根据

根据方法需要自己建立mybatis

# 1.、旧版本的ibatis，也就是没有mapper接口的时候

## 1.建立配置文件与数据库对应（相当于hibernate中的那个）

**<transactionManager type="JDBC">**

**<dataSource type="SIMPLE">**

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

**<property name="JDBC.ConnectionURL" value="jdbc:mysql://localhost:3306/ibatis" />**

**<property name="JDBC.Username" value="root" />**

**<property name="JDBC.Password" value="147094" />**

**<property name="Pool.MaximumActiveConnections" value="10" />**

**<property name="Pool.MaximumIdleConnections" value="5" />**

**<property name="Pool.MaximumCheckoutTime" value="120000" />**

**<property name="Pool.TimeToWait" value="500" />**

**<property name="Pool.PingQuery" value="select 1 from ACCOUNT" />**

**<property name="Pool.PingEnabled" value="false" />**

**<property name="Pool.PingConnectionsOlderThan" value="1" />**

**<property name="Pool.PingConnectionsNotUsedFor" value="1" />**

**</dataSource>**

**</transactionManager>**

**<!-- 映射文件 --> 相当于hibernate中的那个与数据库对应的文件，但是这里的区别就出来了，这里的区别是它是sql语句的集合**

**<sqlMap resource="com/itmyhome/User.xml" />**

## 2. 编写User.xml文件，这里用来编写数据库操作语句

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

**<!DOCTYPE sqlMap**

**PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN"**

**"http://www.ibatis.com/dtd/sql-map-2.dtd">**

**<sqlMap namespace="User">**

**下面这个user是别名，就是代表后面的type的java文件 ，下文用它来对数据进行草操作**

**<typeAlias alias="user" type="com.itmyhome.User" />**

**</select>**

**<!-- 查询所有的 --> 这里的id在java文件中会调用的**

**<select id="getAllUser" resultClass="user">**

**select \* from user**

**</select>**

**</sqlMap>**

## 3.增删改查

**<sqlMap namespace="User">**

**<typeAlias alias="user" type="com.itmyhome.User" />**

**<!-- 查询 -->**

**<!-- parameterClass 传入参数的类型 int = java.lang.Integer**

**resultClass 返回结果的类型 -->**

**<select id="getUser" parameterClass="int" resultClass="user">**

**select \* from user where id = #id#**

**</select>**

**<!-- 查询所有的 -->**

**<select id="getAllUser" resultClass="user">**

**select \* from user**

**</select>**

**<!-- 插入 -->**

**<insert id="insertUser" parameterClass="user">**

**insert into user(id,name,age) values(#id#,#name#,#age#)**

**</insert>**

**<!-- 更新 -->**

**<update id="updateUser" parameterClass="user">**

**update user SET name=#name#, age=#age# WHERE id = #id#**

**</update>**

**<!-- 删除 -->**

**<delete id="deleteUser" parameterClass="java.lang.Integer">**

**delete from user where id = #id#**

**</delete>**

## 4.取得sqlMap

**String resource = "SqlMapConfig.xml";**

**Reader reader = Resources.getResourceAsReader(resource); //读取配置文件**

**sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);**

## 5.进行测试

**List<User> userList = sqlMap.queryForList("getAllUser");**

**System.err.println("得到list的大小为"+userList.size());**

**for(User user:userList){**

**System.out.println(user.toString());**

**}**

**user = (User)sqlMap.queryForObject ("getUser", 3);**

**System.out.println(user.getName());**

**sqlMap.insert("insertUser",u);**

**sqlMap.update("updateUser",u2);**

**sqlMap.delete("deleteUser", 3);**

# 2、简单的mybatis

## 1.创建实体类

**public** **class** User {

**private** String id;

**private** String name;

**private** **int** age;

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

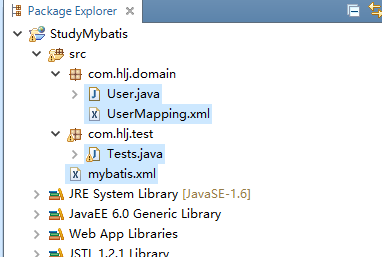
**public** String toString() {

**return** "User [id=" + id + ", name=" + name + ", age=" + age + "]";

}

}

## 2.编写连接数据库配置文件 mybatis.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>

<environments default=*"development"*>

<environment id=*"development"*>

<transactionManager type=*"JDBC"* />

<!-- 配置数据库连接信息 -->

<dataSource type=*"POOLED"*>

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

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

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

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

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource=*"com/hlj/domain/userMapping.xml"*/>

</mappers>

</configuration>

## 3.编写映射文件，添加sql语句

|  |
| --- |
| <?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=*"com.hlj.domain.UserMapping"*>  <!-- 在select标签中编写查询的SQL语句， 设置select标签的id属性为getUser，id属性值必须是唯一的，不能够重复  使用parameterType属性指明查询时使用的参数类型，resultType属性指明查询返回的结果集类型  resultType="com.hlj.domain.User"就表示将查询结果封装成一个User类的对象返回  User类就是users表所对应的实体类  -->  <!--  根据id查询得到一个user对象  -->  <select id=*"getUser"* parameterType=*"String"*  resultType=*"com.hlj.domain.User"*>  select \* from user where id=#{id}  </select>  </mapper> |

