Shiro入门Demo

一、shiro简介

Apache Shiro是一个强大易用的Java安全框架，提供了认证、授权、加密和会话管理等功能：

认证 - 用户身份识别，常被称为用户“登录”；

授权 - 访问控制；

密码加密 - 保护或隐藏数据防止被偷窥；

会话管理 - 每用户相关的时间敏感的状态。

对于任何一个应用程序，Shiro都可以提供全面的安全管理服务。

相对于其他安全框架（比如spring-security），Shiro要简单的多。

二、Shiro的架构介绍

首先，来了解一下Shiro的三个核心组件：Subject, SecurityManager 和 Realms。



（1）Subject：即“当前操作用户”。但是，在Shiro中，Subject这一概念并不仅仅指人，也可以是第三方进程、后台帐户（Daemon Account）或其他类似事物。它仅仅意味着“当前跟软件交互的东西”。但考虑到大多数目的和用途，你可以把它认为是Shiro的“用户”概念。 Subject代表了当前用户的安全操作，SecurityManager则管理所有用户的安全操作。

（2）SecurityManager：它是Shiro框架的核心，典型的Facade模式，Shiro通过SecurityManager来管理内部组件实例，并通过它来提供安全管理的各种服务。

（3）Realm： Realm充当了Shiro与应用安全数据间的“桥梁”或者“连接器”。也就是说，当对用户执行认证（登录）和授权（访问控制）验证时，Shiro会从应用配置的Realm中查找用户及其权限信息。从这个意义上讲，Realm实质上是一个安全相关的DAO：它封装了数据源的连接细节，并在需要时将相关数据提供给Shiro。当配置Shiro时，你必须至少指定一个Realm，用于认证和（或）授权。配置多个Realm是可以的，但是至少需要一个。 Shiro内置了可以连接大量安全数据源（又名目录）的Realm，如LDAP、关系数据库（JDBC）、类似INI的文本配置资源以及属性文件等。如果缺省的Realm不能满足需求，你还可以插入代表自定义数据源的自己的Realm实现。

除前文所讲Subject、SecurityManager 、Realm三个核心组件外，Shiro主要组件还包括：



（4）Authenticator ：认证就是核实用户身份的过程。这个过程的常见例子是大家都熟悉的“用户/密码”组合。多数用户在登录软件系统时，通常提供自己的用户名（当事人）和支持他们的密码（证书）。如果存储在系统中的密码（或密码表示）与用户提供的匹配，他们就被认为通过认证。

（5）Authorizer ：授权实质上就是访问控制 - 控制用户能够访问应用中的哪些内容，比如资源、Web页面等等。

（6）SessionManager ：在安全框架领域，Apache Shiro提供了一些独特的东西：可在任何应用或架构层一致地使用Session API。即，Shiro为任何应用提供了一个会话编程范式 - 从小型后台独立应用到大型集群Web应用。这意味着，那些希望使用会话的应用开发者，不必被迫使用Servlet或EJB容器了。或者，如果正在使用这些容器，开发者现在也可以选择使用在任何层统一一致的会话API，取代Servlet或EJB机制。

（7）CacheManager :对Shiro的其他组件提供缓存支持。

三、项目代码

1 pom.xml中添加与shiro相关的jar包

```

<!-- Spring 整合Shiro需要的依赖 -->

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-core</artifactId>

<version>1.2.1</version>

</dependency>

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-web</artifactId>

<version>1.2.1</version>

</dependency>

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-ehcache</artifactId>

<version>1.2.1</version>

</dependency>

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-spring</artifactId>

<version>1.2.1</version>

</dependency>

```

2 web.xml中添加与shiro相关的配置

```

<!-- shiro过滤器 -->

<filter>

<filter-name>shiroFilter</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

<init-param>

<param-name>targetFilterLifecycle</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>shiroFilter</filter-name>

<url-pattern>\*.html</url-pattern>

<url-pattern>\*.json</url-pattern>

</filter-mapping>

