**springMVc+Ibatis+Mecached简单配置文档**

文档信息：本文档基于spring4.0.0M2，mybatis3.1.1编写的简单的框架整合流程，针对于需要整个两个框架的朋友而编写，并且内容都由demo中拷贝而出，能够完整正确的运行。

框架信息：

框架名称：spring-framework

版本：spring4.0.0M2

框架名称：MyBatis

版本：3.1.1

memcached-win32-1.4.4-14

所需jar包：

//请保证jar包版本的对应性，否则可能出现不可预知错误

aspectjrt.jar commons-logging.jar java\_memcached-release\_2.6.6.jar javax-inject.jar jcl-over-slf4j-1.6.4.jar jsp-api.jar jstl.jar junit4.4.jar log4j-1.2.14.jar logback-access-1.0.13.jar logback-classic-1.0.13.jar logback-core-1.0.13.jar mybatis-3.1.1.jar mybatis-spring-1.1.0.jar mysql-connector-java-5.1.23-bin.jar org.springframework.transaction-3.0.0.RELEASE.jar servlet-api.jar slf4j-api-1.6.4.jar slf4j-log4j12-1.6.4.jar spring-aop-4.0.0.M2.jar spring-beans-4.0.0.M2.jar spring-context-4.0.0.M2.jar spring-context-support-4.0.0.M2.jar spring-core-4.0.0.M2.jar spring-expression-4.0.0.M2.jar spring-jdbc-4.0.0.M2.jar spring-web-4.0.0.M2.jar spring-webmvc-4.0.0.M2.jar spy-2.4.jar spymemcached-2.8.1.jar

# MyBatis配置

## 创建mybatis配置文件（mybatis-config.xml）

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEconfiguration

PUBLIC"-//ibatis.apache.org//DTD Config 3.0//EN"

"http://ibatis.apache.org/dtd/ibatis-3-config.dtd">

<configuration>

<!--给持久层对象配置个别名，能方便调用 -->

<typeAliases>

<typeAliastype=*"test.bean.User"*alias=*"User"*/>

</typeAliases>

<!--接口映射文件的位置 -->

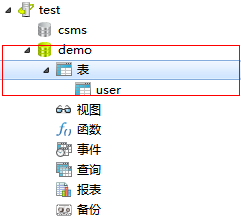
<mappers>

<mapperresource=*"test/dao/IUserDao.xml"*/>

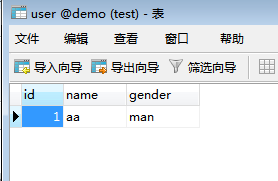
</mappers>

</configuration>

## 创建数据库(demo)



## 创建数据表（user）



创建完毕后请填充测试数据

## 创建实体类（User）

**package**test.bean;

**import**java.io.Serializable;

//继承Serializable表示此类可序列化

**publicclass** User **implements**Serializable{

**privateint**id;

**private** String name;

**private** String gender;

//缓存序列化时需要用到的属性

**privatestaticfinallong***serialVersionUID* = 1L;

**publicint**getId() {

**return**id;

}

**publicvoid**setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return**name;

}

**publicvoid**setName(String name) {

**this**.name = name;

}

**public**StringgetGender() {

**return**gender;

}

**publicvoid**setGender(String gender) {

**this**.gender = gender;

}

}

## 创建dao层接口（IUserDao.java）

**package**test.dao;

**import**test.bean.User;

**publicinterface**IUserDao {

User getUser(String name);

}

## 创建dao层接口所对应的配置文件（IUserDao.xml）

请与接口类使用相同的名字

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEmapper

PUBLIC"-//ibatis.apache.org//DTD Mapper 3.0//EN"

