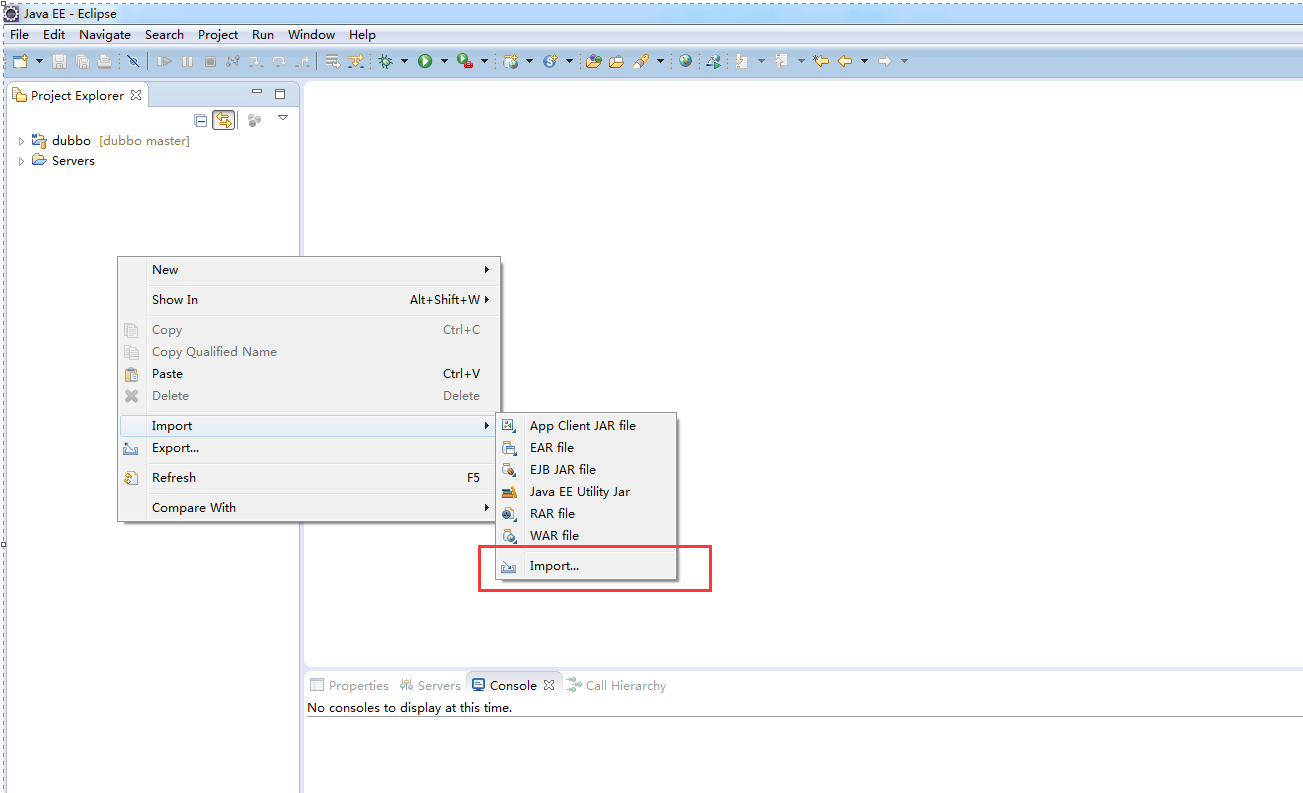
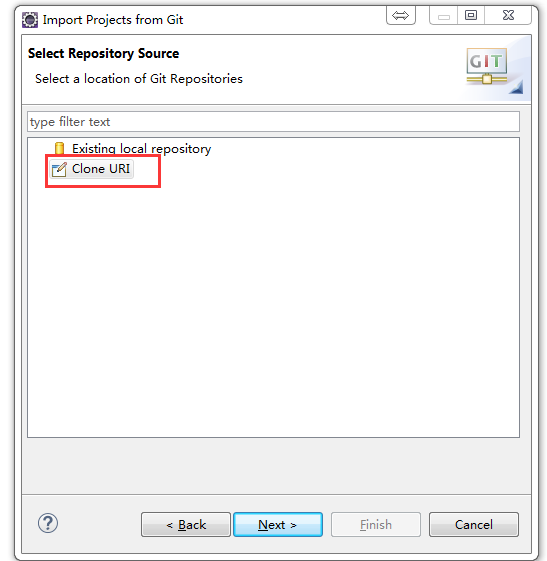
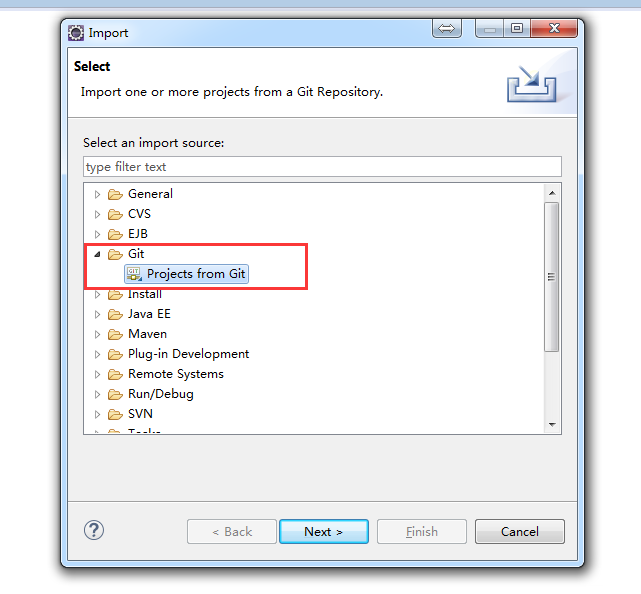
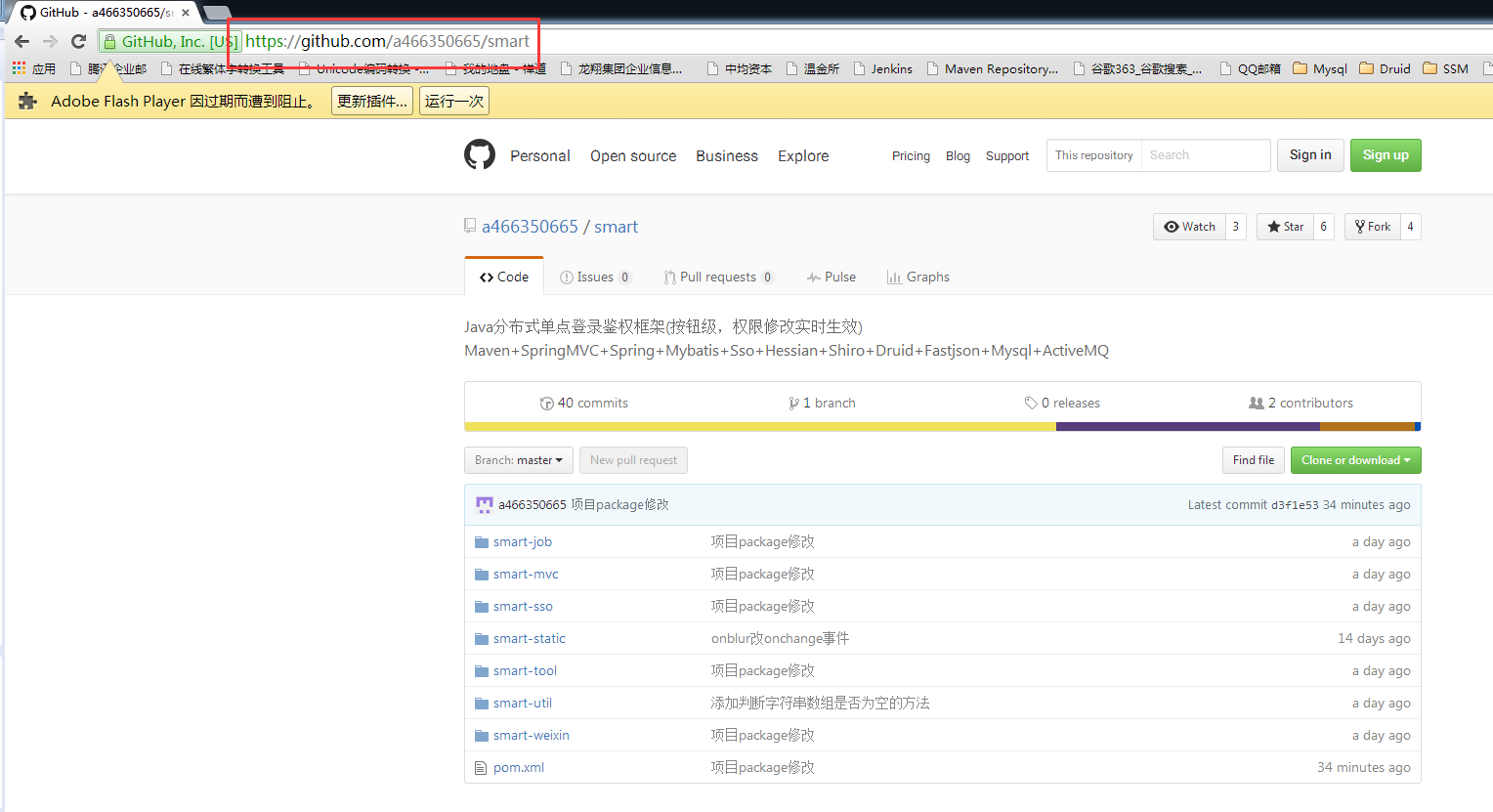
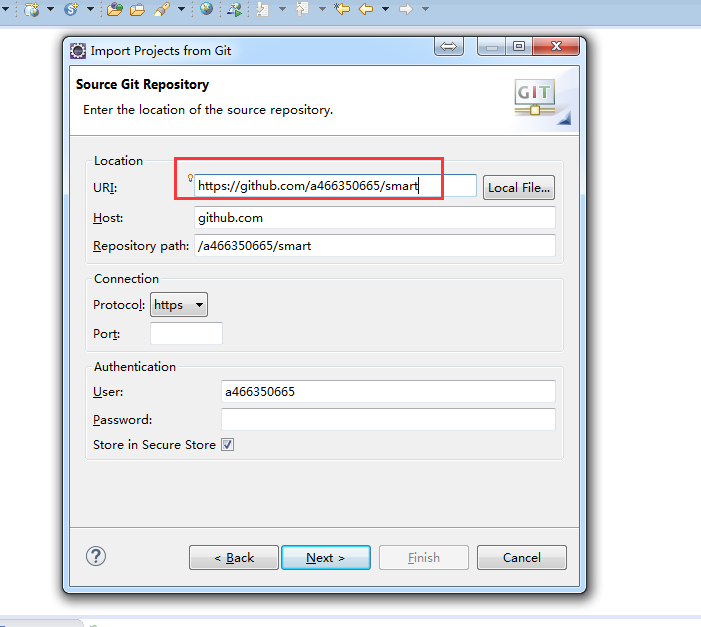
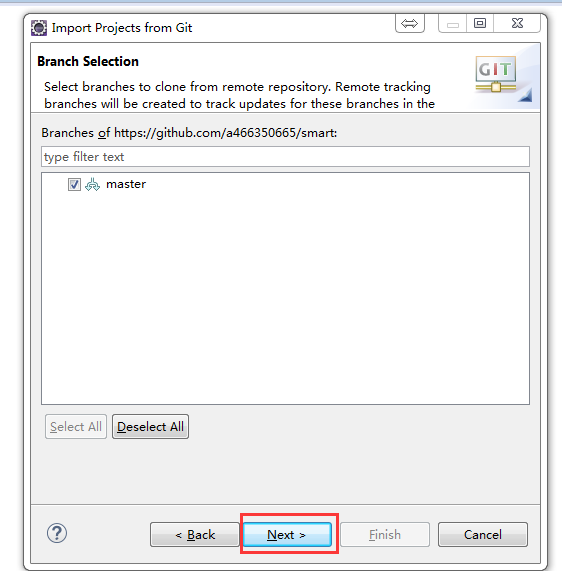
单点登录权限系统服务端

