ssm框架搭建

## 一．Web.xml搭建

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*  xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*  id=*"WebApp\_ID"* version=*"3.1"*>  <display-name>ssm\_web</display-name>  <!-- 加载spring容器 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/applicationContext-\*.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 解决post乱码 -->  <filter>  <filter-name>CharacterEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>        <!-- springmvc的前端控制器 -->  <servlet>  <servlet-name>mvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- contextConfigLocation不是必须的， 如果不配置contextConfigLocation， springmvc的配置文件默认在：WEB-INF/servlet的name+"-servlet.xml" -->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc/springmvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>mvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>    <welcome-file-list>  <welcome-file>login</welcome-file>  </welcome-file-list>  </web-app> |

## Spring搭建

1.applicationContext-dao.cml配置

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  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-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*  *http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>  <!-- 数据库连接池 -->  <!-- 加载配置文件 -->  <context:property-placeholder location=*"classpath:resouces/db.properties"* />  <!-- 数据库连接池 -->  <bean id=*"dataSource"* class=*"com.alibaba.druid.pool.DruidDataSource"*  destroy-method=*"close"*>  <property name=*"url"* value=*"${jdbc.url}"* />  <property name=*"username"* value=*"${jdbc.username}"* />  <property name=*"password"* value=*"${jdbc.password}"* />  <property name=*"driverClassName"* value=*"${jdbc.driver}"* />  <property name=*"maxActive"* value=*"10"* />  <property name=*"minIdle"* value=*"5"* />  </bean>  <!-- 配置sqlsessionFactory -->  <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <property name=*"mapperLocations"* value=*"classpath:mapping/\*/\*.xml"*></property>  <property name=*"configLocation"* value=*"classpath:mybatis/SqlMapConfig.xml"*></property>  <property name=*"dataSource"* ref=*"dataSource"*></property>  </bean>  <!-- 配置扫描包，加载mapper代理对象 -->  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <property name=*"basePackage"* value=*"com.sunshine.mapper"*></property>  <property name=*"sqlSessionFactoryBeanName"* value=*"sqlSessionFactory"*></property>  </bean>  </beans> |

1. applicationContext-service.xml配置

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  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-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*  *http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>  <!-- 扫描包加载Service实现类 -->  <context:component-scan base-package=*"com.sunshine.service"*></context:component-scan>  </beans> |

1. applicationContext-trans.xml配置

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  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-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*  *http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>  <!-- 事务管理器 -->  <bean id=*"transactionManager"*  class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>  <!-- 数据源 -->  <property name=*"dataSource"* ref=*"dataSource"* />  </bean>  <!-- 通知 -->  <tx:advice id=*"txAdvice"* transaction-manager=*"transactionManager"*>  <tx:attributes>  <!-- 传播行为 -->  <tx:method name=*"save\*"* propagation=*"REQUIRED"* />  <tx:method name=*"insert\*"* propagation=*"REQUIRED"* />  <tx:method name=*"add\*"* propagation=*"REQUIRED"* />  <tx:method name=*"create\*"* propagation=*"REQUIRED"* />  <tx:method name=*"delete\*"* propagation=*"REQUIRED"* />  <tx:method name=*"update\*"* propagation=*"REQUIRED"* />  <tx:method name=*"find\*"* propagation=*"SUPPORTS"* read-only=*"true"* />  <tx:method name=*"select\*"* propagation=*"SUPPORTS"* read-only=*"true"* />  <tx:method name=*"get\*"* propagation=*"SUPPORTS"* read-only=*"true"* />  </tx:attributes>  </tx:advice>  <!-- 切面 -->  <aop:config>  <aop:advisor advice-ref=*"txAdvice"*  pointcut=*"execution(\* com.sunshine.service.\*.\*(..))"* />  </aop:config>  </beans> |

