链滴

springSecurity 图片验证码

作者：2457081614

原文链接：https://ld246.com/article/1575992765083

来源网站：链滴

简介

验证码（CAPTCHA）的全称是 Completely Automated Public Turing test to tell Computers and Humans Apart，翻译过来就是“全自动区分计算机和人类的图灵测试”。通俗地讲，验证码就是为防
止恶意用户暴力重试而设置的。不管是用户注册、用户登录，还是论坛发帖，如果不加以限制，一旦某些恶意用户利用计算机发起无限重试，就很容易使系统遭受破坏。

通过过滤器实现

自定义过滤器

在Spring Security中，实现验证码校验的方式有很多种，最简单的方式就是自定义一个专门处理验证码逻辑的过滤器，将其添加到Spring Security过滤器链的合适位置。当匹配到登录请求时，立刻对验证码进行校验，成功则放行，失败则提前结束整个验证请求。

图形验证码过滤器

- 1、验证码图片准备

毋庸置疑，要想实现图形验证码校验功能，首先应当有一个用于获取图形验证码的 API。绘制图形验证码的方法有很多，使用开源的验证码组件即可，例如kaptcha

```
<dependency>
        <groupId>com.github.penggle</groupId>
        <artifactId>kaptcha</artifactId>
        <version>2.3.2</version>
    </dependency>
```

首先配置一个kaptcha实例

```
package club.xwzzy.springbootsecurity_imageverification.config;

import com.google.code.kaptcha.Producer;
import com.google.code.kaptcha.impl.DefaultKaptcha;
import com.google.code.kaptcha.util.Config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.Properties;

@Configuration
public class CaptchaConfig {

    @Bean
    public Producer captcha() {
        // 配置图形验证码的基本参数
        Properties properties = new Properties();
        // 图片宽度
        properties.setProperty("kaptcha.image.width", "150");
        // 图片长度
```

```
        properties.setProperty("kaptcha.image.height", "50");
        // 字符集
        properties.setProperty("kaptcha.textproducer.char.string", "0123456789");
        // 字符长度
        properties.setProperty("kaptcha.textproducer.char.length", "4");
        Config config = new Config(properties);
        // 使用默认的图形验证码实现，当然也可以自定义实现
        DefaultKaptcha defaultKaptcha = new DefaultKaptcha();
        defaultKaptcha.setConfig(config);
        return defaultKaptcha;
    }

}
```

接着创建一个CaptchaController，用于获取图形验证码。

```
package club.xwzzy.springbootsecurity_imageverification.controller;

import com.google.code.kaptcha.Producer;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

import javax.imageio.ImageIO;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.awt.image.BufferedImage;
import java.io.IOException;

@Controller
public class CaptchaController {

    @Autowired
    private Producer captchaProducer;

    @GetMapping("/captcha.jpg")
    public void getCaptcha(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        // 设置内容类型
        response.setContentType("image/jpeg");
        // 创建验证码文本
        String capText = captchaProducer.createText();
        // 将验证码文本设置到session
        request.getSession().setAttribute("captcha", capText);
        // 创建验证码图片
        BufferedImage bi = captchaProducer.createImage(capText);
        // 获取响应输出流
        ServletOutputStream out = response.getOutputStream();
        // 将图片验证码数据写到响应输出流
        ImageIO.write(bi, "jpg", out);
        // 推送并关闭响应输出流
        try {
```

```
            out.flush();
        } finally {
            out.close();
        }
    }

}
```

当用户访问/captcha.jpg时，即可得到一张携带验证码的图片，验证码文本则被存放到session中用后续的校验。

- 2、自定义异常

```
package club.xwzzy.springbootsecurity_imageverification.exception;

import org.springframework.security.core.AuthenticationException;

public class VerificationCodeException extends AuthenticationException {

    public VerificationCodeException () {
        super("图形验证码校验失败");
    }

}
```

- 3、自定义过滤器

```
package club.xwzzy.springbootsecurity_imageverification.filter;


import club.xwzzy.springbootsecurity_imageverification.authentication.SecurityAuthentication
ailureHandler;
import club.xwzzy.springbootsecurity_imageverification.exception.VerificationCodeException;
import org.springframework.security.web.authentication.AuthenticationFailureHandler;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

public class VerificationCodeFilter extends OncePerRequestFilter {

    private AuthenticationFailureHandler authenticationFailureHandler = new SecurityAuthenti
ationFailureHandler();

    @Override
    protected void doFilterInternal(HttpServletRequest httpServletRequest, HttpServletRespon
e httpServletResponse, FilterChain filterChain) throws ServletException, IOException {
```

```
        // 非登录请求不校验验证码
        if (!"/login".equals(httpServletRequest.getRequestURI())) {
            filterChain.doFilter(httpServletRequest, httpServletResponse);
        } else {
            try {
                verificationCode(httpServletRequest);
                filterChain.doFilter(httpServletRequest, httpServletResponse);
            } catch (VerificationCodeException e) {
                authenticationFailureHandler.onAuthenticationFailure(httpServletRequest, httpServl
tResponse, e);
            }
        }
    }

    public void verificationCode (HttpServletRequest httpServletRequest) throws VerificationC
deException {
        String requestCode = httpServletRequest.getParameter("captcha");
        HttpSession session = httpServletRequest.getSession();
        String savedCode = (String) session.getAttribute("captcha");
        if (!StringUtils.isEmpty(savedCode)) {
            // 随手清除验证码，不管是失败还是成功，所以客户端应在登录失败时刷新验证码
            session.removeAttribute("captcha");
        }
        // 校验不通过抛出异常
        if (StringUtils.isEmpty(requestCode) || StringUtils.isEmpty(savedCode) || !requestCode.eq
als(savedCode)) {
            throw new VerificationCodeException();
        }
    }
}
```

