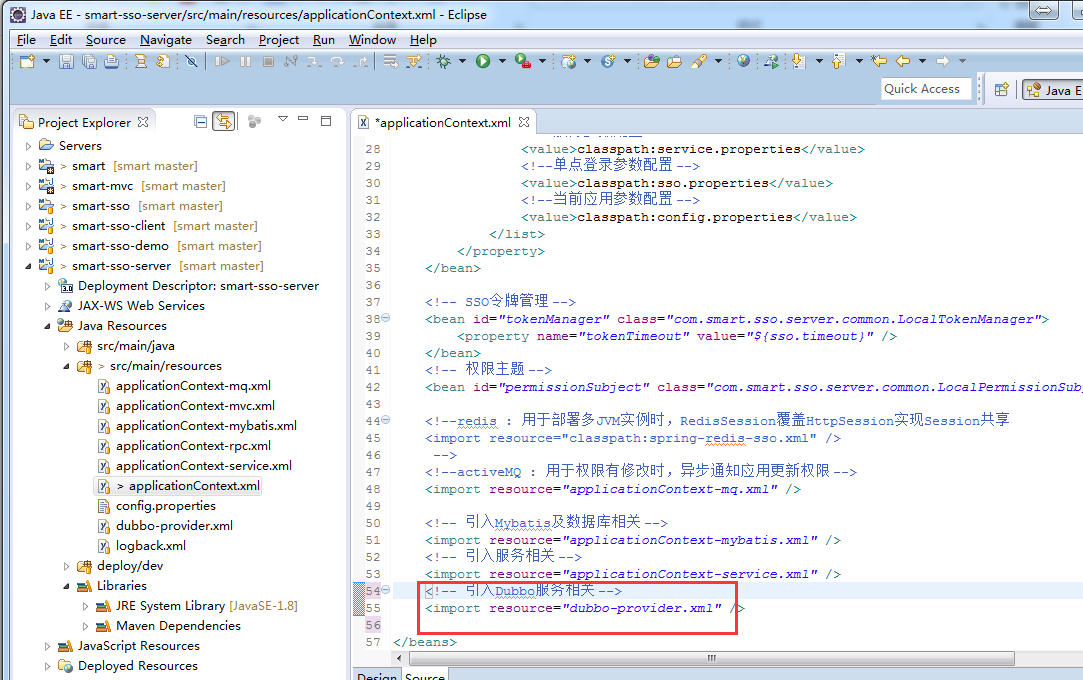


1. 打开smart-sso-server中applicationContext.xml配置文件，去掉dubbo服务注释，并重启成功。

修改前：

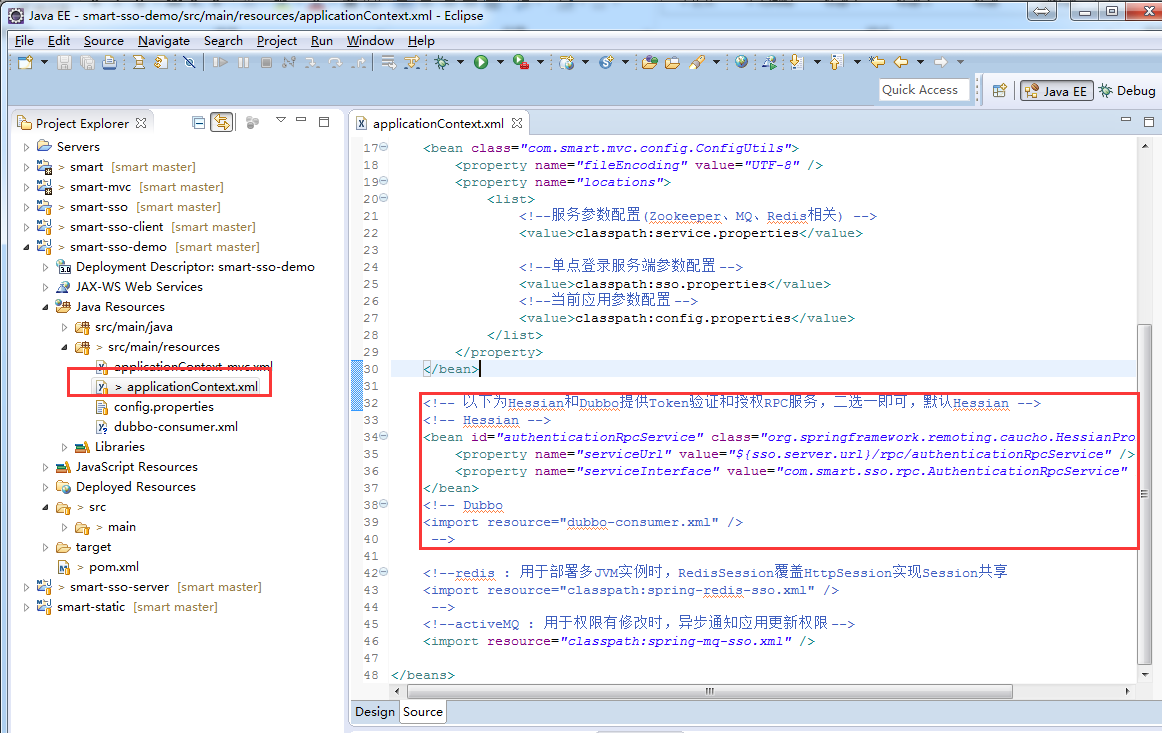


修改后：

