学习目标

- Spring Security 框架简介
- Spring Security 与 Shiro 的区别
- Spirng Security+SpringMVC+Spring 环境整合
- HttpBasic 方式的权限实现
- FormLogin 方法的权限实现
- Spring Security 执行原理分析
- 自定义登录页面，自定义登录请求
- user-service 配置实现用户权限访问控制
- 自定义 UserDetailService 类实现用户权限访问控制

1. Spring Security 框架简介

官网：https://projects.spring.io/spring-security/

Spring Security 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。

Spring Security 的功能：

1）Authentication：认证，就是用户登录。

2）Authorization：授权，判断用户拥有什么权限，可以访问什么资源。

3）安全防护，防止跨站请求，session 攻击等

4）非常容易结合 SpringMVC 进行使用

2. Spring Security 与 Shiro 的区别

2.1. 相同点

1）认证功能

2）授权功能

3）加密功能

4）会话管理

5）缓存支持

6）rememberMe 功能

....

2.2. 不同点

优点：

1）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。

2）Spring Security 功能比 Shiro 更加丰富些，例如安全防护方面

3）Spring Security 社区资源相对比 Shiro 更加丰富

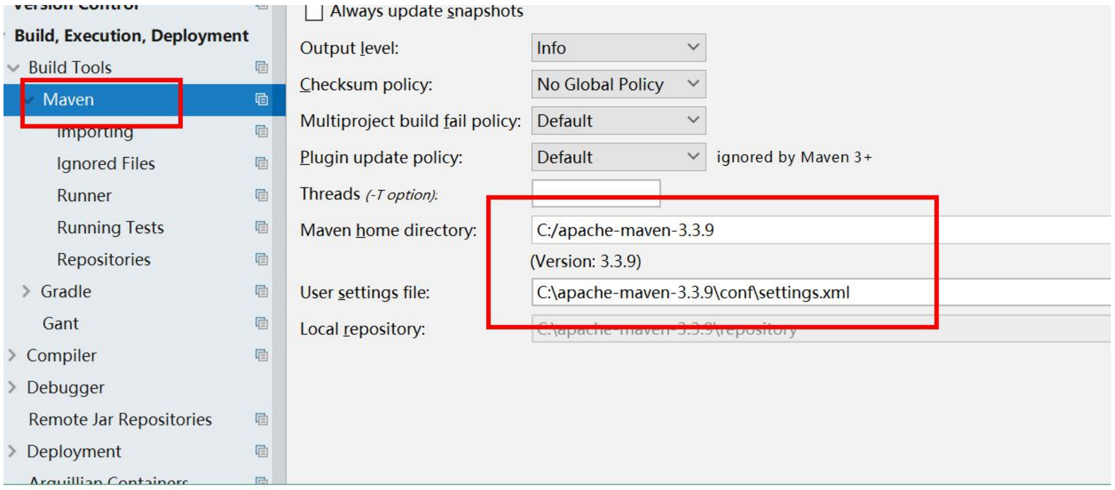
缺点：

1）Shiro 的配置和使用比较简单，Spring Security 上手复杂些。

2）Shiro 依赖性低，不需要任何框架和容器，可以独立运行。Spring Security 依赖 Spring 容器。

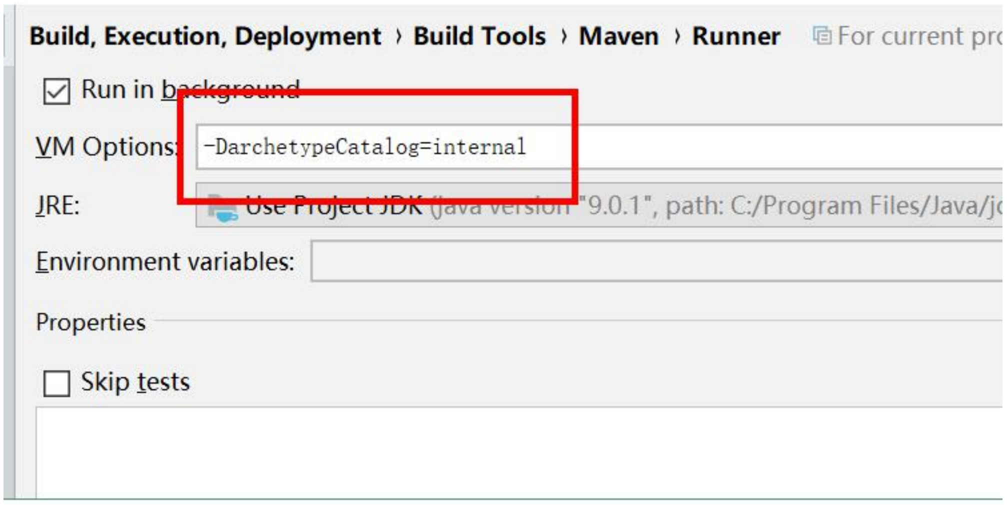
3. Spring Security+SpringMVC+Spring 环境

3.1. IDEA 开发环境配置



Always update snapshots
Output level: Info
Checksum policy: No Global Policy
Multiproject build fail policy: Default
Plugin update policy: Default ignored by Maven 3+
Threads (-T option):
Maven home directory: C:/apache-maven-3.3.9
(Version: 3.3.9)
User settings file: C:\apache-maven-3.3.9\conf\settings.xml
Local repository: C:\apache-maven-3.3.9\repository

