**第一章 回顾jdbc开发**

1）优点：简单易学,上手快,非常灵活构建SQL，效率高

2）缺点：代码繁琐，难以写出高质量的代码（例如：资源的释放，SQL注入安全性等）

开发者既要写业务逻辑，又要写对象的创建和销毁，必须管底层具体数据库的语法

（例如：分页）。

3）适合于超大批量数据的操作，速度快

**第二章 回顾hibernate单表开发**

1）优点：不用写SQL，完全以面向对象的方式设计和访问，不用管底层具体数据库的语法，（例如：分页）便于理解。

2）缺点：处理复杂业务时，灵活度差, 复杂的HQL难写难理解，例如多表查询的HQL语句

3）适合于中小批量数据的操作，速度慢

**第三章 什么是mybatis，mybatis有什么特点**

1）基于上述二种支持，我们需要在中间找到一个平衡点呢？结合它们的优点，摒弃它们的缺点，

这就是myBatis，现今myBatis被广泛的企业所采用。

2）MyBatis 本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。2013年11月迁移到Github。

3）iBATIS一词来源于“internet”和“abatis”的组合，是一个基于Java的持久层框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects（DAO）

4）jdbc/dbutils/springdao，hibernate/springorm，mybaits同属于ORM解决方案之一

**第四章 mybatis快速入门**

1）创建一个mybatis-day01这么一个javaweb工程或java工程

2）导入mybatis和mysql/oracle的jar包到/WEB-INF/lib目录下

|  |
| --- |
|  |

3）创建students.sql表

|  |
| --- |
| --mysql语法  create table students(  id int(5) primary key,  name varchar(10),  sal double(8,2)  );  --oracle语法  create table students(  id number(5) primary key,  name varchar2(10),  sal number(8,2)  ); |

4）创建Student.java

|  |
| --- |
| /\*\*  \* 学生  \* **@author** AdminTC  \*/  **public** **class** Student {  **private** Integer id;  **private** String name;  **private** Double sal;  **public** Student(){}  **public** Integer getId() {  **return** id;  }  **public** **void** setId(Integer id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** Double getSal() {  **return** sal;  }  **public** **void** setSal(Double sal) {  **this**.sal = sal;  }  } |

5）在entity目录下创建StudentMapper.xml配置文件

|  |
| --- |
| <?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="mynamespace">  <insert id="add1">  insert into students(id,name,sal) values(1,'哈哈',7000)  </insert>  <insert id="add2" parameterType="cn.itcast.javaee.mybatis.app05.Student">  insert into students(id,name,sal) values(#{id},#{name},#{sal})  </insert>  </mapper> |

6）在src目录下创建mybatis.xml配置文件

|  |
| --- |
| <?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"/>  <dataSource type="POOLED">  <property name="driver" value="com.mysql.jdbc.Driver"/>  <property name="url" value="jdbc:mysql://127.0.0.1:3306/mybatis"/>  <property name="username" value="root"/>  <property name="password" value="root"/>  </dataSource>  </environment>  </environments>  <mappers>  <mapper resource="cn/itcast/javaee/mybatis/app05/StudentMapper.xml"/>  </mappers>  </configuration> |

7）在util目录下创建MyBatisUtil.java类，并测试与数据库是否能连接

|  |
| --- |
| /\*\*  \* MyBatis工具类  \* **@author** AdminTC  \*/  **public** **class** MyBatisUtil {  **private** **static** ThreadLocal<SqlSession> *threadLocal* = **new** ThreadLocal<SqlSession>();  **private** **static** SqlSessionFactory *sqlSessionFactory*;  **static**{  **try** {  Reader reader = Resources.*getResourceAsReader*("mybatis.xml");  *sqlSessionFactory* = **new** SqlSessionFactoryBuilder().build(reader);  } **catch** (IOException e) {  e.printStackTrace();  **throw** **new** RuntimeException(e);  }  }  **private** MyBatisUtil(){}  **public** **static** SqlSession getSqlSession(){  SqlSession sqlSession = *threadLocal*.get();  **if**(sqlSession == **null**){  sqlSession = *sqlSessionFactory*.openSession();  *threadLocal*.set(sqlSession);  }  **return** sqlSession;  }  **public** **static** **void** closeSqlSession(){  SqlSession sqlSession = *threadLocal*.get();  **if**(sqlSession != **null**){  sqlSession.close();  *threadLocal*.remove();  }  }  **public** **static** **void** main(String[] args) {  Connection conn = MyBatisUtil.*getSqlSession*().getConnection();  System.*out*.println(conn!=**null**?"连接成功":"连接失败");  }  } |