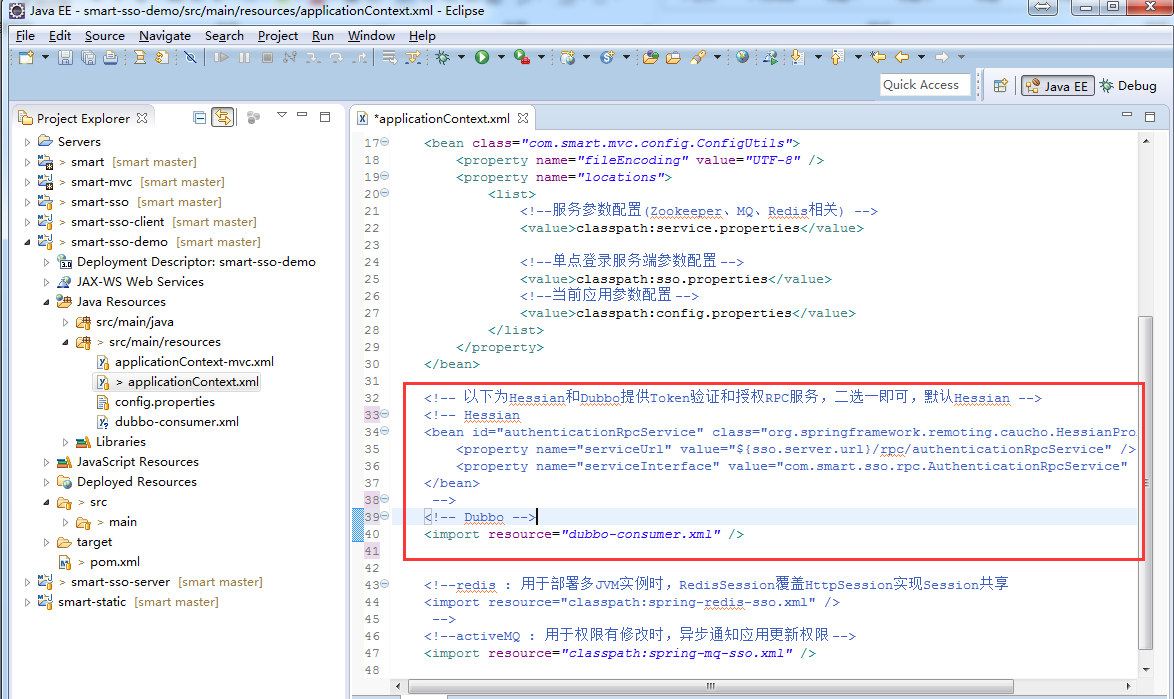


1. 打开smart-sso-demo中applicationContext.xml配置文件，去掉dubbo服务注释，并注释掉Hessian相关配置。

修改前：



修改后：



1. 之后可以使用dubbo-admin工程查看服务端和客户端是不是都启动正常。

