Please visit website: http://cxyroad.com

理大量用户和应用程序的企业尤为重要。

`下面 V 哥通过一个实际案例来介绍单点登录的实现。`

Java 单点登录 (SSO) 案例代码

单点登录 (SSO) 是一种允许用户使用一组登录凭据访问多个应用程序的系统。以下是一个简单的 Java 单点登录实现示例，使用 Spring Boot 框架。

1. 创建 Spring Boot 项目

首先，使用 Spring Initializr ([start.spring.io/](http://cxyroad.com/”https://start.spring.io/”)) 创建一个 Spring Boot 项目，添加以下依赖：

* Spring Web
* Spring Security
* Thymeleaf

2. 配置文件

在 src/main/resources/application.properties 中配置一些基本的属性。

```

server.port=8080
spring.thymeleaf.cache=false

```

3. 安全配置

创建一个安全配置类 SecurityConfig.java。

```

```

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.b
```

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class MainController {

    @GetMapping("/")
    public String home() {
        return "home";
    }

    @GetMapping("/login")
    public String login() {
        return "login";
    }

    @GetMapping("/register")
    public String register() {
        return "register";
    }
}
```

6. 首页和注册页面

创建首页和注册页面的 Thymeleaf 模板。

* home.html
* register.html

注意事项

* 安全性：确保使用 HTTPS 协议来保护用户数据的传输。
* 密码存储：不要以明文形式存储密码，使用强哈希算法（如 BCrypt）进行加密存储。
* 会话管理：合理配置会话超时和会话管理策略，防止会话劫持。
* 跨站请求伪造 (CSRF)：启用 CSRF 保护，防止恶意网站的攻击。
* 错误处理：合理处理登录失败和认证错误的情况，避免泄露敏感信息。
* 用户权限管理：根据用户角色分配不同的访问权限，确保系统的安全性。
* 日志记录：记录关键操作的日志，便于问题追踪和安全审计。
* 多因素认证：考虑引入多因素认证，增加系统的安全性。
* 兼容性：确保单点登录系统与不同的应用程序兼容，特别是那些使用不同技术栈的应用程序。
* 维护和更新：定期更新系统和依赖库，修复已知的安全漏洞。

通过以上步骤和注意事项，你可以构建一个基本的单点登录系统，并确保其安全性和可用性。

原文链接: https://juejin.cn/post/7393168663262920742