

学习目标

- RBAC 模型简介
- 数据库权限表结构设计与创建
- 搭建 Spring Security + SSM 运行环境
- 实现用户查询与权限查询持久层方法
- 自定义 UserDetailsService 实现动态数据权限访问
- PasswordEncoder 密码加密
- 自定义图形验证码

1. RBAC 模型简介

[+](#) [★ 收藏](#) | [👍 534](#) | [🔗 37](#)

RBAC 编辑

基于角色的权限访问控制（Role-Based Access Control）作为传统访问控制（自主访问，强制访问）的有前景的代替受到广泛的关注。在RBAC中，权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限。这就极大地简化了权限的管理。在一个组织中，角色是为了完成各种工作而创造，用户则依据它的责任和资格来被指派相应的角色，用户可以很容易地从一个角色被指派到另一个角色。角色可依新的需求和系统的合并而赋予新的权限，而权限也可根据需要而从某角色中回收。角色与角色的关系可以建立起来以囊括更广泛的客观情况。

2. 数据库权限表结构设计与创建

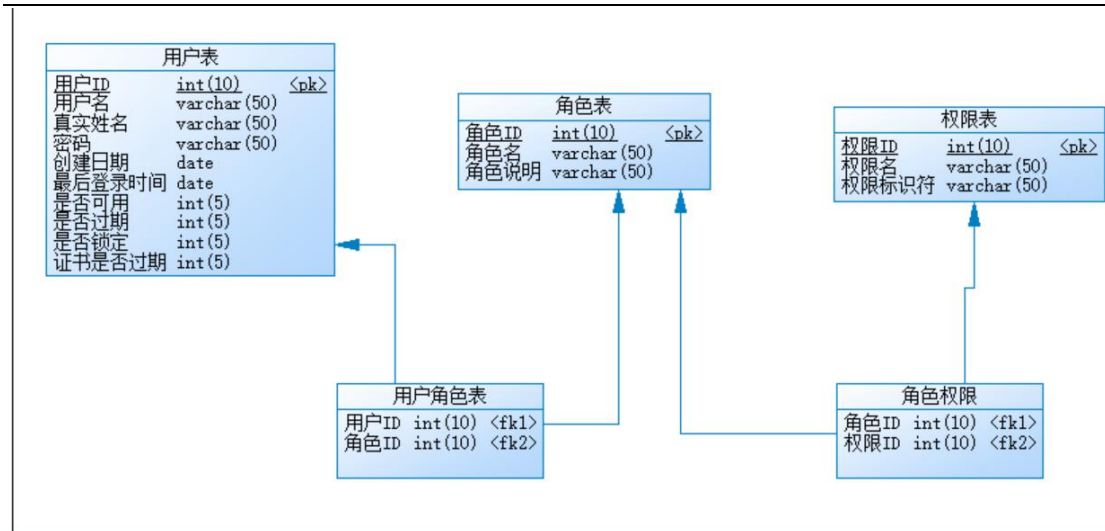
基于 RBAC 权限模型，设计权限表相关表：

- 1) 用户
- 2) 角色
- 3) 权限

用户 和 角色 多对多关系。

角色 和 权限 多对多关系。

使用 PowerDesigner 设计权限表。



3. 搭建 Spring Security + SSM 运行环境

3.1. 建立 maven 项目，配置 pom

```
<?xml version="1.0" encoding="UTF-8" ?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <parent>
    <artifactId>01.spring-security</artifactId>
    <groupId>cn.sm1234</groupId>
    <version>1.0-SNAPSHOT</version>
    <relativePath>../01.springsecurity/pom.xml</relativePath>
  </parent>

  <modelVersion>4.0.0</modelVersion>

  <artifactId>02.springsecurity</artifactId>

  <packaging>war</packaging>
```

```
<properties>

  <jdk.version>1.9</jdk.version>

  <spring.version>4.3.10.RELEASE</spring.version>

  <spring.security.version>4.2.3.RELEASE</spring.security.version>

  <jstl.version>1.2</jstl.version>

  <servlet.version>2.5</servlet.version>

</properties>

<dependencies>

  <!-- Spring dependencies -->

  <dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-core</artifactId>

    <version>${spring.version}</version>

  </dependency>

  <dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-web</artifactId>

    <version>${spring.version}</version>

  </dependency>

  <dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-webmvc</artifactId>

    <version>${spring.version}</version>
```

```
</dependency>

<!-- Spring Security -->

<dependency>

    <groupId>org.springframework.security</groupId>

    <artifactId>spring-security-web</artifactId>

    <version>${spring.security.version}</version>

</dependency>

<dependency>

    <groupId>org.springframework.security</groupId>

    <artifactId>spring-security-config</artifactId>

    <version>${spring.security.version}</version>

</dependency>

<!-- jstl for jsp page -->

<dependency>

    <groupId>jstl</groupId>

    <artifactId>jstl</artifactId>

    <version>${jstl.version}</version>

</dependency>

<dependency>

    <groupId>javax.servlet</groupId>

    <artifactId>servlet-api</artifactId>

    <version>${servlet.version}</version>

    <scope>provided</scope>

</dependency>

<dependency>
```

```
<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.5</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.4</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.0</version>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.1.7</version>

</dependency>
```

```
<dependency>

  <groupId>mysql</groupId>

  <artifactId>mysql-connector-java</artifactId>

  <version>5.1.41</version>

</dependency>

</dependencies>

<build>

  <plugins>

    <!-- jdk 版本插件 -->

    <plugin>

      <groupId>org.apache.maven.plugins</groupId>

      <artifactId>maven-compiler-plugin</artifactId>

      <version>3.2</version>

      <configuration>

        <source>1.9</source>

        <target>1.9</target>

        <encoding>UTF-8</encoding>

        <showWarnings>>true</showWarnings>

      </configuration>

    </plugin>

    <!-- tomcat7 插件 -->

    <plugin>

      <groupId>org.apache.tomcat.maven</groupId>

      <artifactId>tomcat7-maven-plugin</artifactId>

      <version>2.1</version>
```

```
<configuration>
    <port>8080</port>
    <path>/ss1</path>
    <server>tomcat7</server>
</configuration>
</plugin>
</plugins>
</build>
</project>
```

