[**第二十二章 集成验证码——《跟我学Shiro》**](http://jinnianshilongnian.iteye.com/blog/2046041)

**博客分类：**

* [跟我学Shiro](http://jinnianshilongnian.iteye.com/category/305053)

[跟我学Shiro](http://www.iteye.com/blogs/tag/%E8%B7%9F%E6%88%91%E5%AD%A6Shiro)

**目录贴：**[**跟我学Shiro目录贴**](http://jinnianshilongnian.iteye.com/blog/2018398)

在做用户登录功能时，很多时候都需要验证码支持，验证码的目的是为了防止机器人模拟真实用户登录而恶意访问，如暴力破解用户密码/恶意评论等。目前也有一些验证码比较简单，通过一些OCR工具就可以解析出来；另外还有一些验证码比较复杂（一般通过如扭曲、加线条/噪点等干扰）防止OCR工具识别；但是在中国就是人多，机器干不了的可以交给人来完成，所以在中国就有很多打码平台，人工识别验证码；因此即使比较复杂的如填字、算数等类型的验证码还是能识别的。所以验证码也不是绝对可靠的，目前比较可靠还是手机验证码，但是对于用户来说相对于验证码还是比较麻烦的。

对于验证码图片的生成，可以自己通过如Java提供的图像API自己去生成，也可以借助如JCaptcha这种开源Java类库生成验证码图片；JCaptcha提供了常见的如扭曲、加噪点等干扰支持。本章代码基于《第十六章 综合实例》。

**一、添加JCaptcha依赖**

**Java代码  [收藏代码](javascript:void())**

1. <dependency>
2. <groupId>com.octo.captcha</groupId>
3. <artifactId>jcaptcha</artifactId>
4. <version>2.0-alpha-1</version>
5. </dependency>
6. <dependency>
7. <groupId>com.octo.captcha</groupId>
8. <artifactId>jcaptcha-integration-simple-servlet</artifactId>
9. <version>2.0-alpha-1</version>
10. <exclusions>
11. <exclusion>
12. <artifactId>servlet-api</artifactId>
13. <groupId>javax.servlet</groupId>
14. </exclusion>
15. </exclusions>
16. </dependency>

com.octo.captcha . jcaptcha 提供了jcaptcha 核心；而jcaptcha-integration-simple-servlet提供了与Servlet集成。

**二、GMailEngine**

来自<https://code.google.com/p/musicvalley/source/browse/trunk/musicvalley/doc/springSecurity/springSecurityIII/src/main/java/com/spring/security/jcaptcha/GMailEngine.java?spec=svn447&r=447>（目前无法访问了），仿照JCaptcha2.0编写类似GMail验证码的样式；具体请参考com.github.zhangkaitao.shiro.chapter22.jcaptcha.GMailEngine。

**三、MyManageableImageCaptchaService**

提供了判断仓库中是否有相应的验证码存在。

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** MyManageableImageCaptchaService **extends**
2. DefaultManageableImageCaptchaService {
3. **public** MyManageableImageCaptchaService(
4. com.octo.captcha.service.captchastore.CaptchaStore captchaStore,
5. com.octo.captcha.engine.CaptchaEngine captchaEngine,
6. **int** minGuarantedStorageDelayInSeconds,
7. **int** maxCaptchaStoreSize,
8. **int** captchaStoreLoadBeforeGarbageCollection) {
9. **super**(captchaStore, captchaEngine, minGuarantedStorageDelayInSeconds,
10. maxCaptchaStoreSize, captchaStoreLoadBeforeGarbageCollection);
11. }
12. **public** **boolean** hasCapcha(String id, String userCaptchaResponse) {
13. **return** store.getCaptcha(id).validateResponse(userCaptchaResponse);
14. }
15. }

