1.设置Mybatis后台打印调试sql的两种方式：

方案1：

1:首先将ibatis log4j运行级别调到DEBUG可以在控制台打印出ibatis运行的sql语句

2:添加如下语句

###显示SQL语句部分

log4j.logger.com.ibatis=DEBUG

log4j.logger.com.ibatis.common.jdbc.SimpleDataSource=DEBUG

log4j.logger.com.ibatis.common.jdbc.ScriptRunner=DEBUG

log4j.logger.com.ibatis.sqlmap.engine.impl.SqlMapClientDelegate=DEBUG

log4j.logger.Java.sql.Connection=DEBUG

log4j.logger.java.sql.Statement=DEBUG

log4j.logger.java.sql.PreparedStatement=DEBUG

方案2：

最近发现的一种方式，方便快捷

在mybatis.cfg.xml中增加如下配置

<settings>中增加

<setting name="logImpl" value="STDOUT\_LOGGING" />

============================================================================

#{ }表示一个占位符，

#{value} #{ }中的参数名可以是任意，可以value或者其它名称

#{ }接收pojo(简单java对象 model)对象值 通过ognl表达式读取对象的属性值

通过属性.属性.属性..的方式获取对象属性值

${ }表示拼接sql 将接收到的参数内容不加任何修饰的拼接在sql中 使用

${ }拼接sql会引起sql注入 不建议使用

select \* from student where name LIKE '%${value}%'

${value}：接收输入参数的内容 ，如果传入参数类型是简单类型，#{ }中只能

用value

<select id="seelctByName" （映射文件输入参数的类型）parameterType="java.lang.String"（映射文件中输出结果的类型）resultType="model.Student">

<!-- select \* from student where name LIKE #{value} -->

select \* from student where name LIKE '%${value}%' <!-- ${}字符串拼接 -->

</select>

主键和非主键返回的设置

====================================================================================

为了在往后台数据库插入数据时 返回对应的主键id 如果不通过下面那样的设置的话 返回的主键都是0

<mapper namespace="mapper.teacherMapper">

<insert id="save" parameterType="model.Teacher">

<selectKey keyProperty="id" order="AFTER" resultType="java.lang.Integer">

select last\_insert\_id();//mysql数据库中方法

</selectKey>

--------------------------------------------------------------------------

插入数据时 返回非主键id时的设置如下方法

<selectKey keyProperty="id" order="BEFORE" resultType="java.lang.String">

select uuid;//mysql数据库中方法

</selectKey>

insert into teacher (id,name,age)value(#{id}#{name},#{age})

此时id为String类型的非主键

使用mysql的uuid（）生成主鍵

执行过程：

首先通uuid（）得到主鍵，將主键设置得到的Teacher对象的id属性

其次在insert 执行时，从Teacher对象中取出id属性值

----------------------------------------------------------------------------

insert into teacher (name,age)value(#{name},#{age})

</insert>

</mapper>

==============================================================================

修改数据时 在后台必须设置id属性

>>TeacherMapper.xmL

<update id="updateById" parameterType="model.Teacher">

update teacher set `name`=#{name},age=#{age}

where id=#{id};

</update>

>>test

teacher.setId(5);

teacher.setAge("88");

teacher.setName("zhangsanaaa");

session.update("mapper.teacherMapper.updateById", teacher);

=================================================================================

sqsseionFactoryBuilder 创建会话工厂sqlsessionfactory

将sqlsessionfactorybuilder当成一个工具包去使用

把sqlsessionfactory设计成单例模式 创建sqlsessionfactory时直接new sqlsessionfactorybuilder即可

sqlsession是线程不安全的，在sqlsession实现类中除了有接口中的方法（操作数据库方法）和对象属性

所以sqlsession最好的应用场景是在方法体内 是局部变量

===============================================================================

mybatis开发dao方法 mapper代理开发方法

1.<mappers>

<mapper resource="mapper/StudentMapper.xml"/>

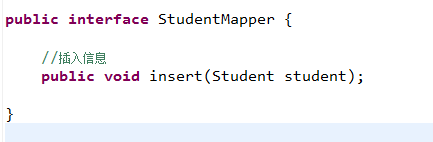
<mapper resource="mapper/TeacherMapper.xml"/>

</mappers>

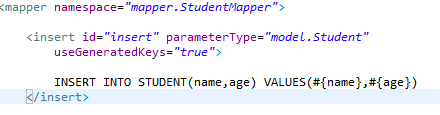
1. 配置StudentMapper.xml
2. StudentMapper.java为接口



