目录

**[一、 实验内容](#_Toc16698_WPSOffice_Level1)** **[3](#_Toc16698_WPSOffice_Level1)**

[1.1目的：](#_Toc5463_WPSOffice_Level2) [3](#_Toc5463_WPSOffice_Level2)

[1.2要求：](#_Toc4550_WPSOffice_Level2) [3](#_Toc4550_WPSOffice_Level2)

**[二、仪器、设备和材料](#_Toc5463_WPSOffice_Level1)** **[3](#_Toc5463_WPSOffice_Level1)**

[1. 适合实验要求的计算机系统 。](#_Toc21879_WPSOffice_Level2) [3](#_Toc21879_WPSOffice_Level2)

[2. IntelliJ IEDA，MySQL。](#_Toc17691_WPSOffice_Level2) [3](#_Toc17691_WPSOffice_Level2)

**[三、需求分析](#_Toc4550_WPSOffice_Level1)** **[3](#_Toc4550_WPSOffice_Level1)**

[3.1ER模型设计](#_Toc21753_WPSOffice_Level2) [3](#_Toc21753_WPSOffice_Level2)

[3.2数据库表设计](#_Toc7213_WPSOffice_Level2) [4](#_Toc7213_WPSOffice_Level2)

[3.3MySql代码](#_Toc26773_WPSOffice_Level2) [4](#_Toc26773_WPSOffice_Level2)

**[四、功能设计](#_Toc21879_WPSOffice_Level1)** **[5](#_Toc21879_WPSOffice_Level1)**

[1.系统为管理系统，需要一个登陆功能。](#_Toc18948_WPSOffice_Level2) [5](#_Toc18948_WPSOffice_Level2)

[2.首先，需要对student进行管理，增删改查。](#_Toc2998_WPSOffice_Level2) [5](#_Toc2998_WPSOffice_Level2)

[3.对building表的增删改查。](#_Toc9940_WPSOffice_Level2) [6](#_Toc9940_WPSOffice_Level2)

[4.对dormitory表的增删改查。](#_Toc15951_WPSOffice_Level2) [6](#_Toc15951_WPSOffice_Level2)

[5.对zszt表的增删改查。](#_Toc24846_WPSOffice_Level2) [6](#_Toc24846_WPSOffice_Level2)

**[五、详细设计](#_Toc17691_WPSOffice_Level1)** **[6](#_Toc17691_WPSOffice_Level1)**

[5.1 student功能](#_Toc4656_WPSOffice_Level2) [6](#_Toc4656_WPSOffice_Level2)

[5.2 building功能](#_Toc29662_WPSOffice_Level2) [12](#_Toc29662_WPSOffice_Level2)

[5.3 dormitory功能](#_Toc27655_WPSOffice_Level2) [16](#_Toc27655_WPSOffice_Level2)

[5.4 zszt功能](#_Toc30371_WPSOffice_Level2) [21](#_Toc30371_WPSOffice_Level2)

**[六、 界面设计](#_Toc21753_WPSOffice_Level1)** **[28](#_Toc21753_WPSOffice_Level1)**

[6.1登陆界面](#_Toc23929_WPSOffice_Level2) [28](#_Toc23929_WPSOffice_Level2)

[6.2 student的两个界面](#_Toc13609_WPSOffice_Level2) [28](#_Toc13609_WPSOffice_Level2)

[6.3 building的两个界面](#_Toc26870_WPSOffice_Level2) [29](#_Toc26870_WPSOffice_Level2)

[6.4 dormitory的两个界面](#_Toc16040_WPSOffice_Level2) [30](#_Toc16040_WPSOffice_Level2)

[6.5 zszt的两个界面](#_Toc17715_WPSOffice_Level2) [30](#_Toc17715_WPSOffice_Level2)

**[七、 结束语](#_Toc7213_WPSOffice_Level1)** **[31](#_Toc7213_WPSOffice_Level1)**

**[八、 参考文献](#_Toc26773_WPSOffice_Level1)** **[32](#_Toc26773_WPSOffice_Level1)**

#### 实验内容

##### 1.1目的：

能设置每栋楼的基本信息。能记录每间房间住宿情况。能够查询、修改、删除每间房的住宿信息。

##### 1.2要求：

用户界面设计：采用窗口式、界面要友好、操作要简单

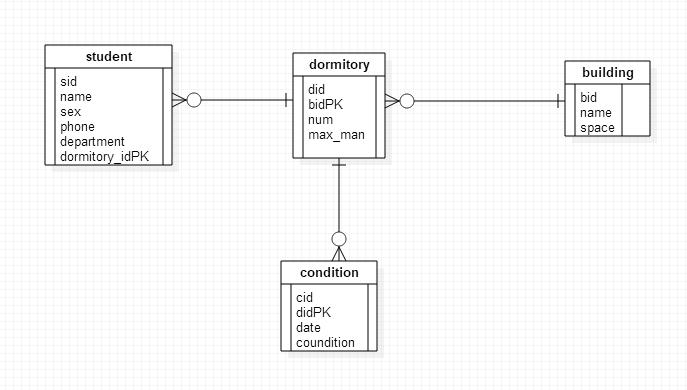
#### 二、仪器、设备和材料

1. 适合实验要求的计算机系统 。

2. IntelliJ IEDA，MySQL。

#### 三、需求分析

##### 3.1ER模型设计



##### 3.2数据库表设计

1）building表，用来存放楼栋信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义说明 | 空值情况 |
| bid | int | 楼栋编号，主键 | 不为空 |
| bname | char | 楼栋名称 | 可以为空 |
|  |  |  |  |

