Shiro 入门教程

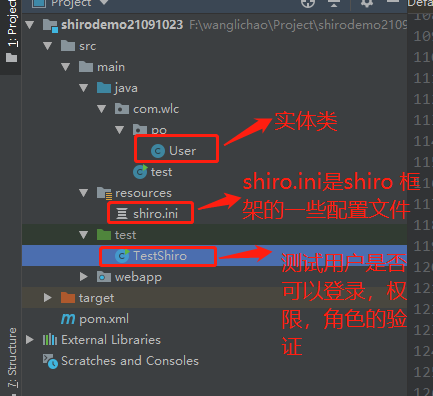
# 1、为什么要使用shiro?

在使用Shiro 之前，做登录，权限什么的都是**五花八门**，各种**花里胡哨**的代码，不同系统的做法很有可能千差万别。  
但是使用 Shiro 这个安全框架之后，大家做权限的方式都**一致化**了，这样的好处就是你的代码我看起来容易，我的代码你也好理解。  
Shiro 也比较成熟，基本上能满足大部分的权限需要。

## 2、入门案例：最基础的案例

介绍：用户信息在shiro.ini 配置文件中，没有经过数据库

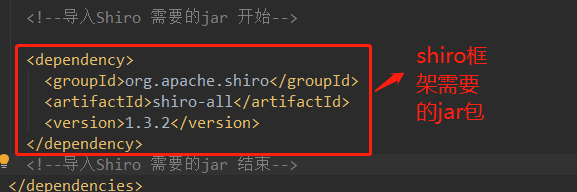
### 目录结构：



### 步骤：

### 0、添加依赖

在pom.xml 添加shiro框架需要的jar包 shiro-all.jar



完整的pom文件

|  |
| --- |
| <?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>com.wlc</groupId>  <artifactId>shiro-demo-20191023</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>war</packaging>   <name>shiro-demo-20191023 Maven Webapp</name>  <!-- *FIXME change it to the project's website* -->  <url>http://www.example.com</url>   <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <maven.compiler.source>1.7</maven.compiler.source>  <maven.compiler.target>1.7</maven.compiler.target>  </properties>   <build>  <finalName>shiro-demo-20191023</finalName>  <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->  <plugins>  <plugin>  <artifactId>maven-clean-plugin</artifactId>  <version>3.1.0</version>  </plugin>  <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_war\_packaging -->  <plugin>  <artifactId>maven-resources-plugin</artifactId>  <version>3.0.2</version>  </plugin>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.8.0</version>  </plugin>  <plugin>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.22.1</version>  </plugin>  <plugin>  <artifactId>maven-war-plugin</artifactId>  <version>3.2.2</version>  </plugin>  <plugin>  <artifactId>maven-install-plugin</artifactId>  <version>2.5.2</version>  </plugin>  <plugin>  <artifactId>maven-deploy-plugin</artifactId>  <version>2.8.2</version>  </plugin>  </plugins>  </pluginManagement>  </build>   <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  <scope>test</scope>  </dependency>   <!--导入Shiro 需要的jar 开始-->   <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-all</artifactId>  <version>1.3.2</version>  </dependency>  <!--导入Shiro 需要的jar 结束-->  </dependencies> </project> |

### 1、shiro.ini

在resources 下面添加shiro.ini 里面定义了角色，用户，权限相关的信息。

|  |
| --- |
| #定义用户 **[users]** #用户名： 张三，密码：1， 角色：admin #用户名： liShiShi，密码2， 角色：productManager #用户名： zhYu，密码：3， 角色：orderManager 张三 = 1 , admin lisi = 2 wangwu = 3,orderManager,admin #定义角色 **[roles]** #admin 管理员 什么都可以做 #productManager: 产品经理，只能做：addProduct,deleteProduct,updateProduct,listProduct #orderManager: 订单经理，只能做： addOrder,delteOrder,updateOrder,listOrder admin=\* productManager=addProduct,deleteProduct,updateProduct,listProduct orderManager=addOrder,deleteOrder,updateOrder,listOrder |

### 2、User.java

在src下面新建 包com.wlc.po 实体类User.java用户的信息

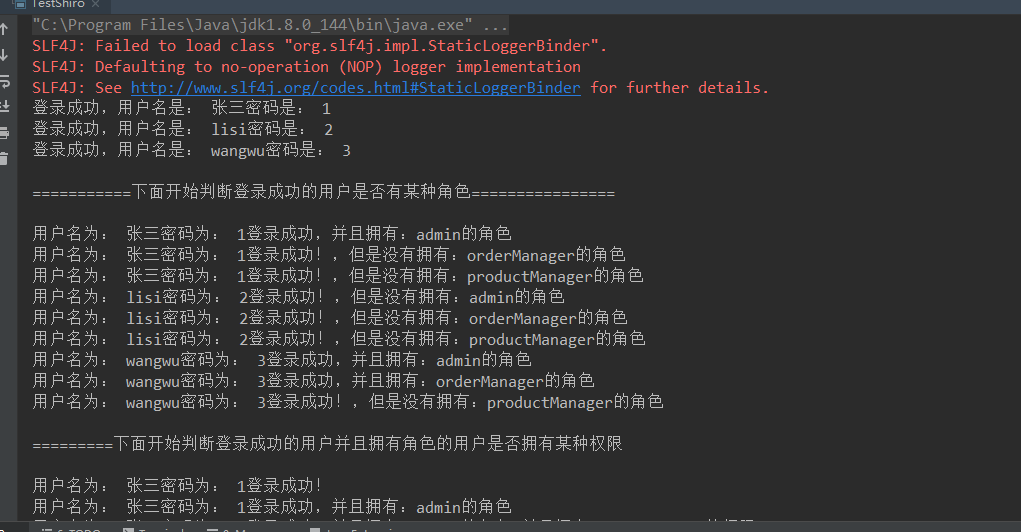
|  |
| --- |
| public class User {  private int id;  private String name;  private String password;   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  }   public User() {  }   public User(int id, String name, String password) {  this.id = id;  this.name = name;  this.password = password;  } } |