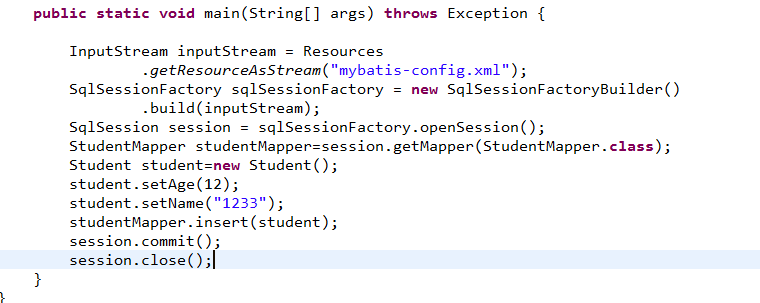
：StudentMapper.java



：StudentMapper.xml



Test：



StudenMapper里的insert 方法 系统 会直接根据你studentMapper.xml文件里的namespace以及相对应的id和mapper接口里的方法 进行匹配

这样写的好处是相比较传统的dao开发 少了接口的实现类的编写

使用mapper代理开发接口需要注意三点：

1. mapper接口里方法 对应的输入参数 和输出 都要和mapper.xml里的parameterType，

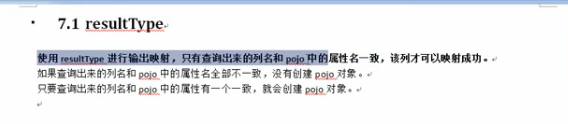
resultType相对应

2.mapper.xml的namespace要对应相对应的mapper接口

3mapper.xml的id要对应相对应的mapper接口里的方法

===============================================================================

resultType 和resultMap的区别在于 ：



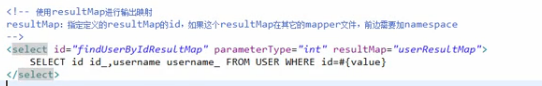


resultMap使用方法：

先定义一个resultMap 然后再使用这个定义了的resultMap



使用刚刚定义的resultMap



resultMap是一种高级映射

======================================================

动态sql

Mybatis 核心是对sql语句进行灵活的操作，通过表达式进行判断 对sql进行灵活的

拼接。

===============================================================

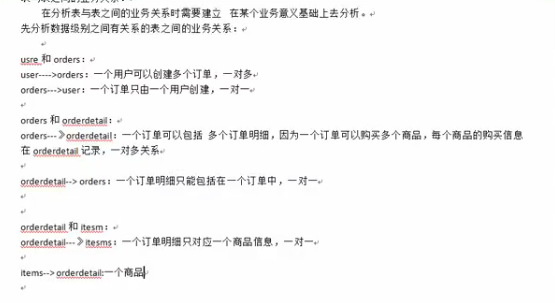
数据模型分析思路：

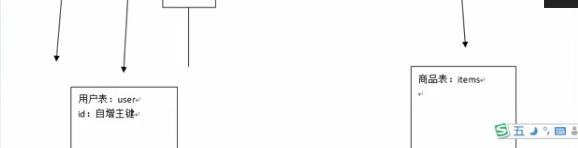
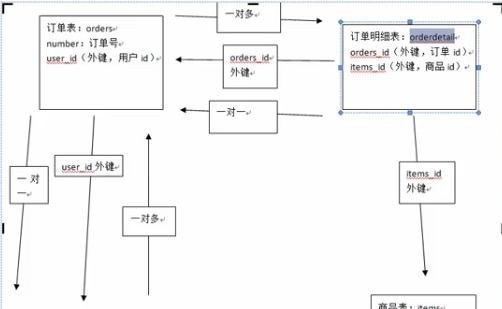
1. 熟悉需求 要分析哪个模块 相当于学习需求的过程
2. 搞清楚每张表重要字段的设置

哪些是重要字段？ 非空字段绝对是重要字段以及是外键的字段

1. 数据库级别表与表之间的关系也就是外间关系
2. 分析表与表之间的业务关系

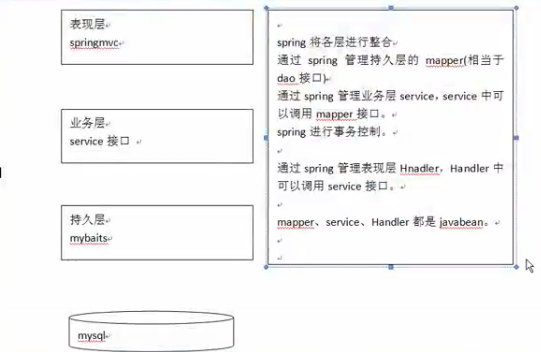
一定要建立在某个业务意义的基础之上去分析



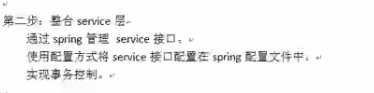


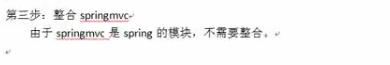
==========================================================