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）。QQ交流群：454343484(群内提供开发工具和文档下载)。这些是基本环境，下面开始下载配置操作：

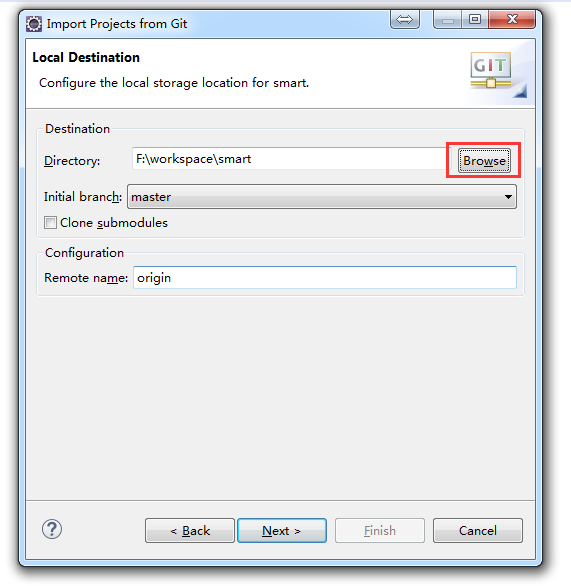
1. 从Github下载Smart项目（https://github.com/a466350665/smart）
2. 右击“Import🡪 Import”，选择“Git->Projects from Git”，再 
3. 选择“Clone URI”, 再  
4. 从github复制项目链接: https://github.com/a466350665/smart，并粘贴到“URI”上, 再



