一、SpringSecurity 框架简介

官网：https://projects.spring.io/spring-security/ SpringSecurity
是强大的，且容易定制的实现认证，与授权的基于 Spring 开发的框架。

SpringSecurity 的功能：

- Authentication：认证，就是用户登录。
- Authorization：授权，判断用户拥有什么权限，可以访问什么资源。
- 安全防护，防止跨站请求，session 攻击等
- 非常容易结合 SpringMVC 进行使用

二、SpringSecurity 与 Shiro 的区别

1. 相同点

认证功能
授权功能
加密功能
会话管理
缓存支持
rememberMe 功能
....

2. 不同点

① 优点：

- Spring Security 基于 Spring 开发，项目如使用 Spring 作为基础，配合 Spring Security 做权限更加方便。而 Shiro 需要和 Spring 进行整合开发。

- SpringSecurity 功能比 Shiro 更加丰富些，例如安全防护方面
- SpringSecurity 社区资源相对比 Shiro 更加丰富

② 缺点：

- Shiro 的配置和使用比较简单，SpringSecurity 上手复杂些。
- Shiro 依赖性低，不需要任何框架和容器，可以独立运行。SpringSecurity 依赖 Spring 容器。

三、SpringSecurity+SpringMVC+Spring 环境

1. 创建 Web 工程，导包

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">

    <!-- 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
          </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. 配置 Spring 和 SpringMVC 文件

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>
```

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:contenxt="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">

</beans>
```

4. 配置 Spring Security（*）

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-class>
	</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>
```

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-5.2.xsd">

        <security:http>
                <security:http-basic/>
        </security:http>

        <security:authentication-manager>
        </security:authentication-manager>

</beans>
```

5. HttpBasic 方式的权限实现

① 编写 ProductController

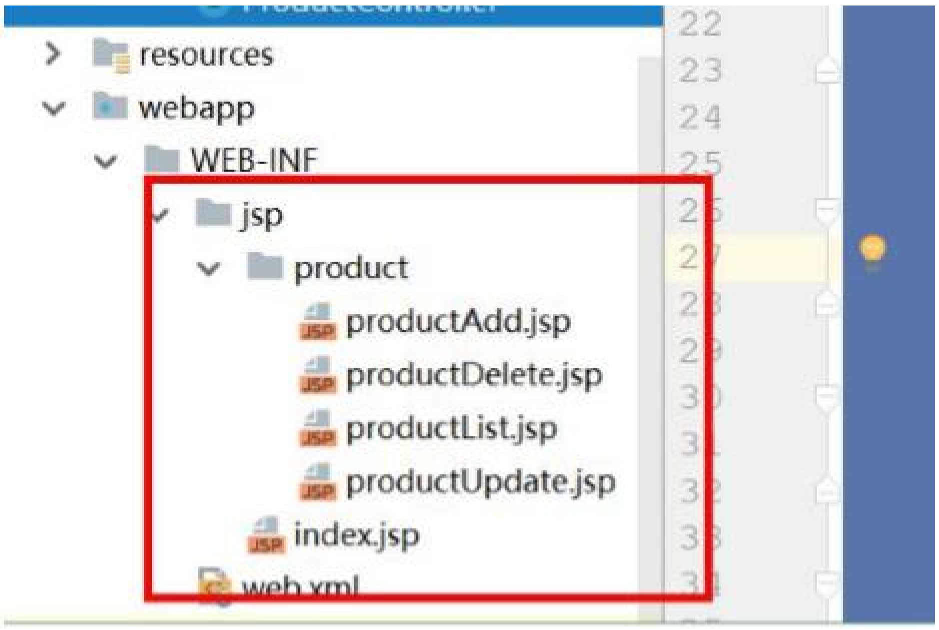
```
package club.banyuan.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/product")
public class ProductController {

    /**
     * 商品添加
     */
    @RequestMapping("/index")
    public String index() {
        return "index";
    }



    /**
     * 商品添加
     */
    @RequestMapping("/add")
    public String add() {
        return "product/productAdd";
    }

    /**
     * 商品修改
     */
    @RequestMapping("/update")
    public String update() {
        return "product/productUpdate";
    }

    /**
     * 商品修改
     */
    @RequestMapping("/list")
    public String list() {
        return "product/productList";
    }

    /**
     * 商品删除
     */
    @RequestMapping("/delete")
    public String delete() {
        return "product/productDelete";
    }
}
```

② 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:contenxt="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-->
    <contenxt:component-scan base-package="club.banyuan"/>

    <!--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>
```

补充页面：



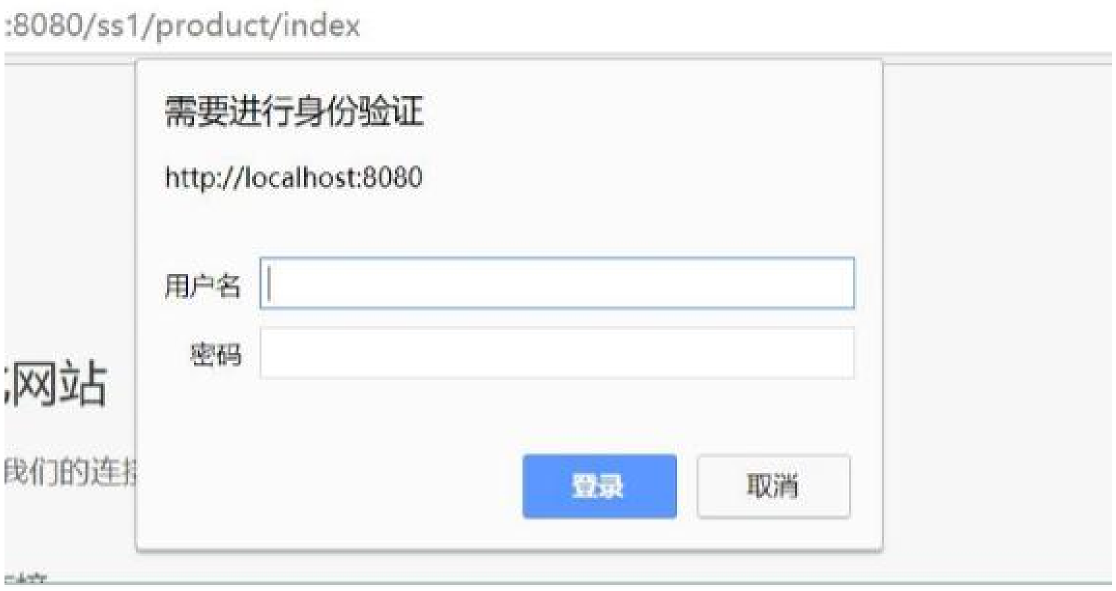
```
> resources
v webapp
  v WEB-INF
    v jsp
      v product
          productAdd.jsp
          productDelete.jsp
          productList.jsp
          productUpdate.jsp
        index.jsp
      web.xml
```

③ 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-5.2.xsd">

    <!-- <security:http>: spring 过滤器链配置:
             1）需要拦截什么资源
             2）什么资源什么角色权限
             3）定义认证方式：HttpBasic，FormLogin（*）
             4）定义登录页面，定义登录请求地址，定义错误处理方式
         -->
    <security:http>
        <!--
            pattern: 需要拦截资源
            access: 拦截方式
                    isFullyAuthenticated(): 该资源需要认证才可以访问
        -->
        <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

        <!-- security:http-basic: 使用 HttpBasic 方式进行登录（认证） -->
        <security:http-basic/>

       <!--
       security:authentication-manager:  认证管理器
            1）认证信息提供方式（账户名，密码，当前用户权限）
    -->
    <security:authentication-manager>
        <security:authentication-provider>
            <security:user-service>
                <security:user      name="eric"      password="{noop}123456"
authorities="ROLE_USER"/>
```

```
                    <security:user         name="jack"         password="{noop}123456"
authorities="ROLE_ADMIN"/>
               </security:user-service>
          </security:authentication-provider>

     </security:authentication-manager>

</beans>
```


