学生姓名：钟玉生

指导老师：陈春华

信息系统安全

课程设计题目：网站的认证与访问控制

****

7/7/2017

目录

[一、 项目内容 2](#_Toc487461973)

[二、 项目要求 2](#_Toc487461974)

[(1) 采用用户名与密码方式实现用户认证： 2](#_Toc487461975)

[(2) 使用Spring Security进行基于角色的访问控制： 2](#_Toc487461976)

[三、 实验原理 2](#_Toc487461977)

[(1) Spring security的简单原理 2](#_Toc487461978)

[(2) Bcrypt算法的简单原理 3](#_Toc487461979)

[四、 实验环境 3](#_Toc487461980)

[(1) 硬件环境 3](#_Toc487461981)

[(2) 软件环境 3](#_Toc487461982)

[五、 实验过程描述 3](#_Toc487461983)

[（1）原infosec系统的脆弱性及不完整性： 3](#_Toc487461984)

[（2）加固infosec系统的措施及功能实现： 3](#_Toc487461985)

[（3）具体实现流程： 4](#_Toc487461986)

[1.实现登录页面login.jsp 4](#_Toc487461987)

[2. 使用spring security拦截url并实现登录角色控制 5](#_Toc487461988)

[六、 实验代码说明 7](#_Toc487461989)

[七、 实验结果与分析 9](#_Toc487461990)

[八、 参考 12](#_Toc487461991)

# 项目内容

项目为java web项目，是一个简单的登录，查看用户信息的服务，仅有两个业务：

1. 获取系统用户列表。
2. 查看用户详细信息。

本项目旨在为以上两个服务提供具体的实现，并且对登录进行认证，和不同登录用户之间的访问权限的控制。

# 项目要求

## (1) 采用用户名与密码方式实现用户认证：

使用bcrypt加密的形式保存密码

## (2) 使用Spring Security进行基于角色的访问控制：

* 1. 非系统认证用户，仅可访问主页“index.jsp”和登录页面“/login.jsp”等静态页面
  2. 管理员角色（ROLE\_ADMIN）看访问“获取系统用户列表业务”，并可访问“查看用户详细信息”业务
  3. 其他认证用户（ROLE\_USER），均可访问“查看用户详细信息”业务，但是仅能查看本用户的信息

# 实验原理

## Spring security的简单原理

使用众多的拦截器对url拦截，以此来管理权限。

整体流程：

* 1. 用户登录，会被AuthenticationProcessingFilter拦截，调用AuthenticationManager的实现，
  2. AuthenticationManager调用providerManager来获取用户验证信息
  3. 如果验证通过会将用户的权限信息封装一个User放到spring的全局缓存SecurityContextHolder中，让后面访问资源时使用
  4. 访问资源，会通过AbstractSecurityInterceptor拦截器拦截，调用FilterInvocationSecurityMetadataSource的方法来获取被拦截url锁需的全部权限，
  5. 调用授权管理器AccessDecisionManager通过spring的全局缓存securityContextHolder获取用户权限
  6. 检查结果如果权限够则返回，权限不够则报错并返回错误页面

## Bcrypt算法的简单原理

Bcrypt算法与MD5/SHA算法有一个很大区别，每次生成的hash值都是不同的，由于bcrypt每次将生成的salt混入生成的密文中，就可以免除存储salt，暴力破解起来也更加困难。

# 实验环境

## 硬件环境

1. **计算机：**Ubuntu 14.04

## 软件环境

1. **运行语言：**Java 1.8
2. **Web容器：**tomcat 8.5
3. **数据库：**mysql 5.5

# 实验过程描述

## （1）原infosec系统的脆弱性及不完整性：

1. 原系统没有实现登录控制，所有访问者均可访问系统提供的所有服务。然而这明显是不合理而且极其危险的，用户列表及用户信息应该只有授权者才能查看。

2. 原系统对用户的密码已明文的形式存储在数据库表中，这是不合理而且危险的，一旦系统变得不安全会导致用户信息的快速泄漏。

## （2）加固infosec系统的措施及功能实现：

1.以用户名和密码的形式实现用户认证，区别普通游客和注册用户

2. 用spring security机制实现角色控制，区别系统管理员与一般用户的权限与可访问资源

3. 使用bcrypt加密的密文存做数据库当中替换原来的明文密码，提高用户的密码安全，同时避免存储salt的约束。

## （3）具体实现流程：

### 1.实现登录页面login.jsp

表1 login.jsp

|  |
| --- |
| Login.jsp |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>InfoSec: 登录</title>  </head>  <body>  <!--<h1>请利用Spring Security机制实现用户登陆</h1> -->  <center>  <h3>用户注册</h3>  <form action=*"login.do"* method=*"post"*>  <input type=*"hidden"* name=*"action"*>  用户:<input type=*"text"* name=*"username"* size=*"10"*><br>  密码:<input type=*"password"* name=*"password"* size=*"10"*><br>  <input type=*"submit"* value=*"login"*>  <input type=*"reset"* value=*"重置"*/>    </form>  </center>  </body>  </html> |