1. applicationContext-redis.xml配置

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:jee=*"http://www.springframework.org/schema/jee"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-4.0.xsd"*>  <context:property-placeholder location=*"classpath:resouces/db.properties"* />  <!-- jedis 配置 -->  <bean id=*"poolConfig"* class=*"redis.clients.jedis.JedisPoolConfig"*>  <property name=*"maxIdle"* value=*"${redis.maxIdle}"*/>  <property name=*"maxWaitMillis"* value=*"${redis.maxWait}"*/>  <property name=*"testOnBorrow"* value=*"${redis.testOnBorrow}"*/>  </bean>  <bean id=*"connectionFactory"* class=*"org.springframework.data.redis.connection.jedis.JedisConnectionFactory"*>  <property name=*"poolConfig"* ref=*"poolConfig"*/>  <property name=*"port"* value=*"${redis.port}"*/>  <property name=*"hostName"* value=*"${redis.host}"*/>  <property name=*"password"* value=*"${redis.pass}"*/>  <property name=*"timeout"* value=*"${redis.timeout}"*></property>  </bean>  <bean id=*"redisTemplate"* class=*"org.springframework.data.redis.core.RedisTemplate"*>  <property name=*"connectionFactory"* ref=*"connectionFactory"* />  <property name=*"keySerializer"*>  <bean  class=*"org.springframework.data.redis.serializer.StringRedisSerializer"* />  </property>  <property name=*"valueSerializer"*>  <bean  class=*"org.springframework.data.redis.serializer.JdkSerializationRedisSerializer"* />  </property>  </bean>  <!-- cache配置 -->  <bean id=*"methodCacheInterceptor"* class=*"com.sunshine.intercept.MethodCacheInterceptor"*>  <property name=*"redisUtil"* ref=*"redisUtil"* />  </bean>  <bean id=*"redisUtil"* class=*"com.sunshine.util.RedisUtil"*>  <property name=*"redisTemplate"* ref=*"redisTemplate"* />  </bean>  <bean id=*"dataSourceExchange"* class=*"com.sunshine.util.DataSourceExchange"*/>  <!--配置切面拦截方法 -->  <aop:config proxy-target-class=*"true"*>  <!--将com.crossoverJie.service包下的所有select开头的方法加入拦截 去掉select则加入所有方法w -->  <aop:pointcut id=*"controllerMethodPointcut"*  expression=*"*  *execution(\* com.sunshine.service.\*.\*(..))"* />  <aop:pointcut id=*"selectMethodPointcut"*  expression=*"*  *execution(\* com.sunshine.mapper.\*Mapper.\*(..))"* />  <aop:advisor advice-ref=*"methodCacheInterceptor"*  pointcut-ref=*"controllerMethodPointcut"* />    <!--所有数据库操作的方法加入切面-->  <aop:aspect ref=*"dataSourceExchange"*>  <aop:pointcut id=*"dataSourcePointcut"* expression=*"execution(\* com.sunshine.service.\*.\*(..))"*/>  <aop:before pointcut-ref=*"dataSourcePointcut"* method=*"before"*/>  <aop:after pointcut-ref=*"dataSourcePointcut"* method=*"after"*/>  </aop:aspect>  </aop:config>  </beans> |

