## 1、SSM框架的搭建环境(Spring,SpringMVC,Mybatis)

本文使用的是maven来构建的环境，开发工具是eclipse，tomcat用的是7，jdk版本1.7

## 2、项目整合和开发的整体流程

pom.xml -> web.xml -> entity (此处是User，与数据库中的字段名相同) -> UserDao -> UserDao.xml(这里是用的mybatis映射，直接动态实现dao接口的实现类) -> UserService -> UserServiceImpl -> UserController,这就是一个大体流程，下面开始详细介绍模块完成。路径图如下：

## 搭建一个maven工程

3.1 转到 New 菜单 Other.. -> Maven -> Maven Project ，然后单击 Next

3.2 在New Maven Project向导中，选择Create a simple project并指定项目保存的目录(如：F:\worksp\Maven\webproject)，然后单击Next。

3.3 在下一个向导中，添加项目信息，选择war打包方式，然后单击Finish。

3.4 报错——[解决 web.xml is missing and <failOnMissingWebXml> is set to true 报错](https://www.cnblogs.com/dongyu666/p/7079414.html)

解决方案：右击项目——>Java EE Tools——>Generate Deployment Descriptor Stub.然后系统会在src/main/webapp/WEB\_INF文件加下创建web.xml文件。错误解决！

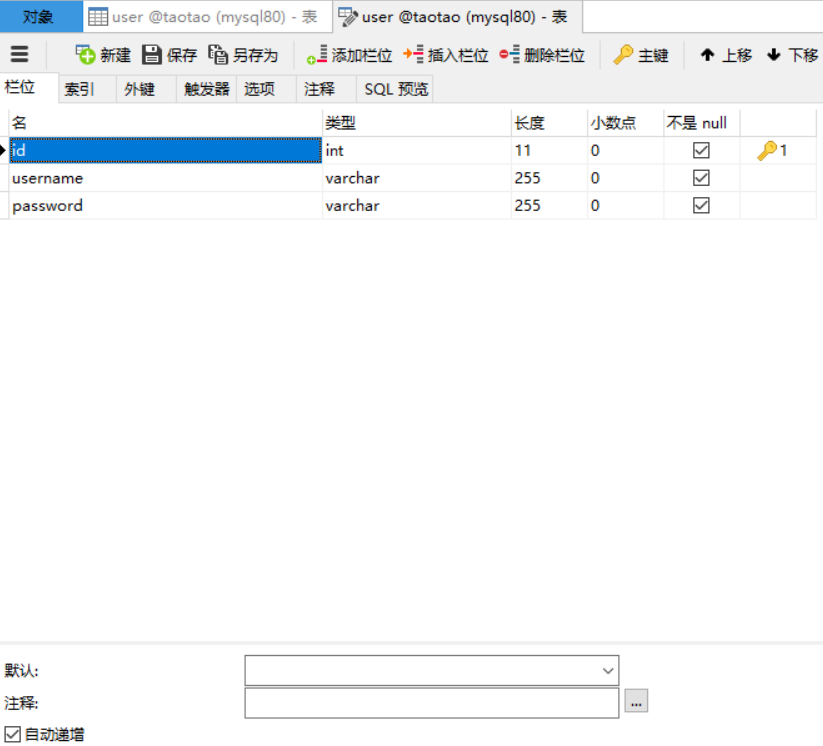
## 配置pom.xml，应用相应的jar包

|  |
| --- |
| <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.ssm.test</groupId>  <artifactId>ssm06</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>    <dependencies>  <dependency>  <!-- 使用junit4 用注解方式 -->  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  <scope>test</scope>  </dependency>  <!-- 补全项目依赖 -->  <!-- 1：日志 java日志：slf4j，log4j，logback，common-logging slf4j 是规范/接口 日志实现:log4j,logback,common-logging  使用：slf4j + logback -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.7.12</version>  </dependency>  <dependency>  <groupId>ch.qos.logback</groupId>  <artifactId>logback-core</artifactId>  <version>1.1.1</version>  </dependency>  <!-- 实现slf4j接口并整合 -->  <dependency>  <groupId>ch.qos.logback</groupId>  <artifactId>logback-classic</artifactId>  <version>1.1.1</version>  </dependency>  <!-- 2：数据库相关依赖 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.35</version>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>c3p0</groupId>  <artifactId>c3p0</artifactId>  <version>0.9.1.2</version>  </dependency>  <!-- DAO框架：MyBatis依赖 -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.3.0</version>  </dependency>  <!-- mybatis自身实现的spring整合依赖 -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.2.3</version>  </dependency>  <!--:3：servlet web相关依赖 -->  <dependency>  <groupId>taglibs</groupId>  <artifactId>standard</artifactId>  <version>1.1.2</version>  </dependency>  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.5.4</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.1.0</version>  </dependency>  <!-- 4：spring依赖 -->  <!-- 1)spring核心依赖 -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>4.2.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>4.2.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.2.0.RELEASE</version>  </dependency>  <!-- 2)spring dao层依赖 -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>4.2.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>4.2.0.RELEASE</version>  </dependency>  <!-- 3)spring web相关依赖 -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>4.2.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>4.2.0.RELEASE</version>  </dependency>  <!-- 4)spring test相关依赖 -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>4.2.0.RELEASE</version>  </dependency>  </dependencies>    <build>  <finalName>userLogin</finalName>  <plugins>  <!-- define the project compile level -->  <!-- maven编译用的版本控制，控制jdk的版本不会回跳 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>2.3.2</version>  <configuration>  <source>1.7</source>  <target>1.7</target>  </configuration>  </plugin>  </plugins>  </build>  </project> |