- 4、添加过滤器

```
 // 将过滤器添加在UsernamePasswordAuthenticationFilter之前
        http.addFilterBefore(new VerificationCodeFilter(), UsernamePasswordAuthenticationFilter
class);
```

- 5、html

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>登录</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body>
        <div class="login">
            <h2>Acced Form</h2>
            <div class="login-top">
                <h1>LOGIN FORM</h1>
                <form action="/login" method="post">
                    <input type="text" name="username" placeholder="username" />
                    <input type="password" name="password" placeholder="password" />
```

```
                <div style="display: flex;">
                    <!-- 新增图形验证码的输入框 -->
                    <input type="text" name="captcha" placeholder="captcha" />
                    <!-- 图片指向图形验证码API -->
                    <img src="/captcha.jpg" alt="captcha" height="50px" width="150px" style="
argin-left: 20px;">
                </div>
                <div class="forgot">
                    <a href="#">forgot Password</a>
                    <input type="submit" value="Login" >
                </div>
            </form>
        </div>
        <div class="login-bottom">
            <h3>New User  <a href="#">Register</a>  Here</h3>
        </div>
    </div>

    <style>
        html,body,div,span,applet,object,iframe,h1,h2,h3,h4,h5,h6,p,blockquote,pre,a,abbr,acr
nym,address,big,cite,code,del,dfn,em,img,ins,kbd,q,s,samp,small,strike,strong,sub,sup,tt,var,b,u
i,dl,dt,dd,ol,nav ul,nav li,fieldset,form,label,legend,table,caption,tbody,tfoot,thead,tr,th,td,articl
,aside,canvas,details,embed,figure,figcaption,footer,header,hgroup,menu,nav,output,ruby,sect
on,summary,time,mark,audio,video{margin:0;padding:0;border:0;font-size:100%;font:inherit;ve
tical-align:baseline;}
        article, aside, details, figcaption, figure,footer, header, hgroup, menu, nav, section {disp
ay: block;}
        ol,ul{list-style:none;margin:0;padding:0;}
        blockquote,q{quotes:none;}
        blockquote:before,blockquote:after,q:before,q:after{content:'';content:none;}
        table{border-collapse:collapse;border-spacing:0;}
        a{text-decoration:none;}
        nav.vertical ul li{    display:block;}
        nav.horizontal ul li{    display: inline-block;}
        img{max-width:100%;}
        body{
            background: #3f3f3f;
            padding:100px 0px 30px 0px;
            font-family: 'Roboto', sans-serif;
            font-size: 100%;
        }
        .login {
            width: 32%;
            margin: 0 auto;
        }
        .login h2 {
            font-size: 30px;
            font-weight: 700;
            color: #fff;
            text-align: center;
            margin: 0px 0px 50px 0px;
            font-family: 'Droid Serif', serif;
        }
        .login-top {
```

```
    background: #E1E1E1;
    border-radius: 25px 25px 0px 0px;
    -webkit-border-radius:  25px 25px 0px 0px;
    -moz-border-radius: 25px 25px 0px 0px;
    -o-border-radius: 25px 25px 0px 0px;
    padding: 40px 60px;
}
.login-top h1 {
    text-align: center;
    font-size: 27px;
    font-weight: 500;
    color: #F45B4B;
    margin: 0px 0px 20px 0px;
}
.login-top input[type="text"] {
    outline: none;
    font-size: 15px;
    font-weight: 500;
    color: #818181;
    padding: 15px 20px;
    background: #CACACA;
    border: 1px solid #ccc;
    border-radius:25px;
    -webkit-border-radius: 25px;
    -moz-border-radius: 25px;
    -o-border-radius: 25px;
    margin: 0px 0px 12px 0px;
    width: 88%;
    -webkit-appearance: none;
}
.login-top input[type="password"]{
    outline: none;
    font-size: 15px;
    font-weight: 500;
    color: #818181;
    padding: 15px 20px;
    background: #CACACA;
    border: 1px solid #ccc;
    border-radius:25px;
    -webkit-border-radius: 25px;
    -moz-border-radius: 25px;
    -o-border-radius: 25px;
    margin: 0px 0px 12px 0px;
    width: 88%;
    -webkit-appearance: none;
}
.forgot  a{
    font-size: 13px;
    font-weight: 500;
    color: #F45B4B;
    display: inline-block;
    border-right: 2px solid #F45B4B;
    padding: 0px 7px 0px 0px;
}
```

