# 第2章：Shiro集成Web

Shiro api文档地址：<http://shiro.apache.org/web.html>

## 环境准备

### 添加jar依赖

|  |
| --- |
| <!-- 添加servlet支持 -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.1.0</version>  </dependency>  <dependency>  <groupId>javax.servlet.jsp</groupId>  <artifactId>javax.servlet.jsp-api</artifactId>  <version>2.3.1</version>  </dependency>  <!-- 添加jstl支持 -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <!-- 添加日志支持 -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.17</version>  </dependency>  <dependency>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  <version>1.2</version>  </dependency>  <!-- 添加shiro支持 -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-core</artifactId>  <version>1.2.4</version>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-web</artifactId>  <version>1.2.4</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.7.12</version>  </dependency> |

### 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"*  id=*"WebApp\_ID"* version=*"2.5"*>  <display-name>shrioweb1</display-name>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  <welcome-file>index.htm</welcome-file>  <welcome-file>index.jsp</welcome-file>  <welcome-file>default.html</welcome-file>  <welcome-file>default.htm</welcome-file>  <welcome-file>default.jsp</welcome-file>  </welcome-file-list>  <!-- 配置shiro监听 -->  <listener>  <listener-class>org.apache.shiro.web.env.EnvironmentLoaderListener</listener-class>  </listener>  <!-- 添加shiro支持 -->  <filter>  <filter-name>ShiroFilter</filter-name>  <filter-class>org.apache.shiro.web.servlet.ShiroFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>ShiroFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  </web-app> |

### shiro配置文件

|  |
| --- |
| #定义用户  [users]  #用户名=密码，角色（管理员）  kazu=123456,admin  #用户名=密码，角色（教师）  jack=123456,teacher  #用户名=密码  tom=123456  #定义角色  [roles]  #admin管理员角色拥有用户所有权限  admin=user:\*  #teacher教师角色拥有学生所有权限  teacher=student:\* |

## Shiro拦截器

|  |
| --- |
|  |

### 案例1：测试匿名访问

#### LoginServlet

|  |
| --- |
| 创建LoginServlet，完成doGet()请求，使用转发跳转到login登录页面 |
|  |

#### Shiro配置文件

|  |
| --- |
| 修改shiro配置文件，添加以下路径 |
| #访问路径  [urls]  #登录请求为anon（匿名请求，不登录也可访问）  /login=anon |

### 案例2：测试登录

#### Login登录页面

|  |
| --- |
| <div>  <font color=*"red"*>${errorInfo }</font>  </div>  <form action=*"login"* method=*"post"*>  userName:<input type=*"text"* name=*"userName"*/><br/>  password:<input type=*"password"* name=*"password"*/><br/>  <input type=*"submit"* value=*"登录"*/>  </form> |

#### doPost()

|  |
| --- |
|  |

### 案例3：测试身份验证

#### 需求

|  |
| --- |
| 测试在浏览器中，未登录时访问/admin下所有的请求 |

#### AdminServlet

|  |
| --- |
|  |

#### Shiro配置文件

|  |
| --- |
| 在urls中添加admin的访问拦截 |
|  |

### 案例4：验证角色

#### 需求

|  |
| --- |
| 访问/student学生资源，前提是访问学生资源必须拥有教师角色才能访问 |

#### Shiro配置文件

|  |
| --- |
|  |

#### 测试

|  |
| --- |
| 测试没有角色：   1. 登录没有teacher角色的账户（如：kazu） 2. 在浏览器中输入/student路径（结果：跳转到没有权限的页面） |
| 测试拥有角色：   1. 登录拥有teacher角色的账户（如：jack） 2. 在浏览器中输入/student路径（结果：404请求） |

### 案例4：验证权限

#### 需求

|  |
| --- |
| 访问/teacher教师资源，创建用户，前提是访问教师资源必须拥有管理员角色才能访问 |

#### Shiro配置文件

|  |
| --- |
|  |

#### 测试

|  |
| --- |
| 测试没有权限：   1. 登录没有创建用户权限的账户（如：jack） 2. 在浏览器中输入/teacher路径（结果：跳转到没有权限的页面） |
| 测试拥有权限：   1. 登录拥有创建用户权限的账户（如：kazu） 2. 在浏览器中输入/teacher路径（结果：404请求） |

## URL匹配方式

|  |  |  |
| --- | --- | --- |
| 占位符 | 含义 | 说明 |
| ? | 匹配一个字符  /admin? | 可以匹配/admin1或/admin2  不可以匹配/admin12  不可以匹配/admin |
| \* | 匹配0到n个字符  /admin\* | 可以匹配 /admin或/admin1或 /admin12  不可以匹配/admin/abc |
| \*\* | 匹配零个或者多个路径  /admin/\*\* | 可以匹配/admin  可以匹配/admin/a  可以匹配/admin/a/b |
| \*/\*\* | 以xxx开头的多个路径/admin\*/\*\* |  |

