**MyBatis是什么?**

开源的持久层框架，底层仍然是jdbc。   
（了解）   
Jdbc:易掌握、性能好、代码繁琐   
Hibernate：难掌握、性能不好（复杂的查询经常需要优化sql），代码简介，可以不写sql。   
Mybatis：比较容易掌握，性能一般，代码简洁，需要些sql。

**编程步骤()**

Step1：导包（mybatis,ojdbc，junit）   
Step2:添加MyBatis配置文件

SqlMapConfig.xml的配置：

<configuration>

<environments default="environment">

<environment id="environment">

<transactionManager type="JDBC" />

<dataSource type="POOLED">

<property name="driver" value="oracle.jdbc.driver.OracleDriver" />

<property name="url"

value="jdbc:oracle:thin:@localhost:1521:xe" />

<property name="username" value="SYSTEM" />

<property name="password" value="tarena" />

</dataSource>

</environment>

</environments>

<mappers>

<!-- 指定映射文件的位4子，注意用/隔开-->

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

</mappers>

</configuration>

SqlMap.xml的配置：

<mapper namespace="test">

<!-- id:每一条sql语句应该有唯一的id（称之为sqlID）

parameterType:参数类型，要写类的完整的名字 -->

<insert id="save" parameterType="entity.Employee">

insert into t\_emp values(t\_emp\_seq.nextval,#{ename},#{salary},#{age})

</insert>

<!-- resultType：返回类型（要写类的完整的名字） -->

<select id="findAll" resultType="entity.Employee">

select \* from t\_emp

</select>

<select id="findById" parameterType="int" resultType="entity.Employee">

select \* from t\_emp where id = #{id1}

</select>

<update id="update" parameterType="entity.Employee">

update t\_emp set ename=#{ename},salary=#{salary},age=#{age} where id=#{id}

</update>

<delete id="delete" parameterType="int">

delete from t\_emp where id=#{id1}

</delete>

Step3：添加实体类。注：实体类的属性名要和表的字段名要一样（大小写可以忽略）

package entity;

public class Employee {

private int id;

private String ename;

private double salary;

private int age;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

Step4：添加映射文件。注：存放sql语句（不要忘记修改MyBatis配置文件，指定映射文件的位置）

<mappers>

<!-- 指定映射文件的位4子 -->

<mapper resource="entity/EmpMapper.xml" />

</mappers>

Step5：调用Mybatis的api访问数据库

public class TestCase {

private SqlSession session;

@Before

public void init(){

String config="SqlMapConfig.xml";

SqlSessionFactoryBuilder ssfb = new SqlSessionFactoryBuilder();

SqlSessionFactory ssf = ssfb.build(TestCase.class.getClassLoader().getResourceAsStream(config));

//获得SQLSession对象

session = ssf.openSession();

}

@Test

public void test1(){

//调用SQLSession对象提供的方法访问数据库

//第一参数是sql语句的id；第二个参数是对象

Employee e = new Employee();

e.setEname("大道");

e.setAge(22);

e.setId(1);

e.setSalary(213);

session.insert("test.save",e);

//提交事务（添加、修改、删除）

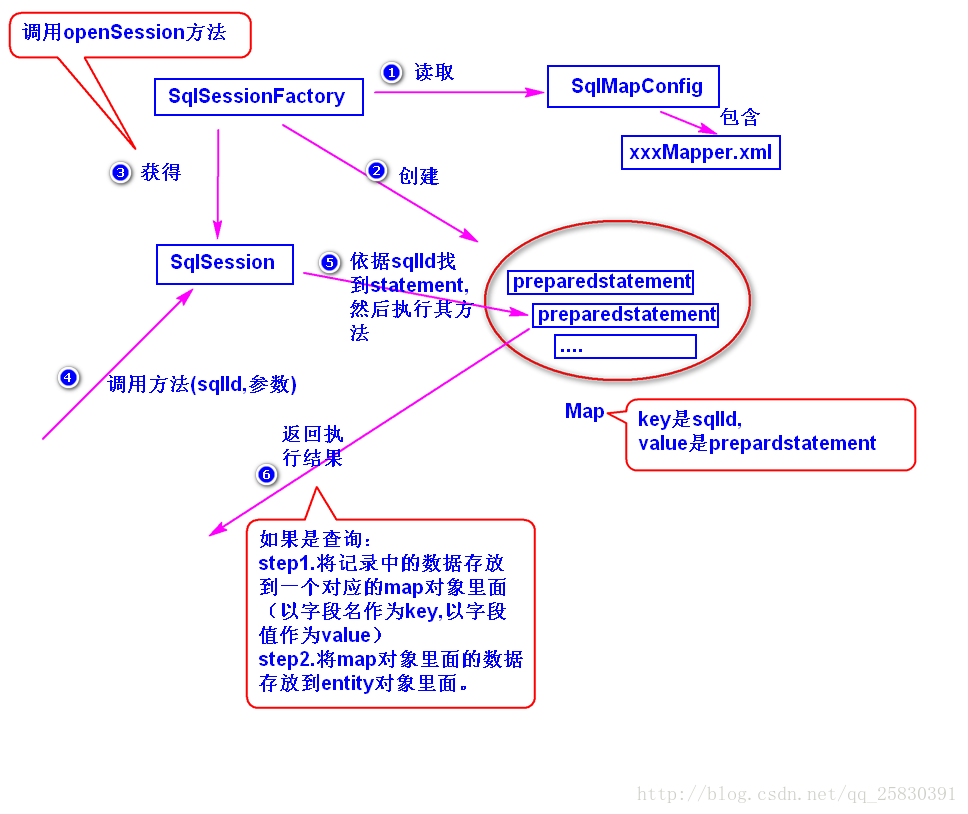
session.commit();

//关闭资源

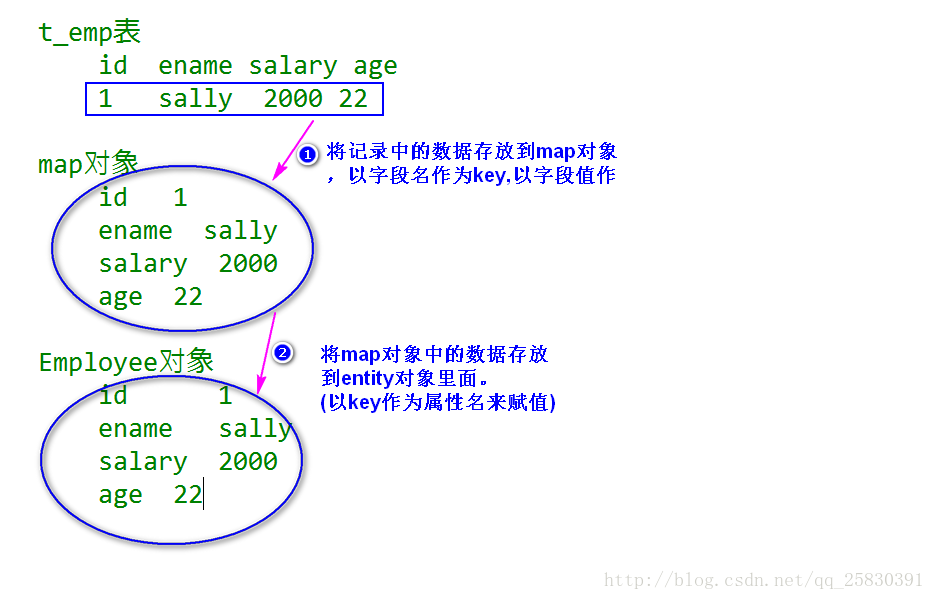
session.close();

}

**原理 (了解)**



**返回Map类型的结果**



EmpMapper.xml配置：

<!-- 返回Map类型的结果 -->

<!-- 即拿到MyBatis查询的中间结果。

注：MyBatis会先将记录存放到一个对应的map对象里面，

然后再将map中的数据存放到对应的实体对象里

map:是java.util.Map的简写形式 -->

<select id="findById2" parameterType="int" resultType="map" >

select \* from t\_emp where id = #{id1}

</select>

测试代码：

@Test

public void test6(){

Map map = session.selectOne("test.findById2",162);

System.out.println(map);

/\*注意，oracle数据库表的字段名都是大写\*/

System.out.println(map.get("ENAME"));

session.close();

}

**解决实体类的属性与表的字段名不一致的情况**

方式一：使用别名：select id empNo,ename name…   
方式二：使用resultMap元素

EmpMapper.xml配置：

<!-- resultMap用来告诉MyBatis，字段名与实体类的属性名的对应关系。 -->

<resultMap type="entity.Emp2" id="emp2ResultMap">

<result property="empNo" column="id"/>

<result property="name" column="ename"/>

</resultMap>

<select id="findById3" parameterType="int" resultMap="emp2ResultMap">

select \* from t\_emp where id = #{id1}

</select>

测试代码

@Test

public void test7(){

Emp2 emp = session.selectOne("test.findById3", 162);

System.out.println(emp);

session.close();

4.4 foreach 用法

SQL语句中有时会使用IN关键字，例如id in (1,2,3)。可以使用${ids}的方式直接获取值，但这种写法不能防止 SQL 注入，想避免 SQL 注入就需要用#{}的方式，这时就要配合使用 foreach 标签满足需求。

foreach 可以对数组、Map 或实现了 Iterable 接口（如 List、Set）的对象进行遍历。数组在处理时会转换为 List 对象进行处理，因此 foreach 遍历的对象可以分为两大类：Iterable 类型和 Map 类型。这两种类型在遍历循环时情况不一样，这一节会通过3个例子来讲解 foreach 的用法。

4.4.1 foreach 实现 in 集合

foreach 实现 in 集合（或数组）是最简单和常用的一种情况，下面介绍如何根据传入的用户 id 集合查询出所有符合条件的用户。首先在 UserMapper 接口中增加如下方法。

/\*\*

\* 根据用户id集合查询

\*

\* @param idList

\* @return

\*/

List<SysUser> selectByIdList(List<Long> idList);

在 UserMapper.xml 中增加如下代码。

<select id="selectByIdList" resultType="tk.mybatis.simple.model.SysUser">

select id,

user\_name userName,

user\_password userPassword,

user\_email userEmail,

user\_info userInfo,

head\_img headImg,

create\_time createTime

from sys\_user

where id in

<foreach collection="list" open="(" close=")" separator="," item="id" index="i">

#{id}

</foreach>

</select>

foreach 包含以下属性。

collection：必填，值为要迭代循环的属性名。这个属性值的情况很多。

item：变量名，值为从迭代对象中取出的每一个值。

index：索引的属性名，在集合数组情况下值为当前索引值，当迭代循环的对象是 Map 类型时，这个值为 Map 的key（键值）。

open：整个循环内容开头的字符串。

close：整个循环内容结尾的字符串。

separator：每次循环的分隔符。

collection 的属性要如何设置呢？来看一下 MyBatis 是如何处理这种类型的参数的。

1. 只有一个数组参数或集合参数

以下代码是 DefaultSqlSession 中的方法，也是默认情况下的处理逻辑。

private Object wrapCollection(final Object object) {

if (object instanceof Collection) {

StrictMap<Object> map = new StrictMap<Object>();

map.put("collection", object);

if (object instanceof List) {

map.put("list", object);

}

return map;

} else if (object != null && object.getClass().isArray()) {

StrictMap<Object> map = new StrictMap<Object>();

map.put("array", object);

return map;

}

return object;

}

当参数类型为 Collection 的时候，默认会转换为 Map 类型，并添加一个 key 为 collection 的值（MyBatis 3.3.0版本增加），如果参数类型是 List 集合，那么就继续添加一个 key 为 list 的值（MyBatis 3.2.8及低版本只有这一个key），这样，当collection="list"时，就能得到这个集合，并对它进行循环操作。

当参数类型为 Array 的时候，也会转换成 Map 类型，默认的 key 为 array。当采用如下方法使用数组参数时，就需要把foreach 标签中的 collection 属性值设置为 array。

/\*\*

\* 根据用户id集合查询

\*

\* @param idArray

\* @return

\*/

List<SysUser> selectByIdList(Long[] idArray);

上面说是数组或集合类型的参数默认的名字，推荐使用@Param来指定参数的名字，这时 collection 就设置为通过@Param注解指定的名字。

2. 有多个参数

第 2 章中讲过，当有多个参数的时候，要使用@Param注解给每个参数指定一个名字，否则在 SQL 中使用参数时就会不方便。因此将collection 设置为 @Param 注解指定的名字即可。

3. 参数是 Map 类型

使用 Map 和使用@Param注解方式类似，将 collection 指定为对应 Map 中的 key 即可。如果要循环所传入的 Map，推荐使用@Param注解指定名字，此时可将 collection 设置为指定的名字，如果不想指定名字，就要使用默认值\_parameter。

4. 参数是一个对象

指定为对象的属性名即可。当使用对象内多层嵌套的对象时，使用属性.属性（集合和数组可以使用下标取值）的方式可以指定深层的属性值。

先来看一个简单的测试代码，验证以上说法。

@Test

public void testSelectByIdList(){

SqlSession sqlSession = getSqlSession();

try {

UserMapper userMapper = sqlSession.getMapper(UserMapper.class);

List<Long> idList = new ArrayList<Long>();

idList.add(1L);

idList.add(1001L);

//业务逻辑中必须校验idList.size() > 0

List<SysUser> userList = userMapper.selectByIdList(idList);

Assert.assertEquals(2, userList.size());

} finally {

//不要忘记关闭sqlSession

sqlSession.close();

}

}

该测试输出的日志如下。

DEBUG [main] - ==> Preparing: select id, user\_name userName,

user\_password userPassword,

user\_email userEmail,

user\_info userInfo,

head\_img headImg,

create\_time createTime

from sys\_user

where id in ( ? , ? )

DEBUG [main] - ==> Parameters: 1(Long), 1001(Long)

TRACE [main] - <== Columns: id, userName, userPassword, userEmail,

userInfo, headImg, createTime

TRACE [main] - <== Row: 1, admin, 123456, admin@mybatis.tk,

<<BLOB>>, <<BLOB>>, 2016-06-07 00:00:00.0

TRACE [main] - <== Row: 1001, test, 123456, test@mybatis.tk,

<<BLOB>>, <<BLOB>>, 2016-06-07 00:00:00.0

DEBUG [main] - <== Total: 2

可以观察日志中打印的SQL语句，foreach元素中的内容最终成为了in ( ? , ? )，根据这部分内容很容易就能理解 open、item、separator 和 close 这些属性的作用。

关于不同集合类型参数的相关内容，建议大家利用上面的基础方法多去尝试，帮助更好地理解。

4.4.2 foreach 实现批量插入

如果数据库支持批量插入，就可以通过 foreach 来实现。批量插入是SQL-92新增的特性，目前支持的数据库有 DB2、SQL Server 2008 及以上版本、PostgreSQL 8.2 及以上版本、MySQL、sqlite 3.7.11 及以上版本、H2。批量插入的语法如下。

INSERT INTO tablename (column-a, [column-b, ...])

VALUES ('value-1a', ['value-1b', ...]),

('value-2a', ['value-2b', ...]),

...

从待处理部分可以看出，后面是一个值的循环，因此可以通过 foreach 实现循环插入。

在 UserMapper 接口中增加如下方法。

/\*\*

\* 批量插入用户信息

\*

\* @param userList

\* @return

\*/

int insertList(List<SysUser> userList);

在 UserMapper.xml 中添加如下 SQL。

<insert id="insertList">

insert into sys\_user(

user\_name, user\_password,user\_email,

user\_info, head\_img, create\_time)

values

<foreach collection="list" item="user" separator=",">

(

#{user.userName}, #{user.userPassword},#{user.userEmail},

#{user.userInfo}, #{user.headImg, jdbcType=BLOB}, #{user.createTime, jdbcType=TIMESTAMP})

</foreach>

</insert>

注意：通过 item 指定了循环变量名后，在引用值的时候使用的是属性.属性（如 user.userName）的方式。

针对该方法编写测试如下。

@Test

public void testInsertList(){

SqlSession sqlSession = getSqlSession();

try {

UserMapper userMapper = sqlSession.getMapper(UserMapper.class);

//创建一个user对象

List<SysUser> userList = new ArrayList<SysUser>();

for(int i = 0; i < 2; i++){

SysUser user = new SysUser();

user.setUserName("test" + i);

user.setUserPassword("123456");

user.setUserEmail("test@mybatis.tk");

userList.add(user);

}

//将新建的对象批量插入数据库中，特别注意，这里的返回值result是执行SQL影响的行数

int result = userMapper.insertList(userList);

Assert.assertEquals(2, result);

} finally {

//为了不影响数据库中的数据导致其他测试失败，这里选择回滚

sqlSession.rollback();

//不要忘记关闭sqlSession

sqlSession.close();

}

}

为了使输出的日志不那么长，这里就测试插入两条数据的情况，输出的日志如下。

DEBUG [main] - ==> Preparing: insert into sys\_user(

user\_name, user\_ password,

user\_email, user\_info,

head\_img, create\_time)

values ( ?, ?,?, ?, ?, ?) ,

( ?, ?,?, ?, ?, ?)