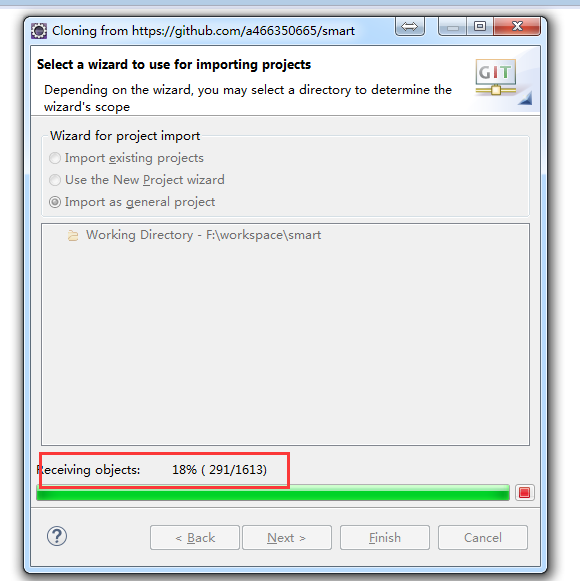


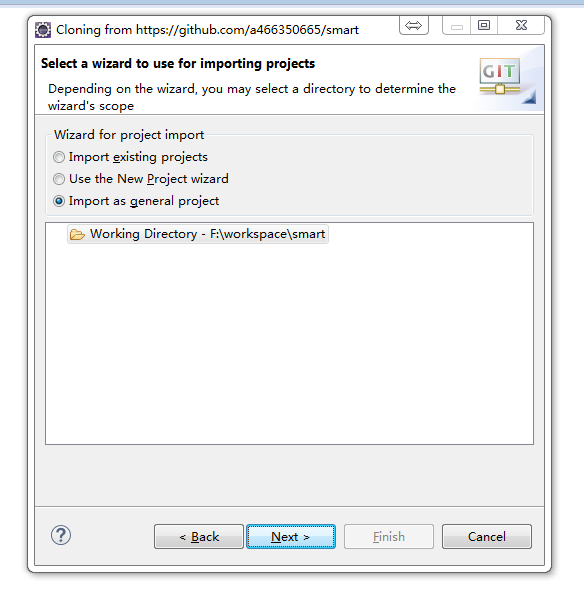


1. 选择本地存放smart项目的目录，再

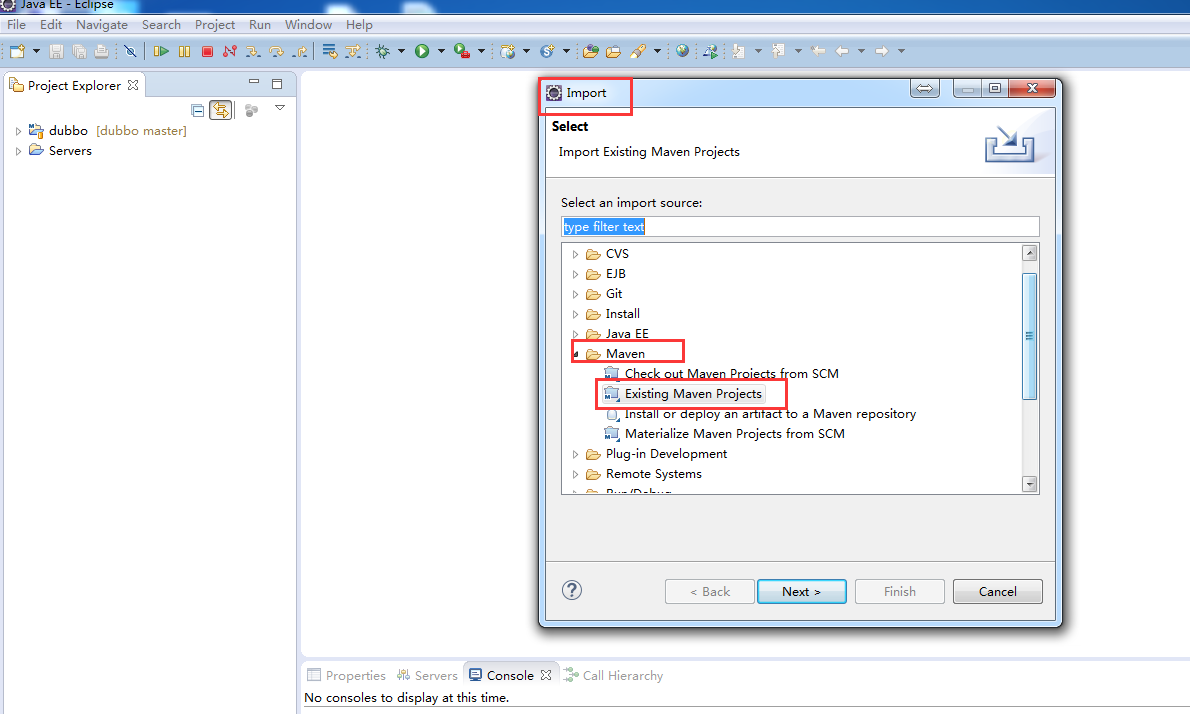


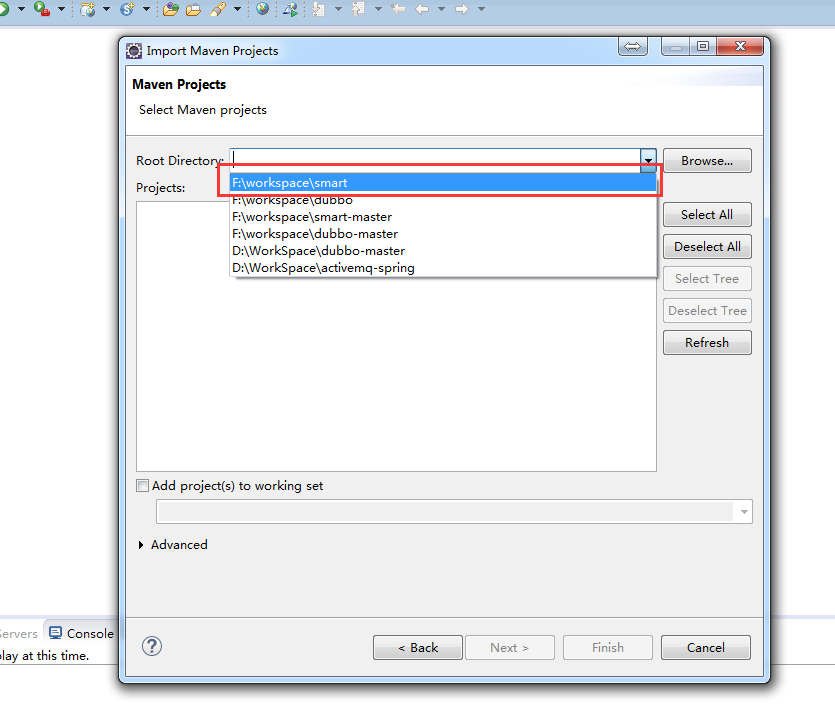
1. 看到如下第二张图就是导入成功了。 “Import as general project”提示让导入并生成项目,也可以直接掉窗口，因为项目已下载到本地。

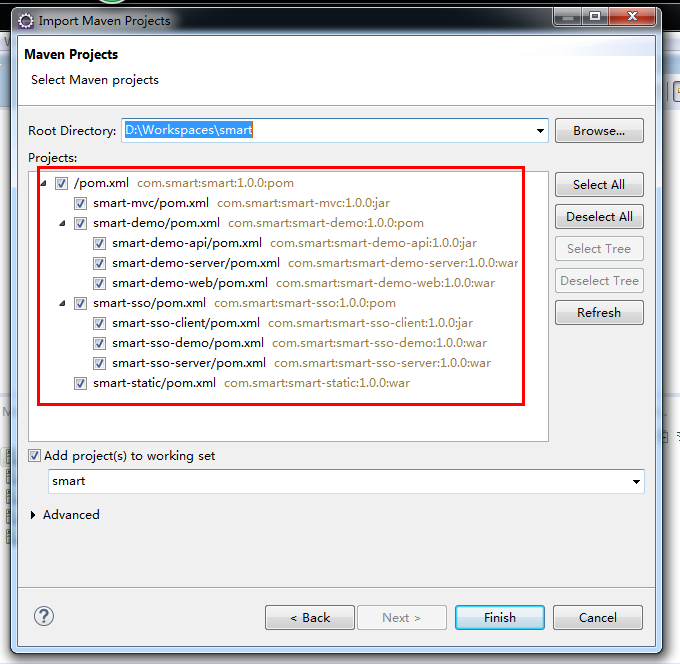




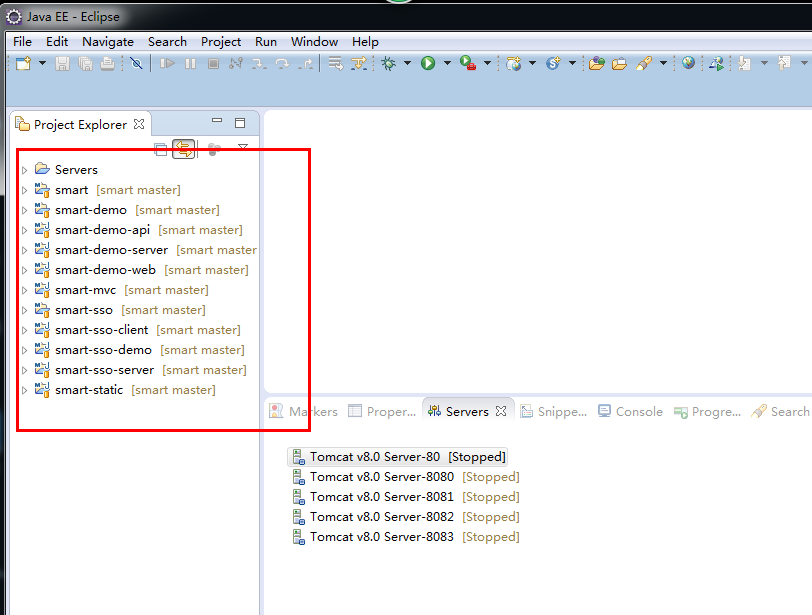
1. 接着就是以引入maven项目的方式，把smart导入到eclipse. 右击“Import🡪 Import”，选择“Maven->Existing Maven Projects”, 再