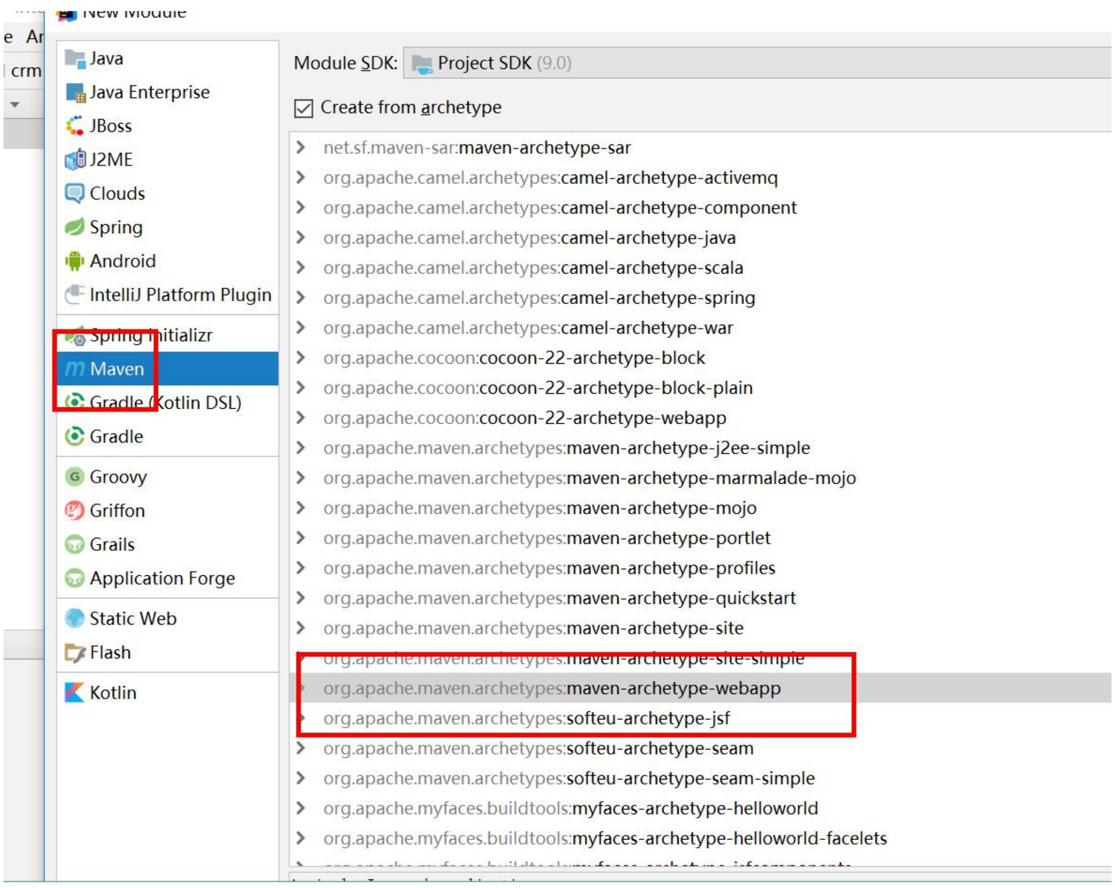
在 Runner 配置了参数：


Build, Execution, Deployment › Build Tools › Maven › Runner

☑ Run in background
VM Options: -DarchetypeCatalog=internal
JRE: Use Project JDK (java version "9.0.1", path: C:/Program Files/Java/j
