|  |
| --- |
| [公司名称] |
| 标题 |
| 副标题 |

|  |
| --- |
| 姚斯元  2018-10-30 |

目录

[一、 123 1](#_Toc3539499)

[1.1 1 1](#_Toc3539500)

[1.1.1 3 1](#_Toc3539501)

# 基本概念

## Spring

        Spring是一个开源框架，Spring是于2003 年兴起的一个轻量级的Java 开发框架，由Rod Johnson 在其著作Expert One-On-One J2EE Development and Design中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。 简单来说，Spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

## SpringMVC

        Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

## MyBatis

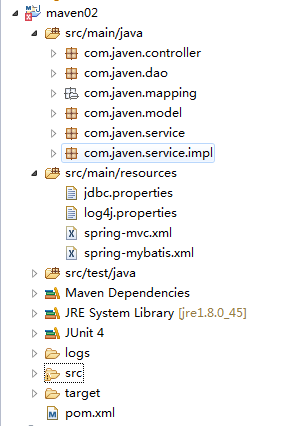
       MyBatis 本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。MyBatis是一个基于Java的持久层框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects（DAO）MyBatis 消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs（Plain Old Java Objects，普通的 Java对象）映射成数据库中的记录。

# 开发环境搭建以及创建Maven Web项目

**参看之前的博文：http://www.cnblogs.com/zyw-205520/p/4767633.html**

# SSM整合

 下面主要介绍三大框架的整合，至于环境的搭建以及项目的创建，参看上面的博文。这次整合我分了2个配置文件，分别是spring-mybatis.xml，包含spring和mybatis的配置文件，还有个是spring-mvc的配置文件，此外有2个资源文件：jdbc.propertis和log4j.properties。完整目录结构如下（最后附上源码下载地址）：



使用框架的版本：

       Spring 4.0.2 RELEASE

       Spring MVC 4.0.2 RELEASE

       MyBatis 3.2.6

## 3.1、Maven引入需要的JAR包

    在pom.xml中引入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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javen.maven01</groupId>

<artifactId>maven01</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>maven01 Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<!-- spring版本号 -->

<spring.version>4.0.2.RELEASE</spring.version>

<!-- mybatis版本号 -->

<mybatis.version>3.2.6</mybatis.version>

<!-- log4j日志文件管理包版本 -->

<slf4j.version>1.7.7</slf4j.version>

<log4j.version>1.2.17</log4j.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<!-- 表示开发的时候引入，发布的时候不会加载此包 -->

<scope>test</scope>

</dependency>

<!-- <dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

</dependency> -->

<!-- spring核心包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-oxm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- mybatis核心包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatis.version}</version>

</dependency>

<!-- mybatis/spring包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.2.2</version>

</dependency>

<!-- 导入java ee jar 包 -->

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-api</artifactId>

<version>7.0</version>

</dependency>

<!-- 导入Mysql数据库链接jar包 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.36</version>

</dependency>

<!-- 导入dbcp的jar包，用来在applicationContext.xml中配置数据库 -->

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.2.2</version>

</dependency>

<!-- JSTL标签类 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- 日志文件管理包 -->

<!-- log start -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${log4j.version}</version>

</dependency>

<!-- 格式化对象，方便输出日志 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.1.41</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${slf4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

<!-- log end -->

<!-- 映入JSON -->

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.13</version>

</dependency>

<!-- 上传组件包 -->

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.3.1</version>

</dependency>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.4</version>

</dependency>

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.9</version>

</dependency>

</dependencies>

<build>

<finalName>maven01</finalName>

<plugins>

<plugin>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<version>9.2.8.v20150217</version>

<configuration>

<httpConnector>

<port>80</port>

</httpConnector>

<stopKey>shutdown</stopKey>

<stopPort>9966</stopPort>

</configuration>

</plugin>

</plugins>

</build>

</project>

复制代码

复制代码

[复制代码](javascript:void(0);)

## 3.2、整合SpringMVC

### 3.2.1、配置spring-mvc.xml

配置里面的注释也很详细，主要是自动扫描控制器，视图模式，注解的启动这三个。

<?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-3.1.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.1.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