**四、JCaptcha工具类**

提供相应的API来验证当前请求输入的验证码是否正确。

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** JCaptcha {
2. **public** **static** **final** MyManageableImageCaptchaService captchaService
3. = **new** MyManageableImageCaptchaService(**new** FastHashMapCaptchaStore(),
4. **new** GMailEngine(), 180, 100000, 75000);
5. **public** **static** **boolean** validateResponse(
6. HttpServletRequest request, String userCaptchaResponse) {
7. **if** (request.getSession(**false**) == **null**) **return** **false**;
8. **boolean** validated = **false**;
9. **try** {
10. String id = request.getSession().getId();
11. validated =
12. captchaService.validateResponseForID(id, userCaptchaResponse)
13. .booleanValue();
14. } **catch** (CaptchaServiceException e) {
15. e.printStackTrace();
16. }
17. **return** validated;
18. }
19. **public** **static** **boolean** hasCaptcha(
20. HttpServletRequest request, String userCaptchaResponse) {
21. **if** (request.getSession(**false**) == **null**) **return** **false**;
22. **boolean** validated = **false**;
23. **try** {
24. String id = request.getSession().getId();
25. validated = captchaService.hasCapcha(id, userCaptchaResponse);
26. } **catch** (CaptchaServiceException e) {
27. e.printStackTrace();
28. }
29. **return** validated;
30. }
31. }

validateResponse()：验证当前请求输入的验证码否正确；并从CaptchaService中删除已经生成的验证码；

hasCaptcha()：验证当前请求输入的验证码是否正确；但不从CaptchaService中删除已经生成的验证码（比如Ajax验证时可以使用，防止多次生成验证码）；

**五、JCaptchaFilter**

用于生成验证码图片的过滤器。

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** JCaptchaFilter **extends** OncePerRequestFilter {
2. **protected** **void** doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) **throws** ServletException, IOException {
4. response.setDateHeader("Expires", 0L);
5. response.setHeader("Cache-Control", "no-store, no-cache, must-revalidate");
6. response.addHeader("Cache-Control", "post-check=0, pre-check=0");
7. response.setHeader("Pragma", "no-cache");
8. response.setContentType("image/jpeg");
9. String id = request.getRequestedSessionId();
10. BufferedImage bi = JCaptcha.captchaService.getImageChallengeForID(id);
11. ServletOutputStream out = response.getOutputStream();
12. ImageIO.write(bi, "jpg", out);
13. **try** {
14. out.flush();
15. } **finally** {
16. out.close();
17. }
18. }
19. }

CaptchaService使用当前会话ID当作key获取相应的验证码图片；另外需要设置响应内容不进行浏览器端缓存。

**Java代码  [收藏代码](javascript:void())**

1. <!-- 验证码过滤器需要放到Shiro之后 因为Shiro将包装HttpSession 如果不，可能造成两次的sesison id 不一样 -->
2. <filter>
3. <filter-name>JCaptchaFilter</filter-name>
4. <filter-**class**>
5. com.github.zhangkaitao.shiro.chapter22.jcaptcha.JCaptchaFilter
6. </filter-**class**>
7. </filter>
8. <filter-mapping>
9. <filter-name>JCaptchaFilter</filter-name>
10. <url-pattern>/jcaptcha.jpg</url-pattern>
11. </filter-mapping>

这样就可以在页面使用/jcaptcha.jpg地址显示验证码图片。

**六、JCaptchaValidateFilter**

用于验证码验证的Shiro过滤器。

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** JCaptchaValidateFilter **extends** AccessControlFilter {
2. **private** **boolean** jcaptchaEbabled = **true**;//是否开启验证码支持
3. **private** String jcaptchaParam = "jcaptchaCode";//前台提交的验证码参数名
4. **private** String failureKeyAttribute = "shiroLoginFailure"; //验证失败后存储到的属性名
5. **public** **void** setJcaptchaEbabled(**boolean** jcaptchaEbabled) {
6. **this**.jcaptchaEbabled = jcaptchaEbabled;
7. }
8. **public** **void** setJcaptchaParam(String jcaptchaParam) {
9. **this**.jcaptchaParam = jcaptchaParam;
10. }
11. **public** **void** setFailureKeyAttribute(String failureKeyAttribute) {
12. **this**.failureKeyAttribute = failureKeyAttribute;
13. }
14. **protected** **boolean** isAccessAllowed(ServletRequest request, ServletResponse response, Object mappedValue) **throws** Exception {
15. //1、设置验证码是否开启属性，页面可以根据该属性来决定是否显示验证码
16. request.setAttribute("jcaptchaEbabled", jcaptchaEbabled);
18. HttpServletRequest httpServletRequest = WebUtils.toHttp(request);
19. //2、判断验证码是否禁用 或不是表单提交（允许访问）
20. **if** (jcaptchaEbabled == **false** || !"post".equalsIgnoreCase(httpServletRequest.getMethod())) {
21. **return** **true**;
22. }
23. //3、此时是表单提交，验证验证码是否正确
24. **return** JCaptcha.validateResponse(httpServletRequest, httpServletRequest.getParameter(jcaptchaParam));
25. }
26. **protected** **boolean** onAccessDenied(ServletRequest request, ServletResponse response) **throws** Exception {
27. //如果验证码失败了，存储失败key属性
28. request.setAttribute(failureKeyAttribute, "jCaptcha.error");
29. **return** **true**;
30. }
31. }