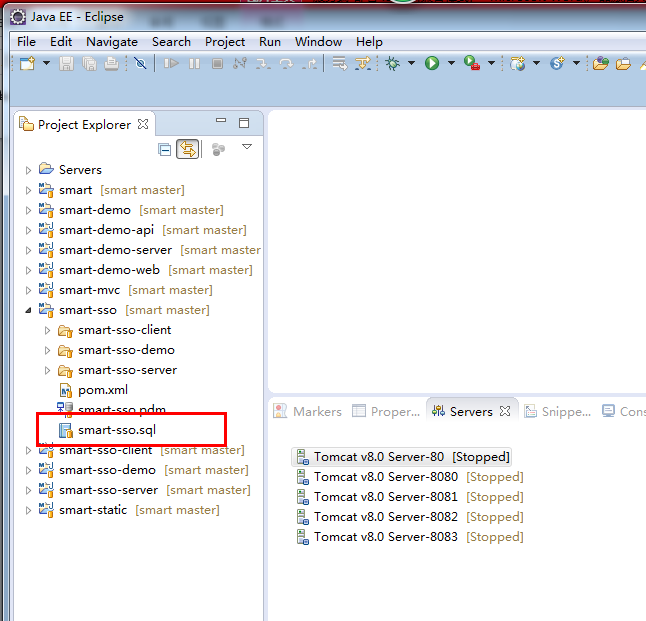
1. 选择下载好的Smart项目本地存放路径，点击

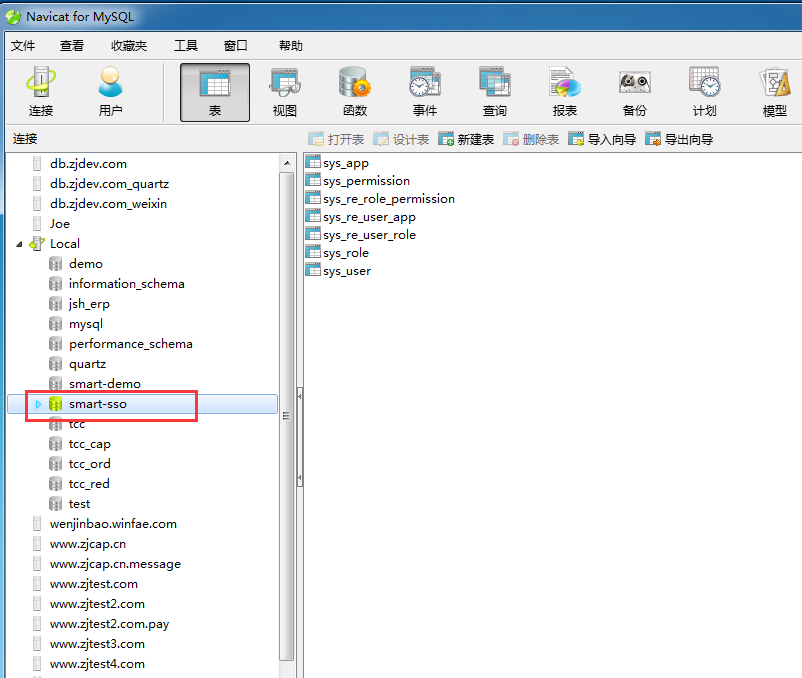


1. 第一次导入，maven需要下载smart所需要的Jar文件，会等待比较长的时间，下载完成后如下图所示：

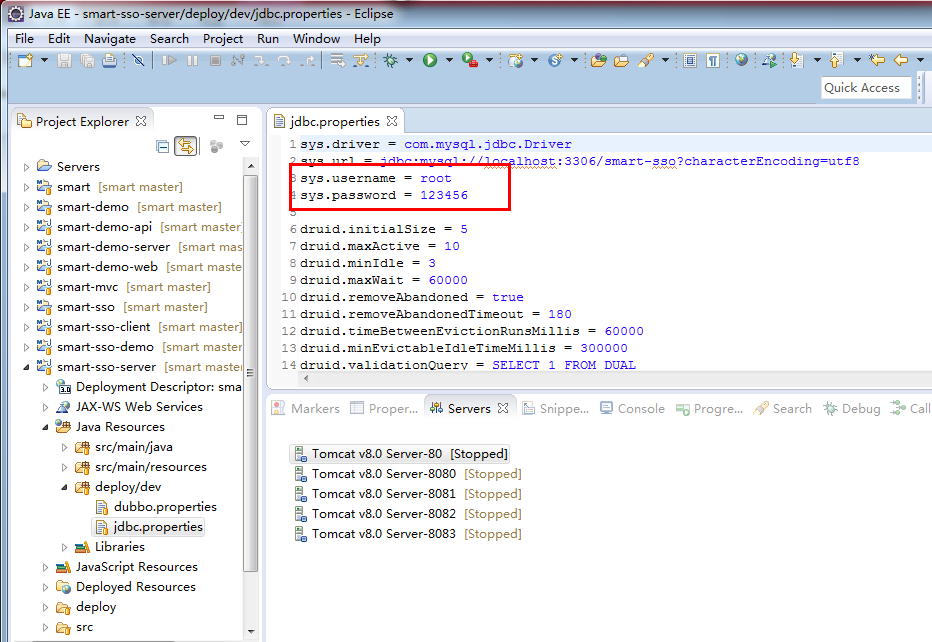


1. smart-sso-server服务端需要连接mysql数据库。如下图，找到对应的sql文件，并创建名称为smart-sso的数据库，并运行sql文件。





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



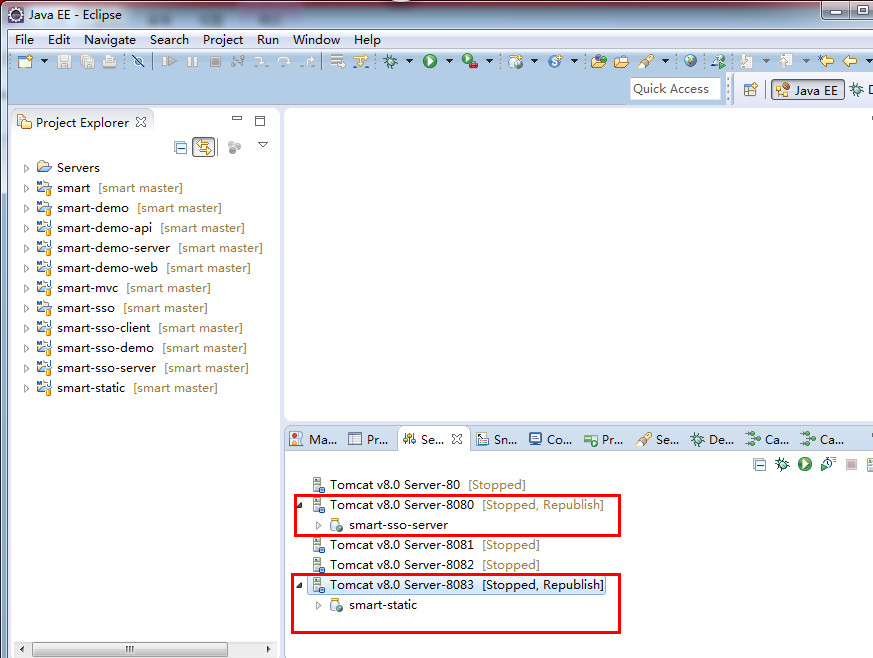
1. 下一步你需要配置两个tomcat实例:

注：启动之前务必保证Mysql服务已经启动成功，如果你是习惯maven插件的形式启动，可以直接在如下两个项目上分别使用tomcat7:run或jetty:run。