<!-- 自动扫描该包，使SpringMVC认为包下用了@controller注解的类是控制器 -->

<context:component-scan base-package="com.javen.controller" />

<!-- 扩充了注解驱动，可以将请求参数绑定到控制器参数 -->

<mvc:annotation-driven/>

<!-- 静态资源处理 css js imgs -->

<mvc:resources location="/resources/\*\*" mapping="/resources"/>

<!--避免IE执行AJAX时，返回JSON出现下载文件 -->

<bean id="mappingJacksonHttpMessageConverter"

class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter">

<property name="supportedMediaTypes">

<list>

<value>text/html;charset=UTF-8</value>

</list>

</property>

</bean>

<!-- 启动SpringMVC的注解功能，完成请求和注解POJO的映射 -->

<bean

class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">

<property name="messageConverters">

<list>

<ref bean="mappingJacksonHttpMessageConverter" /> <!-- JSON转换器 -->

</list>

</property>

</bean>

<!-- 配置文件上传，如果没有使用文件上传可以不用配置，当然如果不配，那么配置文件中也不必引入上传组件包 -->

<bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

<!-- 默认编码 -->

<property name="defaultEncoding" value="utf-8" />

<!-- 文件大小最大值 -->

<property name="maxUploadSize" value="10485760000" />

<!-- 内存中的最大值 -->

<property name="maxInMemorySize" value="40960" />

<!-- 启用是为了推迟文件解析，以便捕获文件大小异常 -->

<property name="resolveLazily" value="true"/>

</bean>

<!-- 配置ViewResolver 。可用多个ViewResolver 。使用order属性排序。 InternalResourceViewResolver 放在最后-->

<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">

<property name="order" value="1"></property>

<property name="mediaTypes">

<map>

<!-- 告诉视图解析器，返回的类型为json格式 -->

<entry key="json" value="application/json" />

<entry key="xml" value="application/xml" />

<entry key="htm" value="text/htm" />

</map>

</property>

<property name="defaultViews">

<list>

<!-- ModelAndView里的数据变成JSON -->

<bean class="org.springframework.web.servlet.view.json.MappingJacksonJsonView" />

</list>

</property>

<property name="ignoreAcceptHeader" value="true"></property>

</bean>

<!-- 定义跳转的文件的前后缀 ，视图模式配置-->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<!-- 这里的配置我的理解是自动给后面action的方法return的字符串加上前缀和后缀，变成一个 可用的url地址 -->

<property name="prefix" value="/WEB-INF/jsp/" />

<property name="suffix" value=".jsp" />

</bean>

</beans>

### 3.2.2、配置web.xml文件

 配置的spring-mvc的Servlet就是为了完成SpringMVC+MAVEN的整合。

web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"

version="3.0">

<display-name>Archetype Created Web Application</display-name>

<!-- Spring和mybatis的配置文件 -->

<!-- <context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-mybatis.xml</param-value>

</context-param> -->

<!-- 编码过滤器 -->

<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>

<!-- Spring监听器 -->

<!-- <listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener> -->

<!-- 防止Spring内存溢出监听器 -->

<!-- <listener>

<listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>

</listener> -->

<!-- Spring MVC servlet -->

<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>

<!-- 此处可以可以配置成\*.do，对应struts的后缀习惯 -->

<url-pattern>/</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>/index.jsp</welcome-file>

</welcome-file-list>

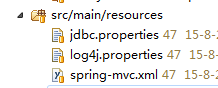
</web-app>

### 3.2.3、Log4j的配置

   为了方便调试，一般都会使用日志来输出信息，Log4j是Apache的一个开放源代码项目，通过使用Log4j，我们可以控制日志信息输送的目的地是控制台、文件、GUI组件，甚至是套接口服务器、NT的事件记录器、UNIX Syslog守护进程等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。

      Log4j的配置很简单，而且也是通用的，下面给出一个基本的配置，换到其他项目中也无需做多大的调整，如果想做调整或者想了解Log4j的各种配置，参看我转载的一篇博文，很详细：<http://blog.csdn.net/zhshulin/article/details/37937365>