```

3 spring-mvc.xml

```

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

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

xmlns:p=*"http://www.springframework.org/schema/p"*

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

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-3.2.xsd*

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

*http://www.springframework.org/schema/context/spring-context-3.2.xsd*

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

*http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd"* default-lazy-init=*"true"*>

<!-- 扫描controller（controller层注入） -->

<context:component-scan base-package=*"com.z.controller"*/>

<!-- 对模型视图添加前后缀 -->

<bean id=*"viewResolver"* class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*

p:prefix=*"/WEB-INF/view/"* p:suffix=*".jsp"*/>

</beans>

```

4 spring-shiro.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-3.0.xsd"*

default-lazy-init=*"true"*>

<description>Shiro Configuration</description>

<!-- Shiro's main business-tier object for web-enabled applications -->

<bean id=*"securityManager"* class=*"org.apache.shiro.web.mgt.DefaultWebSecurityManager"*>

<property name=*"realm"* ref=*"myShiroRealm"* />

<property name=*"cacheManager"* ref=*"cacheManager"* />

</bean>

<!-- 項目自定义的Realm -->

<bean id=*"myShiroRealm"* class=*"com.z.shiro.realm.ShiroRealm"*>

<property name=*"cacheManager"* ref=*"cacheManager"* />

</bean>

<!-- Shiro Filter -->

<bean id=*"shiroFilter"* class=*"org.apache.shiro.spring.web.ShiroFilterFactoryBean"*>

<property name=*"securityManager"* ref=*"securityManager"* />

<property name=*"loginUrl"* value=*"/login.html"* />

<property name=*"successUrl"* value=*"/loginsuccess.html"* />

<property name=*"unauthorizedUrl"* value=*"/error.html"* />

<property name=*"filterChainDefinitions"*>

<value>

/checkLogin.json = anon

/\*\* = authc

</value>

</property>

</bean>

<!-- 用户授权信息Cache -->

<bean id=*"cacheManager"* class=*"org.apache.shiro.cache.MemoryConstrainedCacheManager"* />

<!-- 保证实现了Shiro内部lifecycle函数的bean执行 -->

<bean id=*"lifecycleBeanPostProcessor"* class=*"org.apache.shiro.spring.LifecycleBeanPostProcessor"* />

<!-- AOP式方法级权限检查 -->

<bean class=*"org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator"*

depends-on=*"lifecycleBeanPostProcessor"*>

<property name=*"proxyTargetClass"* value=*"true"* />

</bean>

<bean class=*"org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor"*>

<property name=*"securityManager"* ref=*"securityManager"* />

</bean>

</beans>

```

这个文件中的anon表示不需要验证，authc表示需要验证。

/checkLogin.json = anon表示checkLogin.json不需要验证。

/\*\* = authc表示所有的请求（除checkLogin.json外）都需要验证。

5 java代码

（1）controller

```

**package** com.z.controller;

**import** java.util.HashMap;

**import** java.util.Map;

**import** javax.servlet.http.HttpServletRequest;

**import** org.apache.shiro.SecurityUtils;

**import** org.apache.shiro.authc.UsernamePasswordToken;

**import** org.apache.shiro.authz.annotation.RequiresRoles;

**import** org.apache.shiro.subject.Subject;

**import** org.springframework.stereotype.Controller;

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

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.ResponseBody;

**import** org.springframework.web.servlet.ModelAndView;

**import** com.alibaba.druid.support.json.JSONUtils;

**import** com.z.util.EncryptUtil;

@Controller

**public** **class** IndexController {

@RequestMapping("/index.html")

**public** ModelAndView getIndex(HttpServletRequest request) **throws** Exception {

ModelAndView mav = **new** ModelAndView("index");

**return** mav;

}

@RequestMapping("/login.html")

**public** ModelAndView login() **throws** Exception {

ModelAndView mav = **new** ModelAndView("login");

**return** mav;

}

@RequestMapping("/loginsuccess.html")

**public** ModelAndView loginsuccess() **throws** Exception {

ModelAndView mav = **new** ModelAndView("loginsuccess");

**return** mav;

}

@RequestMapping(value = "newpage1.html")

**public** String shownewpage1() {

Subject currentUser = SecurityUtils.*getSubject*();