3.2. 配置 web.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    version="2.5">
    <!-- SpringSecurity 过滤器链 -->
    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-c
lass>
    </filter>
    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
```

```
</filter-mapping>

<!-- 启动 Spring -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath:applicationContext.xml

classpath:spring-security.xml

</param-value>

</context-param>

<!-- 启动 SpringMVC -->

<servlet>

<servlet-name>DispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:springmvc.xml</param-value>

</init-param>

<!-- 服务器启动加载 Servlet -->

<load-on-startup>1</load-on-startup>
```



```
</servlet>

<servlet-mapping>

  <servlet-name>DispatcherServlet</servlet-name>

  <url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>
```

3.3. 配置 SpringSecurity 和 SSM 文件

3.3.1. applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd">

</beans>
```

3.3.2. spring-security.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:security="http://www.springframework.org/schema/security"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
                           http://www.springframework.org/schema/security
                           http://www.springframework.org/schema/security/spring-security-4.2.xsd">

    <security:http>

        <security:form-login/>

    </security:http>

    <security:authentication-manager>

    </security:authentication-manager>

</beans>
```

3.3.3. springmvc.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
```

```
xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

    http://www.springframework.org/schema/beans

    http://www.springframework.org/schema/beans/spring-beans.xsd

    http://www.springframework.org/schema/mvc

    http://www.springframework.org/schema/mvc/spring-mvc.xsd

    http://www.springframework.org/schema/context

    http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- 扫描 Controller 类-->

    <context:component-scan base-package="cn.sm1234"/>

    <!-- 注解方式处理器映射器和处理器适配器 -->

    <mvc:annotation-driven></mvc:annotation-driven>

    <!-- 视图解析器-->

    <bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

        <!-- 前缀 -->

        <property name="prefix" value="/WEB-INF/jsp/" />

        <!-- 后缀-->

        <property name="suffix" value=".jsp" />

    </bean>

</beans>
```

3.4. MyBatis 整合 Spring

3.4.1. jdbc.properties

```
jdbc.url = jdbc:mysql://localhost:3306/security
jdbc.driverClass = com.mysql.jdbc.Driver
jdbc.username = root
jdbc.password = root
```

3.4.2. applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.alibaba.com/schema/stat"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
http://www.alibaba.com/schema/stat http://www.alibaba.com/schema/stat.xsd">
    <!-- 读取 jdbc.properties -->
    <context:property-placeholder location="classpath:jdbc.properties"/>
```

```
<!-- 连接池 -->

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">

    <property name="url" value="${jdbc.url}"/>

    <property name="driverClassName" value="${jdbc.driverClass}"/>

    <property name="username" value="${jdbc.username}"/>

    <property name="password" value="${jdbc.password}"/>

    <property name="maxActive" value="10"/>

    <property name="maxWait" value="3000"/>

</bean>

<!-- mybatis 整合 Spring -->

<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">

    <property name="dataSource" ref="dataSource"/>

    <!-- 别名扫描 -->

    <property name="typeAliasesPackage" value="cn.sm1234.domain"/>

</bean>

<!-- Mapper 接口扫描 -->

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

    <property name="basePackage" value="cn.sm1234.mapper"/>

</bean>

<!-- 事务配置 -->

<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

    <property name="dataSource" ref="dataSource"/>

</bean>

<tx:annotation-driven/>
```

```
<context:component-scan base-package="cn.sm1234.service"/>

</beans>
```

4. 实现用户查询与权限查询持久层方法

4.1. 创建实体类

4.1.1. User

```
package cn.sm1234.domain;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * @author http://www.sm1234.cn
 * @version 1.0
 * @description cn.sm1234.domain
 * @date 18/4/14
 */
public class User implements UserDetails{
```