DEBUG [main] - ==> Parameters: test0(String), 123456(String),

test@mybatis.tk(String),

null, null, null,

test1(String), 123456(String),

test@mybatis.tk(String),

null, null, null

DEBUG [main] - <== Updates: 2

从日志中可以看到通过批量SQL语句插入了两条数据。

从 MyBatis 3.3.1 版本开始，MyBatis 开始支持批量新增回写主键值的功能（该功能由本书作者提交），这个功能首先要求数据库主键值为自增类型，同时还要求该数据库提供的 JDBC 驱动可以支持返回批量插入的主键值（JDBC 提供了接口，但并不是所有数据库都完美实现了该接口），因此到目前为止，可以完美支持该功能的仅有 MySQL 数据库。由于 SQL Server 数据库官方提供的 JDBC 只能返回最后一个插入数据的主键值，所以不能支持该功能。

如果要在 MySQL 中实现批量插入返回自增主键值，只需要在原来代码基础上做如下修改。

<insert id="insertList" useGeneratedKeys="true" keyProperty="id">

1

和单表一样，增加了useGeneratedKeys和keyProperty两个属性，增加这两个属性后，简单修改测试类，输出 id 值。

//在调用insertList之后

for(SysUser user : userList){

System.out.println(user.getId());

}

执行测试后，可以看到id部分的日志如下。

1023

1024

关于批量插入的内容就介绍这么多，对于不支持该功能的数据库，许多人会通过 select...union all select... 的方式去实现，这种方式不同数据库的实现也不同，并且这种实现也不安全，这里不再提供示例。

4.4.3 foreach 实现动态UPDATE

这一节主要讲当参数类型是 Map 时，foreach 如何实现动态 UPDATE 。

当参数是 Map 类型的时候，foreach 标签的 index 属性值对应的不是索引值，而是 Map 中的 key，利用这个 key 可以实现动态 UPDATE。

现在需要通过指定的列名和对应的值去更新数据，实现代码如下。

<update id="updateByMap">

update sys\_user

set

<foreach collection="\_parameter" item="val" index="key" separator=",">

${key} = #{val}

</foreach>

where id = #{id}

</update>

这里的 key 作为列名，对应的值作为该列的值，通过 foreache 将需要更新的字段拼在 SQL 语句中。

该 SQL 对应在 UserMapper 接口中的方法如下。

/\*\*

\* 通过Map更新列

\*

\* @param map

\* @return

\*/

int updateByMap(Map<String, Object> map);

这里没有通过@Param注解指定参数名，因而MyBatis在内部的上下文中使用了默认值\_parameter作为该参数的key，所以在XML中就使用了\_parameter。编写测试代码如下。

@Test

public void testUpdateByMap(){

SqlSession sqlSession = getSqlSession();

try {

UserMapper userMapper = sqlSession.getMapper(UserMapper.class);

Map<String, Object> map = new HashMap<String, Object>();

//查询条件，同样也是更新字段，必须保证该值存在

map.put("id", 1L);

//要更新的其他字段

map.put("user\_email", "test@mybatis.tk");

map.put("user\_password", "12345678");

//更新数据

userMapper.updateByMap(map);

//根据当前id查询修改后的数据

SysUser user = userMapper.selectById(1L);

Assert.assertEquals("test@mybatis.tk", user.getUserEmail());

} finally {

//为了不影响数据库中的数据导致其他测试失败，这里选择回滚

sqlSession.rollback();

//不要忘记关闭sqlSession

sqlSession.close();

}

}

测试代码输出日志如下。

DEBUG [main] - ==> Preparing: update sys\_user

set user\_email = ? ,

user\_password = ? ,

id = ?

where id = ?

DEBUG [main] - ==> Parameters: test@mybatis.tk(String),

12345678(String),

1(Long),

1(Long)

DEBUG [main] - <== Updates: 1

DEBUG [main] - ==> Preparing: select \* from sys\_user where id = ?

DEBUG [main] - ==> Parameters: 1(Long)

TRACE [main] - <== Columns: id, user\_name, user\_password, user\_email,

user\_info, head\_img, create\_time

TRACE [main] - <== Row: 1, admin, 12345678, test@mybatis.tk,

<<BLOB>>, <<BLOB>>, 2016-06-07 00:00:00.0

DEBUG [main] - <== Total: 1

到这里，foreach 的全部内容就介绍完了，下一节将介绍 bind 的用法。

4.5 bind 用法

bind 标签可以使用 OGNL 表达式创建一个变量并将其绑定到上下文中。在前面的例子中，UserMapper.xml 有一个selectByUser 方法，这个方法用到了 like 查询条件，部分代码如下。

<if test="userName != null and userName != ''">

and user\_name like concat('%', #{userName}, '%')

</if>

使用 concat 函数连接字符串，在 MySQL 中，这个函数支持多个参数，但在 Oracle 中只支持两个参数。由于不同数据库之间的语法差异，如果更换数据库，有些 SQL 语句可能就需要重写。针对这种情况，可以使用 bind 标签来避免由于更换数据库带来的一些麻烦。将上面的方法改为 bind 方式后，代码如下。

<if test="userName != null and userName != ''">

<bind name="userNameLike" value="'%' + userName + '%'"/>

and user\_name like #{userNameLike}

</if>

bind 标签的两个属性都是必选项，name 为绑定到上下文的变量名，value 为 OGNL 表达式。创建一个 bind 标签的变量后，就可以在下面直接使用，使用 bind 拼接字符串不仅可以避免因更换数据库而去修改 SQL，也能预防 SQL 注入。大家可以根据需求，灵活使用 OGNL 表达式来实现功能，关于更多的 OGNL 常用的表达式，会在 4.7 节中详细介绍。

这是我第一次写博客，目的很明确，为了帮助像我一样的springmvc初学者解决一些问题，还有就时对自己返的错误的一个总结。

      从文章标题不难看出，是使用springmvc时出现了404错误，这里对具体的环境做一个描述。

      环境：

      1、项目开发环境为linux，开发工具eclipse。

      2、项目使用maven构建的web工程，框架搭建为spring+springmvc+mybatis。

      问题具体描述：

      从标题不难看出，框架搭建号后，编译部署并启动项目一切正常，并没有报任何的错误，但是从页面访问controller时缺报404[Not Found]的错误。

      解决方案：

      1、在这需要说明的是，由于后台从始至终没有报任何的错误，故我的文章里不会有出错的贴图，有人就会说了，为什么不把自己的配置文件贴给大家看呢，我认为没有必要，  因为接下来我会给大家讲解排错方法，并且贴上已经访问成功的完整代码贴图。其实刚开始一遇见这样的错误，自己也排查了一边没有任何的发现，于是就网上百度各种搜索，其中也搜到一些结果，按照提示操作一通都没有成功。这里我总结一下网上各种帖子的解决方案：

            1、大量文章都是千篇一律的说处理器映射器和处理器适配器以及视图解析器没有配置好，于是教你怎么去配置处理器映射器，这里我总结一个处理器映射器和适配器以及视图解析器的配置

                  1.1、非注解的处理器映射器和适配器配置，注意：在这里处理器映射器的配置必须和处理器适配器的配置是成对的，还有必须在配置文件里声明注入你的controller，并且这个controller必须实现Controller接口。非注解的配置访问是根据声明注入的bean name来进行映射处理的，必须有name,而且name的值必须有根符号（“/”）。

                 1.2、注解的处理器映射器和适配器配置。注解的controller类必须使用@Controller进行注解，类中的方法使用@RequestMapping("")进行注解。注解的配置方式还有一种，即<mvc:annotation-driven/>注解驱动配置，这样更方便，实际项目也是这样使用的。

                 1.3、视图解析器的配置。

                 至此，springmvc的controller需要的处理器映射器和处理器适配器以及视图解析器的配置已经完成，当然实际要运行的时候还必须要有Controller包扫描，不然你的程序在实例化完sqlSessionFactory后会一直卡在那里，不做任何操作，页面也不会有任何的反映。

      2、这网上的第二种解决方法说的最多的就是我上面说的Controller的包扫描没有加或者包扫描的配置是错误的，这如果说其他配置都OK的话，如果没有加包扫描，那么他不回报任何的错误，但是会输出日志，创建SqlSessionFactory的日志是有的，不会什么信息都没有。有的人将Controller的包扫描加成这样

<context:component-scan base-package="com.springmvc.ssm.controller.\*"/>也是什么都没有的。正确的包扫描是这样的：

      3、当然也有说jar包没有，这也有可能是有的，没有jar包报404很正常的。这里要说的是如果出现404，你的判断是jar包没有，那你就得去你的项目发布路径下的WEB-INF下的lib文件夹去找jar包，而不是你的eclipse里看有没有。更需要注意的是，有人在tomcat运行设置过发布路径，那就必须在你设置的发布路径下找，我的设置为tomcat的安装目录下的webapps下了。

      4、其实忘了一个最多的说法，那就是说你的请求地址是错误的，这个只能说自己认真排查，一般人在出现404的时候第一反应就是请求地址的核对，所以这个我也不说了。但是在这里我强调一个现象，就是你的请求地址是正确的，但是你的目录里没有Controller里的方法返回的页面，这样也会出现404,但是这个404和之前的404是不一样的。比如我在我的testController的items映射的方法里返回的itemsLists,实际我的jsp下的JSP文件没有一个itemsLists.jsp，那么页面的错误就会是这样的：

      5、还有一种解决办法说web.xml配置DispatcherServlet的时候servlet的映射关系<url-pattern>配置不正确，应该将<url-pattern>/\*</url-pattern>改为<url-pattern>/\*</url-pattern>，去掉根号后面的星号，当然如果确实这样写了，那这也是一种解决办法。

      以上就是网上所说的解决这一问题的办法，其实也是我遇到这样的问题，在网上一通乱搜，挨个试了个遍。那么网上这么多的解决方法，我们到底应该去按哪个方法执行，像我一样挨个试一遍？就跟我一样，全部试完也不见得能解决你的问题，因为每个人的问题发生的原因都不一样，但是结果都一样，访问Controller失败，返回404。这也就是我写这篇文章的真正原因，我在这里不会贴出我解决该问题的代码，没有用对你，也许你的问题按照我的方法解决不了。那么我在这里将这个问题的解决思路抽取出来，这样大家就不用在网上搜各种解决方案还不一定适合你，只要按着这个思路找问题的根源，就不会有多大的问题。

      解决方案思路：

      1、分析问题原因

            我们既然知道是404，那就肯定是路径的问题，至少在现有的开发环境中大多数404都是各种路径的问题，不管是没有对应的文件还是路径写错了。既然404是路径问题，那么我们再细化这个路径问题，把这个路径可以分为前台访问路径和后台映射路径。

           1.1、前台访问路径：前台访问路径不用说就是我们在地址栏输入的路径，即http请求路径。

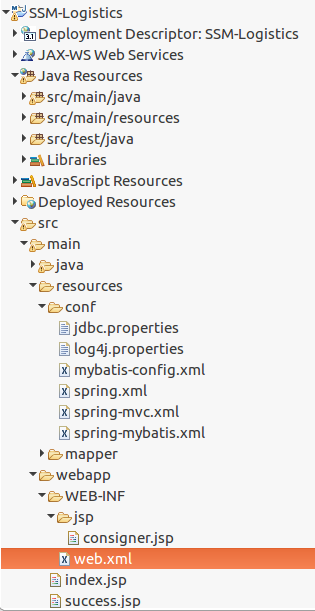
           1.2、后台映射路径：那么后台映射路径就多了，分析这个问题其实也很简单，既然是映射路径，那么他是怎么映射的，从哪里映射的，映射到哪里去，完成这三个问题也就解决了这个问题了。

                    1.2.1、在哪里映射、映射到哪里去：其中从哪里映射和映射到那里去，这个问题很好回答，既然是springmvc，那么这个问题肯定在Controller里了，当然这里的映射到那里去也要根据你自己配置视图解析器的规则去写。

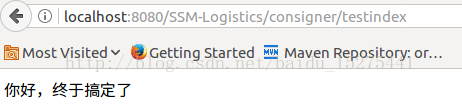
                    1.2.2、怎么映射：然后我们分析这个路径是怎么映射的，这里就有人说了，一个刚学的就去研究spring源码吗？其实我们一般情况下没必要去翻源码查看他的实现原理，但是你如果有能力看他的源码，会加深你对springmvc的理解，对你也是有益无害的。虽然说我们没有必要翻源码但是必须知道他的映射关系是怎么配置的，弄清楚这些，我相信你的问题也就解决了。

       2、解决问题步骤

             我们已经把这个问题细化了，那么就一步步来解决就行了，在这里我会穿插的将我的完整代码贴出来，大家可以参考，初学者直接拿去也是可以运行的。在这里我先把我的工程目录给大家放在这里。

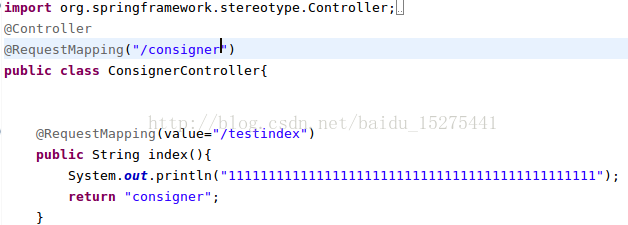


             2.1、前台访问路径：这个没有什么，只能是仔细的比对的你的请求路径是否正确，从项目工程名到到映射的方法名，再到返回的jsp名称。



             2.2、从哪里映射，映射到哪里去：这个问题我们就考虑Controller类中映射是不是不正确，查看@RequestMapping("/映射地址")中的映射地址是否与访问地址对应。

返回路径是否正确，例如返回consigner，在视图解析器配置的路径规则下能不能找到consigner.jsp文件。



               2.3、这里是重点，看他是怎么映射的。我们知道springmvc是有前端控制器来调配映射器和适配器去工作的，那么我们首先要确保前端控制器配置正确。前端控制器配置好了，还需要处理器映射器和适配器以及视图解析器都配置好才可以。那么我们就得一个一个得排查这些所谓的各种器都配置好了吗？我把我的配置贴出来给大家参考。

               映射器、适配器和视图解析器：spring-mvc.xml

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

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

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

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

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

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

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/>

<!-- 扫描controller（controller层注入） -->

<context:component-scan base-package="com.springmvc.ssm.controller"/>

<!-- 非注解的bean声明注入 -->

<!-- <bean name="/testController" class="com.springmvc.ssm.controller.TestController"/> -->

<!-- 非注解处理器映射器 -->

<!-- <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"/> -->

<!-- 非注解处理器适配器 -->

<!-- <bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter"/> -->

<!-- 注解映射器 -->

<!-- <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/> -->

<!-- 注解适配器 -->

<!-- <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"/> -->

<!-- 避免IE在ajax请求时，返回json出现下载 -->

<bean id="jacksonMessageConverter" class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter">

<property name="supportedMediaTypes">

<list>

<value>text/html;charset=UTF-8</value>

</list>

</property>

</bean>

<!-- 对模型视图添加前后缀 -->

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

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

</beans>

                 前端控制器配置：web.xml

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

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

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

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"

version="3.1"

metadata-complete="true">

<context-param>

<param-name>ssm-logistics</param-name>

<param-value>ssm-logistics.root</param-value>

</context-param>

<!-- 配置 Spring -->

<context-param>

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

<param-value>classpath:conf/spring.xml;classpath:conf/spring-mybatis.xml</param-value>

</context-param>

<!-- 日志记录 -->

<context-param>

<!-- 日志配置文件路径 -->

<param-name>log4jConfigLocation</param-name>

<param-value>classpath:conf/log4j.properties</param-value>

</context-param>

<context-param>

<!-- 日志页面的刷新间隔 -->

<param-name>log4jRefreshInterval</param-name>

<param-value>6000</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>

</listener>

<listener>

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

</listener>

<!-- 防止Spring内存溢出监听器 -->

<listener>

<listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>

</listener>

<!-- springMVC核心配置 -->

<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:conf/spring-mvc.xml</param-value>

</init-param>

<load-on-startup>2</load-on-startup>

</servlet>

<servlet-mapping>

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

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

</servlet-mapping>

<welcome-file-list>

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

</welcome-file-list>

<!-- 错误跳转页面 -->

<error-page>

<!-- 路径不正确 -->

<error-code>404</error-code>

<location>/WEB-INF/errorpage/404.jsp</location>

</error-page>

<error-page>

<!-- 没有访问权限，访问被禁止 -->

<error-code>405</error-code>

<location>/WEB-INF/errorpage/405.jsp</location>

</error-page>

<error-page>

<!-- 内部错误 -->

<error-code>500</error-code>

<location>/WEB-INF/errorpage/500.jsp</location>

</error-page>

<!-- 字符集过滤器 -->

<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>

<init-param>

<param-name>forceEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

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

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

</filter-mapping>

</web-app>

           2.4、如果说前面这些都配置好了，问题还是存在，我们就得去查看配置的这些器都起作用了没有，即这些配置文件有没有一步步的都加载到spring容器中。那么我们要在web.xml里面加载那些东西呢？因为我是SSM框架，所以spring的配置文件，springmvc的配置文件，mybatis的配置文件，mysql数据库的配置文件，spring和mybatis整合的配置文件等都要加载到spring容器中。当然每个人写配置文件的方式不一样，加载的方式也不一样，但是都有一定的约定必须遵循。我是这样加载的：大家可以看的web.xml配置文件，我在这里面加载了spring配置文件和springmvc的配置文件以及spring和mybatis整合的配置文件。现在将另外两个配置文件贴出来：

           spring.xml

           spring这里我引入了jdbc的配置文件，还加了service包的自动扫描，将service层注入到容器中

<?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">

<!-- 引入jdbc配置文件 -->

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

<!-- 扫描文件（自动将servicec层注入） -->

<context:component-scan base-package="com.springmvc.ssm.service"/>

</beans>

           spring-mybatis.xml

           这里我设置了连接池和事务管理，引入了mybatis的配置文件，加了mapper.xml及dao文件的自动扫描注入

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

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

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

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.2.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

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

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

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init"

destroy-method="close" >

<property name="driverClassName">

<value>${jdbc\_driverClassName}</value>

</property>

<property name="url">

<value>${jdbc\_url}</value>

</property>

<property name="username">

<value>${jdbc\_username}</value>

</property>

<property name="password">

<value>${jdbc\_password}</value>

</property>

<!-- 连接池最大使用连接数 -->

<property name="maxActive">

<value>20</value>

</property>

<!-- 初始化连接大小 -->

<property name="initialSize">

<value>1</value>

</property>

<!-- 获取连接最大等待时间 -->

<property name="maxWait">

<value>60000</value>

</property>

<!-- 连接池最大空闲 -->

<property name="maxIdle">

<value>20</value>

</property>

<!-- 连接池最小空闲 -->

<property name="minIdle">

<value>3</value>

</property>

<!-- 自动清除无用连接 -->

<property name="removeAbandoned">

<value>true</value>

</property>

<!-- 清除无用连接的等待时间 -->

<property name="removeAbandonedTimeout">

<value>180</value>

</property>

<!-- 连接属性 -->

<property name="connectionProperties">

<value>clientEncoding=UTF-8</value>

</property>

</bean>

<!-- mybatis文件配置，扫描所有mapper文件 -->

<bean id="sqlSessionFactory"

class="org.mybatis.spring.SqlSessionFactoryBean"

p:dataSource-ref="dataSource"

p:configLocation="classpath:conf/mybatis-config.xml"

p:mapperLocations="classpath:mapper/\*.xml"/><!-- configLocation为mybatis属性 mapperLocations为所有mapper-->

<!-- spring与mybatis整合配置，扫描所有dao -->

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

p:basePackage="com.springmvc.ssm.dao"

p:sqlSessionFactoryBeanName="sqlSessionFactory"/>

<!-- 对数据源进行事务管理 -->

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager"

p:dataSource-ref="dataSource"/>

</beans>

         2.5、如果以上都配置好了，还是不成功，那么就得在看看这些自动化的东西到底有没有自动化成功，例如service的自动扫描有没有起作用，Controller的自动扫描有没有起作用或者说是配置正确不正确。我的Controller的自动扫描是在spring.xml文件完成的。

