# [SSM三大框架整合详细教程（Spring+SpringMVC+MyBatis）](http://www.itnose.net/detail/6074493.html)

使用SSM（Spring、SpringMVC和Mybatis）已经有三个多月了，项目在技术上已经没有什么难点了，基于现有的技术就可以实现想要的功能，当然肯定有很多可以改进的地方。之前没有记录SSM整合的过程，这次刚刚好基于自己的一个小项目重新搭建了一次，而且比项目搭建的要更好一些。以前解决问题的过程和方法并没有及时记录，以后在自己的小项目中遇到我再整理分享一下。这次，先说说三大框架整合过程。个人认为使用框架并不是很难，关键要理解其思想，这对于我们提高编程水平很有帮助。不过，如果用都不会，谈思想就变成纸上谈兵了！！！先技术，再思想。实践出真知。（可通过图片水印查看博客地址）

1、基本概念

**1.1、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。

**1.2、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。

**1.3、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。

2、开发环境搭建

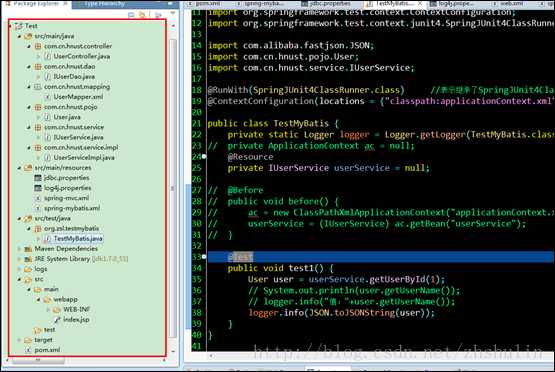
如果需要，参看之前的博文：http://blog.csdn.net/zhshulin/article/details/30779873

3、Maven Web项目创建

如果需要，参看之前的博文：http://blog.csdn.net/zhshulin/article/details/37921705

4、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：



使用框架都是较新的版本：

       Spring 4.0.2 RELEASE

       Spring MVC 4.0.2 RELEASE

       MyBatis 3.2.6

**2.1、Maven引入需要的JAR包**

         为了方便后面说的时候不需要引入JAR包，我这里直接给出所有需要的JAR包，这都是基本的JAR包，每个包的是干什么的都有注释，就不再多说了。

pom.xml

<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>4.11</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>

<!-- 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>5.1.30</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>

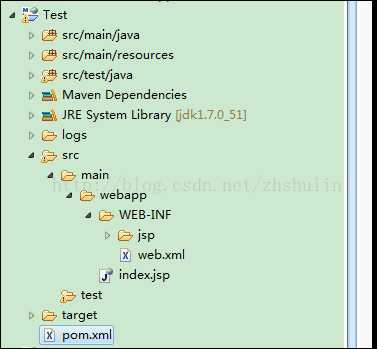
<version>1.9</version>

</dependency>

</dependencies>

**2.2、Spring与MyBatis的整合**

所有需要的JAR包都引入以后，首先进行Spring与MyBatis的整合，然后再进行JUnit测试，先看一个项目结构图：



**2.2.1、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://10.221.10.111:8080/db\_zsl

username=demao

password=demao

#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

**2.2.2、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.cn.hnust" />

<!-- 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/cn/hnust/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.cn.hnust.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>

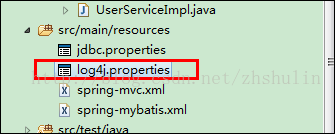
**2.2.3、Log4j的配置**

      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

#定义LOG输出级别

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

**2.2.4、JUnit测试**

      经过以上步骤（到4.2.2，log4j不配也没影响），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。

2.2.4.1、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：

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=2 DEFAULT CHARSET=utf8;

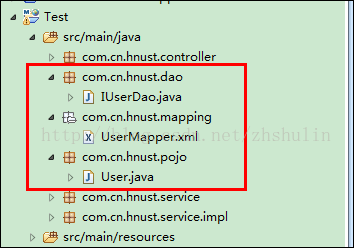
/\*Data for the table `user\_t` \*/

insert into `user\_t`(`id`,`user\_name`,`password`,`age`) values (1,'测试','sfasgfaf',24);