```
private Integer id; //int(10) NOT NULL,

private String username; //varchar(50) DEFAULT NULL,

private String realname; //varchar(50) DEFAULT NULL,

private String password; //varchar(50) DEFAULT NULL,

private Date createDate; //date DEFAULT NULL,

private Date lastLoginTime; //date DEFAULT NULL,

private boolean enabled; //int(5) DEFAULT NULL,

private boolean accountNonExpired; //int(5) DEFAULT NULL,

private boolean accountNonLocked; //int(5) DEFAULT NULL,

private boolean credentialsNonExpired; //int(5) DEFAULT NULL,

//用户拥有的所有权限

private List<GrantedAuthority> authorities = new
ArrayList<GrantedAuthority>();

public List<GrantedAuthority> getAuthorities() {

    return authorities;

}

public void setAuthorities(List<GrantedAuthority> authorities) {

    this.authorities = authorities;

}

public Integer getId() {

    return id;

}

public void setId(Integer id) {

    this.id = id;
```

```
}

@Override

public String getUsername() {

    return username;

}

public void setUsername(String username) {

    this.username = username;

}

public String getRealname() {

    return realname;

}

public void setRealname(String realname) {

    this.realname = realname;

}

@Override

public String getPassword() {

    return password;

}

public void setPassword(String password) {

    this.password = password;

}

public Date getCreateDate() {
```



```
        return createDate;
    }

    public void setCreateDate(Date createDate) {
        this.createDate = createDate;
    }

    public Date getLastLoginTime() {
        return lastLoginTime;
    }

    public void setLastLoginTime(Date lastLoginTime) {
        this.lastLoginTime = lastLoginTime;
    }

    @Override
    public boolean isEnabled() {
        return enabled;
    }

    public void setEnabled(boolean enabled) {
        this.enabled = enabled;
    }

    @Override
    public boolean isAccountNonExpired() {
        return accountNonExpired;
    }
}
```

```
public void setAccountNonExpired(boolean accountNonExpired) {  
  
    this.accountNonExpired = accountNonExpired;  
  
}  
  
@Override  
  
public boolean isAccountNonLocked() {  
  
    return accountNonLocked;  
  
}  
  
public void setAccountNonLocked(boolean accountNonLocked) {  
  
    this.accountNonLocked = accountNonLocked;  
  
}  
  
@Override  
  
public boolean isCredentialsNonExpired() {  
  
    return credentialsNonExpired;  
  
}  
  
public void setCredentialsNonExpired(boolean credentialsNonExpired) {  
  
    this.credentialsNonExpired = credentialsNonExpired;  
  
}  
  
}
```

4.1.2. Role

```
package cn.sm1234.domain;
```

```
/**
```

```
* @author http://www.sm1234.cn
* @version 1.0
* @description cn.sm1234.domain
* @date 18/4/14
*/

public class Role {

    private Integer id; //int(10) NOT NULL,

    private String roleName; //varchar(50) DEFAULT NULL,

    private String roleDesc; //varchar(50) DEFAULT NULL,

    public Integer getId() {

        return id;

    }

    public void setId(Integer id) {

        this.id = id;

    }

    public String getRoleName() {

        return roleName;

    }

    public void setRoleName(String roleName) {

        this.roleName = roleName;

    }

    public String getRoleDesc() {

        return roleDesc;

    }

}
```

```
public void setRoleDesc(String roleDesc) {  
    this.roleDesc = roleDesc;  
}  
}
```

4.1.3. Permission

```
package cn.sm1234.domain;  
  
/**  
 * @author http://www.sm1234.cn  
 * @version 1.0  
 * @description cn.sm1234.domain  
 * @date 18/4/14  
 */  
  
public class Permission {  
    private Integer id; //int(10) NOT NULL,  
    private String permName; //varchar(50) DEFAULT NULL,  
    private String permTag; //varchar(50) DEFAULT NULL,  
  
    public Integer getId() {  
        return id;  
    }  
  
    public void setId(Integer id) {  
        this.id = id;  
    }  
}
```

```
public String getPermName() {  
  
    return permName;  
  
}  
  
public void setPermName(String permName) {  
  
    this.permName = permName;  
  
}  
  
public String getPermTag() {  
  
    return permTag;  
  
}  
  
public void setPermTag(String permTag) {  
  
    this.permTag = permTag;  
  
}  
  
}
```

4.2. 编写持久层接口

```
package cn.sm1234.mapper;  
  
import cn.sm1234.domain.Permission;  
import cn.sm1234.domain.User;  
  
import java.util.List;  
  
/**  
 * @author http://www.sm1234.cn
```

```
* @version 1.0
* @description cn.sm1234.mapper
* @date 18/4/14
*/

public interface UserMapper {

    /**
     * 查询当前用户对象
     */

    public User findByUsername(String username);