:8080/ss1/product/index

需要进行身份验证

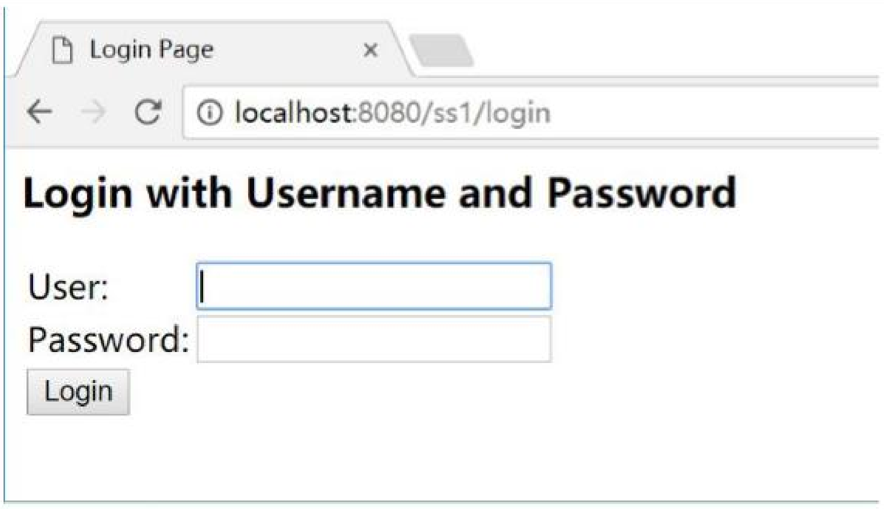
http://localhost:8080

6. FormLogin 方法的权限实现（*）

```
<?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.xsd
              http://www.springframework.org/schema/security

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

     <!-- <security:http>: spring 过滤器链配置：
               1）需要拦截什么资源
```

```
                2）什么资源什么角色权限
                3）定义认证方式：HttpBasic，FormLogin（*）
                4）定义登录页面，定义登录请求地址，定义错误处理方式
            -->
    <security:http>
        <!--
            pattern: 需要拦截资源
            access: 拦截方式
                    isFullyAuthenticated(): 该资源需要认证才可以访问
                    isAnonymous():只有匿名用户才可以访问（如果登录用户就无法访
问）
                    permitAll():允许所有人（匿名和登录用户）方法

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

        <!-- security:http-basic: 使用 HttpBasic 方式进行登录（认证） -->
<!--            <security:http-basic/>-->



        <security:form-login/>



    </security:http>

    <!--
       security:authentication-manager:  认证管理器
            1）认证信息提供方式（账户名，密码，当前用户权限）
    -->
    <security:authentication-manager>
        <security:authentication-provider>
            <security:user-service>
                <security:user        name="eric"        password="{noop}123456"
authorities="ROLE_USER"/>
                <security:user        name="jack"        password="{noop}123456"
```

```
authorities="ROLE_ADMIN"/>
               </security:user-service>
          </security:authentication-provider>

     </security:authentication-manager>
</beans>
```


Login Page

localhost:8080/ss1/login

Login with Username and Password

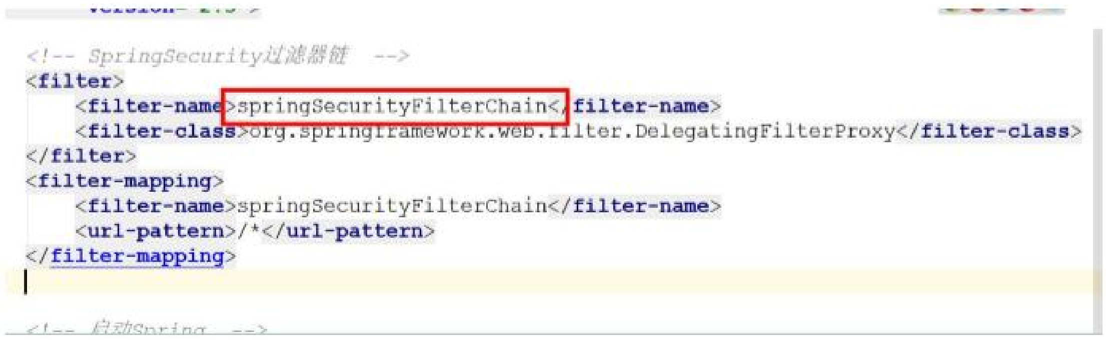
User:

Password:

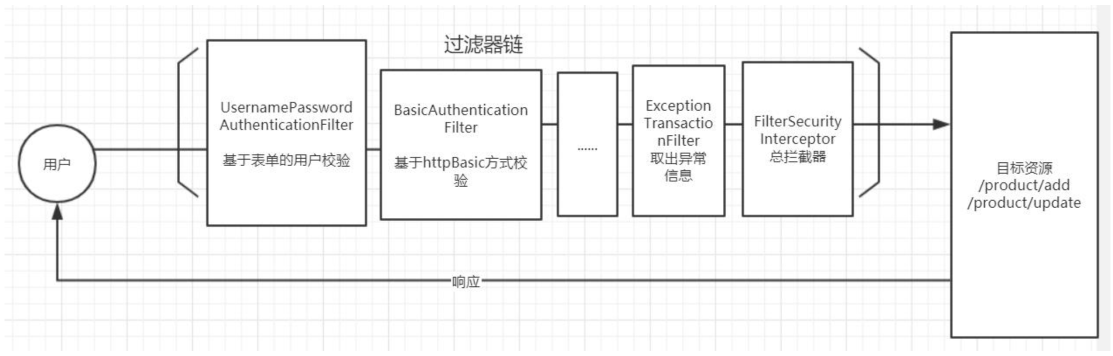
Login

① SpringSecurity 执行原理分析

底层：核心 SpringSecucrityFilterChain



```
<!-- SpringSecurity过滤器链 -->
<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>

<!-- 启动Spring -->
```


过滤器链
用户
UsernamePassword AuthenticationFilter
基于表单的用户校验
BasicAuthentication Filter
基于httpBasic方式校验
......
Exception TransactionFilter
取出异常信息
FilterSecurity Interceptor
总拦截器
目标资源
/product/add
/product/update
响应

