# mybatis

学习内容如下：

1： session 调用配置文件

2：接口调用实现类进行增删改查

3：直接调用接口进行增删改查（接口调用配置文件进行增删改查）

mybaitis的前身叫ibatis的前身<batis

mybatis有两种配置文件： SqlMapConfig.xml 一个

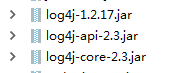
PersonMapper(PersonDao) PersonMapper.xml 多个

log4j.xml

内容

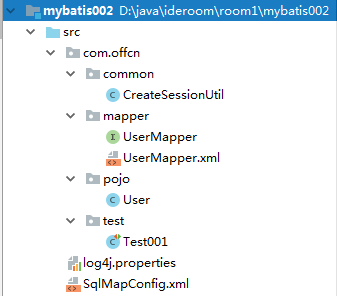
*# Global logging configuration***log4j.rootLogger**=**DEBUG, stdout***# Console output...***log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender  
log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout  
log4j.appender.stdout.layout.ConversionPattern**=**%5p [%t] - %m%n**

所需jar包



mybatis框架使用步骤如下

1：项目架构



加载SqlMapConfig.xml  
public class CreateSessionUtil {

public static SqlSession getSesion(){

InputStream resourceAsStream = null;

try {

//通过io流加载SqlMapConfig配置文件

resourceAsStream = Resources.getResourceAsStream("SqlMapConfig.xml");

} catch (IOException e) {

}

**//mybatis通过读取配置文件信息（全局配置文件和映射文件），构造出SqlSessionFactory，即会话工厂。**

SqlSessionFactory build = new SqlSessionFactoryBuilder().build(resourceAsStream);

**//通过SqlSessionFactory，可以创建SqlSession即会话。Mybatis是通过SqlSession来操作数据库的。**

**//openSession的主要就是初始化了SqlMapConfig.xml的配置信息和DefaultSqlSession**

SqlSession session = build.openSession();

return session;

}

**UserMapper.java(接口)**  
**public interface** UserMapper {  
 **public int** deleteUserById(**int** id);  
 **public** User finUserById(**int** id);  
 **public void** instUser(User user);  
}

**SqlMapConfig.xml**  
 <!--  
 别名 type类名 alias自定义名-  
 坑1： Caused by: org.apache.ibatis.builder.BuilderException: Error creating document instance. Cause: org.xml.sax.SAXParseException; lineNumber: 33; columnNumber: 17; 元素类型为 "configuration" 的内容必须匹配 "(properties?,settings?,typeAliases?,typeHandlers?,objectFactory?,objectWrapperFactory?,reflectorFactory?,plugins?,environments?,databaseIdProvider?,mappers?)"。  
  **别名 不能放到environments下面去**--><**typeAliases**>  
 <**typeAlias type="com.offcn.pojo.User" alias="user"**></**typeAlias**>  
 </**typeAliases**>

*<!-- 配置mybatis的环境信息 -->*<**environments default="development"**>  
 <**environment id="development"**>  
 *<!-- 配置JDBC事务控制，由mybatis进行管理 -->* <**transactionManager type="JDBC"**></**transactionManager**>  
 *<!-- 配置数据源，采用dbcp连接池 -->* <**dataSource type="POOLED"**>  
 <**property name="driver" value="com.mysql.jdbc.Driver"**/>  
 <**property name="url" value="jdbc:mysql://localhost:3306/m001?useUnicode=true&amp;characterEncoding=utf8"**/>  
 <**property name="username" value="root"**/>  
 <**property name="password" value="root"**/>  
 </**dataSource**>  
 </**environment**>  
</**environments**>  
  
 <**mappers**>  
 *<!--  
 resource相对路径  
 class接口  
 package加载包  
 三选一  
 -->* <**mapper resource="com/offcn/mapper/UserMapper.xml"**></**mapper**>  
 *<!--<mapper class="com.offcn.mapper.UserMapper"></mapper>  
 <package name="com.offcn.mapper"></package>-->* </**mappers**>

**UserMapper.xml**

*<!--  
namespace映射 接口  
-->*

**//链接到userMapper**<**mapper namespace="com.offcn.mapper.UserMapper"**>  
  
<!-- public int deleteUserById(**int** id);--><**delete id="deleteUserById" parameterType="int"** >  
 delete from user where uid = #{id}  
</**delete**>  
  
 <!-- public User finUserById(int id);--><**select id="finUserById" parameterType="int" resultType="com.offcn.pojo.User"**>  
 select *\** from user where uid = #{uid}  
</**select**>  
 *<!--public void instUser(User user);  
 当我们的表主键自增的时候，我想获取到主键的值 （uid的值）  
 -->* <**insert id="instUser" parameterType="user"** >  
 /\*  
 **resultTyp 返回的类型  
 keyProperty类中的属性  
 keyColumn数据库中表的字段名  
 order 在运行之前还是之后获取这个值  
 select last\_insert\_id(); 固定的** \*/  
 <**selectKey resultType="int" keyProperty="uid" keyColumn="uid" order="AFTER"**>  
 select last\_insert\_id();  
 </**selectKey**>  
 insert into user (uname) values (#{uname});  
 </**insert**>