8）在dao目录下创建StudentDao.java类并测试

|  |
| --- |
| /\*\*  \* 持久层  \* **@author** AdminTC  \*/  **public** **class** StudentDao {  /\*\*  \* 增加学生（无参）  \*/  **public** **void** add1() **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.insert("mynamespace.add1");  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  }  MyBatisUtil.*closeSqlSession*();  }  /\*\*  \* 增加学生（有参）  \*/  **public** **void** add2(Student student) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.insert("mynamespace.add2",student);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  }  MyBatisUtil.*closeSqlSession*();  }  **public** **static** **void** main(String[] args) **throws** Exception{  StudentDao dao = **new** StudentDao();  dao.add1();  dao.add2(**new** Student(2,"呵呵",8000D));  }  } |

**第五章 mybatis工作流程**

1）通过Reader对象读取src目录下的mybatis.xml配置文件(该文本的位置和名字可任意)

2）通过SqlSessionFactoryBuilder对象创建SqlSessionFactory对象

3）从当前线程中获取SqlSession对象

4）事务开始，在mybatis中默认

5）通过SqlSession对象读取StudentMapper.xml映射文件中的操作编号，从而读取sql语句

6）事务提交，必写

7）关闭SqlSession对象，并且分开当前线程与SqlSession对象，让GC尽早回收

**第六章 mybatis配置文件祥解（mybatis.xml）**

1）以下是StudentMapper.xml文件，提倡放在与实体同目录下，文件名任意

|  |
| --- |
| <?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="mynamespace">  <insert id="add1">  insert into students(id,name,sal) values(1,'哈哈',7000)  </insert>  <insert id="add2" parameterType="cn.itcast.javaee.mybatis.app05.Student">  insert into students(id,name,sal) values(#{id},#{name},#{sal})  </insert>  </mapper> |

**第七章 mybatis映射文件祥解(StudentMapper.xml)**

1）以下是mybatis.xml文件，提倡放在src目录下，文件名任意

|  |
| --- |
| <?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"/>  <dataSource type="POOLED">  <property name="driver" value="com.mysql.jdbc.Driver"/>  <property name="url" value="jdbc:mysql://127.0.0.1:3306/mybatis"/>  <property name="username" value="root"/>  <property name="password" value="root"/>  </dataSource>  </environment>  </environments>  <mappers>  <mapper resource="cn/itcast/javaee/mybatis/app05/StudentMapper.xml"/>  </mappers>  </configuration> |

