基于Spring MVC的用户注册与登录系统实验报告

系统功能概述

1.用户注册：创建新账户

2.用户登录：身份验证与会话管理

3.主界面：展示所有注册用户

4.安全控制：防止未授权访问

核心代码实现

1. 用户实体类 (User.java)

java

public class User {

private Integer id;

private String uname; // 用户名

private String password; // 加密密码

private Date createTime; // 注册时间

// Getters & Setters

}

2. 注册功能实现

注册页面 (register.jsp)：

jsp

<form action="${pageContext.request.contextPath}/register" method="post">

用户名：<input type="text" name="uname"><br>

密码：<input type="password" name="password"><br>

<input type="submit" value="注册">

</form>

注册控制器 (UserController.java)：

java

@PostMapping("/register")

public String register(

@RequestParam String uname,

@RequestParam String password,

RedirectAttributes ra) {

// 1. 密码加密

String encryptedPwd = BCrypt.hashpw(password, BCrypt.gensalt());

// 2. 创建用户对象

User user = new User();

user.setUname(uname);

user.setPassword(encryptedPwd);

user.setCreateTime(new Date());

// 3. 保存到数据库

userService.saveUser(user);

// 4. 重定向到登录页

ra.addFlashAttribute("msg", "注册成功，请登录");

return "redirect:/login";

}

3. 登录功能实现

登录页面 (login.jsp)：

jsp

<form action="${pageContext.request.contextPath}/login" method="post">

用户名：<input type="text" name="uname"><br>

密码：<input type="password" name="password"><br>

<input type="submit" value="登录">

${errorMsg} <!-- 错误信息显示 -->

</form>

登录控制器 (UserController.java)：

java

@PostMapping("/login")

public String login(

@RequestParam String uname,

@RequestParam String password,

HttpSession session,

Model model) {

// 1. 查询用户

User user = userService.findByUname(uname);

// 2. 验证账户

if (user == null || !BCrypt.checkpw(password, user.getPassword())) {

model.addAttribute("errorMsg", "用户名或密码错误");

return "login"; // 返回登录页

}

// 3. 创建会话

session.setAttribute("currentUser", user);

// 4. 重定向到主界面

return "redirect:/index";

}

4. 主界面实现

用户控制器 (UserController.java)：

java

@GetMapping("/index")

public String mainPage(Model model, HttpSession session) {

// 1. 检查登录状态

if (session.getAttribute("currentUser") == null) {

return "redirect:/login"; // 未登录重定向

}

// 2. 查询所有用户

List<User> allUsers = userService.findAllUsers();

// 3. 传递给视图

model.addAttribute("allUsers", allUsers);

return "index";

}

主界面视图 (index.jsp)：

jsp

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>

<body>

<h3>欢迎您，${sessionScope.currentUser.uname}！</h3>

查询出所有用户：

<br>

<c:forEach items="${allUsers}" var="map">

${map.uname} (注册于：<fmt:formatDate value="${map.createTime}" pattern="yyyy-MM-dd"/>)<br>

</c:forEach>

<a href="${pageContext.request.contextPath}/logout">退出登录</a>

</body>

</html>

5. 退出功能

java

@GetMapping("/logout")

public String logout(HttpSession session) {

session.invalidate(); // 销毁会话

return "redirect:/login";

}

安全增强设计

1.密码安全

使用BCrypt强哈希算法加密存储

java

// 加密

String hashed = BCrypt.hashpw(rawPassword, BCrypt.gensalt());

// 验证

BCrypt.checkpw(inputPassword, storedHash);

2.会话固定攻击防护

java

@PostMapping("/login")

public String login(...) {

// 登录成功后更换Session ID

session.invalidate();

session = request.getSession(true);

// ...设置用户信息

}

3.访问控制

java

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

.antMatchers("/index").authenticated() // 需登录

.antMatchers("/register", "/login").permitAll()

.and()

.formLogin()

.loginPage("/login");

}

}

4.CSRF防护

jsp

<!-- 在表单中添加CSRF令牌 -->

<input type="hidden"

name="${\_csrf.parameterName}"

value="${\_csrf.token}"/>

实验测试用例

测试场景 输入数据 预期结果

正常注册 uname=test, pwd=123 跳转登录页显示"注册成功"

重复用户名注册 uname=已存在用户 提示"用户名已存在"

正确登录 有效用户名/密码 进入主界面显示用户列表

错误密码登录 正确用户+错误密码 提示"用户名或密码错误"

未登录访问主界面 直接访问/index 重定向到登录页

登录后退出 点击退出链接 会话销毁，重定向到登录页

系统优化建议

1.前端验证

html

<input type="text" name="uname" required minlength="3">

<input type="password" name="password" required minlength="6">

2.密码强度检测

js

// register.jsp

document.querySelector("input[name='password']").addEventListener('input', e => {

let strength = checkPasswordStrength(e.target.value);

// 显示强度提示

});

3.记住我功能

java

@PostMapping("/login")

public String login(..., @RequestParam(required=false) Boolean rememberMe) {

if (rememberMe != null) {

// 创建持久化Cookie

}

}

4.登录日志审计

java

@Service

public class LoginLogger {

@Async

public void logLogin(User user, String ip) {

// 记录登录时间/IP到数据库

}

}

实验总结

本系统完整实现了基于Spring MVC的用户注册、登录和主界面功能：

1.技术栈

前端：JSP + JSTL + Bootstrap（可选）

后端：Spring MVC + Spring Security

安全：BCrypt加密 + CSRF防护 + 会话管理

2.关键实现

注册时密码加密存储

登录态会话管理

主界面数据动态绑定

分层架构（Controller/Service/DAO）

3.安全实践

密码哈希存储

会话固定防护

访问控制拦截器

参数化SQL查询防注入

4.扩展方向

添加邮箱验证

集成OAuth2第三方登录

实现权限分级管理

加入登录失败锁定机制

核心收获：通过本实验掌握了Spring MVC的核心开发模式，理解了Web应用的安全防护体系，实践了从数据存储到前端展示的完整流程。特别强调了"永远不存储明文密码"和"始终验证用户输入"两大安全原则。