1、创建java项目

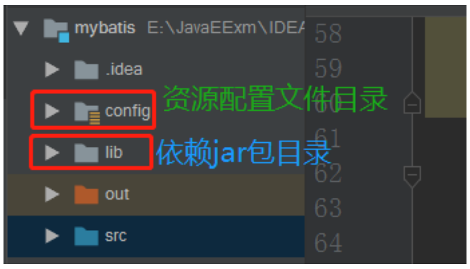
2、项目中创建config资源目录

3、项目中创建lib依赖目录

4、导入依赖jar包

5、创建测试类及javaBean类

6、创建资源配置文件



重点1、相关的jar包

asm-3.3.1.jar

cglib-2.2.2.jar

commons-logging-1.1.1.jar

javassist-3.17.1-GA.jar

junit-4.7.jar **junit**测试**jar**包

log4j-1.2.17.jar

log4j-api-2.0-rc1.jar

log4j-core-2.0-rc1.jar

mybatis-3.2.7.jar **mybatis**核心**jar**包

mysql-connector-java-5.1.7-bin.jar **mysql**数据驱动**jar**包

slf4j-api-1.7.5.jar

slf4j-log4j12-1.7.5.jar

重点2、资源配置文件

mybatis核心配置文件（全局配置文件：加载mybatis基础支持信息）

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 约束信息 -->
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<!-- mybatis全局配置文件：加载mybatis基础支持信息，
    和spring进行整合被废弃 -->
<configuration>
<environments default="development">
<environment id="development">
<!--jdbc事务管理，mybatis负责执行管理  -->
<transactionManager type="JDBC"/>
<!-- jdbc数据源，mybatis负责管理 -->
```

```
<dataSource type="POOLED">
<property name="driver" value="com.mysql.jdbc.Driver"/>
<property name="url" value="jdbc:mysql://localhost:3306/sshcrm03?characterEncoding=utf-8"/>
<property name="username" value="root"/>
<property name="password" value="123456"/>
</dataSource>
</environment>
</environments>
<mappers>
<!-- 导入javaBean对应的映射文件，以便加载配置映射 -->
<mapper resource="napper/Users.xml"/>
</mappers>
</configuration>
```

javabean映射配置文件　xxxx.xml

```
<?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">
 <!-- 统一存放SQL语据的配置文件 -->
<mapper namespace="Test">
    <!--
     需求：查询所有
     参数：无
     返回值类型：集合类型list<Users>
     select:查询语句标签
     id:本条查询语句唯一标识
     resultType:返回值类型
     -->
   <select id="findAll"  resultType="cn.rong.zx.Users">
    select * from users
   </select>
   <!--
     需求：根据id查询一条数据
     参数：有
     返回值类型：集合类型list<Users>
     select:查询语句标签
     id:本条查询语句唯一标识
     resultType:返回值类型
     parameterType: 输入参数类型
    #{}:表示占位符，括号填写的参数名。如果说传递参数是基本类型：int long String等等。
    #{}当中可以是任意的值
    如果说传递参数是pojo类型，使用ognl(对象导航语言)表达式进行获取：属性.属性.属性
     -->
   <select id="findUbyID" parameterType="int"  resultType="cn.rong.zx.Users">
    select * from users where u_id=#{id}
   </select>

   <!--
   需求：模糊查询
    参数：用户名
    返回值类型：集合类型list<Users>
    resultType:返回值类型
    parameterType: 输入参数类型
    #{}:占位符，如果是基本类型，原样进行获取，如果是字符类型：‘张’
         select * from users where uname like "%"#{uname}"%"
    ${}: 表示sql拼接，不管是什么值，不加修饰的拼接
    ${}:如果传递参数是基本类型：只能是value.如果是pojo类型：使用ognl表达式进行获取
        select * from users where uname like '%${value}%'
   -->
   <select id="findUserByuname" parameterType="string" resultType="cn.rong.zx.Users">
       select * from users where uname like '%${value}%'
    </select>

   <!--
   需求：保存用户数据
   参数：users对象
   返回值：保存当前用户数据生成的id值 useGeneratedKeys="true" keyProperty="id"
   -->
   <insert id="addUsers" parameterType="cn.rong.zx.Users" useGeneratedKeys="true" keyProperty="i
      <!--<selectKey keyProperty="u_id" order="AFTER" resultType="int">-->
         <!--SELECT LAST_INSERT_ID()-->
      <!--</selectKey>-->
        insert into users values(#{u_id},#{uname},#{upwd})
   </insert>

     <!--
     需求：更新数据（根据id修改数据）
     参数：Users对象
     -->
    <update id="updateUsersz" parameterType="cn.rong.zx.Users">
        update users set uname=#{uname},upwd=#{upwd} where u_id=#{u_id}
     <!--   update users set uname=#{uname} where u_id=#{u_id}     -->
```

```
    </update>
</mapper>
```