1. dormitory表，用来存放宿舍信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义说明 | 空值情况 |
| did | int | 宿舍编号，主键 | 自增主键 |
| bid | int | 宿舍所在楼栋编号，外键 | 不为空 |
| num | char | 宿舍门牌 | 不为空 |
| max\_man | int | 宿舍规格（几人） | 不为空 |

1. student表，用来存放学生信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义说明 | 空值情况 |
| sid | int | 学生编号，主键 | 不为空 |
| name | char | 学生姓名 | 不为空 |
| sex | char | 性别 | 不为空 |
| phone | char | 联系电话 | 可为空 |
| department | char | 学生专业 | 不为空 |
| did | int | 学生所住宿舍编号，外键 | 可为空 |

1. zszt表，用来存放每天的宿舍住宿状态

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义说明 | 空值情况 |
| zid | int | 编号，主键 | 不为空 |
| did | int | 宿舍编号，外键 | 不为空 |
| date | date | 日期 | 不为空 |
| zt | char | 宿舍情况 | 可为空 |

##### 3.3MySql代码

1）building表

CREATE TABLE `building` (

`bid` int NOT NULL AUTO\_INCREMENT,

`name` varchar(5),

`place` varchar(20),

PRIMARY KEY (`bid`)

) ENGINE=InnoDB AUTO\_INCREMENT=32 DEFAULT CHARSET=utf8;

2）dormitory表

CREATE TABLE `dormitory` (

`did` int(11) NOT NULL AUTO\_INCREMENT,

`bid` int(11) NOT NULL,

`num` varchar(5) NOT NULL,

`max\_man` int(11) NOT NULL,

PRIMARY KEY (`did`),

UNIQUE KEY `uniquekey` (`bid`,`num`) USING BTREE,

CONSTRAINT `dormitory\_ibfk\_1` FOREIGN KEY (`bid`) REFERENCES `building` (`bid`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

3）zszt表

CREATE TABLE `zszt` (

`zid` int(11) NOT NULL AUTO\_INCREMENT,

`did` int(11) NOT NULL,

`date` date NOT NULL,

`zt` char(50) DEFAULT NULL,

PRIMARY KEY (`zid`),

UNIQUE KEY `uniquekey` (`did`,`date`) USING BTREE,

CONSTRAINT `zszt\_ibfk\_1` FOREIGN KEY (`did`) REFERENCES `dormitory` (`did`)

) ENGINE=InnoDB AUTO\_INCREMENT=39 DEFAULT CHARSET=utf8;

4)student表

CREATE TABLE `student` (

`sid` int NOT NULL AUTO\_INCREMENT,

`sname` varchar(10) NOT NULL,

`sex` varchar(1) NOT NULL CHECK('男' '女'),

`phone` varchar(11),

`department` varchar(20) NOT NULL,

`did` int DEFAULT NULL,

PRIMARY KEY (`sid`),

foreign key (`did`) references dormitory(`did`)

) ENGINE=InnoDB AUTO\_INCREMENT=32 DEFAULT CHARSET=utf8;

#### 四、功能设计

1.系统为管理系统，需要一个登陆功能。

2.首先，需要对student进行管理，增删改查。

3.对building表的增删改查。

4.对dormitory表的增删改查。

5.对zszt表的增删改查。

#### 五、详细设计

##### 5.1 student功能

Mapper：

这里主要是sql语句的编写。

1. 查询学生列表

<select id="queryStudents" resultMap="studentMap">

SELECT s.sid,s.name,s.sex,s.phone,s.department,d.did,d.num,b.bid,b.bname,b.place

FROM

student s,dormitory d,building b

WHERE s.did = d.did

AND

d.bid = b.bid

ORDER BY sid;

</select>

1. 根据did（dormitory的id）查询学生列表

<select id="queryStudentByDid" resultMap="studentMap">

SELECT s.sid,s.name,s.sex,s.phone,s.department,d.did,d.num,b.bid,b.bname,b.place

FROM

student s,dormitory d,building b

WHERE s.did = d.did

AND

d.bid = b.bid

AND

s.did = #{did}

ORDER BY sid;

</select>

1. 根据did（dormitory的id）查询学生人数

<select id="queryStudentCountByDid" resultType="int">

SELECT count(\*)