## 5、配置web.xml

|  |
| --- |
| <web-app xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee*  *http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*  version=*"3.1"*  metadata-complete=*"true"* >  <!-- 修改servlet版本为3.1 -->    <!-- 解决post的乱码 -->  <filter>  <filter-name>CharacterEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <!-- 初始化参数，设置编码为utf-8 -->  <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>    <!-- 配置DispatcherServlet -->  <servlet>  <servlet-name>userLogin</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- 配置springMVC需要加载的配置文件  spring-dao.xml,spring-service.xml,spring-web.xml  Mybatis -> spring -> springMVC  -->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/spring-\*.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>userLogin</servlet-name>  <!-- 默认匹配所有的请求 -->  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |

## 6、创建数据库和对应的表



## 7、创建实体类javaBean(pojo)和dao层及映射文件

使用mybatis的逆向工程实现“pojo、dao和mapper映射文件”

## Spring整合Mybaits的配置文（spring-dao.xml）

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx.xsd"*>  <!-- 配置整合mybatis过程 -->  <!-- 1:配置数据库相关参数 properties的属性:${url}-->  <context:property-placeholder location=*"classpath:jdbc.properties"*/>  <!-- 2:数据库连接池 -->  <bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*>  <!-- 配置连接池属性 -->  <property name=*"driverClass"* value=*"${jdbc.driver}"*></property>  <property name=*"jdbcUrl"* value=*"${jdbc.url}"*></property>  <property name=*"user"* value=*"${jdbc.username}"*></property>  <property name=*"password"* value=*"${jdbc.password}"*></property>    <!-- c3p0连接池的私有属性 -->  <property name=*"maxPoolSize"* value=*"30"*></property>  <property name=*"minPoolSize"* value=*"10"*></property>  <!-- 关闭连接后不自动commit -->  <property name=*"autoCommitOnClose"* value=*"false"*></property>  <!-- 获取连接超时时间 -->  <property name=*"checkoutTimeout"* value=*"1000"*></property>  <!-- 当获取连接失败重试次数 -->  <property name=*"acquireRetryAttempts"* value=*"2"*></property>  </bean>    <!-- 约定大于配置 -->  <!-- 3:配置sqlsessionfactory对象 -->  <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <!-- 注入数据库连接池 -->  <property name=*"dataSource"* ref=*"dataSource"*></property>  <!-- 配置mybatis全局配置文件:mybatis-config.xml -->  <property name=*"configLocation"* value=*"classpath:mybatis-config.xml"*></property>    <!-- 扫描entity包 使用别名 com.sy.entity.User ->User -->  <property name=*"typeAliasesPackage"* value=*"com.ssm.pojo"*></property>    <!-- 扫描sql配置文件：mapping需要的xml文件 -->  <property name=*"mapperLocations"* value=*"classpath:com/ssm/dao/\*.xml"*></property>  </bean>    <!-- 4：配置扫描Dao接口的包，动态实现Dao接口，注入到Spring容器中 -->  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <!-- 注入sqlsessionfactory -->  <property name=*"sqlSessionFactoryBeanName"* value=*"sqlSessionFactory"*></property>  <!-- 给出需要扫描的Dao接口包 -->  <property name=*"basePackage"* value=*"com.ssm.dao"*></property>  </bean>  </beans> |

## 9、Jdbc.properties

|  |
| --- |
| jdbc.driver = com.mysql.jdbc.Driver  jdbc.url = jdbc:mysql://localhost:3306/taotao?characterEncoding=utf-8  jdbc.username = root  jdbc.password = 123456 |

## 10、配置mybatis-config.xml（全局的mybatis配置文件）

|  |
| --- |
| <?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>  </configuration> |

## 编写UserService类和UserServiceInterface接口

|  |
| --- |
| **package** com.ssm.service;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** com.ssm.dao.UserMapper;  **import** com.ssm.pojo.User;  **import** com.ssm.pojo.UserExample;  **import** com.ssm.pojo.UserExample.Criteria;  **public** **class** UserService **implements** UserServiceInterface{    @Autowired  UserMapper userMapper;  **public** User findUserByNameAndPass(String username, String password) {  // **TODO** Auto-generated method stub  UserExample userExample = **new** UserExample();  Criteria criteria = userExample.createCriteria();  criteria.andUsernameEqualTo(username);  criteria.andPasswordEqualTo(password);  List<User> userList = userMapper.selectByExample(userExample);    **if**(userList.size() <= 0){  **return** **null**;  }  **return** userList.get(0);  }  } |

|  |
| --- |
| **package** com.ssm.service;  **import** com.ssm.pojo.User;  **public** **interface** UserServiceInterface {  **public** User findUserByNameAndPass(String username , String password);  } |