下面给出配置文件目录：



log4j.properties

log4j.rootLogger=INFO,Console,File

#定义日志输出目的地为控制台

log4j.appender.Console=org.apache.log4j.ConsoleAppender

log4j.appender.Console.Target=System.out

#可以灵活地指定日志输出格式，下面一行是指定具体的格式

log4j.appender.Console.layout = org.apache.log4j.PatternLayout

log4j.appender.Console.layout.ConversionPattern=[%c] - %m%n

#文件大小到达指定尺寸的时候产生一个新的文件

log4j.appender.File = org.apache.log4j.RollingFileAppender

#指定输出目录

log4j.appender.File.File = logs/ssm.log

#定义文件最大大小

log4j.appender.File.MaxFileSize = 10MB

# 输出所以日志，如果换成DEBUG表示输出DEBUG以上级别日志

log4j.appender.File.Threshold = ALL

log4j.appender.File.layout = org.apache.log4j.PatternLayout

log4j.appender.File.layout.ConversionPattern =[%p] [%d{yyyy-MM-dd HH\:mm\:ss}][%c]%m%n

### 3.2.4、使用Jetty测试

****

package com.javen.model;

public class User {

private Integer id;

private String userName;

private String password;

private Integer age;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName == null ? null : userName.trim();

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password == null ? null : password.trim();

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

@Override

public String toString() {

return "User [id=" + id + ", userName=" + userName + ", password="

+ password + ", age=" + age + "]";

}

}

package com.javen.controller;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import com.javen.model.User;

@Controller

@RequestMapping("/user")

// /user/\*\*

public class UserController {

private static Logger log=LoggerFactory.getLogger(UserController.class);

// /user/test?id=1

@RequestMapping(value="/test",method=RequestMethod.GET)

public String test(HttpServletRequest request,Model model){

int userId = Integer.parseInt(request.getParameter("id"));

System.out.println("userId:"+userId);

User user=null;

if (userId==1) {

user = new User();

user.setAge(11);

user.setId(1);

user.setPassword("123");

user.setUserName("javen");

}

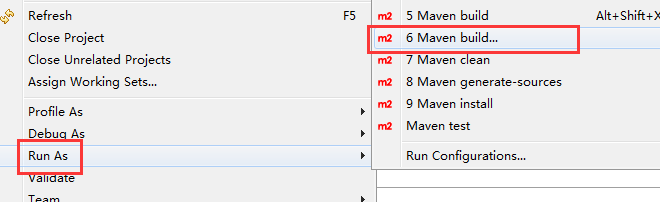
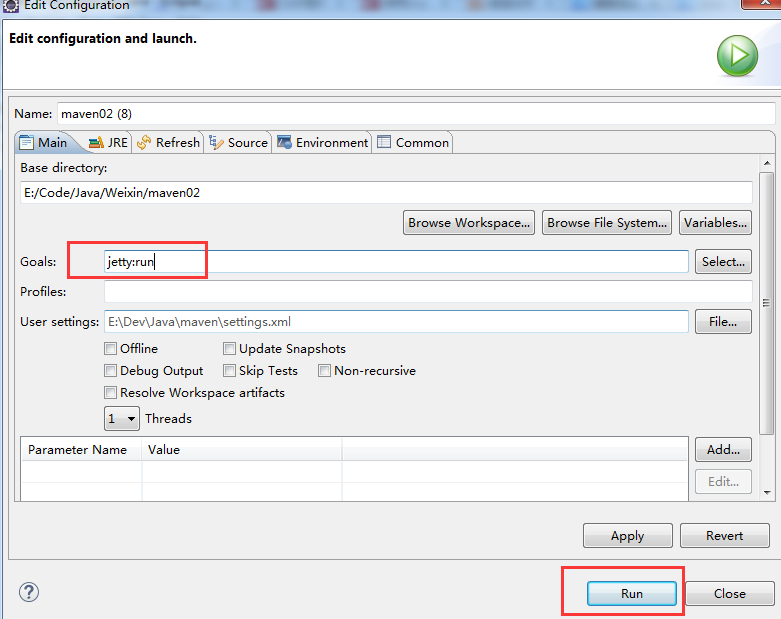
log.debug(user.toString());

model.addAttribute("user", user);

