**Spring4 MVC Hibernate4集成**

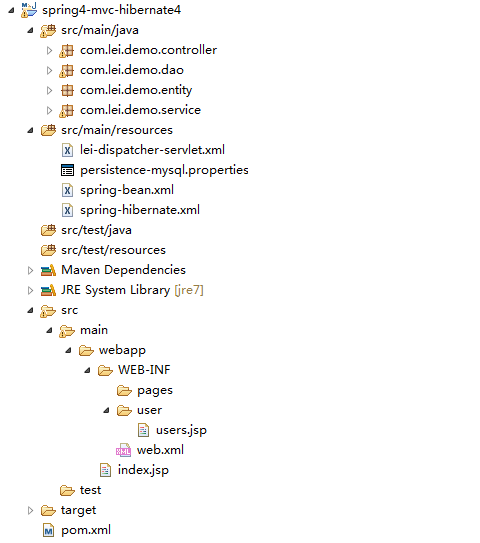
**一、    本文所用环境**

Spring4.0.3.RELEASE

Hibernate4.3.5.Final

Mysql

**二、    工程目录**



**三、    Maven添加依赖**

用Maven创建项目，pom.xml如下：

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

**四、    新建数据库表**

数据库采用Mysql，新建users表，我们演示操作此表

[复制代码](javascript:void(0);)

CREATE TABLE `users` (

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

`age` int(**11**) DEFAULT NULL,

`nice\_name` varchar(**32**) DEFAULT NULL,

`user\_name` varchar(**32**) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

[复制代码](javascript:void(0);)

创建Entity类Users.java见下

[复制代码](javascript:void(0);)

package com.lei.demo.entity;

import javax.persistence.\*;

@Entity(name="users")

public class Users {

public Users(){

super();

}

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

@Column(name="id")

private Integer id;

@Column(name="user\_name",length=32)

private String user\_name;

@Column(name="age")

private Integer age;

@Column(name="nice\_name",length=32)

private String nice\_name;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getUser\_name() {

return user\_name;

}

public void setUser\_name(String user\_name) {

this.user\_name = user\_name;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

public String getNice\_name() {

return nice\_name;

}

public void setNice\_name(String nice\_name) {

this.nice\_name = nice\_name;

}

}

[复制代码](javascript:void(0);)

**五、    配置文件**

**1.        首先看一下Web.xml**

[复制代码](javascript:void(0);)

<?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"

xmlns:web="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"

id="WebApp\_ID" version="3.0">

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

<context-param>

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

<param-value>classpath:/spring-\*.xml</param-value>

</context-param>

<listener>

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

</listener>

<!-- 定义DispatcherServlet -->

<servlet>

<servlet-name>lei-dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<!-- 默认/WEB-INF/[servlet名字]-servlet.xml加载上下文，

如果配置了contextConfigLocation参数，

将使用classpath:/lei-dispatcher-servlet.xml加载上下文

-->

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

<param-value>classpath:/lei-dispatcher-servlet.xml</param-value>

</init-param>

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

</servlet>

<!-- 拦截匹配的请求，这里所有请求采用名字为lei-dispatcher的DispatcherServlet处理 -->

<servlet-mapping>

<servlet-name>lei-dispatcher</servlet-name>

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

</servlet-mapping>

</web-app>

[复制代码](javascript:void(0);)

**2.        lei-dispatcher-servlet.xml**

[复制代码](javascript:void(0);)

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

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

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

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

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

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

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

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/tx

http://www.springframework.org/schema/tx/spring-tx-4.0.xsd

">

<!-- 启动自动扫描该包下所有的Bean(例如@Controller) -->

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

<!-- 定义视图解析器 -->

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

<property name="prefix">

<value>/WEB-INF/user/</value>

</property>

<property name="suffix">

<value>.jsp</value>

</property>

</bean>

</beans>

[复制代码](javascript:void(0);)

**3.        Spring-hibernate的整合**

spring-hibernate.xml文件

[复制代码](javascript:void(0);)

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

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

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

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

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

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

xsi:schemaLocation="

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

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

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

http://www.springframework.org/schema/context/spring-context-4.0.xsd

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

http://www.springframework.org/schema/aop/spring-aop-4.0.xsd

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

http://www.springframework.org/schema/tx/spring-tx-4.0.xsd

">

<!-- Hibernate4 -->

<!-- 加载资源文件 其中包含变量信息，必须在Spring配置文件的最前面加载，即第一个加载-->

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

<bean id="sessionFactory"

class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">

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

<property name="packagesToScan">

<list>

<!-- 可以加多个包 -->

<value>com.lei.demo.entity</value>

</list>

</property>

<property name="hibernateProperties">

<props>

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

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

<prop key="hibernate.show\_sql">${hibernate.show\_sql}</prop>

<!-- <prop key="hibernate.current\_session\_context\_class">thread</prop> -->

</props>

</property>

</bean>

<!-- 数据库映射 -->

<!-- class="org.apache.tomcat.dbcp.dbcp.BasicDataSource" -->

<!-- class="org.springframework.jdbc.datasource.DriverManagerDataSource" -->

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

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

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

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

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

</bean>

<!-- 配置Hibernate事务管理器 -->

<bean id="transactionManager"

