### Spring+SpringMVC+MyBatis整合（用户登录）

#### 15.0 编写User用户表

**创建t\_user表：**

CREATE TABLE `t\_user` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`userName` varchar(20) DEFAULT NULL,

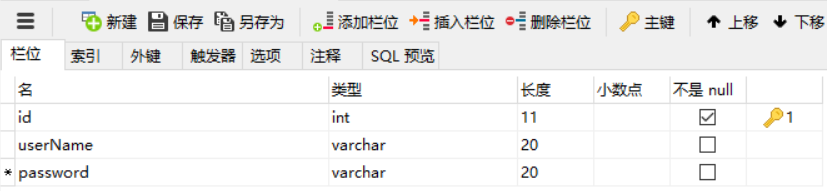
`password` varchar(20) DEFAULT NULL,

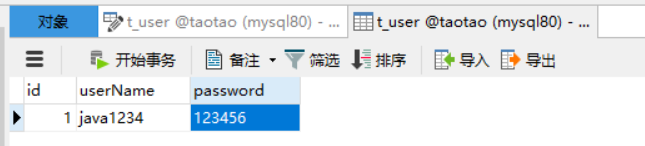
PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8;

**添加数据：**

insert into `t\_user`(`id`,`userName`,`password`) values (1,'java1234','123456');





#### 15.1 导入jar包



#### 15.2 配置web.xml文件（添加spring和SpringMVC的支持）

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance http://www.springmodules.org/schema/cache/springmodules-cache.xsd http://www.springmodules.org/schema/cache/springmodules-ehcache.xsd"* 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>MyBatisPro05</display-name>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file> //欢迎页面  </welcome-file-list>    <!-- Spring配置文件 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param>    <!-- Spring监听器 -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>    <!-- 编码过滤器 ，针对post请求乱码-->  <filter>  <filter-name>encodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <async-supported>true</async-supported>  <init-param>  <param-name>encoding</param-name>  <param-value>UTF-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>    <!-- 添加对springmvc的支持 前端控制器-->  <servlet>  <servlet-name>springMVC</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring-mvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  <async-supported>true</async-supported>  </servlet>  <servlet-mapping>  <servlet-name>springMVC</servlet-name>  <url-pattern>\*.do</url-pattern>  </servlet-mapping>  </web-app> |

#### 15.3 spring的配置文件（applicationContext.xml），配置dao和service的扫描、数据源、SqlSessionFactory、事务管理

|  |
| --- |
| <?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:aop=*"http://www.springframework.org/schema/aop"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:jee=*"http://www.springframework.org/schema/jee"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd*  *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/jee http://www.springframework.org/schema/jee/spring-jee-4.0.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd"*>    <!-- 自动扫描dao层和service层，将bean注入容器 -->  <context:component-scan base-package=*"com.java1234.dao"* />  <context:component-scan base-package=*"com.java1234.service"* />    <!-- 配置数据源 关联mysql数据库-->  <bean id=***"dataSource"***  class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>  <property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"*/>  <property name=*"url"* value=*"jdbc:mysql://localhost:3306/db\_mybatis"*/>  <property name=*"username"* value=*"root"*/>  <property name=*"password"* value=*"123456"*/>  </bean>  <!-- 配置mybatis的sqlSessionFactory -->  <bean id=***"sqlSessionFactory"*** class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <property name=*"dataSource"* ref=*"dataSource"* />  <!-- 自动扫描mappers.xml映射文件，接口和sql映射 -->  <property name=*"mapperLocations"* value=*"classpath:com/java1234/mappers/\*.xml"*></property>  <!-- mybatis配置文件 -->  <property name=*"configLocation"* value=*"classpath:mybatis-config.xml"*></property>  </bean>  <!-- DAO接口所在包名，Spring会自动查找其下的类 -->  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <property name=*"basePackage"* value=*"com.java1234.dao"* />  <property name=*"sqlSessionFactoryBeanName"* value=*"sqlSessionFactory"*></property>  </bean>  <!-- (事务管理)transaction manager, use JtaTransactionManager for global tx -->  <bean id=*"transactionManager"*  class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>  <property name=*"dataSource"* ref=*"dataSource"* />  </bean>    <!-- 配置事务通知属性 -->  <tx:advice id=*"txAdvice"* transaction-manager=*"transactionManager"*>  <!-- 定义事务传播属性 -->  <tx:attributes>  <tx:method name=*"insert\*"* propagation=*"REQUIRED"* />  <tx:method name=*"update\*"* propagation=*"REQUIRED"* />  <tx:method name=*"edit\*"* propagation=*"REQUIRED"* />  <tx:method name=*"save\*"* propagation=*"REQUIRED"* />  <tx:method name=*"add\*"* propagation=*"REQUIRED"* />  <tx:method name=*"new\*"* propagation=*"REQUIRED"* />  <tx:method name=*"set\*"* propagation=*"REQUIRED"* />  <tx:method name=*"remove\*"* propagation=*"REQUIRED"* />  <tx:method name=*"delete\*"* propagation=*"REQUIRED"* />  <tx:method name=*"change\*"* propagation=*"REQUIRED"* />  <tx:method name=*"get\*"* propagation=*"REQUIRED"* read-only=*"true"* />  <tx:method name=*"find\*"* propagation=*"REQUIRED"* read-only=*"true"* />  <tx:method name=*"load\*"* propagation=*"REQUIRED"* read-only=*"true"* />  <tx:method name=*"\*"* propagation=*"REQUIRED"* read-only=*"true"* />  </tx:attributes>  </tx:advice>    <!-- 配置事务切面 -->  <aop:config>  <aop:pointcut id=*"serviceOperation"*  expression=*"execution(\* com.java1234.service.\*.\*(..))"* />  <aop:advisor advice-ref=*"txAdvice"* pointcut-ref=*"serviceOperation"* />  </aop:config>  </beans> |