Environment variables:
Properties
☐ Skip tests

-DarchetypeCatalog=internal

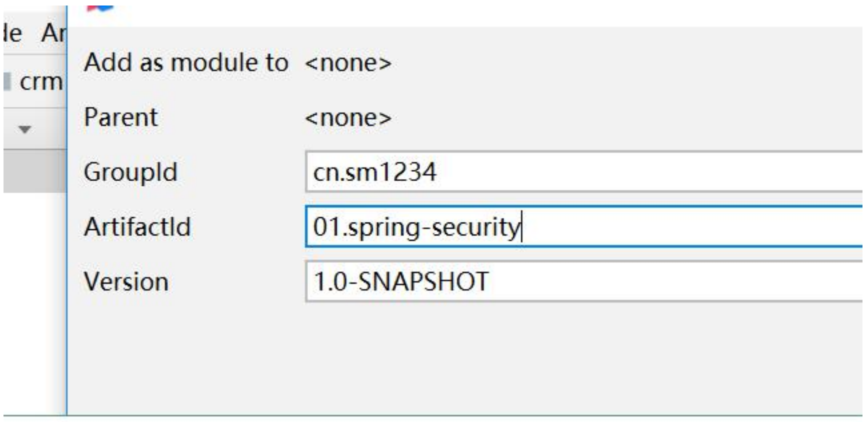
3.2. 创建 Maven 项目



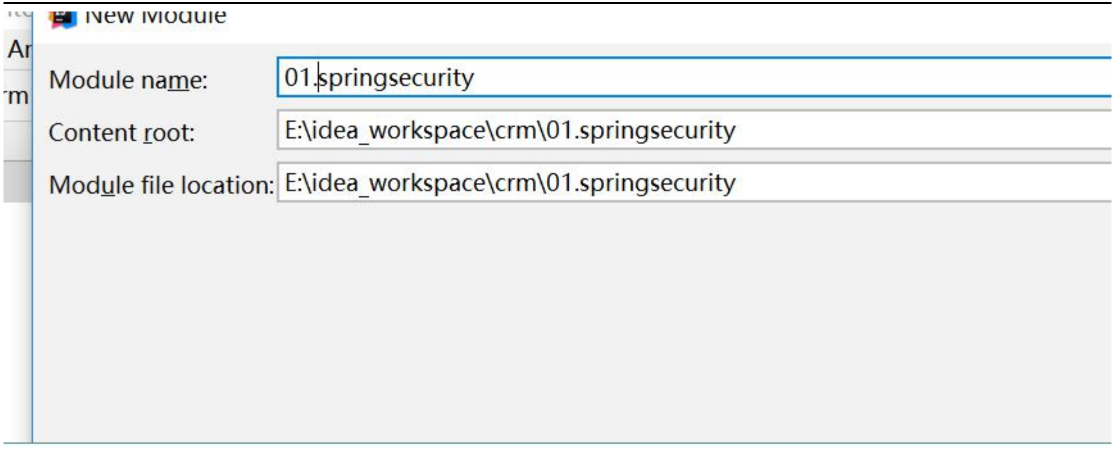
Module SDK: Project SDK (9.0)