FROM

student s

WHERE

s.did = #{did}

</select>

1. 根据sid查询学生

<select id="queryStudentBySid" resultMap="studentMap">

SELECT s.sid,s.name,s.sex,s.phone,s.department,d.did,d.num,b.bid,b.bname,b.place

FROM

student s,dormitory d,building b

WHERE s.did = d.did

AND

d.bid = b.bid

AND

s.sid = #{sid}

ORDER BY sid;

</select>

1. 插入学生

<insert id="insertStudent" parameterType="com.youyi.dormitory.entity.Student" useGeneratedKeys="true" keyProperty="sid" keyColumn="sid">

INSERT INTO

student(name,sex,phone,department,did)

VALUES

(#{name},#{sex},#{phone},#{department},#{dormitory.did});

</insert>

1. 更新学生

<update id="updateStudent" parameterType="com.youyi.dormitory.entity.Student">

UPDATE student

<set>

<if test="name != null">name = #{name},</if>

<if test="sex != null">sex = #{sex},</if>

<if test="phone != null">phone = #{phone},</if>

<if test="department != null">department = #{department},</if>

<if test="dormitory != null">did = #{dormitory.did}</if>

</set>

WHERE sid = #{sid};

</update>

1. 根据sid删除学生

<delete id="deleteStudent">

DELETE FROM student

WHERE

sid = #{sid};

</delete>

1. 根据did删除学生

<delete id="deleteStudentByDid">

DELETE FROM student

WHERE

did = #{did};

</delete>

1. 把did设置为null这个函数用于删除dormitory。

<update id="updateStudentDidToNull">

UPDATE student

SET

did = NULL

WHERE did=#{did};

</update>

Dao层：

这只是个接口，mapper与其一一映射：

public interface StudentDao {

List<Student> queryStudents();

Student queryStudentBySid(int sid);

List<Student> queryStudentByDid(int did);

int queryStudentCountByDid(int did);

int insertStudent(Student student);

int updateStudent(Student student);

int deleteStudent(int sid);

int deleteStudentByDid(int did);

int updateStudentDidToNull(int did);

}

Service层：

@Service

public class StudentServiceImpl implements StudentService {

@Autowired

private StudentDao studentDao;

@Override

public DormitoryResult queryStudents() {

List<Student> studentList = studentDao.queryStudents();

if (studentList != null && studentList.size() > 0) {

return DormitoryResult.ok(studentList);

} else {

return DormitoryResult.notFount("查询失败");

}

}

@Override

public DormitoryResult queryStudentBySid(int sid) {

if (sid <= 0) {

return DormitoryResult.notFount("查询学生条件错误");

} else {

Student student = studentDao.queryStudentBySid(sid);

if (student == null) {

return DormitoryResult.notFount("查询学生结果为空");

} else {

return DormitoryResult.ok(student);

}

}

}

@Override

public DormitoryResult queryStudentsByDid(int did) {

if (did <= 0) {

return DormitoryResult.notFount("查询学生条件错误");

} else {

List<Student> studentList = studentDao.queryStudentByDid(did);

if (studentList != null || studentList.size() > 0) {

return DormitoryResult.ok(studentList);

} else {

return DormitoryResult.notFount("查询学生结果为空");

}

}

}

@Override

public Map<String,Object> queryStudentCountByDid(int did) {

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

if (did <= 0) {

modleMap.put("status","404");

modleMap.put("number","0");

modleMap.put("msg","查询条件错误");

return modleMap;

} else {

int number = studentDao.queryStudentCountByDid(did);

if (number > 0) {

modleMap.put("number",number);

return modleMap;

} else {

modleMap.put("status","404");

modleMap.put("number","0");

modleMap.put("msg","查询结果为空");

return modleMap;

}

}

}

@Override

public DormitoryResult addStudent(Student student) {

if(student == null){

return DormitoryResult.notFount("插入学生为空");

}else{

try{

int effectedNum = studentDao.insertStudent(student);

if (effectedNum>0){

return DormitoryResult.ok();

}else {

return DormitoryResult.notFount("添加学生失败");

}

}catch (Exception e){

throw new DormitoryException("插入学生错误："+e.getMessage());

}

}

}

@Override

public DormitoryResult updateStudent(Student student) {

if(student == null){

return DormitoryResult.notFount("插入学生为空");

}else{

try{

int effectedNum = studentDao.updateStudent(student);

if (effectedNum>0){

return DormitoryResult.ok();

}else {

return DormitoryResult.notFount("修改学生失败");

}

}catch (Exception e){

throw new DormitoryException("修改学生错误："+e.getMessage());

}

}

}

@Override

public DormitoryResult deleteStudent(int sid) {

if(sid>0){

try{

int effectedNum = studentDao.deleteStudent(sid);

if(effectedNum>0){

return DormitoryResult.ok();

}

else{

return DormitoryResult.notFount("删除学生失败");

}

}catch (Exception e){

throw new DormitoryException("删除学生错误");

}

}

return DormitoryResult.notFount("删除学生条件错误");

}

}

Controller层：

@RestController

public class StudentController {

@Autowired

private StudentService studentService;