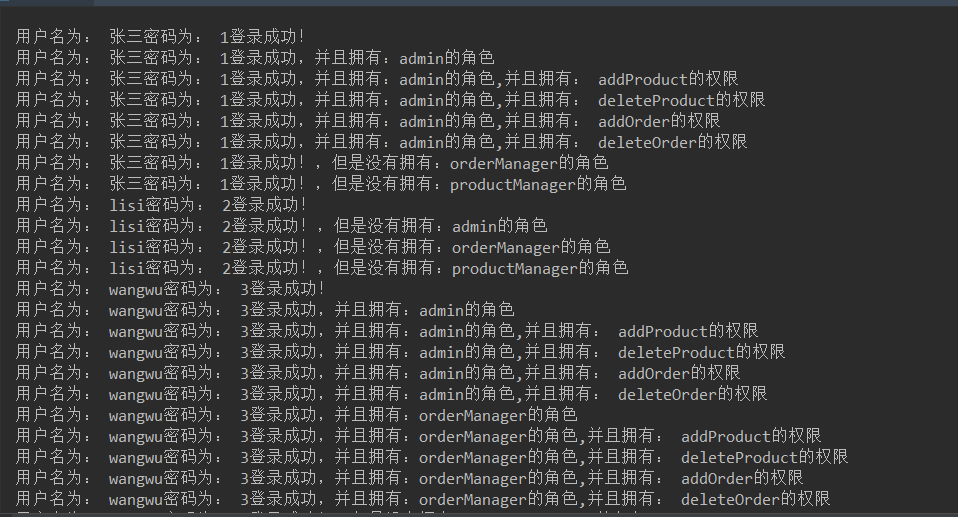
### 3、TestShiro.java

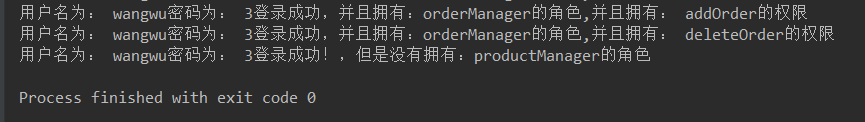
在test文件下面新建测试类TestShiro.java

|  |
| --- |
| import org.apache.shiro.SecurityUtils; import org.apache.shiro.authc.AuthenticationException; import org.apache.shiro.authc.UsernamePasswordToken; import org.apache.shiro.config.IniSecurityManagerFactory; import org.apache.shiro.mgt.SecurityManager; import org.apache.shiro.subject.Subject; import org.apache.shiro.util.Factory;  import java.util.ArrayList; import java.util.List;  */\*\*  \* describe:  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/23  \*/* public class TestShiro {   //获取当前的登录的对象 Subject  private Subject getSubject(User user) {  //加载配置文件，并获取工厂  Factory<SecurityManager> securityManagerFactory =  new IniSecurityManagerFactory("classpath:shiro.ini");  //获取管理者实例  SecurityManager securityManager = securityManagerFactory.getInstance();  //把安全管理放入全局对象  SecurityUtils.*setSecurityManager*(securityManager);  //全局对象通过安全管理生成subject 对象  Subject subject = SecurityUtils.*getSubject*();   return subject;  }   //判断用户是否可以登录的方法  private boolean login(User user) {  //获取当前的对象 subject  Subject subject = getSubject(user);  //如果登录过了，就退出  if (subject.isAuthenticated()) {  subject.logout();  }  //封装用户的数据  UsernamePasswordToken token = new UsernamePasswordToken(user.getName(), user.getPassword());  //然后把用户的token 最终传入到 realm 中进行对比  try {  subject.login(token);  } catch (AuthenticationException e) {  return false;  }  return subject.isAuthenticated();  }   //判断登录成功的用户是否具有某种角色的方法  private boolean hasRole(User user, String role) {  Subject subject = getSubject(user);  return subject.hasRole(role);  }   //判断登陆成功的用户，拥有的角色，是否配置可访问的权限  private boolean isPermitted(User user, String permit) {  Subject subject = getSubject(user);  return subject.isPermitted(permit);  }   public static void main(String[] args) {  TestShiro testShiro = new TestShiro();  //创建用户，角色，权限  //1、定义用户  User user1 = new User();  user1.setName("张三");  user1.setPassword("1");   User user2 = new User();  user2.setName("lisi");  user2.setPassword("2");   User user3 = new User();  user3.setName("wangwu");  user3.setPassword("3");   //把用户放入到list集合中  List<User> userList = new ArrayList<>();  userList.add(user1);  userList.add(user2);  userList.add(user3);   //2、定义角色  String roleAdmin = "admin";  String roleProductManager = "productManager";  String roleOrderManager = "orderManager";   //把角色放入到list 集合中  List<String> roleList = new ArrayList<>();  roleList.add(roleAdmin);  roleList.add(roleOrderManager);  roleList.add(roleProductManager);   //3、定义权限  String permitAddProduct = "addProduct";  String permitDeleteProduct = "deleteProduct";  String permitAddOrder = "addOrder";  String permitDeleteOrder = "deleteOrder";   //把权限放入到list 集合中  List<String> permitList = new ArrayList<>();  permitList.add(permitAddProduct);  permitList.add(permitDeleteProduct);  permitList.add(permitAddOrder);  permitList.add(permitDeleteOrder);   //判断用户是否可以登录  for (User user : userList) {  if (testShiro.login(user)) {  System.*out*.println("登录成功，用户名是： " + user.getName() + "密码是： " + user.getPassword());  } else {  System.*out*.println("登录失败！用户名是： " + user.getName() + "密码是： " + user.getPassword());  }  }   System.*out*.println("");  System.*out*.println("===========下面开始判断登录成功的用户是否有某种角色================");  System.*out*.println("");  for (User user : userList) {  if (testShiro.login(user)) {  for (String role : roleList) {  if (testShiro.hasRole(user, role)) {  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() + "" +  "登录成功，并且拥有：" + role + "的角色");  } else {  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() +  "登录成功！，但是没有拥有：" + role + "的角色");  }  }   } else {  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() + "登录失败！");  }   }   System.*out*.println();  System.*out*.println("=========下面开始判断登录成功的用户并且拥有角色的用户是否拥有某种权限");  System.*out*.println();  for (User user : userList) {  //判断是否登录成功  if (testShiro.login(user)) {  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() + "登录成功！");  //判断是否拥有角色  for (String role : roleList) {  if (testShiro.hasRole(user, role)) {  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() + "" +  "登录成功，并且拥有：" + role + "的角色");  //判断该角色是否拥有权限  for (String permit: permitList) {  if(testShiro.isPermitted(user,permit)){  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() + "" +  "登录成功，并且拥有：" + role + "的角色,并且拥有： " + permit + "的权限");  }else{  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() + "" +  "登录成功，并且拥有：" + role + "的角色,但是没有拥有： " + permit + "的权限");  }  }  } else {  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() +  "登录成功！，但是没有拥有：" + role + "的角色");  }  }  } else {  System.*out*.println("用户名为： " + user.getName() + "密码为： " + user.getPassword() + "登录失败！");  }    }  } } |