总拦截器先拿到之前所有拦截器的执行结果，若是有问题就抛出异常

② 自定义登录页面，自定义登录请求

自定义登录页面啊（/WEB-INF/login.jsp）

默认的请求地址必须为/login meithod=post

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>登录页面</title>
</head>
<body>
<h3>登录页面</h3>
<form action="${pageContext.request.contextPath}/securityLogin" method="post">
 用户名:<input type="text" name="username"/><br/>
 用户名:<input type="password" name="password"/><br/>
 <input type="submit" value="登录"/>
</form>
</body>
</html>
```

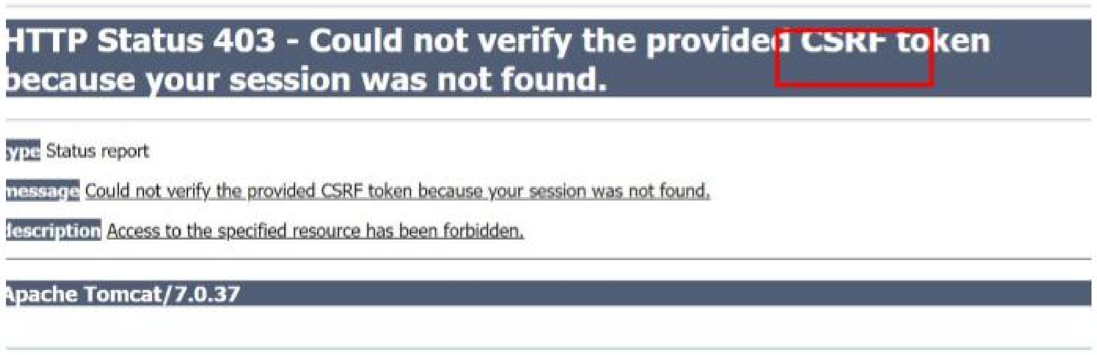
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:security="http://www.springframework.org/schema/security"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
  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

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

    <!-- <security:http>: spring 过滤器链配置：
              1）需要拦截什么资源
              2）什么资源什么角色权限
              3）定义认证方式：HttpBasic，FormLogin（*）
              4）定义登录页面，定义登录请求地址，定义错误处理方式
         -->
    <security:http>
        <!--
            pattern：需要拦截资源
            access：拦截方式
                isFullyAuthenticated()：该资源需要认证才可以访问
                isAnonymous():只有匿名用户才可以访问（如果登录用户就无法访
问）
                permitAll():允许所有人（匿名和登录用户）方法

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

        <!-- security:http-basic：使用 HttpBasic 方式进行登录（认证） -->
<!--        <security:http-basic/>-->

        <!--
        login-page：自定义登录页面
         login-processing-url:登录请求地址
         -->
<!--        <security:form-login/>-->
        <security:form-login                             login-page="/userLogin"
login-processing-url="/securityLogin" default-target-url="/product/index"/>

        <!-- 关闭 Spring Security CSRF 机制 -->
        <security:csrf disabled="true"/>
    </security:http>

    <!--
       security:authentication-manager：  认证管理器
            1）认证信息提供方式（账户名，密码，当前用户权限）
    -->
    <security:authentication-manager>

        <security:authentication-provider>
            <security:user-service>
                <security:user        name="eric"        password="{noop}123456"
authorities="ROLE_USER"/>
                <security:user        name="jack"        password="{noop}123456"
authorities="ROLE_ADMIN"/>
            </security:user-service>
        </security:authentication-provider>

    </security:authentication-manager>

</beans>
```

- 自定义登录请求（MainController）

```
package club.banyuan.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class MainController {

    /**
     *  登录页面
     * @return
     */
    @RequestMapping("/userLogin")
    public String login(){
        return "login";
    }
}
```

http://localhost:8080/spring_security_demo_war_exploded/userLogin


HTTP Status 403 - Could not verify the provided CSRF token because your session was not found.

type Status report

message Could not verify the provided CSRF token because your session was not found.

description Access to the specified resource has been forbidden.

Apache Tomcat/7.0.37

防范跨站攻击，暂时关闭

```
<!-- 关闭 Spring Security CSRF 机制 -->
<security:csrf disabled="true"/>
```

③ 使用 配置实现用户权限访问控制

```
<?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.xsd
            http://www.springframework.org/schema/security

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

    <!-- <security:http>: spring 过滤器链配置：
              1）需要拦截什么资源
              2）什么资源什么角色权限
              3）定义认证方式：HttpBasic，FormLogin（*）
              4）定义登录页面，定义登录请求地址，定义错误处理方式
         -->
    <security:http>
```

```
        <!--
            pattern: 需要拦截资源
            access: 拦截方式
                    isFullyAuthenticated(): 该资源需要认证才可以访问
                    isAnonymous():只有匿名用户才可以访问（如果登录用户就无法访
问）
                    permitAll():允许所有人（匿名和登录用户）方法

        -->
        <security:intercept-url pattern="/product/index" access="permitAll()"/>
        <security:intercept-url pattern="/userLogin" access="permitAll()"/>
        <security:intercept-url                              pattern="/product/add"
access="hasRole('ROLE_USER')"/>
        <security:intercept-url                           pattern="/product/update"
access="hasRole('ROLE_USER')"/>
        <security:intercept-url                             pattern="/product/list"
access="hasRole('ROLE_ADMIN')"/>
        <security:intercept-url                           pattern="/product/delete"
access="hasRole('ROLE_ADMIN')"/>
        <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

        <!-- security:http-basic: 使用 HttpBasic 方式进行登录（认证） -->
<!--        <security:http-basic/>-->

        <!--
        login-page: 自定义登录页面
         login-processing-url:登录请求地址
         -->
<!--        <security:form-login/>-->
        <security:form-login                               login-page="/userLogin"
login-processing-url="/securityLogin" default-target-url="/product/index"/>

        <!-- 自定义权限不足处理 -->
        <security:access-denied-handler error-page="/error"/>

        <!-- 关闭 Spring Security CSRF 机制 -->
```

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


        <!--
            security:authentication-manager:  认证管理器
                1）认证信息提供方式（账户名，密码，当前用户权限）
        -->
        <security:authentication-manager>
            <security:authentication-provider>
                <security:user-service>
                    <security:user          name="eric"          password="{noop}123456"
authorities="ROLE_USER"/>
                    <security:user          name="jack"          password="{noop}123456"
authorities="ROLE_ADMIN"/>
                </security:user-service>
            </security:authentication-provider>


        </security:authentication-manager>


</beans>
```

添加错误页面/WEB-INF/jsp/error.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>未授权提示页面</title>
</head>
<body>
亲，你没有权限访问该资源!
</body>
</html>
```

④ 自定义 UserDetailService 类实现用户权限 访问控制

关键：使用 UserDetailService 接口

- 创建 UserDetailService 接口实现类（club.banyuan.security.MyUserDetailService）

```
package club.banyuan.security;

import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;

public class MyUserDetailService implements UserDetailsService {


    /**
     * loadUserByUsername: 读取用户信息
     * @param username
     * @return
     * @throws UsernameNotFoundException
     */
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        //UserDetails: 封装用户数据的接口
        User user = new User( "test","{noop}123456"
                ,
AuthorityUtils.commaSeparatedStringToAuthorityList("ROLE_USER"));



        return user;
    }
}
```

