

# MD5 的两次加密以及提升性能技巧

## 目录

任务目标 .....	2
知识点 .....	2
实现思路 .....	3
编码步骤 .....	3
源码参考 .....	13
回马枪总结 .....	14

实训邦  
www.sxbang.net

## 任务目标

- 使用 MD5 分别在客户端和服务端加密并保存
- 提升 SpringBoot 性能的小技巧

## 知识点

MD5:

前端MD5加密:

下载地址: <https://github.com/placemaker/jquery-MD5>

MD5加盐处理:

盐 (Salt) 在密码学中, 是指通过在密码任意固定位置插入特定的字符串, 让散列后的结果和使用原始密码的散列结果不相符, 这种过程称之为“加盐”。

例如:

```
private static final String salt = "springboot";//特定盐

public static String inputPassToFormPass(String inputPass) {

    String str = inputPass + salt; //加盐

    //更多是下面这种加盐

    String str = salt.charAt(0)+inputPass + salt.charAt(2);

    return md5(str);

}
```

由于加了 Salt, 即便数据库泄露了, 但是由于密码都是加了 Salt 之后的散列, 坏人们的数据字典已经无法直接匹配, 明文密码被破解出来的概率也大大降低。

# 实现思路

为什么要进行加密？

当用户在注册的时候，注册进去存到数据库的密码是明码显示的，这样是不安全的，所以我们在注册的时候可以使用 MD5 的加密方式将用户的密码进行加密后存进数据库。

怎么做？

第一步在前端进行 MD5 加盐加密传输；第二步在后端在进行一次随机的加盐加密，最后将随机盐以及两次 MD5 加密的密码存储在数据库中。

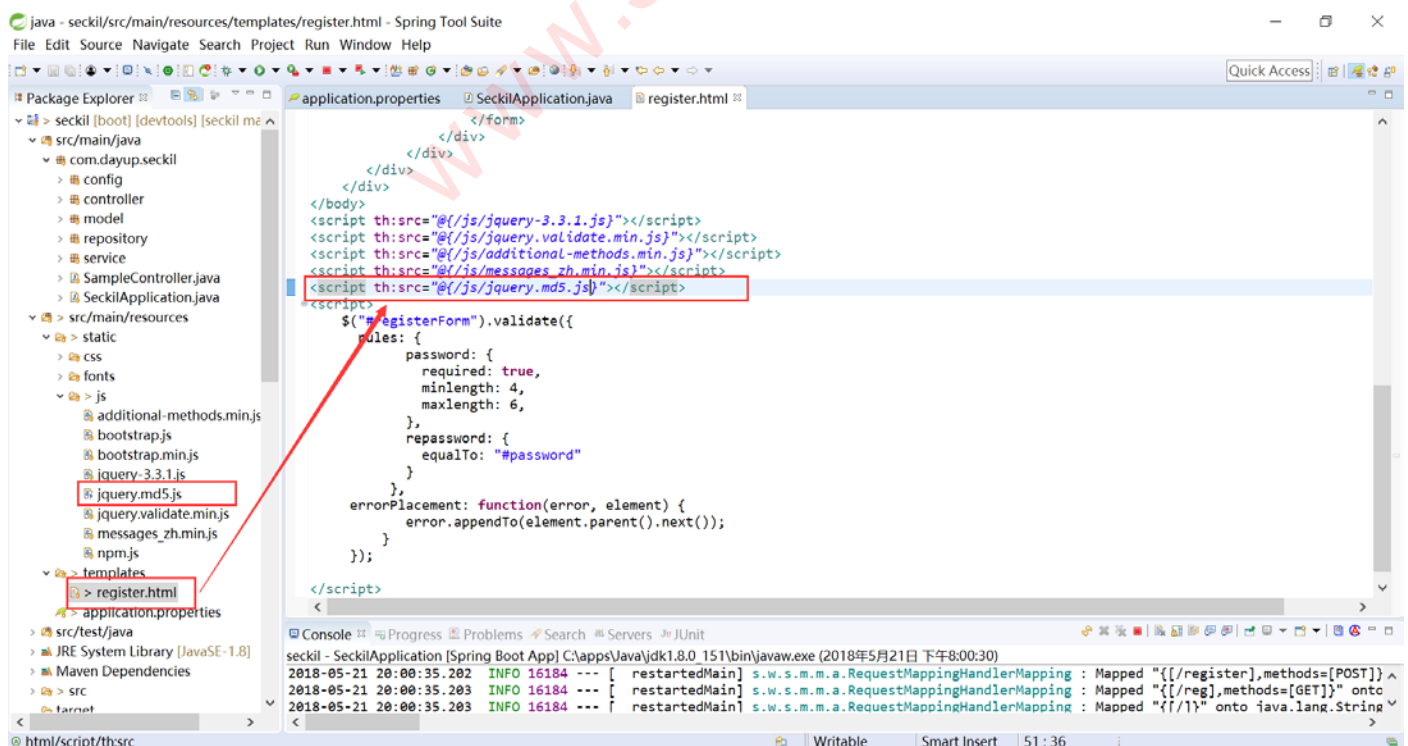
## 编码步骤

### 1. 使用 MD5 分别在客户端和服务端加密并保存

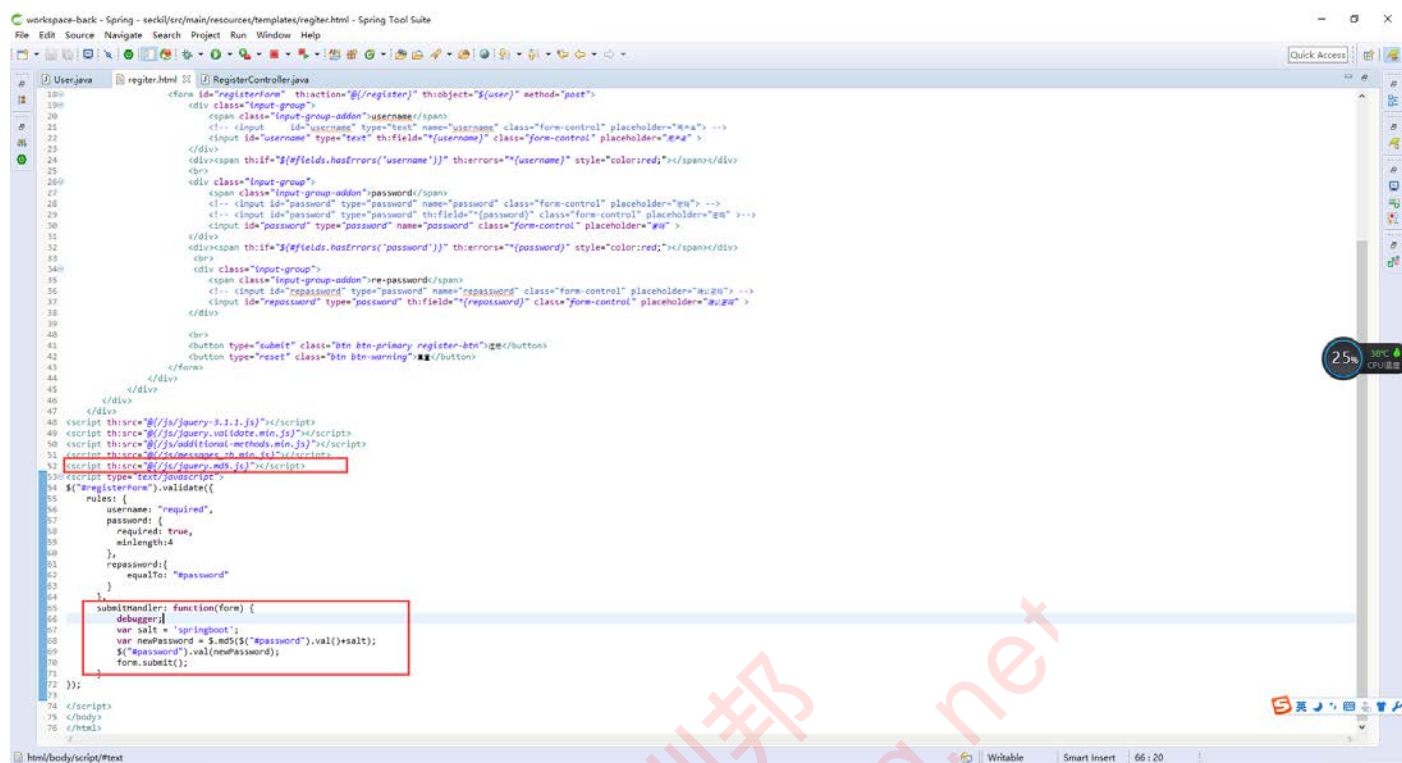
Step1:前端 MD5 下载&引入

下载地址: <https://github.com/placemaker/jquery-MD5>

下载之后解压，将下面的文件复制到我们的项目中。

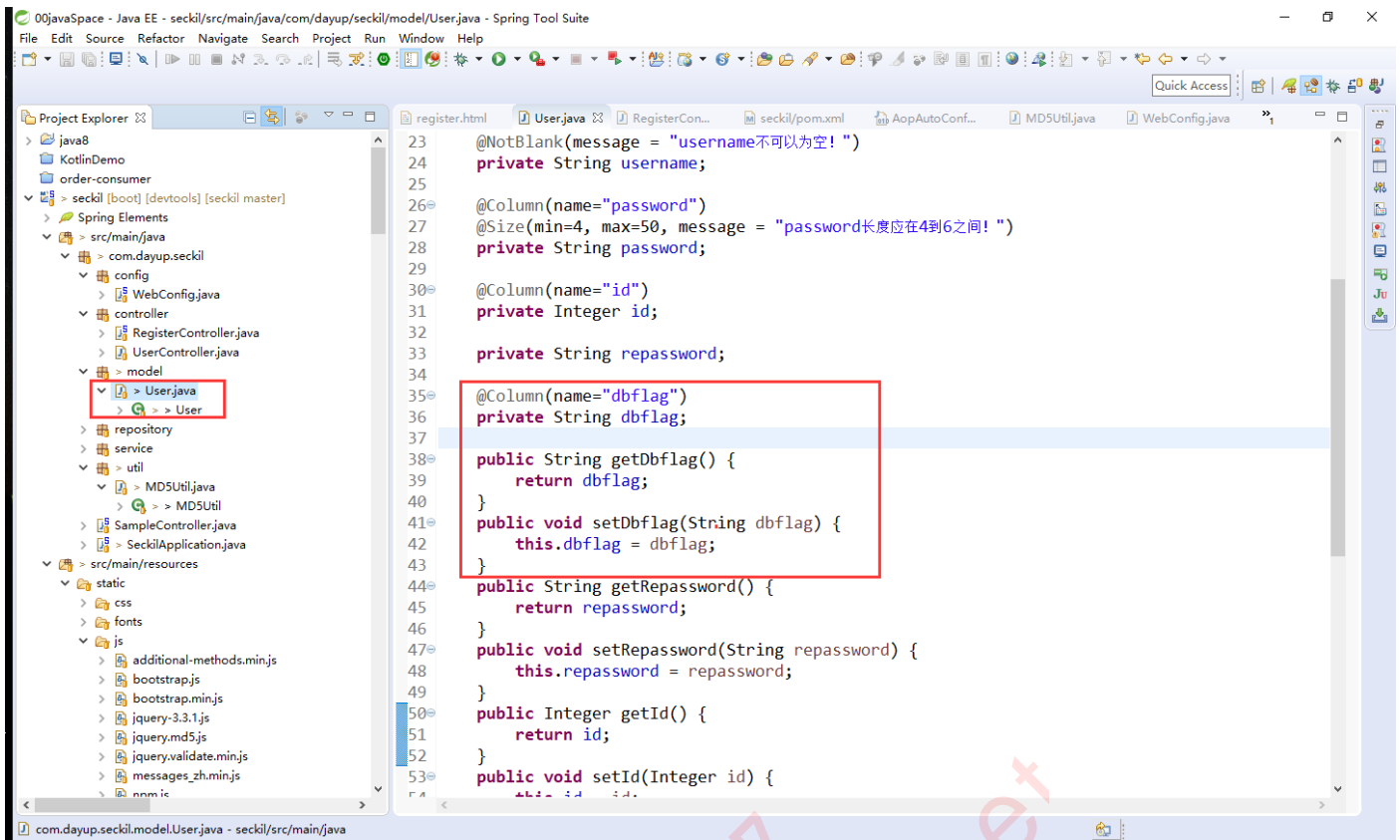


Step2: 使用 jquery.md5 进行加密, 并传输到 RegisterController.