2.2.4.2、利用MyBatis Generator自动创建代码

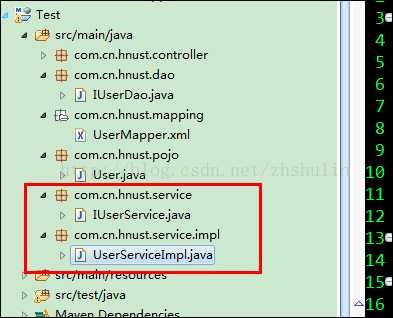
参考博文：http://blog.csdn.net/zhshulin/article/details/23912615

       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：



2.2.4.3、建立Service接口和实现类

目录结构：



下面给出具体的内容：

IUserService.jave

package com.cn.hnust.service;

import com.cn.hnust.pojo.User;

public interface IUserService {

public User getUserById(int userId);

}

UserServiceImpl.java

package com.cn.hnust.service.impl;

import javax.annotation.Resource;

import org.springframework.stereotype.Service;

import com.cn.hnust.dao.IUserDao;

import com.cn.hnust.pojo.User;

import com.cn.hnust.service.IUserService;

@Service("userService")

public class UserServiceImpl implements IUserService {

@Resource

private IUserDao userDao;

@Override

public User getUserById(int userId) {

// TODO Auto-generated method stub

return this.userDao.selectByPrimaryKey(userId);

}

}

2.2.4.4、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 org.zsl.testmybatis;

import javax.annotation.Resource;

import org.apache.log4j.Logger;

import org.junit.Before;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.test.context.ContextConfiguration;

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

import com.alibaba.fastjson.JSON;

import com.cn.hnust.pojo.User;

import com.cn.hnust.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());

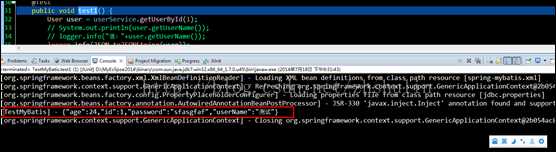
// logger.info("值："+user.getUserName());

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

}

}

测试结果：



至此，完成Spring和mybatis这两大框架的整合，下面在继续进行SpringMVC的整合。

**2.3、整合SpringMVC**

         上面已经完成了2大框架的整合，SpringMVC的配置文件单独放，然后在web.xml中配置整合。

**2.3.1、配置spring-mvc.xml**

配置里面的注释也很详细，在此就不说了，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.cn.hnust.controller" />

<!--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"><!-- MappingJackson2HttpMessageConverter -->

<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>

<!-- 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>

<!-- 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" />

</bean>

</beans>

**2.3.2、配置web.xml文件**

         这里面对spring-mybatis.xml的引入以及配置的spring-mvc的Servlet就是为了完成SSM整合，之前2框架整合不需要在此处进行任何配置。配置一样有详细注释，不多解释了。

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>

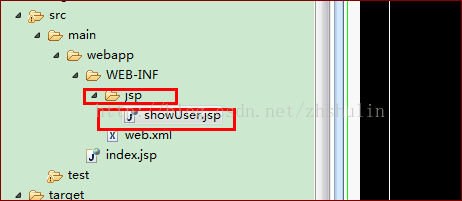
</welcome-file-list>

</web-app>

**2.3.3、测试**

        至此已经完成了SSM三大框架的整合了，接下来测试一下，如果成功了，那么恭喜你，如果失败了，继续调试吧，作为程序员就是不停的与BUG做斗争！

2.3.3.1、新建jsp页面



showUser.jsp   此页面仅输出一下用户名，完成一个完整的简单流程。

<%@ 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>

2.3.3.2、建立UserController类

UserController.java  控制器

package com.cn.hnust.controller;

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

import com.cn.hnust.pojo.User;

import com.cn.hnust.service.IUserService;

@Controller

@RequestMapping("/user")

public class UserController {

@Resource

private IUserService userService;

@RequestMapping("/showUser")

public String toIndex(HttpServletRequest request,Model model){

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

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

model.addAttribute("user", user);

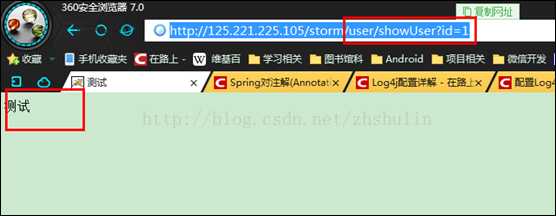
return "showUser";

}

}

2.3.3.3、部署项目

输入地址：localhost:8080/项目名称/user/showUser?id=1



至此，SSM三大框架的整合就完成了，在此基础上可再添加其他功能。

## 补充一

1. 所需要Jar包.  
  