         2.6、如果说controller配置没有错误，那就看看是不是jar包的问题，记得在工程发布的路径下查看。

         2.7、如果说还不成功，那就只能继续深入看源码了。还有一个方法就是启动的时候使用DEBUG模式，且在发布时去掉autoloading enabled，这样如果是源码问题就会卡出来，可以看到提示。不过相信走到这一步基本也就把错误排查出来了，一般不会去看源码提示，因为除非导包的问题，源码不会有错。

        至此我将我的思路和错误排除的方法告诉大家，希望可以帮助到一些人。这篇文章是一个错误处理的排查思路分析及解决的文档，不是教大家怎么构建工程的文档，所以文件不是很全，但是配置文件都是没有问题的，只要更改目录和名称就可以只在项目中使用了。有什么问题希望留言大家共同探讨。我将其他的文件贴出来，以便大家可以直接复制去调试。

       附件：

       1、jdbc.properties

jdbc\_driverClassName=com.mysql.jdbc.Driver

jdbc\_url=jdbc:mysql://localhost:3306/SSYS\_LOGISTICS?useUnicode=true&characterEncoding=utf-8

jdbc\_username=root

jdbc\_password=root

       2、log4j.properties

### set log levels ###

#log4j.rootLogger = debug , stdout , D , E

log4j.rootLogger = debug , stdout , D

### output to the console ###

log4j.appender.stdout = org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target = System.out

log4j.appender.stdout.layout = org.apache.log4j.PatternLayout

#log4j.appender.stdout.layout.ConversionPattern = %d{ABSOLUTE} %5p %c{ 1 }:%L - %m%n

log4j.appender.stdout.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [%c]-[%p] %m%n

### Output to the log file ###

log4j.appender.D = org.apache.log4j.DailyRollingFileAppender

log4j.appender.D.File = /home/workspace/logs/log.log

log4j.appender.D.Append = true

log4j.appender.D.Threshold = DEBUG

log4j.appender.D.layout = org.apache.log4j.PatternLayout

log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

### Save exception information to separate file ###

log4j.appender.D = org.apache.log4j.DailyRollingFileAppender

log4j.appender.D.File = /home/workspace/logs/error.log

log4j.appender.D.Append = true

log4j.appender.D.Threshold = ERROR

log4j.appender.D.layout = org.apache.log4j.PatternLayout

log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

       3、mybatis-cofig.xml

<?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>

<!-- 命名空间 -->

<typeAliases>

<!-- <typeAlias alias="User" type="com.springmvc.model.User"/> -->

<package name="com.springmvc.ssm.model"/>

<package name="com.springmvc.ssm.pojo"/>

</typeAliases>

<!-- 映射map -->

<mappers>

</mappers>

</configuration>

好了在这里说下我的错误吧，我是在复制web.xml配置文件的时候，没有将mybatis和spring的整合文件加载进来，所以大多数的错误都是人为的，而且是不起眼的小问题，可就是这些不起眼的小问题，总是折腾人。我们还是要认真排查每一行代码，认真对待每一件事。事无大小事，人无贵贱人！哈哈！

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

作者：ITwelision

来源：CSDN

原文：https://blog.csdn.net/baidu\_15275441/article/details/73930961

版权声明：本文为博主原创文章，转载请附上博文链接！

利用Intellij+MAVEN搭建Spring+Mybatis+MySql+SpringMVC项目详解

这两天在公司实习，师父让我先熟悉Intellij软件的使用和SpringMVC框架。话不多说，先利用这些搭建一个小环境吧。

1 创建MAVEN项目：File-NEW-MAVEN-create from..前打勾--选下面的org.apache..archetypes:maven-archetype-webapp,点击next

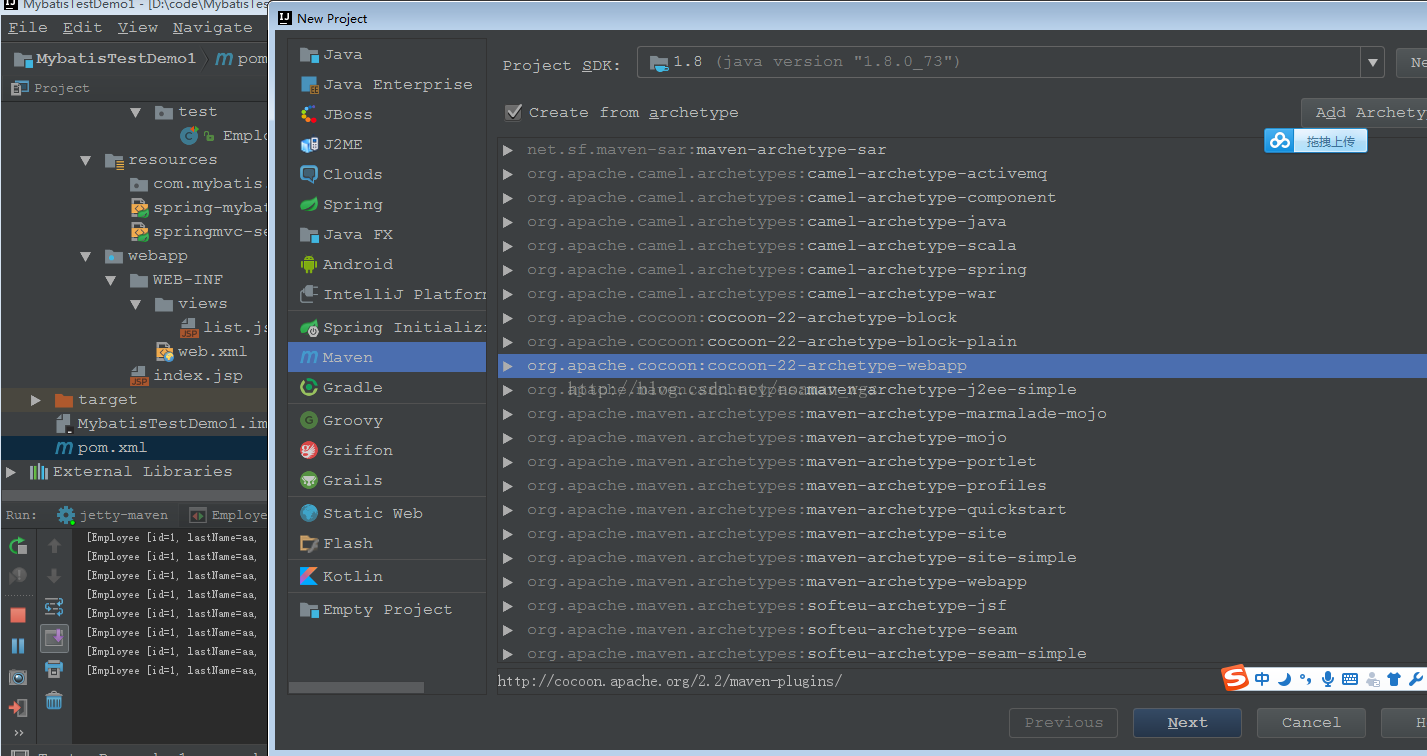
                 后面的项目名称：

GroupId:项目的名称，

ArtiFactId:项目的模块名称（建议用项目名称-模块名称来表示），

Version:项目版本的名称

如：groupID：SpringMVC；ArtiFactId：SpringMVC-Demo，version:默认完成后，IDEA就自动给我们构建了一个空的maven项目.



2  配置pom.xml文件，这里利用<dependency></dependency>标签导入需要的库。这里需要导入的有junit , spring , springmvc , mysql , mybatis , jstl等等，详见下面的配置文件：

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>MybatisTest</groupId>

<artifactId>MybatisTest-Demo1</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<name>MybatisTest-Demo1 Maven Webapp</name>

<url>http://maven.apache.org</url>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!--mysql driver-->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.6</version>

</dependency>

<!--spring-->

<!-- spring-aop -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>4.3.5.RELEASE</version>

</dependency>

<!--spring-tx -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>4.3.5.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>4.3.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>4.3.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.1.RELEASE</version>

</dependency>

<!--Spring Web + Spring MVC-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.3.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.1.RELEASE</version>

</dependency>

<!--NoClassDefFoundError: javax/servlet/jsp/jstl/core/Config-->

<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>1.1.2</version>

</dependency>

<!--servlet/jsp api start-->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.servlet.jsp/jsp-api -->

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>jsp-api</artifactId>

<version>2.2</version>

</dependency>

<!--c3p0-->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5.1</version>

</dependency>

<!--aspectj-->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.8.6</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.8.6</version>

</dependency>

<!--jdbc-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>3.0.5.RELEASE</version>

</dependency>

<!--mybatis-->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.1</version>

</dependency>

<!--mybatis spring整合-->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.0</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>RELEASE</version>

</dependency>

<!--spring-test-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>3.2.3.RELEASE</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>RELEASE</version>

</dependency>

</dependencies>

<build>

<finalName>MybatisTest-Demo1</finalName>

<plugins>

<!--servlet容器 jetty插件-->

<plugin>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<version>9.3.10.v20160621</version>

</plugin>

<!--mybatis 逆向工程插件-->

<plugin>

<groupId>org.mybatis.generator</groupId>

<artifactId>mybatis-generator-maven-plugin</artifactId>

<version>1.3.2</version>

<configuration>

<verbose>true</verbose>

<overwrite>true</overwrite>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

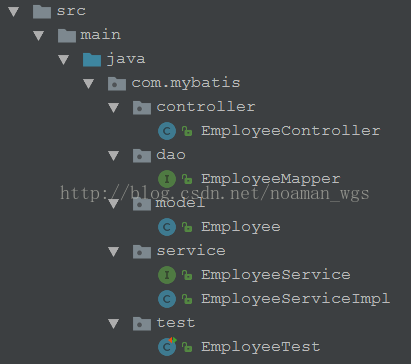
</plugin>

</plugins>

</build>

</project>

3 src下面新建java包，将其设置为source root(在java包上点击右键，选择Mark Directory As Source Root),依次建如图类结构：



1）创建接口EmployeeMapper.java,这里是采用Mybatis的注解的方式，要在后面的spring-mybatis文件中去注册接口

package com.mybatis.dao;

import com.mybatis.model.Employee;

import org.apache.ibatis.annotations.Select;

import java.util.List;

/\*\*

\* Created by wanggenshen\_sx on 2016/12/26.

\*/

public interface EmployeeMapper {

@Select("select id,lastName,email from employee where id=#{id}")

Employee getEmployeeById(int id);

@Select("select \* from employee")

List<Employee> getAllEmployees();

}

2）实体类：Employee.java

package com.mybatis.model;

/\*\*

\* Created by wanggenshen\_sx on 2016/12/23.

\*/

public class Employee {

private int id;

private String lastName;

private String email;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

@Override

public String toString() {

return "Employee [id=" + id + ", lastName=" + lastName + ", email="

+ email+" ]" ;

}

public Employee(){}

}

3) Service层：

创建EmployeeService接口

package com.mybatis.service;

import com.mybatis.model.Employee;

import java.util.List;

/\*\*

\* Created by wanggenshen\_sx on 2016/12/26.

\*/

public interface EmployeeService {

Employee getEmployee(int id);

List<Employee> getEmployees();

}

创建EmployeeServiceImpl实现类：注意添加@Service注解

package com.mybatis.service;

import com.mybatis.dao.EmployeeMapper;

import com.mybatis.model.Employee;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

/\*\*

\* Created by wanggenshen\_sx on 2016/12/26.

\*/

@Service

public class EmployeeServiceImpl implements EmployeeService {

public EmployeeServiceImpl() {

System.out.printf("init EmployeeServiceImpl");

}

@Autowired

private EmployeeMapper employeeMapper;

@Override

public Employee getEmployee(int id){

return employeeMapper.getEmployeeById(id);

}

public List<Employee> getEmployees(){

return employeeMapper.getAllEmployees();

}

}

4）在spring配置文件中整合mybatis:spring-mybatis.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:context="http://www.springframework.org/schema/context"

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

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

<!--自动扫描含有@Service的类，将其注入为bean -->

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

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

<!-- 1. 数据源 : DriverManagerDataSource -->

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

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

<property name="url" value="jdbc:mysql://localhost:3306/mytest"/>

<property name="username" value="root"/>

<property name="password" value="920614"/>

</bean>

<!--

2. mybatis的SqlSession的工厂: SqlSessionFactoryBean

dataSource / typeAliasesPackage

-->

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

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

<property name="typeAliasesPackage" value="com.mybatis.model"/>

</bean>

<!--

3. mybatis自动扫描加载Sql映射文件 : MapperScannerConfigurer

sqlSessionFactory / basePackage

-->

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

<property name="basePackage" value="com.mybatis.dao"/>

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />

</bean>

<!-- 4. 事务管理 : DataSourceTransactionManager -->

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

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

</bean>

<!-- 5. 使用声明式事务 -->

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

</beans>

建测试类：EmployeeTest.java

package com.mybatis.test;

import com.mybatis.model.Employee;

import com.mybatis.service.EmployeeService;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.test.context.ContextConfiguration;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import java.util.List;

/\*\*

\* Created by wanggenshen\_sx on 2016/12/27.

\*/

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration(locations = { "classpath:spring-mybatis.xml" })

public class EmployeeTest {

@Autowired

private EmployeeService employeeService;

@Test

public void testGetEmployeeById(){

Employee employee = employeeService.getEmployee(1);

System.out.print(employee);

}

@Test

public void testGetAll(){

List<Employee> employees=employeeService.getEmployees();

System.out.print(employees);

}

}

能输出并显示数据库表的内容，即表示Spring整合Mybatis成功，接下来加入SpringMVC内容，将读取的内容显示在页面上。

4 创建springmvc-servlet.xml,配置视图信息。

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

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

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

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

http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

<!--注解-->

<mvc:annotation-driven/>

<!--处理静态资源-->

<mvc:default-servlet-handler/>

<!-- 自动扫描controller包下的有@Controller注解的，注入为bean -->

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

<!--配置视图解析器-->

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

<property name="prefix" value="/WEB-INF/views/"/>

<property name="suffix" value=".jsp"/>

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

</bean>

</beans>

5 建控制类：

EmployeeController.java

package com.mybatis.controller;

import com.mybatis.model.Employee;

import com.mybatis.service.EmployeeService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

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

import java.util.List;

import java.util.Map;

/\*\*

\* Created by wanggenshen\_sx on 2016/12/26.

\*/

@Controller

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

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

public String list(Map<String,Object> map){

List<Employee> employees = employeeService.getEmployees();

/\*测试用

if(employees==null)

System.out.println("null");

else

System.out.println(employees);\*/

map.put("employees",employees);

return "list";

}

}

6 建立显示页面: Web-app的WEB-INF下建views包下建list.jsp：

注意此处一定要加上

<%@ page isELIgnored="false" %>

否则无法使用jstl标签。

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%--

Created by IntelliJ IDEA.

User: wanggenshen\_sx

Date: 2016/12/26

Time: 17:25

To change this template use File | Settings | File Templates.

--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%@ page isELIgnored="false" %>

<html>

<head>

<title>Show Page22</title>

</head>

<body>

<table border="1" cellspacing="0" cellpadding="10">

<tr>

<th>ID</th>

<th>LastName</th>

<th>Email</th>

</tr>

<c:forEach items="${requestScope.employees}" var="emp">

<tr>

<th>${emp.id}</th>

<th>${emp.lastName}</th>

<th>${emp.email}</th>

</tr>

</c:forEach>

</table>

</body>

</html>

7 配置web.xml ,分别指定spring,springmvc配置文件的位置。注意这里一定要有四个部分，即<context-param>,<servlet>,<servlet-mapping>,<listener>.

  我在这里没有添加<listener>标签，spring无法初始化，出现了【Caused by: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'employeeController': Unsatisfied dependency expressed through field 'employeeService': No qualifying bean of type [com.mybatis.service.EmployeeService] found for dependency [com.mybatis.service.EmployeeService]: expected at least 1 bean which qualifies as autowire candidate for this dependency. Dependency annotations: 】错误。

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app>

<context-param>

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

<param-value>classpath:spring-mybatis.xml</param-value>

</context-param>

<listener>

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

</listener>

<!-- 防止spring内存溢出监听器 -->

<listener>

<listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>

</listener>

<!--配置springmvc DispatcherServlet-->

<servlet>

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

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

<init-param>

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

<param-value>classpath:springmvc-servlet.xml</param-value>

</init-param>

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

</servlet>

<servlet-mapping>

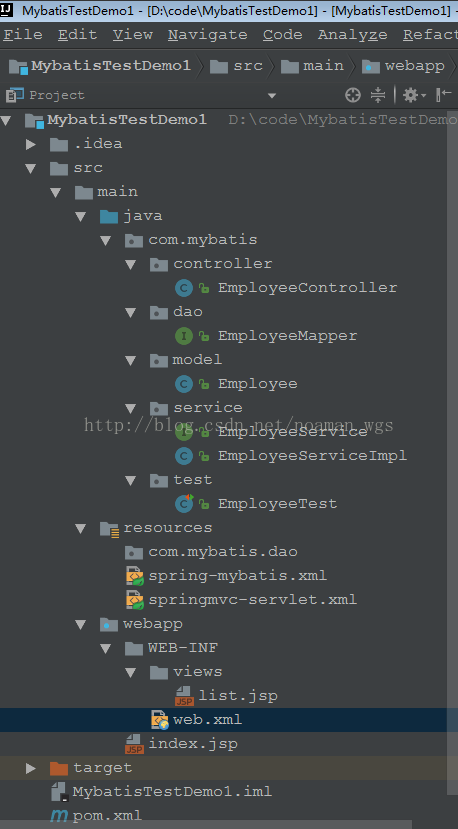
<servlet-name>springmvc-servlet</servlet-name>

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

</servlet-mapping>

</web-app>

8 项目整体结构：



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

作者：是Guava不是瓜娃

来源：CSDN

原文：https://blog.csdn.net/noaman\_wgs/article/details/53893948

版权声明：本文为博主原创文章，转载请附上博文链接！

**一、使用IDEA，利用maven整合ssm框架（spring+springmvc+mybatis+mysql））**

2017年08月27日 21:54:28 [匆匆四月](https://me.csdn.net/zyf2333) 阅读数 6180

**1.工具**

* apache-tomcat 7
* apache-maven（后面详细讲！这东西我一开始也很懵逼）
* mysql 5以上的版本
* navicat（有了这个就不用cmd了，可以直接显示数据库，如下图，请自行百度绿色版QAQ）
* IntelliJ IDEA 2016.3.4（<http://www.jb51.net/softs/527945.html>，根据连接教程慢慢安装吧~）