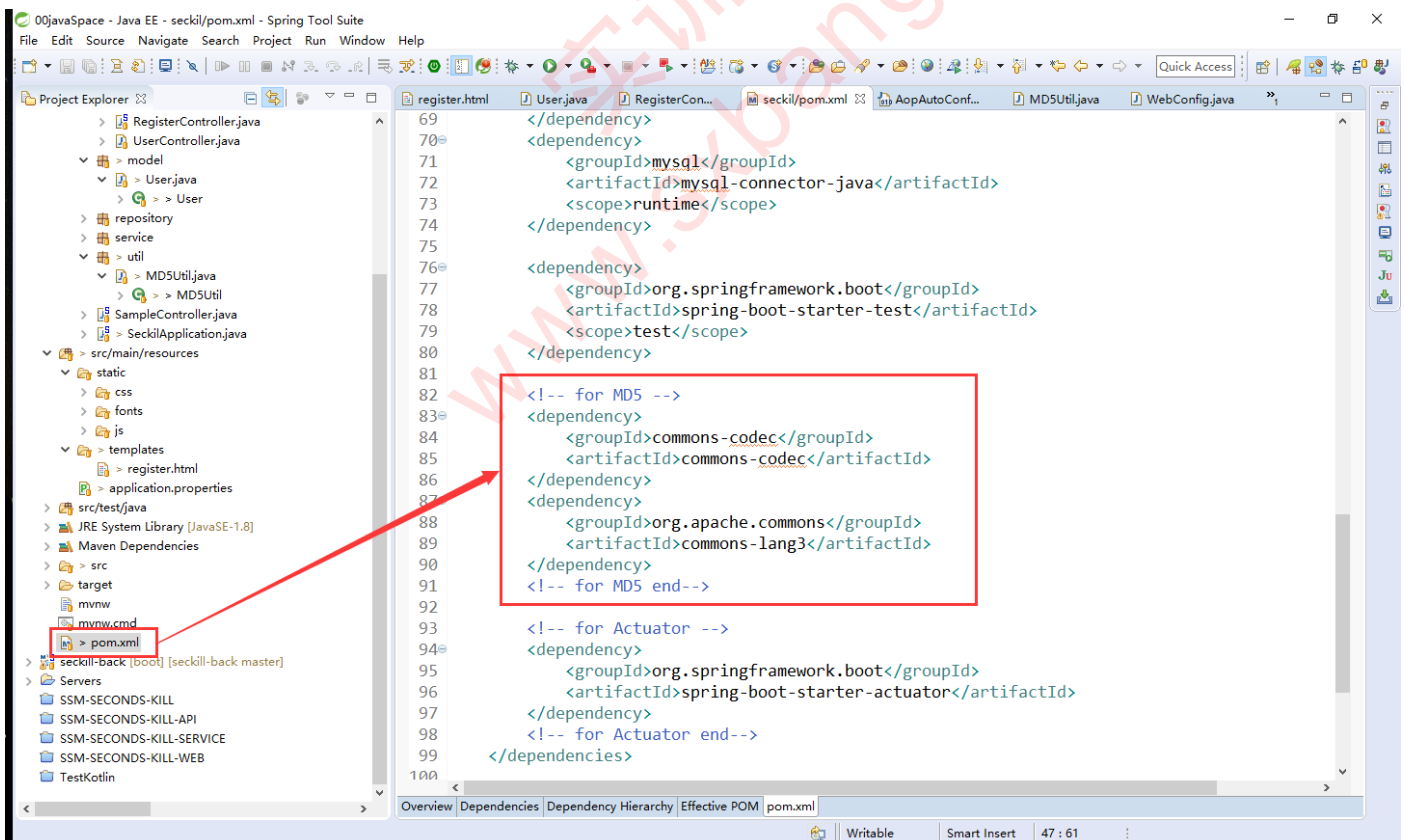


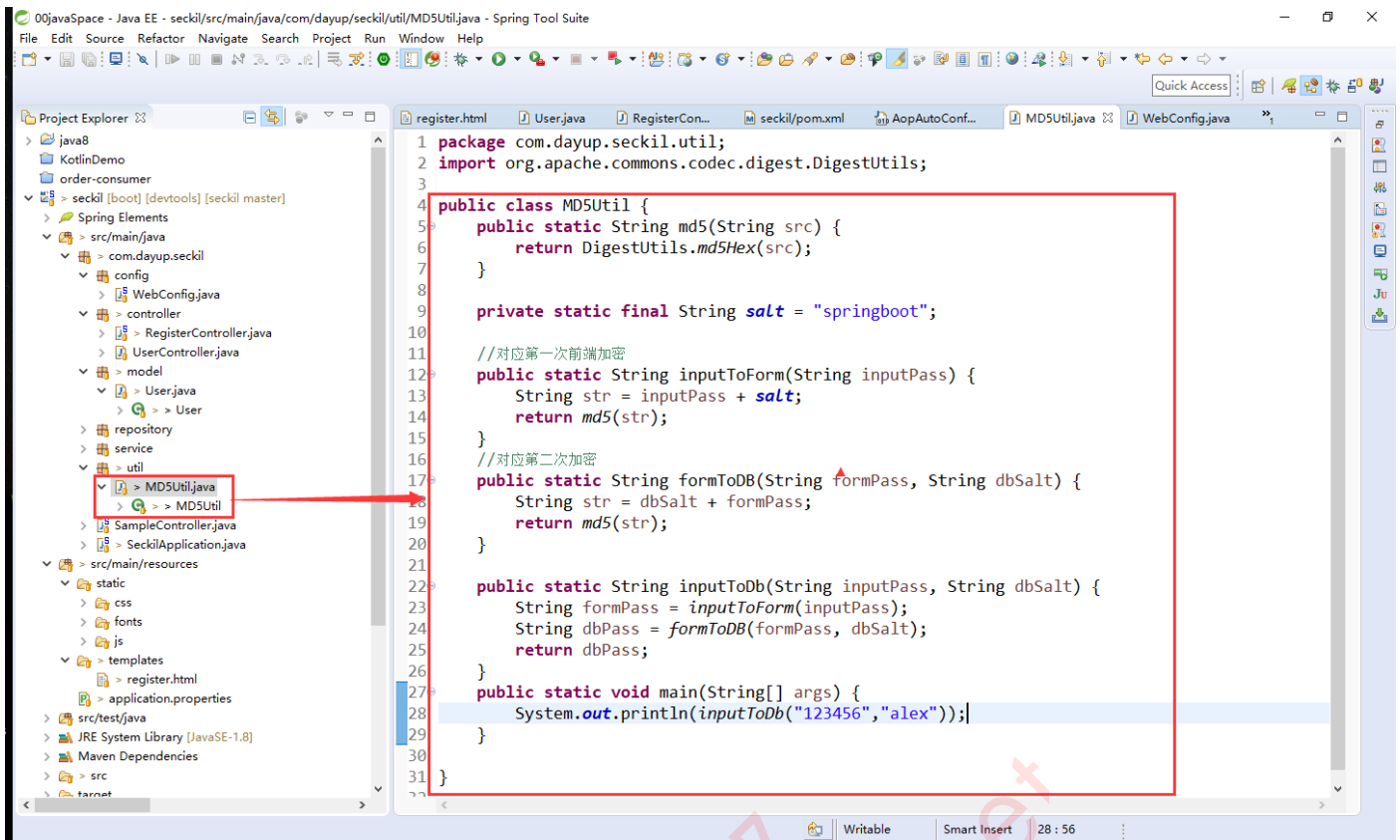
```
100 <form id="registerForm" th:action="@{/register}" th:object="${user}" method="post">
101 <div class="input-group">
102 <span class="input-group-addon">username</span>
103 <input id="username" type="text" name="username" class="form-control" placeholder="用户名">
104 </div>
105 <div class="input-group">
106 <span class="input-group-addon">password</span>
107 <input id="password" type="password" name="password" class="form-control" placeholder="密码">
108 </div>
109 <div class="input-group">
110 <span class="input-group-addon">re-password</span>
111 <input id="repassword" type="password" name="repassword" class="form-control" placeholder="再次输入">
112 </div>
113 <div>
114 <button type="submit" class="btn btn-primary register-btn">注册</button>
115 <button type="reset" class="btn btn-warning">重置</button>
116 </div>
117 </div>
118 </div>
119 <script th:src="@{/js/jquery-3.1.1.js}"></script>
120 <script th:src="@{/js/jquery.validate.min.js}"></script>
121 <script th:src="@{/js/validate.min.js}"></script>
122 <script th:src="@{/js/jquery.md5.js}"></script>
123 <script type="text/javascript">
124 $(function() {
125     $('#registerForm').validate({
126         rules: {
127             username: "required",
128             password: {
129                 required: true,
130                 minlength: 4
131             },
132             repassword: {
133                 equalTo: "password"
134             }
135         },
136         submitHandler: function(form) {
137             debugger;
138             var salt = 'springboot';
139             var newPassword = $.md5($('#password').val()+salt);
140             $('#repassword').val(newPassword);
141             form.submit();
142         }
143     });
144 }
145 </script>
146 </body>
147 </html>
```