//Spring3.0.1包  
  
org.springframework.web-3.0.1 系列  
  
//公共包  
  
slf4j-api-1.5.6.jar slf4j-log4j12-1.5.6.jar log4j-1.2.13.jar  
  
commons-logging-1.1.1.jar asm-3.1.jar cglib-2.2.jar  
  
//mybatis与Spring的整合所需的包  
  
mybatis-3.0.5.jar   
  
aopalliance-1.0.jar   
  
mybatis-spring-1.0.1.jar  
  
mybatis-generator-core-1.3.1.jar(mybatis代码生成器包)  
  
//jdbc driven  
  
mysql-connector-java-3.1.6-bin.jar  
  
//JSR验证-Hibernate validate 4.1  
  
hibernate-validator-4.1.0.Final.jar  
  
validation-api-1.0.0.GA.jar  
  
//Spring Json 支持包  
  
jackson-all-1.8.1.jar

2. web.xml配置  
  
Servlet配置  
  
org.springframework.web.servlet.DispatcherServlet  
  
init-param配置servlet初始化文件.  
  
以及servlet-mapping配置.  
  
应用路径配置  
  
webAppRootKey  
  
Log4j配置  
  
Log4jConfigLocation  
  
Log4jRefreshInterval  
  
Spring上下文配置  
  
contextConfigLocation  
  
Spring字符集过滤器配置  
  
org.springframework.web.filter.CharacterEncodingFilter  
  
Spring监听器配置  
  
org.springframework.web.context.ContextLoaderListener  
  
log4j监听器配置  
  
org.springframework.web.util.Log4jConfigListener

3. spring mvc - servlet.xml配置  
  
启动mvc注解驱动  
  
<mvc:annotation-driven/>  
  
组件scanner主要是自动去注入指定包里的对象  
  
<context:component-scan  
  
base-package="com.los.mvc.controller"/>  
  
ViewResolver & View 映射关系  
  
InternalResourceViewResolver 基于resource对jsp/jstl的支持  
  
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
  
<property name="prefix" value="/WEB-INF/views/"/>  
  
<property name="suffix" value=".jsp"/>  
  
<!-- InternalResourceViewResolver viewClass默认值就是JstlView -->  
  
<property name="viewClass" value="org.springframework.web.servlet.view.JstlView"></property>  
  
</bean>  
  
自定义拦截器配置  
  
<mvc:interceptors>  
  
<mvc:interceptor>  
  
<mvc:mapping path="/json\*"/>  
  
<bean class="com.los.mvc.interceptor.MyInterceptor"></bean>  
  
</mvc:interceptor>  
  
</mvc:interceptors>  
  
国际化配置  
  
<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">  
  
<property name="basename" value="message"></property>  
  
</bean>  
  
  
  
4. Spring上下文 -- applicationContext.xml 配置  
  
支持注解  
  
<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"></bean>  
  
<bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"></bean>  
  
<import resource="service.xml"/>  
  
<import resource="dao.xml"/>  
  
<import resource="orm.xml"/>  
  
  
  
service.xml dao.xml 配置@service 和 @Repository  
  
  
  
5. Mybatis3.0.5-Spring 整合 -- orm.xml  
  
DataSource配置  
  
<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/mvc"/>  
  
<property name="username" value="root" />  
  
<property name="password" value="root" />  
  
</bean>  
  
  
  
注册事务管理器(Mybatis将事务转交给Spring来管理)  
  
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
  
<property name="dataSource" ref="dataSource" />  
  
</bean>  
  
SqlSessionFactory配置(Mybatis核心是sqlSessionFactory来获取orm处理对象, dataSource, mapperLocations配置mybaits自动生成的xml文件.就是注入映射关系.)  
  
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
  
<property name="dataSource" ref="dataSource" />  
  
<property name="mapperLocations" value="classpath:/com/los/mvc/mapper/\*.xml" />  
  
</bean>  
  
MapperScanner配置.自动去搜索mapper里的对象,并注入.  
  
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
  
<property name="basePackage" value="com.los.mvc.dao" />  
  
</bean>  
  
启动Spring注解事务  
  
<tx:annotation-driven/>  
  
  
  
6. mybatis自动生成器配置 -- generatorConfig.xml  
  
sqlMapGenerator sqlMpper.xml生成器  
  
javaClientGenerator ModelDao生成器  
  
javaModelGenerator Model生成器  
  
com.los.util.MBG.java 运行会自动生成mybatis代码.然后再配置orm.xml  
  
  
  
7. Controller层配置  
  
类注解  
  
@Controller  
  
@RequestMapping("/json")为访问该层的路径.  
  