### 4、测试结果



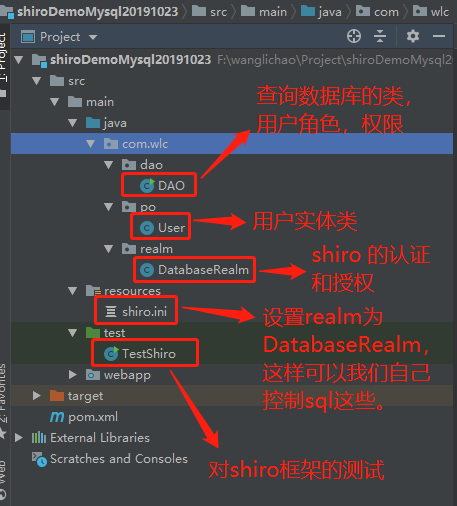




## 3、入门案例升级

**用户信息存放于数据库，基于数据库进行登录权限的认证和授权，但是 用户的密码没有加密，是用明文保存在数据库中的。**

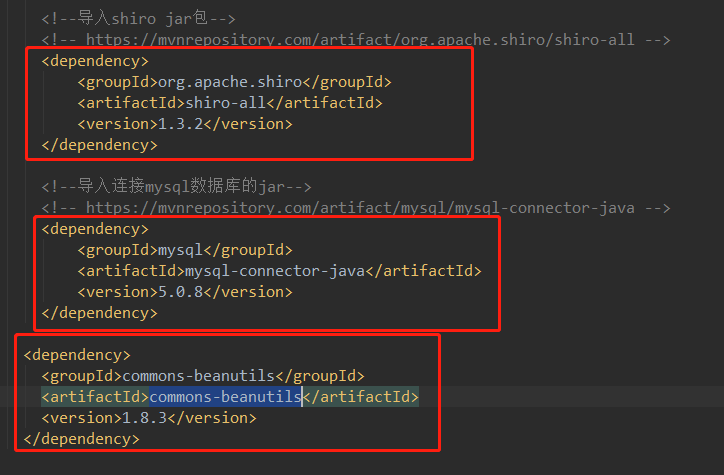
### 目录结构



### 步骤：

### 0. pom依赖

添加pom依赖



完整的pom.xml文件

|  |
| --- |
| <?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>com.wlc</groupId>  <artifactId>shiroDemoMysql</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>war</packaging>   <name>shiroDemoMysql Maven Webapp</name>  <!-- *FIXME change it to the project's website* -->  <url>http://www.example.com</url>   <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <maven.compiler.source>1.7</maven.compiler.source>  <maven.compiler.target>1.7</maven.compiler.target>  </properties>   <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  <scope>test</scope>  </dependency>   <!--导入shiro jar包-->  <!-- https://mvnrepository.com/artifact/org.apache.shiro/shiro-all -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-all</artifactId>  <version>1.3.2</version>  </dependency>   <!--导入连接mysql数据库的jar-->  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.0.8</version>  </dependency>   <dependency>  <groupId>commons-beanutils</groupId>  <artifactId>commons-beanutils</artifactId>  <version>1.8.3</version>  </dependency>     </dependencies>   <build>  <finalName>shiroDemoMysql</finalName>  <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->  <plugins>  <plugin>  <artifactId>maven-clean-plugin</artifactId>  <version>3.1.0</version>  </plugin>  <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_war\_packaging -->  <plugin>  <artifactId>maven-resources-plugin</artifactId>  <version>3.0.2</version>  </plugin>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.8.0</version>  </plugin>  <plugin>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.22.1</version>  </plugin>  <plugin>  <artifactId>maven-war-plugin</artifactId>  <version>3.2.2</version>  </plugin>  <plugin>  <artifactId>maven-install-plugin</artifactId>  <version>2.5.2</version>  </plugin>  <plugin>  <artifactId>maven-deploy-plugin</artifactId>  <version>2.8.2</version>  </plugin>  </plugins>  </pluginManagement>  </build> </project> |

### User.java 实体类

在com.wlc.po这个包下面新建User.java 实体类，用来存放用户信息

|  |
| --- |
| public class User {  private int id ;  private String name ;  private String password ;   public User() {  }   public User(int id, String name, String password) {  this.id = id;  this.name = name;  this.password = password;  }   @Override  public String toString() {  return "User{" +  "id=" + id +  ", name='" + name + '\'' +  ", password='" + password + '\'' +  '}';  }   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  } } |

### DAO.java

在com.wlc.dao 这个包下面新建DAO.java ，用来查询用户密码，角色，权限信息。

|  |
| --- |
| import java.sql.\*; import java.util.HashSet; import java.util.Set;  */\*\*  \* describe:查询数据库的类  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/23  \*/* public class DAO {   public DAO() {  try {  Class.*forName*("com.mysql.jdbc.Driver");  } catch (ClassNotFoundException e) {  e.printStackTrace();  }  }   */\*\*  \* 获取数据库连接  \*\*/* public Connection getConnection() throws SQLException {  return DriverManager.*getConnection*("jdbc:mysql://127.0.0.1:3306/shiro02?characterEncoding=UTF-8",  "scott", "tiger");   }   */\*\*  \* 根据用户名 获取用户名密码  \*\*/* public String getPassword(String userName) {  try {  Connection connection = getConnection();  String sql = "select password from user where name =?";  PreparedStatement preparedStatement = connection.prepareStatement(sql);  preparedStatement.setString(1, userName);  ResultSet resultSet = preparedStatement.executeQuery();  while (resultSet.next()) {  return resultSet.getString("password");  }  } catch (SQLException e) {  e.printStackTrace();  }  return null;  }   */\*\*  \* 根据用户名获取用户的角色  \*\*/* public Set<String> listRole(String userName) {  Set<String> roleSet = new HashSet<>();  String sql = "select r.name from user u left join user\_role ur on ur.uid = u.id left join role r on ur.rid = r.id where u.name = ?";  roleSet = commonSqlPre(sql, userName, roleSet);  return roleSet;  }   */\*\*  \* 公共部分的查询，抽出到一个方法中，提高重用性  \*\*/* public Set<String> commonSqlPre(String sql, String userName, Set<String> permissionSet) {  try {  Connection connection = getConnection();  PreparedStatement preparedStatement = connection.prepareStatement(sql);  preparedStatement.setString(1, userName);  ResultSet resultSet = preparedStatement.executeQuery();  while (resultSet.next()) {  permissionSet.add(resultSet.getString(1));  }  } catch (SQLException e) {  e.printStackTrace();  }  return permissionSet;  }   */\*\*  \* 根据用户名，获取用户所具有的权限  \*\*/* public Set<String> listPermissions(String userName) {  Set<String> permissionSet = new HashSet<>();  String sql = "SELECT p.name FROM USER u LEFT JOIN user\_role ur ON ur.uid = u.id left join role r on ur.rid = r.id LEFT JOIN role\_permission rp ON rp.rid = ur.rid LEFT JOIN permission p ON rp.pid = p.id where u.name=?";  permissionSet = commonSqlPre(sql, userName, permissionSet);  return permissionSet;  }    public static void main(String[] args) {  DAO dao = new DAO();  System.*out*.println(dao.listPermissions("zhang3"));   }  } |

### DatabaseRealm.java

在com.wlc.realm 新建 DatabaseRealm.java，用来对用户名和面的 验证和授权

这个类继承了AuthorizingRealm.java,

其中：doGetAuthorizationInfo 这个方法用来做授权

doGetAuthenticationInfo：这个方法用来做认证

|  |
| --- |
| import com.wlc.dao.DAO; import org.apache.shiro.authc.\*; 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 java.util.Set;  */\*\*  \* describe:  \* 继承 AuthorizingRealm 类，登录最后的验证时在realm中验证的  \* DatabaseRealm 就是用来通过数据库 验证用户，和相关授权的类。  \* 两个方法分别做验证和授权：  \* doGetAuthenticationInfo(), 验证  \*  \* doGetAuthorizationInfo() 授权  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/23  \*/* public class DatabaseRealm extends AuthorizingRealm {  */\*\*  \* 授权  \* \*\*/* @Override  protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {  //这个部分是进行授权的代码，如果可以进入到这里，就说明 登录已经通过了  //获取用户名  String userName = (String) principalCollection.getPrimaryPrincipal();  //根据用户名获取用户的角色  Set<String> roleSet = new DAO().listRole(userName);  //根据用户名获取用户拥有的角色的权限信息  Set<String> permissionsSet = new DAO().listPermissions(userName);   //授权对象  SimpleAuthorizationInfo simpleAuthorizationInfo = new SimpleAuthorizationInfo();  //把通过 dao 获取到的 角色和权限放入到 授权对象中  simpleAuthorizationInfo.setRoles(roleSet);  simpleAuthorizationInfo.setStringPermissions(permissionsSet);  return simpleAuthorizationInfo;  }   */\*\*验证\*\*/* @Override  protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken authenticationToken) throws AuthenticationException {  //获取用户名和密码  /\* String password = authenticationToken.getCredentials().toString();\*/  UsernamePasswordToken token = (UsernamePasswordToken) authenticationToken;  String password = new String(token.getPassword());  String userName = authenticationToken.getPrincipal().toString();  //读取数据库中的用户名和密码  String passWordInnDb = new DAO().getPassword(userName);  //数据库中的密码为空或者和密码不一样，就报错  if(null == passWordInnDb || !passWordInnDb.equals(password)){  throw new AuthenticationException();  }  //最后把用户的 用户名和密码放到 认证信息里面  SimpleAuthenticationInfo simpleAuthenticationInfo = new SimpleAuthenticationInfo(userName,password,getName());  return simpleAuthenticationInfo;  } } |