**2.前言**

前不久学完了spring和mybatis基础，于是决定来整合ssm。但是跟着学长的博客教程走的时候，出现了很多问题，于是打算转载一下他的博客，并加上我自己的理解，来写这么一篇博客。那么，请务必包含耐性，内容较多，慢慢来实现ssm的helloworld入门项目（原出处：<http://blog.csdn.net/lxfhahaha/article/details/57078994>）

**3.步骤**

**1）准备工作**

1.1 mysql数据库安装，此处直接引用学长的教程

<http://blog.csdn.net/lxfhahaha/article/details/56847353>

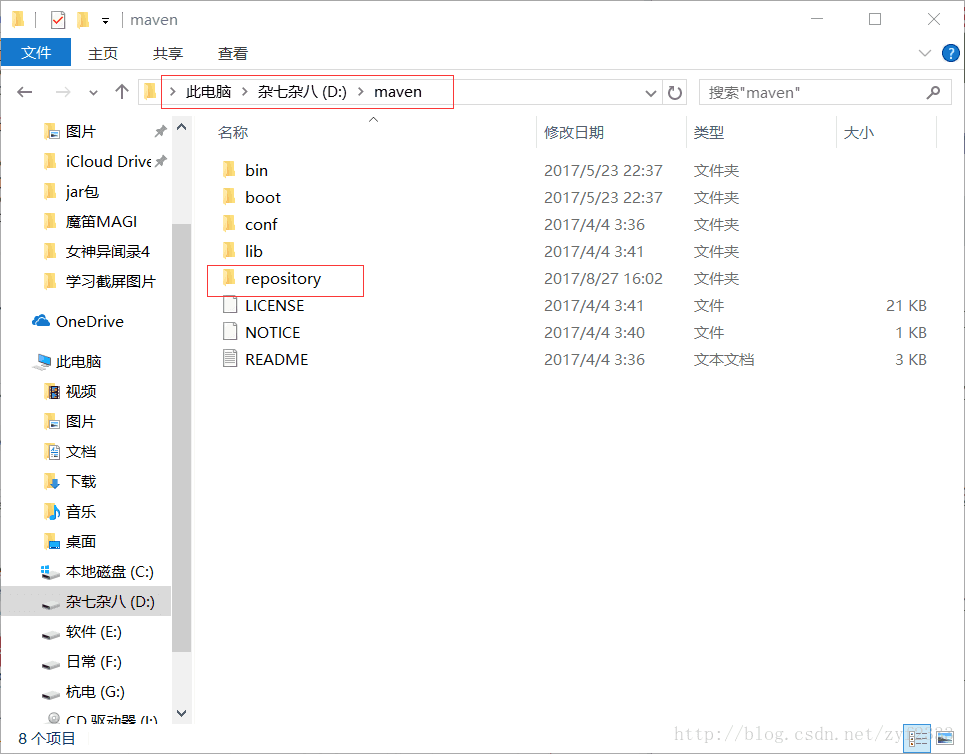
1.2 Tomcat安装

<http://blog.csdn.net/zyf2333/article/details/77387649>。

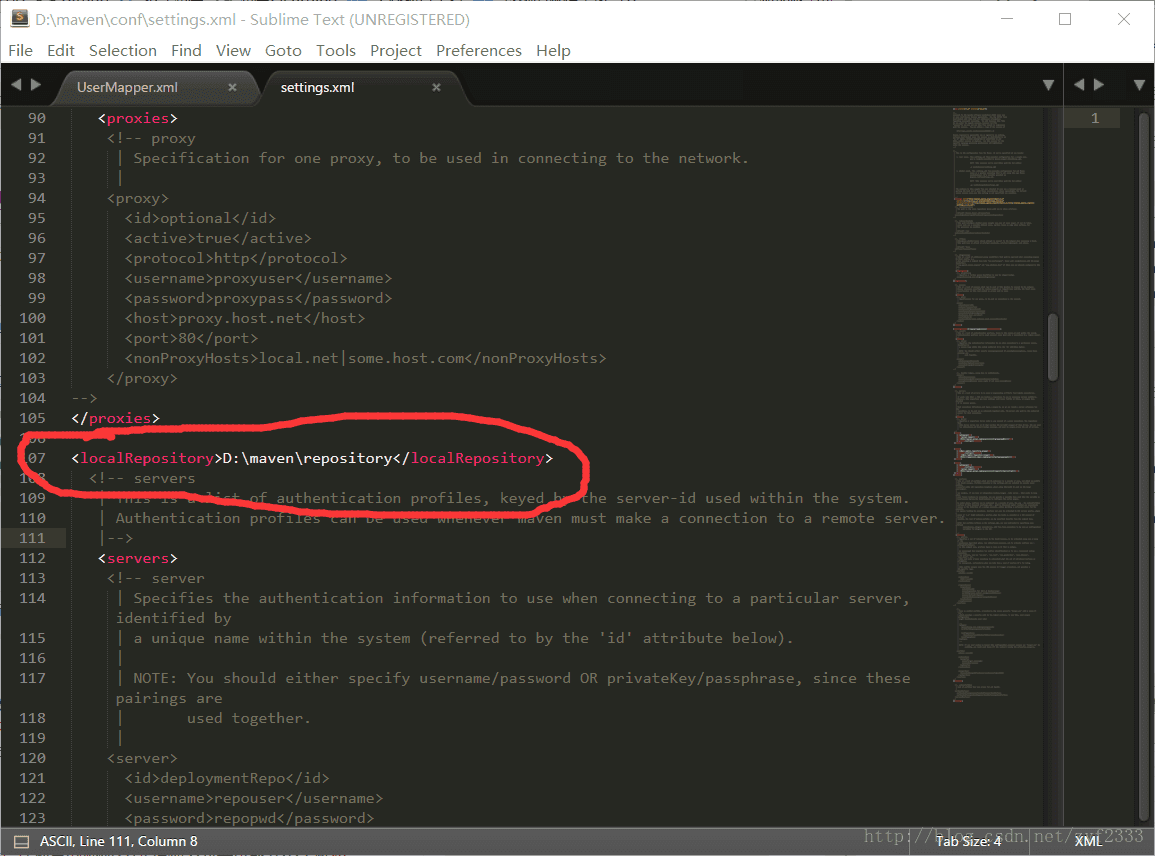
1.3 maven安装

（安装教程：<https://jingyan.baidu.com/article/fd8044fadc1ac15031137a12.html>）

此处我是下载到D盘，然后自己在maven目录下创建了一个repository文件夹（随便取名）



然后打开conf文件夹下的settings.xml，在大约100行左右，修改圈中的目录，此处目录写你刚刚创建的repository文件夹的目录，如上图中，我的repository目录是 D:\maven\repository，那此处就改成我的目录（没有<localRepository>的话就自己添加就是的了）

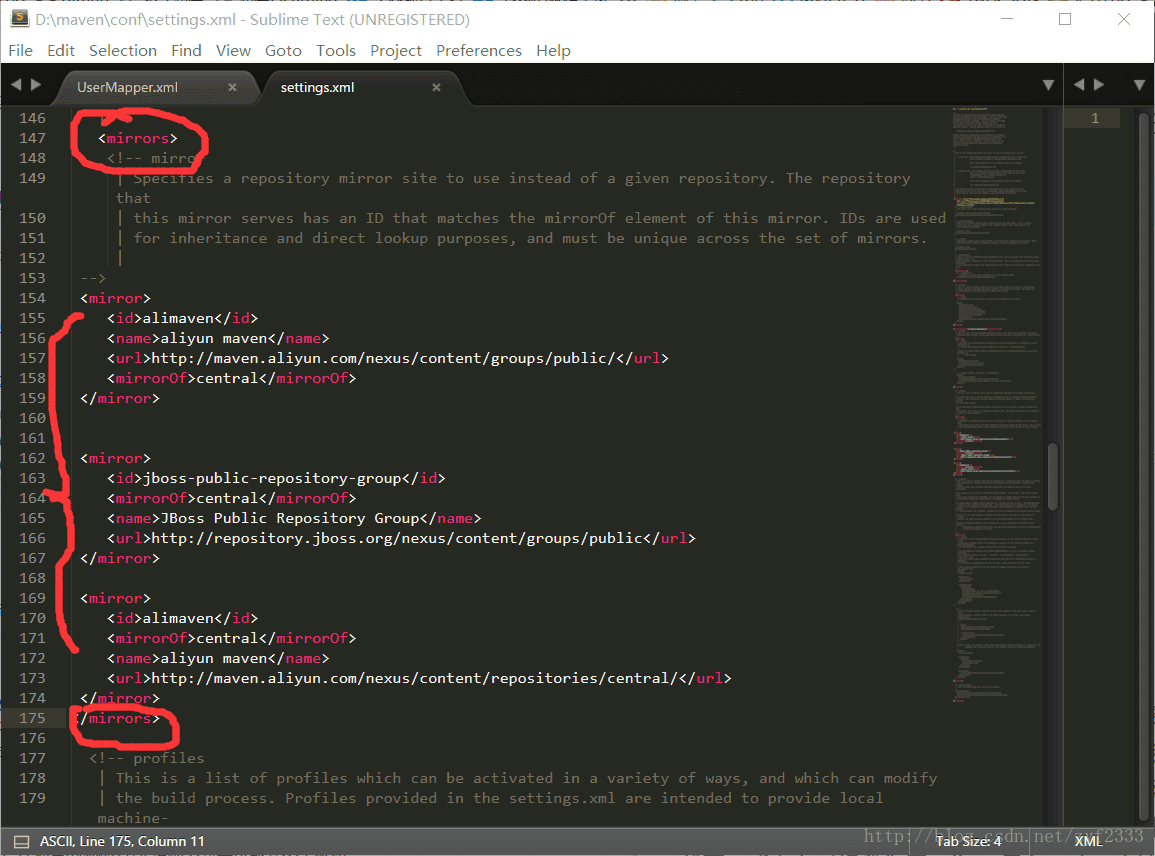


接下来在大约150行左右，在<mirrors> 元素之间添加如下代码

<mirror>  
 <id>alimaven</id>  
 <name>aliyun maven</name>  
 <url>http://maven.aliyun.com/nexus/content/groups/public/</url>  
 <mirrorOf>central</mirrorOf>   
 </mirror>  
 <mirror>  
 <id>jboss-public-repository-group</id>  
 <mirrorOf>central</mirrorOf>  
 <name>JBoss Public Repository Group</name>  
 <url>http://repository.jboss.org/nexus/content/groups/public</url>  
 </mirror>  
 <mirror>  
 <id>alimaven</id>  
 <mirrorOf>central</mirrorOf>  
 <name>aliyun maven</name>  
 <url>http://maven.aliyun.com/nexus/content/repositories/central/</url>  
 </mirror>

1

最后效果如下



1.3.1 maven是干嘛的？

答：百度上也有，说是管理jar包的。通俗点就是：如果你以前有过使用eclipse的经验，就会知道，那么当我们需要引用外来jar包的时候，都会自己找jar包，然后下载并添加到项目运行环境中。而使用maven，则可以自动下载，不用我们自己去下载了。或者有时候我们用eclipse导入spring支持的包时候，也需要自己添加所需要的jar包，运用maven就可以自动添加了。

1.3.2 创建repository文件夹干嘛？

答：这个文件夹是用来存放maven自动下载的jar包的。我们第一次使用的时候，会进行下载，这样会耗一点时间，以后我们再用就直接用这个文件夹下的jar包了。

1.3.3 为什么要修改setting.xml中的<localRepository>

答：我们上面说了，maven会下载jar包并存放到自己创好的repository文件夹下。这个操作就是设置让maven把下载下来的jar包放入repository文件夹

1.3.4 为什么要修改setting.xml中的<mirror> ？

答：我们说了，maven会自动下载。那么它从哪个网站下载呢？此处就要用到这个<mirror> 了。由于maven默认的下载地址是国外的网站，可能由于墙的原因下载会超级超级慢。因此我们自动添加几个国内的网站，此处就用了阿里云。可以多添加几个网站，因为有些网站可能jar包不是很全，因此多添加几个以防万一