Step3.修改数据实体 User.java, 增加“盐”属性

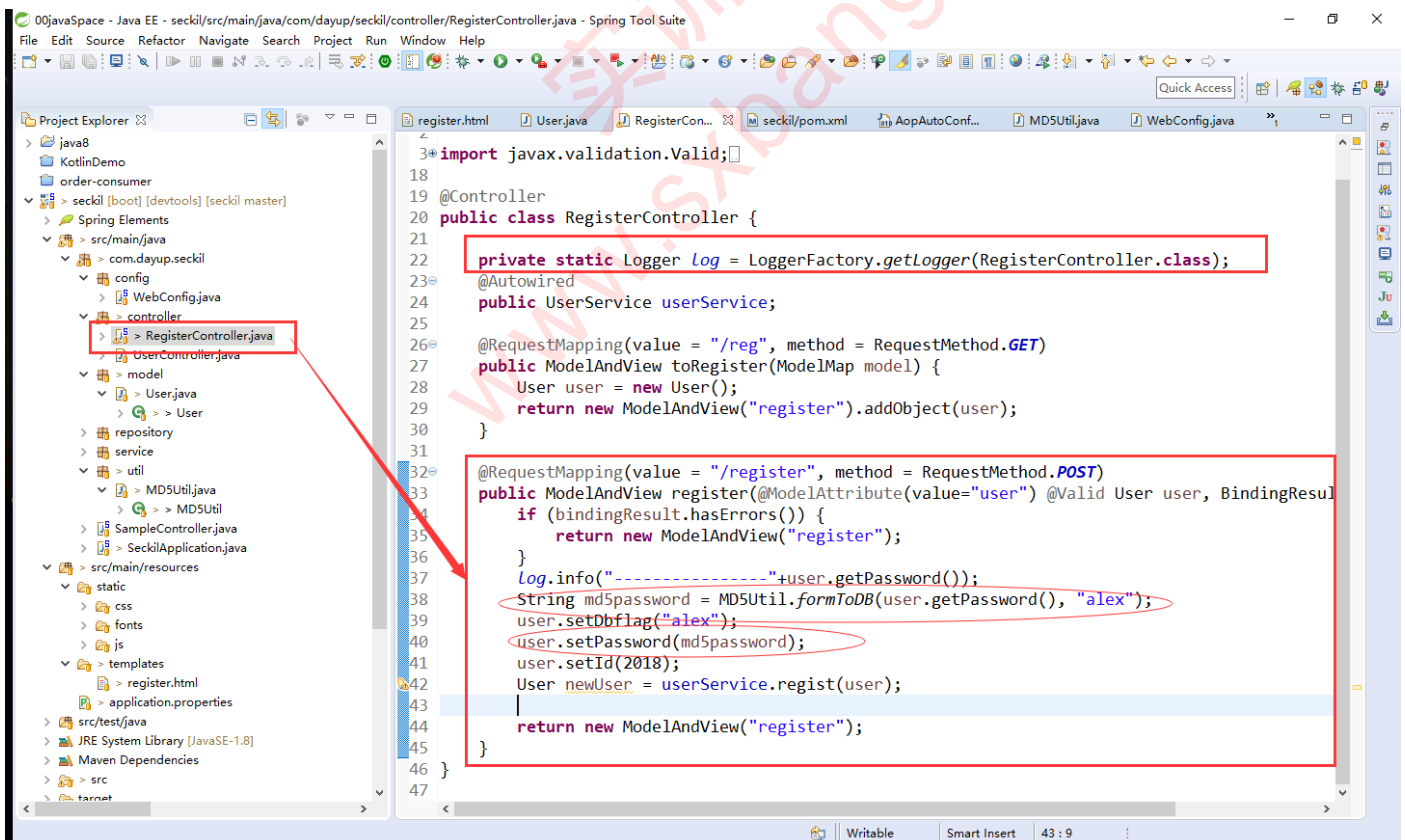


Step4.增加 MD5Util 工具类，引入下面内容：





Step5.控制器 RegisterController 进行第二次加密处理:

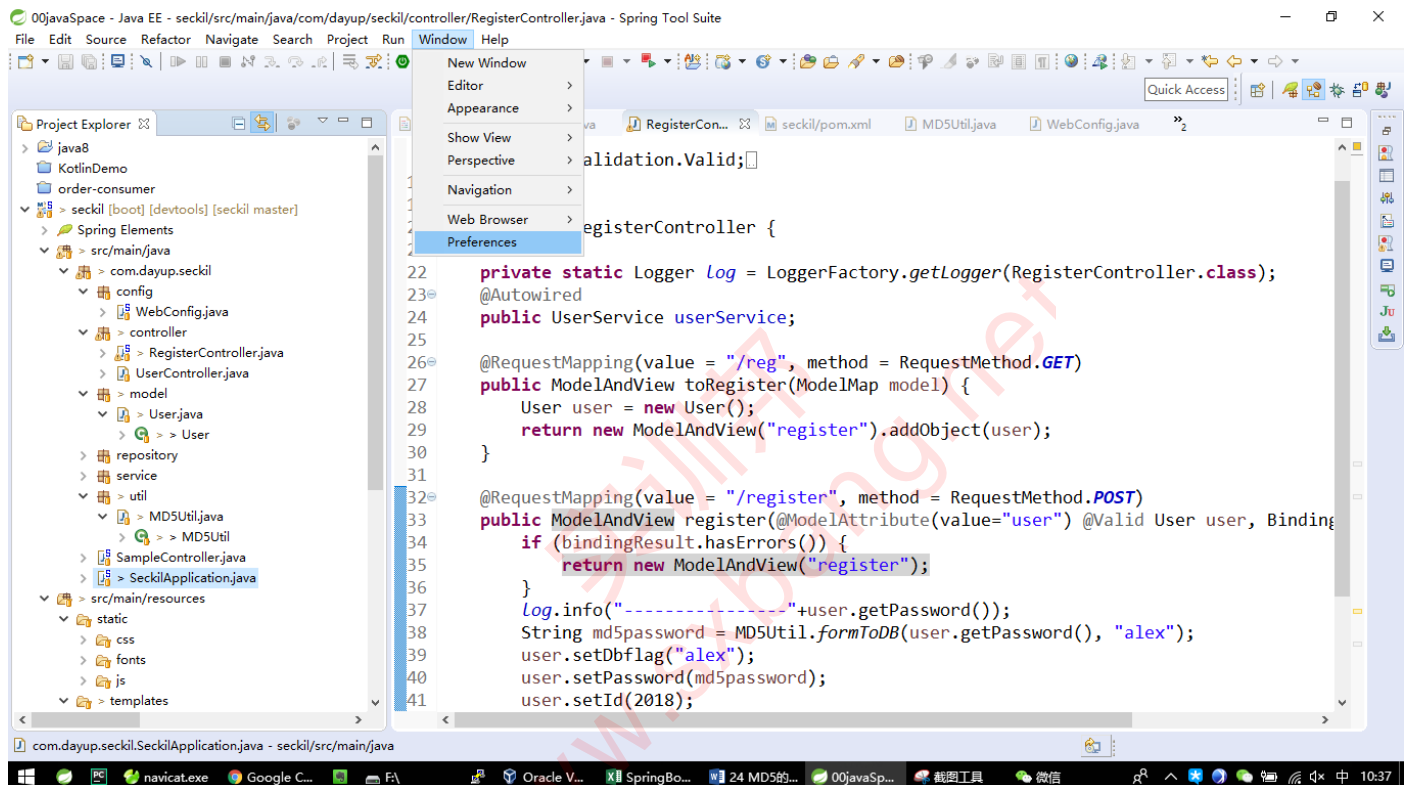


## 2. 提升 SpringBoot 性能的小技巧:

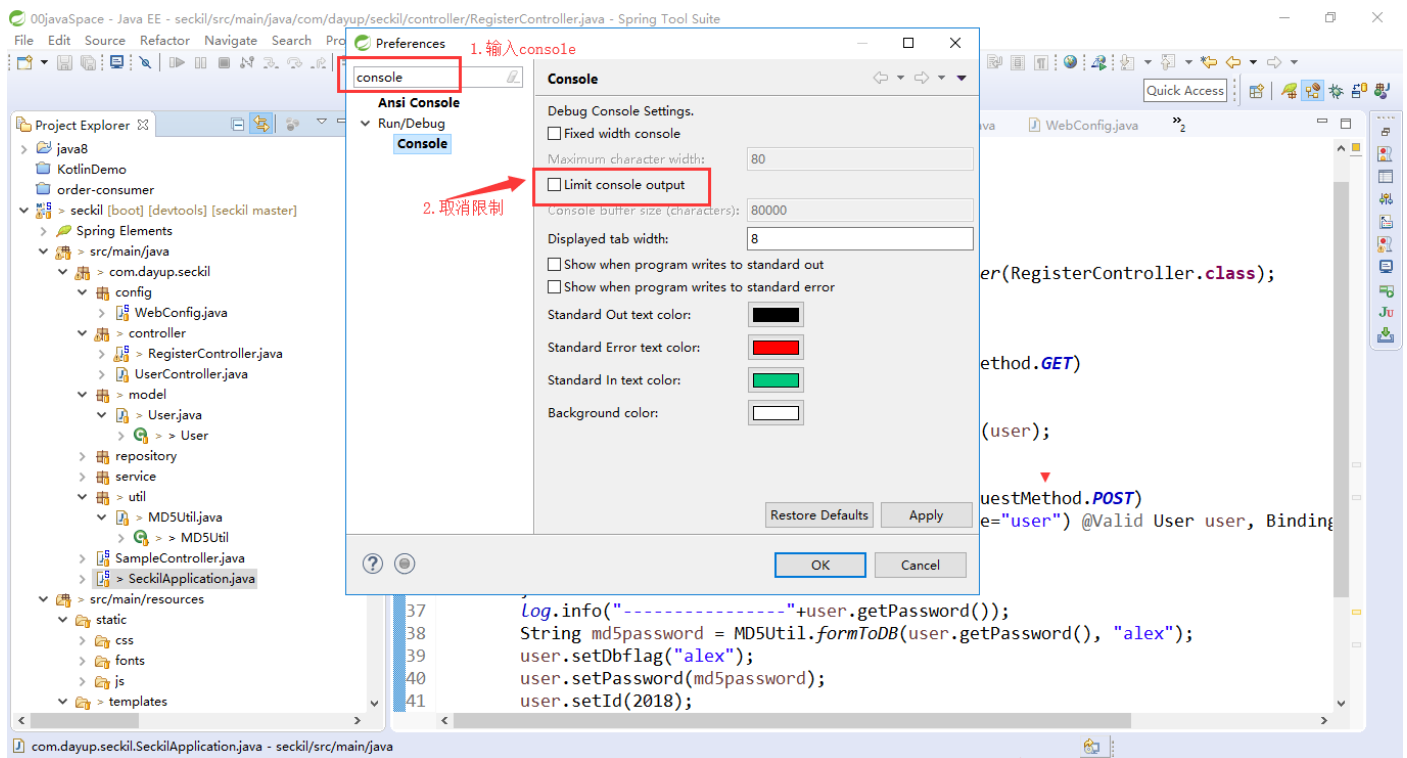
### Step1: 更改加载配置的方式

SpringBoot 自动配置给我们提供很大便利, 难免会增加应用的启动时间、内存和 CPU 的消耗等, 如果对这方面要求很高, 就可以根据自动配置的原理进行一些优化。首先弄清楚一个应用需要加载哪些配置? 我们可以通过设置 Debug 启动, 见以下步骤:

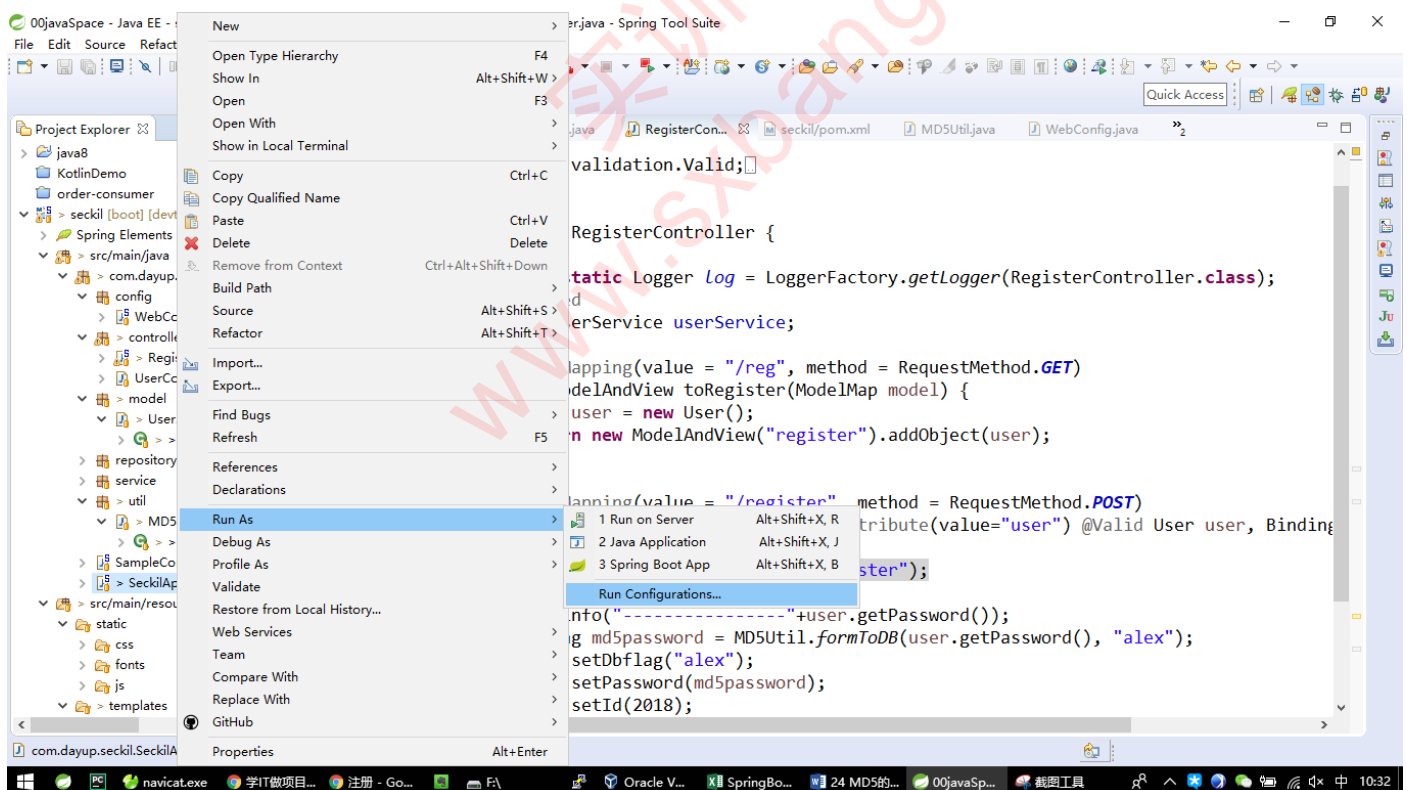
首先, 为了保证 console 控制台可以看到全部日志文件, 设置我们的 STS:



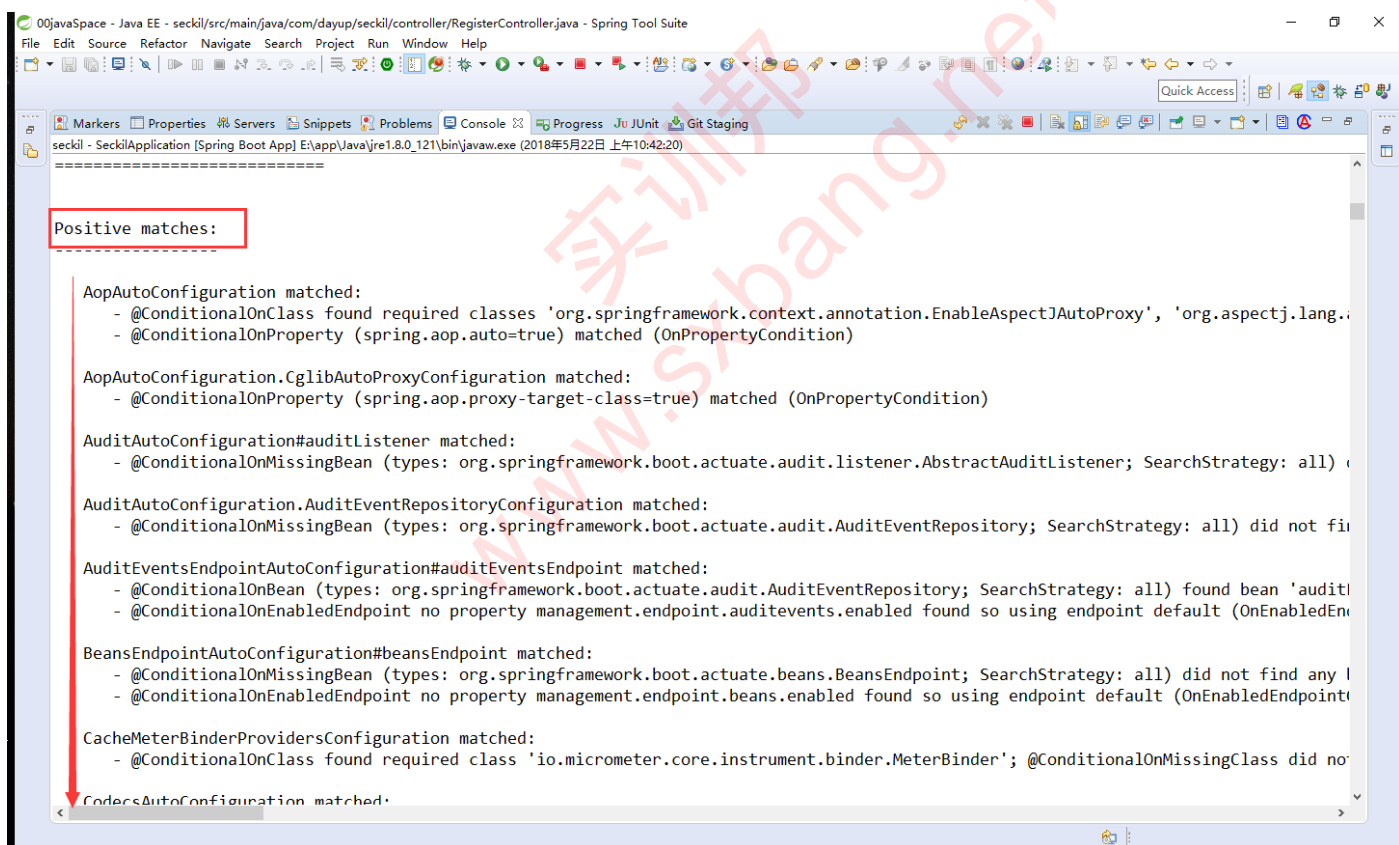
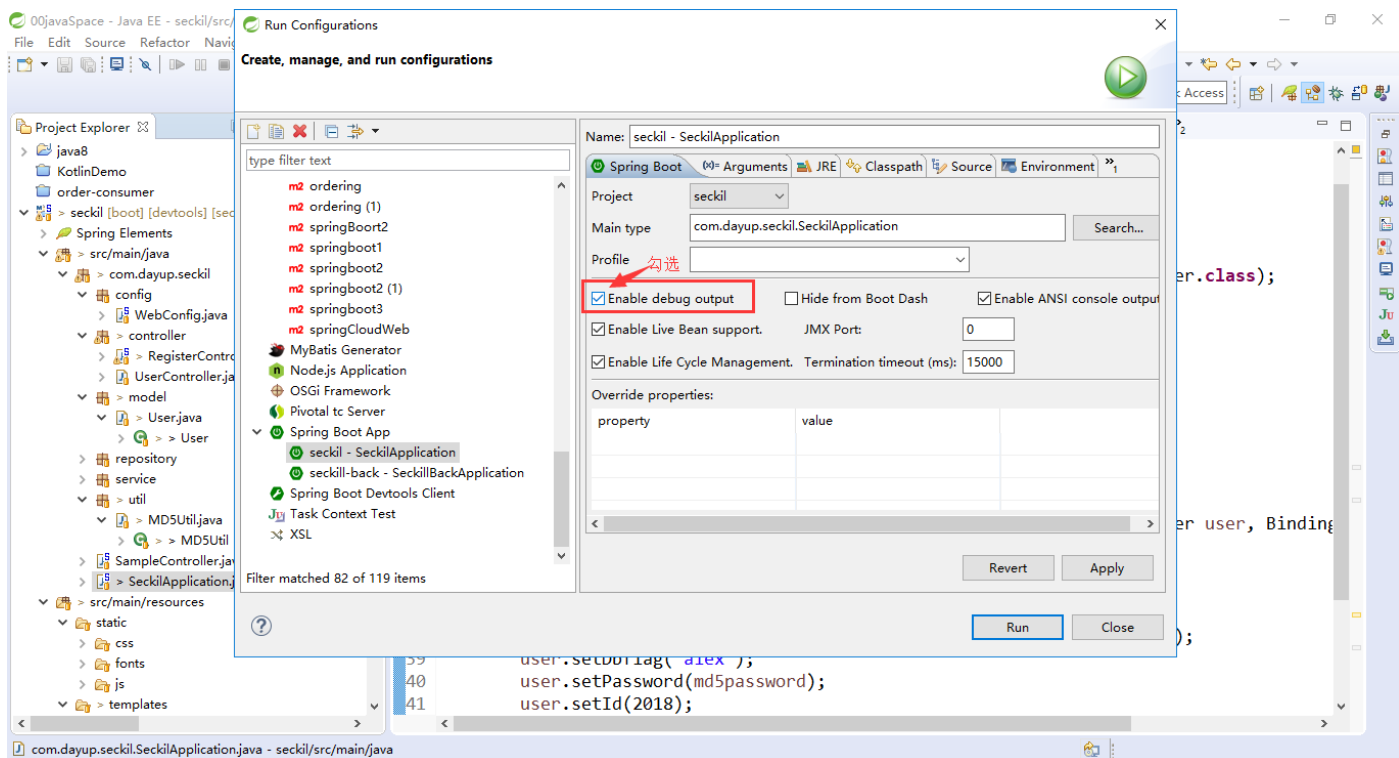