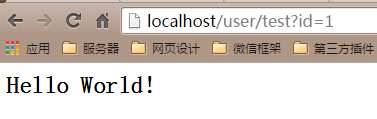
return "index";

}

}

** **

**在浏览器中输入：http://localhost/user/test?id=1**

****

**到此 SpringMVC+Maven 整合完毕**

### 3.3 Spring与MyBatis的整合

**取消3.2.2 web.xml中注释的代码**

3.3.1、建立JDBC属性文件

jdbc.properties（文件编码修改为utf-8）

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://localhost:3306/maven

username=root

password=root

#定义初始连接数

initialSize=0

#定义最大连接数

maxActive=20

#定义最大空闲

maxIdle=20

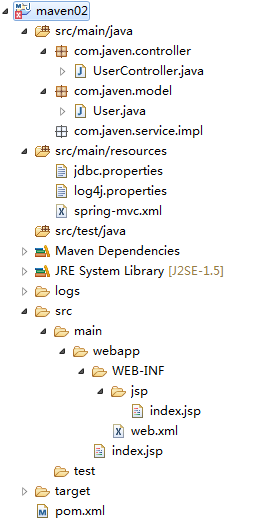
#定义最小空闲

minIdle=1

#定义最长等待时间

maxWait=60000

**此时的目录结构为**



### 3.3.2、建立spring-mybatis.xml配置文件

    这个文件就是用来完成spring和mybatis的整合的。这里面也没多少行配置，主要的就是自动扫描，自动注入，配置数据库。注释也很详细，大家看看就明白了。

spring-mybatis.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: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-3.1.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.1.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

<!-- 自动扫描 -->

<context:component-scan base-package="com.javen" />

<!-- 引入配置文件 -->

<bean id="propertyConfigurer"

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="location" value="classpath:jdbc.properties" />

</bean>

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"

destroy-method="close">

<property name="driverClassName" value="${driver}" />

<property name="url" value="${url}" />

<property name="username" value="${username}" />

<property name="password" value="${password}" />

<!-- 初始化连接大小 -->

<property name="initialSize" value="${initialSize}"></property>

<!-- 连接池最大数量 -->

<property name="maxActive" value="${maxActive}"></property>

<!-- 连接池最大空闲 -->

<property name="maxIdle" value="${maxIdle}"></property>

<!-- 连接池最小空闲 -->

<property name="minIdle" value="${minIdle}"></property>

<!-- 获取连接最大等待时间 -->

<property name="maxWait" value="${maxWait}"></property>

</bean>

<!-- spring和MyBatis完美整合，不需要mybatis的配置映射文件 -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<!-- 自动扫描mapping.xml文件 -->

<property name="mapperLocations" value="classpath:com/javen/mapping/\*.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.javen.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>

</beans>

### JUnit测试

  经过以上步骤，我们已经完成了Spring和mybatis的整合，这样我们就可以编写一段测试代码来试试是否成功了。

#### **创建测试用表**

既然我们需要测试，那么我们就需要建立在数据库中建立一个测试表，这个表建的很简单，SQL语句为：

-- ----------------------------

-- Table structure for `user\_t`

-- ----------------------------

DROP TABLE IF EXISTS `user\_t`;

CREATE TABLE `user\_t` (

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

`user\_name` varchar(40) NOT NULL,

`password` varchar(255) NOT NULL,

`age` int(4) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of user\_t

-- ----------------------------

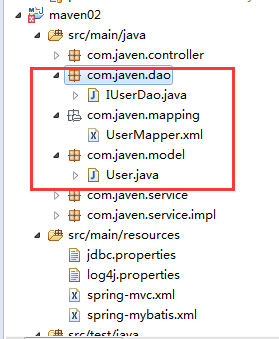
INSERT INTO `user\_t` VALUES ('1', '测试', '345', '24');

