EmployeeMapper集成测试报告

测试人：巩昊东

测试时间：2020/6/7

一，测试对象及测试策略

该类是Employee在model上的映射，为service提供适当的curd操作接口，映射的表为employee表。

采用自底向上的集成测试策略，此处的集成测试策略的基础在于对于底层模块的单元测试（EmployeeModel类，以及dateConverter类，详见两类的单元测试文档。）成功后的基础上，本集成测试成功后同样作为其他测试的测试基础。

在测试用例设计上，考虑到此处逻辑的特点，统一使用等价类法，通过每个方法的具体情况在其缺陷假设和异常区域中做出测试的选择。

其方法如下：

1.int deleteByPrimaryKey(Integer id);

按照主键删除一个条目，返回删除的数量（1或0）

2.int insert(Employee record);

插入条目，返回插入的数量（1或0）

3.int insertSelective(Employee record);

选择性插入一条记录，返回插入的数量（1或0）

4.Employee selectByPrimaryKey(Integer id);

通过主键查询一条记录，返回查询到的记录

5.int updateByPrimaryKeySelective(Employee record);

根据主键选择性更新一条记录，返回更新的数量（1或0）

6.int updateByPrimaryKey(Employee record);

根据主键更新一条记录，返回更新的数量（1或0）

7.List<Employee> getEmployeeByPage(@Param("page") Integer page, @Param("size") Integer size, @Param("emp") Employee employee,@Param("beginDateScope") Date[] beginDateScope);

通过分页查询信息，参数分别为显示第几页的信息，每一页多少条，入职起止时间，详细查询条件，返回结果的列表

8.Long getTotal(@Param("emp") Employee employee,@Param("beginDateScope") Date[] beginDateScope);

查询满足要求的记录数量。返回数量。

9.Integer maxWorkID();

返回表中记录的workID的最大值。

10.Integer addEmps(@Param("list") List<Employee> list);

批量添加成员，返回添加成功的数量。

11.Employee getEmployeeById(Integer id);

通过ID查询成员，返回成员

12.List<Employee> getEmployeeByPageWithSalary(@Param("page") Integer page, @Param("size") Integer size);

分页查询员工及其薪资信息，不设查询条件，该方法接受的参数有两个，分别为页数，页大小，返回结果列表。

13.Integer updateEmployeeSalaryById(@Param("eid") Integer eid, @Param("sid") Integer sid);

更新empsalary中的记录，当原本的记录不存在时新建记录，当原本记录存在时删除原纪录新建另一条，返回值为涉及的记录条数，即不存在时返回1，存在时返回2。

二，测试用例设计

1. int deleteByPrimaryKey(Integer id);

该方法接受一个interger id，通过等价类方式设计两个用例，分别为有效id和无效id，测试如下：

|  |  |  |
| --- | --- | --- |
| 接口 | int deleteByPrimaryKey(Integer id); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 | 已知的有效id和无效id | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 有效id | 删除记录，返回1 | 删除记录，返回1 |
| 无效id | 返回0 | 返回0 |
|  |  |  |

2. int insert(Employee record);

根据实际的业务逻辑，设计3个用例，如下

|  |  |  |
| --- | --- | --- |
| 接口 | int insert(Employee record); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 | 无 | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 正常填充数据（见附表，下同） | 1 | 1 |
| 部分填充数据 | 1 | 1 |
| 空的Employee类 | 1 | 1 |

3.int insertSelective(Employee record);

预期的行为应当是“如果有空值就插入默认值”，结合这一方法的特点，因此设计两个等价类用例如下

|  |  |  |
| --- | --- | --- |
| 接口 | int insertSelective(Employee record); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 正常填充数据 | 1 | 1 |
| 部分填充数据 | 1，空值为默认值 | 1，空值为默认值 |
|  |  |  |

4.Employee selectByPrimaryKey(Integer id);

|  |  |  |
| --- | --- | --- |
| 接口 | Employee selectByPrimaryKey(Integer id); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 | 已知的有效id和无效id | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 有效id | 相应的的数据 | 相应的数据 |
| 无效id | Null | Null |
|  |  |  |

5.int updateByPrimaryKeySelective(Employee record);

|  |  |  |
| --- | --- | --- |
| 接口 | int updateByPrimaryKeySelective(Employee record); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 完整数据 | 1 | 1 |
| 部分数据 | 1，为null的字段未改变 | 1，为null的字段未改变 |
|  |  |  |

6.int updateByPrimaryKey(Employee record);

|  |  |  |
| --- | --- | --- |
| 接口 | int updateByPrimaryKey(Employee record); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 完整数据 | 1 | 1 |
| 部分数据 | 1，缺失部分为null | 1，缺失部分为null |
|  |  |  |