☑ Create from archetype

net.sf.maven-sar:maven-archetype-sar
org.apache.camel.archetypes:camel-archetype-activemq
org.apache.camel.archetypes:camel-archetype-component
org.apache.camel.archetypes:camel-archetype-java
org.apache.camel.archetypes:camel-archetype-scala
org.apache.camel.archetypes:camel-archetype-spring
org.apache.camel.archetypes:camel-archetype-war
org.apache.cocoon:cocoon-22-archetype-block
org.apache.cocoon:cocoon-22-archetype-block-plain
org.apache.cocoon:cocoon-22-archetype-webapp
org.apache.maven.archetypes:maven-archetype-j2ee-simple
org.apache.maven.archetypes:maven-archetype-marmalade-mojo
org.apache.maven.archetypes:maven-archetype-mojo
org.apache.maven.archetypes:maven-archetype-portlet
org.apache.maven.archetypes:maven-archetype-profiles
org.apache.maven.archetypes:maven-archetype-quickstart
org.apache.maven.archetypes:maven-archetype-site
org.apache.maven.archetypes:maven-archetype-site-simple
org.apache.maven.archetypes:maven-archetype-webapp
org.apache.maven.archetypes:softeu-archetype-jsf
org.apache.maven.archetypes:softeu-archetype-seam
org.apache.maven.archetypes:softeu-archetype-seam-simple
org.apache.myfaces.buildtools:myfaces-archetype-helloworld
org.apache.myfaces.buildtools:myfaces-archetype-helloworld-facelets



Add as module to	<none>
Parent	<none>
GroupId	cn.sm1234
ArtifactId	01.spring-security
Version	1.0-SNAPSHOT


New Module
Module name: 01.springsecurity
Content root: E:\idea_workspace\crm\01.springsecurity
Module file location: E:\idea_workspace\crm\01.springsecurity

3.3. 导入相关的坐标

使用的框架：

1）Spring（包括 Spring-MVC）

2）ServletAPI

3）Spring-Security（*）

```xml
<?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>cn.sm1234</groupId>

  <artifactId>01.spring-security</artifactId>

  <version>1.0-SNAPSHOT</version>

  <packaging>war</packaging>


  <url>http://www.example.com</url>
```

```xml
<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>
  </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>
```

3.4. 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</liste
ner-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-cl
ass>
```

```
        <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>
```

3.5. 配置 Spring 和 SpringMVC 文件

3.5.1. 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>
```

3.5.2. 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">


</beans>
```

3.6. 配置 Spring Security（*）

web.xml

```
<!-- 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</liste
ner-class>
</listener>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        classpath:applicationContext.xml
        classpath:spring-security.xml
    </param-value>
</context-param>
```

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:http-basic/>

    </security:http>


    <security:authentication-manager>



    </security:authentication-manager>

</beans>
```

4. HttpBasic 方式的权限实现

4.1. 编写 ProductController

```
package cn.sm1234.controller;



import org.springframework.stereotype.Controller;

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

```
/**
 * @author http://www.sm1234.cn
 * @version 1.0
 * @description PACKAGE_NAME
 * @date 18/4/12
 */
@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";
    }

    /**
```

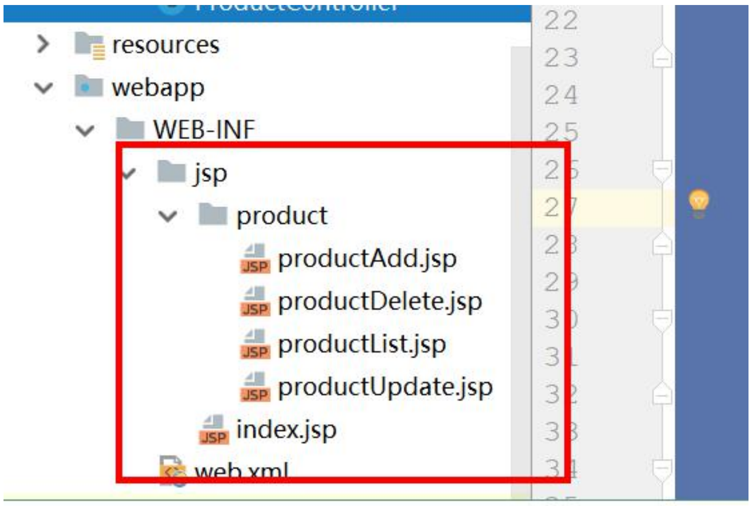
```
     * 商品修改
     */
    @RequestMapping("/update")
    public String update(){
        return "product/productUpdate";
    }

