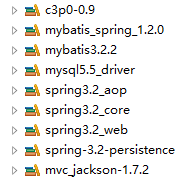
## 10.4 ssi整合

### 10.4.1创建项目

新建项目后规划好各层的包。

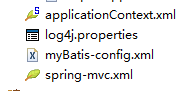
### 10.4.2导入包



### 10.4.3整合spring与mybatis

调整spring与mybatis配置文件

### 10.4.4创建、编写配置文件：



myBatis-config.xml文件

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

<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<!-- 通过别名简化对类的使用

<typeAliases>

<typeAlias type="cn.itcast.entity.Dept" alias="Dept" />

</typeAliases>

<mappers>

<mapper resource="cn/itcast/entity/DeptMapper.xml" />

</mappers>

-->

</configuration>

applicationContext.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:tx=*"http://www.springframework.org/schema/tx"*

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

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

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

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*

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

*http://www.springframework.org/schema/tx/spring-tx-3.2.xsd*

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

<!-- 配置数据源，记得去掉myBatis-config.xml的数据源相关配置 -->

<bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost:3306/mybatis?useUnicode=true&amp;characterEncoding=UTF-8"* />

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"root"* />

</bean>

<!-- 配置session工厂 -->

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

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

<property name=*"configLocation"* value=*"classpath:myBatis-config.xml"* />

</bean>

<!-- 配置事务管理器,管理数据源事务处理-->

<bean id=*"transactionManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

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

</bean>

<!-- 配置事务通知 -->

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

<tx:attributes>

<!-- 默认只处理运行时异常，可加rollback-for="Exception/Throwable"等处理所有异常或包括错误 -->

*<tx:method name="insert\*" propagation="REQUIRED" rollback-for="Exception"/>*

*<tx:method name="update\*" propagation="REQUIRED" rollback-for="Exception"/>*

*<tx:method name="delete\*" propagation="REQUIRED" rollback-for="Exception"/>*

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

</tx:attributes>

</tx:advice>

<!-- 配置切面织入的范围,后边要把事务边界定在service层 -->

<aop:config>

<aop:advisor advice-ref=*"advice"* pointcut=*"execution(\* cn.itcast.scm.dao.impl.\*.\*(..))"*/>

</aop:config>

<!-- 配置SessionTemplate，已封装了繁琐的数据操作-->

<bean id=*"sqlSessionTemplate"* class=*"org.mybatis.spring.SqlSessionTemplate"*>

<constructor-arg name=*"sqlSessionFactory"* ref=*"sqlSessionFactory"*/>

</bean>

<context:component-scan base-package="\*"/>

</beans>

### 10.4.5编写实体及sql映射文件

* 如没有建库表，先建库表,可参考如下sql:

drop database if exists mybatis;

create database mybatis CHARACTER SET UTF8;

use mybatis;

create table dept(

dept\_id int primary key auto\_increment,

dept\_name varchar(50),

dept\_address varchar(50)

);

insert into dept(dept\_name,dept\_address) values('研发部一部','广州');

insert into dept(dept\_name,dept\_address) values('研发部二部','广州');

insert into dept(dept\_name,dept\_address) values('研发部三部','深圳');

select \* from dept;

* 编写实体类

**public** **class** Dept **implements** Serializable {

**private** Integer deptId;

**private** String deptName;

**private** String deptAddress;

**public** Integer getDeptId() {

**return** deptId;

}

**public** **void** setDeptId(Integer deptId) {

**this**.deptId = deptId;

}

**public** String getDeptName() {

**return** deptName;

}

**public** **void** setDeptName(String deptName) {

**this**.deptName = deptName;

}

**public** String getDeptAddress() {

**return** deptAddress;

}

**public** **void** setDeptAddress(String deptAddress) {

**this**.deptAddress = deptAddress;

}

@Override

**public** String toString() {

**return** "Dept [deptId=" + deptId + ", deptName=" + deptName

+ ", deptAddress=" + deptAddress + "]";

}

}

* sql映射文件，并将相关信息映射到mybatis-config.xml文件。

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

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"cn.itcast.entity.DeptMapper"*>

<resultMap type=*"Dept"* id=*"deptResultMap"*>

<id property=*"deptId"* column=*"dept\_id"* />

<result property=*"deptName"* column=*"dept\_name"* />

<result property=*"deptAddress"* column=*"dept\_address"* />

</resultMap>

<!-- id和命名空间用来定位SQL语句，parameterType表示参数的类型，resultMap返回类型 -->

<select id=*"selectDept"* parameterType=*"Integer"* resultMap=*"deptResultMap"*>

<!--参数的写法#{deptID} -->

select \* from dept where dept\_id=#{deptID}

</select>

<insert id=*"insertDept"* parameterType=*"Dept"*>

insert into dept(dept\_name,dept\_address) values(#{deptName},#{deptAddress});

</insert>

</mapper>

myBatis-config.xml文件修改后为如下内容

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

<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<!-- 通过别名简化对类的使用 -->

<typeAliases>

<typeAlias type=*"cn.itcast.entity.Dept"* alias=*"Dept"* />

</typeAliases>

<mappers>

<mapper resource=*"cn/itcast/entity/DeptMapper.xml"* />

</mappers>

</configuration>

### 10.4.6编写Dao接口及实现

DeptDaoImpl.java

@Repository("deptDao")