**第八章MybatisUtil工具类的作用**

1）在静态初始化块中加载mybatis配置文件和StudentMapper.xml文件一次

2）使用ThreadLocal对象让当前线程与SqlSession对象绑定在一起

3）获取当前线程中的SqlSession对象，如果没有的话，从SqlSessionFactory对象中获取SqlSession对象

4）获取当前线程中的SqlSession对象，再将其关闭，释放其占用的资源

|  |
| --- |
| /\*\*  \* MyBatis工具类  \* **@author** AdminTC  \*/  **public** **class** MyBatisUtil {  **private** **static** ThreadLocal<SqlSession> *threadLocal* = **new** ThreadLocal<SqlSession>();  **private** **static** SqlSessionFactory *sqlSessionFactory*;  **static**{  **try** {  Reader reader = Resources.*getResourceAsReader*("mybatis.xml");  *sqlSessionFactory* = **new** SqlSessionFactoryBuilder().build(reader);  } **catch** (IOException e) {  e.printStackTrace();  **throw** **new** RuntimeException(e);  }  }  **private** MyBatisUtil(){}  **public** **static** SqlSession getSqlSession(){  SqlSession sqlSession = *threadLocal*.get();  **if**(sqlSession == **null**){  sqlSession = *sqlSessionFactory*.openSession();  *threadLocal*.set(sqlSession);  }  **return** sqlSession;  }  **public** **static** **void** closeSqlSession(){  SqlSession sqlSession = *threadLocal*.get();  **if**(sqlSession != **null**){  sqlSession.close();  *threadLocal*.remove();  }  }  **public** **static** **void** main(String[] args) {  Connection conn = MyBatisUtil.*getSqlSession*().getConnection();  System.*out*.println(conn!=**null**?"连接成功":"连接失败");  }  } |

**第九章 基于MybatisUtil工具类，完成CURD操作**

1）StudentDao.java

|  |
| --- |
| /\*\*  \* 持久层  \* **@author** AdminTC  \*/  **public** **class** StudentDao {  /\*\*  \* 增加学生  \*/  **public** **void** add(Student student) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.insert("mynamespace.add",student);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  /\*\*  \* 修改学生  \*/  **public** **void** update(Student student) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.update("mynamespace.update",student);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  /\*\*  \* 查询单个学生  \*/  **public** Student findById(**int** id) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  Student student = sqlSession.selectOne("mynamespace.findById",id);  **return** student;  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  /\*\*  \* 查询多个学生  \*/  **public** List<Student> findAll() **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  **return** sqlSession.selectList("mynamespace.findAll");  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  /\*\*  \* 删除学生  \*/  **public** **void** delete(Student student) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.delete("mynamespace.delete",student);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  } |

2）StudentMapper.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="mynamespace">  <insert id="add" parameterType="cn.itcast.javaee.mybatis.app09.Student">  insert into students(id,name,sal) values(#{id},#{name},#{sal})  </insert>  <update id="update" parameterType="cn.itcast.javaee.mybatis.app09.Student">  update students set name=#{name},sal=#{sal} where id=#{id}  </update>  <select id="findById" parameterType="int" resultType="cn.itcast.javaee.mybatis.app09.Student">  select id,name,sal from students where id=#{anything}  </select>  <select id="findAll" resultType="cn.itcast.javaee.mybatis.app09.Student">  select id,name,sal from students  </select>  <delete id="delete" parameterType="cn.itcast.javaee.mybatis.app09.Student">  delete from students where id=#{id}  </delete>  </mapper> |

**第十章 分页查询**

1）StudentDao.java

|  |
| --- |
| /\*\*  \* 持久层  \* **@author** AdminTC  \*/  **public** **class** StudentDao {  /\*\*  \* 增加学生  \*/  **public** **void** add(Student student) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.insert("mynamespace.add",student);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  /\*\*  \* 无条件分页查询学生  \*/  **public** List<Student> findAllWithFy(**int** start,**int** size) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  Map<String,Integer> map = **new** LinkedHashMap<String,Integer>();  map.put("pstart",start);  map.put("psize",size);  **return** sqlSession.selectList("mynamespace.findAllWithFy",map);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  /\*\*  \* 有条件分页查询学生  \*/  **public** List<Student> findAllByNameWithFy(String name,**int** start,**int** size) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  Map<String,Object> map = **new** LinkedHashMap<String,Object>();  map.put("pname","%"+name+"%");  map.put("pstart",start);  map.put("psize",size);  **return** sqlSession.selectList("mynamespace.findAllByNameWithFy",map);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  **public** **static** **void** main(String[] args) **throws** Exception{  StudentDao dao = **new** StudentDao();  System.*out*.println("-----------Page1");  **for**(Student s:dao.findAllByNameWithFy("哈",0,3)){  System.*out*.println(s.getId()+":"+s.getName()+":"+s.getSal());  }  System.*out*.println("-----------Page2");  **for**(Student s:dao.findAllByNameWithFy("哈",3,3)){  System.*out*.println(s.getId()+":"+s.getName()+":"+s.getSal());  }  System.*out*.println("-----------Page3");  **for**(Student s:dao.findAllByNameWithFy("哈",6,3)){  System.*out*.println(s.getId()+":"+s.getName()+":"+s.getSal());  }  System.*out*.println("-----------Page4");  **for**(Student s:dao.findAllByNameWithFy("哈",9,3)){  System.*out*.println(s.getId()+":"+s.getName()+":"+s.getSal());  }  }  } |