其中 User 类就是 UserDetail 实现类，用于封装数据库账户信息

- spring-Security.xml 配置

```
<?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.xsd
                http://www.springframework.org/schema/security

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

      <!-- <security:http>: spring 过滤器链配置:
                1）需要拦截什么资源
                2）什么资源什么角色权限
                3）定义认证方式：HttpBasic，FormLogin（*）
                4）定义登录页面，定义登录请求地址，定义错误处理方式
            -->
      <security:http>
            <!--
                pattern: 需要拦截资源
                access: 拦截方式
                        isFullyAuthenticated(): 该资源需要认证才可以访问
                        isAnonymous():只有匿名用户才可以访问（如果登录用户就无法访
问）
                        permitAll():允许所有人（匿名和登录用户）方法

            -->
            <security:intercept-url pattern="/product/index" access="permitAll()"/>
            <security:intercept-url pattern="/userLogin" access="permitAll()"/>
            <security:intercept-url                              pattern="/product/add"
access="hasRole('ROLE_USER')"/>
            <security:intercept-url                           pattern="/product/update"
access="hasRole('ROLE_USER')"/>
            <security:intercept-url                               pattern="/product/list"
access="hasRole('ROLE_ADMIN')"/>
            <security:intercept-url                            pattern="/product/delete"
access="hasRole('ROLE_ADMIN')"/>
            <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>
```

```
        <!-- security:http-basic: 使用 HttpBasic 方式进行登录（认证） -->
<!--        <security:http-basic/>-->

        <!--
        login-page: 自定义登录页面
         login-processing-url:登录请求地址
         -->
<!--        <security:form-login/>-->
        <security:form-login                              login-page="/userLogin"
login-processing-url="/securityLogin" default-target-url="/product/index"/>

        <!-- 自定义权限不足处理 -->
        <security:access-denied-handler error-page="/error"/>

        <!-- 关闭 Spring Security CSRF 机制 -->
        <security:csrf disabled="true"/>
    </security:http>

    <!--
      security:authentication-manager:  认证管理器
            1）认证信息提供方式（账户名，密码，当前用户权限）
    -->
    <security:authentication-manager>
        <!--  自定义 UserDetailService 方式-->

        <security:authentication-provider
user-service-ref="myUserDetailService">
<!--        <security:authentication-provider>-->
<!--            <security:user-service>-->
<!--                  <security:user  name="eric"  password="{noop}123456"
authorities="ROLE_USER"/>-->
<!--                  <security:user  name="jack"  password="{noop}123456"
authorities="ROLE_ADMIN"/>-->
<!--            </security:user-service>-->
        </security:authentication-provider>
    </security:authentication-manager>

    <bean                                                    id="myUserDetailService"
class="club.banyuan.security.MyUserDetailService"/>

</beans>
```

⑤ 自定义登录成功与失败处理逻辑

关键:

- 登录成功处理：AuthenticationSuccessHandler
- 登录失败处理：AuthenticationFailureHandler

自定义登录成功逻辑

```
package club.banyuan.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;

public          class          MyAuthenticationSuccessHandler          implements
AuthenticationSuccessHandler{
    //ObjectMapper: jackson 框架的工具类，用于转换对象为 json 字符串
    private ObjectMapper objectMapper = new ObjectMapper();

    /**
     * @param request
     * @param response
     * @param authentication ： 代表认证成功后的信息
     * @throws IOException
     * @throws ServletException
     */
    @Override
    public void onAuthenticationSuccess(HttpServletRequest request,
HttpServletResponse response, Authentication authentication) throws IOException,
ServletException {

        //返回 json 字符串给前端
        Map result = new HashMap();
        result.put("success",true);

        String json = objectMapper.writeValueAsString(result);
        response.setContentType("text/json;charset=utf-8");
        response.getWriter().write(json);
    }

}
```

自定义登录失败逻辑

```
package club.banyuan.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;
```

```java
import java.util.Map;

public class MyAuthenticationFailureHandler implements AuthenticationFailureHandler {
    //ObjectMapper: jackson 框架的工具类，用于转换对象为 json 字符串
    private ObjectMapper objectMapper = new ObjectMapper();

    @Override
    public void onAuthenticationFailure(HttpServletRequest request, HttpServletResponse response, AuthenticationException exception) throws IOException, ServletException {
        //返回 json 字符串给前端
        Map result = new HashMap();
        result.put("succcess",false);

        String json = objectMapper.writeValueAsString(result);
        response.setContentType("text/json;charset=utf-8");
        response.getWriter().write(json);
    }

}
```

配置

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

    <!-- <security:http>: spring 过滤器链配置:
            1）需要拦截什么资源
```

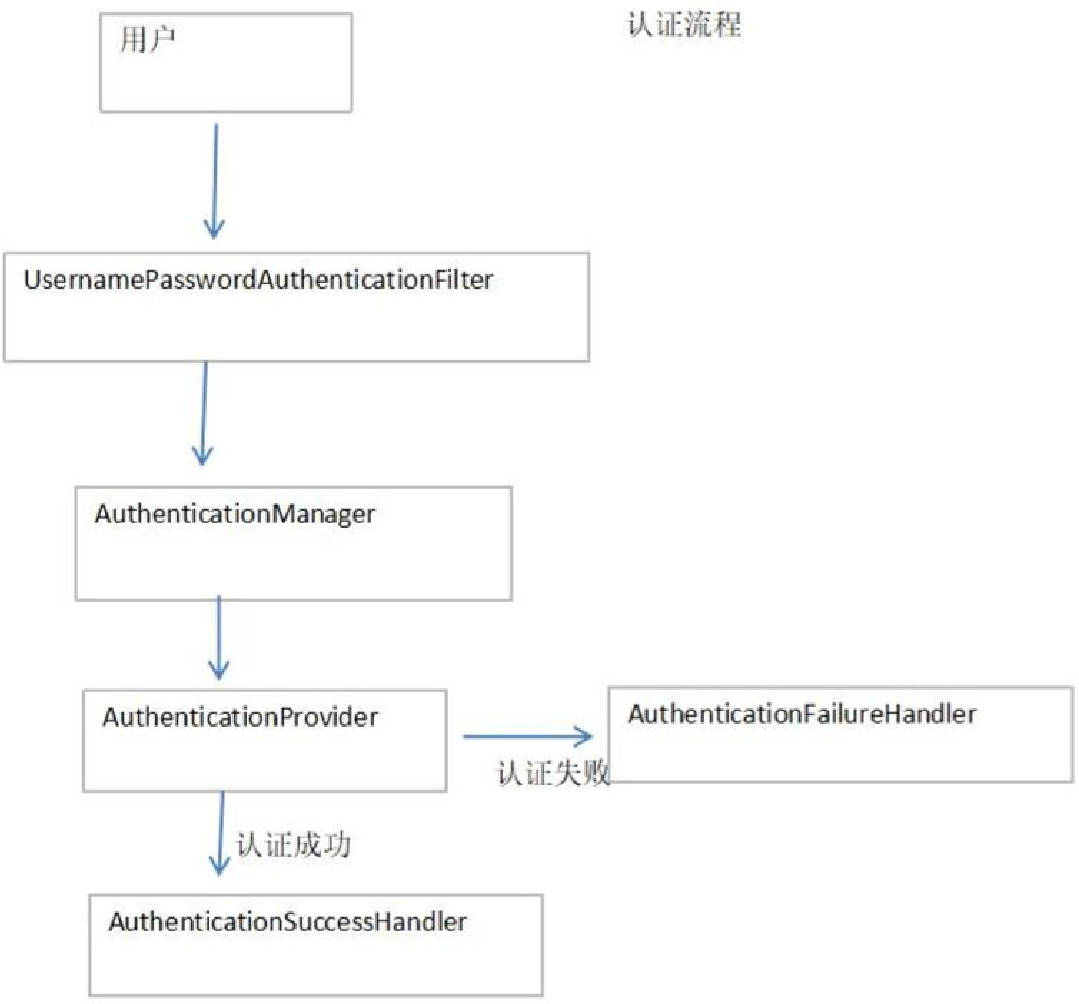
```
                2）什么资源什么角色权限
                3）定义认证方式：HttpBasic，FormLogin（*）
                4）定义登录页面，定义登录请求地址，定义错误处理方式
            -->
    <security:http>
        <!--
            pattern: 需要拦截资源
            access: 拦截方式
                    isFullyAuthenticated(): 该资源需要认证才可以访问
                    isAnonymous():只有匿名用户才可以访问(如果登录用户就无法访问)
                    permitAll():允许所有人（匿名和登录用户）方法