#### 15.4 spring-mvc.xml的配置（扫描controller层以及配置视图）

|  |
| --- |
| <?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:aop=*"http://www.springframework.org/schema/aop"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:jee=*"http://www.springframework.org/schema/jee"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd*  *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/jee http://www.springframework.org/schema/jee/spring-jee-4.0.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd"*>  <!-- 使用注解的包，包括子集 扫描@Controller的注解-->  <context:component-scan base-package=*"com.java1234.controller"* />  <!-- 视图解析器 -->  <bean id=*"viewResolver"*  class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp"* />  <property name=*"suffix"* value=*".jsp"*></property>  </bean>  </beans> |

#### 15.5 配置mybatis-config.xml（定义别名，方便在\*mappper.xml文件中，可以不使用实体的全名）

|  |
| --- |
| <?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>  <!-- 别名 -->  <typeAliases>  <package name=*"com.java1234.entity"*/>  </typeAliases>  </configuration> |

#### 15.6 编写实体com.java1234.entity.User（entity层或者pojo层）

|  |
| --- |
| **package** com.java1234.entity;  **public** **class** User {  **private** Integer id;  **private** String userName;  **private** String password;  ......get和set方法  } |

#### 15.7 dao层接口（com.java1234.dao.UserDao）

|  |
| --- |
| **package** com.java1234.dao;  **import** com.java1234.entity.User;  **public** **interface** UserDao {  **public** User login(User user);  } |

#### 15.8 mappers层sql映射（实现dao接口到Sql的映射）

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace=*"com.java1234.dao.UserDao"*>  <resultMap type=*"User"* id=*"UserResult"*>  <result property=*"id"* column=*"id"*/>  <result property=*"userName"* column=*"userName"*/>  <result property=*"password"* column=*"password"*/>  </resultMap>    <select id=*"login"* parameterType=*"User"* resultMap=*"UserResult"*>  select \* from t\_user where userName=#{userName} and password=#{password}  </select>  </mapper> |

#### 15.9 Service层接口（com.java1234.service.UserService）

|  |
| --- |
| **package** com.java1234.service;  **import** com.java1234.entity.User;  **public** **interface** UserService {  **public** User login(User user);  } |

#### 15.10 Serivce层实现（com.java1234.service. impl. UserServiceImpl）

|  |
| --- |
| package com.java1234.service.impl;  import javax.annotation.Resource;  import org.springframework.stereotype.Service;  import com.java1234.dao.UserDao;  import com.java1234.entity.User;  import com.java1234.service.UserService;  @Service("userService")  public class UserServiceImpl implements UserService{  @Resource  private UserDao userDao;  @Override  public User login(User user) {  return userDao.login(user);  }  } |

#### 15.11 controller层实现（com.java1234.controller. UserController）

|  |
| --- |
| package com.java1234.controller;  import javax.annotation.Resource;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpSession;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import com.java1234.entity.User;  import com.java1234.service.UserService;  @Controller  @RequestMapping("/user")  public class UserController {  @Resource  private UserService userService;    @RequestMapping("/login")  public String login(User user,HttpServletRequest request){  User resultUser=userService.login(user);  if(resultUser==null){  request.setAttribute("user", user); //用于回显数据  request.setAttribute("errorMsg", "用户名或密码错误！");  return "index";  }else{  HttpSession session=request.getSession(); //用户登录成功，放到session去  session.setAttribute("currentUser", resultUser);  return "redirect:/success.jsp";  }  }  } |

#### 15.12 视图层（index.jap和success.jsp）

**index.jsp**

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>Insert title here</title>  </head>  <body>  <form action=*"*${pageContext.request.contextPath }*/user/login.do"* method=*"post"*>  userName:<input type=*"text"* name=*"userName"* value=*"*${user.userName }*"*/><br/>  password:<input type=*"password"* name=*"password"* value=*"*${user.password }*"*><br/>  <input type=*"submit"* value=*"login"*/><font color=*"red"*>${errorMsg }</font>  </form>  </body>  </html> |

**success.jsp**

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>Insert title here</title>  </head>  <body>  欢迎：${currentUser.userName }  </body>  </html> |

#### 15.13 总结

**配置文件：**

web.xml（包括spring配置、SpringMVC配置、编码过滤器配置）

——spring配置（dao和service扫描、dataSource、SqlSessionFactory、dao接口、事务配置）

——JDBC.properties配置（mysql驱动、账号、密码配置）

——MyBatis配置文件加载（别名）和\*Mapper.xml文件（sql和dao映射）加载

——SpringMVC配置（扫描controller层注解、自动注入配置、视图解析配置）

Log4j.properties的配置（控制台、输出文件，及格式）

**MVC模式的各层：**

——POJO层或者entity层（javaBean，比如User）

——Dao层接口（UserDao，定义方法）

——Mapper层（定义UserMapper.xml，映射到UserDao接口方法）

——Service层接口（UserService）

——ServiceImpl层实现（UserServiceImpl）

——Controller层实现（UserController）