@GetMapping(value = "/querystudents")

@ResponseBody

public DormitoryResult queryStudents(){

DormitoryResult result = studentService.queryStudents();

return result;

}

@GetMapping(value = "/querystudentbysid/{sid}")

public DormitoryResult queryStudentBySid(@PathVariable("sid") int sid){

DormitoryResult result = studentService.queryStudentBySid(sid);

return result;

}

@GetMapping(value = "/querystudentbydid/{did}")

public DormitoryResult queryStudentByDid(@PathVariable("did") int did){

DormitoryResult result = studentService.queryStudentsByDid(did);

return result;

}

@GetMapping(value = "/querystudentcountbydid/{did}")

public Map<String,Object> queryStudentCountByDid(@PathVariable("did") int did){

Map<String,Object> result = studentService.queryStudentCountByDid(did);

return result;

}

@PostMapping(value = "/addstudent")

public DormitoryResult addStudent(HttpServletRequest request){

String studentStr = HttpServletRequestUtil.getString(request, "studentStr");

Student student = null;

ObjectMapper mapper = new ObjectMapper();

try {

student = mapper.readValue(studentStr, Student.class);

} catch (IOException e) {

e.printStackTrace();

}

DormitoryResult result = studentService.addStudent(student);

return result;

}

/\*@PostMapping(value = "/addstudent")

public DormitoryResult addStudent(Student student){

DormitoryResult result = studentService.addStudent(student);

return result;

}\*/

@PostMapping(value = "/updatestudent")

public DormitoryResult updateStudent(HttpServletRequest request){

String studentStr = HttpServletRequestUtil.getString(request, "studentStr");

Student student = null;

ObjectMapper mapper = new ObjectMapper();

try {

student = mapper.readValue(studentStr, Student.class);

} catch (IOException e) {

e.printStackTrace();

}

DormitoryResult result = studentService.updateStudent(student);

System.out.print(result);

return result;

}

/\*@PostMapping(value = "/updatestudent")

public DormitoryResult updateStudent(Student student){

DormitoryResult result = studentService.updateStudent(student);

return result;

}\*/

@GetMapping(value = "/deletestudent/{sid}")

DormitoryResult deleteStudent(@PathVariable("sid") int sid){

DormitoryResult result = studentService.deleteStudent(sid);

return result;

}

}

##### 5.2 building功能

Mapper：

1. 查询building列表

<select id="queryBuildings" resultType="com.youyi.dormitory.entity.Building">

SELECT b.bid,b.bname,b.place

FROM

building b

ORDER BY bid;

</select>

1. 根据bid查询building

<select id="queryBuildingById" resultType="com.youyi.dormitory.entity.Building" parameterType="int">

SELECT b.bid,b.bname,b.place

FROM

building b

WHERE b.bid = #{bid}

</select>

1. 插入building

<insert id="insertBuilding" parameterType="com.youyi.dormitory.entity.Building" useGeneratedKeys="true" keyProperty="bid" keyColumn="bid">

INSERT INTO

building(bname,place)

VALUES

(#{bname},#{place});

</insert>

1. 更新building

<update id="updateBuilding" parameterType="com.youyi.dormitory.entity.Building">

UPDATE building

<set>

<if test="bname != null">bname = #{bname},</if>

<if test="place != null">place = #{place}</if>

</set>

WHERE bid = #{bid};

</update>

1. 删除building

<delete id="deleteBuilding">

DELETE FROM building

WHERE

bid = #{bid};

</delete>

Dao层：

是个接口，与mapper一一映射

Service层：

@Service

public class BuildingServiceImpl implements BuildingService{

@Autowired

private BuildingDao buildingDao;

@Autowired

private DormitoryService dormitoryService;

@Override

public DormitoryResult queryBuildingById(int bid) {

if(bid<1){

return DormitoryResult.notFount("查询条件错误");

}

Building building = buildingDao.queryBuildingById(bid);

//判断是否查询到结果

if (building != null) {

return DormitoryResult.ok(building);

}

return DormitoryResult.notFount("查询失败");

}

@Override

public DormitoryResult queryBuildings() {

List<Building> buildingList = buildingDao.queryBuildings();

if(buildingList != null && buildingList.size()>0){

return DormitoryResult.ok(buildingList);

}else {

return DormitoryResult.notFount("查询失败");

}

}

@Override

public DormitoryResult addBuilding(Building building){

if(building == null){

return DormitoryResult.notFount("添加内容为空");

}

try{

int effectedNum = buildingDao.insertBuilding(building);

if(effectedNum <0){

return DormitoryResult.notFount("添加building失败");

}

return DormitoryResult.ok();

}catch (Exception e){

throw new DormitoryException("添加building失败:"+e.getMessage());

}

}

@Override

public DormitoryResult modifyBuilding(Building building){

if(building != null){

try{

int effectedNum = buildingDao.updateBuilding(building);

if(effectedNum<0){

return DormitoryResult.notFount("修改building失败");

}

return DormitoryResult.ok();