  
  
方法注解  
  
@RequestMapping(method = RequestMethod.GET) 只有get方法才能访问.  
  
@ResponseBody 自动将返回的封装成json,方法返回值必须是map<String,?>类型.  
  
@RequestMapping(value="/doLogin") value=”doLogin”为访问该方法的handler mapping  
  
return "login/login";会通过ViewResolver找到对应的view  
  
return "redirect:/user/toRegister.html";为spring-mvc的重定向.  
  
@InitBinder()为绑定器,可以为request传来的数据进行数据类型转换.  
  
  
  
数据自动验证  
  
方法中参数需要有后面的两个(@Valid User user,BindingResult result).@Valid的支持标准是JSR,Hibernate Validate 4是对该标准比较好的实现.需要在Model类中配置验证的注解.判断验证是否正确通过result.hasErrors()或者result.hasFieldErrors()来判断,通过result.getAllErrors()或者result.getFieldErrors()来获取Errors然后遍历Errors获取相关想要的信息,例如Error.getDeafaultMessage(),这个是获取错误信息.具体的错误验证机制还地在Model类中配置.  
  
  
  
属性注解  
  
@Autowired 会为该属性自动注入bean,默认方式是byType.也可以用@Resource这个注解默认是byName.  
  
  
  
8. Service层配置.(业务层)  
  
类注解  
  
@Service 为@Component的子注解,分工更明细.  
  
@Transactional 可以为该业务类声明一个全类的事务.也可以将事务写在方法上.根据不同的需要.  
  
  
  
方法注解  
  
@Transactional(readOnly = true)  
  
@Transactional(readOnly = false, propagation = Propagation.REQUIRED, rollbackFor = Exception.class) 类的事务声明,可以设置隔离级别和传播属性,以及要回滚的异常名或者异常类,不需要回滚的异常名或者异常类.异常通常抛出给controller层来处理  
  
  
  
属性注解  
  
@Autowired @Resource

9. Repository层配置.(持久层DaoImpl)  
  
类注解  
  
@Repository 为@Component的子注解,意为持久层,分工更明细.一般不在这层处理事务.  
  
  
  
10.Entry层配置(Model层)  
  
类注解  
  
@Entry  
  
验证注解,常用的有:  
  
@NotEmpty  
  
@NotNull  
  
@Size(min=2,max=10,message=”xx必须在{min}和{max}之间”)  
  
@Email  
  
@DecimalMax  
  
@AssertFalse @AssertTrue  
  
@Null  
  
@Valid  
  
@URL(protocol=,host=, port=,regexp=, flags=)  
  
一般情况下属性或者方法可以放多个约束注解,hibernate validate会以随机的顺序去验证这些约束.所以多个注解约束会有可能同一个属性返回多个message.所以有时候需要只返回一条message,则需要使用验证组Groups来达成.组别序列可以把一系列的组别按照一定的顺序排列在一起，然后逐个验证，只要有一个组别验证失败，就不继续验证剩余的组别。  
  
@GroupSequence({User.class,GroupB.class,GroupC.class})验证组的顺序,约束里不指定group的为默认的User.class组.  
  
约束组放在类前,User.class为默认的约束组,GroupB,GroupC为空的接口.写在User外同个java文件下.  
  
@NotEmpty(message="密码不能为空")   
  
@Size(min=4,max=20,message="密码长度必须在{min}-{max}范围内",groups = GroupB.class)  
  