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

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

4.2. 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="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>
```

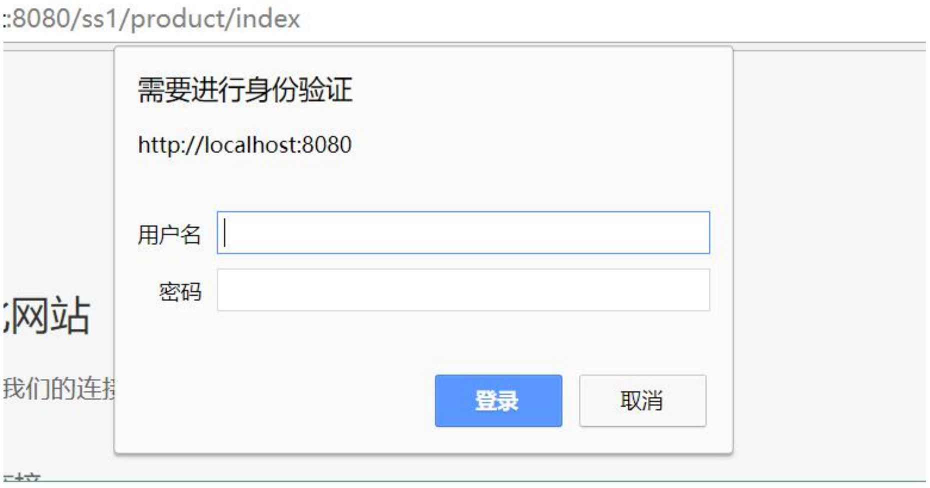
补充页面：



4.3. 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-4.2.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: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="123456"
authorities="ROLE_USER"/>
            </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

用户名

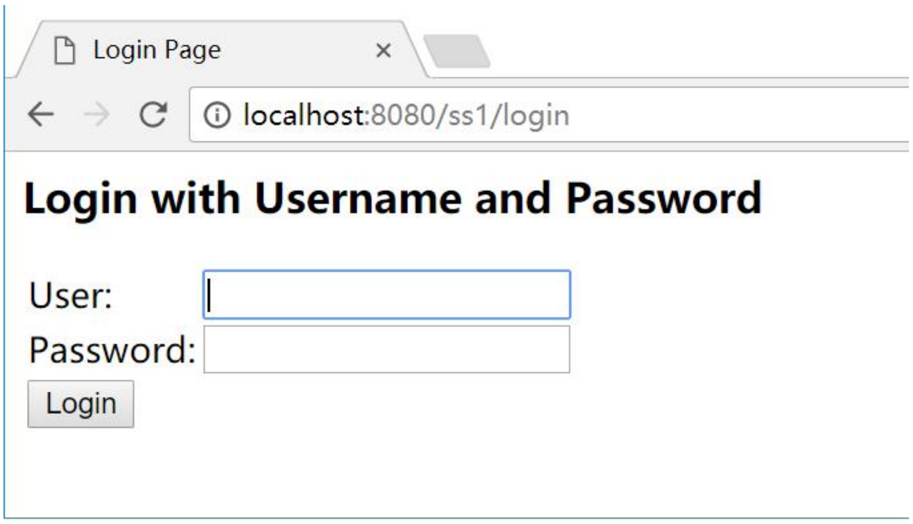
密码

登录 取消

5. 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-4.2.xsd

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



http://www.springframework.org/schema/security/spring-security-4.2.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

                  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="123456"
authorities="ROLE_USER"/>

            </security:user-service>

        </security:authentication-provider>

    </security:authentication-manager>

</beans>
```