    /**
     * 查询当前用户拥有的权限
     */

    public List<Permission> findPermissionByUsername(String username);
}
}
```

4.3. sql 映射文件

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="cn.sm1234.mapper.UserMapper">

    <!-- 查询用户 -->

    <select id="findByUsername" parameterType="string" resultType="user">

        select * from sys_user where username = #{value}

    </select>
}
```

```
<!-- 查询用户的权限 -->

<select id="findPermissionByUsername" parameterType="string"
resultType="permission">

    select permission.*

    from

        sys_user user

        inner join sys_user_role user_role on user.id = user_role.user_id

        inner join sys_role_permission role_permission on user_role.role_id
= role_permission.role_id

        inner join sys_permission permission on role_permission.perm_id =
permission.id

    where user.username = #{value};

</select>

</mapper>
```

4.4. 编写持久层测试类

```
import cn.sm1234.domain.Permission;

import cn.sm1234.domain.User;

import cn.sm1234.mapper.UserMapper;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.test.context.ContextConfiguration;
```

```
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import java.util.List;

/**
 * @author http://www.sm1234.cn
 * @version 1.0
 * @description PACKAGE_NAME
 * @date 18/4/14
 */

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext.xml")
public class UserMapperTest {

    @Autowired

    private UserMapper userMapper;

    @Test

    public void testFindByUsername() {

        User user = userMapper.findByUsername("eric");

        System.out.println(user);

    }

    @Test

    public void testFindPermissionByUsername() {

        List<Permission> list= userMapper.findPermissionByUsername("jack");

        for (Permission perm:list) {

            System.out.println(perm.getPermName()+"-"+perm.getPermTag());

        }

    }

}
```



```
}  
  
}
```

5. 自定义 UserDetailsService 实现动态数据权限访问

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<beans xmlns="http://www.springframework.org/schema/beans"  
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
       xmlns:security="http://www.springframework.org/schema/security"  
       xsi:schemaLocation="http://www.springframework.org/schema/beans  
                           http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  
                           http://www.springframework.org/schema/security  
                           http://www.springframework.org/schema/security/spring-security-4.2.xsd">  
  
    <security:http>  
        <!-- 拦截资源 -->  
        <security:intercept-url pattern="/product/list"  
                                access="hasAuthority('ROLE_LIST_PRODUCT')"/>  
        <security:intercept-url pattern="/product/add"  
                                access="hasAuthority('ROLE_ADD_PRODUCT')"/>  
        <security:intercept-url pattern="/product/update"  
                                access="hasAuthority('ROLE_UPDATE_PRODUCT')"/>  
        <security:intercept-url pattern="/product/delete"  
                                access="hasAuthority('ROLE_DELETE_PRODUCT')"/>  
    </security:http>  
</beans>
```

```
<security:intercept-url pattern="/userLogin" access="permitAll()" />

<security:intercept-url pattern="/product/index" access="permitAll()" />

<security:intercept-url pattern="/**" access="isFullyAuthenticated()" />

<security:form-login login-page="/userLogin" />

<!-- 权限不足处理 -->

<security:access-denied-handler error-page="/error" />

<security:csrf disabled="true" />

</security:http>

<security:authentication-manager>

  <security:authentication-provider

    user-service-ref="myUserDetailService" />

  </security:authentication-provider>

</security:authentication-manager>

<bean id="myUserDetailService"

  class="cn.sm1234.security.MyUserDetailService" />

</beans>
```

```
package cn.sm1234.security;

import cn.sm1234.domain.Permission;

import cn.sm1234.domain.User;

import cn.sm1234.mapper.UserMapper;

import org.apache.log4j.Logger;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.GrantedAuthority;
```

```
import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import

org.springframework.security.core.userdetails.UsernameNotFoundException;

import java.util.ArrayList;

import java.util.List;

/**
 * @author http://www.sm1234.cn
 * @version 1.0
 * @description cn.sm1234.security
 * @date 18/4/14
 */

public class MyUserDetailService implements UserDetailsService {

    private Logger logger = Logger.getLogger(MyUserDetailService.class);

    @Autowired

    private UserMapper userMapper;

    @Override

    public UserDetails loadUserByUsername(String username) throws

UsernameNotFoundException {

        //根据用户名查询用户信息

        User user = userMapper.findByUsername(username);

        //根据用户名查询当前用户所有权限
```

```
        List<Permission> permList =
userMapper.findPermissionByUsername(username);

        //authorities: 存放所有用户权限

        List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();

        for(Permission perm:permList){

            GrantedAuthority authority = new
SimpleGrantedAuthority(perm.getPermTag());

            authorities.add( authority );

        }

        //把所有权限赋值给 user

        user.setAuthorities(authorities);

        logger.info("当前用户: "+user);

        return user;

    }
}
```

6. 登录成功与登录失败处理

6.1. 同步方式处理

```
<security:form-login login-page="/userLogin"
authentication-failure-url="/userLogin?error=true"
authentication-success-forward-url="/product/index"/>
```

6.2. 异步方式处理