其中需要加入几个类：

1. 加入com.sunshine.intercept.MethodCacheInterceptor类：

|  |
| --- |
| **package** com.sunshine.intercept;  **import** com.sunshine.util.RedisUtil;  **import** org.aopalliance.intercept.MethodInterceptor;  **import** org.aopalliance.intercept.MethodInvocation;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** java.util.ArrayList;  **import** java.util.List;  /\*\*  \* Created by chenjie on 2016/11/16.  \*/  **public** **class** MethodCacheInterceptor **implements** MethodInterceptor {  **private** Logger logger = LoggerFactory.*getLogger*(MethodCacheInterceptor.**class**);  **private** RedisUtil redisUtil;  **private** List<String> targetNamesList; // 不加入缓存的service名称  **private** List<String> methodNamesList; // 不加入缓存的方法名称  **private** Long defaultCacheExpireTime; // 缓存默认的过期时间  **private** Long xxxRecordManagerTime; //  **private** Long xxxSetRecordManagerTime; //  /\*\*  \* 初始化读取不需要加入缓存的类名和方法名称  \*/  **public** MethodCacheInterceptor() {  **try** {  // 分割字符串 这里没有加入任何方法  String[] targetNames = {};  String[] methodNames = {};  // 加载过期时间设置  defaultCacheExpireTime = 3600L;  xxxRecordManagerTime = 60L;  xxxSetRecordManagerTime = 60L;  // 创建list  targetNamesList = **new** ArrayList<String>(targetNames.length);  methodNamesList = **new** ArrayList<String>(methodNames.length);  Integer maxLen = targetNames.length > methodNames.length ? targetNames.length  : methodNames.length;  // 将不需要缓存的类名和方法名添加到list中  **for** (**int** i = 0; i < maxLen; i++) {  **if** (i < targetNames.length) {  targetNamesList.add(targetNames[i]);  }  **if** (i < methodNames.length) {  methodNamesList.add(methodNames[i]);  }  }  } **catch** (Exception e) {  e.printStackTrace();  }  }    /\*\*  \* 是否加入缓存  \*  \* **@return**  \*/  **private** **boolean** isAddCache(String targetName, String methodName) {  **boolean** flag = **true**;  **if** (targetNamesList.contains(targetName)  || methodNamesList.contains(methodName)) {  flag = **false**;  }  **return** flag;  }  /\*\*  \* 创建缓存key  \*  \* **@param** targetName  \* **@param** methodName  \* **@param** arguments  \*/  **private** String getCacheKey(String targetName, String methodName,  Object[] arguments) {  StringBuffer sbu = **new** StringBuffer();  sbu.append(targetName).append("\_").append(methodName);  **if** ((arguments != **null**) && (arguments.length != 0)) {  **for** (**int** i = 0; i < arguments.length; i++) {  sbu.append("\_").append(arguments[i]);  }  }  **return** sbu.toString();  }  **public** **void** setRedisUtil(RedisUtil redisUtil) {  **this**.redisUtil = redisUtil;  }  **public** Object invoke(MethodInvocation invocation) **throws** Throwable {  Object value = **null**;  String targetName = invocation.getThis().getClass().getName();  String methodName = invocation.getMethod().getName();  // 不需要缓存的内容  //if (!isAddCache(StringUtil.subStrForLastDot(targetName), methodName)) {  **if** (!isAddCache(targetName, methodName)) {  // 执行方法返回结果  **return** invocation.proceed();  }  Object[] arguments = invocation.getArguments();  String key = getCacheKey(targetName, methodName, arguments);  logger.debug("redisKey: " + key);  **try** {  // 判断是否有缓存  **if** (redisUtil.exists(key)) {  **return** redisUtil.get(key);  }  // 写入缓存  value = invocation.proceed();  **if** (value != **null**) {  **final** String tkey = key;  **final** Object tvalue = value;  **new** Thread(**new** Runnable() {  **public** **void** run() {  **if** (tkey.startsWith("com.service.impl.xxxRecordManager")) {  redisUtil.set(tkey, tvalue, xxxRecordManagerTime);  } **else** **if** (tkey.startsWith("com.service.impl.xxxSetRecordManager")) {  redisUtil.set(tkey, tvalue, xxxSetRecordManagerTime);  } **else** {  redisUtil.set(tkey, tvalue, defaultCacheExpireTime);  }  }  }).start();  }  } **catch** (Exception e) {  e.printStackTrace();  **if** (value == **null**) {  **return** invocation.proceed();  }  }  **return** value;  }  } |