        -->
        <security:intercept-url pattern="/product/index" access="permitAll()"/>
        <security:intercept-url pattern="/userLogin" access="permitAll()"/>
        <security:intercept-url pattern="/product/add" access="hasRole('ROLE_USER')"/>
        <security:intercept-url pattern="/product/update" access="hasRole('ROLE_USER')"/>
        <security:intercept-url pattern="/product/list" access="hasRole('ROLE_ADMIN')"/>
        <security:intercept-url pattern="/product/delete" access="hasRole('ROLE_ADMIN')"/>
        <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

        <!-- security:http-basic: 使用 HttpBasic 方式进行登录（认证） -->
<!--        <security:http-basic/>-->

        <!--
        login-page: 自定义登录页面
         login-processing-url:登录请求地址
         -->
<!--        <security:form-login/>-->
<!--                                    <security:form-login    login-page="/userLogin"
login-processing-url="/securityLogin" default-target-url="/product/index"/>-->
        <security:form-login login-page="/userLogin" login-processing-url="/securityLogin"
                                    default-target-url="/product/index"

authentication-success-handler-ref="myAuthenticationSuccessHandler"
```

```
authentication-failure-handler-ref="myAuthenticationFailureHandler"/>

            <!-- 自定义权限不足处理 -->
            <security:access-denied-handler error-page="/error"/>

            <!-- 关闭 Spring Security CSRF 机制 -->
            <security:csrf disabled="true"/>
        </security:http>

        <!--
            security:authentication-manager:   认证管理器
                1）认证信息提供方式（账户名，密码，当前用户权限）
        -->
        <security:authentication-manager>
            <!--  自定义 UserDetailService 方式-->

            <security:authentication-provider user-service-ref="myUserDetailService">
<!--            <security:authentication-provider>-->
<!--                <security:user-service>-->
<!--                                            <security:user  name="eric"  password="{noop}123456"
authorities="ROLE_USER"/>-->
<!--                                            <security:user  name="jack"  password="{noop}123456"
authorities="ROLE_ADMIN"/>-->
<!--                </security:user-service>-->
            </security:authentication-provider>

        </security:authentication-manager>

        <bean id="myUserDetailService" class="club.banyuan.security.MyUserDetailService"/>

        <bean                                                    id="myAuthenticationSuccessHandler"
class="club.banyuan.security.MyAuthenticationSuccessHandler"/>

        <bean                                                    id="myAuthenticationFailureHandler"
class="club.banyuan.security.MyAuthenticationFailureHandler"/>
</beans>
```

修改 MyUserDetailService

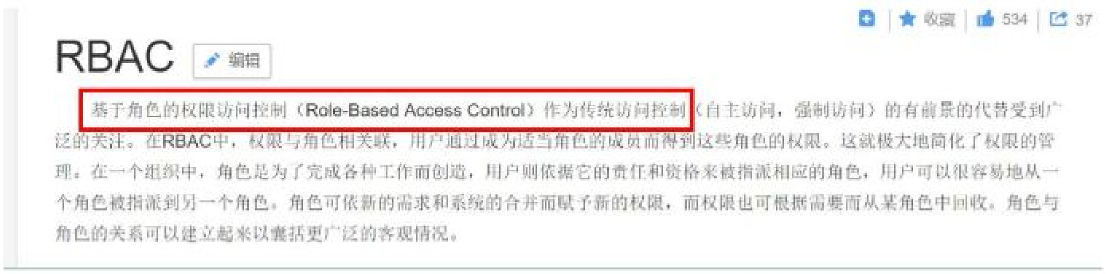
```
package club.banyuan.security;

import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

public class MyUserDetailService implements UserDetailsService {

    /**
     * loadUserByUsername: 读取用户信息
     * @param username
     * @return
     * @throws UsernameNotFoundException
     */
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        System.out.println(username);
        //UserDetails: 封装用户数据的接口

//          User user = new User( "test","{noop}123456"
//                      , AuthorityUtils.commaSeparatedStringToAuthorityList("ROLE_USER"));

        User user = new User( "test","{noop}123456",
true,true,true,true,AuthorityUtils.commaSeparatedStringToAuthorityList("ROLE_USER"));
        return user;
    }
}
```

⑥ 源码分析用户认证流程


认证流程
用户
UsernamePasswordAuthenticationFilter
AuthenticationManager
AuthenticationProvider
认证失败
AuthenticationFailureHandler
认证成功
AuthenticationSuccessHandler

四、 SpringSecurity+SSM

1. RBAC 模型简介



RBAC

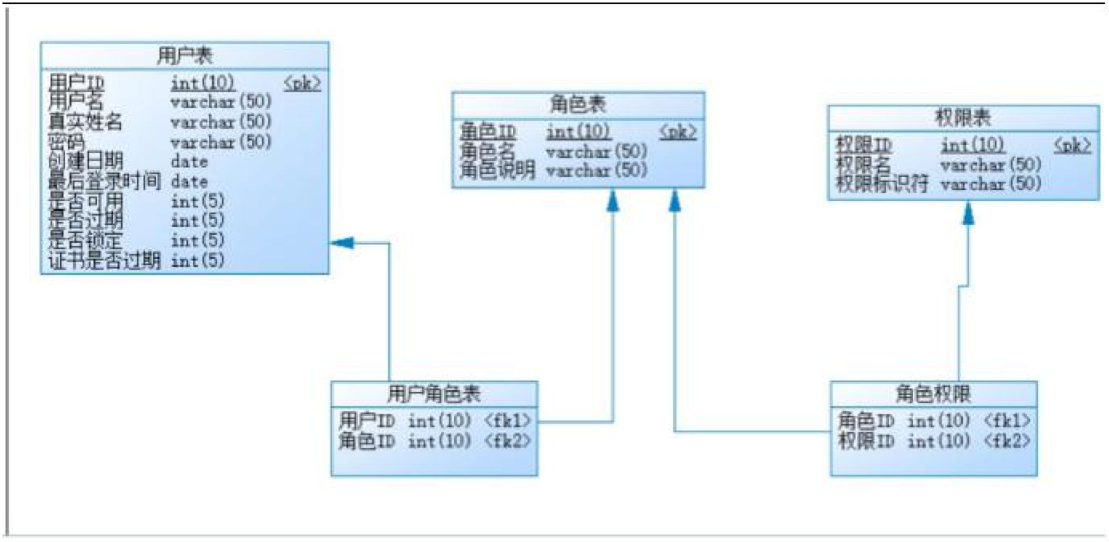
基于角色的权限访问控制（Role-Based Access Control）作为传统访问控制（自主访问，强制访问）的有前景的代替受到广泛的关注。在RBAC中，权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限。这就极大地简化了权限的管理。在一个组织中，角色是为了完成各种工作而创造，用户则依据它的责任和资格来被指派相应的角色，用户可以很容易地从一个角色被指派到另一个角色。角色可依新的需求和系统的合并而赋予新的权限，而权限也可根据需要而从某角色中回收。角色与角色的关系可以建立起来以囊括更广泛的客观情况。

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。



2. 搭建 SpringSecurity+SSM 运行环境

① 建立 web 项目，导入 jar 包

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">

    <display-name>maven-ssm</display-name>
    <welcome-file-list>
        <welcome-file>/</welcome-file>
    </welcome-file-list>
    <!-- post 乱码过滤器 -->
    <filter>
```

```
        <filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>utf-8</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>CharacterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!-- 前端控制器 -->
    <servlet>
        <servlet-name>ssmServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <!--
        contextConfigLocation 不是必须的， 如果不配置 contextConfigLocation，
        springmvc 的配置文件默认在：WEB-INF/servlet 的 name+"-servlet.xml"
        -->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:applicationContext.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>ssmServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <!-- SpringSecurity 过滤器链  -->
    <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>

</web-app>
```