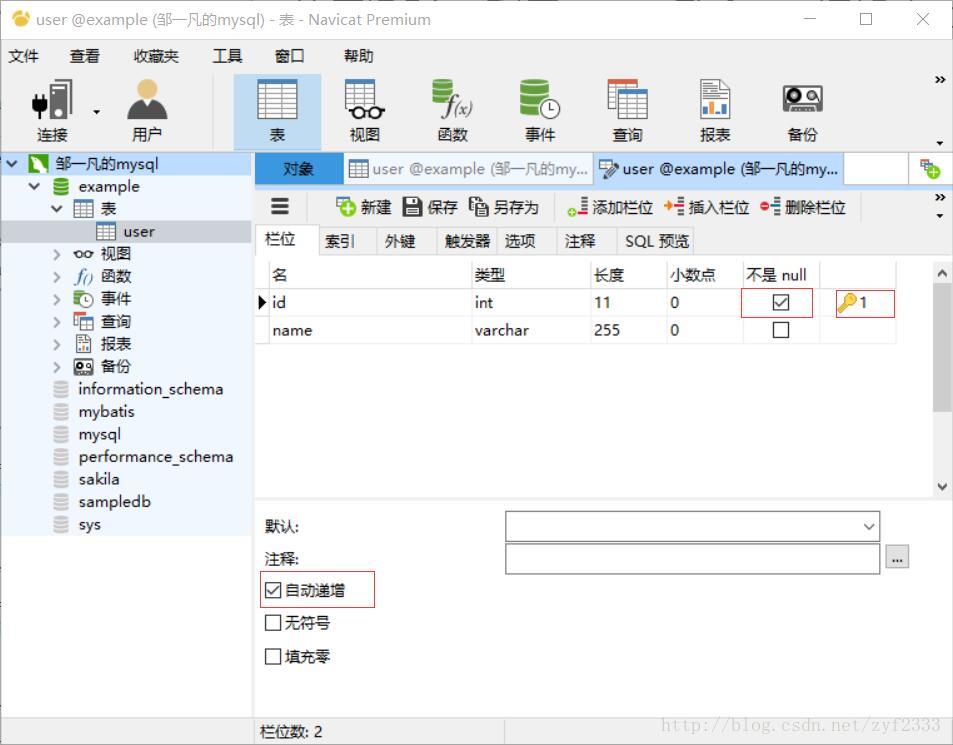
1.3.5 要是添加了好几个网站后也没有我想要的jar包呢？

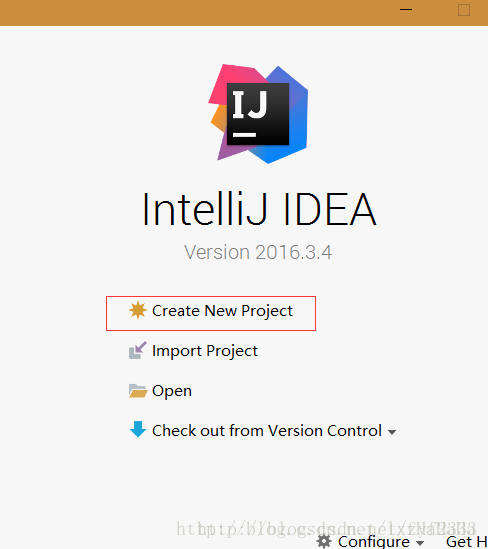
答：要么再加几个网站，要么自己手动添加（方法后面讲）

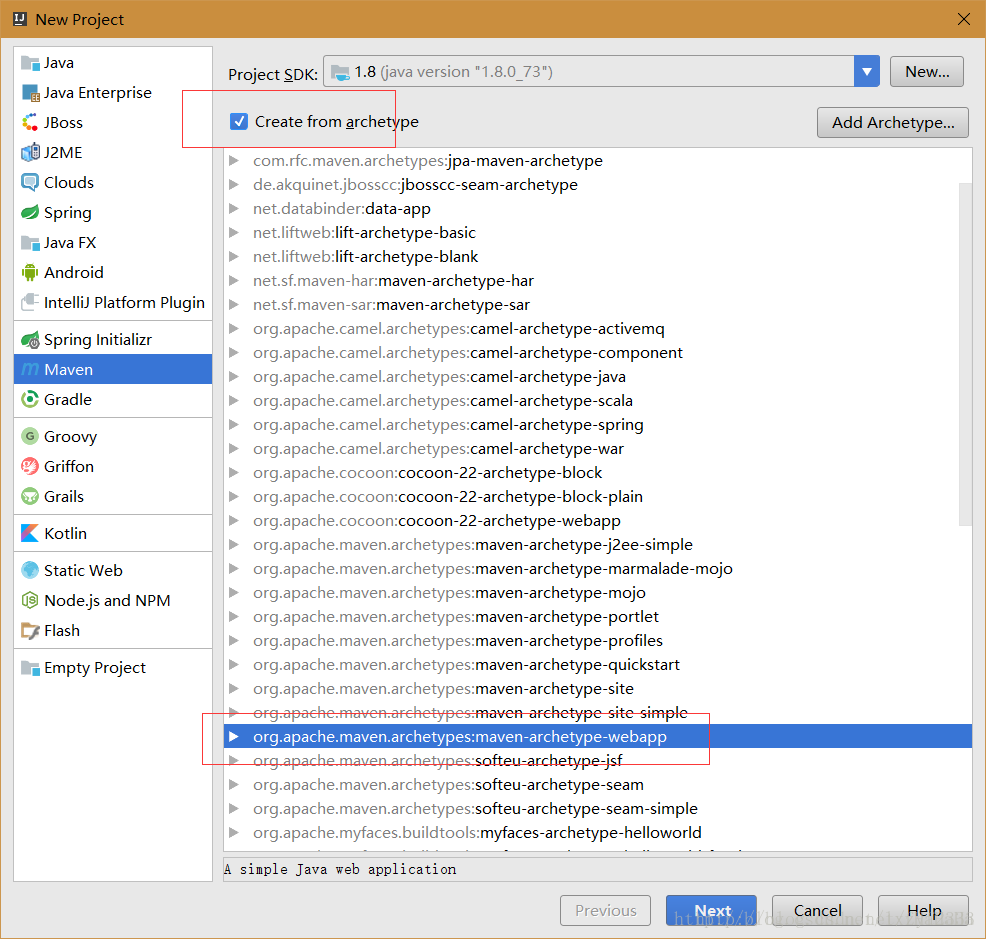
1.4 自行安装IDEA，这里不多说了

**2）创建maven项目**

在创建之前，请自行在数据库中创建名为”example”的数据库，里面有个user表。如下：

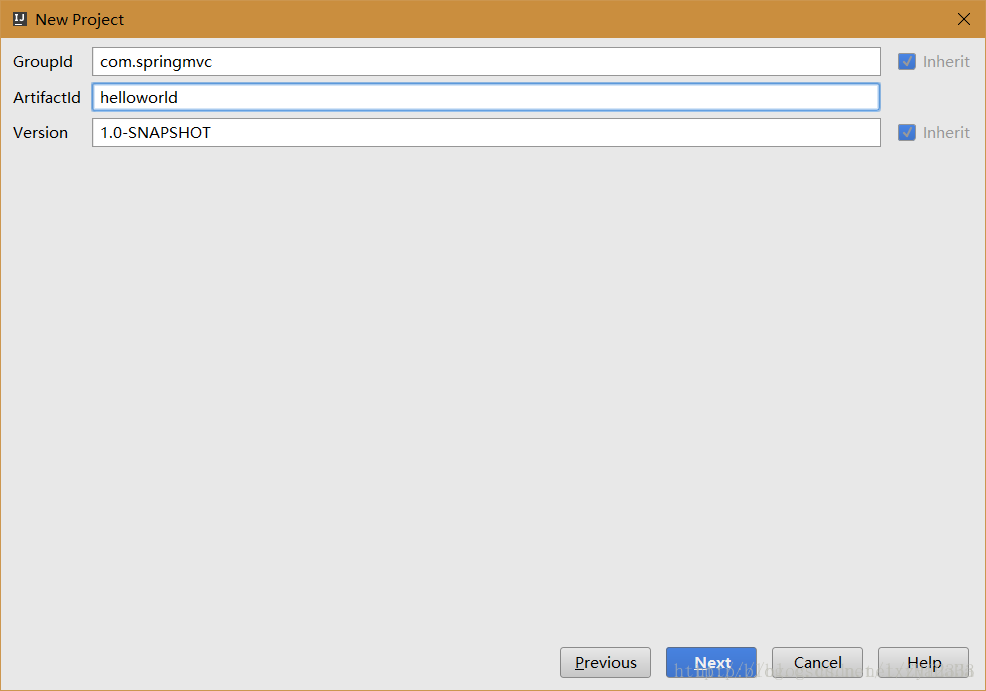




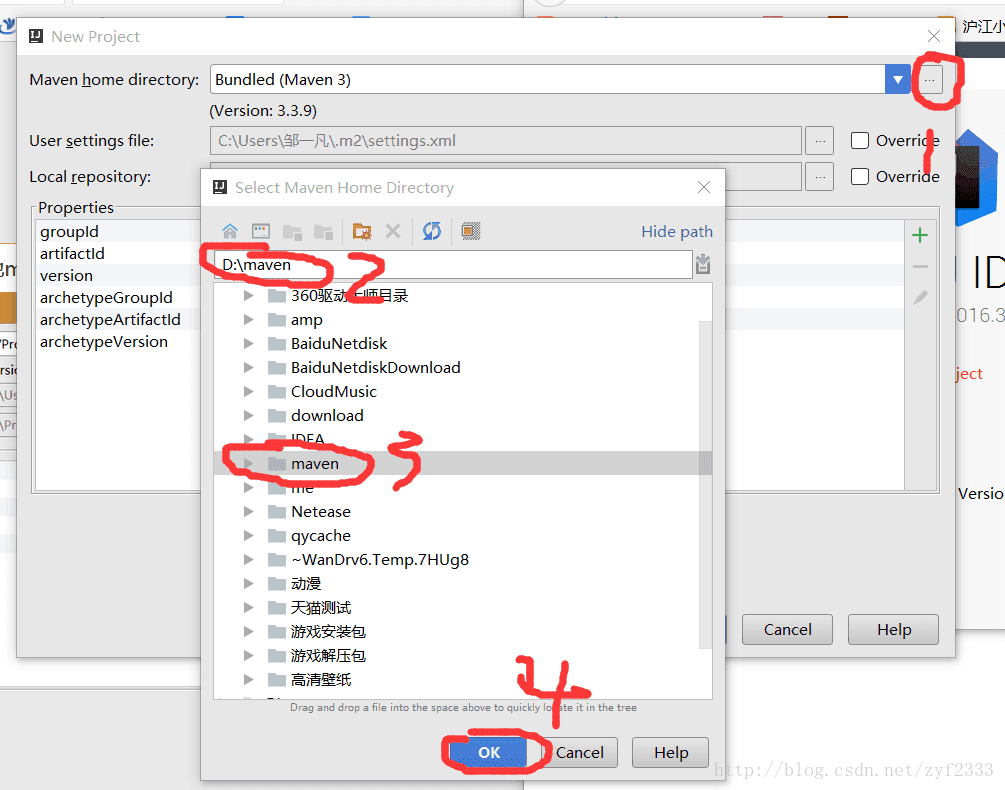


点击next，填写对应的GroupId,ArtifactId（GroupId随意填，我也不太知道有什么用。ArtifactId就是你项目名）

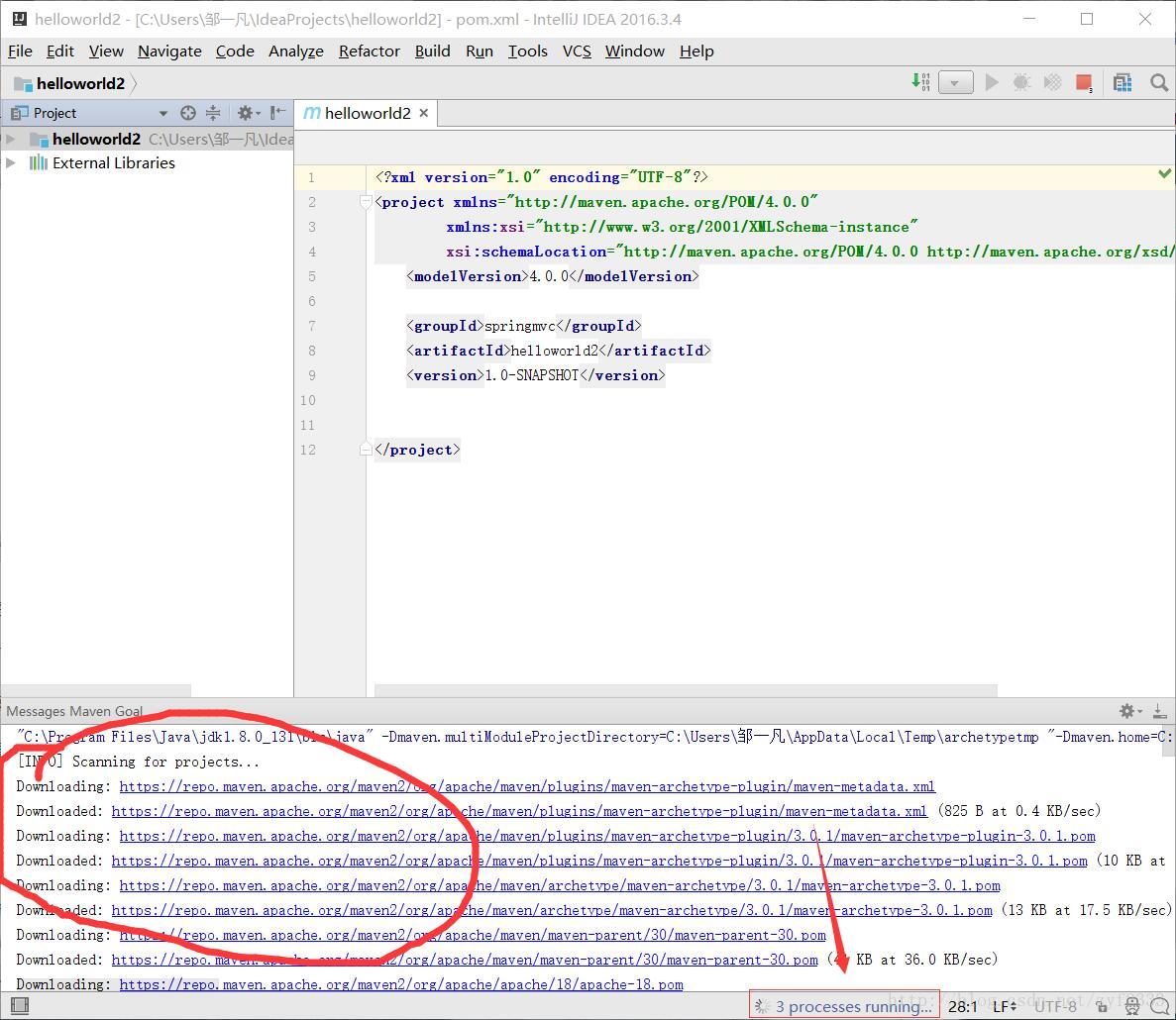
（PS：接下来不要纠结我的项目名称，因为我是同时开一个helloworld2，然后又用到了我已经架构好了的helloworld项目）



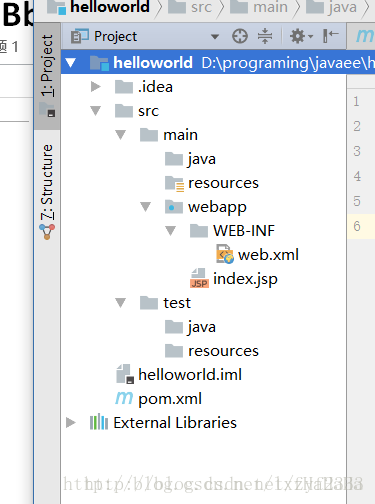
点击next，此处就用到maven了。



然后next，最后finish，然后慢慢等。刚刚开始创建项目的时候，IDEA要花点时间来创建项目，等一下就好了(ps:有点慢的，慢慢等)



创建完了后，原生态目录应该如下（目录说明：.idea和helloworld.iml不用管，也不要删，是IDEA本身的配置文件；src就是我们写项目的地方了。**pom.xml是用来配合maven的。我们要用什么jar包，写在pom.xml中，d盘中的maven就会自动从<mirror> 中下载到repository文件夹中**）



接着，执行如下操作：

1. 在main的java文件夹上，鼠标右键，找到 make directory as，选择Sources root

2. 在main的resources文件夹上，鼠标右键，找到make directory as，选择Resources root

3. 在test的java文件夹上，鼠标右键，找到make directory as，选择Test Sources root

4. 在test的resources文件夹上，鼠标右键，找到make directory as，选择Test Resources root

继续修改目录结构，增加一些文件夹及文件，这些都是等一下要用的到的

1. src->main->java下新建package com，再新建package springmvc，这两个对应我们刚开始的groupid

2. springmvc包下新建四个包，controller包用来放前台url映射文件，dao用来放与数据库的接口文件，entity用来放实体类文件，service用来放自己定义的接口

3. applicationContext.xml是spring配置文件

4. sqlmap文件夹里面放实体与数据库的映射文件

5. generator.xml和generator.properties是mybatis的配置文件

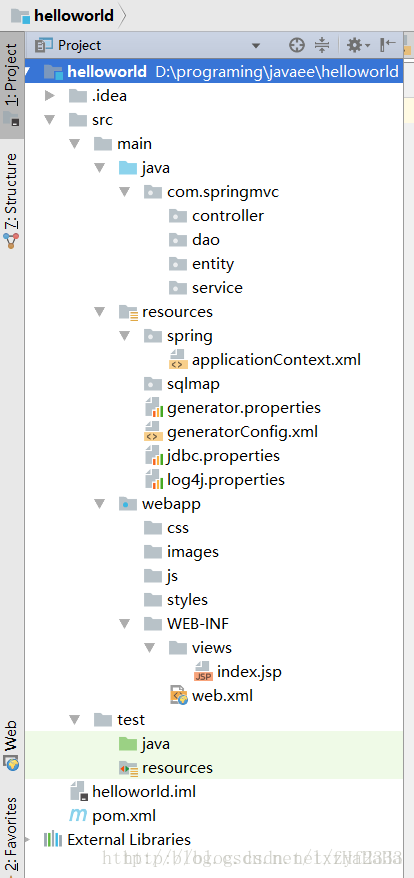
6. jdbc.properties是数据库相关信息

7. log4j.properties是日志配置文件

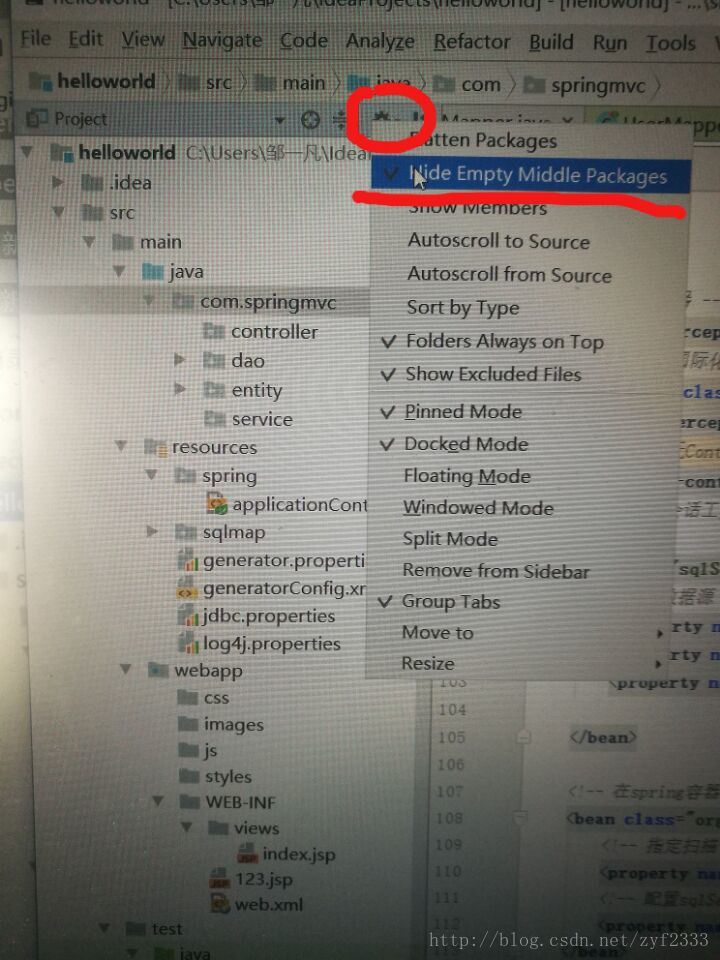
8. webapp下新建css、images等文件，用来放前端资源

9. web-inf下新建views文件件，jsp文件就放这里面了

修改后目录如下：



（**PS：此处有些人项目可能不会像我这样是“树枝状”，而是变成了”com.springmvc.controller.dao.entity.service”。即包都挤在一起去了。解决方案：如图，把划线部分勾掉就好了**）



**3）项目配置文件**

3.1、配置pom.xml文件，引入所需要的jar包

之前说了，pom.xml是用来配合maven使用的，我们想要什么jar包，直接在pom.xml中写出来，maven就会自动帮我们下载。当你刚刚开始用maven时候，repository文件夹里面还是空的，因此要花时间来下载，请直接把接下来的代码复制黏贴。在这里，我们只需要关注<dependency>标签就好了

1

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 <http://maven.apache.org/maven-v4_0_0.xsd>">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.springtest</groupId>  
 <artifactId>helloworld</artifactId>  
 <packaging>war</packaging>  
 <version>1.0-SNAPSHOT</version>  
 <name>helloworld Maven Webapp</name>  
 <url>http://maven.apache.org</url>  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context-support</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-test</artifactId>  
 <version>3.1.1.RELEASE</version>  
 </dependency>  
 <!-- springframe end -->

<!-- mybatis核心包 -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.3.1</version>  
 </dependency>  
 <!-- mybatis/spring包 -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>1.2.2</version>  
 </dependency>  
 <!-- mysql驱动包 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.29</version>  
 </dependency>  
 <!-- junit测试包 -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
 </dependency>  
 <!-- jstl -->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.servlet.jsp</groupId>  
 <artifactId>jsp-api</artifactId>  
 <version>2.2.1-b03</version>  
 <scope>provided</scope>  
 </dependency>

<!-- servlet -->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 <version>2.5</version>  
 <scope>provided</scope>  
 </dependency>  
 <!-- json数据 -->  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-core</artifactId>  
 <version>2.8.5</version>  
 </dependency>  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>2.8.5</version>  
 </dependency>  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-annotations</artifactId>  
 <version>2.8.5</version>  
 </dependency>  
 <dependency>  
 <groupId>org.codehaus.jackson</groupId>  
 <artifactId>jackson-mapper-asl</artifactId>  
 <version>1.9.4</version>  
 </dependency>  
 <dependency>  
 <groupId>net.sf.json-lib</groupId>  
 <artifactId>json-lib</artifactId>  
 <version>2.4</version>  
 <classifier>jdk15</classifier>  
 </dependency>  
 <!-- commons -->  
 <dependency>  
 <groupId>commons-lang</groupId>  
 <artifactId>commons-lang</artifactId>  
 <version>2.4</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-logging</groupId>  
 <artifactId>commons-logging</artifactId>  
 <version>1.1</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-pool</groupId>  
 <artifactId>commons-pool</artifactId>  
 <version>1.5.6</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-dbcp</groupId>  
 <artifactId>commons-dbcp</artifactId>  
 <version>1.4</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-beanutils</groupId>  
 <artifactId>commons-beanutils</artifactId>  
 <version>1.8.3</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-httpclient</groupId>  
 <artifactId>commons-httpclient</artifactId>  
 <version>3.1</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-collections</groupId>  
 <artifactId>commons-collections</artifactId>  
 <version>3.1</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-codec</groupId>  
 <artifactId>commons-codec</artifactId>  
 <version>1.9</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.annotation</groupId>  
 <artifactId>jsr250-api</artifactId>  
 <version>1.0</version>  
 </dependency>

<dependency>  
 <groupId>net.sf.ezmorph</groupId>  
 <artifactId>ezmorph</artifactId>  
 <version>1.0.6</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.activation</groupId>  
 <artifactId>activation</artifactId>  
 <version>1.1</version>  
 </dependency>  
 <dependency>  
 <groupId>taglibs</groupId>  
 <artifactId>standard</artifactId>  
 <version>1.1.2</version>  
 </dependency>  
 <!-- 日志文件管理包 -->  
 <!-- log start -->  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.5</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-log4j12</artifactId>  
 <version>1.7.21</version>  
 </dependency>  
 </dependencies>  
 <build>  
 <finalName>helloworld</finalName>  
 <plugins>  
 <plugin>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-maven-plugin</artifactId>  
 <version>1.3.2</version>  
 <configuration>  
 <!--允许移动生成的文件-->  
 <verbose>true</verbose>  
 <!--允许覆盖生成的文件-->  
 <overwrite>true</overwrite>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

138

235

3.2、jdbc.properties

driver=com.mysql.jdbc.Driver  
url=jdbc:mysql://localhost:3306/example?useUnicode=true&characterEncoding=utf-8  
username=root  
#下面输入自己数据库的密码  
password=输入自己数据库的密码吧  
#定义初始连接数  
initialSize=0  
#定义最大连接数  
maxActive=20  
#定义最大空闲  
maxIdle=20  
#定义最小空闲  
minIdle=1  
#定义最长等待时间  
maxWait=60000

