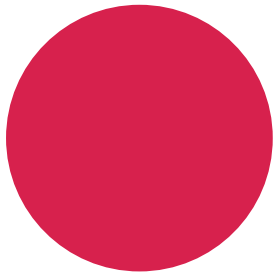
# 如何有效的使用代码覆盖率提高代码质量

主讲人：邱化峰

拉扎斯网络科技（上海）有限公司

# 目录

# 初级使用代码覆盖率

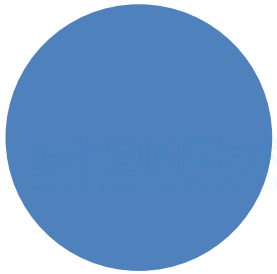


## Branches

- IF ELSE
- Case
- ? :
- &&
- ||

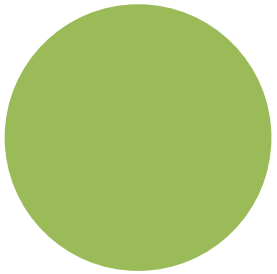
## Cxty

- 判定结构的复杂度
- 独立线性路径条数



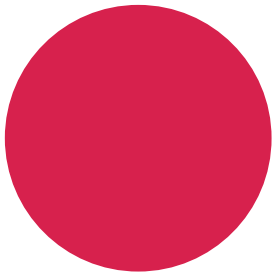
## Lines

- 行覆盖



## Methods

- 方法覆盖



初级使用

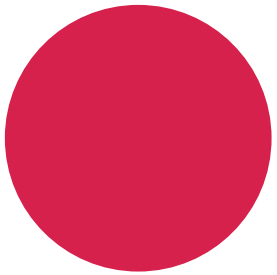
| | |
|---|---|
| 红色 | No coverage: No instruction in the line has been executed |
| 绿色 | Full coverage: All instructions in the line have been executed |
| 黄色 | Partial coverage: Only a part of the instruction in the line have been executed |

# 初级使用

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| exportCollectOrder(DeliveryOrderSearchVo, HttpServletResponse, HttpServletRequest) | 0% | 0% | 3 | 3 | 24 | 24 |
| exportListForDeliveryOrder(DeliveryOrderSearchVo, HttpServletResponse, HttpServletRequest) | 0% | 0% | 3 | 3 | 32 | 32 |
| syncAndCheckDeliveryOrderItem(OrderItemVo) | 11% | 20% | 5 | 6 | 20 | 27 |
| syncDeliveryOrder(OrderItemVo) | 60% | 75% | 2 | 5 | 9 | 28 |
| findCollectOrder(DeliveryOrderSearchVo) | 71% | n/a | 0 | 1 | 2 | 6 |
| findDeliveryOrderTotal(DeliveryOrderSearchVo) | 75% | 50% | 4 | 5 | 3 | 17 |
| stringToList(String) | 76% | 83% | 1 | 4 | 2 | 10 |
| findCollectOrderByPg(DeliveryOrderSearchVo) | 78% | 50% | 2 | 3 | 4 | 16 |
| findDeliveryPlayOrderTotal(SupplierOrderCountIpt) | 82% | 88% | 1 | 5 | 3 | 18 |
| isAllDeliveryOrderItemGenNew(DeliveryOrderItemVo) | 84% | 50% | 2 | 3 | 1 | 5 |
| isAllDeliveryOrderItemGen(DeliveryOrderItemVo) | 84% | 50% | 2 | 3 | 1 | 5 |
| findDeliveryOrderCollectWithPg(SupplierOrderCountIpt) | 86% | 67% | 2 | 4 | 4 | 20 |
| findDeliveryOrderWithPg(SupplierOrderCountIpt) | 86% | 67% | 2 | 4 | 4 | 20 |
| getTakeawayDate() | 90% | 50% | 2 | 3 | 1 | 8 |
| getSupplierId() | 93% | 50% | 3 | 4 | 1 | 8 |
| getTradeAreaDischargeName(List) | 93% | 81% | 3 | 9 | 3 | 27 |
| findDeliveryOrderByPg(DeliveryOrderSearchVo) | 94% | 69% | 11 | 19 | 6 | 63 |
| findDeliveryOrder(DeliveryOrderSearchVo) | 94% | 72% | 9 | 17 | 5 | 53 |
| setSupplierSearchDate(DeliveryOrderSearchVo) | 94% | 50% | 3 | 4 | 1 | 12 |

