目录

[项目目录结构图： 2](#_Toc342749405)

[第一步：准备 2](#_Toc342749406)

[Sprng中的jar包 2](#_Toc342749407)

[Hibernate中的jar包 3](#_Toc342749408)

[其它jar包 4](#_Toc342749409)

[第二步：配置文件 5](#_Toc342749410)

[从hibernate中复制并修改文件 5](#_Toc342749411)

[修改hibernate.properties文件 6](#_Toc342749412)

[修改log4j.properties文件 6](#_Toc342749413)

[编写spring配置文件 7](#_Toc342749414)

[新建applicationContext.xml文件 7](#_Toc342749415)

[新建spring-servlet.xml文件 9](#_Toc342749416)

[修改web.xml文件 9](#_Toc342749417)

[第三步：业务逻辑 10](#_Toc342749418)

[Model层 10](#_Toc342749419)

[Dao层 11](#_Toc342749420)

[Service层 12](#_Toc342749421)

[Controller层（Action层） 13](#_Toc342749422)

[View层 14](#_Toc342749423)

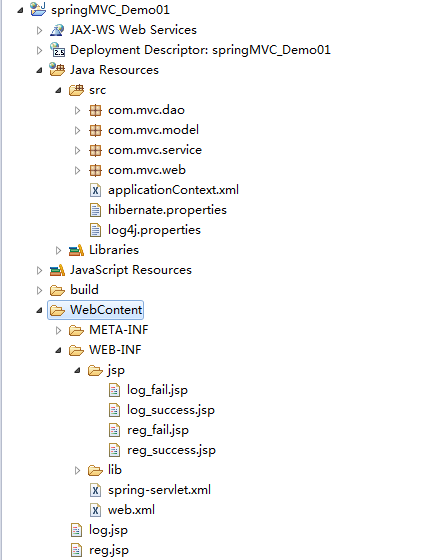
[第四步：测试 15](#_Toc342749424)

[开启服务器 15](#_Toc342749425)

[在浏览器中浏览 16](#_Toc342749426)

[运行结果： 16](#_Toc342749427)

# 项目目录结构图：



项目结构图

# 第一步：准备

## Sprng中的jar包

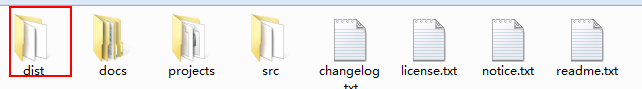
首先从spring官网<http://www.springsource.org/download/community>中下载最新版本的spring版本，本例采用的是spring-framework-3.1.1.RELEASE。初学者最好采用和此版本一样的版本。下载后如图：



解压得到文件夹：



打开文件夹：



选择如下jar包：



将以上jar包复制到你的/web-INF下的/lib中（以下简称lib）。

## Hibernate中的jar包

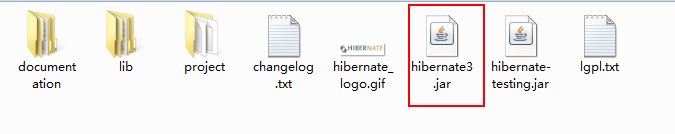
准备hibernate的jar包，从hibernate官网<http://www.hibernate.org/downloads>下载hibernate版本，本例采用的是hibernate-distribution-3.6.10.Final，下载后如下：

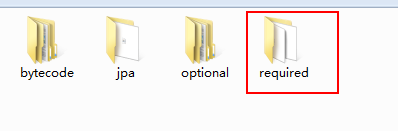


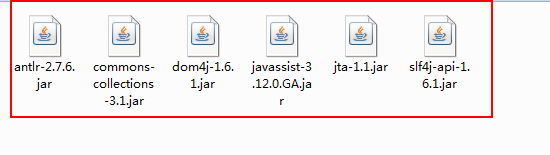
解压后：



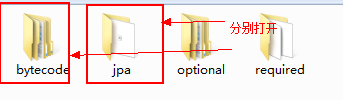
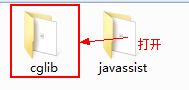
打开hibernate文件夹，先选择如图中的jar包：

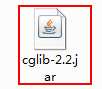
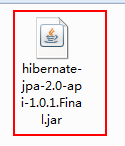
再打开hibernate文件夹中的lib文件夹：