3.3、applicationContext.xml（配置spring）

<?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:context="http://www.springframework.org/schema/context"  
 xmlns:beans="http://www.springframework.org/schema/beans"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 <http://www.springframework.org/schema/beans/spring-beans-4.3.xsd>  
 <http://www.springframework.org/schema/context>  
 <http://www.springframework.org/schema/context/spring-context-4.3.xsd>  
 <http://www.springframework.org/schema/mvc>  
 <http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd>" >

<!-- ①：对com.springmvc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.springmvc"/>  
 <mvc:annotation-driven />  
 <!-- 静态资源访问 -->  
 <!--如果webapp下你新建了文件夹，想访问里面的静态资源，那么就要在这配置一下-->  
 <mvc:resources location="/images/" mapping="/images/\*\*"/>  
 <mvc:resources location="/css/" mapping="/css/\*\*"/>  
 <mvc:resources location="/styles/" mapping="/styles/\*\*"/>  
 <mvc:resources location="/js/" mapping="/js/\*\*"/>

<!-- Configures the @Controller programming model  
 <mvc:annotation-driven />-->  
 <!-- ②：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">  
 <property name="messageConverters">  
 <list>  
 <ref bean="mappingJacksonHttpMessageConverter"/>  
 </list>  
 </property>  
 </bean>

<bean id="mappingJacksonHttpMessageConverter" class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter">  
 <property name="supportedMediaTypes">  
 <list>  
 <value>application/json;charset=UTF-8</value>  
 </list>  
 </property>  
 </bean>

<!-- 配置视图解析器，把控制器的逻辑视频映射为真正的视图 -->  
 <!-- /WEB-INF/jsp/start.jsp -->  
 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>  
 <property name="prefix" value="/WEB-INF/views/" />  
 <property name="suffix" value=".jsp" />  
 </bean>

<!-- 配置dbcp数据库连接池 -->

<!-- <context:property-placeholder location="classpath:db.properties"/> -->  
 <!--数据库配置 -->  
 <bean id = "propertyConfigurer" class = "org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  
 <property name="locations">  
 <list>  
 <value>classpath:jdbc.properties</value>  
 </list>  
 </property>

</bean>

<!-- 数据库连接池 -->  
 <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">  
 <property name="driverClassName" value="${driver}"/>  
 <property name="url" value="${url}"/>  
 <property name="username" value="${username}"/>  
 <property name="password" value="${password}"/>  
 <property name="initialSize" value="1"/>  
 <property name="maxActive" value="100"/>  
 <property name="maxIdle" value="5"/>  
 <property name="maxWait" value="80000"/>  
 </bean>

<!-- 配置事务管理器 -->  
 <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
 <property name="dataSource" ref="dataSource" />  
 </bean>

<!-- 拦截器 -->  
 <mvc:interceptors>  
 <!-- 国际化操作拦截器 如果采用基于（请求/Session/Cookie）则必需配置 -->  
 <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />  
 </mvc:interceptors>  
 <!-- 定义无Controller的path<->view直接映射 -->  
 <mvc:view-controller path="/" view-name="redirect:/" />  
 <!-- 配置会话工厂SqlSessionFactory -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <!-- 数据源 -->  
 <property name="dataSource" ref="dataSource"/>  
 <property name="mapperLocations" value="classpath:sqlmap/\*Mapper.xml"/>  
 <property name="typeAliasesPackage" value="com.springmvc.entity" />

</bean>

<!-- 在spring容器中配置mapper的扫描器产生的动态代理对象在spring的容器中自动注册，bean的id就是mapper类名（首字母小写）-->  
 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <!-- 指定扫描包的路径，就是mapper接口的路径，多个包中间以 半角逗号隔开 -->  
 <property name="basePackage" value="com.springmvc.dao"/>  
 <!-- 配置sqlSessionFactoryBeanName -->  
 <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>  
 </bean>

</beans>

3.4、log4j.properties

### set log levels ###  
log4j.rootLogger = INFO,D

###输出到控制台###  
log4j.logger.toConsole=debug,stdout  
log4j.appender.stdout = org.apache.log4j.ConsoleAppender  
log4j.appender.stdout.Target = System.out  
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern =%d{yyyy-MM-dd HH\:mm\:ss} [%5p] - %c -%F(%L) -%m%n

### 输出到日志文件 ###  
## 其中File 可以采用绝对路径 亦可以采用相对路径 catalina.home 是tomcat目录 如果没有logs文件夹 将会报错 ##  
## 更加倾向于将日志文件放在 tomcat的日志目录下${catalina.home}/logs/salesmanb2bError.log ##  
## 绝对路径如：e:/salesmanb2bError.log ##  
log4j.logger.daily=INFO,D  
log4j.appender.D = org.apache.log4j.DailyRollingFileAppender  
log4j.appender.D.File = ${catalina.home}/logs/helloworld/helloworld.log  
log4j.appender.D.Append = true  
log4j.appender.D.Threshold = INFO  
log4j.appender.D.layout = org.apache.log4j.PatternLayout  
log4j.appender.D.layout.ConversionPattern =%d{yyyy-MM-dd HH\:mm\:ss} [%5p] - %c -%F(%L) -%m%n

##log4j.logger.org.apache.ibatis=debug,stdout  
##log4j.logger.java.sql=debug,stdout

3.5、web.xml（在WEB-INF下）

<?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" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee <http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd>" version="3.0">  
 <display-name>helloworld</display-name>  
 <!-- 配置编码方式-->  
 <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>  
 <init-param>  
 <param-name>forceEncoding</param-name>  
 <param-value>true</param-value>  
 </init-param>  
 </filter>  
 <filter-mapping>  
 <filter-name>encodingFilter</filter-name>  
 <url-pattern>/\*</url-pattern>  
 </filter-mapping>

<!-- 配置springmvc的前端控制器 指向spring-mvc.xml 程序在启动的时候就加载springmvc 可以接受所有请求 load-on-startup：表示启动容器时初始化该Servlet； -->  
 <servlet>  
 <servlet-name>springServlet</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <!-- 可以自定义servlet.xml配置文件的位置和名称， 默认为WEB-INF目录下，名称为[<servlet-name>]-servlet.xml，如spring-servlet.xml -->  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value> classpath:spring/applicationContext.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
 </servlet>  
 <!-- 将前端URL请求和后台处理方法controller建立对应关系-->  
 <servlet-mapping>  
 <servlet-name>springServlet</servlet-name>  
 <url-pattern>/</url-pattern>  
 </servlet-mapping>

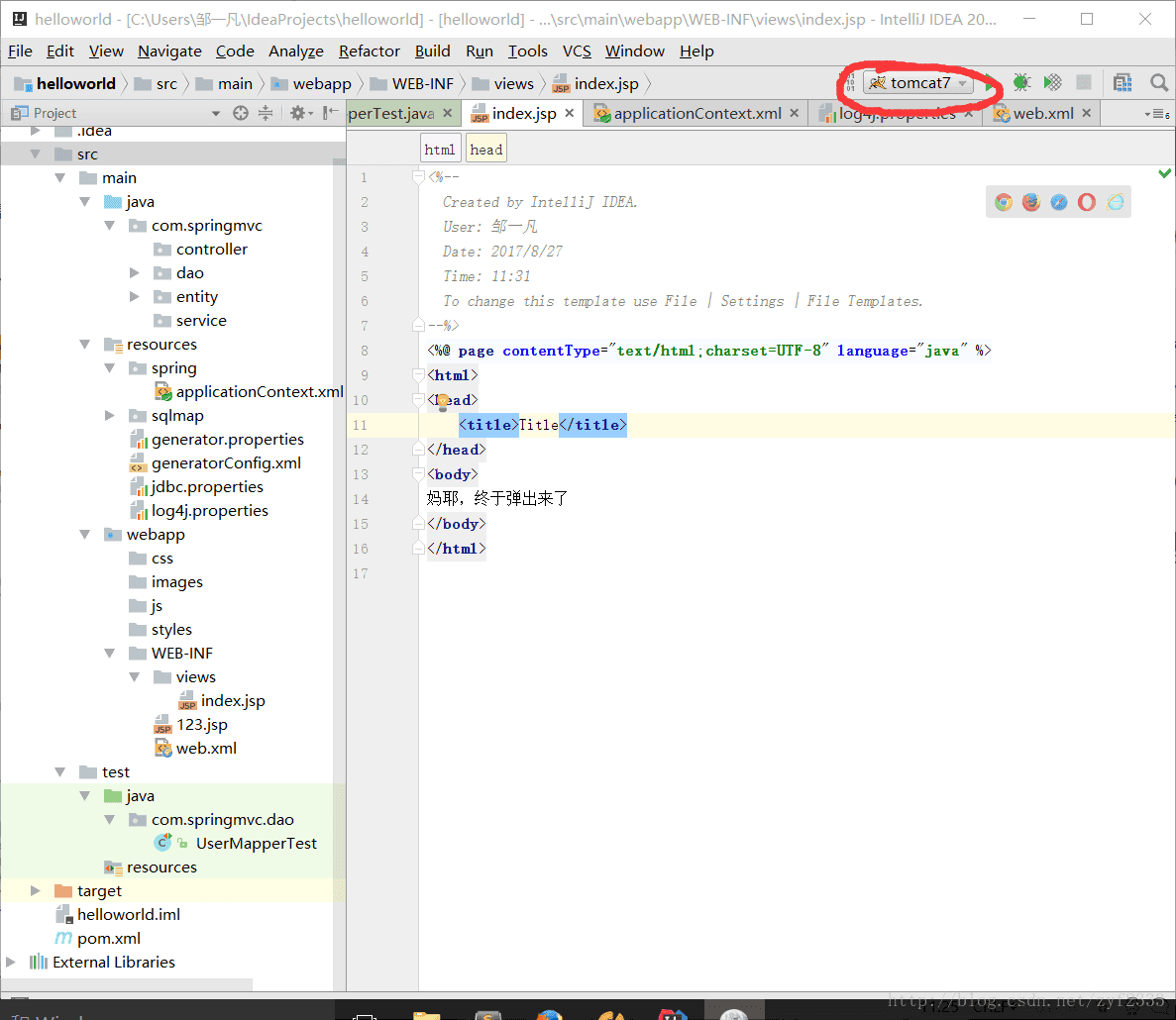
<!-- Spring配置 -->  
 <listener>  
 <listener-class>  
 org.springframework.web.context.ContextLoaderListener  
 </listener-class>  
 </listener>  
 <!-- 取消对某一类文件的拦截-->  
 <servlet-mapping>  
 <servlet-name>default</servlet-name>  
 <url-pattern>\*.md</url-pattern>  
 </servlet-mapping>

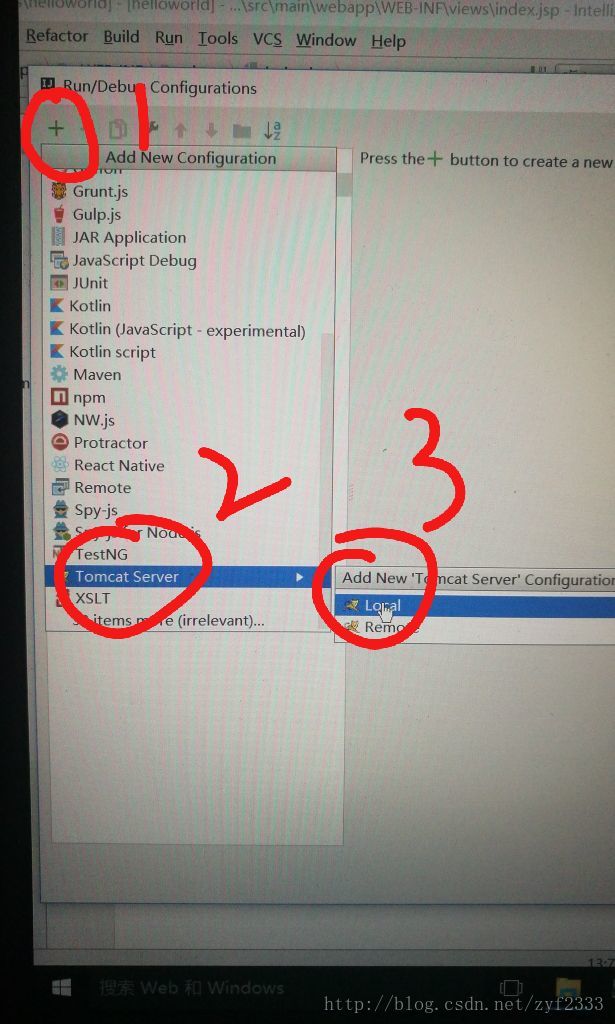
<context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value> classpath:spring/applicationContext.xml</param-value>  
 </context-param>  
 <!-- 欢迎页面-->  
 <welcome-file-list>  
 <welcome-file>/WEB-INF/views/index.jsp</welcome-file>  
 </welcome-file-list>  
 <!--404错误展示页面，可自行配置-->  
 <!--<error-page>-->  
 <!--<error-code>404</error-code>-->  
 <!--<location>/WEB-INF/views/404.jsp</location>-->  
 <!--</error-page>-->  
 <!--设置session失效时间为30分钟 -->  
 <session-config>  
 <session-timeout>600</session-timeout>  
 </session-config>  
</web-app>

3.6、在WEB-INF/views/下创建index.jsp。自己随便写点啥东西

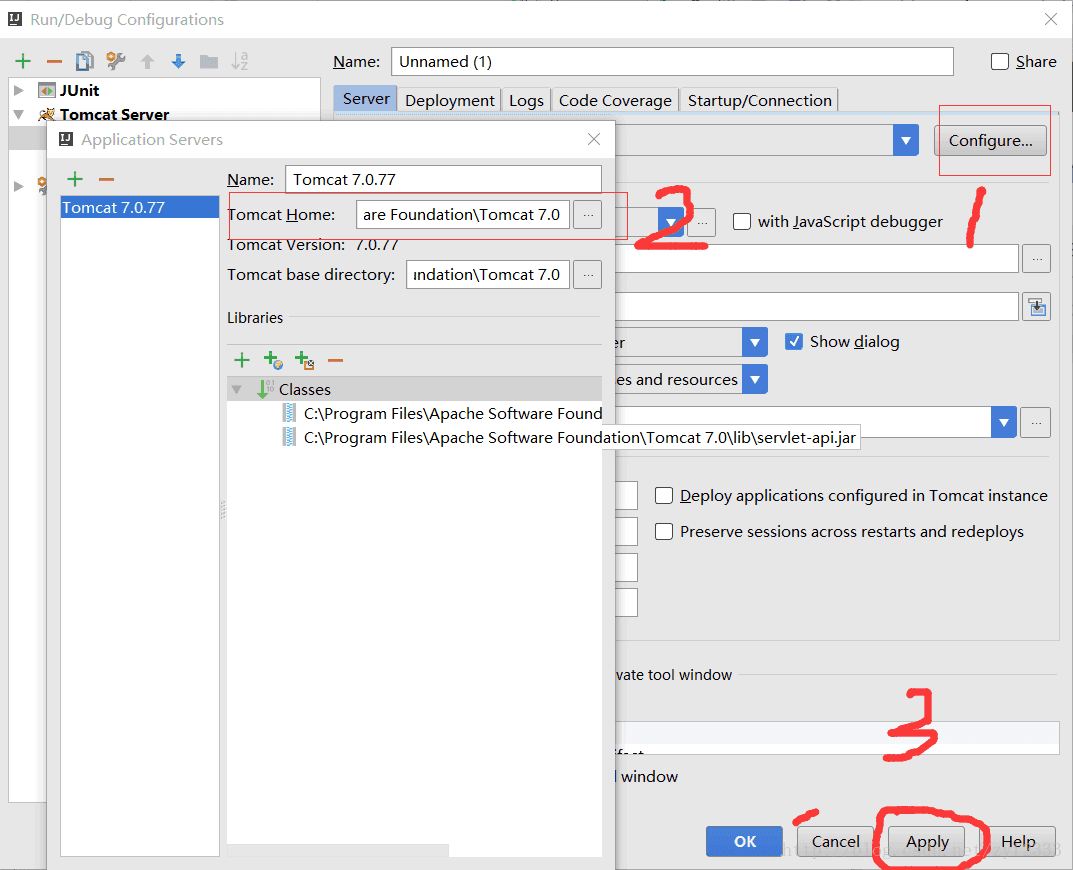
**4）配置Tomcat**

点击如图所示区域，进入“Edit Configurations”

