品优购电商系统开发

第 4 章

安全框架与商家入驻审核

传智播客.黑马程序员

课程目标

目标 1：完成商品分类管理

目标 2：完成商家申请入驻、商家审核、商家登陆等功能

1.Spring Security 框架入门

1.1 Spring Security 简介

Spring Security 是一个能够为基于 Spring 的企业应用系统提供声明式的安全访问控制解决方案的安全框架。它提供了一组可以在 Spring 应用上下文中配置的 Bean，充分利用了 Spring IoC，DI（控制反转 Inversion of Control ,DI:Dependency Injection 依赖注入）和 AOP（面向切面编程）功能，为应用系统提供声明式的安全访问控制功能，减少了为企业系统安全控制编写大量重复代码的工作

1.2 Spring Security 入门小 Demo

1.2.1 最简单 Demo

（1）创建工程 spring-security-demo ,pom.xml 内容

```
<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/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.itcast.demo</groupId>

    <artifactId>spring-security-demo</artifactId>

    <packaging>war</packaging>

    <version>0.0.1-SNAPSHOT</version>

    <properties>
```

```
        <spring.version>4.2.4.RELEASE</spring.version>
    </properties>
    <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>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
            <version>${spring.version}</version>
        </dependency>
```

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-web</artifactId>
            <version>4.1.0.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-config</artifactId>
            <version>4.1.0.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
```

```
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
      <plugins>
        <!-- java 编译插件 -->
         <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.2</version>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                    <encoding>UTF-8</encoding>
                </configuration>
         </plugin>
        <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <configuration>
                    <!-- 指定端口 -->
                    <port>9090</port>
```

```
                    <!-- 请求路径 -->
                    <path>/</path>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

（2）创建 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">
  <servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class
>
    <!-- 指定加载的配置文件 ，通过参数 contextConfigLocation 加载-->
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:springmvc-servlet.xml</param-value>
    </init-param>
  </servlet>
```

```
  <servlet-mapping>

    <servlet-name>springmvc</servlet-name>

    <url-pattern>*.do</url-pattern>

  </servlet-mapping>

     <context-param>

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

        <param-value>classpath:spring-security.xml</param-value>

     </context-param>

     <listener>

        <listener-class>

            org.springframework.web.context.ContextLoaderListener

        </listener-class>

     </listener>

     <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）创建 spring 配置文件 springmvc-servlet.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:p="http://www.springframework.org/schema/p"

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

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

    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>
```

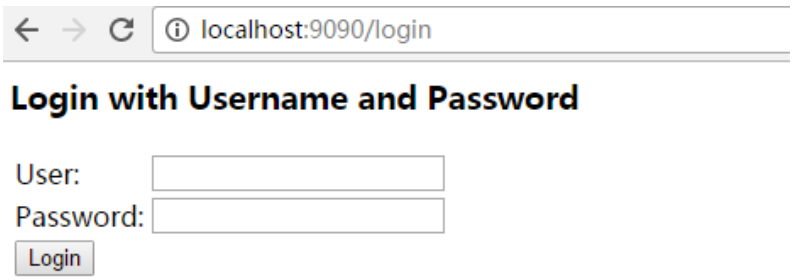
（4）创建 index.html　　内容略

（5）创建 spring 配置文件 spring-security.xml

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

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

    xmlns:bean="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.xsd

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

         <http>

         <intercept-url pattern="/*" access="hasRole('ROLE_USER')" />

         <form-login/>
```

```
    </http>

    <!-- 认证管理器 -->

    <authentication-manager>

        <authentication-provider>

            <user-service>

                <user authorities="ROLE_USER

" name="guest" password="guest" />

            </user-service>

        </authentication-provider>

    </authentication-manager>

</bean:beans>
```

此案例我们没有登录页，而是使用了系统自动生成的登陆页，效果如下：



1.2.2 用户自定义登录页

实际开发中，我们不可能使用系统生成的登录页，而是使用我们自己的登录页。

（1）构建登陆页：

```
<!DOCTYPE html>

<html>

<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>登陆</title>
</head>
<body>
    <form action='/login' method='POST'>
        <table>
            <tr>
                <td>用户名:</td>
                <td><input type='text' name='username' value=''></td>
            </tr>
            <tr>
                <td>密码:</td>
                <td><input type='password' name='password' /></td>
            </tr>
            <tr>
                <td colspan='2'><input name="submit" type="submit"
                    value="登陆" /></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

（2）构建登陆失败页　login_error.html（内容略）

（3）添加 favicon.ico 到根目录

（4）修改 spring 配置文件 spring-security.xml