class="org.springframework.orm.hibernate4.HibernateTransactionManager">

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

</bean>

<!-- 配置事务异常封装 -->

<bean id="persistenceExceptionTranslationPostProcessor"

class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor" />

<!-- 声明式容器事务管理 ,transaction-manager指定事务管理器为transactionManager -->

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

<tx:attributes>

<tx:method name="add\*" propagation="REQUIRED" />

<tx:method name="get\*" propagation="REQUIRED" />

<tx:method name="\*" read-only="true" />

</tx:attributes>

</tx:advice>

<aop:config expose-proxy="true">

<!-- 只对业务逻辑层实施事务 -->

<aop:pointcut id="txPointcut" expression="execution(\* com.lei.demo.service..\*.\*(..))" />

<!-- Advisor定义，切入点和通知分别为txPointcut、txAdvice -->

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

</aop:config>

</beans>

[复制代码](javascript:void(0);)

事务采用tx:advice和aop:config的xml配置方式实现。事务的其他实现方式见[***《Spring MVC事务配置》***](http://www.cnblogs.com/leiOOlei/p/3725911.html)

资源文件*persistence-mysql.properties*如下：

[复制代码](javascript:void(0);)

# jdbc.X

jdbc.driverClassName=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://yourServerIP:3306/yourDatabase?createDatabaseIfNotExist=true

jdbc.user=user

jdbc.pass=password

# hibernate.X

hibernate.connection.driverClass=org.gjt.mm.mysql.Driver

hibernate.connection.url=jdbc:mysql:// yourServerIP:3306/yourDatabase

hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

hibernate.connection.username=user

hibernate.connection.password=password

hibernate.show\_sql=true

hibernate.hbm2ddl.auto=create-drop

[复制代码](javascript:void(0);)

**六、    DAO、Service**

UsersDAO.java如下：

[复制代码](javascript:void(0);)

package com.lei.demo.dao;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import com.lei.demo.entity.Users;

public class UsersDAO {

private SessionFactory sessionFactory;

public void setSessionFactory(SessionFactory sessionFactory) {

this.sessionFactory = sessionFactory;

}

public SessionFactory getSessionFactory() {

return sessionFactory;

}

public List<Users> getAllUser(){

String hsql="from users";

Session session = sessionFactory.getCurrentSession();

Query query = session.createQuery(hsql);

return query.list();

}

}

[复制代码](javascript:void(0);)

UserService.java如下：

[复制代码](javascript:void(0);)

package com.lei.demo.service;

import com.lei.demo.dao.\*;

public class UserService {

private UsersDAO userDao;

public int userCount(){

return userDao.getAllUser().size();

}

public UsersDAO getUserDao() {

return userDao;

}

public void setUserDao(UsersDAO userDao) {

this.userDao = userDao;

}

}

[复制代码](javascript:void(0);)

**七、    加载Bean配置文件**

这里主要是加载上边的dao、service

[复制代码](javascript:void(0);)

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

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

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

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-3.2.xsd

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

http://www.springframework.org/schema/context/spring-context-3.2.xsd">

<bean id="usersDAO" class="com.lei.demo.dao.UsersDAO">

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

</bean>

<bean id="userService" class="com.lei.demo.service.UserService">

<property name="userDao" ref="usersDAO"></property>

</bean>

</beans>

[复制代码](javascript:void(0);)

当然你也可以不采用xml配置bean，采用注释加载bean，见[***《Spring3系列7- 自动扫描组件或Bean》***](http://www.cnblogs.com/leiOOlei/p/3547589.html)、[***《Spring 开启Annotation <context:annotation-config> 和 <context:component-scan>诠释及区别》***](http://www.cnblogs.com/leiOOlei/p/3713989.html)

**八、    JSP页面和Controller**

WEB-INF/user/users.jsp如下：

[复制代码](javascript:void(0);)

<%@ 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>Message : ${message}</h1>

</body>

</html>

[复制代码](javascript:void(0);)

写Controller访问上边的jsp

[复制代码](javascript:void(0);)

package com.lei.demo.controller;

import javax.annotation.Resource;

import com.lei.demo.service.\*;

import org.springframework.stereotype.Controller;

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

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

import org.springframework.web.servlet.ModelAndView;

@Controller

@RequestMapping("/user")

public class UserController {

@Resource(name="userService")

private UserService service;

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

public ModelAndView hello2(){

ModelAndView mv = new ModelAndView();

mv.addObject("message", "HelloMVC");

mv.setViewName("users");

return mv;

}

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

public ModelAndView count(){

int c = service.userCount();

ModelAndView mv = new ModelAndView();

mv.addObject("message", c);

mv.setViewName("users");

return mv;

}

}

[复制代码](javascript:void(0);)

上边的controller定义了两个路径

http://localhost:8080/spring4-mvc-hibernate4/user/manager

http://localhost:8080/spring4-mvc-hibernate4/user/count

以上两个路径都调用WEB-INF/user/users.jsp

参考资料《Spring3系列13-Controller和@RequestMapping》

**九、    运行结果**