## Shiro标签的使用

### 导入shiro标签库

|  |
| --- |
|  |

### <shiro:guest>

|  |
| --- |
| <shiro:guest>  游客访问 <a href = *"login.jsp"*></a>  </shiro:guest> |

### <shiro:user>

|  |
| --- |
| **user 标签：用户已经通过认证\记住我 登录后显示响应的内容**  <shiro:user>  欢迎[<shiro:principal/>]登录 <a href = *"logout"*>退出</a>  </shiro:user> |

### <shiro:principal/>

|  |
| --- |
| principal 标签：显示用户身份信息，默认调用Subject.getPrincipal()获取，即Primary Principal |

### <shiro:hasRole>

|  |
| --- |
| <shiro:hasRole name=*"admin"*>  欢迎有admin角色的用户！  </shiro:hasRole> |

### <shiro:hasPermission>

|  |
| --- |
| <shiro:hasPermission name=*"student:create"*>  欢迎有student:create权限的用户！  </shiro:hasPermission> |

### <shiro:authenticated>

|  |
| --- |
| authenticated标签：用户身份验证通过，即 Subject.login() 登录成功 不是记住我登录的<br>  <shiro:authenticated>  用户[<shiro:principal/>] 已身份验证通过  </shiro:authenticated> |

### <shiro:notAuthenticated>

|  |
| --- |
| notAuthenticated标签：用户未进行身份验证，即没有调用Subject.login进行登录,包括"记住我"也属于未进行身份验证<br>  <shiro:notAuthenticated>  未身份验证(包括"记住我")  </shiro:notAuthenticated> |

## 加密

### 散列算法

|  |
| --- |
| 散列算法一般用于生成数据的摘要信息，是一种不可逆的算法，一般适合存储密码之类的数据，常见的散列算法如MD5、SHA等。一般进行散列时最好提供一个salt（盐），比如加密密码“admin”，产生的散列值是“21232f297a57a5a743894a0e4a801fc3”，可以到一些md5 解密网站很容易的通过散列值得到密码“admin”，即如果直接对密码进行散列相对来说破解更容易，此时我们可以加一些只有系统知道的干扰数据，如用户名和ID（即盐）；这样散列的对象是“密码+用户名+ID”，这样生成的散列值相对来说更难破解。 |

### Base64加密

|  |
| --- |
| /\*\*  \* base64加密  \* **@param** str  \* **@return**  \*/  **public** **static** String encBase64(String str){  **return** Base64.*encodeToString*(str.getBytes());  } |

### Base64解密

|  |
| --- |
| /\*\*  \* base64解密  \* **@param** str  \* **@return**  \*/  **public** **static** String decBase64(String str){  **return** Base64.*decodeToString*(str);  } |

### MD5加密

|  |
| --- |
| /\*\*  \* md5加密  \* **@param** str 密码  \* **@param** salt 盐值（佐料），建议值是唯一的  \* **@return**  \*/  **public** **static** String md5Hash(String str,String salt) {  **return** **new** Md5Hash(str, salt).toString();  }    /\*\*  \* md5加密  \* **@param** str 密码  \* **@param** salt 佐料  \* **@param** count 加密次数  \* **@return**  \*/  **public** **static** String md5Hash(String str,String salt,**int** count) {  **return** **new** Md5Hash(str, salt,count).toString();  } |

### 完整案例

|  |
| --- |
| **import** org.apache.shiro.codec.Base64;  **import** org.apache.shiro.crypto.hash.Md5Hash;  **public** **class** PasswordUtil {  /\*\*  \* base64加密  \* **@param** str  \* **@return**  \*/  **public** **static** String encBase64(String str){  **return** Base64.*encodeToString*(str.getBytes());  }    /\*\*  \* base64解密  \* **@param** str  \* **@return**  \*/  **public** **static** String decBase64(String str){  **return** Base64.*decodeToString*(str);  }      /\*\*  \* md5加密  \* **@param** str 密码  \* **@param** salt 盐值（佐料），建议值是唯一的  \* **@return**  \*/  **public** **static** String md5Hash(String str,String salt) {  **return** **new** Md5Hash(str, salt).toString();  }    /\*\*  \* md5加密  \* **@param** str 密码  \* **@param** salt 佐料  \* **@param** count 加密次数  \* **@return**  \*/  **public** **static** String md5Hash(String str,String salt,**int** count) {  **return** **new** Md5Hash(str, salt,count).toString();  }        **public** **static** **void** main(String[] args) {  String password = "123456";  System.***out***.println("Base64加密："+PasswordUtil.*encBase64*(password));  System.***out***.println("Base64解密："+PasswordUtil.*decBase64*(PasswordUtil.*encBase64*(password)));  System.***out***.println("MD5加密："+PasswordUtil.*md5Hash*(password, "test"));  System.***out***.println("MD5加密："+PasswordUtil.*md5Hash*(password, "test",2));  }  } |