如果@NotEmpty验证失败了,就不会继续验证@Size

## 配置实例演示

最近做了个maven管理的springmvc+spring+mybatis，还用到了阿里巴巴的 fastjson和druid连接池，配置文件如下

pom.xml文件

**[html]** [view plaincopy](http://blog.csdn.net/qyl445/article/details/9837579)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<project** xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"**>**
4. **<modelVersion>**4.0.0**</modelVersion>**
5. **<groupId>**com.dahafo.demo**</groupId>**
6. **<name>**um**</name>**
7. **<packaging>**war**</packaging>**
8. **<version>**1.0.0-BUILD-SNAPSHOT**</version>**
9. **<properties>**
10. **<java-version>**1.7**</java-version>**
11. **<org.springframework-version>**3.2.3.RELEASE**</org.springframework-version>**
12. **<org.aspectj-version>**1.6.10**</org.aspectj-version>**
13. **<org.slf4j-version>**1.6.6**</org.slf4j-version>**
14. **</properties>**
15. **<dependencies>**
16. <!-- Spring -->
17. **<dependency>**
18. **<groupId>**org.springframework**</groupId>**
19. **<artifactId>**spring-context**</artifactId>**
20. **<version>**${org.springframework-version}**</version>**
21. **<exclusions>**
22. <!-- Exclude Commons Logging in favor of SLF4j -->
23. **<exclusion>**
24. **<groupId>**commons-logging**</groupId>**
25. **<artifactId>**commons-logging**</artifactId>**
26. **</exclusion>**
27. **</exclusions>**
28. **</dependency>**
29. **<dependency>**
30. **<groupId>**org.springframework**</groupId>**
31. **<artifactId>**spring-webmvc**</artifactId>**
32. **<version>**${org.springframework-version}**</version>**
33. **</dependency>**
34. **<dependency>**
35. **<groupId>**org.springframework**</groupId>**
36. **<artifactId>**spring-jdbc**</artifactId>**
37. **<version>**${org.springframework-version}**</version>**
38. **</dependency>**
40. <!-- AspectJ -->
41. **<dependency>**
42. **<groupId>**org.aspectj**</groupId>**
43. **<artifactId>**aspectjrt**</artifactId>**
44. **<version>**${org.aspectj-version}**</version>**
45. **</dependency>**
47. <!-- Logging -->
48. **<dependency>**
49. **<groupId>**org.slf4j**</groupId>**
50. **<artifactId>**slf4j-api**</artifactId>**
51. **<version>**${org.slf4j-version}**</version>**
52. **</dependency>**
53. **<dependency>**
54. **<groupId>**org.slf4j**</groupId>**
55. **<artifactId>**jcl-over-slf4j**</artifactId>**
56. **<version>**${org.slf4j-version}**</version>**
57. **<scope>**runtime**</scope>**
58. **</dependency>**
59. **<dependency>**
60. **<groupId>**org.slf4j**</groupId>**
61. **<artifactId>**slf4j-log4j12**</artifactId>**
62. **<version>**${org.slf4j-version}**</version>**
63. **<scope>**runtime**</scope>**
64. **</dependency>**
65. **<dependency>**
66. **<groupId>**log4j**</groupId>**
67. **<artifactId>**log4j**</artifactId>**
68. **<version>**1.2.15**</version>**
69. **<exclusions>**
70. **<exclusion>**
71. **<groupId>**javax.mail**</groupId>**
72. **<artifactId>**mail**</artifactId>**
73. **</exclusion>**
74. **<exclusion>**
75. **<groupId>**javax.jms**</groupId>**
76. **<artifactId>**jms**</artifactId>**
77. **</exclusion>**
78. **<exclusion>**
79. **<groupId>**com.sun.jdmk**</groupId>**
80. **<artifactId>**jmxtools**</artifactId>**
81. **</exclusion>**
82. **<exclusion>**
83. **<groupId>**com.sun.jmx**</groupId>**
84. **<artifactId>**jmxri**</artifactId>**
85. **</exclusion>**
86. **</exclusions>**
87. **<scope>**runtime**</scope>**
88. **</dependency>**
90. <!-- @Inject -->
91. **<dependency>**
92. **<groupId>**javax.inject**</groupId>**
93. **<artifactId>**javax.inject**</artifactId>**
94. **<version>**1**</version>**
95. **</dependency>**
97. <!-- Servlet -->
98. **<dependency>**
99. **<groupId>**javax.servlet**</groupId>**
100. **<artifactId>**servlet-api**</artifactId>**
101. **<version>**2.5**</version>**
102. **<scope>**provided**</scope>**
103. **</dependency>**
104. **<dependency>**
105. **<groupId>**javax.servlet.jsp**</groupId>**
106. **<artifactId>**jsp-api**</artifactId>**
107. **<version>**2.1**</version>**
108. **<scope>**provided**</scope>**
109. **</dependency>**
110. **<dependency>**
111. **<groupId>**javax.servlet**</groupId>**
112. **<artifactId>**jstl**</artifactId>**
113. **<version>**1.2**</version>**
114. **</dependency>**
115. <!-- mybatis -->
116. **<dependency>**
117. **<groupId>**org.mybatis**</groupId>**
118. **<artifactId>**mybatis**</artifactId>**
119. **<version>**3.2.1**</version>**
120. **</dependency>**
121. **<dependency>**
122. **<groupId>**org.mybatis**</groupId>**
123. **<artifactId>**mybatis-spring**</artifactId>**
124. **<version>**1.2.0**</version>**
125. **</dependency>**
126. **<dependency>**
127. **<groupId>**mysql**</groupId>**
128. **<artifactId>**mysql-connector-java**</artifactId>**
129. **<version>**5.1.26**</version>**
130. **</dependency>**