}catch (Exception e){

throw new DormitoryException("添加building失败:"+e.getMessage());

}

}

return DormitoryResult.notFount("修改内容为空");

}

@Override

public DormitoryResult deleteBuilding(int bid){

if(bid <1){

return DormitoryResult.notFount("删除内容错误");

}

try{

int effectedNum = buildingDao.deleteBuilding(bid);

if(effectedNum<0) {

return DormitoryResult.notFount("删除building失败");

}else{

return DormitoryResult.ok();

}

}catch (Exception e){

throw new DormitoryException("删除building异常："+e.getMessage());

}

}

}

Controller层：

@RestController

public class BuildingController {

@Autowired

private BuildingService buildingService;

@GetMapping(value = "/querybuildingbyid/{bid}")

public DormitoryResult queryBuildingById(@PathVariable("bid") int bid){

DormitoryResult result = buildingService.queryBuildingById(bid);

return result;

}

@GetMapping(value = "/querybuildings")

public DormitoryResult queryBuildings(){

DormitoryResult result = buildingService.queryBuildings();

return result;

}

@PostMapping(value = "/addbuilding")

public DormitoryResult addBuilding(HttpServletRequest request) {

String studentStr = HttpServletRequestUtil.getString(request, "buildingStr");

Building building = null;

ObjectMapper mapper = new ObjectMapper();

try {

building = mapper.readValue(studentStr, Building.class);

} catch (IOException e) {

e.printStackTrace();

}

DormitoryResult result = buildingService.addBuilding(building);

return result;

}

@PostMapping(value = "/modifybuilding")

public DormitoryResult modifyBuilding(HttpServletRequest request){

String studentStr = HttpServletRequestUtil.getString(request, "buildingStr");

Building building = null;

ObjectMapper mapper = new ObjectMapper();

try {

building = mapper.readValue(studentStr, Building.class);

} catch (IOException e) {

e.printStackTrace();

}

DormitoryResult result = buildingService.modifyBuilding(building);

return result;

}

@GetMapping(value = "/deletebuilding/{bid}")

public DormitoryResult deleteBuilding(@PathVariable("bid") int bid){

DormitoryResult result = buildingService.deleteBuilding(bid);

return result;

}

}

##### 5.3 dormitory功能

Mapper：

1. 查询dormitory列表

<select id="queryDormitories" resultMap="dormitoryMap">

SELECT d.did,d.num,d.max\_man,b.bid,b.bname,b.place

FROM

dormitory d,building b

WHERE d.bid = b.bid

ORDER BY did;

</select>

1. 根据did查询dormitory列表

<select id="queryDormitoryById" resultMap="dormitoryMap" parameterType="int">

SELECT d.did,d.num,d.max\_man,b.bid,b.bname,b.place

FROM

dormitory d,building b

WHERE d.bid = b.bid

AND d.did = #{did}

ORDER BY did;

</select>

1. 插入dormitory

<insert id="insertDormitory" useGeneratedKeys="true" parameterType="com.youyi.dormitory.entity.Dormitory"

keyProperty="did" keyColumn="did">

INSERT INTO

dormitory(bid,num,max\_man)

VALUES

(#{building.bid},#{num},#{maxMan});

</insert>

1. 更新dormitory

<update id="updateDormitory" parameterType="com.youyi.dormitory.entity.Dormitory">

UPDATE dormitory

<set>

<if test="building != null">bid = #{building.bid},</if>

<if test="num != null">num = #{num},</if>

<if test="maxMan != null">max\_man = #{maxMan}</if>

</set>

WHERE did = #{did};

</update>

1. 删除dormitory

<delete id="deleteDormitory">

DELETE FROM dormitory

WHERE

did = #{did};

</delete>

Dao层：

这只是个接口，与mapper映射，方便service层调用

Service层：

@Service

public class DormitoryServiceImpl implements DormitoryService{

@Autowired

private DormitoryDao dormitoryDao;

@Autowired

private StudentDao studentDao;

@Override

public DormitoryResult queryDormitoryById(int did) {

if(did<=0){

return DormitoryResult.notFount("查询宿舍条件错误");

}

Dormitory dormitory = dormitoryDao.queryDormitoryById(did);

if(dormitory == null){

return DormitoryResult.notFount("查询宿舍结果为空");

}

return DormitoryResult.ok(dormitory);

}

@Override

public DormitoryResult queryDormitories() {

List<Dormitory> dormitoryList = dormitoryDao.queryDormitories();

if(dormitoryList != null && dormitoryList.size()>0){

return DormitoryResult.ok(dormitoryList);

}else {

return DormitoryResult.notFount("查询宿舍结果为空");

}

}

@Override

public DormitoryResult addDormitory(Dormitory dormitory){

if(dormitory != null){

try{

int effectedNum = dormitoryDao.insertDormitory(dormitory);

if(effectedNum>0){

return DormitoryResult.ok();

}

else{

return DormitoryResult.notFount("添加宿舍失败");