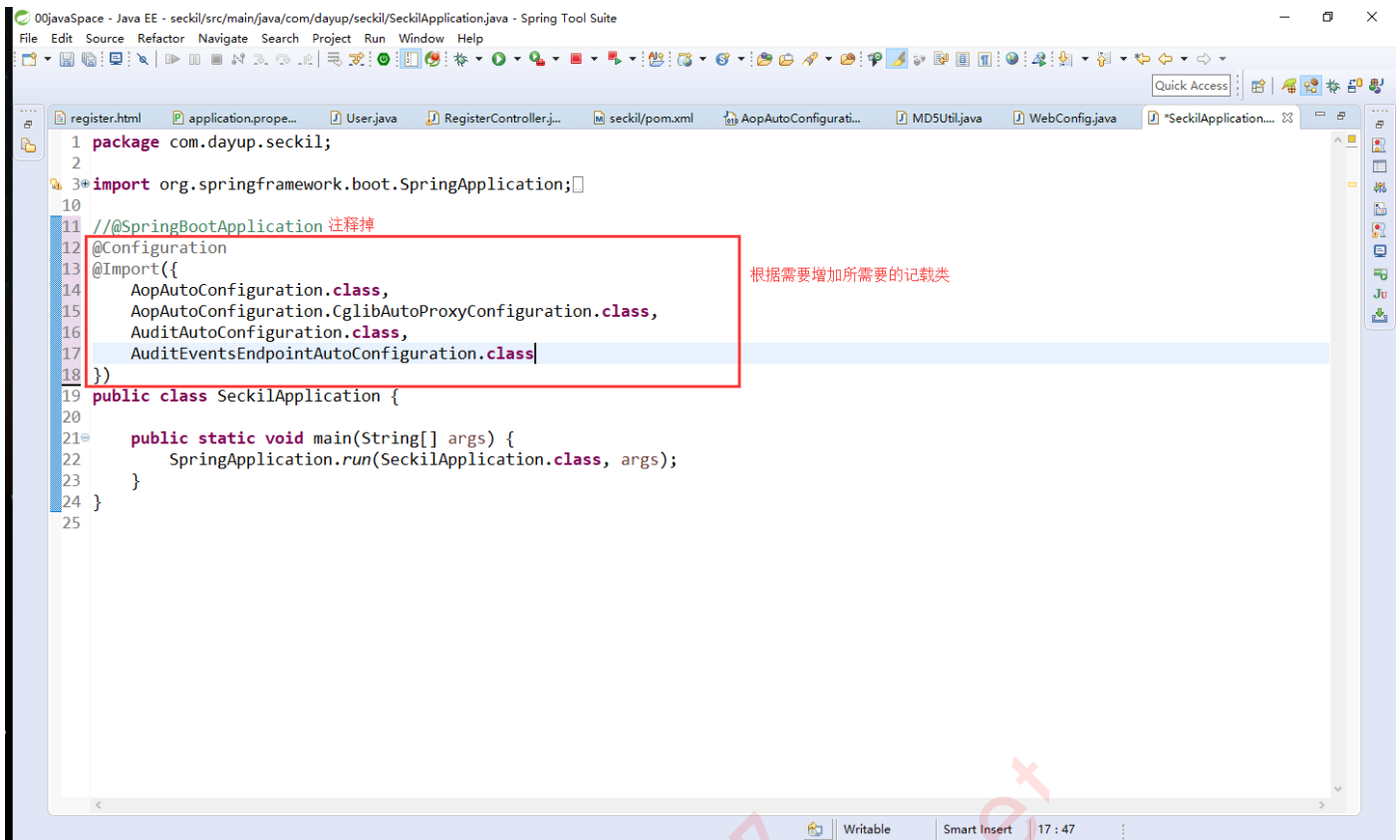
## 设置 Debug 模式







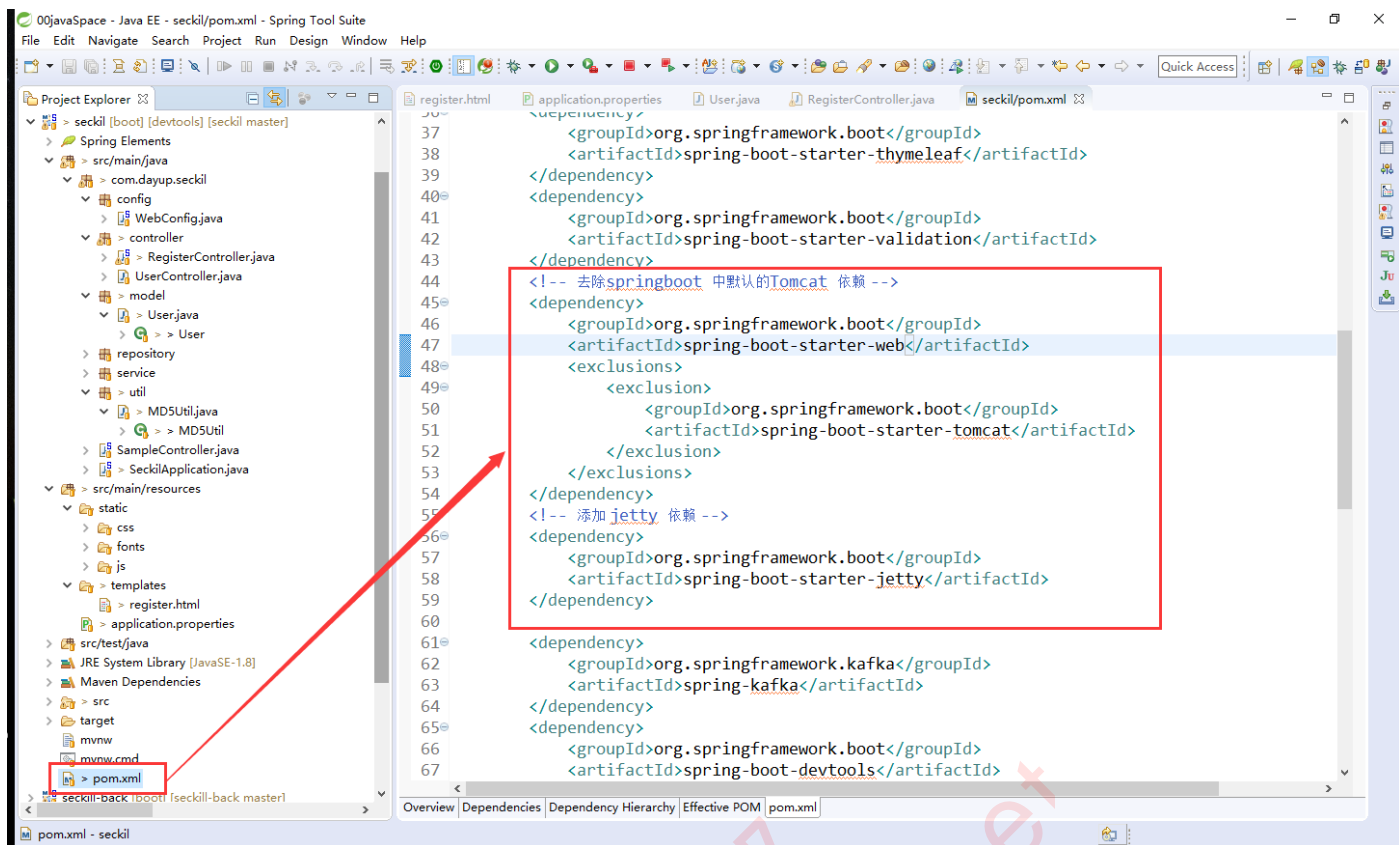
在控制台中找到 Positive matches，这些就是本应用需要加载的一些配置类，我们需要整理，然后修改我们的启动文件，见下图：