### shiro.ini

在resources这个文件夹下面新建shiro.ini

主要用来指定 安全管理这个securityManager.realm = 我们自己重写的那个DatabaseRealm.java

|  |
| --- |
| **[main]** databaseRealm=com.wlc.realm.DatabaseRealm securityManager.realms = $databaseRealm |

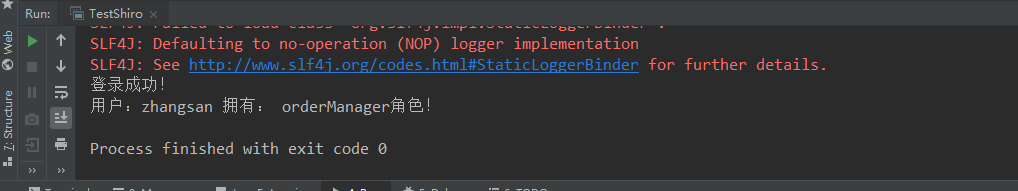
### TestShiro.java

在test文件夹下面下新建TestShiro.java

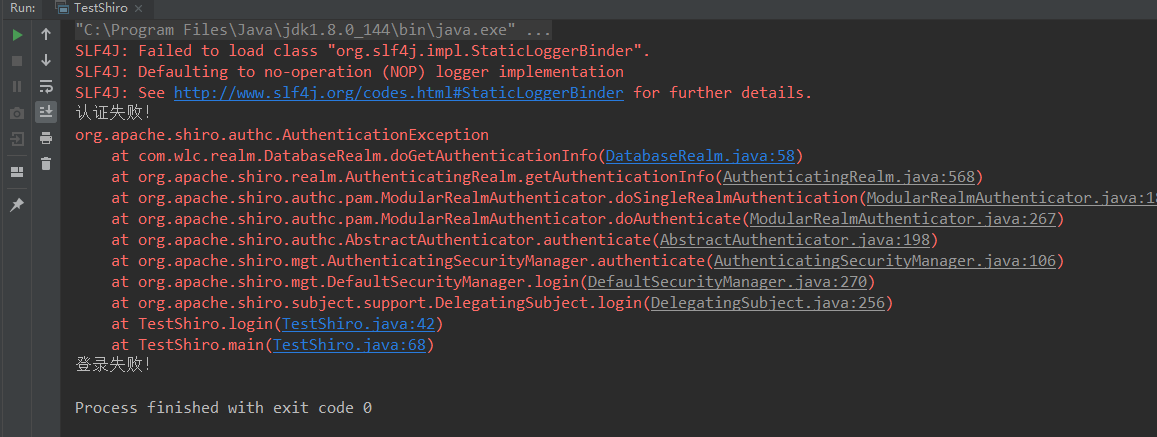
|  |
| --- |
| import org.apache.shiro.SecurityUtils; import org.apache.shiro.authc.AuthenticationException; import org.apache.shiro.authc.UsernamePasswordToken; import org.apache.shiro.config.IniSecurityManagerFactory; import org.apache.shiro.mgt.SecurityManager; import org.apache.shiro.subject.Subject; import org.apache.shiro.util.Factory;  */\*\*  \* describe:  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/23  \*/* public class TestShiro {   public Subject getSubject(User user){  //获取工厂  Factory<SecurityManager> factory= new IniSecurityManagerFactory("classpath:shiro.ini");  //获取安全管理者实例  SecurityManager securityManager = factory.getInstance();  //把安全管理者实例放入到全局对象  SecurityUtils.*setSecurityManager*(securityManager);  //获取subject  Subject subject = SecurityUtils.*getSubject*();  return subject;  }  */\*\*  \* 登录的方法  \*\*/* public boolean login(User user) {  //获取subject 当前对象  Subject subject = getSubject(user);  //如果用户已经登录过了，就退出，可以做 一个账户只能一个用户登录，多个用户登录，就会把另外一个  //用户顶下去  if(subject.isAuthenticated()){  subject.logout();  }  UsernamePasswordToken token = new UsernamePasswordToken(user.getName(),user.getPassword());  try {  subject.login(token);  } catch (AuthenticationException e) {  System.*out*.println("认证失败！");  e.printStackTrace();  return false;  }  return subject.isAuthenticated();  }   //查看用户的角色  public boolean hasRole(User user,String role){  Subject subject = getSubject(user);  return subject.hasRole(role);  }   public static void main(String[] args) {  TestShiro testShiro = new TestShiro();  //创建用户  User user = new User();  user.setName("zhangsan");  user.setPassword("1");   //创建角色  String role = "orderManager";   //登录验证  if (testShiro.login(user)) {  System.*out*.println("登录成功！");  if(testShiro.hasRole(user,role)){  System.*out*.println("用户：" + user.getName() + " 拥有： " + role + "角色！");  }else{  System.*out*.println("用户：" + user.getName() + "没有拥有：" + role + "角色！");  }  }else{  System.*out*.println("登录失败！");  }   } } |