}

}catch (Exception e){

throw new DormitoryException("添加宿舍错误："+e.getMessage());

}

}

else {

return DormitoryResult.notFount("添加宿舍内容不能为空");

}

}

@Override

public DormitoryResult modifyDormitory(Dormitory dormitory) {

if(dormitory != null){

try{

int effectedNum = dormitoryDao.updateDormitory(dormitory);

if(effectedNum>0){

return DormitoryResult.ok();

}

else{

return DormitoryResult.notFount("修改宿舍失败");

}

}catch (Exception e) {

throw new DormitoryException("修改宿舍错误：" + e.getMessage());

}

}

else {

return DormitoryResult.notFount("宿舍未修改");

}

}

@Override

@Transactional

public DormitoryResult deleteDormitory(int did) {

if(did>0){

//将学生对应宿舍为did的置为null

try {

int effected = studentDao.updateStudentDidToNull(did);

if(effected<=0) {

return DormitoryResult.notFount("重置学生宿舍失败");

}

}catch (Exception e) {

throw new DormitoryException("重置学生宿舍异常："+e.getMessage());

}

//删除宿舍

try {

int effectedNum = dormitoryDao.deleteDormitory(did);

if(effectedNum>0){

return DormitoryResult.ok();

}

else {

return DormitoryResult.notFount("删除宿舍失败");

}

}catch (Exception e) {

throw new DormitoryException("删除宿舍异常："+e.getMessage());

}

}

else {

return DormitoryResult.notFount("删除宿舍条件不能为空");

}

}

}

Controller层：

@RestController

public class DormitoryController {

@Autowired

private DormitoryService dormitoryService;

@GetMapping(value = "/querydormitories")

public DormitoryResult queryDormitories(){

DormitoryResult result = dormitoryService.queryDormitories();

return result;

}

@GetMapping(value = "/querydormitorybyid/{did}")

public DormitoryResult queryDormitoryById(@PathVariable("did") int did){

DormitoryResult result = dormitoryService.queryDormitoryById(did);

return result;

}

@PostMapping(value = "/adddormitory")

public DormitoryResult addDormitory(HttpServletRequest request){

String dormitoryStr = HttpServletRequestUtil.getString(request, "dormitoryStr");

Dormitory dormitory = null;

ObjectMapper mapper = new ObjectMapper();

try {

dormitory = mapper.readValue(dormitoryStr, Dormitory.class);

} catch (IOException e) {

e.printStackTrace();

}

DormitoryResult result = dormitoryService.addDormitory(dormitory);

return result;

}

@PostMapping(value = "/modifydormitory")

public DormitoryResult modifyDomitory(HttpServletRequest request){

String dormitoryStr = HttpServletRequestUtil.getString(request, "dormitoryStr");

Dormitory dormitory = null;

ObjectMapper mapper = new ObjectMapper();

try {

dormitory = mapper.readValue(dormitoryStr, Dormitory.class);

} catch (IOException e) {

e.printStackTrace();

}

DormitoryResult result = dormitoryService.modifyDormitory(dormitory);

return result;

}

@GetMapping(value = "/deletedormitory/{did}")

public DormitoryResult deleteDormitory(@PathVariable("did") int did){

DormitoryResult result = dormitoryService.deleteDormitory(did);

return result;

}

}

##### 5.4 zszt功能

Mapper：

1. 查询zszt列表

<select id="queryZszts" resultMap="zsztMap">

SELECT z.zid,z.date,z.zt,d.did,d.num,d.max\_man,b.bid,b.bname,b.place

FROM

zszt z,dormitory d,building b

WHERE d.did = z.did

AND d.bid=b.bid

ORDER BY zid;

</select>

1. 根据zid查询zszt

<select id="queryZsztByZid" resultMap="zsztMap">

SELECT z.zid,z.date,z.zt,d.did,d.num,d.max\_man,b.bid,b.bname,b.place

FROM

zszt z,dormitory d,building b

WHERE z.did = d.did

AND

d.bid = b.bid

AND

z.zid = #{zid}

ORDER BY zid;

</select>

1. 根据did查询zszt

<select id="queryZsztByDid" resultMap="zsztMap">

SELECT z.zid,z.date,z.zt,d.did,d.num,d.max\_man,b.bid,b.bname,b.place

FROM

zszt z,dormitory d,building b

WHERE z.did = d.did

AND

d.bid = b.bid

AND

z.did = #{did}

ORDER BY zid;

</select>

1. 插入zszt

<insert id="insertZszt" parameterType="com.youyi.dormitory.entity.Zszt" useGeneratedKeys="true" keyProperty="zid" keyColumn="zid">

INSERT INTO

zszt(did,date,zt)

VALUES

(#{dormitory.did},#{date},#{zt});

</insert>

1. 更新zszt

<update id="updateZszt" parameterType="com.youyi.dormitory.entity.Zszt">

UPDATE zszt

<set>

<if test="dormitory != null">did = #{dormitory.did},</if>

<if test="date != null">date = #{date},</if>

<if test="zt != null">zt = #{zt}</if>

</set>

WHERE zid = #{zid};