初级使用

```
687.        try{
688.            //添加根据角色组设置的数据权限进行控制
689.            supplierOrderCountIpt = supplierOrderConditionService.addDataLevelLimit(supplierOrderCountIpt);
690.            if (supplierOrderCountIpt == null) {
691.                return page;
692.            }
693.            //该部分代码注释掉，为了支持根据角色组设置的数据权限进行控制
694.  //        boolean isBDGroup = groupService.isBDGroup(ApplicationUserContext.getUser().getId());
695.            supplierOrderCountIpt.setUserId(getSupplierId());
696.            setSupplierOrderCountIptDetail(supplierOrderCountIpt);
697.            int totalCount;
698.  //        if (isBDGroup) {
699.  //            setQueryCondition4BD(supplierOrderCountIpt);
700.  //            totalCount = deliveryOrderItemSlaveDao.countDeliveryPlanOrder4BD(supplierOrderCountIpt);
701.  //        } else {
702.                totalCount = deliveryOrderItemSlaveDao.countDeliveryPlanOrder(supplierOrderCountIpt);
703.  //        }
704.
705.            page.setTotalRows(totalCount);
706.            page.setCurPage(supplierOrderCountIpt.getPagination().getCurPage());
707.            if(totalCount <= 0){
708.                return new Pagination<>();
709.            }
710.            List<DeliveryOrderItem> list;
711.  //        if (isBDGroup) {
712.  //            list = deliveryOrderItemSlaveDao.findDeliveryPlanOrderByPg4BD(supplierOrderCountIpt);
713.  //        } else {
714.                list = deliveryOrderItemSlaveDao.findDeliveryPlanOrderByPg(supplierOrderCountIpt);
715.  //        }
716.
717.            if(CollectionUtils.isEmpty(list)){
718.                return page;
719.            }
720.            List<DeliveryOrderItemVo> listVo = getDeliveryOrderItemVos(list);
721.
722.            page.setData(listVo);
723.        }catch (Exception e){
```