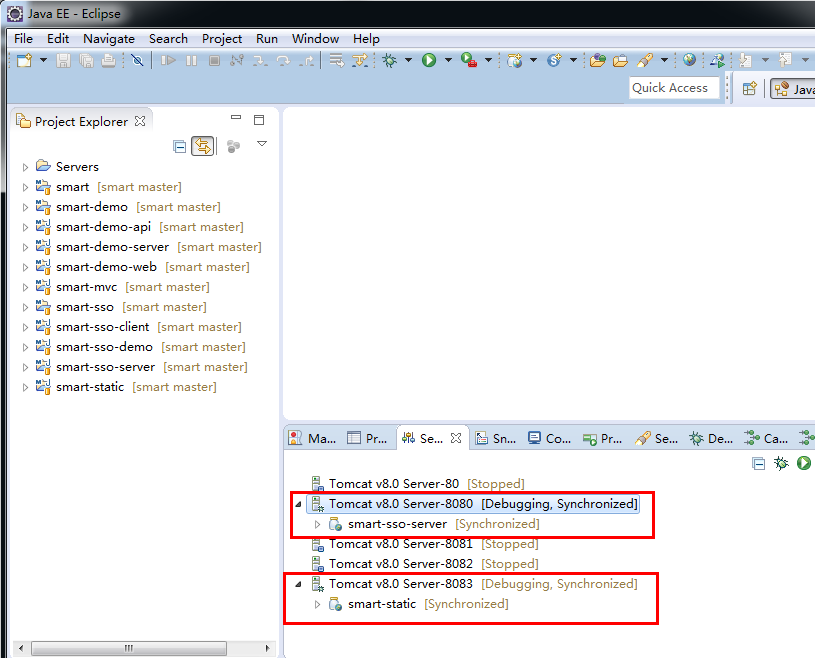
非Maven插件启动步骤如下：

1. 把smart-static(公用静态资源)加入到8083端口的tomcat实例
2. 把smart-sso-server（单点登录权限系统服务端）加入到8080端口的tomcat实例

如下图所示：



1. 启动成功后如下图所示：



1. 服务启动正确，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smart-sso-server系统。如下图所示：





1. 接下来你可以根据部署文档配置客户端程序了。客户端提供了两个版本的:

smart-sso-demo(极简版)：相当于一个入门hello world，部署参考《客户端（极简版）-部署.docx》

smart-demo(进阶版)：涵盖zookeeper、dubbo等分布式框架，是smart在分布式场景的应用，部署参考《客户端（进阶版）-部署.docx》

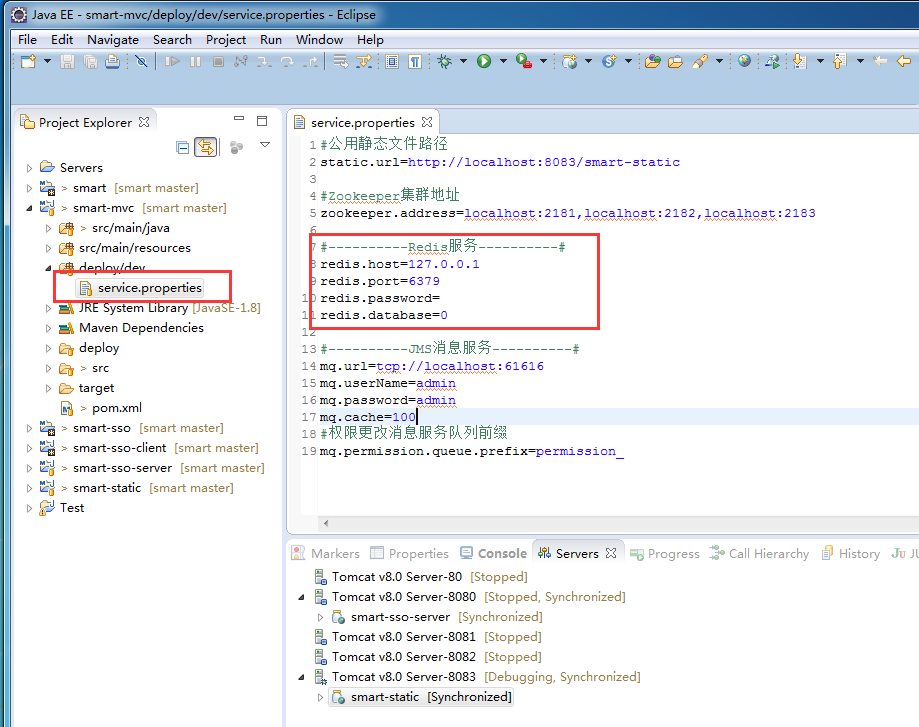
**选配软件配置**：

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。

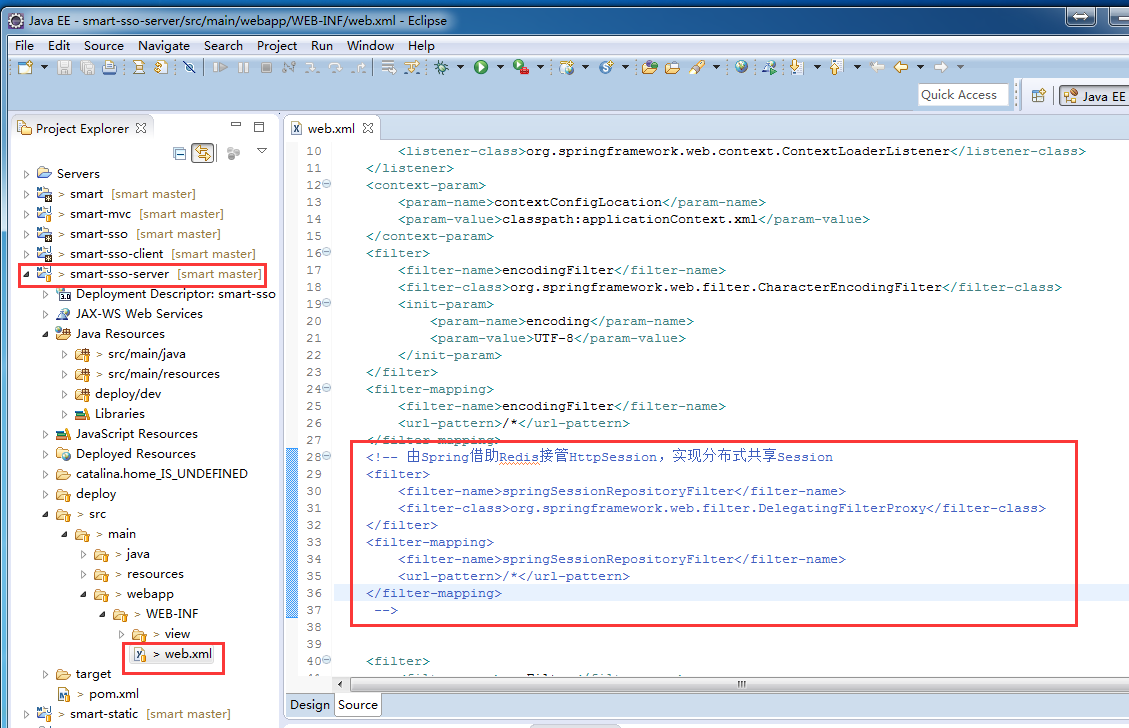
Redis的安装配置，这里就不描述。如下配置请保持redis服务已正常启动。

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

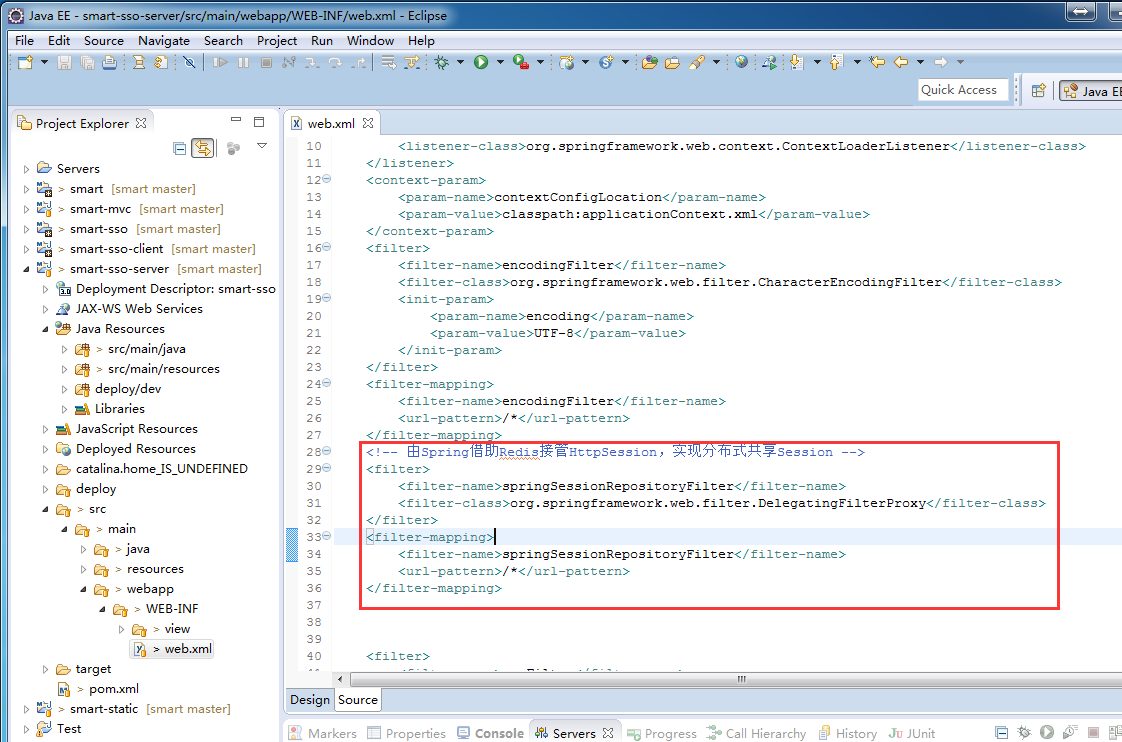


1. 打开smart-sso-server中web.xml,去掉Spring拦截session注释，如下图所示：

修改前：

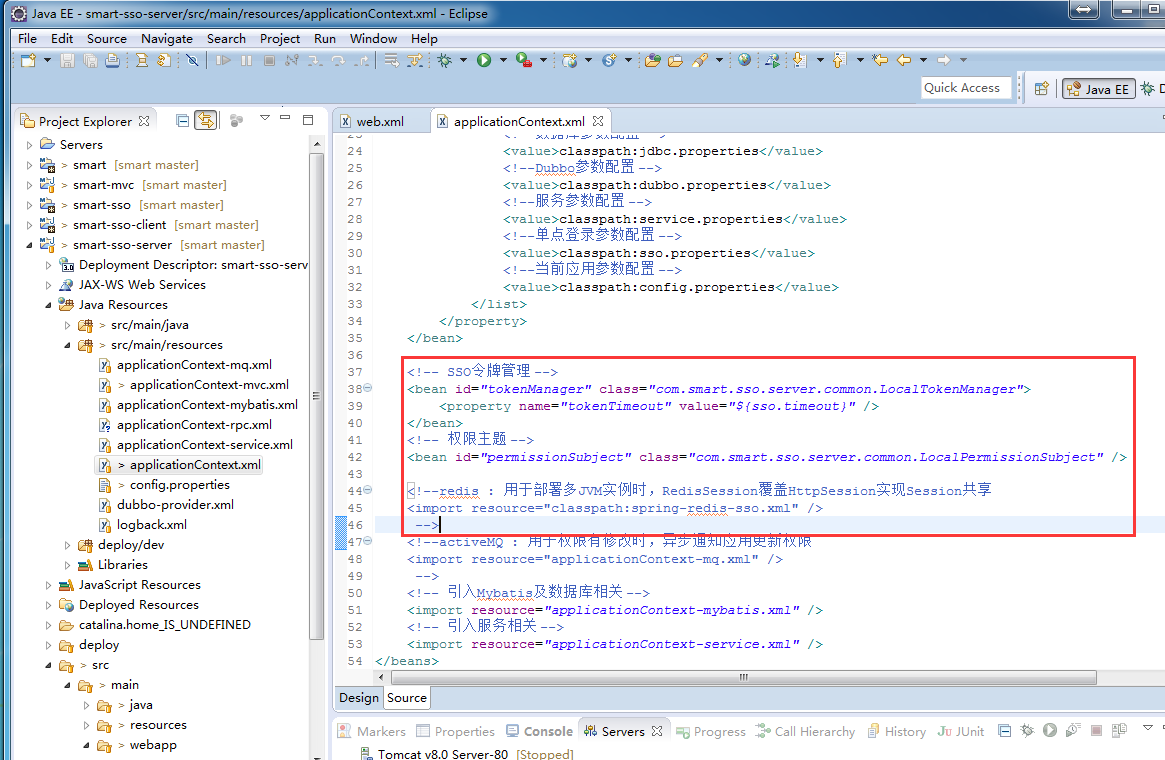


修改后：

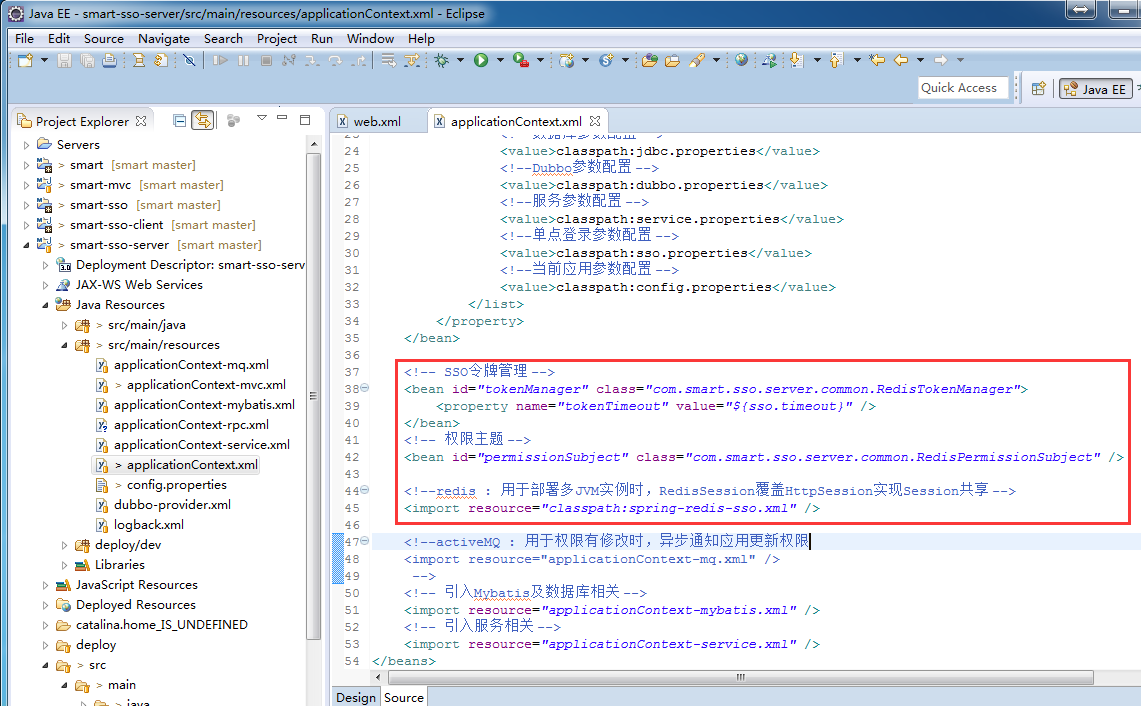


1. 打开smart-sso-server中applicationContext.xml配置文件,修改下图所示红色框住的三处。
2. 把LocalTokenManager改为RedisTokenManager
3. 把LocalPermissionSubject改为RedisPermissionSubject
4. 去掉<import resource="classpath:spring-redis-sso.xml" />注释