### 测试结果

阿萨德登录通过的结果为：



登录验证失败的结果为：



**注意事项：**

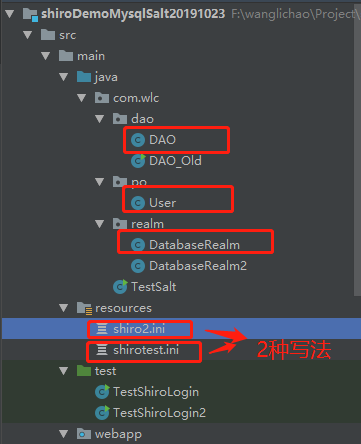
**如果不导入commons-beanutils.jar 会报下面的错误**

|  |
| --- |
| "C:\Program Files\Java\jdk1.8.0\_144\bin\java.exe" "-javaagent:F:\wanglichao\location\IDEA\_location\IntelliJ IDEA 2018.3\lib\idea\_rt.jar=52218:F:\wanglichao\location\IDEA\_location\IntelliJ IDEA 2018.3\bin" -Dfile.encoding=UTF-8 -classpath "C:\Program Files\Java\jdk1.8.0\_144\jre\lib\charsets.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\deploy.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\access-bridge-64.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\cldrdata.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\dnsns.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\jaccess.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\jfxrt.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\localedata.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\nashorn.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\sunec.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\sunjce\_provider.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\sunmscapi.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\sunpkcs11.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\ext\zipfs.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\javaws.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\jce.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\jfr.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\jfxswt.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\jsse.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\management-agent.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\plugin.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\resources.jar;C:\Program Files\Java\jdk1.8.0\_144\jre\lib\rt.jar;F:\wanglichao\Project\shiroDemoMysql20191023\target\test-classes;F:\wanglichao\Project\shiroDemoMysql20191023\target\classes;F:\wanglichao\Maven\_repository\junit\junit\4.11\junit-4.11.jar;F:\wanglichao\Maven\_repository\org\hamcrest\hamcrest-core\1.3\hamcrest-core-1.3.jar;F:\wanglichao\Maven\_repository\org\apache\shiro\shiro-all\1.3.2\shiro-all-1.3.2.jar;F:\wanglichao\Maven\_repository\org\apache\shiro\shiro-guice\1.3.2\shiro-guice-1.3.2.jar;F:\wanglichao\Maven\_repository\org\apache\shiro\shiro-core\1.3.2\shiro-core-1.3.2.jar;F:\wanglichao\Maven\_repository\org\slf4j\slf4j-api\1.6.4\slf4j-api-1.6.4.jar;F:\wanglichao\Maven\_repository\commons-beanutils\commons-beanutils\1.8.3\commons-beanutils-1.8.3.jar;F:\wanglichao\Maven\_repository\com\google\inject\guice\3.0\guice-3.0.jar;F:\wanglichao\Maven\_repository\javax\inject\javax.inject\1\javax.inject-1.jar;F:\wanglichao\Maven\_repository\aopalliance\aopalliance\1.0\aopalliance-1.0.jar;F:\wanglichao\Maven\_repository\com\google\inject\extensions\guice-multibindings\3.0\guice-multibindings-3.0.jar;F:\wanglichao\Maven\_repository\mysql\mysql-connector-java\5.0.8\mysql-connector-java-5.0.8.jar" TestShiro  SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".  SLF4J: Defaulting to no-operation (NOP) logger implementation  SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.  Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/commons/logging/LogFactory  at org.apache.commons.beanutils.ConvertUtilsBean.<init>(ConvertUtilsBean.java:157)  at org.apache.commons.beanutils.BeanUtilsBean.<init>(BeanUtilsBean.java:117)  at org.apache.commons.beanutils.BeanUtilsBean$1.initialValue(BeanUtilsBean.java:68)  at org.apache.commons.beanutils.ContextClassLoaderLocal.get(ContextClassLoaderLocal.java:153)  at org.apache.commons.beanutils.BeanUtilsBean.getInstance(BeanUtilsBean.java:80)  at org.apache.commons.beanutils.PropertyUtilsBean.getInstance(PropertyUtilsBean.java:114)  at org.apache.commons.beanutils.PropertyUtils.getPropertyDescriptor(PropertyUtils.java:460)  at org.apache.shiro.config.ReflectionBuilder.isTypedProperty(ReflectionBuilder.java:409)  at org.apache.shiro.config.ReflectionBuilder.applyProperty(ReflectionBuilder.java:702)  at org.apache.shiro.config.ReflectionBuilder.applySingleProperty(ReflectionBuilder.java:364)  at org.apache.shiro.config.ReflectionBuilder.applyProperty(ReflectionBuilder.java:325)  at org.apache.shiro.config.ReflectionBuilder$AssignmentStatement.doExecute(ReflectionBuilder.java:955)  at org.apache.shiro.config.ReflectionBuilder$Statement.execute(ReflectionBuilder.java:887)  at org.apache.shiro.config.ReflectionBuilder$BeanConfigurationProcessor.execute(ReflectionBuilder.java:765)  at org.apache.shiro.config.ReflectionBuilder.buildObjects(ReflectionBuilder.java:260)  at org.apache.shiro.config.IniSecurityManagerFactory.buildInstances(IniSecurityManagerFactory.java:167)  at org.apache.shiro.config.IniSecurityManagerFactory.createSecurityManager(IniSecurityManagerFactory.java:130)  at org.apache.shiro.config.IniSecurityManagerFactory.createSecurityManager(IniSecurityManagerFactory.java:108)  at org.apache.shiro.config.IniSecurityManagerFactory.createInstance(IniSecurityManagerFactory.java:94)  at org.apache.shiro.config.IniSecurityManagerFactory.createInstance(IniSecurityManagerFactory.java:46)  at org.apache.shiro.config.IniFactorySupport.createInstance(IniFactorySupport.java:123)  at org.apache.shiro.util.AbstractFactory.getInstance(AbstractFactory.java:47)  at TestShiro.getSubject(TestShiro.java:22)  at TestShiro.login(TestShiro.java:34)  at TestShiro.main(TestShiro.java:59)  Caused by: java.lang.ClassNotFoundException: org.apache.commons.logging.LogFactory  at java.net.URLClassLoader.findClass(URLClassLoader.java:381)  at java.lang.ClassLoader.loadClass(ClassLoader.java:424)  at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:335)  at java.lang.ClassLoader.loadClass(ClassLoader.java:357)  ... 25 more  Process finished with exit code 1 |

## 4、入门案例升级

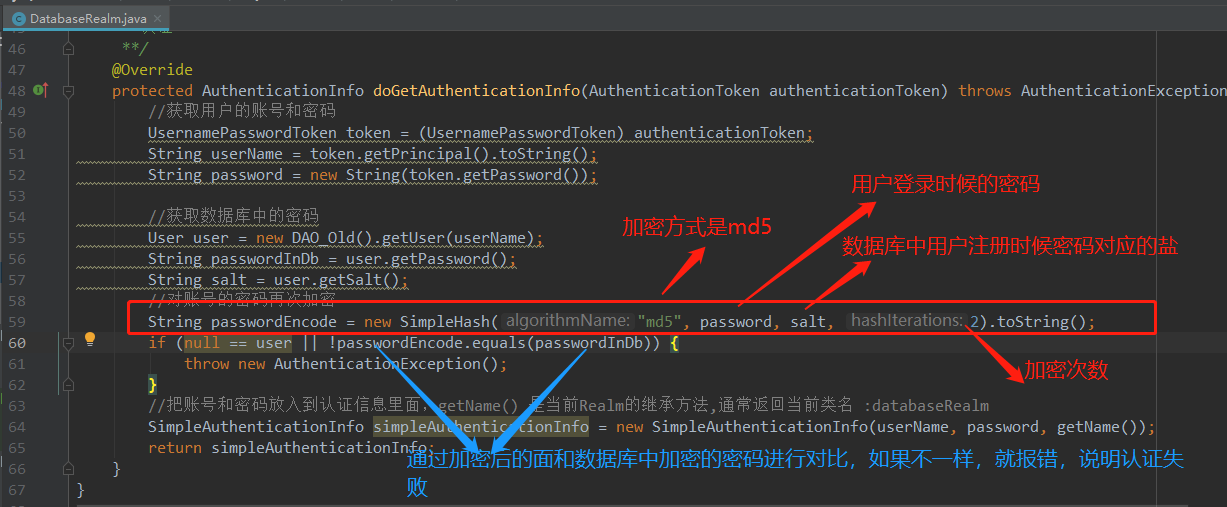
基于数据库，用户密码是用md5加密过的，并且是加盐的，这样每个人的密码都不一样的。

目录结构



这个 案例 shiro 的认证 有2种方式，一种是 通过

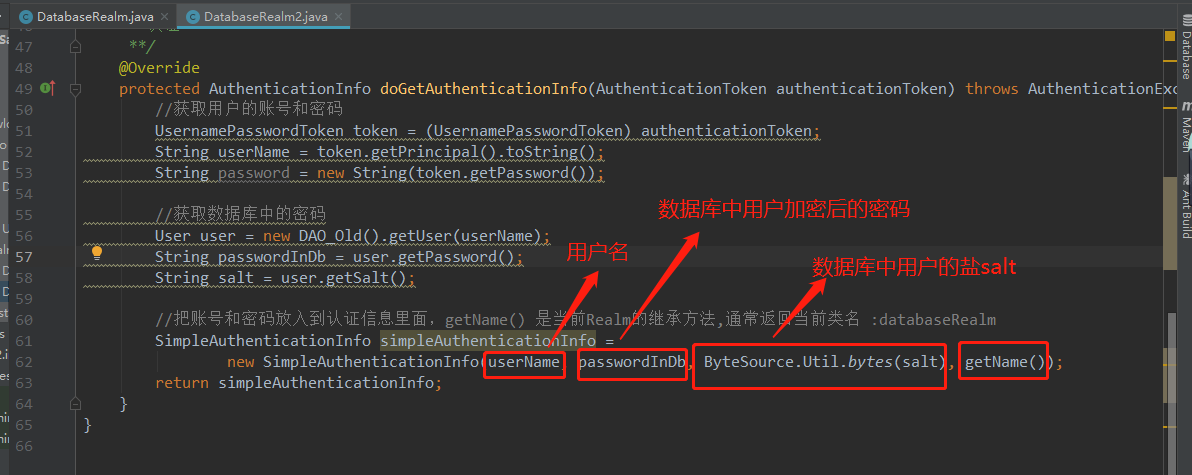
获取用户的密码，以及数据库中存的用户的盐 然后加密得到 passwordEncode ，然后和数据库中用户加密的密码进行对比，如果一样，就说明认证成功，否则，认证失败