初级使用

会看代码覆盖率的报告，知道各种颜色的含义

知道那些行（代码）被执行过，那些未被执行过

知道通过代码覆盖率，补充测试用例

# 中级使用代码覆盖率

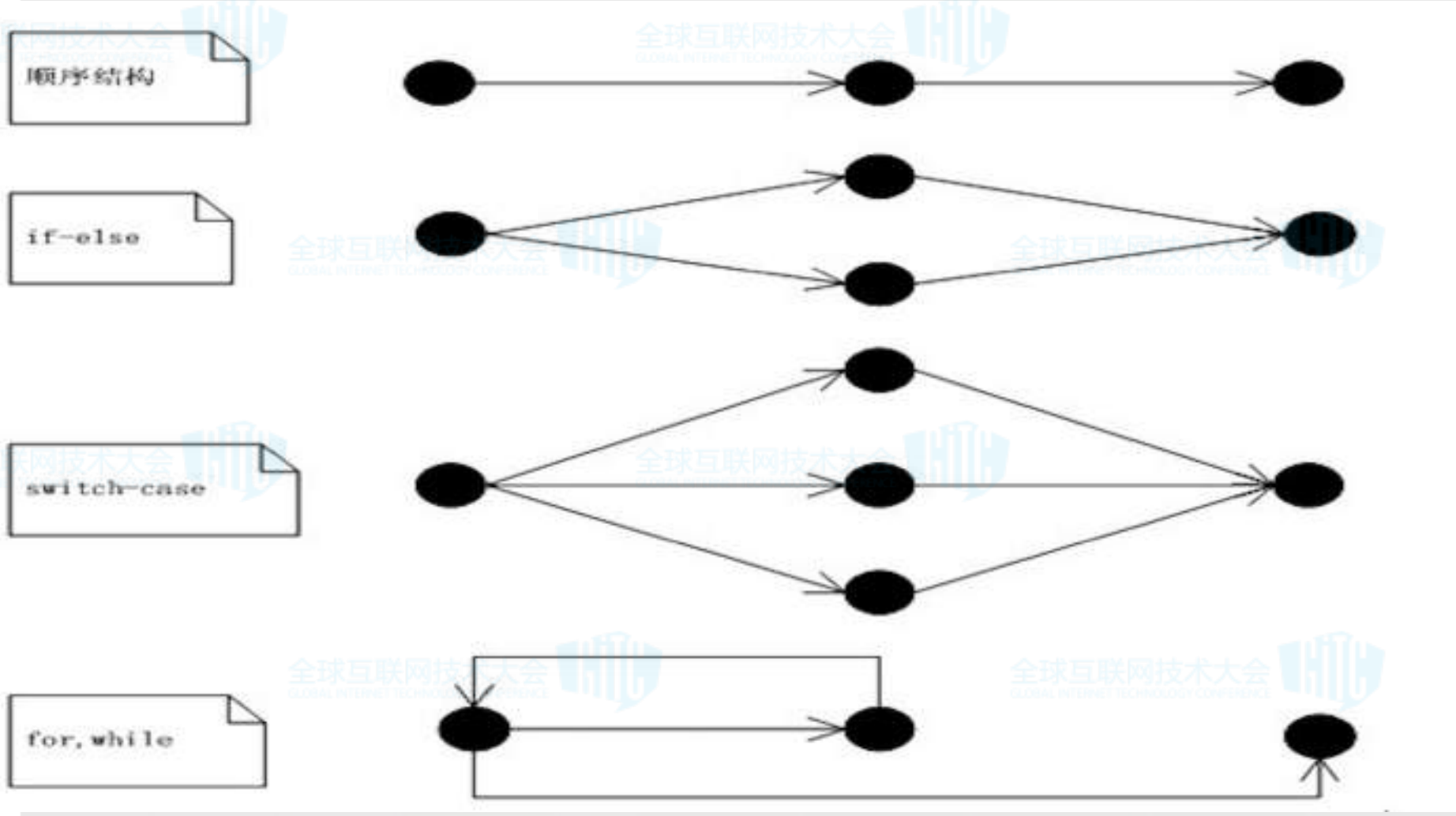
中级使用

| | |
|---|---|
| 圈复杂度 | 代码复杂度的衡量标准，程序的可能错误和高的圈复杂度有着很大关系，圈复杂度可以成为编码及重构的重要参考指标 |
| 最少用例数 | 覆盖所有情况使用的最少用例数，在系统软件和军事软件有75%的在使用，将圈复杂度维持在10以下有很多实际的和经济的理由，低于10的软件是非常简单且很容易跟踪的， |
| 降低圈复杂度的重构技术 | 参考 《代码大全》 《重构 改善既有代码的设计》 |

使用的计算公式是V(G) = e – n + 2 , e 代表在控制流图中的边的数量（对应代码中顺序结构的部分），n 代表在控制流图中的节点数量，包括起点和终点

（1、所有终点只计算一次，即便有多个return或者throw；2、节点对应代码中的分支语句）

# 中级使用



# 中级使用

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines |
|---|---|---|---|---|---|---|---|---|
| findDeliveryOrderByPg(DeliveryOrderSearchVo) | | 94% | | 69% | 11 | 19 | 6 | 63 |
| findDeliveryOrder(DeliveryOrderSearchVo) | | 94% | | 72% | 9 | 17 | 5 | 53 |
| getSupplyDishInfo(Long, Date, SupplierDishOpt) | | 0% | | 0% | 9 | 9 | 34 | 34 |
| getRestTradeName(List, HashMap) | | 95% | | 88% | 2 | 9 | 3 | 36 |
| getTradeAreaDischargeName(List) | | 93% | | 81% | 3 | 9 | 3 | 27 |
| syncAndCheckDeliveryOrderItem(OrderItemVo) | | 11% | | 20% | 5 | 6 | 20 | 27 |
| updateSupplyDishStatus(Long, String) | | 0% | | 0% | 6 | 6 | 21 | 21 |
| setQueryCondition4BD(SupplierOrderCountIpt) | | 0% | | 0% | 5 | 5 | 12 | 12 |
| syncDeliveryOrder(OrderItemVo) | | 60% | | 75% | 2 | 5 | 9 | 28 |
| filterDisChange(List) | | 0% | | 0% | 5 | 5 | 9 | 9 |
| findDeliveryOrderTotal(DeliveryOrderSearchVo) | | 75% | | 50% | 4 | 5 | 3 | 17 |
| findDeliveryPlayOrderTotal(SupplierOrderCountIpt) | | 82% | | 88% | 1 | 5 | 3 | 18 |
| stringToList(String) | | 76% | | 83% | 1 | 4 | 2 | 10 |
| findDeliveryOrderWithPg(SupplierOrderCountIpt) | | 86% | | 67% | 2 | 4 | 4 | 20 |
| findDeliveryOrderCollectWithPg(SupplierOrderCountIpt) | | 86% | | 67% | 2 | 4 | 4 | 20 |
| setSupplierSearchDate(DeliveryOrderSearchVo) | | 94% | | 50% | 3 | 4 | 1 | 12 |
| getSupplierId() | | 93% | | 50% | 3 | 4 | 1 | 8 |
| exportListForDeliveryOrder(DeliveryOrderSearchVo, HttpServletResponse, HttpServletRequest) | | 0% | | 0% | 3 | 3 | 32 | 32 |
| exportCollectOrder(DeliveryOrderSearchVo, HttpServletResponse, HttpServletRequest) | | 0% | | 0% | 3 | 3 | 24 | 24 |