```
.forgot  a:hover{
    color: #818181;
}
.forgot input[type="submit"] {
    background: #F45B4B;
    color: #FFF;
    font-size: 17px;
    font-weight: 400;
    padding: 8px 7px;
    width: 20%;
    display: inline-block;
    cursor: pointer;
    border-radius: 6px;
    -webkit-border-radius: 19px;
    -moz-border-radius: 6px;
    -o-border-radius: 6px;
    margin: 0px 7px 0px 3px;
    outline: none;
    border: none;
}
.forgot input[type="submit"]:hover {
    background:#818181;
    transition: 0.5s all;
    -webkit-transition: 0.5s all;
    -moz-transition: 0.5s all;
    -o-transition: 0.5s all;
}
.forgot {
    text-align: right;
}
.login-bottom {
    background: #E15748;
    padding: 30px 65px;
    border-radius: 0px 0px 25px 25px;
    -webkit-border-radius:  0px 0px 25px 25px;
    -moz-border-radius: 0px 0px 25px 25px;
    -o-border-radius: 0px 0px 25px 25px;
    text-align: right;
    border-top: 2px solid #AD4337;
}
.login-bottom h3 {
    font-size: 14px;
    font-weight: 500;
    color: #fff;
}
.login-bottom h3 a {
    font-size: 25px;
    font-weight: 500;
    color: #fff;
}
.login-bottom h3 a:hover {
    color:#696969;
    transition: 0.5s all;
    -webkit-transition: 0.5s all;
```

```
    -moz-transition: 0.5s all;
    -o-transition: 0.5s all;
}
.copyright p {
    font-size: 15px;
    font-weight: 400;
    color: #fff;
}
.copyright p a{
    font-size: 15px;
    font-weight: 400;
    color: #E15748;
}
.copyright p a:hover{
    color: #fff;
    transition: 0.5s all;
    -webkit-transition: 0.5s all;
    -moz-transition: 0.5s all;
    -o-transition: 0.5s all;
}
@media(max-width:1440px){
    .login {
        width: 35%;
    }
}
@media(max-width:1366px){
    .login {
        width: 37%;
    }
}
@media(max-width:1280px){
    .login {
        width: 40%;
    }
}
@media(max-width:1024px){
    .login {
        width: 48%;
    }
}
@media(max-width:768px){
    .login {
        width: 65%;
    }
    .login-top h1 {
        font-size: 25px;
    }
    .login-bottom h3 a {
        font-size: 22px;
    }
    body {
        padding: 100px 0px 0px 0px;
    }
    .login h2 {
```

```
        font-size: 28px;
    }
}
@media(max-width:640px){
    .login-top h1 {
        font-size: 23px;
    }
    .forgot input[type="submit"] {
        font-size: 15px;
        width: 22%;
    }
    .login-top input[type="text"] {
        padding: 12px 20px;
    }
    .login-top input[type="password"] {
        padding: 12px 20px;
    }
    .login-bottom h3 a {
        font-size: 19px;
    }
    .login-bottom h3 {
        font-size: 13px;
    }
    body {
        padding: 110px 0px 0px 0px;
    }
}
@media(max-width:480px){
    .login {
        width: 80%;
    }
    .login-top h1 {
        font-size: 21px;
    }
    .login-top input[type="text"] {
        width: 85%;
    }
    .login-top {
        padding: 30px 40px;
    }
    .login-top input[type="password"] {
        width: 85%;
    }
    .login h2 {
        font-size: 25px;
    }
}
@media(max-width:320px){
    .login {
        width: 90%;
    }
    .login-top {
        padding: 20px 25px;
    }
```

```
            .login-top input[type="text"] {
                width: 81%;
                padding: 10px 20px;
                font-size: 13px;
                margin: 0px 0px 7px 0px;
            }
            .login-top input[type="password"] {
                width: 81%;
                padding: 10px 20px;
                font-size: 13px;
                margin: 0px 0px 7px 0px;
            }
            .forgot input[type="submit"] {
                font-size: 11px;
                width: 25%;
                padding: 6px 7px;
            }
            .forgot  a {
                font-size: 11px;
            }
            .login-bottom {
                padding: 20px 25px;
            }
            .login-bottom h3 {
                font-size: 11px;
            }
            .login-bottom h3 a {
                font-size: 17px;
            }
            body {
                padding: 50px 0px 0px 0px;
            }
            .copyright p {
                font-size: 13px;
            }
            .copyright p a{
                font-size: 13px;
            }
            .login h2 {
                font-size: 23px;
                margin:0px 0px 35px 0px;
            }
        }
    </style>
</body>
</html>
```

至此，使用过滤器的方式实现验证码结束，属于Servlet层面，简单、易理解。

代码地址