```
package cn.sm1234.security;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.security.core.Authentication;
import
org.springframework.security.web.authentication.AuthenticationSuccessHandler
;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

/**
 * @author http://www.sm1234.cn
 * @version 1.0
 * @description cn.sm1234.security
 * @date 18/4/15
 */
public class MyAuthenticationSuccessHandler implements
AuthenticationSuccessHandler {

    private ObjectMapper objectMapper = new ObjectMapper();

    @Override
```

```
public void onAuthenticationSuccess(HttpServletRequest request,
HttpServletResponse response, Authentication authentication) throws
IOException, ServletException {

    //返回 json 数据

    Map result = new HashMap();

    result.put("success", true);

    String json = objectMapper.writeValueAsString(result);

    response.setContentType("text/json;charset=utf-8");

    response.getWriter().write(json);

}
}
```

```
package cn.sm1234.security;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.springframework.security.core.AuthenticationException;

import
org.springframework.security.web.authentication.AuthenticationFailureHandler
;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;
```

```
/**
 * @author http://www.sm1234.cn
 * @version 1.0
 * @description cn.sm1234.security
 * @date 18/4/15
 */

public class MyAuthenticationFailureHandler implements
AuthenticationFailureHandler {

    private ObjectMapper objectMapper = new ObjectMapper();

    @Override
    public void onAuthenticationFailure(HttpServletRequest request,
    HttpServletResponse response, AuthenticationException exception) throws
IOException, ServletException {

        //返回 json 数据

        Map result = new HashMap();

        result.put("success", false);

        String json = objectMapper.writeValueAsString(result);

        response.setContentType("text/json;charset=utf-8");

        response.getWriter().write(json);

    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xmlns:security="http://www.springframework.org/schema/security"
```

```
    xsi:schemaLocation="http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
                        http://www.springframework.org/schema/security
                        http://www.springframework.org/schema/security/spring-security-4.2.xsd">

    <security:http>

        <!-- 拦截资源 -->

        <security:intercept-url pattern="/product/list"
        access="hasAuthority('ROLE_LIST_PRODUCT')"/>

        <security:intercept-url pattern="/product/add"
        access="hasAuthority('ROLE_ADD_PRODUCT')"/>

        <security:intercept-url pattern="/product/update"
        access="hasAuthority('ROLE_UPDATE_PRODUCT')"/>

        <security:intercept-url pattern="/product/delete"
        access="hasAuthority('ROLE_DELETE_PRODUCT')"/>

        <security:intercept-url pattern="/userLogin" access="permitAll()"/>

        <security:intercept-url pattern="/js/**" access="permitAll()"/>

        <security:intercept-url pattern="/product/index" access="permitAll()"/>

        <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

        <security:form-login login-page="/userLogin"
        authentication-success-handler-ref="myAuthenticationSuccessHandler"
        authentication-failure-handler-ref="myAuthenticationFailureHandler"/>

        <!-- 权限不足处理 -->

        <security:access-denied-handler error-page="/error"/>

        <security:csrf disabled="true"/>

    </security:http>
```



```
<security:authentication-manager>
  <security:authentication-provider
user-service-ref="myUserDetailService"/>
</security:authentication-manager>

<bean id="myUserDetailService"
class="cn.sm1234.security.MyUserDetailsService"/>

<bean id="myAuthenticationSuccessHandler"
class="cn.sm1234.security.MyAuthenticationSuccessHandler"/>
<bean id="myAuthenticationFailureHandler"
class="cn.sm1234.security.MyAuthenticationFailureHandler"/>

</beans>
```

7. PasswordEncoder 密码加密

关键：PasswordEncoder 接口的实现类

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:security="http://www.springframework.org/schema/security"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
http://www.springframework.org/schema/security
```

```
http://www.springframework.org/schema/security/spring-security-4.2.xsd">

<security:http>

  <!-- 拦截资源 -->

  <security:intercept-url pattern="/product/list"
access="hasAuthority('ROLE_LIST_PRODUCT')"/>

  <security:intercept-url pattern="/product/add"
access="hasAuthority('ROLE_ADD_PRODUCT')"/>

  <security:intercept-url pattern="/product/update"
access="hasAuthority('ROLE_UPDATE_PRODUCT')"/>

  <security:intercept-url pattern="/product/delete"
access="hasAuthority('ROLE_DELETE_PRODUCT')"/>

  <security:intercept-url pattern="/userLogin" access="permitAll()"/>

  <security:intercept-url pattern="/js/**" access="permitAll()"/>

  <security:intercept-url pattern="/product/index" access="permitAll()"/>

  <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

  <security:form-login login-page="/userLogin"
authentication-success-handler-ref="myAuthenticationSuccessHandler"
authentication-failure-handler-ref="myAuthenticationFailureHandler"/>

  <!-- 权限不足处理 -->

  <security:access-denied-handler error-page="/error"/>

  <security:csrf disabled="true"/>

</security:http>

<security:authentication-manager>

  <security:authentication-provider
```

```
user-service-ref="myUserDetailService">

    <!--使用加密算法对用户输入的密码进行加密，然后和数据库的密码进行配对 -->
    <security:password-encoder ref="passwordEncoder"/>

</security:authentication-provider>

</security:authentication-manager>

<bean id="myUserDetailService"
class="cn.sm1234.security.MyUserDetailsService"/>

<bean id="myAuthenticationSuccessHandler"
class="cn.sm1234.security.MyAuthenticationSuccessHandler"/>

<bean id="myAuthenticationFailureHandler"
class="cn.sm1234.security.MyAuthenticationFailureHandler"/>

<bean id="passwordEncoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>

</beans>
```