Spring 和mybatis的整合

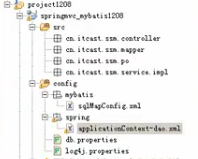




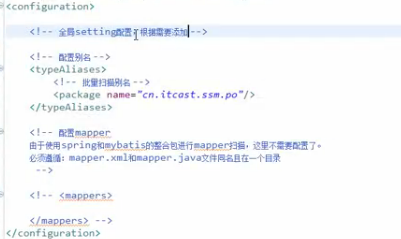




整合的项目架构设计



1.Mybatis 的配置文件 sqlMapConfig.xml

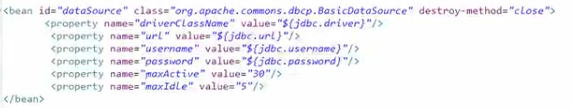


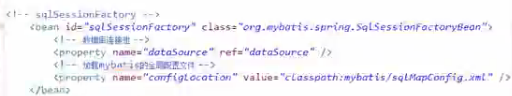
spring的配置文件

1. applicationContext-dao.xml







c



1. applicationContext-service.xml



5.applicationContext-transcation.xml

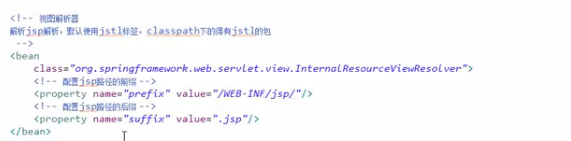


6.整合springmvc

Springmvc.xml







1. 配置前端控制器

web.xml



开始开发------------------

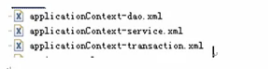


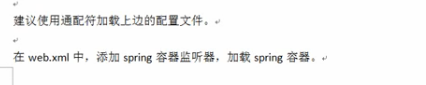














\* 指的就是符合上述条件的xml 加载到spring的ioc容器中

测试

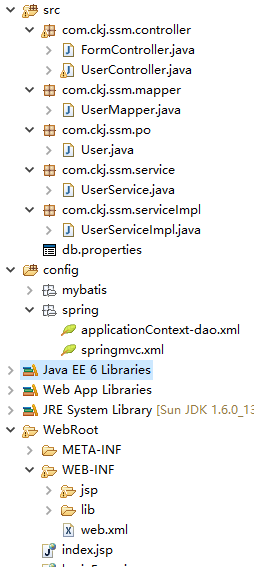




总结：

整合项目之后的心得：

整体框架：

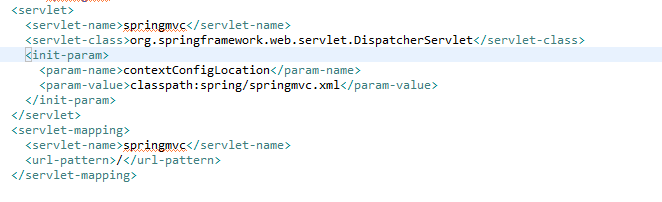




第一loginForm.jsp的编写



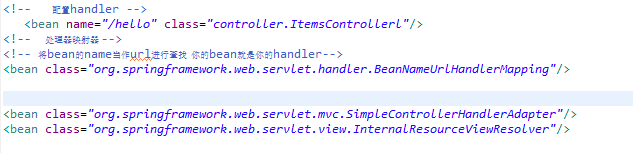
第二：配置文件 web.xml



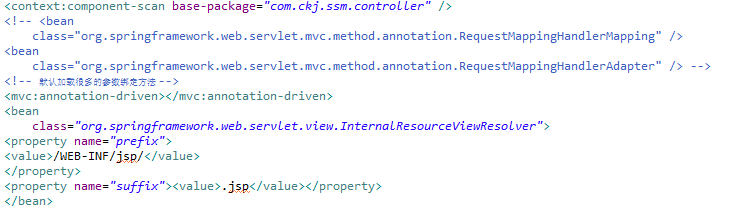
Jsp页面提交的请求/login经过上面的web.xml配置文件到Springmvc.Xml。