"http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">

<!-- 务必配置正确namespace就是所映射的接口类-->

<mappernamespace=*"test.dao.IUserDao"*>

<!-- resultType="User"这个使用的是配置文件里面的别名（配置文件为mybatis-config.xml） -->

<selectid=*"getUser"*parameterType=*"string"*resultType=*"User"*>

select \* from user where name=#{name}

</select>

</mapper>

## 在spring配置文件中配置数据源（完整配置见spring部分）

<!-- 获取JDBC连接属性 -->

<context:property-placeholderlocation=*"/WEB-INF/config/jdbc.properties"*/>

<!-- 配置数据源 -->

<beanid=*"dataSource"*

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

<propertyname=*"driverClassName"*>

<value>${jdbc.driverClassName}</value>

</property>

<propertyname=*"url"*>

<value>${jdbc.url}</value>

</property>

</bean>

<!-- sqlSessionFactory -->

<!-- MyBatis在spring中Bean的配置，都是固定的 -->

<beanid=*"sqlSessionFactory"*class=*"org.mybatis.spring.SqlSessionFactoryBean"*>

<propertyname=*"configLocation"*value=*"/WEB-INF/config/mybatis-config.xml"*/>

<propertyname=*"dataSource"*ref=*"dataSource"*/>

</bean>

<beanid=*"sqlSession"*class=*"org.mybatis.spring.SqlSessionTemplate"*>

<constructor-argindex=*"0"*ref=*"sqlSessionFactory"*/>

</bean>

<!-- 配置映射器 -->

<beanid=*"userMapper"*class=*"org.mybatis.spring.mapper.MapperFactoryBean"*>

<propertyname=*"mapperInterface"*value=*"test.dao.IUserDao"*/>

<propertyname=*"sqlSessionFactory"*ref=*"sqlSessionFactory"*/>

</bean>

<!-- 为业务逻辑层注入数据的对象 -->

<beanid=*"userServer"*class=*"test.server.UserServerImpl"*>

<propertyname=*"userDao"*ref=*"userMapper"*></property>

<propertyname=*"memcachedClient"*ref=*"memcachedClient"*></property>

</bean>

## 配置数据库连接参数文件（jdbc.properties）

jdbc.driverClassName=com.mysql.jdbc.Driver

jdbc.url=jdbc\:mysql\://localhost\:3306/demo?user\=test&password\=test

# Memcached配置

## Memcached安装

一、下载memcached的稳定版本,然后解压到某个目录下面，我放到了d:\app\memcached

[下载链接](http://www.splinedancer.com/memcached-win32/)

找到c:\WINDOWS\system32\cmd.exe，右键以管理员身份运行，否则会报错,切换到memcached目录下面,

 安装

Cmd切换目录的代码为cd –d d:/memcached

Cmd代码  [收藏代码](javascript:void())

1. memcached.exe –d install

 启动

Cmd代码  [收藏代码](javascript:void())

1. memcached.exe -d start

## spring自动生成memcached对象

在spring配置文件中配置（完成内容见spring部分）

<!--memcached注入-->

<beanid=*"memcachedClient"*class=*"net.spy.memcached.spring.MemcachedClientFactoryBean"*>

