链滴

Vue Springboot （包括后端解决跨域）实现登录验证码功能详细完整版

作者：Acechengui

原文链接：https://ld246.com/article/1608278023871

来源网站：链滴

利用Hutool 基于Vue、ElementUI、Springboot （跨域）实现登验证码功能

前言

提示：实现此功能建立在你至少能够基础使用vue、elementui、springboot

一、Hutool是什么?

工欲善其事必先利其器！ Hutool 就是这么一款超级强力的工具类。

在大家日常工作中，都常常会做如下这些非常繁琐的工作：

1. 日期与字符串转换
2. 文件操作
3. 转码与反转码

4. 随机数生成
5. 压缩与解压
6. 编码与解码
7. CVS文件操作
8. 缓存处理
9. 加密解密
10. 定时任务
11. 邮件收发
12. 二维码创建
13. FTP 上传与下载
14. 图形验证码生成

等等等等...

以上这些事情，要么自己动手写代码，要么从零零碎碎的各个不同的地方去找各种零零散散的代码来造成自己需要的样子。

现在不用了，您只需要一个 hutool 那么这些功能都是现成滴了，而且非常好用。。

想去使用了解的我这里给个传送门

HuTool官方文档传送门

二、下面开始步入正题：使用步骤

1.先引入Hutool依赖

可全部引入，也可以引入部分，我这里就全部引入了：

```
<!-- 在pom.xml文件中引入hutool -->
<dependency>
    <groupId>cn.hutool</groupId>
    <artifactId>hutool-all</artifactId>
    <version>5.2.5</version>
</dependency>
```

2.控制层

我这里先讲一下思路：

前端发起一个请求，后端控制层利用hutool编写生成验证码的代码，并将验证码的内容保存到sessio中，在登录控制层中获取这个session的值,与当前登录前端输入的验证码的值对比，若相同则验证成，既可以执行后面的操作，若不相同，则要返回并提示验证码错误。

下面是控制层代码：

```
    //生成验证码
    @GetMapping("/api/getCode")
    public void  getCode(HttpServletResponse response, HttpSession session) throws IOExcept
on {
```

```
        // 定义图形验证码的长、宽、验证码位数、线性数量
        //还有更多功能自行研究，这里只做简单实现
        LineCaptcha lineCaptcha = CaptchaUtil.createLineCaptcha(125, 50,4,5);
        //保存到session
        session.setAttribute("code", lineCaptcha.getCode());
        //响应输出流
        ServletOutputStream outputStream = response.getOutputStream();
        //将生成的验证码图片通过流的方式返回给前端
        ImageIO.write(lineCaptcha.getImage(), "JPEG", outputStream);
    }
```

下面再是登录控制，因为我这里是用了shiro,不懂shiro的可以自行百度，不过其实也不用管，重点看证码实现，然后再结合自己业务使用。

```
//登录控制
@PostMapping("/api/login")
    public Result login(@RequestBody User requestUser,HttpSession session) {
        //获得存储在session中的验证码（不会shiro的不用管，看这有关验证码的就行）
        String sessionCheckCode = (String) session.getAttribute("code");
        //防XFS HtmlUtils.htmlEscape(username);
        String username = requestUser.getUsername();
        username = HtmlUtils.htmlEscape(username);

        Subject subject = SecurityUtils.getSubject();
        //判断验证码是否与前端输入的值相同 （不会shiro的不用管，看这有关验证码的就行）
        if (requestUser.getCaptcha() != null && sessionCheckCode.equals(requestUser.getCaptc
a())) {
            //移除验证码
            session.removeAttribute("code");
            //验证认证状态
            if (!subject.isAuthenticated()) {
                // 组装一个token
                UsernamePasswordToken token = new UsernamePasswordToken(username, reques
User.getPassword());
                // 开启记住我功能
                if (requestUser.isRememberMe()) {
                    token.setRememberMe(true);
                }
                try {
                    //验证成功 做认证
                    subject.login(token);
                    User user = userService.findByUsername(username);
                    if (!user.isEnabled()) {
                        return ResultFactory.buildFailResult("该用户已被禁用");
                    }
                    return ResultFactory.buildSuccessResult(username);
                } catch (IncorrectCredentialsException e) {
                    return ResultFactory.buildFailResult("密码错误");
                } catch (UnknownAccountException e) {
                    return ResultFactory.buildFailResult("账号不存在");
                }
            }
        }
        return  ResultFactory.buildFailResult("验证码不正确");
```

```
    }
```