1. 加入com.sunshine.util.RedisUtil

|  |
| --- |
| **package** com.sunshine.util;  **import** org.apache.log4j.Logger;  **import** org.springframework.data.redis.core.RedisTemplate;  **import** org.springframework.data.redis.core.ValueOperations;  **import** java.io.Serializable;  **import** java.util.Set;  **import** java.util.concurrent.TimeUnit;  /\*\*  \* Created by chenjie on 2016/11/14.  \*/  **public** **class** RedisUtil {  **private** Logger logger = Logger.*getLogger*(RedisUtil.**class**);  **private** RedisTemplate<Serializable, Object> redisTemplate;  /\*\*  \* 批量删除对应的value  \*  \* **@param** keys  \*/  **public** **void** remove(**final** String... keys) {  **for** (String key : keys) {  remove(key);  }  }  /\*\*  \* 批量删除key  \*  \* **@param** pattern  \*/  **public** **void** removePattern(**final** String pattern) {  Set<Serializable> keys = redisTemplate.keys(pattern);  **if** (keys.size() > 0)  redisTemplate.delete(keys);  }  /\*\*  \* 删除对应的value  \*  \* **@param** key  \*/  **public** **void** remove(**final** String key) {  **if** (exists(key)) {  redisTemplate.delete(key);  }  }  /\*\*  \* 判断缓存中是否有对应的value  \*  \* **@param** key  \* **@return**  \*/  **public** **boolean** exists(**final** String key) {  **return** redisTemplate.hasKey(key);  }  /\*\*  \* 读取缓存  \*  \* **@param** key  \* **@return**  \*/  **public** Object get(**final** String key) {  Object result = **null**;  ValueOperations<Serializable, Object> operations = redisTemplate  .opsForValue();  result = operations.get(key);  **return** result;  }  /\*\*  \* 写入缓存  \*  \* **@param** key  \* **@param** value  \* **@return**  \*/  **public** **boolean** set(**final** String key, Object value) {  **boolean** result = **false**;  **try** {  ValueOperations<Serializable, Object> operations = redisTemplate  .opsForValue();  operations.set(key, value);  result = **true**;  } **catch** (Exception e) {  e.printStackTrace();  }  **return** result;  }  /\*\*  \* 写入缓存  \*  \* **@param** key  \* **@param** value  \* **@return**  \*/  **public** **boolean** set(**final** String key, Object value, Long expireTime) {  **boolean** result = **false**;  **try** {  ValueOperations<Serializable, Object> operations = redisTemplate  .opsForValue();  operations.set(key, value);  redisTemplate.expire(key, expireTime, TimeUnit.***SECONDS***);  result = **true**;  } **catch** (Exception e) {  e.printStackTrace();  }  **return** result;  }  **public** **void** setRedisTemplate(  RedisTemplate<Serializable, Object> redisTemplate) {  **this**.redisTemplate = redisTemplate;  }  } |

1. 加入com.sunshine.util.DataSourceExchange

|  |
| --- |
| **package** com.sunshine.util;  **import** org.aspectj.lang.JoinPoint;  /\*\*  \* Function:拦截器方法  \*  \* **@author** chenjiec  \* Date: 2017/1/3 上午12:34  \* **@since** JDK 1.7  \*/  **public** **class** DataSourceExchange {  /\*\*  \* **@param** point  \*/  **public** **void** before(JoinPoint point) {  //获取目标对象的类类型  Class<?> aClass = point.getTarget().getClass();  //获取包名用于区分不同数据源  String whichDataSource = aClass.getName().substring(25, aClass.getName().lastIndexOf("."));  **if** ("ssmone".equals(whichDataSource)) {  DataSourceHolder.*setDataSources*(Constants.***DATASOURCE\_ONE***);  } **else** {  DataSourceHolder.*setDataSources*(Constants.***DATASOURCE\_TWO***);  }  }  /\*\*  \* 执行后将数据源置为空  \*/  **public** **void** after() {  DataSourceHolder.*setDataSources*(**null**);  }  } |