8. 自定义图形验证码

8.1. 制作一个图形验证码

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>

<%@ page import="java.util.Random"%>

<%@ page import="java.io.OutputStream"%>

<%@ page import="java.awt.Color"%>
```

```
<%@ page import="java.awt.Font"%>
<%@ page import="java.awt.Graphics"%>
<%@ page import="java.awt.image.BufferedImage"%>
<%@ page import="javax.imageio.ImageIO"%>
<%
    int width = 80;
    int height = 32;
    //create the image
    BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();
    // set the background color
    g.setColor(new Color(0xDCDCDC));
    g.fillRect(0, 0, width, height);
    // draw the border
    g.setColor(Color.black);
    g.drawRect(0, 0, width - 1, height - 1);
    // create a random instance to generate the codes
    Random rdm = new Random();
    String hash1 = Integer.toHexString(rdm.nextInt());
    // make some confusion
    for (int i = 0; i < 50; i++) {
        int x = rdm.nextInt(width);
        int y = rdm.nextInt(height);
        g.drawOval(x, y, 0, 0);
    }
    // generate a random code
    String capstr = hash1.substring(0, 4);
    session.setAttribute("key", capstr);
%>
```

```
g.setColor(new Color(0, 100, 0));

g.setFont(new Font("Candara", Font.BOLD, 24));

g.drawString(capstr, 8, 24);

g.dispose();

response.setContentType("image/jpeg");

out.clear();

out = pageContext.pushBody();

OutputStream strm = response.getOutputStream();

ImageIO.write(image, "jpeg", strm);

strm.close();

%>
```

```
/**
 * 生成验证码
 */
@RequestMapping("/imageCode")
public String imageCode(){
    return "imageCode";
}
```

```
<security:intercept-url pattern="/imageCode*" access="permitAll()"/>
```

8.2. 在登录页面使用图形验证码

```
<form method="post" id="loginForm">
```

```

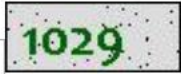
用户名:<input type="text" name="username"/><br/>
密码:<input type="password" name="password"/><br/>
验证码:<input type="text" name="imageCode"/><br/>
<input type="button" id="loginBtn" value="登录"/>
</form>

```

登录页面

用户名:

密码:

验证码: 

8.3. 自定义验证码校验过滤器

```

package cn.sm1234.security;

import org.springframework.security.core.AuthenticationException;

import
org.springframework.security.web.authentication.AuthenticationFailureHandler
;

import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;

```

```
import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

/**
 * @author http://www.sm1234.cn
 * @version 1.0
 * @description cn.sm1234.security
 * @date 18/4/15
 */

public class ImageCodeAuthenticationFilter extends OncePerRequestFilter {

    private AuthenticationFailureHandler authenticationFailureHandler;

    public void setAuthenticationFailureHandler(AuthenticationFailureHandler authenticationFailureHandler) {

        this.authenticationFailureHandler = authenticationFailureHandler;

    }

    @Override

    protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response, FilterChain filterChain) throws ServletException,

IOException {

        //判断当前请求 是否为登录请求

        if( request.getRequestURI().contains("/login") ){

            //校验验证码
```

```
try {  
  
    //获取用户输入的验证码  
  
    final String imageCode = request.getParameter("imageCode");  
  
    //获取系统生成的验证码  
  
    String key = (String)request.getSession().getAttribute("key");  
  
    if(StringUtils.isEmpty(imageCode.trim())){  
  
        throw new ImageCodeException("验证码必须输入");  
  
    }  
  
    if(!imageCode.trim().equals(key.trim())){  
  
        throw new ImageCodeException("验证码不一致");  
  
    }  
  
}catch (AuthenticationException e){  
  
    //交给自定义 AuthenFailureHandler 处理  
  
authenticationFailureHandler.onAuthenticationFailure(request, response, e);  
  
    return;  
  
}  
  
}  
  
filterChain.doFilter(request, response);  
  
}  
}
```

注意：修改 MyAuthenticationFailureHandler：

```
public class MyAuthenticationFailureHandler implements
```



```
AuthenticationFailureHandler {  
  
    private ObjectMapper objectMapper = new ObjectMapper();  
  