写到这里，还是把实体类也提供一下吧，好人做到底。 提示：我这里用了jpa、lombok你们不用管就里面放了一个属性存放验证码。 详细里面都写了

```
package top.wangxingjun.separate.entity;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import lombok.*;
import org.hibernate.validator.constraints.Length;

import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.Pattern;
import java.util.List;

/**
 * 用户类
 *
 * @author wxj
 * @Date 2020/8/10
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "user")
@ToString
@JsonIgnoreProperties({"handler","hibernateLazyInitializer"})
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    /**
     * Username.
     */
    @NotEmpty(message = "账号不能为空")
    private String username;

    /**
     * Password.
     */
    @NotEmpty(message = "密码不能为空")
    @Length(min = 3,message = "密码长度不能低于3位")
    private String password;

    /**
```

```
     * Salt for encoding.
     */
    private String salt;

    /**
     * Real name.
     */
    private String name;

    /**
     * Phone number.
     */
    @NotEmpty(message = "手机号码不能为空")
    @Pattern(regexp = "^1[3456789]\\d{9}$",message = "手机格式不正确")
    private String phone;

    /**
     * Email address.
     *
     * A Email address can be null,but should be correct if exists.
     */
    @NotEmpty(message = "邮箱不能为空")
    @Email(message = "请输入正确的邮箱")
    private String email;

    /**
     * User status.
     */
    private boolean enabled;

    /**
     * Transient property for storing role owned by current user.
     */
    @Transient
    private List<AdminRole> roles;

    @Transient
    //看这个属性就行 我这里使用了注解，所以不用写getter/setter方法,你们没用上的当然得自己手
生成
    private String captcha;//验证码

    /**
     * Transient property for rememberMe pwd by current user.
     */
    @Transient
    private boolean rememberMe;//记住我
}
```

3.下面到前端登录界面

这里我只贴关键代码

```
<el-form-item>
    <span class="svg-container svg-container_login">
      <svg class="icon" aria-hidden="true">
        <use xlink:href="#icon-yanzhengma"></use>
      </svg>
    </span>
    <el-input
      v-model="loginForm.captcha"
      placeholder="验证码"
      clearable
      style="width:42%;"
      maxlength="4"
    />
    <a href="javascript:void(0)" @click="createCode()"
      ><img :src="verifitycode" style="width:20%;" alt="图片加载失败"
    /></a>
  </el-form-item>
```

下面是data、mounted() 、methods中相关代码：

```
 data() {
  return {
    loginForm: {
      username: "admin",
      password: "",
      captcha: "",
      rememberMe: false
    },
    verifitycode: ""
  };
},
mounted() {
  //相当于初始化
  this.createCode();
},
methods: {
  //获取验证码
  createCode() {
    //二进制方式读取图片流  验证码
    this.$axios
      .get("/getCode", {
        responseType: "arraybuffer"
      })
      .then(response => {
        return (
          "" +
          btoa(
            new Uint8Array(response.data).reduce(
              (data, byte) => data + String.fromCharCode(byte),""
            )
          )
        );
      })
```

```
        .then(data => {
          this.verifitycode = data;
        });
    }
```

上面最核心的便是通过二进制流的方式获取后端返回回来的图片验证码，当然前后端返利肯定还牵扯跨域问题，跨域解决方式有几种。咳咳，咋又扯到这了，额...好吧。那我也把我后端跨域解决的部分出来吧（还可以用注解的方式@CrossOrigin，自行了解），我这里使用重写WebMvcConfigurer方进行CORS实现跨域访问。

```
package top.wangxingjun.separate.config;

import org.springframework.boot.SpringBootConfiguration;
import org.springframework.web.servlet.config.annotation.*;

/**
 * @author wxj
 * @Date 2020/8/10
 */
@SpringBootConfiguration
public class MyWebConfigurer implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        //所有请求都允许跨域，使用这种配置方法就不能在 interceptor 中再配置 header 了
        registry.addMapping("/**")
                .allowCredentials(true)
                .allowedOrigins("*")
                .allowedMethods("POST", "GET", "PUT", "OPTIONS", "DELETE")
                .allowedHeaders("*")
                .maxAge(3600);
    }

}
```

好了~结束