1. 加入DataSourceHolder类：

|  |
| --- |
| **package** com.sunshine.util;  /\*\*  \* Function:动态数据源  \*  \* **@author** chenjiec  \* Date: 2017/1/2 上午12:19  \* **@since** JDK 1.7  \*/  **public** **class** DataSourceHolder {  **private** **static** **final** ThreadLocal<String> ***dataSources*** = **new** ThreadLocal<String>();  **public** **static** **void** setDataSources(String dataSource) {  ***dataSources***.set(dataSource);  }  **public** **static** String getDataSources() {  **return** ***dataSources***.get();  }  } |

1. 加入Constants类：

|  |
| --- |
| **package** com.sunshine.util;  /\*\*  \* Function:  \*  \* **@author** chenjiec  \* Date: 2017/1/2 下午9:17  \* **@since** JDK 1.7  \*/  **public** **class** Constants {  **public** **final** **static** String ***DATASOURCE\_ONE*** = "ssm1DataSource";  **public** **final** **static** String ***DATASOURCE\_TWO*** = "ssm2DataSource";  } |

1. POM.XML

|  |
| --- |
| <!-- redis -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.7.3</version>  </dependency>  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-redis</artifactId>  <version>1.6.1.RELEASE</version>  </dependency> |

1. applicationContext-security.xml配置
2. applicationContext-security.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans:beans xmlns=*"http://www.springframework.org/schema/security"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:beans=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"*  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://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd"*>    <!--  自定义登录界面  intercept-url:被拦截或被保护的资源  default-target-url:登录成功后默认到的页面  password-parameter:登录页面的密码的name，一定要和页面上的name一样  username-parameter:登录页面用户名的name,同上  authentication-failure-url:登录失败到的页面  login-processing-rul:登录所走的url，即form表单中的action的值  -->    <http auto-config=*"true"*>  <form-login login-page=*"/login"*  default-target-url=*"/selectStudent"*  password-parameter=*"password"*  username-parameter=*"username"*  authentication-failure-url=*"/login"*  login-processing-url=*"/springsecuritylogin"*/>  <intercept-url pattern=*"/login"* access=*"IS\_AUTHENTICATED\_ANONYMOUSLY"*/>  <intercept-url pattern=*"/selectStudent"* access=*"ROLE\_USER,ROLE\_ADMIN"*/>  </http>    <beans:bean id=*"userService"* class=*"com.sunshine.springsecurity.impl.UserDetailsServiceImpl"*/>    <!-- 配置登录用户 -->  <authentication-manager alias=*"authenticationManager"*>  <authentication-provider user-service-ref=*"userService"*>  <!-- <user-service>  <user authorities="ROLE\_USER" name="guest" password="guest"/>  <user name="aaa" password="aaa" authorities="ROLE\_ADMIN"/>  </user-service> -->  </authentication-provider>  </authentication-manager>    </beans:beans> |

1. UserDetailsServiceImpl类的编写

|  |
| --- |
| package com.sunshine.springsecurity.impl;  import java.util.Collection;  import java.util.HashSet;  import java.util.List;  import java.util.Set;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.security.core.GrantedAuthority;  import org.springframework.security.core.authority.GrantedAuthorityImpl;  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;  import org.springframework.stereotype.Service;  import com.sunshine.pojo.Login;  import com.sunshine.pojo.UserRole;  import com.sunshine.service.LoginService;  import com.sunshine.service.UserRoleService;  @Service  public class UserDetailsServiceImpl implements UserDetailsService{    @Autowired  private LoginService loginService;    @Autowired  private UserRoleService userRoleService;  public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  Login login = loginService.queryByName(username);  if(login==null){  throw new UsernameNotFoundException("USERNAME NOT FOUND = " + username);  }    User user = null;    boolean enabled = true;  boolean accountNonExpired = true;  boolean credentialsNonExpired = true;    boolean accountNonLocked = true;  Collection<GrantedAuthority> authorities = userrole(login);    user = new User(username, login.getPassword(), enabled, accountNonExpired,  credentialsNonExpired, accountNonLocked, authorities);  return user;  }    @SuppressWarnings("deprecation")  private Set<GrantedAuthority> userrole(Login login){  Set<GrantedAuthority> authSet = new HashSet<GrantedAuthority>();    List<UserRole> list = userRoleService.role(login.getId());  for (UserRole user\_Role : list) {  authSet.add(new GrantedAuthorityImpl(user\_Role.getRolename()));  }    return authSet;  }  } |