Login with Username and Password

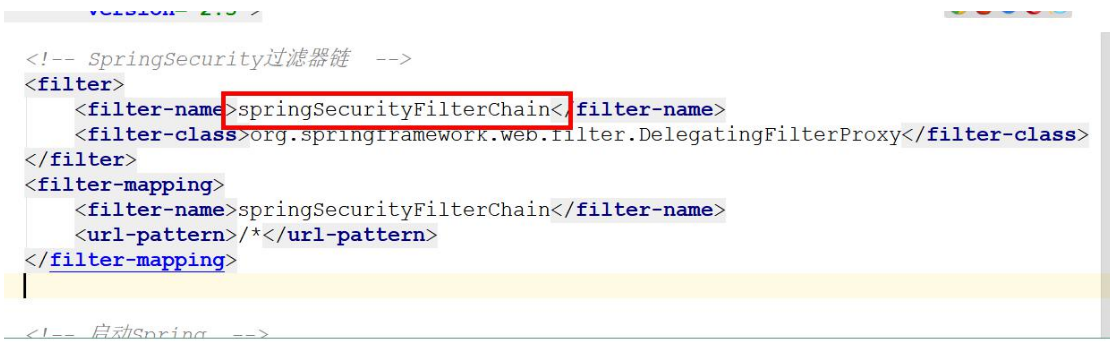
User:

Password:

Login

6. Spring Security 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>
```

7. 自定义登录页面，自定义登录请求

7.1. 自定义登录页面

```
<?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>: 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/>-->
```

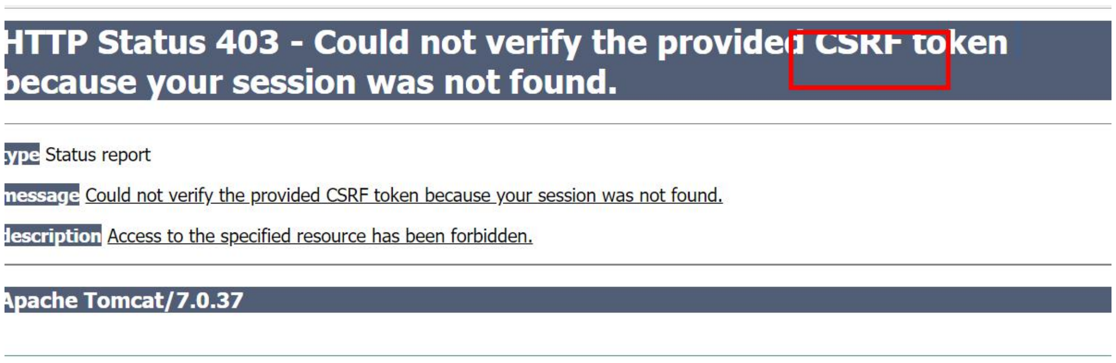
```
        <!-- login-page: 自定义登录页面 -->

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


    </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="123456" authorities="ROLE_USER"/>

            </security:user-service>

        </security:authentication-provider>

    </security:authentication-manager>

</beans>
```

7.2. 自定义登录请求

```
<?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>: 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 login-page="/userLogin"
```

```
login-processing-url="/securityLogin"/>


     <!-- 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="123456"
authorities="ROLE_USER"/>
            </security:user-service>
        </security:authentication-provider>
    </security:authentication-manager>
</beans>
```



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

8. user-service 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-4.2.xsd
          http://www.springframework.org/schema/security

http://www.springframework.org/schema/security/spring-security-4.2.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
                    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 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="123456"
authorities="ROLE_USER"/>

                <security:user name="jack" password="123456"
authorities="ROLE_ADMIN"/>

            </security:user-service>

        </security:authentication-provider>

    </security:authentication-manager>

</beans>
```