2）StudentMapper.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="mynamespace">  <insert id="add" parameterType="cn.itcast.javaee.mybatis.app10.Student">  insert into students(id,name,sal) values(#{id},#{name},#{sal})  </insert>  <select id="findAllWithFy" parameterType="map" resultType="cn.itcast.javaee.mybatis.app10.Student">  select id,name,sal from students limit #{pstart},#{psize}  </select>  <select id="findAllByNameWithFy" parameterType="map" resultType="cn.itcast.javaee.mybatis.app10.Student">  select id,name,sal from students where name like #{pname} limit #{pstart},#{psize}  </select>  </mapper> |

思考：oracle分页如何做呢？

**第十一章 动态SQL操作之查询**

1. 查询条件不确定，需要根据情况产生SQL语法，这种情况叫动态SQL
2. 参见<<动态SQL—查询.JPG>>

StudentDao.java

|  |
| --- |
| /\*\*  \* 持久层  \* **@author** AdminTC  \*/  **public** **class** StudentDao {  /\*\*  \* 动态SQL--查询  \*/  **public** List<Student> dynaSQLwithSelect(String name,Double sal) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  Map<String,Object> map = **new** LinkedHashMap<String, Object>();  map.put("pname",name);  map.put("psal",sal);  **return** sqlSession.selectList("mynamespace.dynaSQLwithSelect",map);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  **public** **static** **void** main(String[] args) **throws** Exception{  StudentDao dao = **new** StudentDao();    List<Student> studentList1 = dao.dynaSQLwithSelect("哈哈",**null**);  **for**(Student student : studentList1){  System.*out*.println(student.getId()+":"+student.getName()+":"+student.getSal());  }  System.*out*.println("--------------");  List<Student> studentList2 = dao.dynaSQLwithSelect(**null**,7000D);  **for**(Student student : studentList2){  System.*out*.println(student.getId()+":"+student.getName()+":"+student.getSal());  }  System.*out*.println("--------------");  List<Student> studentList3 = dao.dynaSQLwithSelect("哈哈",7000D);  **for**(Student student : studentList3){  System.*out*.println(student.getId()+":"+student.getName()+":"+student.getSal());  }  System.*out*.println("--------------");  List<Student> studentList4 = dao.dynaSQLwithSelect(**null**,**null**);  **for**(Student student : studentList4){  System.*out*.println(student.getId()+":"+student.getName()+":"+student.getSal());  }  System.*out*.println("--------------");  }  } |

StudentMapper.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="mynamespace">  <select id="dynaSQLwithSelect" parameterType="map" resultType="cn.itcast.javaee.mybatis.app11.Student">  select id,name,sal from students  <where>  <if test="pname!=null">  and name=#{pname}  </if>  <if test="psal!=null">  and sal=#{psal}  </if>  </where>  </select>  </mapper> |