**if**(currentUser.hasRole("administrator")){

**return** "newPage";

}**else**{

**return** "pagenofound";

}

}

@RequestMapping("/newpage2.html")

**public** String shownewpage2() {

Subject currentUser = SecurityUtils.*getSubject*();

**if**(currentUser.isPermitted("newPage2.html")){

**return** "newPage";

}**else**{

**return** "pagenofound";

}

}

@RequestMapping("/error.html")

**public** String error() {

**return** "error";

}

@RequestMapping(value="/checkLogin.json",method=RequestMethod.*POST*)

@ResponseBody

@RequiresRoles("admin")

**public** String checkLogin(String username,String password) {

Map<String, Object> result = **new** HashMap<String, Object>();

**try**{

UsernamePasswordToken token = **new** UsernamePasswordToken(username, EncryptUtil.*encryptMD5*(password));

Subject currentUser = SecurityUtils.*getSubject*();

**if** (!currentUser.isAuthenticated()){

token.setRememberMe(**true**);

currentUser.login(token);//验证角色和权限

}

}**catch**(Exception ex){

ex.printStackTrace();

}

result.put("success", **true**);

**return** JSONUtils.*toJSONString*(result);

}

@RequestMapping(value="/logout.json",method=RequestMethod.*POST*)

@ResponseBody

**public** String logout() {

Map<String, Object> result = **new** HashMap<String, Object>();

result.put("success", **true**);

Subject currentUser = SecurityUtils.*getSubject*();

currentUser.logout();

**return** JSONUtils.*toJSONString*(result);

}

@RequestMapping(value="/newpage1.json",method=RequestMethod.*POST*)

@ResponseBody

**public** String newpage1() {

Map<String, Object> result = **new** HashMap<String, Object>();

result.put("success", **true**);

**return** JSONUtils.*toJSONString*(result);

}

@RequestMapping(value="/newpage2.json",method=RequestMethod.*POST*)

@ResponseBody

**public** String newpage2() {

Map<String, Object> result = **new** HashMap<String, Object>();

result.put("success", **true**);

**return** JSONUtils.*toJSONString*(result);

}

}

```

（2）Realm

```

**package** com.z.shiro.realm;

**import** java.util.HashSet;

**import** java.util.Set;

**import** org.apache.shiro.authc.AuthenticationException;

**import** org.apache.shiro.authc.AuthenticationInfo;

**import** org.apache.shiro.authc.AuthenticationToken;

**import** org.apache.shiro.authc.SimpleAuthenticationInfo;

**import** org.apache.shiro.authc.UsernamePasswordToken;

**import** org.apache.shiro.authz.AuthorizationInfo;

**import** org.apache.shiro.authz.SimpleAuthorizationInfo;

**import** org.apache.shiro.realm.AuthorizingRealm;

**import** org.apache.shiro.subject.PrincipalCollection;

**import** com.z.util.EncryptUtil;

**public** **class** ShiroRealm **extends** AuthorizingRealm {

**private** **static** **final** String *USERNAME* = "admin";

**private** **static** **final** String *PASSWORD* = "123456";

@Override

**protected** AuthenticationInfo doGetAuthenticationInfo(

AuthenticationToken authcToken) **throws** AuthenticationException {

UsernamePasswordToken token = (UsernamePasswordToken) authcToken;

**if**(token.getUsername().equals(*USERNAME*)){

**return** **new** SimpleAuthenticationInfo(*USERNAME*, EncryptUtil.*encryptMD5*(*PASSWORD*), getName());

}**else**{

**throw** **new** AuthenticationException();

}

}

@Override

**protected** AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {

Set<String> roleNames = **new** HashSet<String>();

Set<String> permissions = **new** HashSet<String>();

roleNames.add("administrator");

permissions.add("newPage.html");

SimpleAuthorizationInfo info = **new** SimpleAuthorizationInfo(roleNames);

info.setStringPermissions(permissions);

**return** info;

}

}

```

注意，controller中的currentUser.hasRole("administrator")和currentUser.isPermitted("newPage2.html")会调用Realm中的doGetAuthorizationInfo()。

6 jsp代码

（1）index.jsp