```
    <http pattern="/login_error.html" security="none"></http>

    <http pattern="/login.html" security="none"></http>

    <http pattern="/*.ico" security="none"></http>

    <http>

        <intercept-url pattern="/*" access="hasRole('ROLE_USER')" />

        <form-login login-page="/login.html" default-target-url="/index.html"
authentication-failure-url="/login_error.html"/>

        <csrf disabled="true"/>

    </http>
```

security="none"　设置此资源不被拦截.

login-page：指定登录页面。
authentication-failure-url：指定了身份验证失败时跳转到的页面。
default-target-url：指定了成功进行身份验证和授权后默认呈现给用户的页面。

csrf disabled="true"　关闭 csrf ,如果不加会出现错误

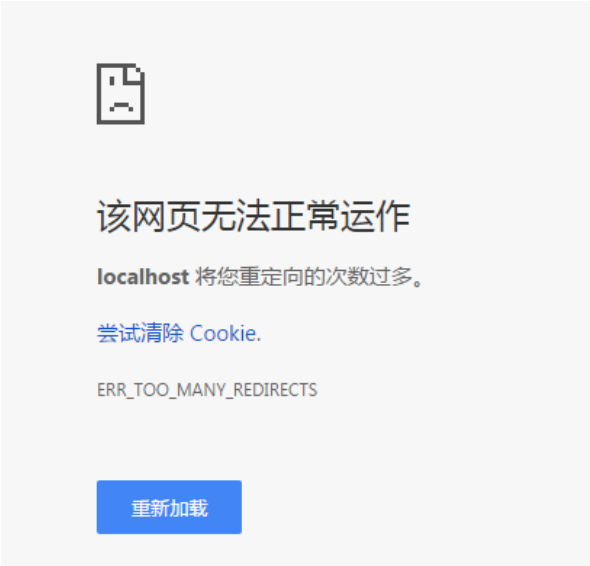
HTTP Status 403 - Could not verify the provided CSRF token because your session was not found.

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/7.0.47

CSRF（Cross-site request forgery）跨站请求伪造，也被称为“One Click Attack”或者 Session Riding，通常缩写为 CSRF 或者 XSRF，是一种对网站的恶意利用。

如果你没有设置登录页 security="none"　，将会出现以下错误



1.2.3 always-use-default-target

always-use-default-target：指定了是否在身份验证通过后总是跳转到 default-target-url 属性指定的 URL。

2.运营商系统登录与安全控制

2.1 需求分析

完成运营商登陆功能

2.2 登陆功能的实现

2.2.1 配置文件

（1）修改 pinyougou-manager-web 的 pom.xml ，添加依赖

```
<!-- 身份验证 -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
</dependency>
```

（2）修改 web.xml

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring/spring-security.xml</param-value>
</context-param>
<listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
  <filter>
```

```
        <!-- 注意：过滤器的名称必须是 springSecurityFilterChain -->
            <filter-name>springSecurityFilterChain</filter-name>
            <filter-class>org.springframework.web.filter.DelegatingFilterProxy</fil
        ter-class>

    </filter>

    <filter-mapping>

        <filter-name>springSecurityFilterChain</filter-name>

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

    </filter-mapping>
```

（3）pinyougou-manager-web 的 spring 目录下添加配置文件 spring-security.xml

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

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

    xmlns:bean="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.xsd

                        http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-4.1.xsd">

    <http pattern="/login.html" security="none"></http>

    <http pattern="/loginerror.html" security="none"></http>

    <http pattern="/css/**" security="none"></http>

    <http pattern="/img/**" security="none"></http>

    <http pattern="/js/**" security="none"></http>

    <http pattern="/plugins/**" security="none"></http>

    <http>

        <intercept-url pattern="/**" access="hasRole('ROLE_USER')" />
```

```
        <form-login login-page="/login.html" login-processing-url="/login"
always-use-default-target="true"

            default-target-url="/admin/index.html"
authentication-failure-url="/loginerror.html" />