**public** **class** DeptDaoImpl{

@Resource

**private** SqlSessionTemplate sqlSessionTemplate;

/\*\*

\* 根据部门编号查询部门信息

\* **@param** deptId 部门编号

\* **@return** 部门信息

\*/

**public** Dept selectDept(Integer deptId){

Dept dept= sqlSessionTemplate.selectOne("cn.itcast.entity.DeptMapper.selectDept", deptId);

**return** dept;

}

/\*\*

\* 添加部门信息

\* **@param** dept 部门信息

\* **@return** 添加成功的记录数

\*/

**public** **int** insertDept(Dept dept){

System.*out*.println("------dao.dept:"+dept);

**return** sqlSessionTemplate.insert("cn.itcast.entity.DeptMapper.insertDept", dept);

}

}

### 10.4.7测试spring与mybatis整合

**public** **class** TestDeptDao {

//@Resource //这里没法使用，后继版本有其它方式可以注入

**static** **private** DeptDaoImpl *deptDao*;

@BeforeClass

**public** **static** **void** setUpBeforeClass() **throws** Exception {

ApplicationContext context =**new** ClassPathXmlApplicationContext("applicationContext.xml");

*deptDao*=(DeptDaoImpl) context.getBean("deptDao");

}

@AfterClass

**public** **static** **void** tearDownAfterClass() **throws** Exception {

}

@Test

**public** **void** testSelectDept() {

System.*out*.println(*deptDao*.selectDept(1));

}

@Test

**public** **void** testInsertDept() {

Dept dept=**new** Dept();

//dept.setDeptId(117);

dept.setDeptName("name117");

dept.setDeptAddress("address117");

System.*out*.println("受影响行数："+*deptDao*.insertDept(dept));

}

}

### 10.4.8整合springmvc

修改web.xml文件，加入springmvc相关信息，编写 控制器类及相关jsp 文件

* spring-mvc.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:mvc=*"http://www.springframework.org/schema/mvc"*

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

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

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*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd*

*"*>

<mvc:annotation-driven></mvc:annotation-driven>

<context:component-scan base-package=*"\*"*/>

</beans>

* web.xml文件配置

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

<web-app version=*"3.0"*

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

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

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

*http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"*>

<display-name></display-name>

<welcome-file-list>

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

</welcome-file-list>

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

</listener>

<context-param>

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

<param-value>classpath:applicationContext.xml</param-value>

</context-param>

<servlet>

<servlet-name>mvc</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>

</servlet>

<servlet-mapping>

<servlet-name>mvc</servlet-name>

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

</servlet-mapping>

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<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>

</web-app>

* 编写控制器类

@Controller

@RequestMapping(value="/dept")

**public** **class** DeptAction {

@Resource

**private** DeptDaoImpl deptDao;

@RequestMapping(value="/insert")

**public** String insert(Dept dept){

System.*out*.println("---action.dept:"+dept);

deptDao.insertDept(dept);

**return** "forward:/jsp/main.jsp";

}

}

* 缩写跳转页面

/jsp/main.jsp

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

<html>

<head>

</head>

<body>

this is main jsp

</body>

</html>

### 10.4.9测试ssi整合

缩写测试页面

index.jsp

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

<html>

<head>

</head>

<body>

<form action=*"dept/insert.action"* method=*"post"*>

名称：<input type=*"text"* name=*"deptName"*><br>

地址：<input type=*"text"* name=*"deptAddress"*><br>

<input type=*"submit"* value=*"ok"*>

</form>

</body>

</html>

## 10.3优化

### 10.3.1中文乱码

中文乱码处理，在web.xml中配置拦截器（参考前面）

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<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>

### 10.3.2添加业务层

1).添加业务层相关包、接口及实现

接口包：cn.itcast.service

实现类包：cn.itcast.service.impl

编写接口与实现类（实现类用@Service进行注解，dao接口结合下边的配置，通过@Autowired方式注入代理实例），略。

### 10.3.3添加dao层接口

略

### 10.3.4修改applicationContext.xml与spring-mvc.xml文件

添加如下内容：

<!-- 把事务边界定在service层 -->

<aop:config>

<aop:advisor advice-ref=*"advice"* pointcut=*"execution(\* cn.itcast.scm.service.impl.\*.\*(..))"*/>

</aop:config>

<!-- 自动扫描组件，要把controller去除，他们是在spring-mvc.xml中配置，如果不去除会影响事务管理。 -->

<context:component-scan base-package=*"cn.itcast"*>

<context:exclude-filter type=*"annotation"* expression=*"org.springframework.stereotype.Controller"* />

</context:component-scan>

<!-- 配置 转换器，对于在basePackage设置的包（包括子包）下的接口类，如果在Mapper.xml文件中定义过，

将被转换成spring的BEAN，在调用 的地方通过@Autowired方式将可以注入接口实例-->

<bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>

<property name=*"sqlSessionFactory"* ref=*"sqlSessionFactory"*/>

<property name=*"basePackage"* value=*"cn.itcast.scm.dao"*/>

</bean>

spring-mvc.xml

<!-- 扫描所有的controller 但是不扫描service -->

<context:component-scan base-package=*"cn.itcast"*>

<context:include-filter type=*"annotation"*

expression=*"org.springframework.stereotype.Controller"* />

<context:exclude-filter type=*"annotation"*

expression=*"org.springframework.stereotype.Service"* />

</context:component-scan>

### 10.3.4修改sql映射文件中命名空间

### 10.3.5修改各层的调用

控制器类通过业务层接口调用业务层，业务层再通过dao接口（可删除dao实现类，及测试类）获取代理对象执行相关SQL，进行数据的操作