</update>

1. 根据zid删除zszt

<delete id="deleteZsztByZid">

DELETE FROM zszt

WHERE

zid = #{zid};

</delete>

Dao层：

这是个接口，与mapper映射，方便service层调用

Service层：

@Service

public class ZsztServiceImpl implements ZsztService{

@Autowired

private ZsztDao zsztDao;

@Override

public DormitoryResult queryZszts() {

List<Zszt> zsztList =zsztDao.queryZszts();

if(zsztList != null && zsztList.size()>0){

return DormitoryResult.ok(zsztList);

}else{

return DormitoryResult.notFount("查询住宿情况为空");

}

}

@Override

public DormitoryResult queryZsztByZid(int zid) {

if(zid >0){

Zszt zszt = zsztDao.queryZsztByZid(zid);

if(zszt != null){

return DormitoryResult.ok(zszt);

}else{

throw new DormitoryException("查询住宿情况失败");

}

}

return DormitoryResult.notFount("查询住宿条件错误");

}

@Override

public DormitoryResult queryZsztByDate(Date date) {

if(date != null){

List<Zszt> zsztList = zsztDao.queryZsztByDate(date);

if(zsztList != null){

return DormitoryResult.ok(zsztList);

}else{

throw new DormitoryException("查询住宿情况失败");

}

}

return DormitoryResult.notFount("查询住宿日期错误");

}

@Override

public DormitoryResult queryZsztByDid(int did) {

if(did >0){

List<Zszt> zsztList = zsztDao.queryZsztByDid(did);

if(zsztList != null){

return DormitoryResult.ok(zsztList);

}else{

throw new DormitoryException("查询住宿情况失败");

}

}

return DormitoryResult.notFount("查询住宿条件错误");

}

@Override

public DormitoryResult addZszt(Zszt zszt) {

if(zszt != null){

try{

int effectedNum = zsztDao.insertZszt(zszt);

if(effectedNum>0){

return DormitoryResult.ok();

}

return DormitoryResult.notFount("插入住宿情况失败");

}catch (Exception e){

throw new DormitoryException("插入住宿情况异常："+e.getMessage());

}

}

return DormitoryResult.notFount("插入住宿情况为空");

}

@Override

public DormitoryResult modifyZszt(Zszt zszt) {

if(zszt != null){

try{

int effectedNum = zsztDao.updateZszt(zszt

);

if(effectedNum>0){

return DormitoryResult.ok();

}

return DormitoryResult.notFount("更新住宿情况失败");

}catch (Exception e){

throw new DormitoryException("更新住宿情况异常："+e.getMessage());

}

}

return DormitoryResult.notFount("更新住宿情况为空");

}

@Override

public DormitoryResult deleteZsztByZid(int zid) {

if(zid>0){

try{

int effectedNum = zsztDao.deleteZsztByZid(zid);

if(effectedNum>0){

return DormitoryResult.ok();

}

return DormitoryResult.notFount("删除住宿情况失败");

}

catch (Exception e){

throw new DormitoryException("删除住宿情况异常："+e.getMessage());

}

}

return DormitoryResult.notFount("删除住宿情况参数为空");

}

@Override

public DormitoryResult deleteZsztByDid(int did) {

if(did>0){

try{

int effectedNum = zsztDao.deleteZsztByDid(did);

if(effectedNum>0){

return DormitoryResult.ok();

}

return DormitoryResult.notFount("删除住宿情况失败");

}

catch (Exception e){

throw new DormitoryException("删除住宿情况异常："+e.getMessage());

}

}

return DormitoryResult.notFount("删除住宿情况参数为空");

}

@Override

public DormitoryResult deleteZsztByDate(Date date) {

if(date != null){

try{

int effectedNum = zsztDao.deleteZsztByDate(date);

if(effectedNum>0){

return DormitoryResult.ok();

}

return DormitoryResult.notFount("删除住宿情况失败");

}

catch (Exception e){

throw new DormitoryException("删除住宿情况异常："+e.getMessage());

}

}

return DormitoryResult.notFount("删除住宿情况参数为空"); }

}

Controller层：

@RestController

public class ZsztController {

@Autowired

private ZsztService zsztService;

@GetMapping(value = "/queryzszts")

public DormitoryResult queryZszts(){

DormitoryResult result = zsztService.queryZszts();

return result;

}

@GetMapping(value = "/queryzsztbyzid/{zid}")

public DormitoryResult queryZsztByZid(@PathVariable("zid") int zid){

DormitoryResult result = zsztService.queryZsztByZid(zid);

return result;

}

@GetMapping(value = "/queryzsztbydate/{date}")

public DormitoryResult queryZsztByDate(@PathVariable("date") String date) throws ParseException {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date thedate = sdf.parse(date);

DormitoryResult result = zsztService.queryZsztByDate(thedate);

return result;

}

@GetMapping(value = "/queryzsztbydid/{did}")

public DormitoryResult queryZsztByDid(@PathVariable("did") int did){

DormitoryResult result = zsztService.queryZsztByDid(did);

return result;