**测试类：**

*/\*ctrl + alt +L 排版\*/*@Test  
**public void** test001() **throws** Exception {  
  
 SqlSession sesion = CreateSessionUtil.*getSesion*();  
  
 UserMapper userMapper = sesion.getMapper(UserMapper.**class**);  
  
 **int** i = userMapper.deleteUserById(1002);  
  
 System.***out***.println(i);  
  
 sesion.commit();  
 sesion.close();  
}

------------------------输入参数映射----------------

种类如下：

1： int

*<!--public Employee getEmployeeById(int id);-->*<**select id="getEmployeeById" parameterType="int" resultType="com.offcn.pojo.Employee"**>  
 select *\** from employee where eid = #{id}  
</**select**>

2：String

*<!--public List<Employee> getEmployeeByAddress(String address);-->* <**select id="getEmployeeByAddress" parameterType="java.lang.String" resultType="com.offcn.pojo.Employee"**>  
 select *\** from employee where address = #{id}  
 </**select**>

3：Person 自定义对象

*<!-- public Employee getEmployeeByLoginUserVo(LoginUserVo loginUserVo);-->*<**select id="getEmployeeByLoginUserVo"parameterType="com.offcn.pojo.dto.LoginUserVo" resultType="com.offcn.pojo.Employee"**>  
 select *\** from employee where ename = #{name} and address = #{addre}  
 </**select**>

3.1 多参   
*<!-- public Employee getEmpByEnameAndPassword(String ename , String address);  
 多参： 不写parameterType  
 第一种办法：参数按照下标  
 单参：String，int 。。。可以不写（除了对象）  
-->* <**select id="getEmpByEnameAndPassword" resultType="com.offcn.pojo.Employee"**>  
 select *\** from employee where ename = #{0} and address = #{1}  
 </**select**>

4：Map

》》》》int  
 *<!--  
 public Employee getEmpByEnameAndPasswordMap(Map<String,Object> map);  
 参数为Map的时候，要根据键值对的键来！！！  
-->* <**select id="getEmpByEnameAndPasswordMap"**

**resultType="com.offcn.pojo.Employee" parameterType="java.util.Map"**>  
 select *\** from employee where ename = #{ss} and address = #{mm}  
 </**select**>

EmployeeMapper:

<**mapper** **namespace**=**"com.offcn.mapper.EmployeeMapper"**>

<!--public Employee getEmployeeById(int id);-->

<select id="**getEmployeeById**" **resultType**="**com.offcn.pojo.Employee**">

select \* from employee where eid =#{id}

</select>

<!--public void insertEmp(Employee employee);-->

<**insert** id=**"insertEmp"** **parameterType**=**"com.offcn.pojo.Employee"**>

insert into employee values (#{eid},#{ename},#{address},#{age},#{birthday})

</**insert**>

<!-- public Employee getEmpByEnameAndPassword(String ename , String address);

多参： 不写parameterType

第一种办法：参数按照**下标**

单参：String，int 。。。可以不写（除了对象）-->

<**select** id=**"getEmpByEnameAndPassword"** **resultType**=**"com.offcn.pojo.Employee"**>

select \* from employee where ename=#{arg0} and address=#{arg1}

</**select**>

<!--public Employee getEmpByEnameAndPasswordMap(Map<String,Object> map);

参数为Map的时候，要根据键值对的键来！！！-->

<**select**

id=**"getEmpByEnameAndPasswordMap"** **resultType**=**"com.offcn.pojo.Employee"** **parameterType**=**"java.util.Map"**>

select \* from employee where ename=#{**ss**} and address=#{**mm**}

</**select**>

<!-- public Employee getEmployeeByLoginUserVo(LoginUserVo loginUserVo);-->

<**select** id=**"getEmployeeByLoginUserVo"** **parameterType**=**"com.offcn.pojo.dto.LoginUserVo"**

**resultType**="com.offcn.pojo.Employee">

select \* from employee where ename = #{name} and address = #{addre}

</**select**>

<!--public List<Employee> getStringByAddress(String address);最后一个list<string返回>-->

<!--那么上面说过单参数可以不用填写入参类型,这里我们使用**模糊查询**concat('%',#{name},'%')-->

<**select** id="**getStringByAddress**" **resultType**="**com.offcn.pojo.Employee**">

**select \* from employee where address like concat('%',#{address},'%')**

</**select**>

</mapper>