1. log4j配置

|  |
| --- |
| ### set log levels ###  log4j.rootLogger = debug, stdout  ### \u8F93\u51FA\u5230\u63A7\u5236\u53F0 ###  log4j.appender.stdout = org.apache.log4j.ConsoleAppender  log4j.appender.stdout.Target = System.out  log4j.appender.stdout.layout = org.apache.log4j.PatternLayout  log4j.appender.stdout.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n  ### \u8F93\u51FA\u5230\u65E5\u5FD7\u6587\u4EF6 ###  log4j.appender.D = org.apache.log4j.DailyRollingFileAppender  log4j.appender.D.File = D:/javaproject/spring\_springmvc\_mybatis\_03/logs/log.log  log4j.appender.D.Append = true  log4j.appender.D.Threshold = DEBUG ## \u8F93\u51FADEBUG\u7EA7\u522B\u4EE5\u4E0A\u7684\u65E5\u5FD7  log4j.appender.D.layout = org.apache.log4j.PatternLayout  log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n  ### \u4FDD\u5B58\u5F02\u5E38\u4FE1\u606F\u5230\u5355\u72EC\u6587\u4EF6 ###  log4j.appender.D = org.apache.log4j.DailyRollingFileAppender  log4j.appender.D.File = D:/javaproject/spring\_springmvc\_mybatis\_03/logs/error.log ## \u5F02\u5E38\u65E5\u5FD7\u6587\u4EF6\u540D  log4j.appender.D.Append = true  log4j.appender.D.Threshold = ERROR ## \u53EA\u8F93\u51FAERROR\u7EA7\u522B\u4EE5\u4E0A\u7684\u65E5\u5FD7!!!  log4j.appender.D.layout = org.apache.log4j.PatternLayout  log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n  log4j.logger.org.springframework.security=DEBUG |

1. db.properties配置

|  |
| --- |
| jdbc.driver=com.mysql.jdbc.Driver  jdbc.url=jdbc:mysql://localhost:3306/student?characterEncoding=utf-8  jdbc.username=root  jdbc.password=root  FTP\_ADDRESS=192.168.228.137  FTP\_PORT=21  FTP\_USERNAME=ftpuser  FTP\_PASSWORD=ftpuser  FTP\_BASE\_PATH=/home/ftpuser  # Redis settings  redis.host=localhost  redis.port=6379  redis.pass=  redis.maxIdle=100  redis.maxActive=300  redis.maxWait=1000  redis.testOnBorrow=true  redis.timeout=100000  # \u4E0D\u9700\u8981\u52A0\u5165\u7F13\u5B58\u7684\u7C7B  targetNames=xxxRecordManager,xxxSetRecordManager,xxxStatisticsIdentificationManager  # \u4E0D\u9700\u8981\u7F13\u5B58\u7684\u65B9\u6CD5  methodNames=  #\u8BBE\u7F6E\u7F13\u5B58\u5931\u6548\u65F6\u95F4  com.service.impl.xxxRecordManager=60  com.service.impl.xxxSetRecordManager=60  defaultCacheExpireTime=3600  fep.local.cache.capacity=10000 |