第二种方式是：使用Shiro提供的 HashedCredentialsMatcher 帮我们做

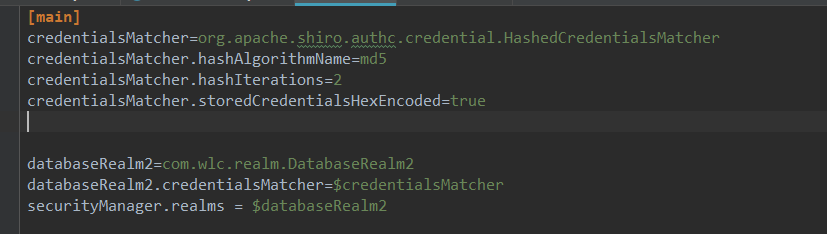
**第一步：**  
在创建 SimpleAuthenticationInfo 的时候，把数据库中取出来的密文以及盐作为参数传递进去。

|  |
| --- |
| SimpleAuthenticationInfo =  new SimpleAuthenticationInfo(userName,passwordInDB,ByteSource.Util.bytes(salt),getName()); |



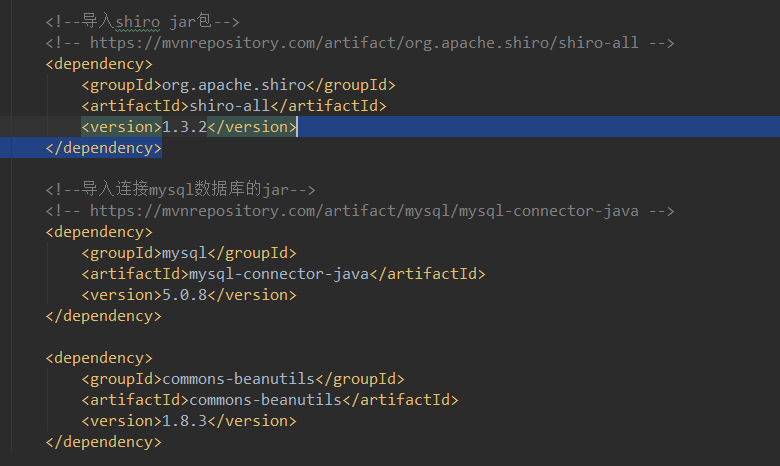
第二步：

修改shiro.ini：为DatabaseRealm 指定credentialsMatcher，其中就指定了算法是 md5, 次数为2，storedCredentialsHexEncoded 这个表示计算之后以密文为16进制。这样Shiro就拿着在subject.login() 时传入的UsernamePasswordToken 中的**源密码**、**据库里的密文**和**盐巴**，以及**配置文件里指定的算法参数**，自己去进行相关匹



下面开始写项目的步骤：

### 0、pom依赖



完整的pom.xml 文件

|  |
| --- |
| <?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>com.wlc</groupId>  <artifactId>shiroDemoMysqlSalt20191023</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>war</packaging>   <name>shiroDemoMysqlSalt20191023 Maven Webapp</name>  <!-- *FIXME change it to the project's website* -->  <url>http://www.example.com</url>   <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <maven.compiler.source>1.7</maven.compiler.source>  <maven.compiler.target>1.7</maven.compiler.target>  </properties>   <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  <scope>test</scope>  </dependency>   <!--导入shiro jar包-->  <!-- https://mvnrepository.com/artifact/org.apache.shiro/shiro-all -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-all</artifactId>  <version>1.3.2</version>  </dependency>   <!--导入连接mysql数据库的jar-->  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.0.8</version>  </dependency>   <dependency>  <groupId>commons-beanutils</groupId>  <artifactId>commons-beanutils</artifactId>  <version>1.8.3</version>  </dependency>    </dependencies>   <build>  <finalName>shiroDemoMysqlSalt20191023</finalName>  <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->  <plugins>  <plugin>  <artifactId>maven-clean-plugin</artifactId>  <version>3.1.0</version>  </plugin>  <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_war\_packaging -->  <plugin>  <artifactId>maven-resources-plugin</artifactId>  <version>3.0.2</version>  </plugin>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.8.0</version>  </plugin>  <plugin>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.22.1</version>  </plugin>  <plugin>  <artifactId>maven-war-plugin</artifactId>  <version>3.2.2</version>  </plugin>  <plugin>  <artifactId>maven-install-plugin</artifactId>  <version>2.5.2</version>  </plugin>  <plugin>  <artifactId>maven-deploy-plugin</artifactId>  <version>2.8.2</version>  </plugin>  </plugins>  </pluginManagement>  </build> </project> |