134. <!-- Test -->
135. **<dependency>**
136. **<groupId>**junit**</groupId>**
137. **<artifactId>**junit**</artifactId>**
138. **<version>**4.7**</version>**
139. **<scope>**test**</scope>**
140. **</dependency>**
141. **<dependency>**
142. **<groupId>**com.alibaba**</groupId>**
143. **<artifactId>**druid**</artifactId>**
144. **<version>**0.2.20**</version>**
145. **</dependency>**
147. **<dependency>**
148. **<groupId>**org.aspectj**</groupId>**
149. **<artifactId>**aspectjweaver**</artifactId>**
150. **<version>**1.7.2**</version>**
151. **</dependency>**
152. **<dependency>**
153. **<groupId>**com.alibaba**</groupId>**
154. **<artifactId>**fastjson**</artifactId>**
155. **<version>**1.1.34**</version>**
156. **</dependency>**
157. **<dependency>**
158. **<groupId>**com.fasterxml.jackson.core**</groupId>**
159. **<artifactId>**jackson-core**</artifactId>**
160. **<version>**2.2.2**</version>**
161. **</dependency>**
162. **<dependency>**
163. **<groupId>**com.fasterxml.jackson.core**</groupId>**
164. **<artifactId>**jackson-databind**</artifactId>**
165. **<version>**2.2.2**</version>**
166. **</dependency>**
167. **<dependency>**
168. **<groupId>**commons-fileupload**</groupId>**
169. **<artifactId>**commons-fileupload**</artifactId>**
170. **<version>**1.2.2**</version>**
171. **</dependency>**
172. **</dependencies>**








182. **<build>**
183. **<plugins>**
184. **<plugin>**
185. **<artifactId>**maven-eclipse-plugin**</artifactId>**
186. **<version>**2.9**</version>**
187. **<configuration>**
188. **<additionalProjectnatures>**
189. **<projectnature>**org.springframework.ide.eclipse.core.springnature**</projectnature>**
190. **</additionalProjectnatures>**
191. **<additionalBuildcommands>**
192. **<buildcommand>**org.springframework.ide.eclipse.core.springbuilder**</buildcommand>**
193. **</additionalBuildcommands>**
194. **<downloadSources>**true**</downloadSources>**
195. **<downloadJavadocs>**true**</downloadJavadocs>**
196. **</configuration>**
197. **</plugin>**
198. **<plugin>**
199. **<groupId>**org.apache.maven.plugins**</groupId>**
200. **<artifactId>**maven-compiler-plugin**</artifactId>**
201. **<version>**2.5.1**</version>**
202. **<configuration>**
203. **<source>**1.6**</source>**
204. **<target>**1.6**</target>**
205. **<compilerArgument>**-Xlint:all**</compilerArgument>**
206. **<showWarnings>**true**</showWarnings>**
207. **<showDeprecation>**true**</showDeprecation>**
208. **</configuration>**
209. **</plugin>**
210. **<plugin>**
211. **<groupId>**org.codehaus.mojo**</groupId>**
212. **<artifactId>**exec-maven-plugin**</artifactId>**
213. **<version>**1.2.1**</version>**
214. **<configuration>**
215. **<mainClass>**org.test.int1.Main**</mainClass>**
216. **</configuration>**
217. **</plugin>**
218. **</plugins>**
219. **</build>**
220. **<artifactId>**demo-um**</artifactId>**
221. **</project>**

web.xml

**[html]** [view plaincopy](http://blog.csdn.net/qyl445/article/details/9837579)[在CODE上查看代码片](https://code.csdn.net/snippets/536921)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"**>**