<propertyname=*"servers"*value=*"localhost:11211"*/>

<propertyname=*"protocol"*value=*"BINARY"*/>

<propertyname=*"transcoder"*>

<beanclass=*"net.spy.memcached.transcoders.SerializingTranscoder"*>

<propertyname=*"compressionThreshold"*value=*"1024"*/>

</bean>

</property>

<propertyname=*"opTimeout"*value=*"50"*/>

<propertyname=*"timeoutExceptionThreshold"*value=*"1998"*/>

<propertyname=*"hashAlg"*>

<valuetype=*"net.spy.memcached.DefaultHashAlgorithm"*>KETAMA\_HASH</value>

</property>

<propertyname=*"locatorType"*value=*"CONSISTENT"*/>

<propertyname=*"failureMode"*value=*"Redistribute"*/>

<propertyname=*"useNagleAlgorithm"*value=*"false"*/>

</bean>

## 注入业务类中

完整内容见spring配置部分

<!--为业务逻辑层注入数据的对象 -->

<beanid=*"userServer"*class=*"test.server.UserServerImpl"*>

<propertyname=*"userDao"*ref=*"userMapper"*></property>

<propertyname=*"memcachedClient"*ref=*"memcachedClient"*></property>

</bean>

注入后在业务类中直接调用即可（具体调用见spring部分）

# Spring配置

## web.xml中添加spring配置

<!—spring监听配置-->

<context-param>

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

<param-value>/WEB-INF/config/applicationContext.xml</param-value>

</context-param>

<listener>

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

</listener>

## web.xml中添加springMVC配置

<!--springmvc入口配置 start -->

<servlet>

<servlet-name>springMVC</servlet-name>

<!--springmvc总控制器 --><servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

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

<param-value>/WEB-INF/config/mvc-dispatcher-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>

<!--springmvc入口配置 end -->

## 编写springMVC配置文档（mvc-dispatcher-servlet.xml）

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<beansxmlns=*"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"*

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"*>

<!--spring 组件扫描配置 -->

<context:component-scanbase-package=*"test.controller"*/>

<!--配置视图解析器 (该配置的是jstl解析器的一个扩展解析类) -->

<beanclass=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<!—请求转发的目录-->

<propertyname=*"prefix"*value=*"/WEB-INF/pages/"*/>

<!—请求转发的文件类型-->

<propertyname=*"suffix"*value=*".jsp"*/>

</bean>

</beans>

## 编写spring配置文档（applicationContext.xml）

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<beansxmlns=*"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"*

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"*>

<!--获取JDBC连接属性 -->

<context:property-placeholderlocation=*"/WEB-INF/config/jdbc.properties"*/>

<!--配置数据源 -->

<beanid=*"dataSource"*

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

<propertyname=*"driverClassName"*>

<value>${jdbc.driverClassName}</value>

</property>

<propertyname=*"url"*>

<value>${jdbc.url}</value>

</property>

</bean>

<!--sqlSessionFactory -->

<!--MyBatis在spring中Bean的配置，都是固定的 -->

<beanid=*"sqlSessionFactory"*class=*"org.mybatis.spring.SqlSessionFactoryBean"*>

<propertyname=*"configLocation"*value=*"/WEB-INF/config/mybatis-config.xml"*/>

<propertyname=*"dataSource"*ref=*"dataSource"*/>

</bean>

<beanid=*"sqlSession"*class=*"org.mybatis.spring.SqlSessionTemplate"*>

<constructor-argindex=*"0"*ref=*"sqlSessionFactory"*/>

</bean>

<!--配置映射器 -->

<beanid=*"userMapper"*class=*"org.mybatis.spring.mapper.MapperFactoryBean"*>

<propertyname=*"mapperInterface"*value=*"test.dao.IUserDao"*/>

<propertyname=*"sqlSessionFactory"*ref=*"sqlSessionFactory"*/>

</bean>

<!--memcached注入-->

<beanid=*"memcachedClient"*class=*"net.spy.memcached.spring.MemcachedClientFactoryBean"*>

<propertyname=*"servers"*value=*"localhost:11211"*/>

<propertyname=*"protocol"*value=*"BINARY"*/>

<propertyname=*"transcoder"*>

<beanclass=*"net.spy.memcached.transcoders.SerializingTranscoder"*>

<propertyname=*"compressionThreshold"*value=*"1024"*/>

</bean>

</property>

<propertyname=*"opTimeout"*value=*"50"*/>

<propertyname=*"timeoutExceptionThreshold"*value=*"1998"*/>

<propertyname=*"hashAlg"*>

<valuetype=*"net.spy.memcached.DefaultHashAlgorithm"*>KETAMA\_HASH</value>

</property>

<propertyname=*"locatorType"*value=*"CONSISTENT"*/>

<propertyname=*"failureMode"*value=*"Redistribute"*/>

<propertyname=*"useNagleAlgorithm"*value=*"false"*/>

</bean>

<!--为业务逻辑层注入数据的对象 -->

<beanid=*"userServer"*class=*"test.server.UserServerImpl"*>

<propertyname=*"userDao"*ref=*"userMapper"*></property>

<propertyname=*"memcachedClient"*ref=*"memcachedClient"*></property>

</bean>

<beanid=*"login"*class=*"test.controller.UserController"* >

<propertyname=*"server"* ref=*"userServer"*></property>

</bean>

</beans>

## 配置数据库连接参数文件（jdbc.properties）

jdbc.driverClassName=com.mysql.jdbc.Driver

jdbc.url=jdbc\:mysql\://localhost\:3306/demo?user\=test&password\=test

## 编写spring控制器调用的服务接口（IUserServer）

packagetest.server;