说明：在页面代码中，提供了一个表单<form>,表单中提供了用户名和密码的输入框，以让用户数据认证信息，同时声明处理该submit事件的url为login.do。

### 2. 使用spring security拦截url并实现登录角色控制

#### A. 使用web.xml实现url拦截

表 2 部分web.xml代码

|  |
| --- |
| 添加spring security拦截器 |
| <filter>  <filter-name>springSecurityFilterChain</filter-name>  <filter-class>  org.springframework.web.filter.DelegatingFilterProxy  </filter-class>  </filter>  <filter-mapping>  <filter-name>springSecurityFilterChain</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |

说明：对该项目提供的所有服务都进行拦截

#### B.在spring context配置对应的bean以实现拦截控制，

表 3 部分spring-context.xml代码

|  |
| --- |
| 导入spring-security.xml的配置文件 |
| <import resource=*"spring-security.xml"* /> |

#### C. 配置spring-security.xml

表 4 spring-security.xml

|  |
| --- |
| 配置以实现用户登录控制和角色访问权限控制 |
| <security:http security=*"none"* pattern=*"/index.jsp"* />  <security:http security=*"none"* pattern=*"/login.jsp"* />  <security:http auto-config=*"true"*>  <security:form-login login-page=*"/login.jsp"*  login-processing-url=*"/login.do"*  username-parameter=*"username"*  password-parameter=*"password"* />  <security:logout logout-url=*"/logout"*/>  <security:intercept-url pattern=*"/users"* access=*"ROLE\_ADMIN"* />  <security:intercept-url pattern=*"/\*\*"* access=*"ROLE\_USER,ROLE\_ADMIN"* />  </security:http>      <security:authentication-manager alias=*"authenticationManager"*>  <security:authentication-provider>  <password-encoder ref=*"encoder"* />  <security:jdbc-user-service  data-source-ref=*"dataSource"*  users-by-username-query=*"select username, password, enabled from user where username = ?"*  authorities-by-username-query=*"select username, rolecode from user where username = ?"*/>  </security:authentication-provider>  </security:authentication-manager>    <beans:bean id=*"encoder"*  class=*"org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"*>  </beans:bean> |

说明：详细情况在实验代码说明

#### D. 实现用户信息详情的实现和控制

原系统用restful的风格，通过传入用户id，来查询用户详情，但是，这样容易导致其他用户通过暴力穷举的方式连接以查看其他用户详情。故，这里做了修改，通过spring-security提供的已认证信息的全局缓存来获取对应的用户资料，而不是通过传入用户id。

表 5 获取用户详细信息

|  |
| --- |
| 通过spring-security的登录者的全局缓存实现 |
| @RequestMapping(value = "/user", method = RequestMethod.***GET***)  **public** String getUser2(ModelMap model) {  Object principal = SecurityContextHolder.*getContext*().getAuthentication().getPrincipal();  String username = "";  **if** (principal **instanceof** UserDetails) {  username = ((UserDetails) principal).getUsername();  }  **try** {  User user = userMapper.getUserByUsername(username);  model.addAttribute("user", user);  **return** "user\_detail";  } **catch** (Exception e) {  // **TODO**: handle exception  **return** "error";  }  } |

说明：修改了url匹配模式，去除传入的路径参数id，而是从spring-security的缓存提供中获取用户名，再获取对应的用户详情。

#### E. 修改数据库中的密码存放形式，以支持原有的密码登录

生成原密码密文

表 6 生成密文密码

|  |
| --- |
| String pwd\_plaintext = "123456";    String encoded\_password = PasswordHelper.*getBCryptPassword*(pwd\_plaintext);    System.***out***.println(encoded\_password); |

更新数据库：

表 7 以mulan用户为例，（id = 5039）

|  |
| --- |
| update user set password='$2a$10$acJgZFUbkFELPvDeNVINIOcjT3aSXu6rJ.3ZWh3F jueKYoP2DRYvC' where id=5039; |