8. <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
9. **<context-param>**
10. **<param-name>**contextConfigLocation**</param-name>**
11. **<param-value>**/WEB-INF/spring/root-context.xml**</param-value>**
12. **</context-param>**
13. <!-- 用spring Encoding,解决乱码问题 -->
14. **<filter>**
15. **<filter-name>**encodingFilter**</filter-name>**
16. **<filter-class>**org.springframework.web.filter.CharacterEncodingFilter**</filter-class>**
17. **<init-param>**
18. **<param-name>**encoding**</param-name>**
19. **<param-value>**UTF-8**</param-value>**
20. **</init-param>**
21. **<init-param>**
22. **<param-name>**forceEncoding**</param-name>**
23. **<param-value>**true**</param-value>**
24. **</init-param>**
25. **</filter>**
26. **<filter-mapping>**
27. **<filter-name>**encodingFilter**</filter-name>**
28. **<url-pattern>**/\***</url-pattern>**
29. **</filter-mapping>**
30. <!-- Creates the Spring Container shared by all Servlets and Filters -->
31. **<listener>**
32. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
33. **</listener>**



38. <!-- Processes application requests -->
39. **<servlet>**
40. **<servlet-name>**appServlet**</servlet-name>**
41. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>**
42. **<init-param>**
43. **<param-name>**contextConfigLocation**</param-name>**
44. **<param-value>**/WEB-INF/spring/appServlet/servlet-context.xml**</param-value>**
45. **</init-param>**
46. **<load-on-startup>**1**</load-on-startup>**
47. **</servlet>**
49. **<servlet-mapping>**
50. **<servlet-name>**appServlet**</servlet-name>**
51. **<url-pattern>**/**</url-pattern>**
52. **</servlet-mapping>**
54. **</web-app>**

root-context.xml spring的主配置文件

**[html]** [view plaincopy](http://blog.csdn.net/qyl445/article/details/9837579)[在CODE上查看代码片](https://code.csdn.net/snippets/536921)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
6. http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd
7. "**>**
8. **<context:annotation-config** **/>**
9. **<context:component-scan** base-package="com.dahafo.demo.um" **/>**
10. **<import** resource="spring-mybatis.xml"**/>**
11. **</beans>**

servlet-context.xml springmvc的主配置文件

**[html]** [view plaincopy](http://blog.csdn.net/qyl445/article/details/9837579)[在CODE上查看代码片](https://code.csdn.net/snippets/536921)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans:beans** xmlns="http://www.springframework.org/schema/mvc"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:beans="http://www.springframework.org/schema/beans"
5. xmlns:context="http://www.springframework.org/schema/context"
6. xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd
7. http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"**>**
10. <!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
12. <!-- Enables the Spring MVC @Controller programming model -->
13. **<annotation-driven** **/>**
14. **<context:component-scan** base-package="com.dahafo.demo.um.controller" **/>**
16. <!-- Handles HTTP GET requests for /resources/\*\* by efficiently serving up static resources in the ${webappRoot}/resources directory -->
17. **<resources** mapping="/resources/\*\*" location="/resources/" **/>**
18. **<resources** mapping="/css/\*\*" location="/css/" **/>**
19. **<resources** mapping="/images/\*\*" location="/images/" **/>**
20. **<resources** mapping="/js/\*\*" location="/js/" **/>**
22. <!-- Jackson转换器 -->
23. **<beans:bean** id="mappingJacksonHttpMessageConverter" class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter" **/>**
24. <!-- fastjson转换器 -->
25. **<beans:bean** id="fastJsonHttpMessageConverter"  class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter"**/>**
26. **<beans:bean** class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"  **>**
27. **<beans:property** name="messageConverters"**>**
28. **<beans:list>**
29. **<beans:ref** bean="fastJsonHttpMessageConverter" **/>**<!-- json转换器 -->
30. **</beans:list>**
31. **</beans:property>**
32. **</beans:bean>**
33. <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
34. **<beans:bean** class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>**
35. **<beans:property** name="prefix" value="/WEB-INF/views/" **/>**
36. **<beans:property** name="suffix" value=".jsp" **/>**
37. **</beans:bean>**
39. **<beans:bean** id="maxUploadSize" class="org.springframework.web.multipart.commons.CommonsMultipartResolver"**>**
40. **<beans:property** name="maxUploadSize" value="32505856" **/>**<!-- 上传文件大小限制为31M，31\*1024\*1024 -->
41. **<beans:property** name="maxInMemorySize" value="4096" **/>**
42. **</beans:bean>**


