添加验证码大致可以分为三个步骤：

1. 根据随机数生成验证码图片；
2. 将验证码图片显示到登录页面；
3. 认证流程中加入验证码校验。

Spring Security的认证校验是由UsernamePasswordAuthenticationFilter过滤器完成的，所以我们的验证码校验逻辑应该在这个过滤器之前。

#### 1、生成图形验证码

验证码功能需要用到spring-social-config依赖：

<dependency>

<groupId>org.springframework.social</groupId>  
 <artifactId>spring-social-config</artifactId>  
</dependency>

首先定义一个验证码对象ImageCode：

@Data

public class ImageCode {  
  
 */\*\*  
 \* image图片  
 \*/* private BufferedImage image;  
  
 */\*\*  
 \* 验证码  
 \*/* private String code;  
  
 */\*\*  
 \* 过期时间  
 \*/* private LocalDateTime expireTime;  
  
 public ImageCode(BufferedImage image, String code, int expireIn) {  
 this.image = image;  
 this.code = code;  
 this.expireTime = LocalDateTime.*now*().plusSeconds(expireIn);  
 }  
  
 public ImageCode(BufferedImage image, String code, LocalDateTime expireTime) {  
 this.image = image;  
 this.code = code;  
 this.expireTime = expireTime;  
 }  
  
 */\*\*  
 \* 用于判断验证码是否过期  
 \*/* boolean isExpire() {  
 return LocalDateTime.*now*().isAfter(expireTime);  
 }  
}

接着定义一个ValidateCodeController，用于处理生成验证码请求：

*/\*\**

*\* SessionStrategy封装了一些处理Session的方法，  
 \* 包含了setAttribute、getAttribute和removeAttribute方法  
 \*/*private SessionStrategy sessionStrategy = new HttpSessionSessionStrategy();  
  
@GetMapping("/code/image")  
public void createCode(HttpServletRequest request, HttpServletResponse response) throws IOException {  
 ImageCode imageCode = createImageCode();  
 //将生成的验证码对象存储到Session中  
 sessionStrategy.setAttribute(new ServletWebRequest(request), *SESSION\_KEY\_IMAGE\_CODE*, imageCode);  
 //通过IO流将生成的图片输出到登录页面上  
 ImageIO.*write*(imageCode.getImage(), "jpeg", response.getOutputStream());  
}

生成图片验证码，createImageCode()方法：

private ImageCode createImageCode() {

int width = 100;  
 int height = 36;  
 // 验证码位数  
 int length = 4;  
 // 验证码有效时间 60s  
 int expireIn = 60;  
  
 BufferedImage image = new BufferedImage(width, height, BufferedImage.*TYPE\_INT\_RGB*);  
  
 Graphics g = image.getGraphics();  
  
 Random random = new Random();  
  
 g.setColor(getRandColor(200, 250));  
 g.fillRect(0, 0, width, height);  
 g.setFont(new Font("Times New Roman", Font.*ITALIC*, 20));  
 g.setColor(getRandColor(160, 200));  
 for (int i = 0; i < 155; i++) {  
 int x = random.nextInt(width);  
 int y = random.nextInt(height);  
 int xl = random.nextInt(12);  
 int yl = random.nextInt(12);  
 g.drawLine(x, y, x + xl, y + yl);  
 }  
  
 StringBuilder sRand = new StringBuilder();  
 for (int i = 0; i < length; i++) {  
 String rand = String.*valueOf*(random.nextInt(10));  
 sRand.append(rand);  
 g.setColor(new Color(20 + random.nextInt(110), 20 + random.nextInt(110), 20 + random.nextInt(110)));  
 g.drawString(rand, 13 \* i + 6, 16);  
 }  
  
 g.dispose();  
  
 return new ImageCode(image, sRand.toString(), expireIn);  
}

获取随机颜色，getRandColor()方法：

private Color getRandColor(int fc, int bc) {

Random random = new Random();  
 if (fc > 255) {  
 fc = 255;  
 }  
  
 if (bc > 255) {  
 bc = 255;  
 }  
 int r = fc + random.nextInt(bc - fc);  
 int g = fc + random.nextInt(bc - fc);  
 int b = fc + random.nextInt(bc - fc);  
 return new Color(r, g, b);  
}

#### 2、改造登录页

登录页加入如下代码：

<span style="display: inline">

<input type="text" name="imageCode" placeholder="验证码" style="width: 50%;"/>  
 <img src="/code/image"/>  
</span>

src属性对应ValidateController的createImageCode方法。

要使生成验证码的请求不被拦截，需要在BrowserSecurityConfig的configure方法中配置面拦截。

.antMatchers("/authentication/require",

"/login.html",  
 "/code/image").permitAll() // 无需认证的请求路径

启动项目，访问<http://localhost:8080/login.html>



#### 3、验证流程添加验证码校验

自定义验证码异常，继承AuthenticationException类。校验验证码的过程中，可能会抛出各种验证码类型的异常，比如“验证码错误”、“验证码已过期”等

public class ValidateCodeException extends AuthenticationException {

private static final long *serialVersionUID* = 5022575393500654458L;  
  
 ValidateCodeException(String message) {  
 super(message);  
 }  
}

Spring Security实际上是由许多过滤器组成的过滤器链，处理用户登录逻辑的过滤器为