# 实验代码说明

spring-security.xml配置说明

表 8 spring-security.xml 具体配置说明

|  |  |
| --- | --- |
| 1 | <security:http security=*"none"* pattern=*"/index.jsp"* />  <security:http security=*"none"* pattern=*"/login.jsp"* /> |
|  | 对index.jsp 和 login.jsp 放行，不做安全登录控制，以让游客访问主页以及登录 |
|  |  |
| 2 | <security:form-login login-page=*"/login.jsp"*  login-processing-url=*"/login.do"*  username-parameter=*"username"*  password-parameter=*"password"* /> |
|  | 设置表单登录拦截器详情，设置表单用户名和密码的key，以及处理表单登录的url |
|  |  |
| 3 | <security:logout logout-url=*"/logout"*/> |
|  | 设置账户退出连接，以注销spring-security中的登录账户缓存 |
|  |  |
| 4 | <security:intercept-url pattern=*"/users"* access=*"ROLE\_ADMIN"* />  <security:intercept-url pattern=*"/\*\*"* access=*"ROLE\_USER,ROLE\_ADMIN"* /> |
|  | 设置访问资源角色权限，用户列表只能管理员才能访问，其他（只剩下用户详情）则可以由普通用户和管理员访问 |
|  |  |
| 5 | <security:authentication-manager alias=*"authenticationManager"*>  <security:authentication-provider>  <password-encoder ref=*"encoder"* />  <security:jdbc-user-service  data-source-ref=*"dataSource"*  users-by-username-query=*"select username, password, enabled from user where username = ?"*  authorities-by-username-query=*"select username, rolecode from user where username = ?"*/>  </security:authentication-provider>  </security:authentication-manager> |
|  | 设置角色控制账户的来源，通过查看数据库表，自行拟定对应的数据库查询语句，从而作为验证账户来源 |
|  |  |
| 6 | <beans:bean id=*"encoder"*  class=*"org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"*>  </beans:bean> |
|  | 对应5中的密码解析，使用bcrypt来进行密码的加密，从而实现以密文形式存储在数据库，但用户已明文形式能成功访问 |

# 实验结果与分析

**1.首页：**

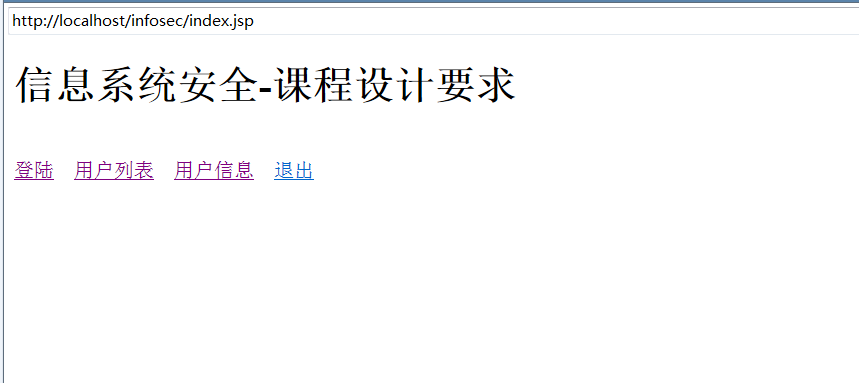


图 1 首页

**2.登录页：**

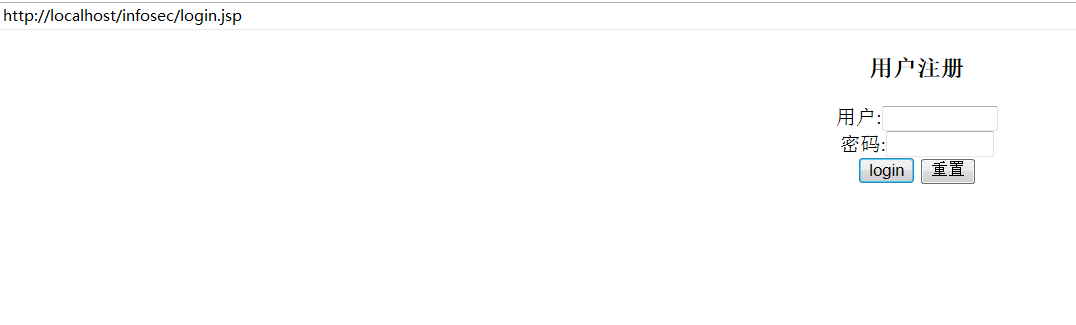


图 2 登录页

**3.其他页**

未登录访问需登录页面会自动跳转到登录页面：

图片同图2

**4.登录管理员成功后：**

**4.1用户详情页：**

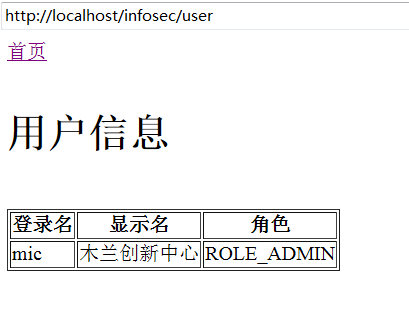


图 3 用户详情页

**4.2用户列表页：**



图 4 用户列表页

退出后不能查看，页面同登录页。

**5.普通用户登录后：**

**5.1用户详情页：**



图 5 用户信息页

**5.2尝试查看用户列表页：**

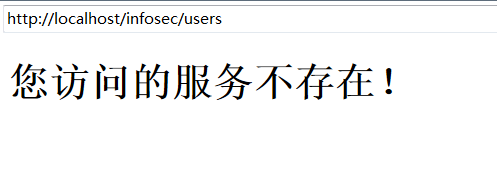


图 6 权限不够

**6. 退出后需再次登录。**

同登录页。

**结论：**

成功实现用户登录控制与角色访问控制。

# 参考

1. Spring Security Reference

<https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/>

1. Elim的博客 spring-security

<http://elim.iteye.com/category/182468/f>