③ 配置 SpringSecurity 和 SSM 文件

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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-4.3.xsd">

    <import resource="applicationContext-dao.xml"/>
    <import resource="applicationContext-mvc.xml"/>
    <import resource="applicationContext-service.xml"/>
    <import resource="applicationContext-trans.xml"/>
    <import resource="applicationContext-security.xml"/>
</beans>
```

④ MyBatis 整合 Spring

3. 实现用户查询与权限查询持久层方法

① 创建实体类

- User

```
package club.banyuan.entity;

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;

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;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                ", realname='" + realname + '\'' +
                ", password='" + password + '\'' +
                ", createDate=" + createDate +
                ", lastLoginTime=" + lastLoginTime +
                ", enabled=" + enabled +
                ", accountNonExpired=" + accountNonExpired +
                ", accountNonLocked=" + accountNonLocked +
                ", credentialsNonExpired=" + credentialsNonExpired +
                ", authorities=" + authorities +
                '}';
    }
}
```

- Role

```
package club.banyuan.entity;

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;
    }
}
```

- Permission

```
package club.banyuan.entity;
```

```
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;
    }
}
```

② 编写持久层接口

```
package club.banyuan.dao;
```

```
import club.banyuan.entity.Permission;
import club.banyuan.entity.User;

import java.util.List;

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);

}
```

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">
<!--namepace:dao 接口路径-->
<mapper namespace="club.banyuan.dao.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 编写持久层测试类

```
package club.banyuan.test;

import club.banyuan.dao.UserMapper;
import club.banyuan.entity.Permission;
import club.banyuan.entity.User;
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;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext-dao.xml")
```

```
public class TestDao {
    @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());
        }
    }
}
```

4. 自定义 UserDetailService 实现动态数据权 限访问

```
package club.banyuan.security;


import club.banyuan.dao.UserMapper;
import club.banyuan.entity.Permission;
import club.banyuan.entity.User;
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;

public class MyUserDetailService implements UserDetailsService {

    @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);
        if(user!=null) {
            // 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);

            System.out.println("当前用户：" + user);
        }
        return user;
    }

}
```

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

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

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

          <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入加密,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="club.banyuan.security.MyUserDetailService"/>

        <bean                                          id="myAuthenticationSuccessHandler"
class="club.banyuan.security.MyAuthenticationSuccessHandler"/>
        <bean                                          id="myAuthenticationFailureHandler"
class="club.banyuan.security.MyAuthenticationFailureHandler"/>

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

5. PasswordEncoder 密码加密

关键：PasswordEncoder 接口的实现类

```
package club.banyuan.test;

import org.junit.Test;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

public class TestPassword {
    @Test
    public void testValidatePwd(){
        String pass = "123456";
        BCryptPasswordEncoder      bcryptPasswordEncoder      =      new
BCryptPasswordEncoder();
        String hashPass = bcryptPasswordEncoder.encode(pass);
        System.out.println(hashPass);

        boolean flag = bcryptPasswordEncoder.matches("admin",hashPass);
```

```
            System.out.println(flag);
        }
}
```

6. 登录成功与登录失败处理

① 同步方式处理

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>登录页面</title>
</head>
<body>
<h3>登录页面</h3>
<c:if test="${not empty param.error}">
    <font color="red">用户名或者密码错误</font>
</c:if>
<form action="${pageContext.request.contextPath}/securityLogin" method="post">
    用户名:<input type="text" name="username"/><br/>
    用户名:<input type="password" name="password"/><br/>
    <input type="submit" value="登录"/>
</form></body>
</html>
```

② 异步方式处理

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2020/7/13 0013
  Time: 7:37
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>登录页面</title>
    <script                                                  type="text/javascript"
src="${pageContext.request.contextPath}/js/jquery.min.js"></script>

    <script type="text/javascript">
        $(function () {
            $("#loginBtn").click(function () {

$.post("${pageContext.request.contextPath}/securityLogin",$("#loginForm").serializ
e(),function (data) {
                    if(data.success){

window.location.href="${pageContext.request.contextPath}/product/index";
                    }else{
                        alert("登录失败："+data.errorMsg);
                    }
                },"json");
            }) ;
        });

    </script>
</head>
<body>
<h3>登录页面</h3>
<c:if test="${not empty param.error}">
    <font color="red">用户名或者密码错误</font>
</c:if>
```

```
<form method="post" id="loginForm">
    用户名:<input type="text" name="username"/><br/>
    用户名:<input type="password" name="password"/><br/>
    <input type="button" id="loginBtn" value="登录"/>
</form>
</body>
</html>
```

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


    <security:http>
        <security:intercept-url pattern="/product/index" access="permitAll()"/>
        <security:intercept-url pattern="/userLogin" access="permitAll()"/>
        <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="/js/**" access="permitAll()"/>
        <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>


        <security:form-login login-page="/userLogin" login-processing-url="/securityLogin"

authentication-success-handler-ref="myAuthenticationSuccessHandler"
```

```
authentication-failure-handler-ref="myAuthenticationFailureHandler"/>

            <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入加密,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="club.banyuan.security.MyUserDetailService"/>

        <bean                                    id="myAuthenticationSuccessHandler"
class="club.banyuan.security.MyAuthenticationSuccessHandler"/>
        <bean                                    id="myAuthenticationFailureHandler"
class="club.banyuan.security.MyAuthenticationFailureHandler"/>

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

7. 自定义图形验证码

制作一个图形验证码(WEB-INF/jsp/imageCode.jsp)

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2020/7/19 0019
  Time: 20:35
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ 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();
%>
```

```
package club.banyuan.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class MainController {


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

在登录页面使用图形验证码

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2020/7/13 0013
  Time: 7:37
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>登录页面</title>
    <script                                                     type="text/javascript"
src="${pageContext.request.contextPath}/js/jquery.min.js"></script>

    <script type="text/javascript">
        $(function () {
            $("#loginBtn").click(function () {

$.post("${pageContext.request.contextPath}/securityLogin",$("#loginForm").serializ
e(),function (data) {
                    if(data.success){

window.location.href="${pageContext.request.contextPath}/product/index";
                    }else{
                        alert("登录失败："+data.errorMsg);
                    }
                },"json");
            }) ;
        });

    </script>
</head>
<body>
<h3>登录页面</h3>
<c:if test="${not empty param.error}">
```

```
    <font color="red">用户名或者密码错误</font>
</c:if>
<form method="post" id="loginForm">
    用户名:<input type="text" name="username"/><br/>
    用户名:<input type="password" name="password"/><br/>
    验  证  码  :<input  type="text"  name="imageCode"/><img
src="${pageContext.request.contextPath}/imageCode"/><br/>
    <input type="button" id="loginBtn" value="登录"/>
</form>
</body>
</html>
```