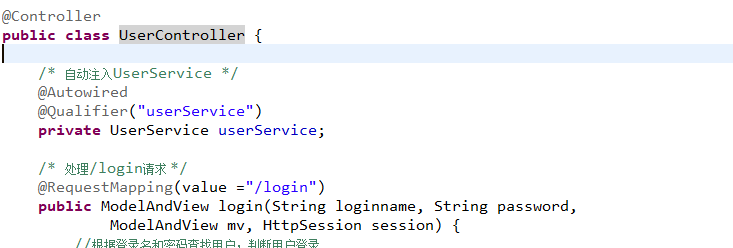
通常这里有两种开发方式：一种是注解controller 还有一种是自己配置handler



注解开发



扫描controller 包找/login对应的controller



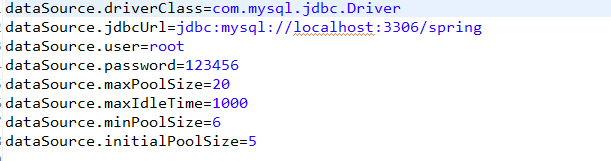


到这里就完成了 login请求对应的页面跳转实现了， 接下来进入 利用mybatis连接数据库

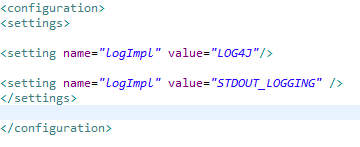
进行取数据 来操作。

首先需要先配置application-dao.xml

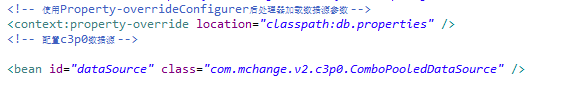
/springmybatis/src/db.properties

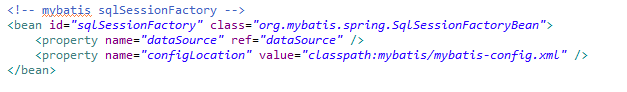


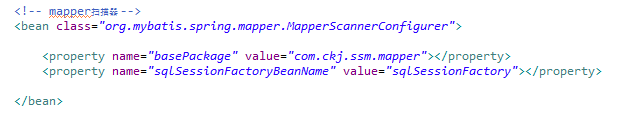
/springmybatis/config/mybatis/mybatis-config.xml



/springmybatis/config/spring/applicationContext-dao.xml

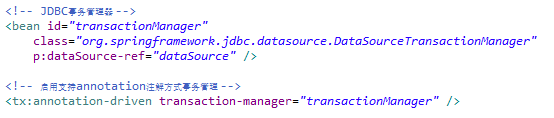




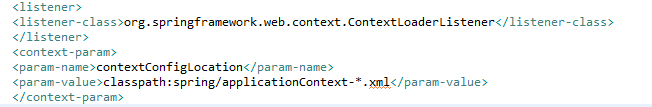




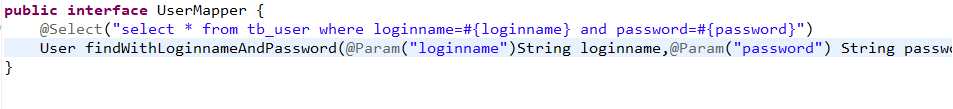
少了这个则无法使用注解了



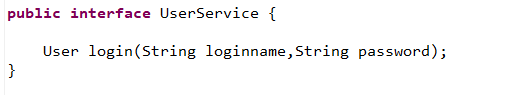
配置文件写好之后 还要记得在web.xml中 进行加载这些配置文件到spring的ioc容器



接下来 先写UserMapper接口



写方法的接口



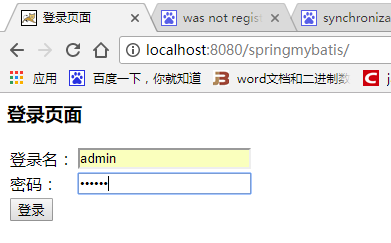
写方法接口的实现类

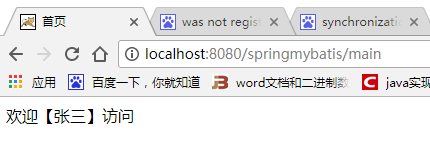


这些都写好之后就可以在controller里调用这些userService方法了



运行截图：





最后整合完成！！！