将此文件夹的jar包全部复制到web项目的lib目录中并将jpa中的jar也复制到lib中。将bytecode中的cglib复制到lib中。分别如图：

## 其它jar包

以上是hibernate和spring中所要加入的包，除些之外还要加入如下包：



以上是所有的springMVC+hibernate的所需jar包。

将所有jar包复制到web项目的lib目录中。

# 第二步：配置文件

## 从hibernate中复制并修改文件

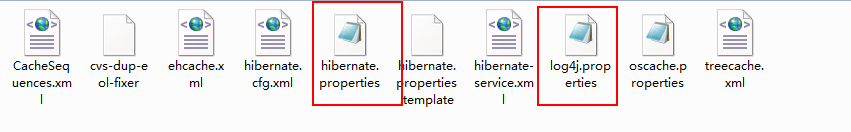
首先从hibernate中的project文件夹中打到etc文件夹，复制其中如图所示的两个文件到classpath下，如图：  


图2-1

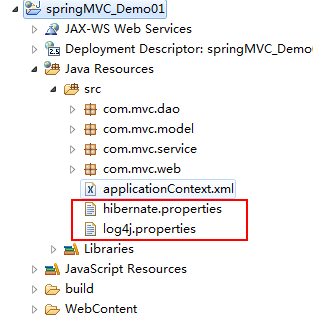


图2-2

### 修改hibernate.properties文件

如下：

## HypersonicSQL

dataSource.password=

dataSource.username=root

dataSource.databaseName=test

dataSource.driverClassName=com.mysql.jdbc.Driver

dataSource.dialect=org.hibernate.dialect.MySQL5Dialect

dataSource.serverName=localhost:3306

dataSource.url=jdbc:mysql://localhost:3306/test

dataSource.properties=user=${dataSource.username};databaseName=${dataSource.databaseName};serverName=${dataSource.serverName};password=${dataSource.password}

dataSource.hbm2ddl.auto=update

#hibernate.connection.url jdbc:hsqldb:hsql://localhost

#hibernate.connection.url jdbc:hsqldb:test

### 修改log4j.properties文件

如下：

log4j.rootLogger=warn, stdout

#log4j.logger.org.hibernate=info

#log4j.logger.org.hibernate=debug

以上两个文件未帖出来的表示不改。

## 编写spring配置文件

### 新建applicationContext.xml文件

在classpath下新建applicationContext.xml文件内容如下：

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:aop="http://www.springframework.org/schema/aop" xmlns:context="http://www.springframework.org/schema/context"

xmlns:p="http://www.springframework.org/schema/p" xmlns:tx="http://www.springframework.org/schema/tx"

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd">

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

<!-- 支持aop注解 -->

<aop:aspectj-autoproxy />

<context:property-placeholder location="classpath:/hibernate.properties" />

<bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDataSource">

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

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

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

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

</bean>

<bean id="sessionFactory"

class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">

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

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect">${dataSource.dialect}</prop>

<prop key="hibernate.hbm2ddl.auto">${dataSource.hbm2ddl.auto}</prop>

<prop key="hibernate.hbm2ddl.auto">update</prop>

</props>

</property>

<property name="packagesToScan">

<list>

<value>com.mvc.model</value><!-- 扫描实体类，也就是平时所说的model -->

</list>

</property>

</bean>

<bean id="hibernateTemplate" class="org.springframework.orm.hibernate3.HibernateTemplate">

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

</bean>

<!-- 配置事务管理 -->

<bean id="txManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">

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

</bean>

<tx:annotation-driven transaction-manager="txManager" />

<aop:config>

<aop:pointcut expression="execution(public \* com.mvc.service.impl.\*.\*(..))"

id="businessService" />

<aop:advisor advice-ref="txAdvice" pointcut-ref="businessService" />

</aop:config>

<tx:advice id="txAdvice" transaction-manager="txManager">

<tx:attributes>

<tx:method name="find\*" read-only="true" propagation="NOT\_SUPPORTED" />

<!-- get开头的方法不需要在事务中运行 。 有些情况是没有必要使用事务的，比如获取数据。开启事务本身对性能是有一定的影响的 -->

<tx:method name="\*" /> <!-- 其他方法在实务中运行 -->

</tx:attributes>

</tx:advice>

</beans>

### 新建spring-servlet.xml文件

在web-INF下新建spring-servlet.xml文件，内容如下：

<?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:mvc="http://www.springframework.org/schema/mvc"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:util="http://www.springframework.org/schema/util"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd

http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd

http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-3.0.xsd">

<!-- 对web包中的所有类进行扫描，以完成Bean创建和自动依赖注入的功能 -->

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

<mvc:annotation-driven /> <!-- 支持spring3.0新的mvc注解 -->

<!-- 启动Spring MVC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"/>

<!--对模型视图名称的解析，即在模型视图名称添加前后缀 -->

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

p:prefix="/WEB-INF/jsp/" p:suffix=".jsp">

<!-- 如果使用jstl的话，配置下面的属性 -->

<property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />

</bean>

</beans>

## 修改web.xml文件

文件内容如下：

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

<web-app version="2.5"

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instanc*e"*

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

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<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:applicationContext\*.xml,/WEB-INF/spring-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.\*</welcome-file>

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

</welcome-file-list>

</web-app>

# 第三步：业务逻辑

## Model层

新建实体类User.java，内容如下：

package com.mvc.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

@Entity

public class User {

private int id;

private String name;

private String password;

@Id

@GeneratedValue

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

## Dao层

新建UserDao.java，内容如下：

package com.mvc.dao;

import java.util.List;

import javax.annotation.Resource;

import org.springframework.orm.hibernate3.HibernateTemplate;

import org.springframework.stereotype.Repository;

import com.mvc.model.User;

@Repository("userDao")

public class UserDao {

@Resource

private HibernateTemplate hibernateTemplate;

public void add(User u) {

hibernateTemplate.save(u);

}

public boolean findUser(User u) {

List<?> list = hibernateTemplate.find("from User u where u.name=?",

u.getName());

if (list.size() > 0) {

return true;

}

return false;

}

public boolean userLogin(User u) {

List<?> list = hibernateTemplate.find(

"from User u where u.name=? and u.password=?", u.getName(),

u.getPassword());

if (list.size() > 0) {

return true;

}

return false;

}

}

## Service层

新建UserService.java，内容如下：

package com.mvc.service;

import javax.annotation.Resource;

import org.springframework.stereotype.Service;

import com.mvc.dao.UserDao;

import com.mvc.model.User;

@Service("userService")

public class UserService {

@Resource

private UserDao userDao;

public void add(User user) {

userDao.add(user);

}

public boolean exist(User user) {

return userDao.findUser(user);

}

public boolean login(User user) {

return userDao.userLogin(user);

}

}

## Controller层（Action层）

新建UserControler.java，内容如下：

package com.mvc.web;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;

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

import com.mvc.model.User;

import com.mvc.service.UserService;

@Controller("userController")

@RequestMapping("/user.do")

public class UserController {

@Resource

private UserService userService;

@RequestMapping(params = "method=reg")

public String reg(User user) {

System.out.println("用户注册");

if (userService.exist(user)) {

return "reg\_fail";

}

userService.add(user);

return "reg\_success";

}

@RequestMapping(params = "method=log")

public String log(User user) {

System.out.println("用户登录");

if (userService.login(user)) {

return "log\_success";

}

return "log\_fail";

}

}

## View层

在WebContent中新建两个jsp文件，分别是reg.jsp和log.jsp。

Reg.jsp的内容如下：

<%@ 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="user.do">

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

<br/>

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

<input type="hidden" name="method" value="reg" />

<br/>

<input type="submit" value="注册" />

</form>

<a href="/springMVC\_Demo01/log.jsp">登录</a>

</body>

</html>

Log.jsp的内容如下：

<%@ 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="user.do">

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

<br/>

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

<input type="hidden" name="method" value="log" />

<br/>

<input type="submit" value="登录" />

</form>

<a href="/springMVC\_Demo01/reg.jsp">注册</a>

</body>

</html>

在/web-inf文件夹中新建jsp文件夹创建四个jsp文件，分别为：reg\_fail、reg\_success、log\_fail、log\_success。四个文件中只做一些简单的提示。

Log\_fail.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>登录失败！用户名或密码错误</h1>