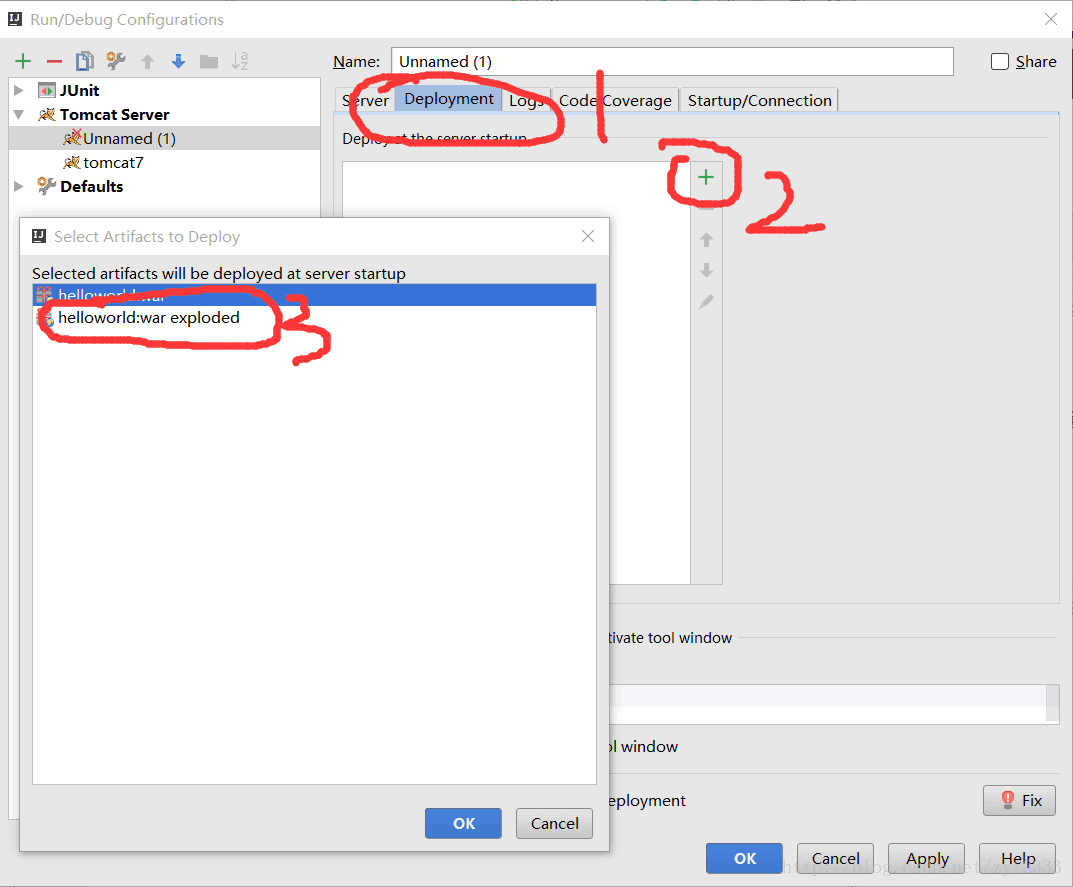




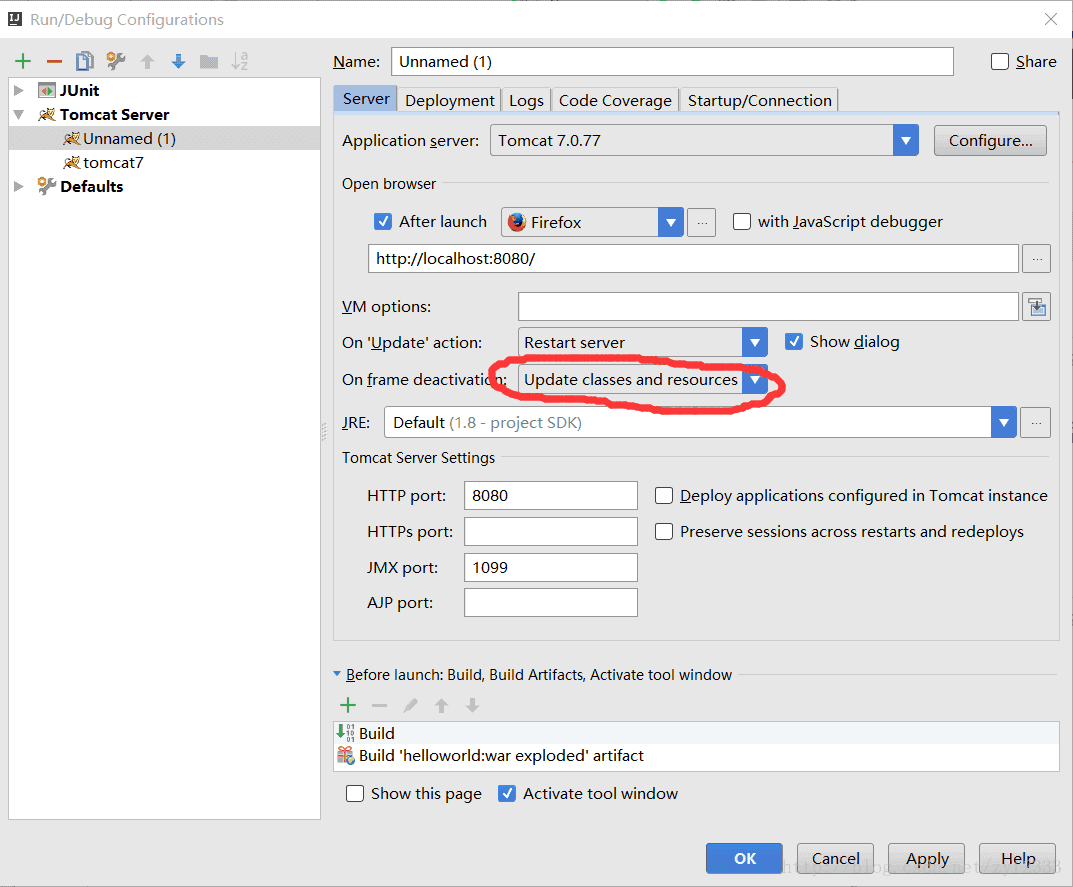
进入设置后，先点击“1”选择设置Tomcat，然后点击“2”选择本地Tomcat所在，最后点击“3”生效



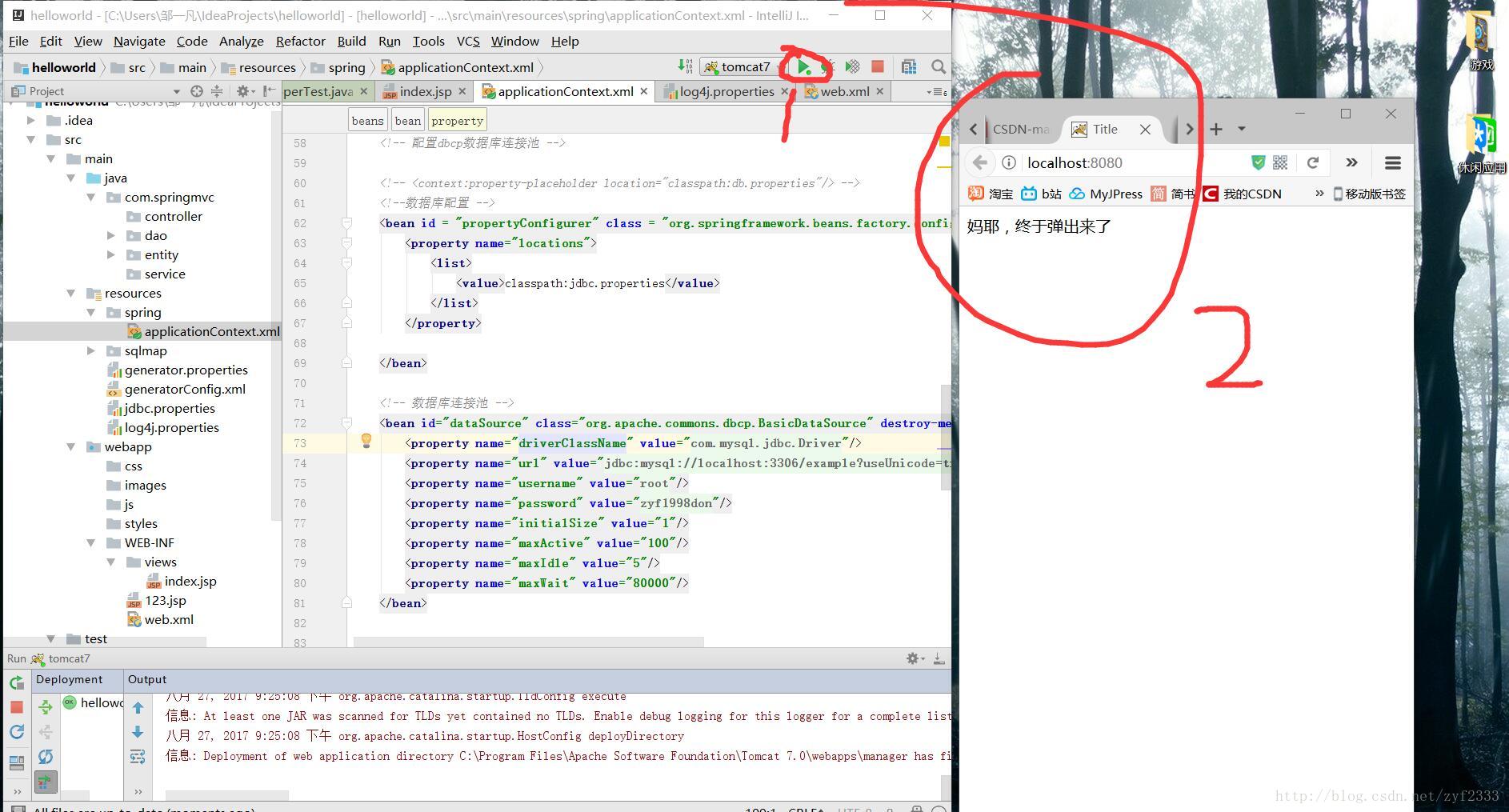
然后将项目部署到Tomcat中，如图（ps：第二步之后选择Artifact选项）



返回来，将如图的”do nothing”改成“update…”



运行



**5）mybatis generator配置**

在applicationContext.xml中添加如下代码

<!-- 配置会话工厂SqlSessionFactory -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <!-- 数据源 -->  
 <property name="dataSource" ref="dataSource"/>  
 <property name="mapperLocations" value="classpath:sqlmap/\*Mapper.xml"/>  
 <property name="typeAliasesPackage" value="com.springmvc.entity" />

</bean>

<!-- 在spring容器中配置mapper的扫描器产生的动态代理对象在spring的容器中自动注册，bean的id就是mapper类名（首字母小写）-->  
 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <!-- 指定扫描包的路径，就是mapper接口的路径，多个包中间以 半角逗号隔开 -->  
 <property name="basePackage" value="com.springmvc.dao"/>  
 <!-- 配置sqlSessionFactoryBeanName -->  
 <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>  
 </bean>

generatorConfig.xml（ps：1、 “<http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd>“该xml中这句话可能红色报错，不用管的。2、第9行的location=”D:\maven\repository\mysql\mysql-connector-java\5.1.29\mysql-connector-java-5.1.29.jar” 是你自己的mysql.jar所在的位置，一定要填写对应的位置，不然就会报错。如果你之前是跟着我的存放目录走的话，应该就是这个路径的）

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE generatorConfiguration PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"  
 "<http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd>">  
<generatorConfiguration>  
 <!--导入属性配置 -->  
 <properties resource="generator.properties"/>

<classPathEntry  
 location="D:\maven\repository\mysql\mysql-connector-java\5.1.29\mysql-connector-java-5.1.29.jar" />  
 <context id="context1">  
 <!-- 注释 -->  
 <commentGenerator>  
 <property name="suppressAllComments" value="true" /><!-- 是否取消注释 -->  
 <property name="suppressDate" value="true" /> <!-- 是否生成注释代时间戳 -->  
 </commentGenerator>

<jdbcConnection driverClass="${driver}"  
 connectionURL="${url}"  
 userId="${username}"  
 password="${password}" />

<!-- 类型转换 -->  
 <javaTypeResolver>  
 <!-- 是否使用bigDecimal， false可自动转化以下类型（Long, Integer, Short, etc.） -->  
 <property name="forceBigDecimals" value="false" />  
 </javaTypeResolver>

<javaModelGenerator targetPackage="${modelPackage}"  
 targetProject="${modelProject}" />  
 <sqlMapGenerator targetPackage="${sqlPackage}" targetProject="${sqlProject}" />  
 <javaClientGenerator targetPackage="${mapperPackage}"  
 targetProject="${mapperProject}" type="XMLMAPPER" />

<!-- 如果需要通配所有表 直接用sql的通配符 %即可 -->  
 <table schema="" tableName="${table}" enableCountByExample="false"  
 enableUpdateByExample="false" enableDeleteByExample="false"  
 enableSelectByExample="false" selectByExampleQueryId="false"/>

</context>  
</generatorConfiguration>

generator.properties

说明：之前已经创建了一个example数据库，里面有个数据表user（第二大步“创建maven项目”下面）。然后modelPackage是等会连接数据库自动生成的实体类保存的位置，sqlPackage是生成xml映射文件保存的位置，mapperPackage是生成数据库接口保存的位置，table是数据库里面的表的名字

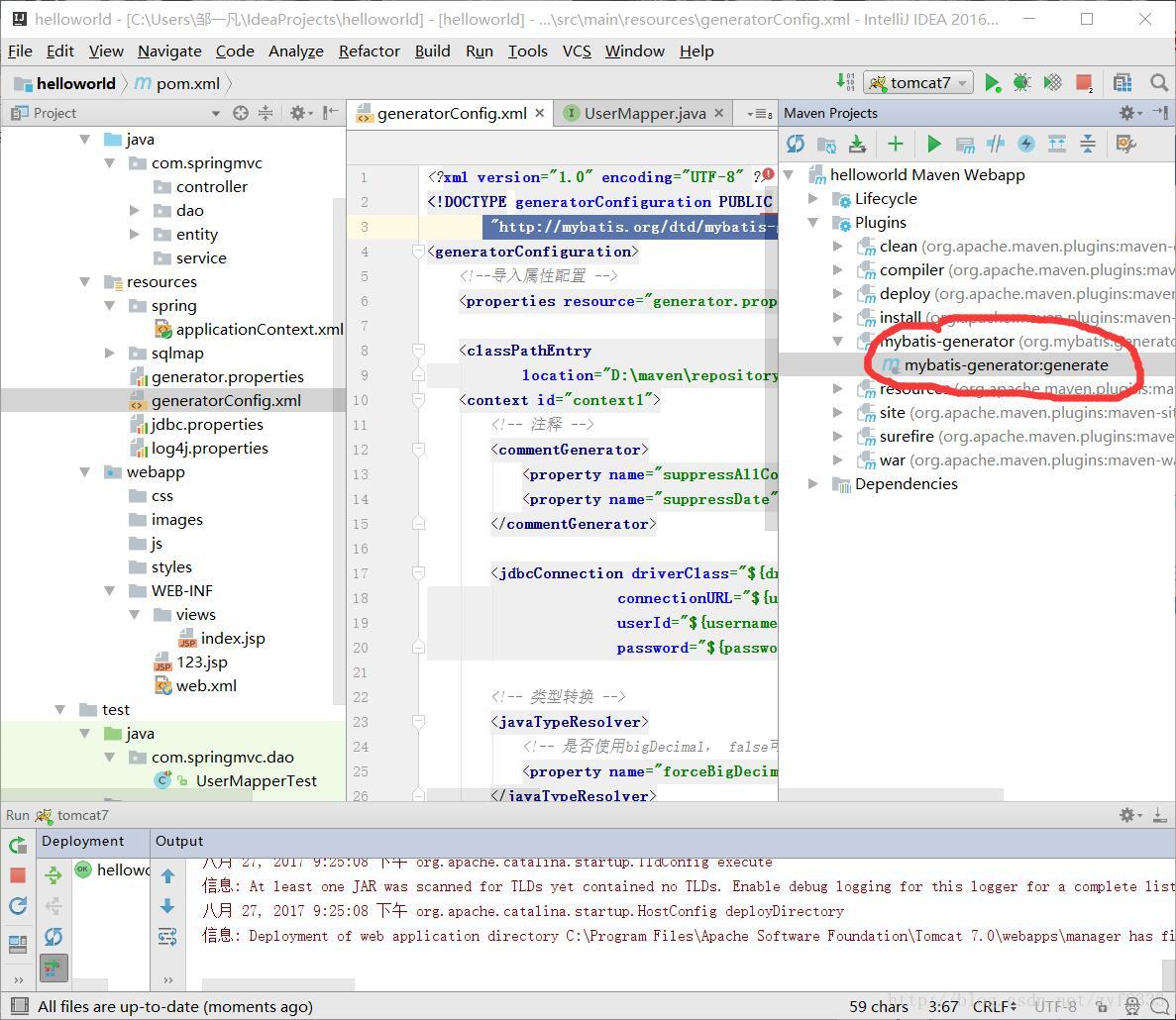
driver=com.mysql.jdbc.Driver  
url=jdbc:mysql://localhost:3306/example?useUnicode=true&characterEncoding=utf-8  
username=root  
password=\*\*\*\*\*\*\*\*\*\*\*\*\*

#entity 包名和 java目录  
modelPackage=com.springmvc.entity  
modelProject=src/main/java  
#sqlmap包名 和resources目录  
sqlPackage=sqlmap  
sqlProject=src/main/resources  
#mapper包名和 java目录  
mapperPackage=com.springmvc.dao  
mapperProject=src/main/java