}

@PostMapping(value = "/addzszt")

public DormitoryResult addZszt(HttpServletRequest request){

String zsztStr = HttpServletRequestUtil.getString(request, "zsztStr");

Zszt zszt = null;

ObjectMapper mapper = new ObjectMapper();

try {

zszt = mapper.readValue(zsztStr, Zszt.class);

} catch (IOException e) {

e.printStackTrace();

}

DormitoryResult result = zsztService.addZszt(zszt);

return result;

}

@PostMapping(value = "/modifyzszt")

public DormitoryResult modifyZszt(HttpServletRequest request){

String zsztStr = HttpServletRequestUtil.getString(request, "zsztStr");

Zszt zszt = null;

ObjectMapper mapper = new ObjectMapper();

try {

zszt = mapper.readValue(zsztStr, Zszt.class);

} catch (IOException e) {

e.printStackTrace();

}

DormitoryResult result = zsztService.modifyZszt(zszt);

return result;

}

@GetMapping(value = "/deletezsztbyzid/{zid}")

public DormitoryResult deleteZsztByZid(@PathVariable("zid") int zid){

DormitoryResult result = zsztService.deleteZsztByZid(zid);

return result;

}

@GetMapping(value = "/deletezsztbydid/{did}")

public DormitoryResult deleteZsztByDid(@PathVariable("did") int did){

DormitoryResult result = zsztService.deleteZsztByDid(did);

return result;

}

@GetMapping(value = "/deletezsztbydate/{date}")

public DormitoryResult deleteZsztByDate(@PathVariable("date") String date) throws ParseException {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date thedate = sdf.parse(date);

DormitoryResult result = zsztService.deleteZsztByDate(thedate);

return result;

}

}

#### 界面设计

##### 6.1登陆界面



##### 6.2 student的两个界面

studentlist界面



studentoperation界面



##### 6.3 building的两个界面

buildinglist界面



buildingoperation界面



##### 6.4 dormitory的两个界面

dormitorylist界面

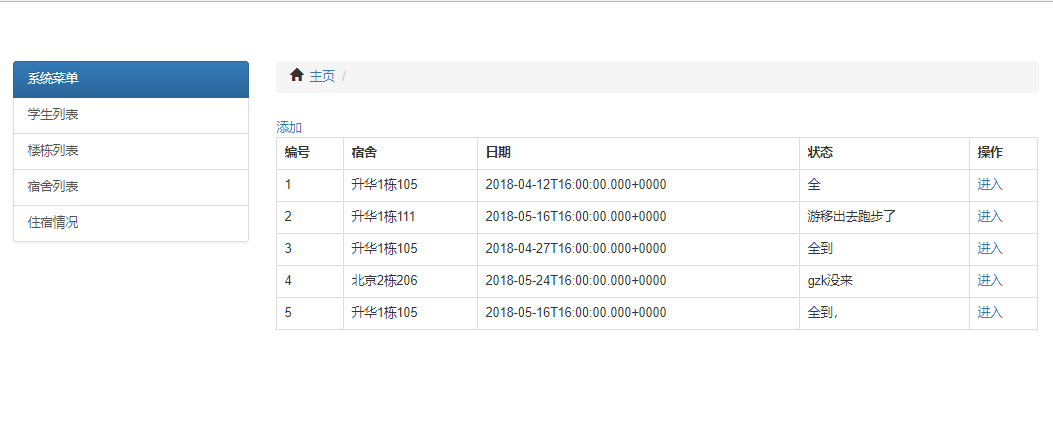


dormitoryoperation界面



##### 6.5 zszt的两个界面

zsztlist界面



zsztoperation界面



#### 结束语

通过本次实验我对springboot框架更加熟悉，学会了怎样用mybatis。其中最重要的application.yml文件的配置，springboot把其他的配置文件都简化了，只需要开发人员自己配置一个文件，其他大多数都可以采用注解的方式实现。

关于mybatis，我熟悉了在mybatis中写级联查询，resultMap的编写，以及各种sql语句。

关于数据库方面，深度了解了数据库设计的问题，譬如ER图的设计，然后需要满足第三范式。

然后在前端方面，我熟悉了ajax异步请求，动态生成h5代码，了解了前端的基础开发和bootstrap的简单应用。

本次课程设计总的来说使我获益良多，复习了javaweb的开发和前端的开发，又巩固了数据库的知识，学习了springboot框架的使用。

##### 参考文献

1. Eric. Java编程思想[M]. 第4版. 机械工业出版社
2. 杜波依斯. MySQL技术内幕[M]. 第4版. 人民邮电出版社
3. 林信良. JSP & Servlet学习笔记[M]. 清华大学出版社
4. 孙卫琴. Tomcat与Java Web开发技术详解[M]. 第2版. 电子工业
5. 毕建信.基于MVC设计模式的Web应用研究与实现[D].武汉:武汉理工大学,2006年5月