修改前：



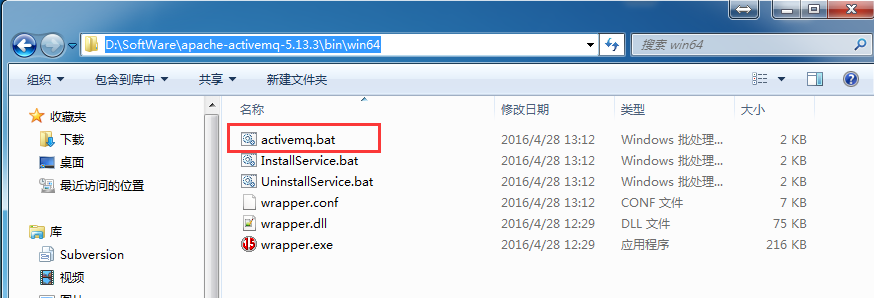
修改后：



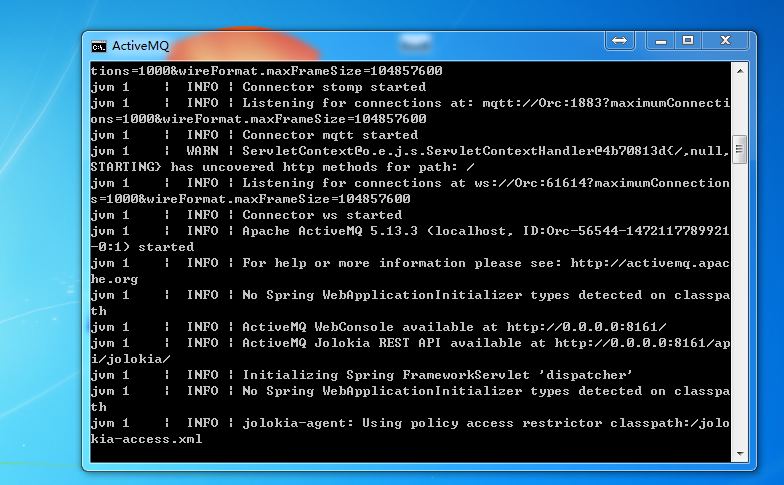
1. 重启项目，没有报错就表示配置成功。也就是说现在支持启动多个smart-sso-server实例，借助nginx反向代理到这些实例，可实现高可用负载均衡。
2. ActiveMQ

注：activeMQ在smart中重要用于单点登录权限系统服务端，权限发生变动修改时，异步通知各个单点登录子系统更新权限，实现权限修改的实时性。

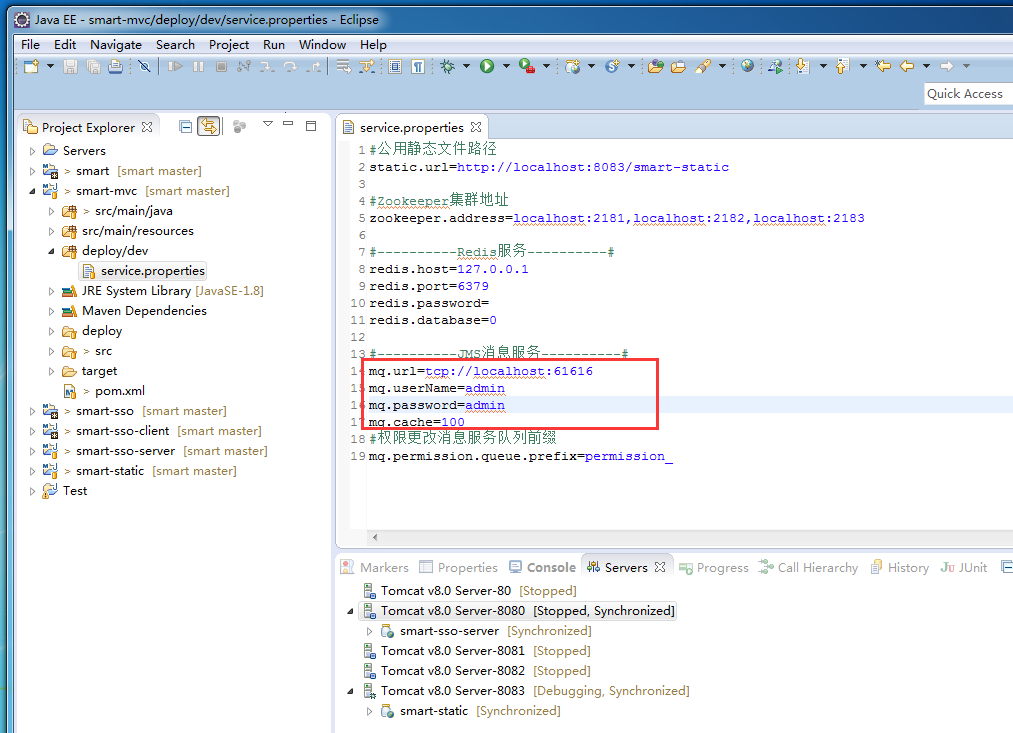
1. 官网下载地址：<http://activemq.apache.org/download-archives.html>，笔者使用的是apache-activemq-5.13.3版本。下载完成后解压，并找到bin路径，双击activemq.bat启动,如下图所示：