            <csrf disabled="true"/>

        <headers>

            <frame-options policy="SAMEORIGIN"/>

        </headers>

    </http>

    <authentication-manager>

        <authentication-provider>

            <user-service>

                <user authorities="ROLE_USER" name="admin" password="123456" />

            </user-service>

        </authentication-provider>

    </authentication-manager>

</bean:beans>
```

2.2.2 登录页面

修改 pinyougou-manager-web 的 login.html

```
<form id="loginform" action="/login" method="post"  class="sui-form">

    <div class="input-prepend"><span class="add-on loginname"></span>

        <input id="prependedInput"  name="username" type="text" placeholder="邮箱/用
户名/手机号" class="span2 input-xfat">

    </div>
```

```
    <div class="input-prepend"><span class="add-on loginpwd"></span>

        <input id="prependedInput"  name="password" type="password" placeholder="请输入密码" class="span2 input-xfat">

    </div>

    <div class="setting">

        <div id="slider">

            <div id="slider_bg"></div>

                <span id="label">>>></span> <span id="labelTip">拖动滑块验证</span>

            </div>

        </div>

        <div class="logined">

        <a class="sui-btn btn-block btn-xlarge btn-danger" onclick="document:loginform.submit()" target="_blank">登  录</a>

        </div>

</form>
```

参照 login.html 创建 loginerror.html 页面

2.3 主界面显示登陆人

2.3.1 后端代码

在 pinyougou-manager-web 新建 LoginController.java

```
package com.pinyougou.manager.controller;

import java.util.HashMap;

import java.util.Map;

import org.springframework.security.core.context.SecurityContextHolder;
```

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

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

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/login")

public class LoginController {

    @RequestMapping("name")

    public Map name(){

        String name= (UserDetails)
SecurityContextHolder.getContext().getAuthentication().getName();

        Map map=new HashMap();

        map.put("loginName", name);

        return map ;

    }

}
```

2.3.2 前端代码

（1）新建 loginService.js

```
//登陆服务层

app.service('loginService',function($http){

    //读取登录人名称

    this.loginName=function(){

        return $http.get('../login/name.do');

    }
```

```
});
```

（2）新建 indexController.js

```
app.controller('indexController' ,function($scope,$controller   ,loginService){
    //读取当前登录人
    $scope.showLoginName=function(){
        loginService.loginName().success(
            function(response){
                $scope.loginName=response.loginName;
            }
        );
    }
});
```

页面上引入 JS，用表达式显示（代码略）

2.4 退出登录

在 pinyougou-manager-web 的 spring-security.xml 的 http 节点中添加配置

```
<logout/>
```

加此配置后，会自动的产生退出登录的地址/logout,如果你不想用这个地址 ，你也可以定义生成的退出地址以及跳转的页面，配置如下

```
<logout logout-url="" logout-success-url=""/>
```

logout-url:退出的地址，会自动生成

logout-success-url:退出后跳转的地址

修改注销的链接

```
<div class="pull-right">

    <a href="../logout" class="btn btn-default btn-flat">注销</a>

</div>
```

3.商家申请入驻[注册]