```

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

</head>

<body>

This is my JSP page. <br>

</body>

</html>

```

（2）login.jsp

```

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<script src=*"*<%=request.getContextPath()%>*/js/jquery-1.8.1.min.js"*></script>

</head>

<body>

username: <input type=*"text"* id=*"username"*><br><br>

password: <input type=*"password"* id=*"password"*><br><br>

<button id=*"loginbtn"*>登录</button>

</body>

<script type=*"text/javascript"*>

$('#loginbtn').click(**function**() {

**var** param = {

username : $("#username").val(),

password : $("#password").val()

};

$.ajax({

type: "post",

url: "<%=request.getContextPath()%>" + "/checkLogin.json",

data: param,

dataType: "json",

success: **function**(data) {

**if**(data.success == **false**){

alert(data.errorMsg);

}**else**{

//登录成功

window.location.href = "<%=request.getContextPath()%>" + "/loginsuccess.html";

}

},

error: **function**(data) {

alert("请求登录失败....");

}

});

});

</script>

</html>

```

（3）loginsuccess.jsp

```

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<script src=*"*<%=request.getContextPath()%>*/js/jquery-1.8.1.min.js"*></script>

</head>

<body>

登录成功...........

<br>

<br>

<button id=*"logout"*>退出</button>

<br>

<br>

<button id=*"newpage1"*>newpage 1</button>

<br>

<br>

<button id=*"newpage2"*>newpage 2</button>

</body>

<script type=*"text/javascript"*>

$('#logout').click(**function**() {

$.ajax({

type: "post",

url: "<%=request.getContextPath()%>" + "/logout.json",

data: {},

dataType: "json",

success: **function**(data) {

**if**(data.success == **false**){

alert(data.errorMsg);

}**else**{

alert("logout success");

//登录成功

window.location.href = "<%=request.getContextPath()%>" + "/login.html";

}

},

error: **function**(data) {

alert("invoke failure....");

}

});

});

$('#newpage1').click(**function**() {

$.ajax({

type: "post",

url: "<%=request.getContextPath()%>" + "/newpage1.json",

data: {},

dataType: "json",

success: **function**(data) {

**if**(data.success == **false**){

alert(data.errorMsg);

}**else**{

window.location.href = "<%=request.getContextPath()%>" + "/newpage1.html";

}

},

error: **function**(data) {

alert("调用失败....");

}

});

});

$('#newpage2').click(**function**() {

$.ajax({

type: "post",

url: "<%=request.getContextPath()%>" + "/newpage2.json",

data: {},

dataType: "json",

success: **function**(data) {

**if**(data.success == **false**){

alert(data.errorMsg);

}**else**{

window.location.href = "<%=request.getContextPath()%>" + "/newpage2.html";

}

},

error: **function**(data) {

alert("调用失败....");

}

});

});