# 中级使用

提取函数— 将独立业务活模块代码单独封装为函数

算法替换—复杂的算法可能会导致Bug，满足功能的前提下，使用简单的算法

分解条件式—复杂的条件表达式封装为函数

合并条件式—将一系列得到相同结果的条件表达式合并

查询函数和修改函数分离—-单一职责原则，强调复用性

合并重复的条件判断——不同的分支有相同的处理，提炼到分之外

# 高级使用代码覆盖率

# 高级使用

## 跟CI的持续集成

1.自动化的回归并收集代码覆盖率

2.查看代码覆盖率的历史记录

## 精准的代码覆盖率

1.识别出所有被修改的方法（新增，删除和修改）

2.可以有效的查看那些被修改的方法是否被测试到

3.通过圈复杂度衡量代码的质量

## 突变测试

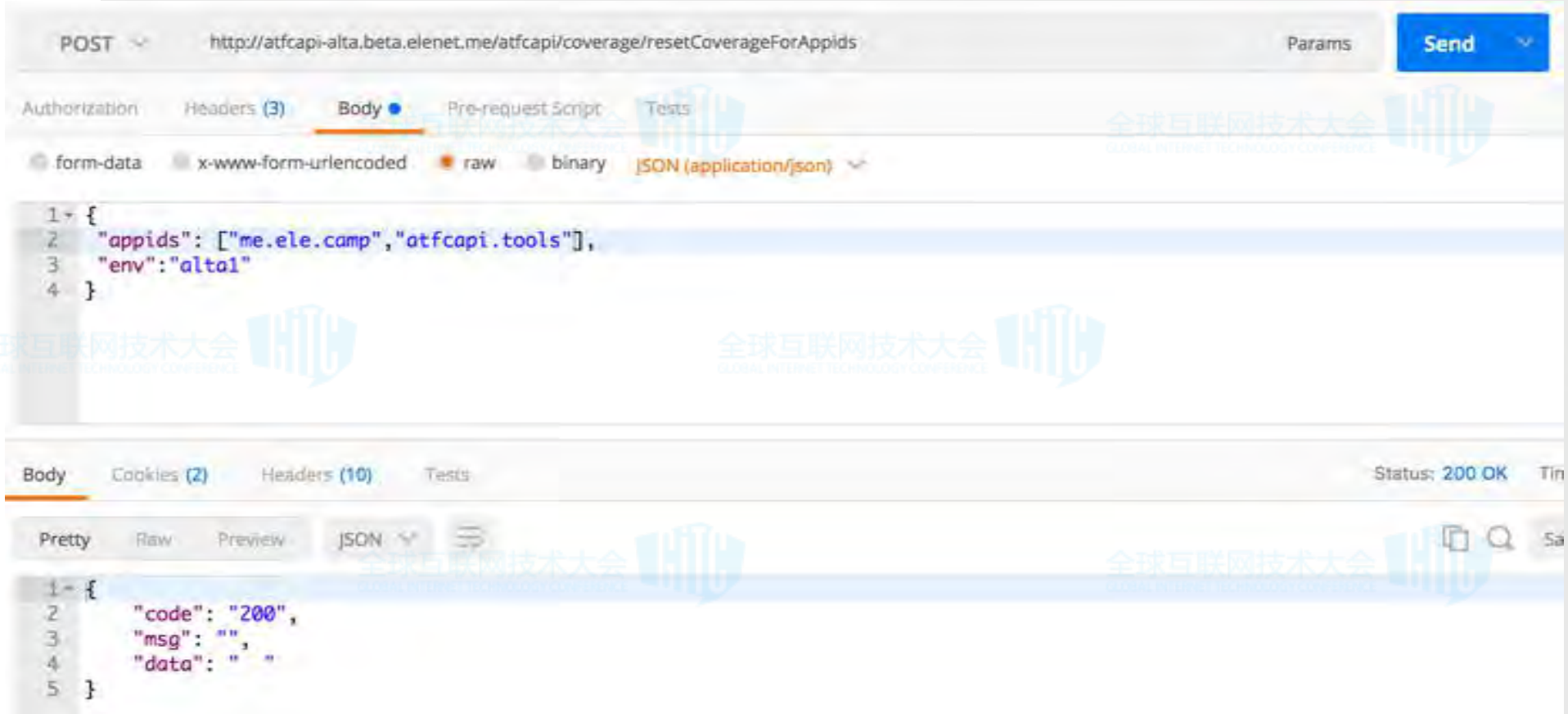
1.为什么要引入突变测试

2.如何通过突变测试来衡量单元测试的质量

3.突变测试的成本及收益及发展

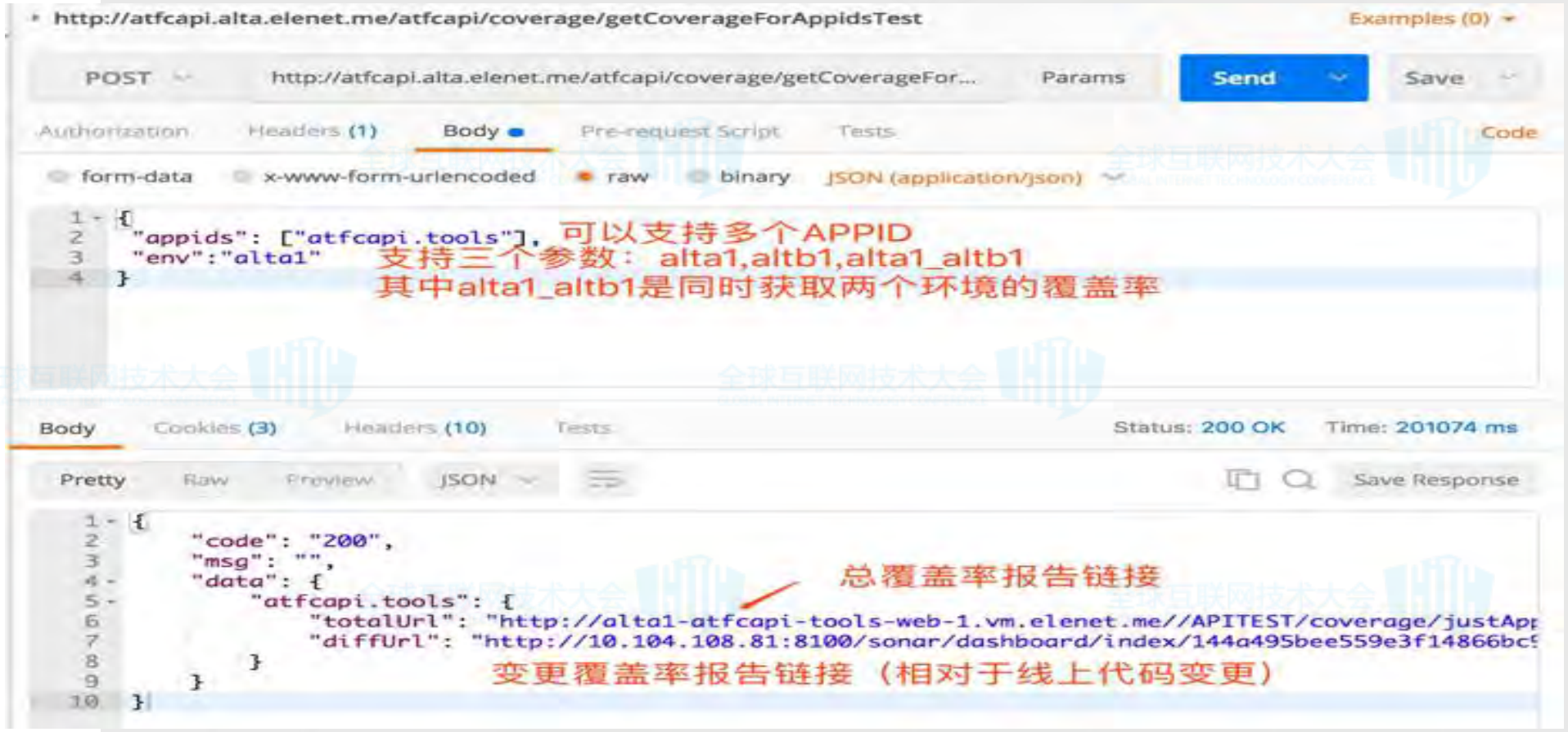
跟发布系统的集成，重置代码覆盖率

CI的持续集成



CI的持续集成

跟发布系统的集成，获取代码覆盖率

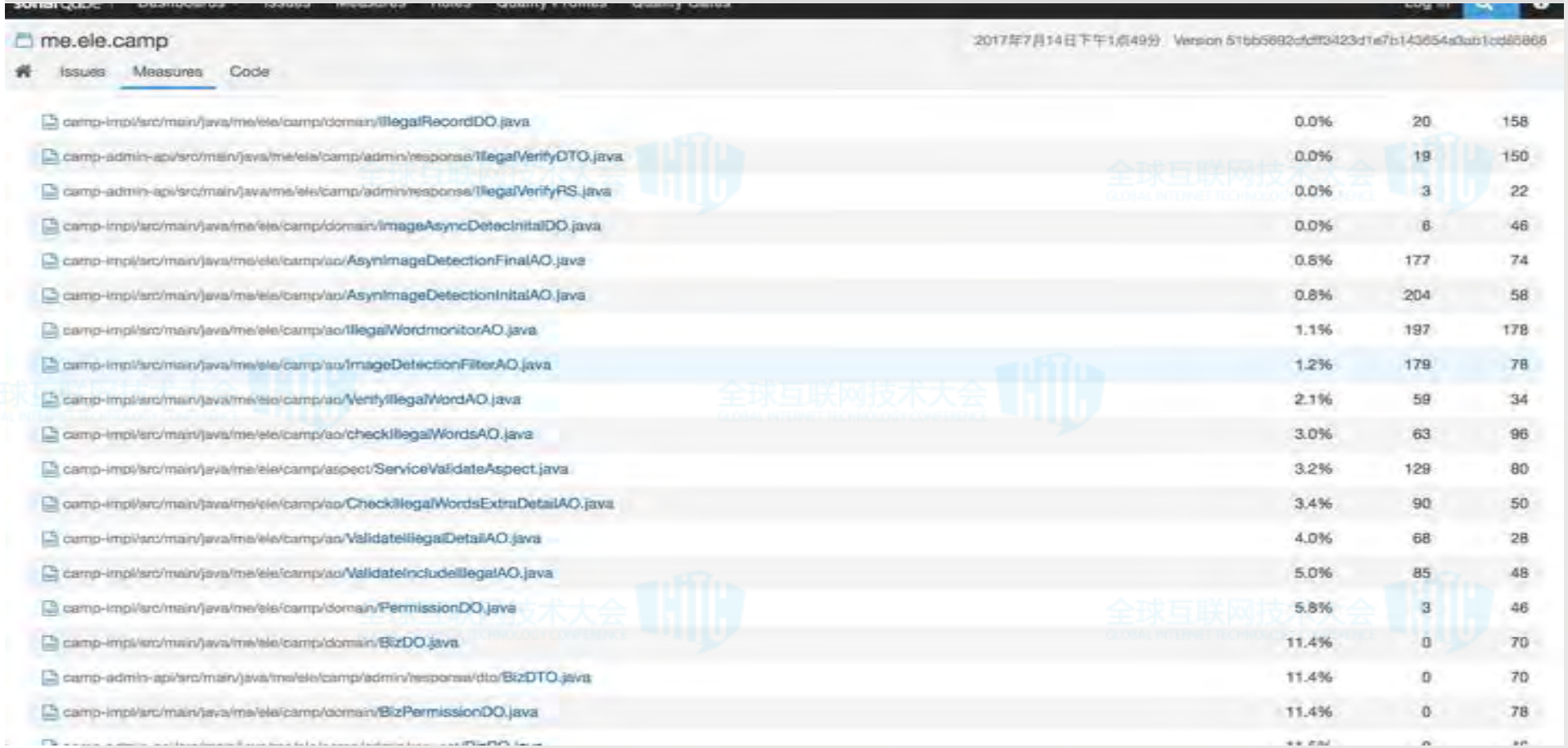


CI的持续集成

跟Sonar的集成

CI的持续集成

跟Sonar的集成



精准的代码覆盖率

识别出所有被修改的方法（新增，删除和修改）


精准的代码覆盖率

可以有效的查看那些被修改的方法是否被测试到