### 1、DAO.java

在com.wlc.dao下面新建DAO.java 文件

|  |
| --- |
| import com.wlc.po.User; import org.apache.shiro.crypto.SecureRandomNumberGenerator; import org.apache.shiro.crypto.hash.SimpleHash;  import java.sql.\*; import java.util.HashSet; import java.util.Set;  */\*\*  \* describe:  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/24  \*/* public class DAO\_Old {   public DAO\_Old() {  try {  Class.*forName*("com.mysql.jdbc.Driver");  } catch (ClassNotFoundException e) {  e.printStackTrace();  }  }   */\*\*  \* 获取数据库连接  \*\*/* public Connection getConnection() throws SQLException {  return DriverManager.*getConnection*("jdbc:mysql://127.0.0.1:3306/shiro02?characterEncoding=UTF-8", "scott", "tiger");  }   */\*\*  \* 注册用户  \*\*/* public int createUser(String name, String password) {  int result = 0;  try {  Connection connection = getConnection();  String sql = "insert into user values(null,?,?,?)";  PreparedStatement preparedStatement = connection.prepareStatement(sql);  String salt = new SecureRandomNumberGenerator().nextBytes().toString();  int time = 2;  String encodePassword = new SimpleHash("md5", password, salt, time).toString();  preparedStatement.setString(1, name);  preparedStatement.setString(2, encodePassword);  preparedStatement.setString(3, salt);  result = preparedStatement.executeUpdate();  } catch (SQLException e) {  e.printStackTrace();  }  return result;  }   */\*\*  \* 根据用户名获取用户的所有信息，包括盐和密码  \* \*\*/* public User getUser(String userName){  User user = null;  try {  Connection connection = getConnection();  String sql = "select name,password,salt from user where name=?";  PreparedStatement preparedStatement = connection.prepareStatement(sql);  preparedStatement.setString(1,userName);  ResultSet resultSet = preparedStatement.executeQuery();  while (resultSet.next()){  user = new User();  user.setName(resultSet.getString("name"));  user.setPassword(resultSet.getString("password"));  user.setSalt(resultSet.getString("salt"));  }  } catch (SQLException e) {  e.printStackTrace();  }  return user;  }   */\*\*  \* 根据用户名 获取用户名密码  \*\*/* public String getPassword(String userName) {  try {  Connection connection = getConnection();  String sql = "select password from user where name =?";  PreparedStatement preparedStatement = connection.prepareStatement(sql);  preparedStatement.setString(1, userName);  ResultSet resultSet = preparedStatement.executeQuery();  while (resultSet.next()) {  return resultSet.getString("password");  }  } catch (SQLException e) {  e.printStackTrace();  }  return null;  }   */\*\*  \* 根据用户名获取用户的角色  \*\*/* public Set<String> listRole(String userName) {  Set<String> roleSet = new HashSet<>();  String sql = "select r.name from user u left join user\_role ur on ur.uid = u.id left join role r on ur.rid = r.id where u.name = ?";  roleSet = commonSqlPre(sql, userName, roleSet);  return roleSet;  }   */\*\*  \* 公共部分的查询，抽出到一个方法中，提高重用性  \*\*/* public Set<String> commonSqlPre(String sql, String userName, Set<String> permissionSet) {  try {  Connection connection = getConnection();  PreparedStatement preparedStatement = connection.prepareStatement(sql);  preparedStatement.setString(1, userName);  ResultSet resultSet = preparedStatement.executeQuery();  while (resultSet.next()) {  permissionSet.add(resultSet.getString(1));  }  } catch (SQLException e) {  e.printStackTrace();  }  return permissionSet;  }   */\*\*  \* 根据用户名，获取用户所具有的权限  \*\*/* public Set<String> listPermissions(String userName) {  Set<String> permissionSet = new HashSet<>();  String sql = "SELECT p.name FROM USER u LEFT JOIN user\_role ur ON ur.uid = u.id left join role r on ur.rid = r.id LEFT JOIN role\_permission rp ON rp.rid = ur.rid LEFT JOIN permission p ON rp.pid = p.id where u.name=?";  permissionSet = commonSqlPre(sql, userName, permissionSet);  return permissionSet;  }   public static void main(String[] args) {  DAO\_Old dao = new DAO\_Old();  //int result = dao.createUser("张三","1");  // System.out.println("新增结果为： " + result);  System.*out*.println(dao.getUser("张三"));  }  } |

### 2、User.java

com.wlc.po下面新建User.java

|  |
| --- |
| public class User {  private int id ;  private String name ;  private String password ;  private String salt ;   public User() {  }   public User(int id, String name, String password, String salt) {  this.id = id;  this.name = name;  this.password = password;  this.salt = salt;  }   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  }   public String getSalt() {  return salt;  }   public void setSalt(String salt) {  this.salt = salt;  }   @Override  public String toString() {  return "User{" +  "id=" + id +  ", name='" + name + '\'' +  ", password='" + password + '\'' +  ", salt='" + salt + '\'' +  '}';  } } |

#### 第一种shiro 的认证方式：

##### 1、DatabaseRealm.java

com.wlc.realm下面新建DatabaseRealm.java文件

|  |
| --- |
| import com.wlc.dao.DAO; import com.wlc.dao.DAO\_Old; import com.wlc.po.User; import org.apache.shiro.authc.\*; import org.apache.shiro.authz.AuthorizationInfo; import org.apache.shiro.authz.SimpleAuthorizationInfo; import org.apache.shiro.crypto.hash.SimpleHash; import org.apache.shiro.realm.AuthorizingRealm; import org.apache.shiro.subject.PrincipalCollection;  import java.util.Set;  */\*\*  \* describe:  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/24  \*/* public class DatabaseRealm extends AuthorizingRealm {  */\*\*  \* 授权  \*\*/* @Override  protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {  //能进入到这里就说明已经登录成功了  System.*out*.println("可以进入到这里就说明登录成功了");  //获取用户名  String userName = principalCollection.getPrimaryPrincipal().toString();   //获取用户名对应的角色和权限  Set<String> roleSet = new DAO().listRoles(userName);  Set<String> permitSet = new DAO().listPermissions(userName);   //授权对象  SimpleAuthorizationInfo simpleAuthorizationInfo = new SimpleAuthorizationInfo();  simpleAuthorizationInfo.setRoles(roleSet);  simpleAuthorizationInfo.setStringPermissions(permitSet);  return simpleAuthorizationInfo;  }    */\*\*  \* 认证  \*\*/* @Override  protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken authenticationToken) throws AuthenticationException {  //获取用户的账号和密码  UsernamePasswordToken token = (UsernamePasswordToken) authenticationToken;  String userName = token.getPrincipal().toString();  String password = new String(token.getPassword());   //获取数据库中的密码  User user = new DAO\_Old().getUser(userName);  String passwordInDb = user.getPassword();  String salt = user.getSalt();  //对账号的密码再次加密  String passwordEncode = new SimpleHash("md5", password, salt, 2).toString();  if (null == user || !passwordEncode.equals(passwordInDb)) {  throw new AuthenticationException();  }  //把账号和密码放入到认证信息里面，getName() 是当前Realm的继承方法,通常返回当前类名 :databaseRealm  SimpleAuthenticationInfo simpleAuthenticationInfo = new SimpleAuthenticationInfo(userName, password, getName());  return simpleAuthenticationInfo;  } } |

##### 2、shirotest.ini

在resources文件夹下面新建shirotest.ini

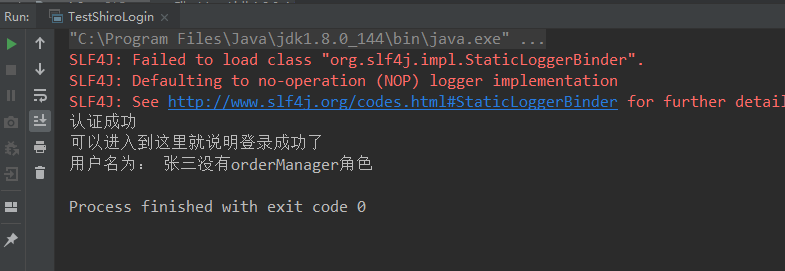
|  |
| --- |
| #[main] databaseRealm=com.wlc.realm.DatabaseRealm securityManager.realms = $databaseRealm |