## 三．Springmvc搭建

|  |
| --- |
| <?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-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"*>  <context:component-scan base-package=*"com.sunshine.controller"* />  <mvc:annotation-driven />  <bean  class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  <!-- 资源映射 -->  <mvc:resources location=*"/WEB-INF/css/"* mapping=*"/css/\*\*"* />  <mvc:resources location=*"/WEB-INF/js/"* mapping=*"/js/\*\*"* />    <!-- 文件上传 -->  <bean id=*"multipartResolver"* class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>  <property name=*"defaultEncoding"* value=*"utf-8"*/>  <property name=*"maxUploadSize"* value=*"1048576000"*/>  <property name=*"maxInMemorySize"* value=*"40960"*/>  </bean>  </beans> |

## 四．Mybatis搭建

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  <!-- 配置分页插件 -->  <plugins>  <plugin interceptor=*"com.github.pagehelper.PageHelper"*>  <!-- 设置数据库类型 Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL六种数据库-->  <property name=*"dialect"* value=*"mysql"*/>  </plugin>  </plugins>  </configuration> |

1. **Pom.xml配置**

|  |
| --- |
| <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>  <parent>  <groupId>com.sunshine</groupId>  <artifactId>ssm\_parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <groupId>com.sunshine</groupId>  <artifactId>ssm\_web</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <dependencies>  <!-- Apache工具组件 -->  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-lang3</artifactId>  </dependency>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-io</artifactId>  </dependency>  <dependency>  <groupId>commons-net</groupId>  <artifactId>commons-net</artifactId>  </dependency>  <!-- Jackson Json处理工具包 -->  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  </dependency>  <!-- httpclient -->  <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  </dependency>  <!-- 单元测试 -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <scope>test</scope>  </dependency>  <!-- 日志处理 -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  </dependency>  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  </dependency>  <!-- Mybatis -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  </dependency>  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  </dependency>  <dependency>  <groupId>com.github.miemiedev</groupId>  <artifactId>mybatis-paginator</artifactId>  </dependency>  <dependency>  <groupId>com.github.pagehelper</groupId>  <artifactId>pagehelper</artifactId>  </dependency>  <!-- MySql -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  </dependency>  <!-- 连接池 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  </dependency>  <!-- JSP相关 -->  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jsp-api</artifactId>  <scope>provided</scope>  </dependency>  <!-- 文件上传组件 -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  </dependency>  <!-- Redis客户端 -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  </dependency>    <dependency>  <groupId>org.springframework.security</groupId>  <artifactId>spring-security-web</artifactId>  <version>${spring.security.version}</version>  </dependency>  <dependency>  <groupId>org.springframework.security</groupId>  <artifactId>spring-security-config</artifactId>  <version>${spring.security.version}</version>  </dependency>  <dependency>  <groupId>org.springframework.security</groupId>  <artifactId>spring-security-cas</artifactId>  <version>${spring.security.version}</version>  <exclusions>  <exclusion>  <artifactId>log4j-over-slf4j</artifactId>  <groupId>org.slf4j</groupId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.springframework.security.oauth</groupId>  <artifactId>spring-security-oauth2</artifactId>  <version>2.0.7.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework.security</groupId>  <artifactId>spring-security-jwt</artifactId>  <version>1.0.4.RELEASE</version>  </dependency>    <!-- redis -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.7.3</version>  </dependency>  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-redis</artifactId>  <version>1.6.1.RELEASE</version>  </dependency>  </dependencies>    <!-- 如果不添加此节点mybatis的mapper.xml文件都会被漏掉。 -->  <build>  <resources>  <resource>  <directory>src/main/java</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  </resources>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-war-plugin</artifactId>  <version>2.6</version>  <configuration>  <attachClasses>true</attachClasses>  <webResources>  <resource>  <directory>src/main/resources</directory>  <targetPath>WEB-INF/classes</targetPath>  </resource>  <!-- <resource>  <directory>src/main/webapp/resources</directory>  <targetPath>resources</targetPath>  </resource> -->  <resource>  <directory>src/main/webapp/WEB-INF</directory>  <targetPath>WEB-INF</targetPath>  </resource>  </webResources>  </configuration>  </plugin>  </plugins>  </build>  </project> |