7.List<Employee> getEmployeeByPage(@Param("page") Integer page, @Param("size") Integer size, @Param("emp") Employee employee,@Param("beginDateScope") Date[] beginDateScope);

|  |  |  |
| --- | --- | --- |
| 接口 | List<Employee> getEmployeeByPage(@Param("page") Integer page, @Param("size") Integer size, @Param("emp") Employee employee,@Param("beginDateScope") Date[] beginDateScope); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| Page=1,size=fullsize（见附表）,无条件 | Fullsize | Fullsize |
| Page=2，size=fullsize，无条件 | 0 | 0 |
| Page=-1 | BadSqlGrammarException | BadSqlGrammarException |
| Page=1,size=10,无条件 | 10 | 10 |
| Page=1，size=10，姓名为“江南” | 10 | 10 |
| Page=25，size=20，劳动合同 | 2 | 2 |
| Page=12，size=15，politicID=3 | 3 | 3 |
| Page=18，size=10，joblevelID=12 | 4 | 4 |
| Page=7，size=8，departmentID=12 | 6 | 6 |
| Page=13，size=12，posID=29 | 2 | 2 |
| Page=31，size=15，日期为  2017-12-15，2018-01-05 | 3 | 3 |

8.Long getTotal(@Param("emp") Employee employee,@Param("beginDateScope") Date[] beginDateScope);

|  |  |  |
| --- | --- | --- |
| 接口 | Long getTotal(@Param("emp") Employee employee,@Param("beginDateScope") Date[] beginDateScope); | |
| 调用关系和描述 |  | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 空employee和null | 633 | 633 |
| 空employee和”2017-12-15”,2018-01-05 | 453 | 453 |
| 插入部分空置数据并查询 | 11 | 11 |

9.Integer maxWorkID();

|  |  |  |
| --- | --- | --- |
| 接口 | Integer maxWorkID(); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 | 已知maxWorkID为65（初始设定） | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 无 | 65 | 65 |
|  |  |  |
|  |  |  |

10.Integer addEmps(@Param("list") List<Employee> list);

|  |  |  |
| --- | --- | --- |
| 接口 | Integer addEmps(@Param("list") List<Employee> list); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 长度2成员列表 | 2 | 2 |
| 空列表 | 0 | 0 |
|  |  |  |

11.Employee getEmployeeById(Integer id);

|  |  |  |
| --- | --- | --- |
| 接口 | Employee getEmployeeById(Integer id); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 有效id | 返回有效id对应的数据 | 返回有效id对应的数据 |
| 无效id | Null | Null |
|  |  |  |

12.List<Employee> getEmployeeByPageWithSalary(@Param("page") Integer page, @Param("size") Integer size);

|  |  |  |
| --- | --- | --- |
| 接口 | List<Employee> getEmployeeByPageWithSalary(@Param("page") Integer page, @Param("size") Integer size); | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| Page=1，size=总量 | 总量 | 总量 |
| Page=2，size=总量 | 总量 | 总量 |
| Page=-1 | BadSqlGrammarException | BadSqlGrammarException |

13.Integer updateEmployeeSalaryById(@Param("eid") Integer eid, @Param("sid") Integer sid);

|  |  |  |
| --- | --- | --- |
| 接口 | Integer updateEmployeeSalaryById(@Param("eid") Integer eid, @Param("sid") Integer sid) | |
| 调用关系和描述 | 框架通过model中定义生成sql语句，向sql发出操作。 | |
| 先决条件 |  | |
| 测试用例 | | |
| 输入 | 预期结果 | 实际结果 |
| 正常输入 | 2 | 2 |
| Eid正常，sid不存在 | DataIntegrityViolationException | DataIntegrityViolationException |
| Eid不存在，sid正常 | 1 | 1 |
| Eid和sid均不存在 | DataIntegrityViolationException | DataIntegrityViolationException |

三，测试总结

利用等价类方法，对13个方法实现了集成测试的全覆盖，由于在单元测试中的修复，部分问题已经被提前解决，不存在严重问题。

总结表：

|  |  |  |
| --- | --- | --- |
|  | 数值 | 比例 |
| 总方法数 | 13 | 100% |
| 缺陷方法数 | 1 | 7% |
| 无法正常运行方法数 | 0 | 0% |
|  |  |  |

附录：数据

完整数据：

template.setId(ID);  
template.setName("野兽先辈");  
template.setGender("男");  
template.setBirthday(new Date(1145,1,4));  
template.setIdCard("114514");  
template.setWedlock("未婚");  
template.setNationId(1);  
template.setNativePlace("上海");  
template.setPoliticId(3);  
template.setEmail("xiabeize@xbz.com");  
template.setPhone("11451490190");  
template.setAddress("下北泽");  
template.setDepartmentId(91);  
template.setJobLevelId(15);  
template.setPosId(33);  
template.setEngageForm("劳务合同");  
template.setTiptopDegree("本科");  
template.setSpecialty("是学生");  
template.setSchool("下北泽综合大学");  
template.setBeginDate(new Date(114,5,14));  
template.setWorkState("在职");  
template.setWorkID("114514");  
template.setConversionTime(new Date(114,5,14));  
template.setNotWorkDate(null);  
template.setBeginContract(new Date(114,5,14));  
template.setEndContract(new Date(114,5,14));  
template.setWorkAge(11);

部分缺失数据：

partEmpty.setId(ID+1);  
partEmpty.setName("野兽先辈");  
partEmpty.setGender("男");  
partEmpty.setBirthday(new Date(114,5,14));

测试执行时数据库中共有633条数据。

有效id为114514。

部分有效id为114515。

无效id为114416。