Step2:将 Tomcat 换成 Jetty

为了提高应用性能，也可以将默认的 tomcat 插件，更换成更小巧的 Jetty。

只需要修改 pom.xml 即可，做法：1.去除 Spring Boot 中默认的 Tomcat 依赖。2.添加 Jetty 依赖。



<!-- 去除 springboot 中默认的 Tomcat 依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<exclusions>

<exclusion>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

</exclusion>

</exclusions>

</dependency>

<!-- 添加 jetty 依赖 -->

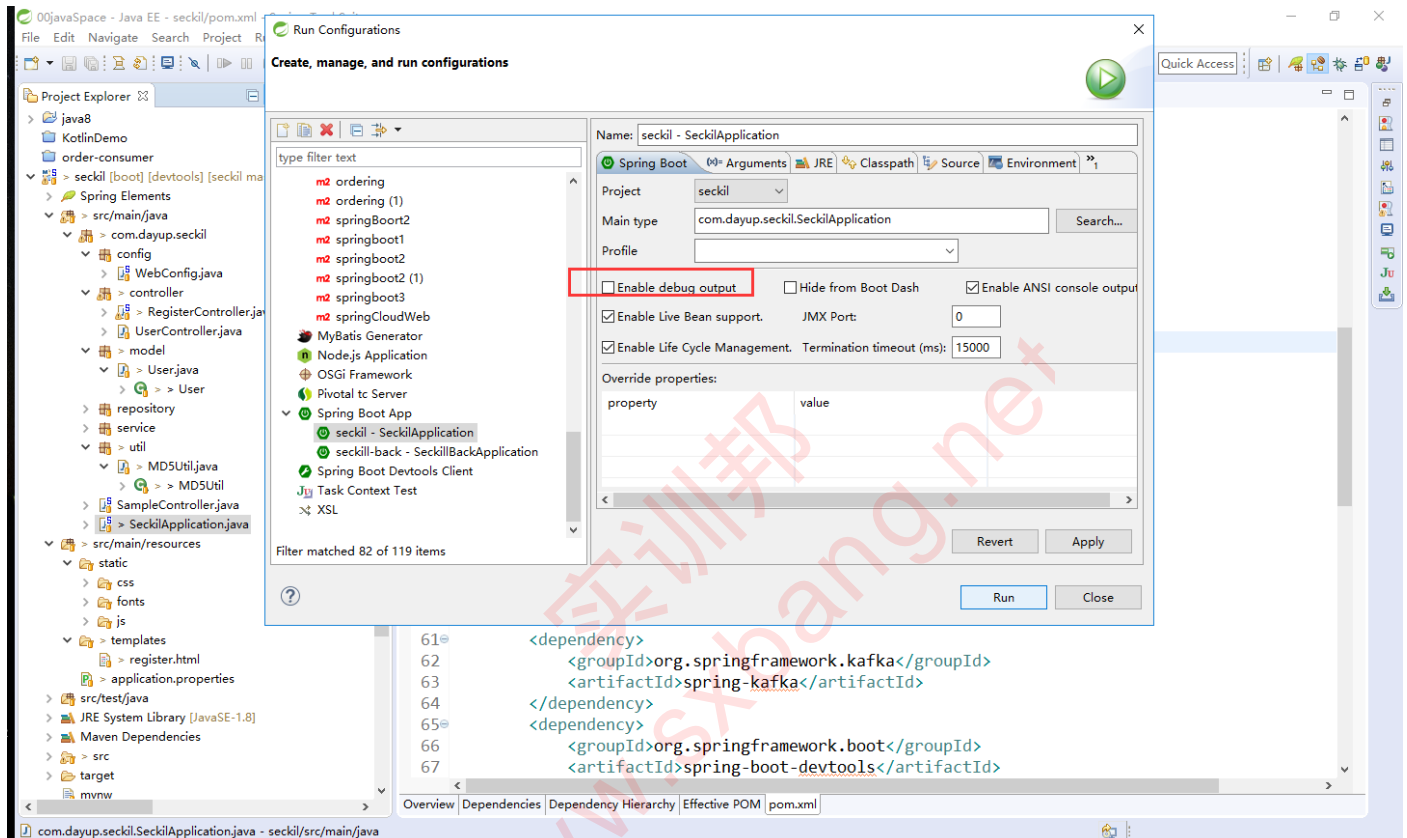
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jetty</artifactId>

</dependency>

取消掉上一步骤说的 debug 模式，启动



```
00javaSpace - Java EE - seckil/pom.xml - Spring Tool Suite
File Edit Navigate Search Project Run Design Window Help
Markers Properties Servers Snippets Problems Console Progress JUnit Git Staging
seckil - SeckilApplication [Spring Boot App] E:\app\Java\jre1.8.0_121\bin\javaw.exe (2018年5月22日 上午10:57:22)

: HHH000412: Hibernate Core {5.2.17.Final}
: HHH000206: hibernate.properties not found
on : HCANN000001: Hibernate Commons Annotations {5.0.1.Final}
: HikariPool-1 - Starting...
: HikariPool-1 - Start completed.
: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
Bean : Initialized JPA EntityManagerFactory for persistence unit 'default'
ping : Mapped "{[/register],methods=[POST]}" onto public org.springframework.web.servlet.ModelAndView com.dayup.seckil.controller.Regis
ping : Mapped "{[/reg],methods=[GET]}" onto public org.springframework.web.servlet.ModelAndView com.dayup.seckil.controller.RegisterCor
ping : Mapped "{[/]}" onto java.lang.String com.dayup.seckil.SampleController.home()
ping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.s
ping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoc
ing : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
pter : Looking for @ControllerAdvice: org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@:
ode : [THYMELEAF][restartedMain] Template Mode 'HTML5' is deprecated. Using Template Mode 'HTML' instead.
: LiveReload server is running on port 35729
: Exposing 2 endpoint(s) beneath base path '/actuator'
ping : Mapped "{[/actuator/health],methods=[GET],produces=[application/vnd.spring-boot.actuator.v2+json || application/json]}" onto put
ping : Mapped "{[/actuator/info],methods=[GET],produces=[application/vnd.spring-boot.actuator.v2+json || application/json]}" onto publ
ping : Mapped "{[/actuator],methods=[GET],produces=[application/vnd.spring-boot.actuator.v2+json || application/json]}" onto protected
: Registering beans for JMX exposure on startup
: Bean with name 'dataSource' has been autodetected for JMX exposure
: Located MBean 'dataSource': registering with JMX server as MBean [com.zaxxer.hikari:name=dataSource,type=HikariDataSource]
sor : Starting beans in phase 2147483547
: Initializing Spring FrameworkServlet 'dispatcherServlet'
: FrameworkServlet 'dispatcherServlet': initialization started
: FrameworkServlet 'dispatcherServlet': initialization completed in 87 ms
: Started ServerConnector@18720647/HTTP/1.1,[http/1.1]{0.0.0.0:8080}
ver : Jetty started on port(s) 8080 (http/1.1) with context path '/'
: Started SeckilApplication in 22.925 seconds (JVM running for 24.342)
```



## 源码参考

详见 Git 地址获取

## 回马枪总结

- 使用 MD5 分别在客户端和服务端加密并保存

第一步在前端进行 MD5 加盐加密传输；

第二步在后端在进行一次随机的加盐加密，最后将随机盐以及两次 MD5 加密的密码存储在数据库中。

- 提升 SpringBoot 性能的小技巧

第一加载文件识别后配置；

第二使用更小巧的 Jetty。

实训邦  
www.sxbang.net