INSERT INTO `user\_t` VALUES ('2', 'javen', '123', '10');

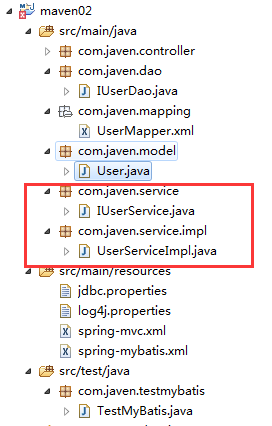
#### **利用MyBatis Generator自动创建代码**

参考博文：<http://blog.csdn.net/zhshulin/article/details/23912615>

 这个可根据表自动创建实体类、MyBatis映射文件以及DAO接口，当然，我习惯将生成的接口名改为IUserDao，而不是直接用它生成的UserMapper。如果不想麻烦就可以不改。完成后将文件复制到工程中。如图：



#### 建立Service接口和实现类

****

下面给出具体的内容：

IUserService.jave

package com.javen.service;

import com.javen.model.User;

public interface IUserService {

public User getUserById(int userId);

}

UserServiceImpl.java

package com.javen.service.impl;

import javax.annotation.Resource;

import org.springframework.stereotype.Service;

import com.javen.dao.IUserDao;

import com.javen.model.User;

import com.javen.service.IUserService;

@Service("userService")

public class UserServiceImpl implements IUserService {

@Resource

private IUserDao userDao;

public User getUserById(int userId) {

// TODO Auto-generated method stub

return this.userDao.selectByPrimaryKey(userId);

}

}

#### **建立测试类**

 测试类在src/test/java中建立，下面测试类中注释掉的部分是不使用Spring时，一般情况下的一种测试方法；如果使用了Spring那么就可以使用注解的方式来引入配置文件和类，然后再将service接口对象注入，就可以进行测试了。

       如果测试成功，表示Spring和Mybatis已经整合成功了。输出信息使用的是Log4j打印到控制台。

package com.javen.testmybatis;

import javax.annotation.Resource;

import org.apache.log4j.Logger;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.test.context.ContextConfiguration;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import com.alibaba.fastjson.JSON;

import com.javen.model.User;

import com.javen.service.IUserService;

@RunWith(SpringJUnit4ClassRunner.class) //表示继承了SpringJUnit4ClassRunner类

@ContextConfiguration(locations = {"classpath:spring-mybatis.xml"})

public class TestMyBatis {

private static Logger logger = Logger.getLogger(TestMyBatis.class);

// private ApplicationContext ac = null;

@Resource

private IUserService userService = null;

// @Before

// public void before() {

// ac = new ClassPathXmlApplicationContext("applicationContext.xml");

// userService = (IUserService) ac.getBean("userService");

// }

@Test

public void test1() {

User user = userService.getUserById(1);

// System.out.println(user.getUserName());

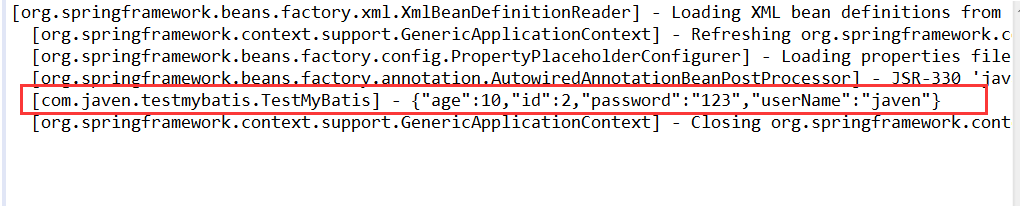
// logger.info("值："+user.getUserName());

logger.info(JSON.toJSONString(user));

}

}

测试结果



#### **3.4.5、建立UserController类**

UserController.java  控制器

package com.javen.controller;

import java.io.File;

import java.io.IOException;

import java.util.Map;

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

import org.apache.commons.io.FileUtils;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.multipart.MultipartFile;

import com.javen.model.User;

import com.javen.service.IUserService;