自定义验证码校验过滤器

```
package club.banyuan.security;

import org.springframework.security.core.AuthenticationException;
import
org.springframework.security.web.authentication.AuthenticationFailureHandler;
import org.springframework.util.StringUtils;
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;

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 AuthentFailureHandler 处理

authenticationFailureHandler.onAuthenticationFailure(request,response,e);
                return;
            }
        }

        filterChain.doFilter(request,response);
    }
}
```

```
package club.banyuan.security;

import org.springframework.security.core.AuthenticationException;

public class ImageCodeException extends AuthenticationException {

    public ImageCodeException(String msg, Throwable t) {
        super(msg, t);
    }

    public ImageCodeException(String msg) {
        super(msg);
    }
}
```

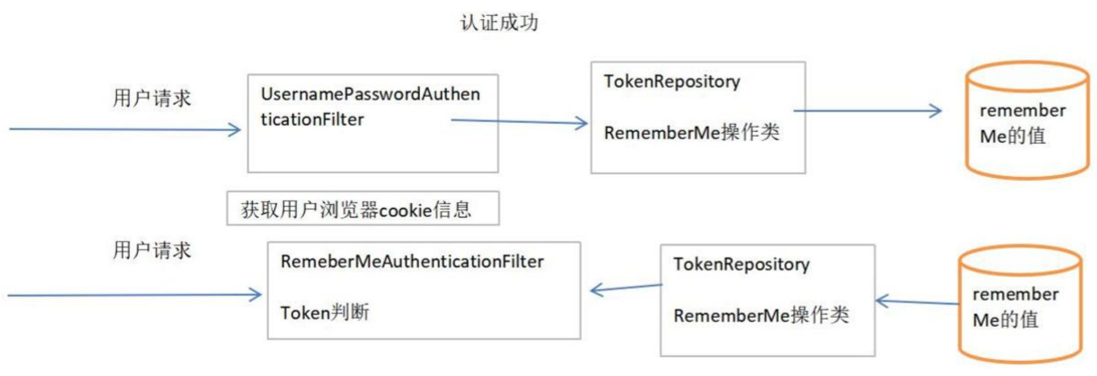
配置 spring-security.xml

```
<?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.xsd
           http://www.springframework.org/schema/security

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

    <security:http>
        <security:intercept-url pattern="/product/index" access="permitAll()"/>
        <security:intercept-url pattern="/userLogin" access="permitAll()"/>
        <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="/js/**" access="permitAll()"/>
        <security:intercept-url pattern="/imageCode*" access="permitAll()"/>
        <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

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

authentication-success-handler-ref="myAuthenticationSuccessHandler"

authentication-failure-handler-ref="myAuthenticationFailureHandler"/>


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


        <security:csrf disabled="true"/>


        <!-- 自定义 Spring Security 过滤器 -->
        <security:custom-filter                ref="imageCodeAuthenticationFilter"
before="FORM_LOGIN_FILTER"/>
    </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入加密,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="club.banyuan.security.MyUserDetailService"/>

    <bean                             id="myAuthenticationSuccessHandler"
class="club.banyuan.security.MyAuthenticationSuccessHandler"/>
    <bean                               id="myAuthenticationFailureHandler"
class="club.banyuan.security.MyAuthenticationFailureHandler"/>

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



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

8. rememberMe 记住我

① 执行流程：



② 在登录页面添加 remember-me

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2020/7/13 0013
  Time: 7:37
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>登录页面</title>
    <script                                                    type="text/javascript"
src="${pageContext.request.contextPath}/js/jquery.min.js"></script>

    <script type="text/javascript">
        $(function () {
            $("#loginBtn").click(function () {
```

```
$.post("${pageContext.request.contextPath}/securityLogin",$("#loginForm").serialize(),function (data) {
                    if(data.success){

window.location.href="${pageContext.request.contextPath}/product/index";
                    }else{
                        alert("登录失败："+data.errorMsg);
                    }
                },"json");
            }) ;
        });

    </script>
</head>
<body>
<h3>登录页面</h3>
<c:if test="${not empty param.error}">
    <font color="red">用户名或者密码错误</font>
</c:if>
<form method="post" id="loginForm">
    用户名:<input type="text" name="username"/><br/>
    用户名:<input type="password" name="password"/><br/>
    验证码:<input type="text" name="imageCode"/><img src="${pageContext.request.contextPath}/imageCode"/><br/>
    记住我:<input type="checkbox" name="remember-me" value="true"><br/>
    <input type="button" id="loginBtn" value="登录"/>
</form>
</body>
</html>
```

③ 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.xsd
            http://www.springframework.org/schema/security

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


    <security:http>
        <security:intercept-url pattern="/product/index" access="permitAll()"/>
        <security:intercept-url pattern="/userLogin" access="permitAll()"/>
        <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="/js/**" access="permitAll()"/>
        <security:intercept-url pattern="/imageCode*" access="permitAll()"/>
        <security:intercept-url pattern="/**" access="isFullyAuthenticated()"/>

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

authentication-success-handler-ref="myAuthenticationSuccessHandler"

authentication-failure-handler-ref="myAuthenticationFailureHandler"/>
```

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

        <security:csrf disabled="true"/>

        <!-- 自定义 Spring Security 过滤器 -->
        <security:custom-filter                    ref="imageCodeAuthenticationFilter"
before="FORM_LOGIN_FILTER"/>

        <!-- 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入加密,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="club.banyuan.security.MyUserDetailService"/>

    <bean                                      id="myAuthenticationSuccessHandler"
class="club.banyuan.security.MyAuthenticationSuccessHandler"/>
    <bean                                      id="myAuthenticationFailureHandler"
class="club.banyuan.security.MyAuthenticationFailureHandler"/>

    <bean                                                       id="passwordEncoder"
```

```
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>

    <bean                                  id="imageCodeAuthenticationFilter"
class="club.banyuan.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>
```