9. 自定义 UserDetailService 类实现用户权限访问控制

关键：使用 UserDetailService 接口

9.1. 创建 UserDetailService 接口实现类

```java
package cn.sm1234.security;



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.UserDetailsService;
```

```
import

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



/**

 * @author http://www.sm1234.cn

 * @version 1.0

 * @description cn.sm1234.security

 * @date 18/4/13

 */

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( "eric","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

9.2. 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-4.2.xsd
            http://www.springframework.org/schema/security

http://www.springframework.org/schema/security/spring-security-4.2.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
                    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 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:user-service>
                &lt;!&ndash;<security:user name="eric" password="123456"
authorities="ROLE_USER"/>
                <security:user name="jack" password="123456"
authorities="ROLE_ADMIN"/>&ndash;&gt;
            </security:user-service>-->
        </security:authentication-provider>
    </security:authentication-manager>

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

</beans>
```

10. 自定义登录成功与失败处理逻辑

关键：

1）登录成功处理：AuthenticationSuccessHandler

2）登录失败处理：AuthenticationFailureHandler

10.1. 自定义登录成功逻辑

```
package cn.sm1234.security;


import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.beans.factory.annotation.Autowired;
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 18/4/13
 */
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("succcess",true);

        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>: 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 login-page="/userLogin"

login-processing-url="/securityLogin" default-target-url="/product/index"

authentication-success-handler-ref="myAuthenticationSuccessHandler"/>


        <!-- 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:user-service>
```

```
              &lt;!&ndash;<security:user name="eric" password="123456"
authorities="ROLE_USER"/>
              <security:user name="jack" password="123456"
authorities="ROLE_ADMIN"/>&ndash;&gt;
          </security:user-service>-->
       </security:authentication-provider>
    </security:authentication-manager>

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

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

10.2. 自定义登录失败逻辑

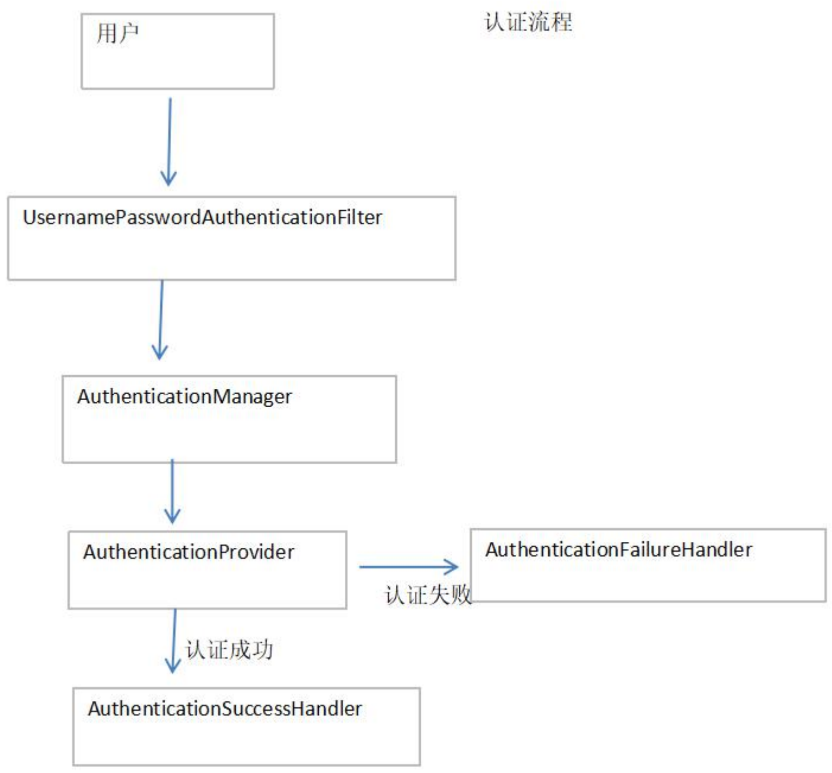
```
package cn.sm1234.security;

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 18/4/13

 */

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);

    }

}
```

```
<?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>: 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 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:user-service>
                &lt;!&ndash;<security:user name="eric" password="123456"
authorities="ROLE_USER"/>
                <security:user name="jack" password="123456"
authorities="ROLE_ADMIN"/>&ndash;&gt;
            </security:user-service>-->
        </security:authentication-provider>
    </security:authentication-manager>

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


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


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

11. 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