| Package | Class | Method | Method Coverage |
| --- | --- | --- | --- |
| me.ele.dt.horus.service.impl | DtEdbrainEtlServiceImpl | getEtlTableTitle() | 100.0% |
| me.ele.dt.horus.service.impl | | | 73.0% |
| me.ele.dt.horus.service.impl | | | 0.0% |
| me.ele.dt.horus.service.impl | | | 0.0% |
| me.ele.dt.horus.service.impl | | | 78.0% |
| com.ele.me.dt.horus.edbrain.hive.connect | HiveConnection | | 30.0% |
| com.ele.me.dt.horus.edbrain.hive.connect | | | 88.0% |
| com.ele.me.dt.horus.edbrain.hive.connect | | | 91.0% |
| com.ele.me.dt.horus.edbrain.hive.connect | | | 90.0% |
| me.ele.dt.horus.service.impl | DtEdbrainTblsServiceImpl | | 70.0% |
| me.ele.dt.horus.service.impl | | | 82.0% |
| com.ele.me.dt.horus.edbrain.spark.connect | SparkConnection | | 0.0% |
| com.ele.me.dt.horus.edbrain.service.impl | EdbrainEtlServiceImpl | | 80.0% |
| com.ele.me.dt.horus.edbrain.service.impl | | | 85.0% |
| com.ele.me.dt.horus.edbrain.service.impl | | | 73.0% |
| com.ele.me.dt.horus.edbrain.service.impl | | | 75.0% |
| com.ele.me.dt.horus.util | EncrypterFactory | | 0.0% |
| com.ele.me.dt.horus.util | | defaultString(String) | 100.0% |

突变测试

## 为什么要引入突变测试

Mutation testing is a technique for systematically mutating source code in order to validate test suites. It makes small changes to a program’s source code and then runs a test suite; if the test suite ever succeeds on mutated code then a flag is raised。