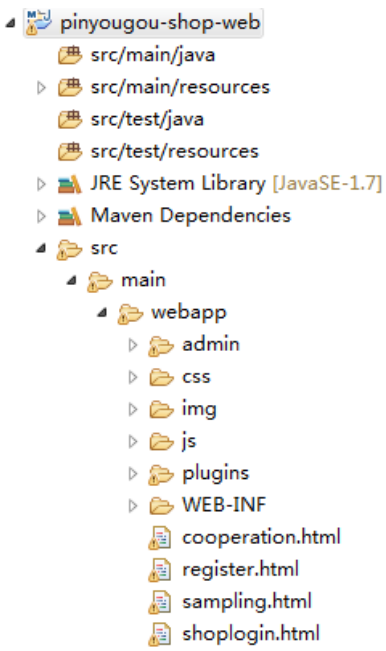
3.1 需求分析

商家申请入驻，需要填写商家相关的信息。待运营商平台审核通过后即可使用使用。

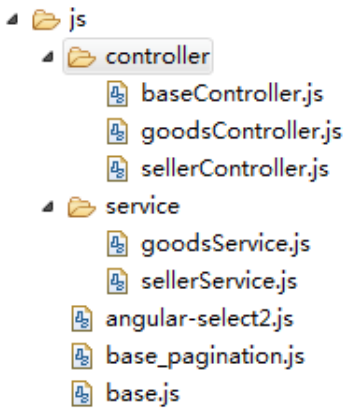
3.2 准备工作

（1）拷贝资源： 将“资源/静态原型/商家管理后台”下的页面拷贝到 pinyougou-shop-web 工程

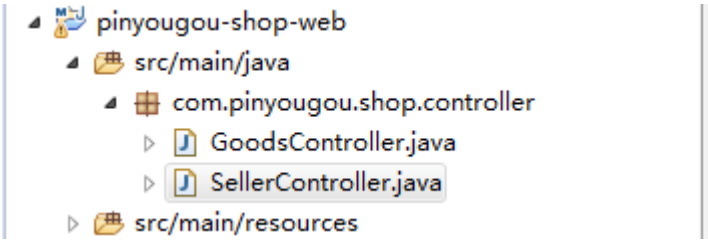
admin	2017/7/29 20:47	文件夹	
css	2017/8/2 21:57	文件夹	
img	2017/7/29 21:02	文件夹	
js	2017/7/29 18:42	文件夹	
plugins	2017/7/31 9:51	文件夹	
cooperation.html	2017/8/2 21:58	Chrome HTML D...	10 KB
register.html	2017/8/2 21:58	Chrome HTML D...	7 KB
sampling.html	2017/7/29 22:03	Chrome HTML D...	18 KB
shoplogin.html	2017/7/29 21:03	Chrome HTML D...	4 KB



（2）参照“运营商后台”构建 js



（3）拷贝后端控制层代码



3.3 前端代码

修改 register.html 引入 JS

```
<script type="text/javascript" src="plugins/angularjs/angular.min.js">  </script>

<script type="text/javascript" src="js/base.js">  </script>

<script type="text/javascript" src="js/service/sellerService.js">  </script>

<script type="text/javascript" src="js/controller/baseController.js">  </script>

<script type="text/javascript" src="js/controller/sellerController.js">  </script>
```

指令

```
<body ng-app="pinyougou" ng-controller="sellerController">
```

绑定表单（部分代码）

```
<div class="control-group">

        <label class="control-label">登陆名（不可修改）：</label>

        <div class="controls">

        <input type="text" ng-model="entity.sellerId" placeholder="登陆名"
class="input-xfat input-xlarge">

        </div>

</div>

<div class="control-group">

    <label class="control-label">登陆密码：</label>

    <div class="controls">

        <input type="password" ng-model="entity.password" placeholder="登陆密码"
class="input-xfat input-xlarge">

    </div>

</div>
```

修改 sellerController.js ，在保存成功后跳转到登陆页

```
    //保存
```

```
$scope.add=function(){
    sellerService.add( $scope.entity  ).success(
        function(response){
            if(response.success){
                location.href='shoplogin.html';
            }else{
                alert(response.message);
            }
        }
    );
}
```

绑定“申请入驻”按钮

```
<a class="sui-btn btn-block btn-xlarge btn-danger" ng-click="add()" target="_blank">
申请入驻</a>
```

3.4 后端代码

修改后端代码，设置默认状态为 0,也可以使用 insertSelctive 进行保存.

4.商家审核

4.1 需求分析

商家申请入驻后，需要网站运营人员在运营商后台进行审核，审核后商家才可以登陆系统。

状态值： 0：未审核 1：已审核 2：审核未通过 3：关闭

4.2 商家待审核列表

修改 seller_1.html

引入 JS

```
<script type="text/javascript" src="../plugins/angularjs/angular.min.js">  </script>

<!-- 分页组件开始 -->

<script src="../plugins/angularjs/pagination.js"></script>

<link rel="stylesheet" href="../plugins/angularjs/pagination.css">

<!-- 分页组件结束 -->

<script type="text/javascript" src="../js/base_pagination.js">  </script>

<script type="text/javascript" src="../js/service/sellerService.js">  </script>

<script type="text/javascript" src="../js/controller/baseController.js">  </script>

<script type="text/javascript" src="../js/controller/sellerController.js">  </script>
```

指令

```
<body    class="hold-transition    skin-red    sidebar-mini"    ng-app="pinyougou"
ng-controller="sellerController" ng-init="searchEntity={status:'0'}">
```