importtest.bean.User;

importtest.dao.IUserDao;

public interface IUserServer {

public User testMethod(String userName);

}

## 编写spring控制器调用的服务类（UserServerImpl）

packagetest.server;

importnet.spy.memcached.MemcachedClient;

importorg.springframework.beans.factory.annotation.Autowired;

importtest.bean.User;

importtest.dao.IUserDao;

public class UserServerImpl implements IUserServer{

privateIUserDaouserDao;

privateMemcachedClientmemcachedClient;

publicIUserDaogetUserDao() {

returnuserDao;

}

//依赖注入，根据属性名自动注入

@Autowired

public void setUserDao(IUserDaouserDao) {

this.userDao = userDao;

}

publicMemcachedClientgetMemcachedClient() {

returnmemcachedClient;

}

//依赖注入（分布式缓存，在spring中自动生成）

@Autowired

public void setMemcachedClient(MemcachedClientmemcachedClient) {

this.memcachedClient = memcachedClient;

}

public User testMethod(String userName){

User user;

//判断缓存中数据是否存在，如果不存在则添加，存在则读取

if(this.memcachedClient.get("user")!=null){

user=(User) this.memcachedClient.get("user");

}else{

user=userDao.getUser(userName);

this.memcachedClient.add("user", 7200, user);

}

return user;

}

}

## 编写spring访问的控制器(UserController)

注意：controller必须位置springMVC配置文件中组件扫描路径下

packagetest.controller;

importjavax.servlet.http.HttpServletRequest;

importjavax.servlet.http.HttpServletResponse;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.stereotype.Controller;

importorg.springframework.web.bind.annotation.RequestMapping;

importorg.springframework.web.servlet.ModelAndView;

importtest.bean.User;

importtest.server.UserServer;

/\*\*

\* 登录控制器

\* @author tanfei

\* @date Feb 1, 2013 9:38:13 AM

\*/

@Controller

public class UserController {

UserServer server;

publicUserServergetServer() {

return server;

}

//依赖注入，根据属性名自动注入

@Autowired

public void setServer(UserServer server) {

this.server = server;

}

//根据访问连接调用控制器，此控制器的调用连接为localhost:8080/demo/login.do

@RequestMapping("login")

publicModelAndViewhandleRequest(HttpServletRequest request, HttpServletResponse response) throws Exception

{

//创建ModelAndView对象，login为返回的jsp页面的名称，全路径是根据在springMVC配置文件中配置的前缀与后缀拼接而成

ModelAndView mode= new ModelAndView("login");

User user=server.testMethod("aa");

//将对象加入mode返回到前台页面

mode.addObject("user", user);

return mode;

}

}

## 编写jsp页面显示后台控制器的返回值（login.jsp）

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<%@pagelanguage=*"java"*import=*"java.util.\*"*pageEncoding=*"gbk"*%>

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

<html>

<head>

<title>My JSP 'index.jsp' starting page</title>

</head>

<body>

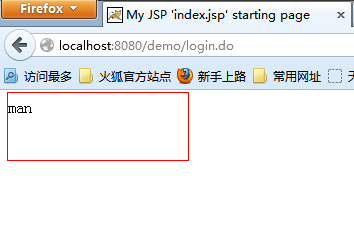
//该实体类有三个属性。ID，姓名（name），性别（gender）

${user.gender}

</body>

</html>

到此即可访问localhost：8080/demo/login.do，如果不出问题则会显示数据库中的gender属性，至此，最简单的配置流程即完成



运行结果