Unit testing ensures your production code is relevant. But what does ensure your testing code is relevant? Come discover mutation testing and make sure your never forget another assert again. In the realm of testing, the code coverage metrics is the most often talked about. A line (or a branch) is considered to have been covered if during test execution this line has been run. However, it doesn't mean that the test has been useful or even that an assert has been coded. Mutation testing creates mutants from code - a deviation from the written code, and test them also. If a test still passes on a mutant, then the test is not relevant

如何通过突变测试来衡量单元测试的质量 - https://github.com/tigerqiu712/jgit-cookbook

突变测试

**org.dstadler.jgit.api**

| Number of Classes | Line Coverage | | Mutation Coverage | |
|---|---|---|---|---|
| 18 | 79% | 245/310 | 1% | 1/108 |

**Breakdown by Class**

| Name | Line Coverage | | Mutation Coverage | |
|---|---|---|---|---|
| CheckMergeStatusOfCommit.java | 83% | 24/29 | 0% | 0/13 |
| GetCommitMessage.java | 91% | 10/11 | 0% | 0/3 |
| GetRefFromName.java | 83% | 5/6 | 0% | 0/1 |
| GetRevCommitFromObjectId.java | 93% | 14/15 | 0% | 0/5 |
| GetRevTreeFromObjectId.java | 92% | 12/13 | 0% | 0/4 |
| ListFilesOfCommitAndTag.java | 0% | 0/38 | 0% | 0/14 |
| PrintRemotes.java | 89% | 8/9 | 0% | 0/2 |
| ReadBlobContents.java | 94% | 17/18 | 0% | 0/7 |
| ReadFileFromCommit.java | 90% | 19/21 | 17% | 1/6 |
| ReadTagFromName.java | 94% | 15/16 | 0% | 0/5 |
| ReadUserConfig.java | 79% | 11/14 | 0% | 0/6 |
| ResolveRef.java | 92% | 11/12 | 0% | 0/4 |
| ShowBranchTrackingStatus.java | 83% | 19/23 | 0% | 0/5 |
| WalkAllCommits.java | 93% | 14/15 | 0% | 0/7 |
| WalkFromToRev.java | 95% | 20/21 | 0% | 0/10 |
| WalkRev.java | 94% | 16/17 | 0% | 0/8 |
| WalkTreeNonRecursive.java | 94% | 15/16 | 0% | 0/4 |
| WalkTreeRecursive.java | 94% | 15/16 | 0% | 0/4 |

突变测试

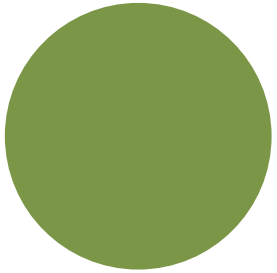
如何通过突变测试来衡量单元测试的质量

```
57 1                    if (!treeWalk.next()) {
58                          throw new IllegalStateException("Did not find expected file 'README.md'");
59                      }
60
61                      ObjectId objectId = treeWalk.getObjectId(0);
62                      ObjectLoader loader = repository.open(objectId);
63
64                      // and then one can the loader to read the file
65 1                    loader.copyTo(System.out);
66                  }
67
68 1                revWalk.dispose();
69              }
70          }
71      }
72  }
```

## Mutations

```
50  1. removed call to java/io/PrintStream::println → SURVIVED
55  1. removed call to org/eclipse/jgit/treewalk/TreeWalk::setRecursive → SURVIVED
56  1. removed call to org/eclipse/jgit/treewalk/TreeWalk::setFilter → SURVIVED
57  1. negated conditional → KILLED
65  1. removed call to org/eclipse/jgit/lib/ObjectLoader::copyTo → SURVIVED
68  1. removed call to org/eclipse/jgit/revwalk/RevWalk::dispose → SURVIVED
```

如何通过突变测试来衡量单元测试的质量



突变测试

Surviving means changing the source code did not change the test result

It ‘s bad!