9. SpringSecurity 权限标签使用

在 JSP 页面导入标签库

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

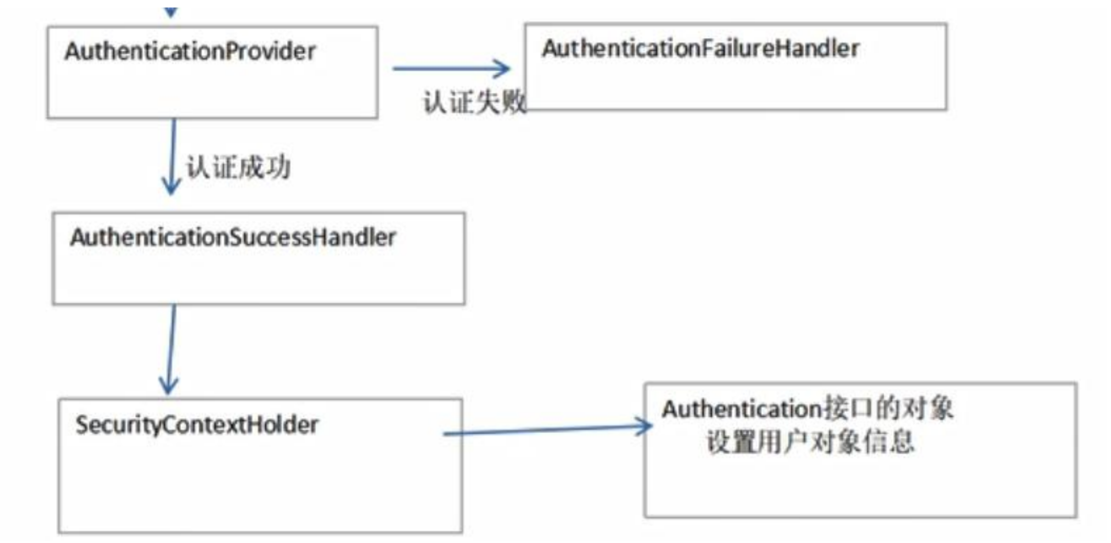
使用 Security 标签

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@            taglib           uri="http://www.springframework.org/security/tags"
prefix="security" %>
<html>
<head>
    <title>首页</title>
</head>
<body>
欢迎用户，${username}<br/>
以下是网站的功能：<br/>
<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>
</body>
</body>
</html>
```

10. 如何获取登录后用户名

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



```
package club.banyuan.controller;
```

```
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/product")
public class ProductController {

    /**
     * 商品添加
     */
    @RequestMapping("/index")
    public String index(Model model) {
        //获取登录后用户: UserDetail 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";
    }

……
}
```

五、Spring Security+ Spring Boot 权限管理

1. 快速搭建 SpringBoot 运行环境

建立 maven 项目，导入坐标

```
<?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">
    <modelVersion>4.0.0</modelVersion>

    <groupId>club.banyuan</groupId>
    <artifactId>spingboot_security</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <!-- Spring Boot 父工程 -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.0.RELEASE</version>
    </parent>

    <dependencies>
        <!-- web 支持，SpringMVC， Servlet 支持等 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- thymeleaf -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
```

```
                <artifactId>spring-boot-starter-thymeleaf</artifactId>
            </dependency>
        </dependencies>



        <properties>
            <java.version>1.8</java.version>
        </properties>

</project>
```

2. 准备访问的资源

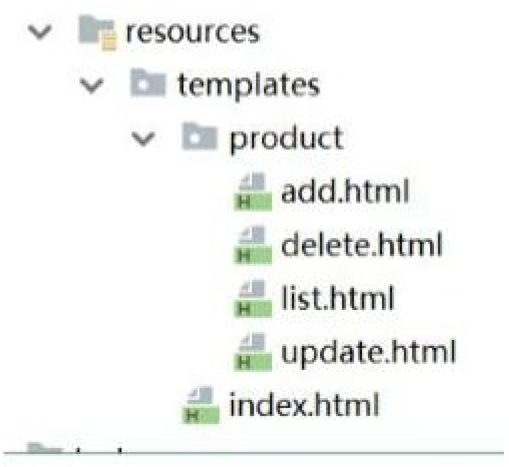
编写 Controller

```
package club.banyuan.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class MainController {

        @RequestMapping("index")
        public String index(){
            return "index";
        }
}
```

```
package club.banyuan.controller;

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

```
@Controller
@RequestMapping("product")
public class ProductController {

    @RequestMapping("add")
    public String add(){
        return "product/add";
    }

    @RequestMapping("update")
    public String update(){
        return "product/update";
    }

    @RequestMapping("delete")
    public String delete(){
        return "product/delete";
    }

    @RequestMapping("list")
    public String list(){
        return "product/list";
    }
}
```

编写页面


```
resources
  templates
    product
      add.html
      delete.html
      list.html
      update.html
    index.html
```

3. 整合 SpringSecurity-HttpBasic 权限实现

导入 SpringSecurity 坐标

```
        <!--
https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-s
ecurity -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
```

编写 SpringSecurityConfig 配置类

```
package club.banyuan.security;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authenticati
onManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecu
rity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityCon
```

```
figurerAdapter;

@Configuration
@EnableWebSecurity   //启动 SpringSecurity 过滤器链
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

    //该方法的作用就是代替之前配置：<security:authentication-manager>
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{

auth.inMemoryAuthentication().withUser("eric").password("{noop}123456").authori
ties("PRODUCT_ADD","PRODUCT_UPDATE");
    }

    //该方法的作用就是代替之前配置：<security:http>
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/**")
                .fullyAuthenticated()
                .and()
                .httpBasic();
    }
}
```

4. 整合 SpringSecurity-FormLogin 权限实现

改为 formLogin：

```
package club.banyuan.security;


import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authenticati
```

```
onManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecu
rity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityCon
figurerAdapter;

@Configuration
@EnableWebSecurity  //启动 SpringSecurity 过滤器链
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

    //该方法的作用就是代替之前配置：<security:authentication-manager>
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{

auth.inMemoryAuthentication().withUser("eric").password("{noop}123456").authori
ties("PRODUCT_ADD","PRODUCT_UPDATE");
    }

    //该方法的作用就是代替之前配置：<security:http>
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/product/add").hasAuthority("PRODUCT_ADD")
                .antMatchers("/product/update").hasAuthority("PRODUCT_UPDAT
E")
                .antMatchers("/product/list").hasAuthority("PRODUCT_LIST")
                .antMatchers("/product/delete").hasAuthority("PRODUCT_DELETE
")
                .antMatchers("/login").permitAll()
                .antMatchers("/**")
                .fullyAuthenticated()
                .and()
                .formLogin().loginPage("/login")
                .and()
                .csrf().disable();
    }
}
```

```
package club.banyuan.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class MainController {

    @RequestMapping("index")
    public String index(){
        return "index";
    }

    @RequestMapping("403")
    public String forbidden(){
        return "403";
    }

    @RequestMapping("login")
    public String login(){
        return "login";
    }
}
```

login.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>登录页面</title>
</head>
<body>
<h3>登录页面</h3>
<form action="/login" method="post">
    用户名:<input type="text" name="username"/><br/>
    用户名:<input type="password" name="password"/><br/>
    <input type="submit" value="登录"/>
</form>
</body>
</html>
```

订制 403（ErrorPageConfig）

```
package club.banyuan.security;

import org.springframework.boot.web.server.ErrorPage;
import org.springframework.boot.web.server.WebServerFactoryCustomizer;
import
org.springframework.boot.web.servlet.server.ConfigurableServletWebServerFactor
y;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpStatus;



@Configuration
public class ErrorPageConfig implements
WebServerFactoryCustomizer<ConfigurableServletWebServerFactory> {

    //ErrorPage:定义错误页面
    //参数一：HttpStatus.FORBIDDEN：  该错误接收什么错误状态码
    //参数二：交给哪个请求处理
    @Override
    public void customize(ConfigurableServletWebServerFactory factory) {
        factory.addErrorPages(new ErrorPage(HttpStatus.FORBIDDEN,"/403"));
    }
}
```

403.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>错误提示页面</title>
</head>
<body>
你无权访问该资源!
</body>
</html>
```