@Controller

@RequestMapping("/user")

// /user/\*\*

public class UserController {

private static Logger log=LoggerFactory.getLogger(UserController.class);

@Resource

private IUserService userService;

// /user/test?id=1

@RequestMapping(value="/test",method=RequestMethod.GET)

public String test(HttpServletRequest request,Model model){

int userId = Integer.parseInt(request.getParameter("id"));

System.out.println("userId:"+userId);

User user=null;

if (userId==1) {

user = new User();

user.setAge(11);

user.setId(1);

user.setPassword("123");

user.setUserName("javen");

}

log.debug(user.toString());

model.addAttribute("user", user);

return "index";

}

// /user/showUser?id=1

@RequestMapping(value="/showUser",method=RequestMethod.GET)

public String toIndex(HttpServletRequest request,Model model){

int userId = Integer.parseInt(request.getParameter("id"));

System.out.println("userId:"+userId);

User user = this.userService.getUserById(userId);

log.debug(user.toString());

model.addAttribute("user", user);

return "showUser";

}

// /user/showUser2?id=1

@RequestMapping(value="/showUser2",method=RequestMethod.GET)

public String toIndex2(@RequestParam("id") String id,Model model){

int userId = Integer.parseInt(id);

System.out.println("userId:"+userId);

User user = this.userService.getUserById(userId);

log.debug(user.toString());

model.addAttribute("user", user);

return "showUser";

}

// /user/showUser3/{id}

@RequestMapping(value="/showUser3/{id}",method=RequestMethod.GET)

public String toIndex3(@PathVariable("id")String id,Map<String, Object> model){

int userId = Integer.parseInt(id);

System.out.println("userId:"+userId);

User user = this.userService.getUserById(userId);

log.debug(user.toString());

model.put("user", user);

return "showUser";

}

// /user/{id}

@RequestMapping(value="/{id}",method=RequestMethod.GET)

public @ResponseBody User getUserInJson(@PathVariable String id,Map<String, Object> model){

int userId = Integer.parseInt(id);

System.out.println("userId:"+userId);

User user = this.userService.getUserById(userId);

log.info(user.toString());

return user;

}

// /user/{id}

@RequestMapping(value="/jsontype/{id}",method=RequestMethod.GET)

public ResponseEntity<User> getUserInJson2(@PathVariable String id,Map<String, Object> model){

int userId = Integer.parseInt(id);

System.out.println("userId:"+userId);

User user = this.userService.getUserById(userId);

log.info(user.toString());

return new ResponseEntity<User>(user,HttpStatus.OK);

}

//文件上传、

@RequestMapping(value="/upload")

public String showUploadPage(){

return "user\_admin/file";

}

@RequestMapping(value="/doUpload",method=RequestMethod.POST)

public String doUploadFile(@RequestParam("file")MultipartFile file) throws IOException{

if (!file.isEmpty()) {

log.info("Process file:{}",file.getOriginalFilename());

}

FileUtils.copyInputStreamToFile(file.getInputStream(), new File("E:\\",System.currentTimeMillis()+file.getOriginalFilename()));

return "succes";

}

}

#### **新建jsp页面**

file.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>

<h1>上传文件</h1>

<form method="post" action="/user/doUpload" enctype="multipart/form-data">

<input type="file" name="file"/>

<input type="submit" value="上传文件"/>

</form>

</body>

</html>

index.jsp

<html>

<body>

<h2>Hello World!</h2>

</body>

</html>

showUser.jsp

<%@ page language="java" import="java.util.\*" pageEncoding="utf-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<title>测试</title>

</head>

<body>

${user.userName}

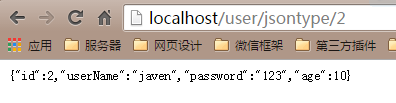
</body>

</html>

**至此，完成Spring+SpingMVC+mybatis这三大框架整合完成。**

#### 部署项目

**输入地址：http://localhost/user/jsontype/2**



  项目下载地址：<https://github.com/Javen205/SSM>