</script>

</html>

```

（4）newPage.jsp

```

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

</head>

<body>

This is a new page. <br>

</body>

</html>

```

（5）pagenofound.jsp

```

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

</head>

<body>

Page not found! <br>

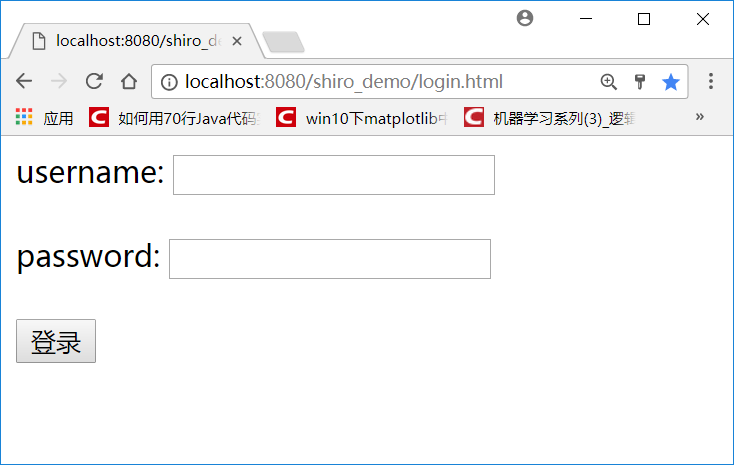
</body>

</html>

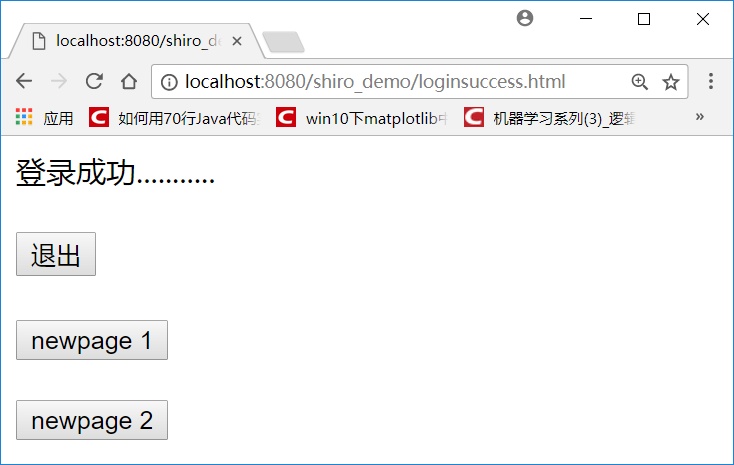
```

四、运行结果

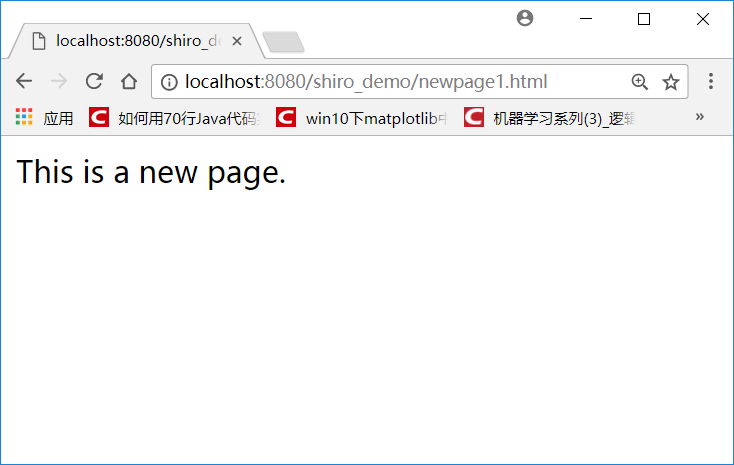
1 在浏览器中输入<http://localhost:8080/shiro_demo/login.html> ，被Shiro拦截后展示的不是index.jsp页面，而是login.jsp页面



2 输入用户名“admin”和密码“123456”后，登录



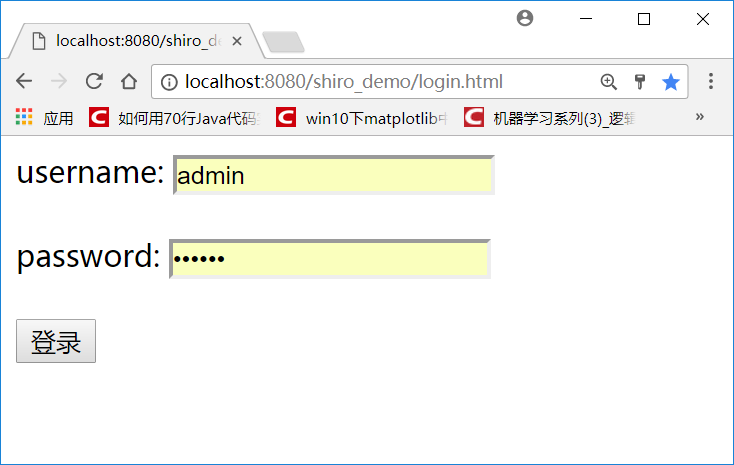
3 点击“newpage 1”按纽，因为有授权，进入newPage.jsp页面



4 点击“newpage 2”按纽，因为没有授权，无法进入newPage.jsp页面，而是进入pagenofound.jsp页面



5 点击“退出”按纽，成功退出登录



五、项目托管

[shiro\_spring源码]( <https://github.com/zhenghaishu/shiro_demo> )