    @Override  
  
    public void onAuthenticationFailure(HttpServletRequest request,  
    HttpServletResponse response, AuthenticationException exception) throws  
    IOException, ServletException {  
  
        //返回 json 数据  
  
        Map result = new HashMap();  
  
        result.put("success", false);  
  
        //错误信息  
        result.put("errorMsg", exception.getMessage());  
  
  
        String json = objectMapper.writeValueAsString(result);  
  
        response.setContentType("text/json;charset=utf-8");  
  
        response.getWriter().write(json);  
  
    }  
}
```

8.4. 配置 spring-security.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<beans xmlns="http://www.springframework.org/schema/beans"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xmlns:security="http://www.springframework.org/schema/security"  
    xsi:schemaLocation="http://www.springframework.org/schema/beans  
        http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  
        http://www.springframework.org/schema/security
```

```
http://www.springframework.org/schema/security/spring-security-4.2.xsd">

<security:http>

  <!-- 拦截资源 -->

  <security:intercept-url pattern="/product/list"
access="hasAuthority('ROLE_LIST_PRODUCT')"/>

  <security:intercept-url pattern="/product/add"
access="hasAuthority('ROLE_ADD_PRODUCT')"/>

  <security:intercept-url pattern="/product/update"
access="hasAuthority('ROLE_UPDATE_PRODUCT')"/>

  <security:intercept-url pattern="/product/delete"
access="hasAuthority('ROLE_DELETE_PRODUCT')"/>

  <security:intercept-url pattern="/userLogin" access="permitAll()"/>

  <security:intercept-url pattern="/js/**" access="permitAll()"/>

  <security:intercept-url pattern="/imageCode*" access="permitAll()"/>

  <security:intercept-url pattern="/product/index" access="permitAll()"/>

  <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

  <!-- 自定义 Spring Security 过滤器 -->

  <security:custom-filter ref="imageCodeAuthenticationFilter"
before="FORM_LOGIN_FILTER"/>

  <security:form-login login-page="/userLogin"
authentication-success-handler-ref="myAuthenticationSuccessHandler"
authentication-failure-handler-ref="myAuthenticationFailureHandler"/>

  <!-- 权限不足处理 -->

  <security:access-denied-handler error-page="/error"/>
```

```
<security:csrf disabled="true"/>
</security:http>

<security:authentication-manager>
  <security:authentication-provider
    user-service-ref="myUserDetailService">
    <!--使用加密算法对用户输入的密码进行加密，然后和数据库的密码进行配对 -->
    <security:password-encoder ref="passwordEncoder"/>
  </security:authentication-provider>
</security:authentication-manager>

<bean id="myUserDetailService"
class="cn.sm1234.security.MyUserDetailsService"/>

<bean id="myAuthenticationSuccessHandler"
class="cn.sm1234.security.MyAuthenticationSuccessHandler"/>

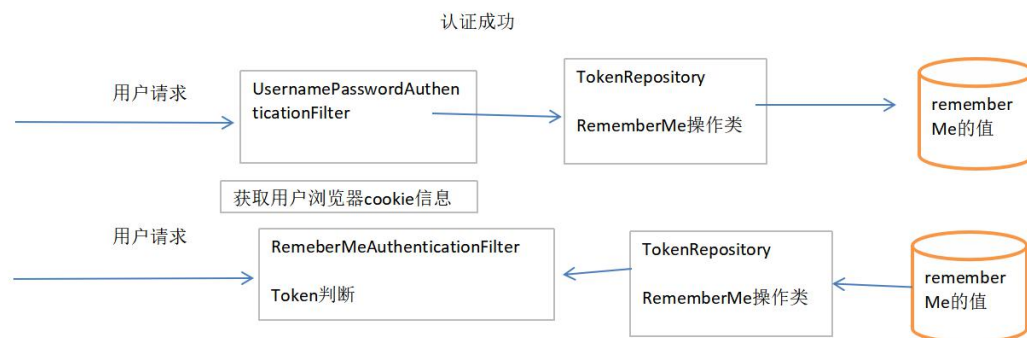
<bean id="myAuthenticationFailureHandler"
class="cn.sm1234.security.MyAuthenticationFailureHandler"/>

<bean id="passwordEncoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>

<bean id="imageCodeAuthenticationFilter"
class="cn.sm1234.security.ImageCodeAuthenticationFilter">
  <property name="authenticationFailureHandler"
ref="myAuthenticationFailureHandler"/>
</bean>
</beans>
```

9. rememberMe 记住我

执行流程：



9.1. 在登录页面添加 remember-me

```

<form method="post" id="loginForm">
    用户名:<input type="text" name="username"/><br/>
    密码:<input type="password" name="password"/><br/>
    验证码:<input type="text" name="imageCode"/><br/>
    记住我:<input type="checkbox" name="remember-me" value="true"/><br/>
    <input type="button" id="loginBtn" value="登录"/>
</form>
  