## 4.进行测试

|  |
| --- |
| **public** **class** Tests {    **public** **static** **void** main(String[] args) {  Tests tests = **new** Tests();  tests.test();  }  **public** **void** test(){  String resource = "mybatis.xml";  //使用类加载器加载mybatis的配置文件（它也加载关联的映射文件）  InputStream is = Tests.**class**.getClassLoader().getResourceAsStream(resource);  //构建sqlSession的工厂  SqlSessionFactory sessionFactory = **new** SqlSessionFactoryBuilder().build(is);    SqlSession session = sessionFactory.openSession();    /\*\*  \* 映射sql的标识字符串，  \* com.hlj.domain.UserMapping是userMapper.xml文件中mapper标签的namespace属性的值，  \* getUser是select标签的id属性值，通过select标签的id属性值就可以找到要执行的SQL  \*/  String statement = "com.hlj.domain.UserMapping.getUser";//映射sql的标识字符串  //执行查询返回一个唯一user对象的sql  User user = session.selectOne(statement, "1123");    System.*out*.println(user.toString());  } |

# 3、mybatis逆向工程，去sinosoft中找吧

# 4、spring boot开始mybatis

## 1、添加mybatis扫描包（其实下面的配置扫的主要还是mapper和mapper.xml）

@Configuration  
public class MybatisConfig {  
  
 @Bean  
 public MapperScannerConfigurer mapperScannerConfigurer(){  
 MapperScannerConfigurer configurer = new MapperScannerConfigurer();  
 configurer.setBasePackage("com.duodian.admore.dao.mybatis.\*");  
 return configurer;  
 }  
  
 @Bean  
 public SqlSessionFactoryBean sessionFactory(DataSource dataSource,ApplicationContext applicationContext) throws IOException{  
 SqlSessionFactoryBean sessionFactoryBean = new SqlSessionFactoryBean();  
 sessionFactoryBean.setDataSource(dataSource);  
 //sessionFactoryBean.setConfigLocation(applicationContext.getResource("classpath:mybatis.xml"));  
 sessionFactoryBean.setMapperLocations(applicationContext.getResources("classpath\*:com/duodian/admore/dao/mybatis/\*\*/mysql/\*.xml"));  
 return sessionFactoryBean;  
 }  
  
}

@Configuration  
public class MybatisConfig {  
  
 @Bean  
 public MapperScannerConfigurer mapperScannerConfigurer(){  
 MapperScannerConfigurer configurer = new MapperScannerConfigurer();  
 configurer.setBasePackage("com.duodian.admore.dao.\*");  
 return configurer;  
 }  
  
 @Bean  
 public SqlSessionFactoryBean sessionFactory(DataSource dataSource,ApplicationContext applicationContext) throws IOException{  
 SqlSessionFactoryBean sessionFactoryBean = new SqlSessionFactoryBean();  
 sessionFactoryBean.setDataSource(dataSource);  
 sessionFactoryBean.setConfigLocation(applicationContext.getResource("classpath:mybatis.xml"));  
  
 Resource[] resources = ArrayUtils.addAll(  
 applicationContext.getResources("classpath\*:com/duodian/admore/dao/mybatis/\*\*/mysql/\*.xml"),  
 applicationContext.getResources("classpath\*:com/duodian/admore/dao/db/\*\*/mysql/\*.xml")  
 );  
 sessionFactoryBean.setMapperLocations(resources);  
 return sessionFactoryBean;  
 }  
  
  
}

## 2、mapper

package com.duodian.admore.dao.mybatis.contact;  
  
import com.duodian.admore.entity.db.contact.ContactRecord;  
import org.apache.ibatis.annotations.Param;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.Pageable;  
  
import java.util.List;  
  
public interface ContactRecordMapper {  
  
  
 /\*\*  
 \* 分页查询连续记录  
 \* @param customerName  
 \* @param pactType  
 \* @param pageNow  
 \* @param pageSize  
 \* @return  
 \*/  
 List<ContactRecord> findPageByCustomerNameAndPactType(  
 @Param("customerName") String customerName,  
 @Param("pactType") String pactType ,  
 @Param("pageNow") int pageNow,  
 @Param("pageSize") int pageSize);  
  
 Long countPageByCustomerNameAndPactType();  
}

**3、mapper.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" *>*<mapper namespace="com.duodian.admore.dao.mybatis.contact.ContactRecordMapper">  
  
  
 <!-- 根据分页查找出 -->  
 <select id="findPageByCustomerNameAndPactType" parameterType="map" resultType="com.duodian.admore.entity.db.contact.ContactRecord">  
  
 SELECT *\** FROM contact\_record as c where  
 (c.customerName like CONCAT('%',#{customerName},'%') or #{customerName} =NULL ) and  
 (c.pactType like CONCAT('%',#{pactType},'%') or #{pactType} =NULL)  
 ORDER BY c.addTime DESC limit #{pageNow},#{pageSize}  
 </select>  
  
 <select id="countPageByCustomerNameAndPactType" resultType="java.lang.Long">  
 select count(*\**) from contact\_record  
 </select>  
  
  
</mapper>