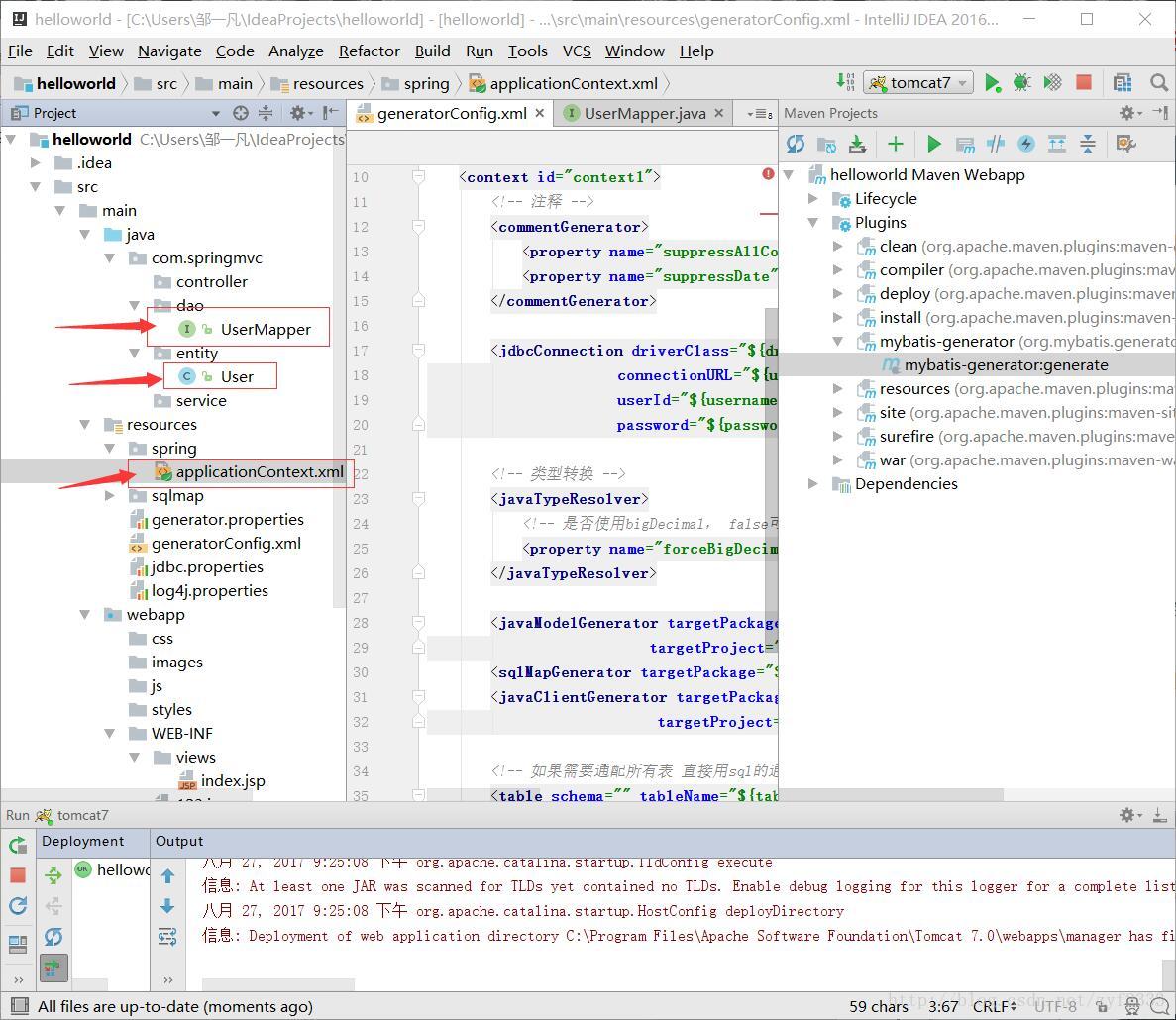
table=user

1

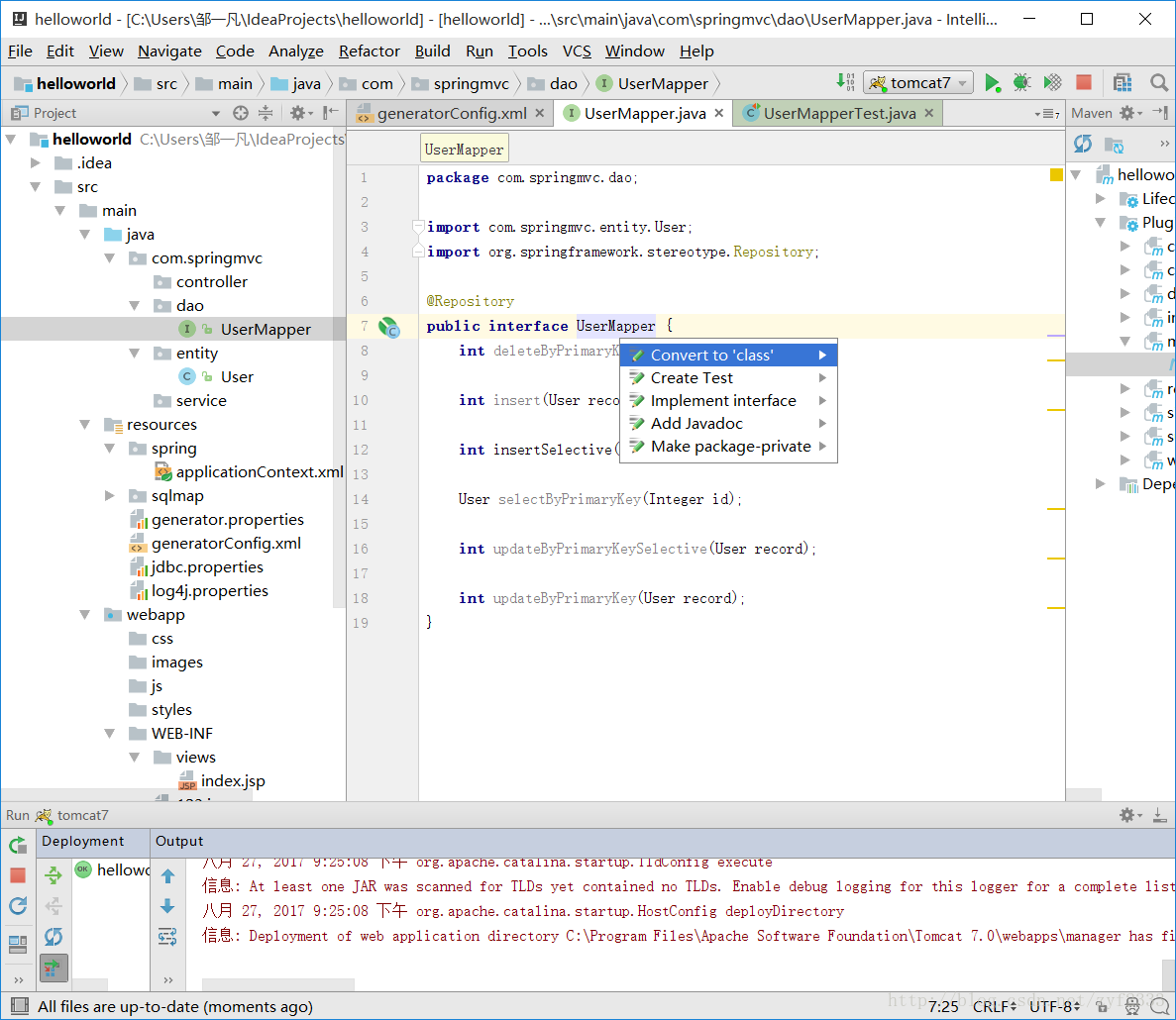
配置完成，双击圈中的地方（ps：要是没有像我一样的maven目录，则按照如下步骤显示：“View->Tool Windows->Maven Project”即可显示出来）



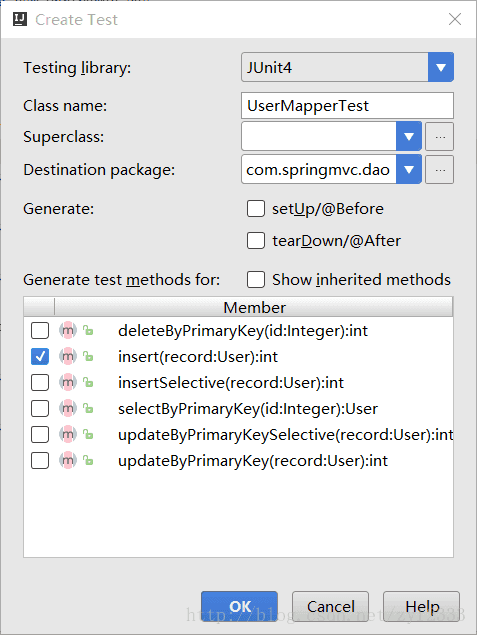
于是会自动创建图中三个文件



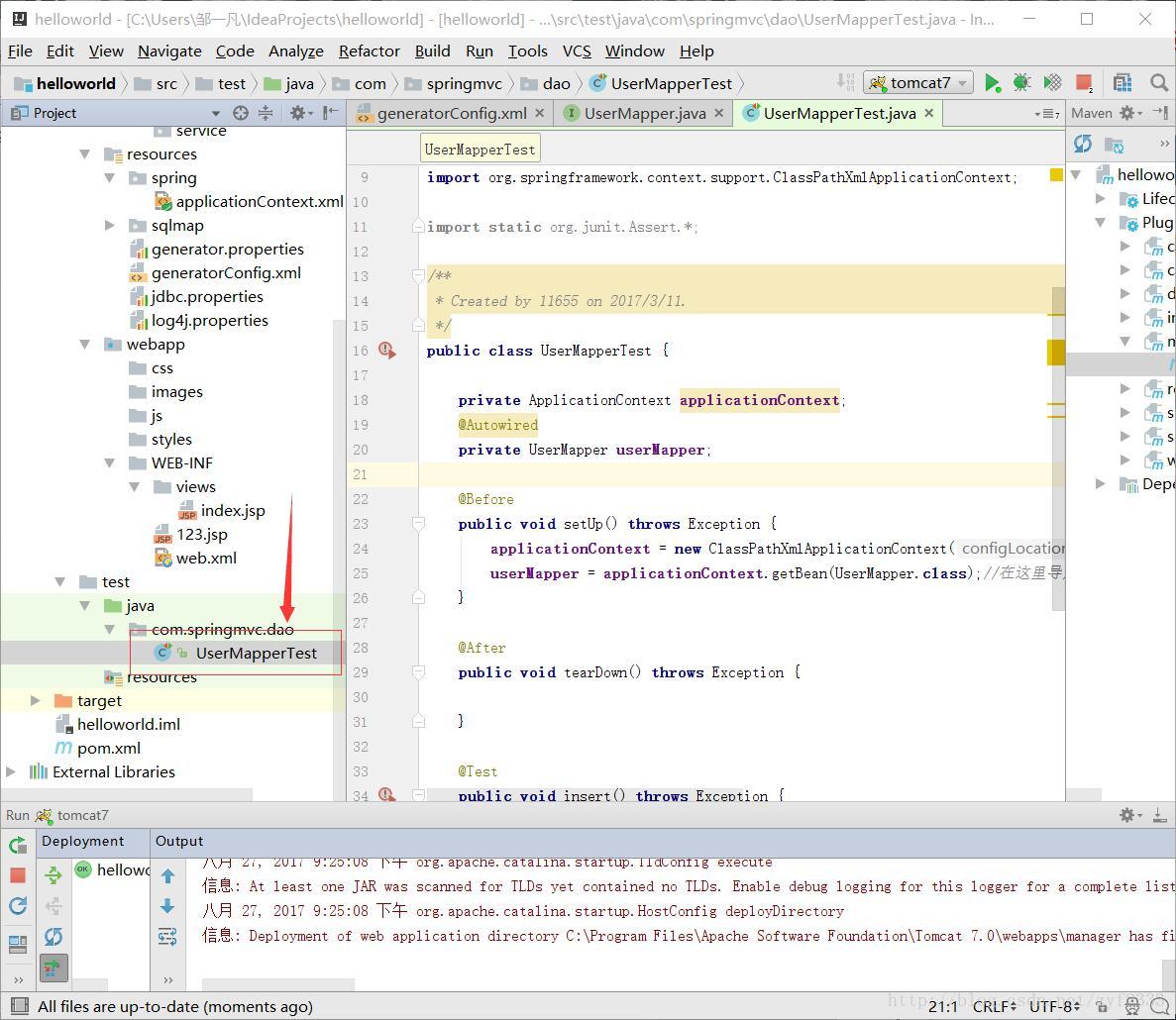
进入UserMapper.java。在接口上方添加@Repository注释，然后将鼠标放到public interface UserMapper处，点击alt+enter键，接着点击create test



勾选insert



于是IDEA会自动帮我们创建个测试类



代码如下

package com.springmvc.dao;

import com.springmvc.entity.User;  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;

import static org.junit.Assert.\*;

/\*\*  
 \* Created by 11655 on 2017/3/11.  
 \*/  
public class UserMapperTest {

private ApplicationContext applicationContext;  
 @Autowired  
 private UserMapper userMapper;

@Before  
 public void setUp() throws Exception {  
 applicationContext = new ClassPathXmlApplicationContext("classpath:spring/applicationContext.xml");//加载spring配置文件  
 userMapper = applicationContext.getBean(UserMapper.class);//在这里导入要测试的  
 }

@After  
 public void tearDown() throws Exception {

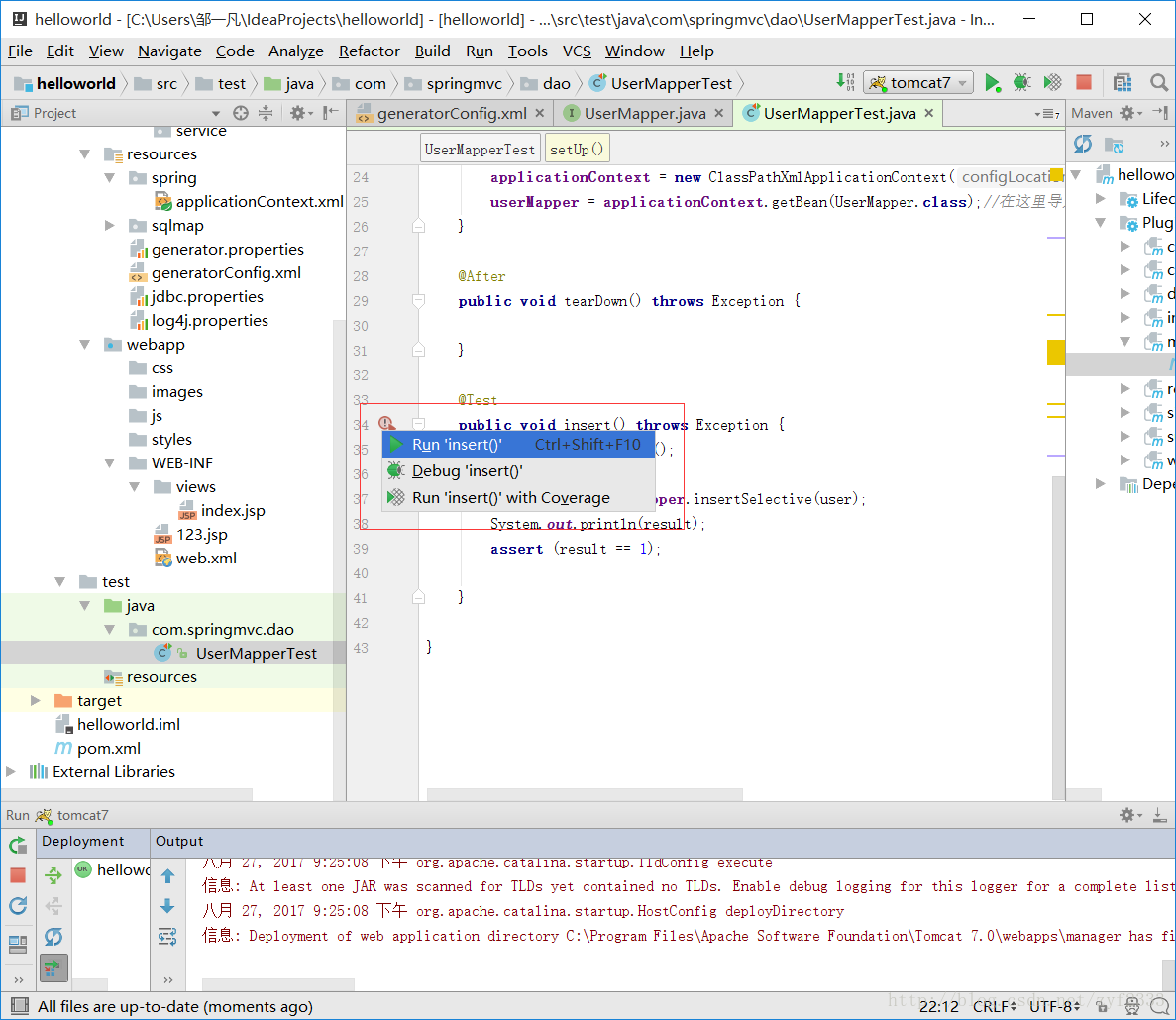
}

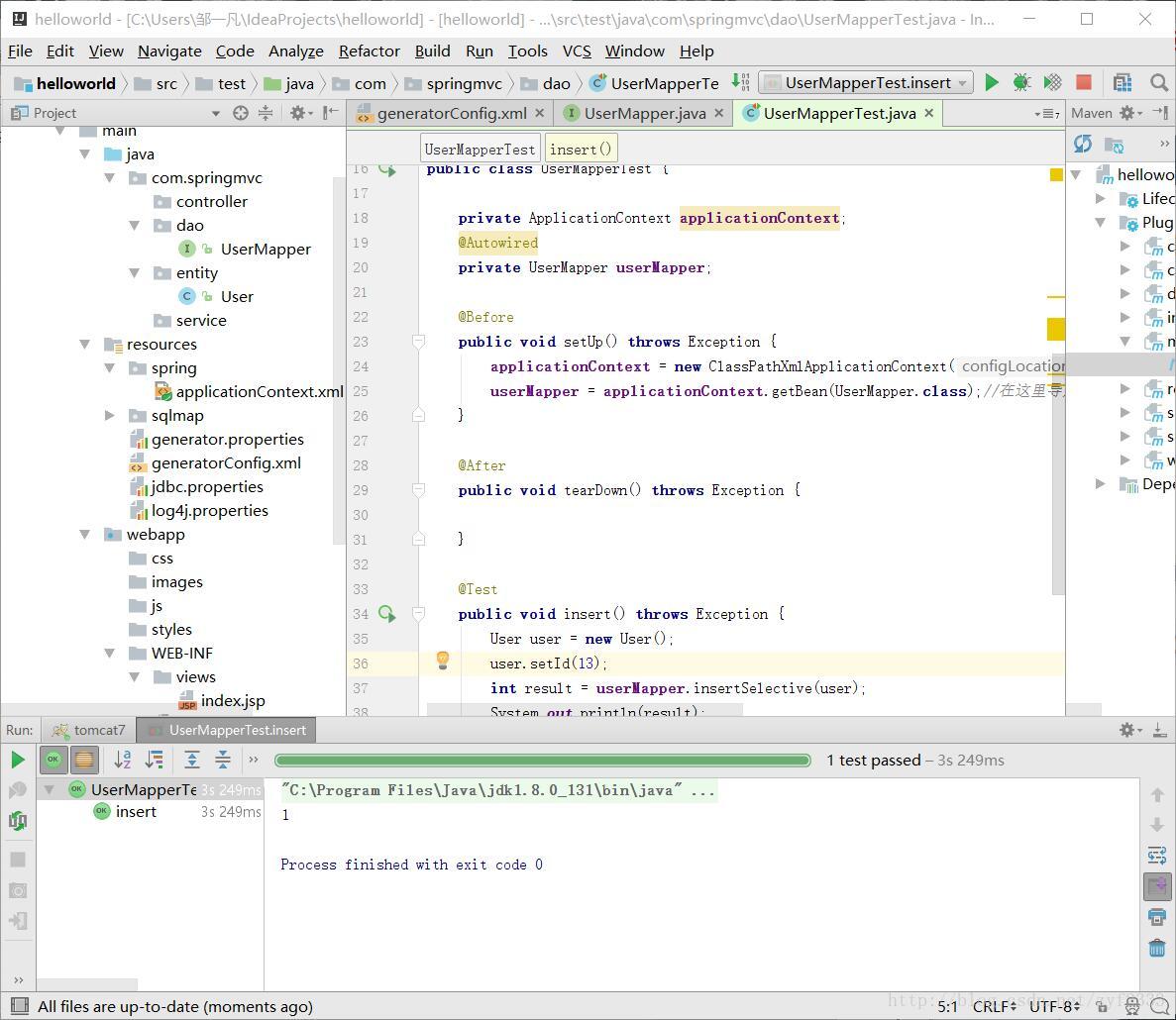
@Test  
 public void insert() throws Exception {  
 User user = new User();  
 user.setId(12);  
 int result = userMapper.insertSelective(user);  
 System.out.println(result);  
 assert (result == 1);

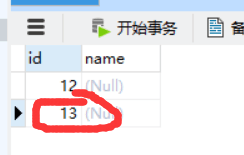
}

}43

然后点击图中有个小感叹号的地方，选择”run insert()”







到此，ssm入门整合结束

源文档 <<https://blog.csdn.net/zyf2333/article/details/77623537>>