加入分页控件

```
   <tm-pagination conf="paginationConf"></tm-pagination>
```

循环

```
<tr ng-repeat="entity in list">

    <td><input  type="checkbox"></td>

    <td>{{entity.sellerId}}</td>

    <td>{{entity.name}}</td>

    <td>{{entity.nickName}}</td>
```

```
    <td>{{entity.linkmanName}}</td>

    <td>{{entity.telephone}}</td>

    <td class="text-center">

    <button type="button" class="btn bg-olive btn-xs" data-toggle="modal"
data-target="#sellerModal" >详情</button>

    </td>

</tr>
```

4.3 商家详情

（1）绑定页面弹出窗口

```
<table class="table table-bordered table-striped"  width="800px">

        <tr>

            <td>公司名称</td>

            <td>{{entity.name}}</td>

        </tr>

        <tr>

            <td>公司手机</td>

            <td>{{entity.mobile}}</td>
```

```
        </tr>
        <tr>
            <td>公司电话</td>
            <td>{{entity.telephone}}</td>
        </tr>
        <tr>
             <td>公司详细地址</td>
             <td>{{entity.addressDetail}}</td>
        </tr>
</table>
```

（2）列表的“详情”按钮

```
<button type="button" class="btn bg-olive btn-xs" data-toggle="modal" data-target="#sellerModal" ng-click="findOne(entity.sellerId)">详情</button>
```

4.4 商家审核

4.4.1 后端代码

（1）在 pinyougou-sellergoods-interface 工程的 SellerService.java 服务接口新增方法定义

```
    /**
     * 更改状态
     * @param id
     * @param status
     */
    public void updateStatus(String sellerId,String status);
```

（2）在 pinyougou-sellergoods-service 的 SellerServiceImpl.java 新增方法

```
@Override
    public void updateStatus(String sellerId, String status) {
        TbSeller seller = sellerMapper.selectByPrimaryKey(sellerId);
        seller.setStatus(status);
        sellerMapper.updateByPrimaryKey(seller);
    }
```

（3）在 pinyougou-manager-web 的 SellerController.java 新增方法

```
/**
 * 更改状态
 * @param sellerId 商家 ID
 * @param status 状态
 */
@RequestMapping("/updateStatus")
public Result updateStatus(String sellerId, String status){
    try {
        sellerService.updateStatus(sellerId, status);
        return new Result(true, "成功");
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        return new Result(false, "失败");
```

```
        }

    }
```

4.4.2 前端代码

修改 pinyougou-manager-web 的 sellerService.js

```
    //更改状态

    this.updateStatus=function(sellerId,status){

        return
$http.get('../seller/updateStatus.do?sellerId='+sellerId+'&status='+status);

    }
```

修改 pinyougou-manager-web 的 sellerController.js

```
    $scope.updateStatus=function(sellerId,status){

        sellerService.updateStatus(sellerId,status).success(

            function(response){

                if(response.success){

                    $scope.reloadList();//刷新列表

                }else{

                    alert("失败");

                }

            }

        );

    }
```

修改按钮，调用方法

```
        <div class="modal-footer">
```

```
            <button class="btn btn-success" data-dismiss="modal" aria-hidden="true"
ng-click="updateStatus(entity.sellerId,'1')">审核通过</button>

        <button class="btn btn-danger"  data-dismiss="modal" aria-hidden="true"
ng-click="updateStatus(entity.sellerId,'2')">审核未通过</button>

          <button class="btn btn-danger" data-dismiss="modal" aria-hidden="true"
ng-click="updateStatus(entity.sellerId,'3')">关闭商家</button>

            <button class="btn btn-default" data-dismiss="modal" aria-hidden="true">
关闭</button>

</div>
```

5.商家系统登录与安全控制

5.1 需求分析

5.2 自定义认证类

（1）pom.xml、web.xml 参照运营商管理后台

（2）在在 pinyougou-shop-web 创建 com.pinyougou.service 包，包下创建类 UserDetailsServiceImpl.java 实现 UserDetailsService 接口

```
package com.pinyougou.service;

import java.util.ArrayList;

import java.util.List;

import org.springframework.security.core.GrantedAuthority;

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

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;

/**

 * 认证类

 * @author Administrator

 *

 */

public class UserDetailsServiceImpl implements UserDetailsService {

    @Override

    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

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

       grantedAuths.add(new SimpleGrantedAuthority("ROLE_USER"));

       return new User(username,"123456", grantedAuths);

    }

}
```