##### 3、TestShiroLogin.java

在test文件夹下面新建TestShiroLogin.java

|  |
| --- |
| import org.apache.shiro.SecurityUtils; import org.apache.shiro.authc.AuthenticationException; import org.apache.shiro.authc.UsernamePasswordToken; import org.apache.shiro.config.IniSecurityManagerFactory; import org.apache.shiro.mgt.SecurityManager; import org.apache.shiro.subject.Subject; import org.apache.shiro.util.Factory;   */\*\*  \* describe:  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/24  \*/* public class TestShiroLogin {   public Subject getSubject() {  //通过Factor工厂来获取安全管理者实例  Factory<SecurityManager> securityManagerFactory = new IniSecurityManagerFactory("classpath:shirotest.ini");  //获取安全管理者SecurityManager 实例  SecurityManager securityManager = securityManagerFactory.getInstance();  //把安全管理者SecurityManager 通过SecurityUtils 放入全局变量里面  SecurityUtils.*setSecurityManager*(securityManager);  //通过安全管理者工具 SecurityUtils获取当前对象subject  Subject subject = SecurityUtils.*getSubject*();  return subject;  }   public boolean login(User user) {  //获取当前登录对象 Subject  Subject subject = getSubject();  //判断用户是否已经登录过了，如果登录过了，就退出  if (subject.isAuthenticated()) {  subject.logout();  }  UsernamePasswordToken token = new UsernamePasswordToken(user.getName(), user.getPassword());  try {  subject.login(token);  } catch (AuthenticationException e) {  System.*out*.println("验证失败！");  e.printStackTrace();  return false;  }  return subject.isAuthenticated();  }   public boolean hasRole(String role){  Subject subject = getSubject();  return subject.hasRole(role);  }   public static void main(String[] args) {  TestShiroLogin testShiroLogin = new TestShiroLogin();  User user = new User();  user.setName("张三");  user.setPassword("1");   String role = "orderManager";   String permit="addOrder";   if(testShiroLogin.login(user)){  System.*out*.println("认证成功");  if(testShiroLogin.hasRole(role)){  System.*out*.println("用户名为： " + user.getName()+ "拥有"+ role + "角色");  if(testShiroLogin.hasPermit(permit)){  System.*out*.println("用户名为： " + user.getName()+ "拥有"+ role + "角色,并且拥有 " + permit + "权限");  }else{  System.*out*.println("用户名为： " + user.getName()+ "拥有"+ role + "角色,没有拥有 " + permit + "权限");  }  }else{  System.*out*.println("用户名为： " + user.getName()+ "没有"+ role + "角色");  }  }else{  System.*out*.println("认证失败！");  }   }   private boolean hasPermit(String permit) {  Subject subject =getSubject();  return subject.isPermitted(permit);  } } |

##### 测试结果：



#### 第二种shiro 的认证方式

##### 1、DatabaseRealm2.java

在com.wlc.realm下面新建 DatabaseRealm2.java

|  |
| --- |
| import com.wlc.dao.DAO; import com.wlc.dao.DAO\_Old; import com.wlc.po.User; import org.apache.shiro.authc.\*; 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 org.apache.shiro.util.ByteSource;  import java.util.Set;  */\*\*  \* describe:  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/24  \*/* public class DatabaseRealm2 extends AuthorizingRealm {  */\*\*  \* 授权  \*\*/* @Override  protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {  //能进入到这里就说明已经登录成功了  System.*out*.println("可以进入到这里就说明登录成功了");  //获取用户名  String userName = principalCollection.getPrimaryPrincipal().toString();   //获取用户名对应的角色和权限  Set<String> roleSet = new DAO().listRoles(userName);  Set<String> permitSet = new DAO().listPermissions(userName);   //授权对象  SimpleAuthorizationInfo simpleAuthorizationInfo = new SimpleAuthorizationInfo();  simpleAuthorizationInfo.setRoles(roleSet);  simpleAuthorizationInfo.setStringPermissions(permitSet);  return simpleAuthorizationInfo;  }    */\*\*  \* 认证  \*\*/* @Override  protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken authenticationToken) throws AuthenticationException {  //获取用户的账号和密码  UsernamePasswordToken token = (UsernamePasswordToken) authenticationToken;  String userName = token.getPrincipal().toString();  String password = new String(token.getPassword());   //获取数据库中的密码  User user = new DAO\_Old().getUser(userName);  String passwordInDb = user.getPassword();  String salt = user.getSalt();   //把账号和密码放入到认证信息里面，getName() 是当前Realm的继承方法,通常返回当前类名 :databaseRealm  SimpleAuthenticationInfo simpleAuthenticationInfo =  new SimpleAuthenticationInfo(userName, passwordInDb, ByteSource.Util.*bytes*(salt), getName());  return simpleAuthenticationInfo;  } } |

##### 2、shiro2.ini

在resources 下面新建shiro2.ini

修改shiro.ini：**为DatabaseRealm 指定credentialsMatcher**，

其中就指定了算法是 md5, 次数为2，

**storedCredentialsHexEncoded** 这个表示计算之后**以密文为16进制**这样**Shiro**就拿着在subject.login() 时传入的**UsernamePasswordToken 中的源密码**， 数据库里的**密文**和**盐巴**，以及**配置文件里指定的算法参数**，自己去进行相关匹配了

|  |
| --- |
| **[main]** credentialsMatcher=org.apache.shiro.authc.credential.HashedCredentialsMatcher credentialsMatcher.hashAlgorithmName=md5 credentialsMatcher.hashIterations=2 credentialsMatcher.storedCredentialsHexEncoded=true   databaseRealm2=com.wlc.realm.DatabaseRealm2 databaseRealm2.credentialsMatcher=$credentialsMatcher securityManager.realms = $databaseRealm2 |

##### 3、TestShiroLogin2.java

在test文件下面新建TestShiroLogin2.java

|  |
| --- |
| import org.apache.shiro.SecurityUtils; import org.apache.shiro.authc.AuthenticationException; import org.apache.shiro.authc.UsernamePasswordToken; import org.apache.shiro.config.IniSecurityManagerFactory; import org.apache.shiro.mgt.SecurityManager; import org.apache.shiro.subject.Subject; import org.apache.shiro.util.Factory;   */\*\*  \* describe:  \*  \** ***@author*** *王立朝  \** ***@date*** *2019/10/24  \*/* public class TestShiroLogin2 {   public Subject getSubject() {  System.*out*.println("getSubject");  //通过Factor工厂来获取安全管理者实例  Factory<SecurityManager> securityManagerFactory =  new IniSecurityManagerFactory("classpath:shiro2.ini");  //获取安全管理者SecurityManager 实例  SecurityManager securityManager = securityManagerFactory.getInstance();  //把安全管理者SecurityManager 通过SecurityUtils 放入全局变量里面  SecurityUtils.*setSecurityManager*(securityManager);  //通过安全管理者工具 SecurityUtils获取当前对象subject  Subject subject = SecurityUtils.*getSubject*();  return subject;  }   public boolean login(User user) {  System.*out*.println("login");  //获取当前登录对象 Subject  Subject subject = getSubject();  //判断用户是否已经登录过了，如果登录过了，就退出  if (subject.isAuthenticated()) {  subject.logout();  }  UsernamePasswordToken token = new UsernamePasswordToken(user.getName(), user.getPassword());  try {  subject.login(token);  } catch (AuthenticationException e) {  System.*out*.println("验证失败！");  e.printStackTrace();  return false;  }  return subject.isAuthenticated();  }   public boolean hasRole(String role){  Subject subject = getSubject();  return subject.hasRole(role);  }   public static void main(String[] args) {  TestShiroLogin2 testShiroLogin = new TestShiroLogin2();  User user = new User();  user.setName("张三");  user.setPassword("1");   String role = "orderManager";   String permit="addOrder";   if(testShiroLogin.login(user)){  System.*out*.println("认证成功");  if(testShiroLogin.hasRole(role)){  System.*out*.println("用户名为： " + user.getName()+ "拥有"+ role + "角色");  if(testShiroLogin.hasPermit(permit)){  System.*out*.println("用户名为： " + user.getName()+ "拥有"+ role + "角色,并且拥有 " + permit + "权限");  }else{  System.*out*.println("用户名为： " + user.getName()+ "拥有"+ role + "角色,没有拥有 " + permit + "权限");  }  }else{  System.*out*.println("用户名为： " + user.getName()+ "没有"+ role + "角色");  }  }else{  System.*out*.println("认证失败！");  }   }   private boolean hasPermit(String permit) {  Subject subject =getSubject();  return subject.isPermitted(permit);  } } |

##### 测试结果：