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）：

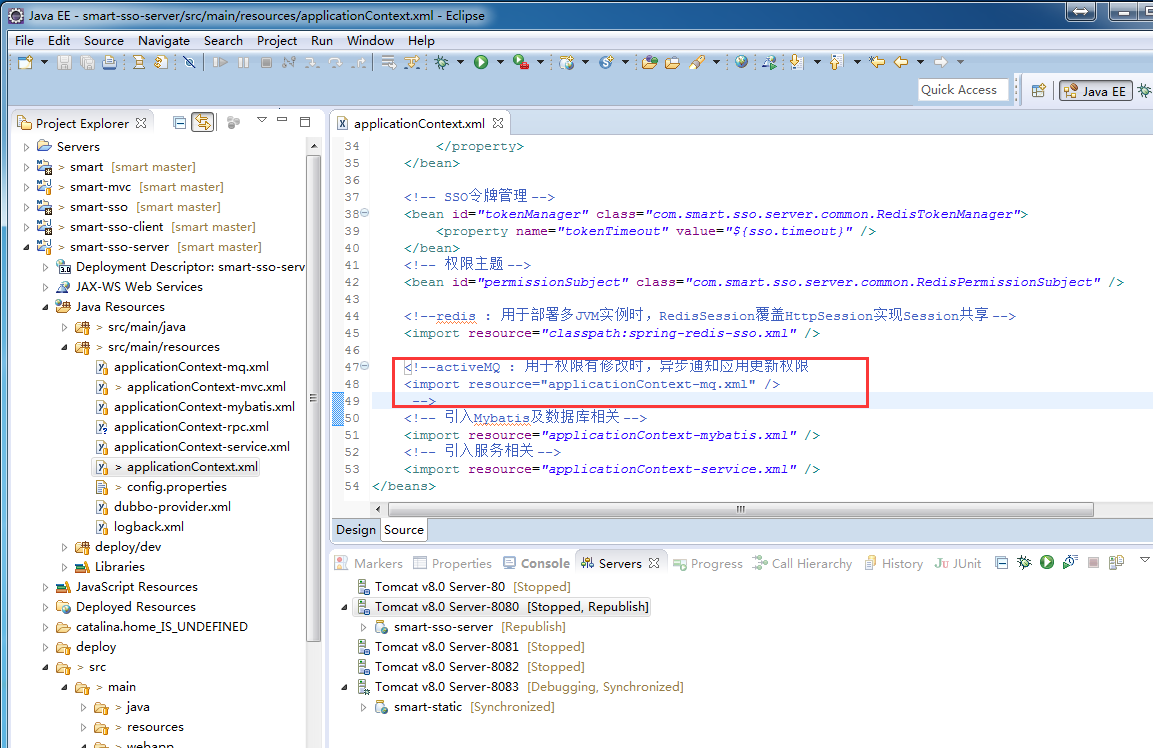


1. 打开smart-mvc中service.properties，修改activeMQ连接信息

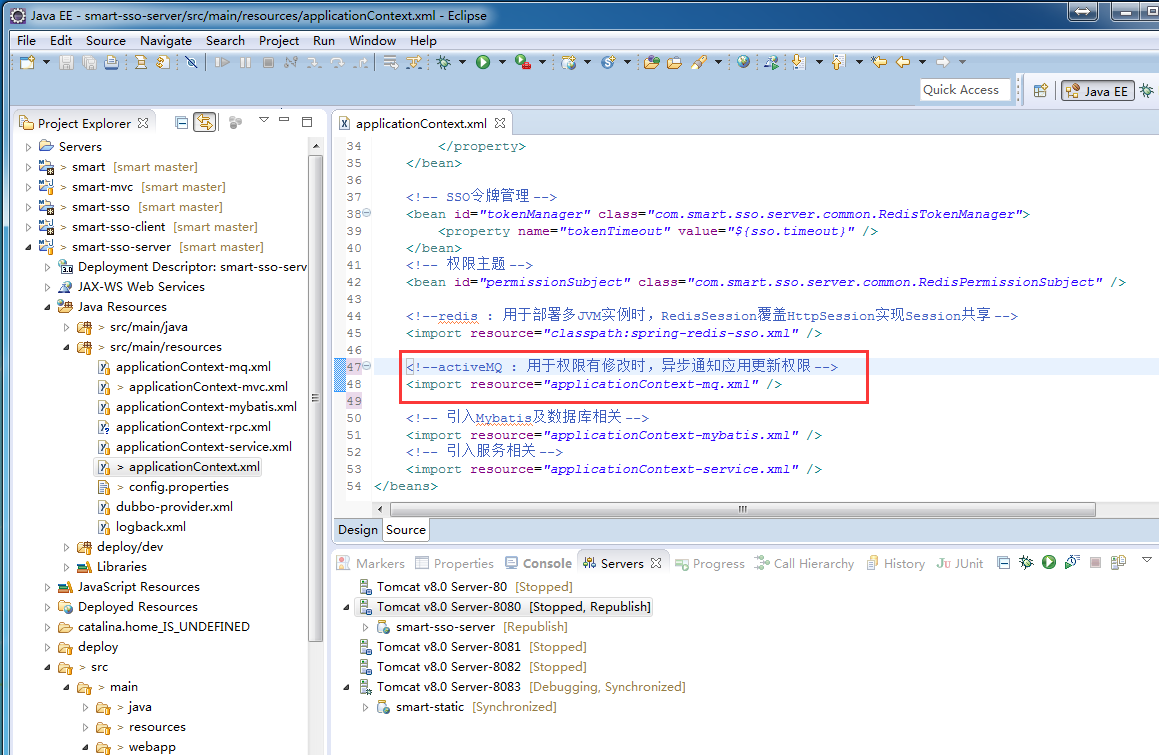


1. 打开smart-sso-server中applicationContext.xml配置文件, 去掉<import resource="applicationContext-mq.xml" />注释，如下图所示：

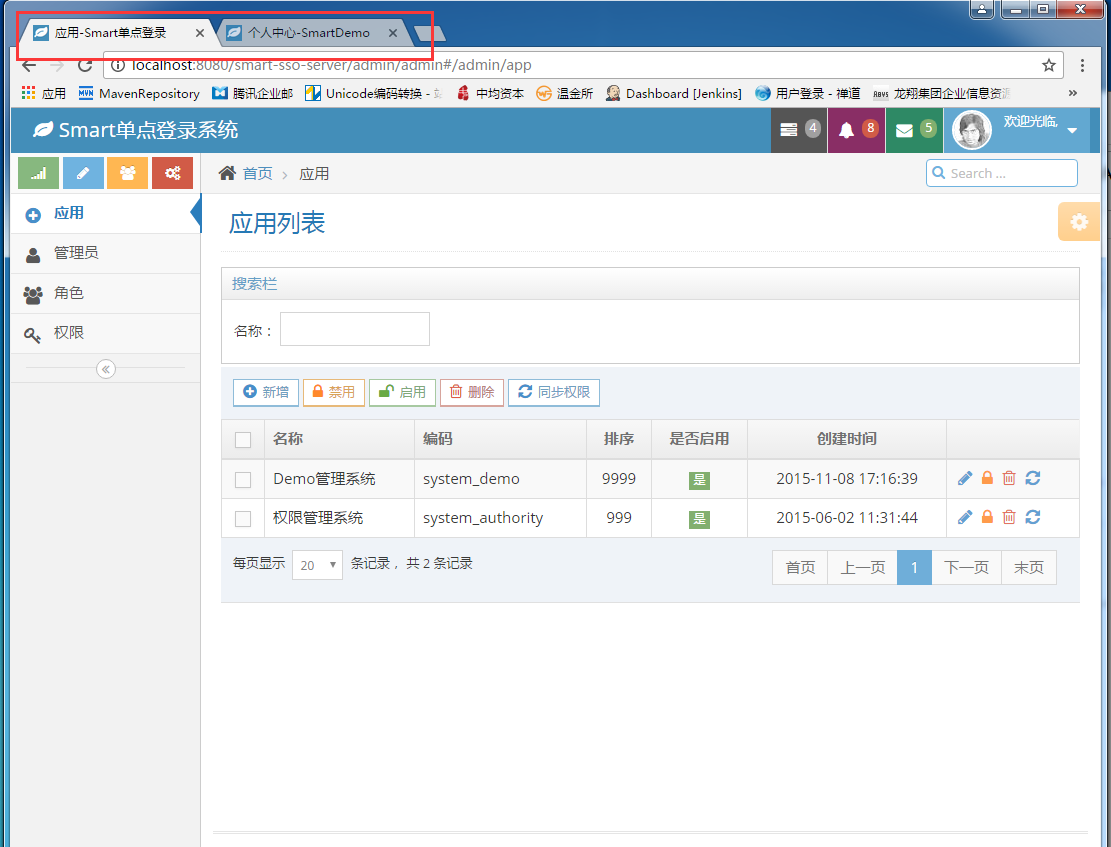
修改前：



修改后：

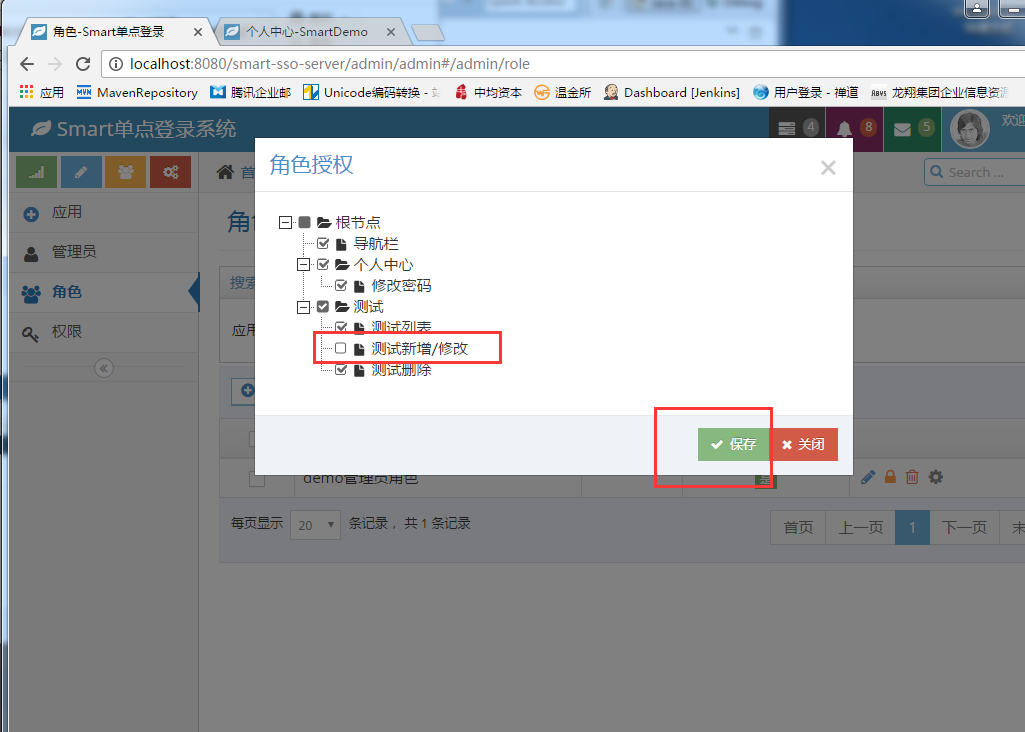


1. 重启项目，没有报错就表示配置成功。
2. 权限实时生效测试
3. 如果你已经部署smart-demo-web客户端。可进行如下测试，看权限的修改是否已实时。浏览器打开两个窗口，分别访问smart-sso-server、smart-demo-web。



1. 定位到smart-sso-server窗口，点击“角色”菜单。给当前登录的admin用户在demo系统中的角色，去掉菜单“测试”下的新增和修改按钮权限，并保存。





1. 此时切换到smart-demo-web窗口，点击“测试”菜单。可以看到列表中已经没有了“新增”按钮。这就表示ActiveMQ起作用了，权限也实时更新了。