12、Spring整合Service层配置文件

Spring-service.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:aop="http://www.springframework.org/schema/aop"  xmlns:tx="http://www.springframework.org/schema/tx"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx.xsd">  <!-- 扫描service包下所有使用注解的类型 -->  <context:component-scan base-package="com.sy.service"/>  <!-- 配置事务管理器 -->  <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  <!-- 注入数据库的连接池 -->  <property name="dataSource" ref="dataSource"/>  </bean>    <!-- 配置基于注解的声明式事务  默认使用注解来管理事务行为  -->  <tx:annotation-driven transaction-manager="transactionManager"/>  </beans> |

1. UserController的代码编写

|  |
| --- |
| **package** com.ssm.controller;  **import** javax.servlet.http.HttpSession;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.Model;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.SessionAttributes;  **import** com.ssm.pojo.User;  **import** com.ssm.service.UserService;  @Controller  @RequestMapping("/user")  //这里用了@SessionAttributes，可以直接把model中的user(也就key)放入其中  //这样保证了session中存在user这个对象  @SessionAttributes("user")  **public** **class** UserController {    @Autowired  **private** UserService userServivce;    //正常访问login页面  @RequestMapping("/login")  **public** String login(){  **return** "login";  }    //表单提交过来的路径  @RequestMapping("/checkLogin")  **public** String checkLogin(User user,Model model){  //调用service方法  user = userServivce.findUserByNameAndPass(user.getUsername(), user.getPassword());  //若有user则添加到model里并且跳转到成功页面  **if**(user != **null**){  model.addAttribute("user",user);  **return** "success";  }  **return** "fail";  }    //测试超链接跳转到另一个页面是否可以取到session值  @RequestMapping("/anotherpage")  **public** String hrefpage(){    **return** "anotherpage";  }    //注销方法  @RequestMapping("/outLogin")  **public** String outLogin(HttpSession session){  //通过session.invalidata()方法来注销当前的session  session.invalidate();  **return** "login";  }  } |

1. Spring-web.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:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  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/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd*  *"*>  <!-- 配置springMVC -->  <!-- 1：开启springMVC注解模式 -->  <!-- 简化配置：  (1)自动注册DefaultAnnotationHandlerMapping，AnnotationMethodHandlerAdapter  (2)提供一系列：数据绑定，数字和日期的format @NumberFormat,@DataTimeFormat  xml,json默认读写支持  -->  <mvc:annotation-driven/>    <!-- 2：配置jsp显示ViewResolver -->  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"viewClass"* value=*"org.springframework.web.servlet.view.JstlView"*/>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  <!-- 3：扫描web相关的bean -->  <context:component-scan base-package=*"com.ssm.controller"* />  </beans> |

## Jsp相关页面编写

login.jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*  pageEncoding=*"utf-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <base href=*"*<%=basePath%>*"*>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <title>登录页面</title>  </head>  <body>    <form action=*"user/checkLogin"* method=*"post"*>  <table>  <tr>  <td>用户名:</td>  <td><input id=*"username"* name=*"username"* type=*"text"*></td>  </tr>  <tr>  <td>密码:</td>  <td><input id=*"password"* name=*"password"* type=*"password"*></td>  </tr>  <tr>  <td><input type=*"submit"* value=*"登录"*></td>  </tr>  </table>  </form>    </body>  </html> |

success.jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*  pageEncoding=*"utf-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <base href=*"*<%=basePath%>*"*>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <title>登录成功页面</title>  </head>  <body>  <div>  <strong> welcome,${sessionScope.user.username}! </strong>  </div>  this is success page!  <a href=*"user/anotherpage"*>点我跳到另一个页面</a>  <form action=*"user/outLogin"*>  <table>  <tr>  <td><input type=*"submit"* value=*"退出登录"*></td>  </tr>  </table>  </form>  </body>  </html> |

fail.jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*  pageEncoding=*"utf-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <base href=*"*<%=basePath%>*"*>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <title>登录失败页面</title>  </head>  <body>  this is a fail page!  </body>  </html> |

Anotherpage.jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*  pageEncoding=*"utf-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <base href=*"*<%=basePath%>*"*>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <title>其他页面</title>  </head>  <body>  <div>  <strong> ${sessionScope.user.username}!!!!! </strong>  </div>  <form action=*"user/outLogin"*>  <table>  <tr>  <td><input type=*"submit"* value=*"退出登录"* ></td>  </tr>  </table>  </form>  </body>  </html> |

13 项目报错  
Maven项目Java Resource 出现红叉

可能是因为WEB-INF目录下缺少web.xml文件，只需新建一个web类型的maven项目，把其自动生成的web.xml拷贝到当前出现红叉的项目的WEB-INF目录下即可，然后右击项目，选择Maven菜单–> Update Project，红叉就会消失