Killed means changing the source code changed the test result

It’s good

突变测试

突变测试的成本及收益及发展

不需要改变测试代码

引入或者配置一下项目即可

测试策略

通过代码覆盖率改进-分支或者异常覆盖率不够

| Package | Class | Method | Method Coverage |
| --- | --- | --- | --- |
| me.ele.minimart.backend.manager | StockInfoManager | queryStockInfoList(StockQueryForPageReqDto) | 79.0% |

```
72.     public List<StockQueryForPageRepDto> queryStockInfoList(StockQueryForPageReqDto stocPageReqDto) {
73.         //组装查询条件
74.         Map<String, Object> params = new HashMap<String, Object>();
75.         if (null == stocPageReqDto.getCityId() && CommonConstant.I.equals(stocPageReqDto.getCityId())) {
76.             LOG.error("StockInfoManager queryStockInfoList 缺少必要参数城市id,dto:{}", stocPageReqDto);
77.             throw new MiniException(ErrorCode.NO_CITY_ID.getErrorCode(), ErrorCode.NO_CITY_ID.getErrorMsg());
78.         }
79.         if (StringUtils.isNotBlank(stocPageReqDto.getEnterpriseName())) {
80.             params.put("enterpriseName", stocPageReqDto.getEnterpriseName());
81.         }
82.         if (StringUtils.isNotBlank(stocPageReqDto.getOutletsName())) {
83.             params.put("name", stocPageReqDto.getOutletsName());
84.         }
85.         params.put("status", Status.ON_SHELF.getValue());
86.         params.put("cityId", stocPageReqDto.getCityId());
87. /*          //符合条件的网点数量
88.         long count = shelfGroupInfoMapper.queryCountByPage(params);*/
89.         List<ShelfGroupDetail> shelfGroupDetailList = shelfGroupInfoMapper.queryStockList(params);
90.         List<StockQueryForPageRepDto> stockQueryForPageRepDtos = Lists.newArrayListWithCapacity(shelfGroupDetailList.s
91.         //封装List
92.         if (!CollectionUtils.isEmpty(shelfGroupDetailList)) {
93.             stockQueryForPageRepDtos = shelfGroupDetailList.stream().map(shelfGroupDetail -> {
94.                 StockQueryForPageRepDto stockQueryForPageRepDto = null;
95.                 //获取该网点下所有货架的待补货金额与销售占比
96.                 Map<String, Object> shelfParam = new HashMap<String, Object>();
97.                 shelfParam.put("groupId", shelfGroupDetail.getId());
98.                 shelfParam.put("shelfStatus", Status.ON_SHELF.getValue());
99.                 shelfParam.put("commodityStatus", Status.ON_SHELF.getValue());
100.                ShelfAssembleInfo shelfAssembleInfo = shelfInfoMapper.getStockSoldInfo(shelfParam);
101.                if (null == shelfAssembleInfo.getShelfGroupId() || null == shelfAssembleInfo.getCommodityId()) {
102.                    return stockQueryForPageRepDto;
103.                }
104.                stockQueryForPageRepDto = BeanPropertyCloneUtil.map(StockQueryForPageRepDto.class, shelfGroupDetail);
105.                //拼接地址
```

测试策略

通过代码覆盖率改进-返回值的校验

| cn.delimont.stm.biz.service.impl.base | | changeStoreVacation(Long, String) | 82.0% |
|---|---|---|---|

```
82.    @Override
83.    public ResponseDTO changeStoreVacation(Long storeId, String name) {
84.
85.        try {
86.            String today = LocalDate.now().format(DateTimeFormatter.ISO_LOCAL_DATE);
87.            StoreDailyExpense storeDailyExpense = storeDailyExpenseMapper.getDailyExpense(storeId,today);
88.            if (storeDailyExpense==null){
89.                storeMapper.changeStoreVacationById(storeId,name);
90.                return ResponseDTO.getSuccessResponse(Boolean.TRUE);
91.            }else {
92.                logger.error("id为"+storeId+"的门店"+ReturnCode.STORE_CHANGE_VACATION_FAIL_TODAY_POWER_REPORT.getMsg());
93.                return ResponseDTO.getErrorResponse(ReturnCode.STORE_CHANGE_VACATION_FAIL_TODAY_POWER_REPORT,Boolean.FALSE);
94.            }
95.        } catch (Exception e) {
96.            logger.error("修改门店休假状态失败, 原因: ",e);
97.            return ResponseDTO.getErrorResponse(ReturnCode.SYS_EXCEPTION,Boolean.FALSE);
98.        }
99.    }
```

测试策略

通过代码覆盖率改进-注解类的测试

| | | | |
|---|---|---|---|
| cn.delimont.sys.api.entity | SysConfig | getIsDelete() | 100.0% |
| cn.delimont.sys.api.entity | | getModifyBy() | 100.0% |
| cn.delimont.sys.api.entity | | getConfigValue() | 100.0% |
| cn.delimont.sys.api.entity | | setCreateTime(Date) | 100.0% |
| cn.delimont.sys.api.entity | | setCreatorId(Integer) | 100.0% |
| cn.delimont.sys.api.entity | | getModifierId() | 100.0% |
| cn.delimont.sys.api.entity | | setConfigValue(String) | 78.0% |
| cn.delimont.sys.api.entity | | getComments() | 100.0% |
| cn.delimont.sys.api.entity | | getCreatorId() | 100.0% |
| cn.delimont.sys.api.entity | | getCreateTime() | 100.0% |
| cn.delimont.sys.api.entity | | setCreateBy(String) | 100.0% |
| cn.delimont.sys.api.entity | | getId() | 100.0% |

```
14.
15. @Validate
16. @Data
17. @NoArgsConstructor
18. @EqualsAndHashCode
19. public class SysConfig implements Serializable {
```

# THANKS !