测试对象：SalaryService类

测试人员：涂成武

测试日期：2020-05-09

1. 测试对象描述

该类为service层的一个业务类，负责调用mapper层mybatis的接口进行数据的crud操作。该类对应的数据库表为salary表。

该类含有如下方法：

1. getAllSalaries()：该方法查询salary表中所有的工资账套，以List<Salary>的形式进行返回。
2. addSalary()：该方法接受参数为一个Salary类型的变量，将其插入数据库表中，若成功返回1，若失败则返回0。
3. deleteSalaryById()：该方法接受Integer类型的一个变量指示需要删除的记录的主键，然后进行删除，返回值为删除的记录数。
4. updateSalarById()：该方法接受参数为一个Salary类型的变量，然后以参数->Id为主键，将变量的非空属性更新到表的对应记录中，返回值为更新的记录数。
5. 测试用例设计
6. 对于getAllSalaries方法，该方法无参数，故只需要一个测试用例来测试其功能是否正确即可，即手动构造一个List<Salary>其中包含数据库中的所有记录，然后将其与该方法的返回值进行比较即可
7. 对于addSalary方法，该方法接受一个Salary类型的变量，对于该方法我们设计了如下四个测试用例：
8. 重复主键：

参数为：Salary toInsert = *new* Salary();

toInsert.setId(13);

预期结果为：抛出org.springframework.dao.DuplicateKeyException类型异常

1. 全字段为空：

参数为：toInsert= new Salary();

预期结果为：返回1(因为数据库设计时指定主键自增)

1. 除主键外与某记录完全相同：

参数为：Salary ele3=*new* Salary();  
ele3.setId(20);  
ele3.setBasicSalary(20000);  
ele3.setBonus(3000);  
ele3.setLunchSalary(500);  
ele3.setTrafficSalary(500);  
ele3.setPensionBase(4000);  
ele3.setPensionPer((*float*) 0.07);  
ele3.setCreateDate(*new* Date(118, 0,25,0,0,0));  
ele3.setMedicalBase(4000);  
ele3.setMedicalPer((*float*) 0.07);  
ele3.setAccumulationFundBase(4000);  
ele3.setAccumulationFundPer((*float*) 0.07);  
ele3.setName("运维部工资账套");

预期结果为：返回1

1. 普通有效：

参数为：ele3.setId(200);

ele3.setBasicSalary(23000);

ele3.setBonus(30001);

ele3.setLunchSalary(5100);

ele3.setTrafficSalary(5000);

ele3.setPensionBase(40000);

ele3.setPensionPer((float) 0.09);

ele3.setCreateDate(new Date(128, 0,25,0,0,0));

ele3.setMedicalBase(4001);

ele3.setMedicalPer((float) 0.07);

ele3.setAccumulationFundBase(40030);

ele3.setAccumulationFundPer((float) 0.07);

ele3.setName("运维部工资账套");

预期结果为：返回1

1. 对于deleteSalaryById方法，设计出如下4个测试用例
   1. 主键为正数，且不存在数据库中：

参数为：15

预期结果为：返回0

* 1. 主键为正数，且存在数据库中：

参数为：14

预期结果为：返回1

* 1. 主键为负数，且存在数据库中：

参数为：-1

预期结果为：返回1

* 1. 主键为负数，且不存在数据库中

参数为：-2

预期结果为：返回0

1. 对于updateSalaryById方法，设计出如下4个测试用例
   1. 待更新的记录的主键存在于数据库中：

参数为：Salary ele3 = new Salary();

ele3.setId(14);

ele3.setBonus(30001);

ele3.setLunchSalary(5100);

ele3.setTrafficSalary(5000);

ele3.setPensionBase(40000);

ele3.setPensionPer((float) 0.09);

ele3.setCreateDate(new Date(128, 0,25,0,0,0));

ele3.setMedicalBase(4001);

ele3.setMedicalPer((float) 0.07);

ele3.setAccumulationFundBase(40030);

ele3.setAccumulationFundPer((float) 0.07);

ele3.setName("运维部工资账套");

预期结果为：返回1

* 1. 待更新的记录的主键存在于数据库中，但除主键外其余属性均为空

参数为：Salary ele4 = new Salary();

ele4.setId(14);

预期结果为：抛出org.springframework.jdbc.BadSqlGrammarException类型异常

* 1. 待更新的记录主键不存在于数据库中，且属性不均为空

参数为：Salary ele5 = new Salary();

ele5.setId(16);

ele5.setBonus(30001);

ele5.setLunchSalary(5100);

ele5.setTrafficSalary(5000);

ele5.setPensionBase(40000);

ele5.setPensionPer((float) 0.09);

ele5.setCreateDate(new Date(128, 0,25,0,0,0));

ele5.setMedicalBase(4001);

ele5.setMedicalPer((float) 0.07);

ele5.setAccumulationFundBase(40030);

ele5.setAccumulationFundPer((float) 0.07);

ele5.setName("运维部工资账套");