<a href="/springMVC\_Demo01/log.jsp">跳到登录页</a>

</body>

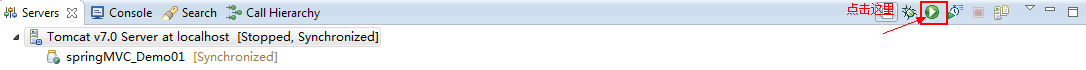
</html>

其它三个只改body中的提示内容,在此不再缀述。

# 第四步：测试

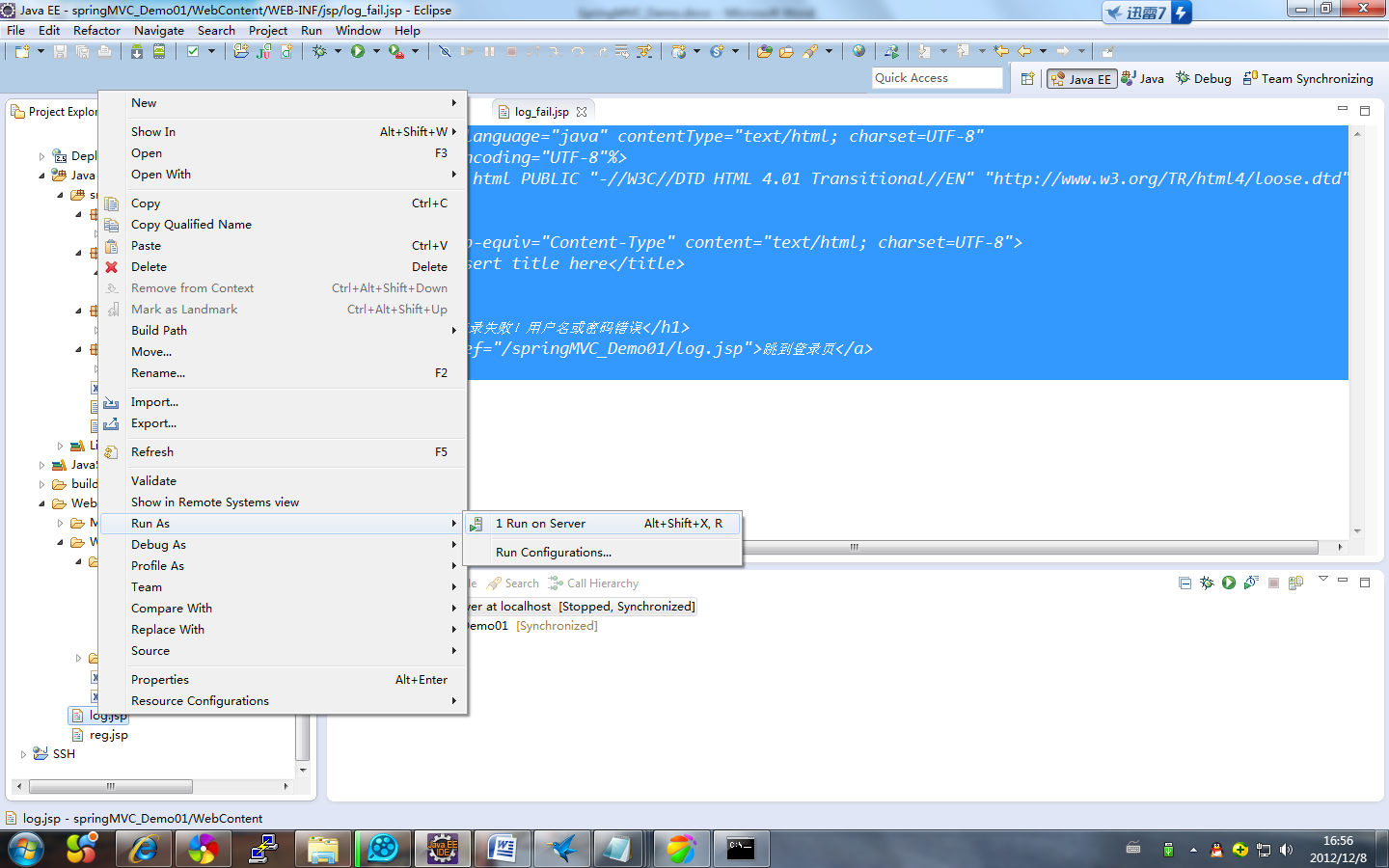
## 开启服务器

打开tomcat服务器如图：



## 在浏览器中浏览

右击reg.jsp文件选择Run As On Server如图：



## 运行结果：

如图：