```

9.2. 配置 spring-security.xml

```

<?xml version="1.0" encoding="UTF-8"?>
  
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:security="http://www.springframework.org/schema/security"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
                           http://www.springframework.org/schema/security
                           http://www.springframework.org/schema/security/spring-security-4.2.xsd">

    <security:http>

        <!-- 拦截资源 -->

        <security:intercept-url pattern="/product/list"
                                access="hasAuthority('ROLE_LIST_PRODUCT')"/>

        <security:intercept-url pattern="/product/add"
                                access="hasAuthority('ROLE_ADD_PRODUCT')"/>

        <security:intercept-url pattern="/product/update"
                                access="hasAuthority('ROLE_UPDATE_PRODUCT')"/>

        <security:intercept-url pattern="/product/delete"
                                access="hasAuthority('ROLE_DELETE_PRODUCT')"/>

        <security:intercept-url pattern="/userLogin" access="permitAll()"/>

        <security:intercept-url pattern="/js/**" access="permitAll()"/>

        <security:intercept-url pattern="/imageCode*" access="permitAll()"/>

        <security:intercept-url pattern="/product/index" access="permitAll()"/>

        <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

        <!-- 自定义 Spring Security 过滤器 -->

        <security:custom-filter ref="imageCodeAuthenticationFilter"
                                before="FORM_LOGIN_FILTER"/>
    </security:http>
</beans>
```

```
<security:form-login login-page="/userLogin"
authentication-success-handler-ref="myAuthenticationSuccessHandler"
authentication-failure-handler-ref="myAuthenticationFailureHandler"/>

<!-- 权限不足处理 -->

<security:access-denied-handler error-page="/error"/>

<security:csrf disabled="true"/>

<!-- 加上 rememberMe 功能 -->

<!-- token-validity-seconds: 有效秒数 -->

<security:remember-me token-repository-ref="jdbcTokenRepository"
token-validity-seconds="3600"/>

</security:http>

<security:authentication-manager>

<security:authentication-provider
user-service-ref="myUserDetailService">

<!--使用加密算法对用户输入的密码进行加密，然后和数据库的密码进行配对 -->

<security:password-encoder ref="passwordEncoder"/>

</security:authentication-provider>

</security:authentication-manager>

<bean id="myUserDetailService"
class="cn.sm1234.security.MyUserDetailsService"/>

<bean id="myAuthenticationSuccessHandler"
class="cn.sm1234.security.MyAuthenticationSuccessHandler"/>

<bean id="myAuthenticationFailureHandler"
```

```
class="cn.sm1234.security.MyAuthenticationFailureHandler"/>

    <bean id="passwordEncoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>

    <bean id="imageCodeAuthenticationFilter"
class="cn.sm1234.security.ImageCodeAuthenticationFilter">
        <property name="authenticationFailureHandler"
ref="myAuthenticationFailureHandler"/>
    </bean>

    <bean id="jdbcTokenRepository"
class="org.springframework.security.web.authentication.rememberme.JdbcTokenR
epositoryImpl">
        <property name="dataSource" ref="dataSource"/>
        <property name="createTableOnStartup" value="true"/>
    </bean>
</beans>
```

10. Spring Security 权限标签使用

10.1. 导入标签库的坐标

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-taglibs</artifactId>
    <version>${spring.security.version}</version>
</dependency>
```

10.2. 在 JSP 页面导入标签库

```
<%@ taglib
uri="http://www.springframework.org/security/tags"
prefix="security" %>
```

10.3. 使用 Security 标签

```
<security:authorize access="hasAuthority('ROLE_ADD_PRODUCT')">
    <a href="${pageContext.request.contextPath}/product/add">商品添加</a><br/>
</security:authorize>

<security:authorize access="hasAuthority('ROLE_UPDATE_PRODUCT')">
    <a href="${pageContext.request.contextPath}/product/update">商品修改
</a><br/>
</security:authorize>

<security:authorize access="hasAuthority('ROLE_LIST_PRODUCT')">
    <a href="${pageContext.request.contextPath}/product/list">商品查询</a><br/>
</security:authorize>

<security:authorize access="hasAuthority('ROLE_DELETE_PRODUCT')">
    <a href="${pageContext.request.contextPath}/product/delete">商品删除
</a><br/>
</security:authorize>
```


11. 如何获取登录后用户名

关键点: SecurityContextHolder 接口, 用于操作认证信息。

```
/**
 * 商品主页
 */
@RequestMapping("/index")
public String index(Model model){

    //获取登录后用户: UserDetails 对象
    Object principal =
SecurityContextHolder.getContext().getAuthentication().getPrincipal();

    if(principal!=null){

        if(principal instanceof UserDetails){

            UserDetails userDetails = (UserDetails)principal;

            String username = userDetails.getUsername();

            model.addAttribute("username",username);

        }

    }

    return "index";
}
```