**第十二章 动态SQL操作之更新**

1. 更新条件不确定，需要根据情况产生SQL语法，这种情况叫动态SQL
2. 参见<<动态SQL—更新.JPG>>

StudentDao.java

|  |
| --- |
| /\*\*  \* 持久层  \* **@author** AdminTC  \*/  **public** **class** StudentDao {  /\*\*  \* 动态SQL--更新  \*/  **public** **void** dynaSQLwithUpdate(Student student) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.update("mynamespace.dynaSQLwithUpdate",student);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  **public** **static** **void** main(String[] args) **throws** Exception{  StudentDao dao = **new** StudentDao();  dao.dynaSQLwithUpdate(**new** Student(10,**null**,5000D));  dao.dynaSQLwithUpdate(**new** Student(10,"哈哈",**null**));  dao.dynaSQLwithUpdate(**new** Student(10,"哈哈",6000D));  }  } |

StudentMapper.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="mynamespace">  <update id="dynaSQLwithUpdate" parameterType="cn.itcast.javaee.mybatis.app12.Student">  update students  <set>  <if test="name!=null">  name=#{name},  </if>  <if test="sal!=null">  sal=#{sal},  </if>  </set>  where id=#{id}  </update>  </mapper> |

**第十三章 动态SQL操作之删除**

1. 根据ID删除学生
2. 参见<<动态SQL—删除.JPG>>

StudentDao.java

|  |
| --- |
| /\*\*  \* 持久层  \* **@author** AdminTC  \*/  **public** **class** StudentDao {  /\*\*  \* 动态SQL--删除  \*/  **public** **void** dynaSQLwithDelete(**int**... ids) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.delete("mynamespace.dynaSQLwithDelete",ids);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  **public** **static** **void** main(String[] args) **throws** Exception{  StudentDao dao = **new** StudentDao();  dao.dynaSQLwithDelete(1,3,5,7);  }  } |

StudentMapper.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="mynamespace">  <!-- item表示迭代的参数 -->  <delete id="dynaSQLwithDelete">  delete from students where id in  <!--  <foreach collection="array" open="(" close=")" separator="," item="ids">  ${ids}  </foreach>  -->  <foreach collection="list" open="(" close=")" separator="," item="ids">  ${ids}  </foreach>  </delete>  </mapper> |

**第十四章 动态SQL操作之插入**

1. 根据条件，插入一个学生

2）参见<<动态SQL—插入.JPG>>

StudentDao.java

|  |
| --- |
| /\*\*  \* 持久层  \* **@author** AdminTC  \*/  **public** **class** StudentDao {  /\*\*  \* 动态SQL--插入  \*/  **public** **void** dynaSQLwithInsert(Student student) **throws** Exception{  SqlSession sqlSession = MyBatisUtil.*getSqlSession*();  **try**{  sqlSession.insert("mynamespace.dynaSQLwithInsert",student);  }**catch**(Exception e){  e.printStackTrace();  sqlSession.rollback();  **throw** e;  }**finally**{  sqlSession.commit();  MyBatisUtil.*closeSqlSession*();  }  }  **public** **static** **void** main(String[] args) **throws** Exception{  StudentDao dao = **new** StudentDao();  dao.dynaSQLwithInsert(**new** Student(1,"哈哈",7000D));  dao.dynaSQLwithInsert(**new** Student(2,"哈哈",**null**));  dao.dynaSQLwithInsert(**new** Student(3,**null**,7000D));  }  } |

StudentMapper.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="mynamespace">  <sql id="key">  <trim suffixOverrides=",">  <if test="id!=null">  id,  </if>  <if test="name!=null">  name,  </if>  <if test="sal!=null">  sal,  </if>  </trim>  </sql>  <sql id="value">  <trim suffixOverrides=",">  <if test="id!=null">  #{id},  </if>  <if test="name!=null">  #{name},  </if>  <if test="sal!=null">  #{sal},  </if>  </trim>  </sql>  <insert id="dynaSQLwithInsert" parameterType="cn.itcast.javaee.mybatis.app14.Student">  insert into students(<include refid="key"/>) values(<include refid="value"/>)  </insert>  </mapper> |

**第十五章 jsp/js/jquery/easyui/json + @springmvc + spring + mybatis + oracle开发**

**【员工管理系统--增加员工】**