（3）在 pinyougou-shop-web 的 spring 目录下创建 spring-security.xml

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

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

    xmlns:bean="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.xsd

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

```
http://www.springframework.org/schema/security/spring-security-4.1.xsd">

    <http pattern="/shoplogin.html" security="none"></http>

    <http pattern="/css/**" security="none"></http>

    <http pattern="/img/**" security="none"></http>

    <http pattern="/js/**" security="none"></http>

    <http pattern="/plugins/**" security="none"></http>

    <http pattern="/seller/add.do" security="none"></http>



    <http>

        <intercept-url pattern="/*" access="hasRole('ROLE_USER')" />

        <form-login login-page="/shoplogin.html" login-processing-url="/login"
always-use-default-target="true"

            default-target-url="/admin/index.html"
authentication-failure-url="/loginerror.html" />

            <csrf disabled="true"/>

        <headers>

            <frame-options policy="SAMEORIGIN"/>

        </headers>

    </http>

    <!-- 认证管理器 -->

    <authentication-manager alias="authenticationManager">

       <authentication-provider user-service-ref='userDetailService'>

       </authentication-provider>

    </authentication-manager>
```

```
    <!-- 认证类 -->

    <bean:bean id="userDetailService"
class="com.pinyougou.service.UserDetailsServiceImpl">

    </bean:bean>

</bean:beans>
```

经过上述配置，用户在输入密码 123456 时就会通过（用户名随意）

5.3 认证类调用服务方法

修改 UserDetailsServiceImpl.java ，添加属性和 getter setter 方法 ，修改 loadUserByUsername 方法

```
    private SellerService sellerService;

    public SellerService getSellerService() {

        return sellerService;

    }

    public void setSellerService(SellerService sellerService) {

        this.sellerService = sellerService;

    }

    @Override

    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

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

       grantedAuths.add(new SimpleGrantedAuthority("ROLE_USER"));

       TbSeller seller = sellerService.findOne(username);

       if(seller!=null){

            return new User(username,seller.getPassword(), grantedAuths);
```

```
        }else{
            return null;
        }
    }
```

修改 pinyougou-shop-web 的 spring-security.xml ，添加如下配置

```
  <!-- 引用 dubbo 服务 -->
    <dubbo:application name="pinyougou-shop-web" />
    <dubbo:registry address="zookeeper://192.168.25.129:2181"/>
    <dubbo:reference id="sellerService"
interface="com.pinyougou.sellergoods.service.SellerService" >
    </dubbo:reference>
    <bean:bean id="userDetailService"
class="com.pinyougou.service.UserDetailsServiceImpl">
        <bean:property name="sellerService" ref="sellerService"></bean:property>
    </bean:bean>
```

经过上述修改后，在登陆页输入用户名和密码与数据库一致即可登陆 。

5.4 密码加密

5.4.1 BCrypt 加密算法

用户表的密码通常使用 MD5 等不可逆算法加密后存储，为防止彩虹表破解更会先使用一个特定的字符串（如域名）加密，然后再使用一个随机的 salt（盐值）加密。 特定字符串是程序代码中固定的，salt 是每个密码单独随机，一般给用户表加一个字段单独存储，比较麻烦。 BCrypt 算法将 salt 随机并混入最终加密后的密码，验证时也无需单独提供之前的 salt，从而无需单独处理 salt 问题。

5.4.2 商家入驻密码加密

商家申请入驻的密码要使用 BCrypt 算法进行加密存储，修改 SellerController.java 的 add 方法

```
/**
     * 增加
     * @param seller
     * @return
     */
    @RequestMapping("/add")
    public Result add(@RequestBody TbSeller seller){
        //密码加密
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        String password = passwordEncoder.encode(seller.getPassword());
        seller.setPassword(password);
        try {
            sellerService.add(seller);
            return new Result(true, "增加成功");
        } catch (Exception e) {
            e.printStackTrace();
            return new Result(false, "增加失败");
        }
    }
```

5.4.3 加密配置

修改 pinyougou-shop-web 的 spring-security.xml ，添加如下配置

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

修改认证管理器的配置

```
<!-- 认证管理器 -->

<authentication-manager alias="authenticationManager">

   <authentication-provider user-service-ref='userDetailService'>

         <password-encoder ref="bcryptEncoder"></password-encoder>

   </authentication-provider>

</authentication-manager>
```

5.5 显示登录名

参照运营商后台

5.6 退出登录

参照运营商后台