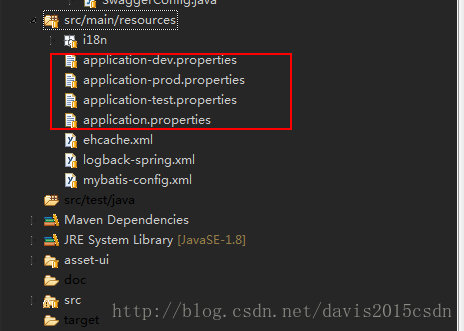
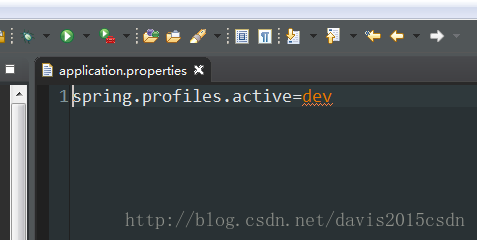
application-dev.properties：开发环境   
application-test.properties：测试环境   
application-prod.properties：生产环境   
至于哪个具体的配置文件会被加载，需要在application.properties文件中通过spring.profiles.active属性来设置，其值对应{profile}值。   
   
如：spring.profiles.active=dev就会加载application-dev.properties配置文件内容   
   
下面，以不同环境配置不同的服务端口为例，进行样例实验。   
针对各环境新建不同的配置文件application-dev.properties、application-test.properties、application-prod.properties   
在这三个文件均都设置不同的server.port属性，如：dev环境设置为8001，test环境设置为8002，prod环境设置为8003   
application.properties中设置spring.profiles.active=dev，就是说默认以dev环境设置

测试不同配置的加载

1. 执行java -jar xxx.jar，可以观察到服务端口被设置为8001，也就是默认的开发环境（dev）
2. 执行java -jar xxx.jar *--spring.profiles.active=test，可以观察到服务端口被设置为8002，也就是测试环境的配置（test）*
3. 执行java -jar xxx.jar *--spring.profiles.active=prod，可以观察到服务端口被设置为8003，也就是生产环境的配置（prod）*

按照上面的实验，可以如下总结多环境的配置思路：   
application.properties中配置通用内容，并设置spring.profiles.active=dev，以开发环境为默认配置   
application-{profile}.properties中配置各个环境不同的内容   
通过命令行方式去激活不同环境的配置