**七、MyFormAuthenticationFilter**

用于验证码验证的Shiro拦截器在用于身份认证的拦截器之前运行；但是如果验证码验证拦截器失败了，就不需要进行身份认证拦截器流程了；所以需要修改下如FormAuthenticationFilter身份认证拦截器，当验证码验证失败时不再走身份认证拦截器。

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** MyFormAuthenticationFilter **extends** FormAuthenticationFilter {
2. **protected** **boolean** onAccessDenied(ServletRequest request, ServletResponse response, Object mappedValue) **throws** Exception {
3. **if**(request.getAttribute(getFailureKeyAttribute()) != **null**) {
4. **return** **true**;
5. }
6. **return** **super**.onAccessDenied(request, response, mappedValue);
7. }
8. }

即如果之前已经错了，那直接跳过即可。

**八、spring-config-shiro.xml**

**Java代码  [收藏代码](javascript:void())**

1. <!-- 基于Form表单的身份验证过滤器 -->
2. <bean id="authcFilter"
3. **class**="com.github.zhangkaitao.shiro.chapter22.jcaptcha.MyFormAuthenticationFilter">
4. <property name="usernameParam" value="username"/>
5. <property name="passwordParam" value="password"/>
6. <property name="rememberMeParam" value="rememberMe"/>
7. <property name="failureKeyAttribute" value="shiroLoginFailure"/>
8. </bean>
9. <bean id="jCaptchaValidateFilter"
10. **class**="com.github.zhangkaitao.shiro.chapter22.jcaptcha.JCaptchaValidateFilter">
11. <property name="jcaptchaEbabled" value="true"/>
12. <property name="jcaptchaParam" value="jcaptchaCode"/>
13. <property name="failureKeyAttribute" value="shiroLoginFailure"/>
14. </bean>
15. <!-- Shiro的Web过滤器 -->
16. <bean id="shiroFilter" **class**="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
17. <property name="securityManager" ref="securityManager"/>
18. <property name="loginUrl" value="/login"/>
19. <property name="filters">
20. <util:map>
21. <entry key="authc" value-ref="authcFilter"/>
22. <entry key="sysUser" value-ref="sysUserFilter"/>
23. <entry key="jCaptchaValidate" value-ref="jCaptchaValidateFilter"/>
24. </util:map>
25. </property>
26. <property name="filterChainDefinitions">
27. <value>
28. /**static**/\*\* = anon
29. /jcaptcha\* = anon
30. /login = jCaptchaValidate,authc
31. /logout = logout
32. /authenticated = authc
33. /\*\* = user,sysUser
34. </value>
35. </property>
36. </bean>

**九、login.jsp登录页面**

**Java代码  [收藏代码](javascript:void())**

1. <c:**if** test="${jcaptchaEbabled}">
2. 验证码：
3. <input type="text" name="jcaptchaCode">
4. <img **class**="jcaptcha-btn jcaptcha-img"
5. src="${pageContext.request.contextPath}/jcaptcha.jpg" title="点击更换验证码">
6. <a **class**="jcaptcha-btn" href="javascript:;">换一张</a>
7. <br/>
8. </c:**if**>

根据jcaptchaEbabled来显示验证码图片。

**十、测试**

输入<http://localhost:8080/chapter22>将重定向到登录页面；输入正确的用户名/密码/验证码即可成功登录，如果输入错误的验证码，将显示验证码错误页面：