46. **</beans:beans>**

spring-mybatis.xml  mybatis的配置文件，数据源用 的是阿里巴巴的 druid 

**[html]** [view plaincopy](http://blog.csdn.net/qyl445/article/details/9837579)[在CODE上查看代码片](https://code.csdn.net/snippets/536921)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:tx="http://www.springframework.org/schema/tx"
6. xmlns:aop="http://www.springframework.org/schema/aop"
7. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd
9. http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
10. http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
11. "**>**
12. <!-- 引入属性文件 -->
13. **<context:property-placeholder** location="classpath:jdbc.properties" **/>**
15. <!-- 配置数据源 -->
16. **<bean** name="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close"**>**
17. **<property** name="url" value="${url}" **/>**
18. **<property** name="username" value="${username}" **/>**
19. **<property** name="password" value="${password}" **/>**
20. **<property** name="initialSize" value="1" **/>**
21. **<property** name="maxActive" value="20" **/>**
22. **<property** name="minIdle" value="1" **/>**
23. **<property** name="maxWait" value="60000" **/>**
24. **<property** name="validationQuery" value="${validationQuery}" **/>**
25. **<property** name="testOnBorrow" value="false" **/>**
26. **<property** name="testOnReturn" value="false" **/>**
27. **<property** name="testWhileIdle" value="true" **/>**
28. **<property** name="timeBetweenEvictionRunsMillis" value="60000" **/>**
29. **<property** name="minEvictableIdleTimeMillis" value="25200000" **/>**
30. **<property** name="removeAbandoned" value="true" **/>**
31. **<property** name="removeAbandonedTimeout" value="1800" **/>**
32. **<property** name="logAbandoned" value="true" **/>**
33. **<property** name="filters" value="mergeStat" **/>**
34. **</bean>**
36. <!-- myBatis文件 -->
37. **<bean** id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"**>**
38. **<property** name="dataSource" ref="dataSource" **/>**
39. **<property** name="mapperLocations" value="classpath:com/dahafo/demo/um/mapping/\*Mapper.xml" **/>**
40. **</bean>**
42. **<bean** class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>**
43. **<property** name="basePackage" value="com.dahafo.demo.um.dao" **/>**
44. **<property** name="sqlSessionFactoryBeanName" value="sqlSessionFactory" **/>**
45. **</bean>**
47. <!-- 配置事务管理器 -->
48. **<bean** id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager"**>**
49. **<property** name="dataSource" ref="dataSource" **/>**
50. **</bean>**
52. <!-- 注解方式配置事物 -->
53. <!-- <tx:annotation-driven transaction-manager="transactionManager" /> -->
55. <!-- 拦截器方式配置事物 -->
56. **<tx:advice** id="transactionAdvice" transaction-manager="transactionManager"**>**
57. **<tx:attributes>**
58. **<tx:method** name="add\*" propagation="REQUIRED" **/>**
59. **<tx:method** name="modify\*" propagation="REQUIRED" **/>**
60. **<tx:method** name="delete\*" propagation="REQUIRED" **/>**
61. **<tx:method** name="find\*" propagation="SUPPORTS" **/>**
62. **<tx:method** name="query" propagation="SUPPORTS" **/>**
63. **<tx:method** name="search\*" propagation="SUPPORTS" **/>**
64. **<tx:method** name="\*" propagation="SUPPORTS" **/>**
65. **</tx:attributes>**
66. **</tx:advice>**
67. **<aop:config>**
68. **<aop:pointcut** id="transactionPointcut" expression="execution(\* com.dahafo.demo.um.service.\*.\*(..))" **/>**
69. **<aop:advisor** pointcut-ref="transactionPointcut" advice-ref="transactionAdvice" **/>**
70. **</aop:config>**
71. <!-- 配置druid监控spring jdbc -->
72. **<bean** id="druid-stat-interceptor" class="com.alibaba.druid.support.spring.stat.DruidStatInterceptor"**>**
73. **</bean>**
74. **<bean** id="druid-stat-pointcut" class="org.springframework.aop.support.JdkRegexpMethodPointcut" scope="prototype"**>**
75. **<property** name="patterns"**>**
76. **<list>**
77. **<value>**com.dahafo.demo.um.service.impl**</value>**
78. **</list>**
80. **</property>**
81. **</bean>**
82. **<aop:config>**
83. **<aop:advisor** advice-ref="druid-stat-interceptor" pointcut-ref="druid-stat-pointcut" **/>**
84. **</aop:config>**
85. **</beans>**

转载注明：

<http://blog.csdn.net/zhshulin>

<http://www.cnblogs.com/chenying99/archive/2012/03/04/2379351.html>

<http://blog.csdn.net/qyl445/article/details/9837579>