预期结果为：返回0

* 1. 待更新记录不存在数据库中，且除主键外其余属性均为空：

参数为：Salary ele6=new Salary();

ele6.setId(18);

预期结果为：抛出org.springframework.jdbc.BadSqlGrammarException类型异常

1. 测试用例执行与结果：
2. getAllSalaries方法：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试模块/对象 | | org.javaboy.vhr.service.SalaryService#getAllSalaries | | | | |
| 是否有测试代码 | | 有 | 测试代码 | org.javaboy.vhr.service.SalaryServiceTest  #getAllSalaries | | |
| 前提条件（如有） | | 无 | | | | |
| 测试用例序号 | 预期结果 | | | | 实际结果 | 备注 |
| 1 | 包含所有Salary的List | | | | 包含所有Salary的List | FailNotEqual |

该测试失败

1. addSalary方法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试模块/对象 | | org.javaboy.vhr.service.SalaryService#addSalary | | | | |
| 是否有测试代码 | | 有 | 测试代码 | org.javaboy.vhr.service.SalaryServiceTest  #addSalary | | |
| 前提条件（如有） | | 无 | | | | |
| 测试用例序号 | 预期结果 | | | | 实际结果 | 备注 |
| 1 | DuplicateKeyException | | | | DuplicateKeyException |  |
| 2 | 1 | | | | 1 |  |
| 3 | 1 | | | | 1 |  |
| 4 | 1 | | | | 1 |  |

1. deleteSalaryById方法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试模块/对象 | | org.javaboy.vhr.service.SalaryService#deleteSalaryById | | | | |
| 是否有测试代码 | | 有 | 测试代码 | org.javaboy.vhr.service.SalaryServiceTest  #deleteSalaryById | | |
| 前提条件（如有） | | 在执行测试用例之前需要在数据库Salary表中添加一条主键为-1的记录 | | | | |
| 测试用例序号 | 预期结果 | | | | 实际结果 | 备注 |
| 1 | 0 | | | | 0 |  |
| 2 | 1 | | | | 1 |  |
| 3 | 1 | | | | 1 |  |
| 4 | 0 | | | | 0 |  |

1. updateSalaryById方法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试模块/对象 | | org.javaboy.vhr.service.SalaryService#updateSalaryById | | | | |
| 是否有测试代码 | | 有 | 测试代码 | org.javaboy.vhr.service.SalaryServiceTest  #updateSalaryById | | |
| 前提条件（如有） | | 无 | | | | |
| 测试用例序号 | 预期结果 | | | | 实际结果 | 备注 |
| 1 | 1 | | | | 1 |  |
| 2 | BadSqlGrammarException | | | | BadSqlGrammarException |  |
| 3 | 0 | | | | 0 |  |
| 4 | BadSqlGrammarException | | | | BadSqlGrammarException |  |

1. failure原因分析与追踪

在测试getAllSalaries方法时，测试用例执行失败，在使用idea提供的junit对比工具对比之后，预期结果与实际结果并没有区别，在查阅相关资料后发现，导致该failure的原因是实体类Salary没有实现equals方法和hashcode方法，导致无法判断两个Salary类型的变量是否相等，导致Assert.assertEquals方法失败。

Bug修复：为Salary类添加equals与hashcode方法如下：

@Override  
*public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 Salary salary = (Salary) o;  
 *return* Objects.*equals*(id, salary.id) &&  
 Objects.*equals*(basicSalary, salary.basicSalary) &&  
 Objects.*equals*(bonus, salary.bonus) &&  
 Objects.*equals*(lunchSalary, salary.lunchSalary) &&  
 Objects.*equals*(trafficSalary, salary.trafficSalary) &&  
 Objects.*equals*(allSalary, salary.allSalary) &&  
 Objects.*equals*(pensionBase, salary.pensionBase) &&  
 Objects.*equals*(pensionPer, salary.pensionPer) &&  
 Objects.*equals*(createDate, salary.createDate) &&  
 Objects.*equals*(medicalBase, salary.medicalBase) &&  
 Objects.*equals*(medicalPer, salary.medicalPer) &&  
 Objects.*equals*(accumulationFundBase, salary.accumulationFundBase) &&  
 Objects.*equals*(accumulationFundPer, salary.accumulationFundPer) &&  
 Objects.*equals*(name, salary.name);  
}  
  
@Override  
*public int* hashCode() {  
 *return* Objects.*hash*(id, basicSalary, bonus, lunchSalary, trafficSalary, allSalary, pensionBase, pensionPer, createDate, medicalBase, medicalPer, accumulationFundBase, accumulationFundPer, name);  
}

再次执行测试用例，测试通过。

1. 测试结果分析：

根据上表与后续的修改，我们可以看出实际结果与预期结果完全一致，说明该类的功能符合我们对其的预期，表明该单元通过了单元测试。