工作汇报：

2017.5.16

主要是环境的搭建：

1. 常规的java环境配置，Jdk和JAVA\_HOME的配置。
2. homebrew工具，方便安装，更新等操作。
3. 创建/home/admin目录时，误操作打开了etc文件的写权限，导致sudo的使用出现问题。
4. 配置本台电脑的git公钥，与自己的github进行连接，方便以后操作。

5.下载idea的时候，会出现system path invalid的问题，因为从官网上下载的时候，会在默认cache文件下面建立安装文件，如果写权限没有开的话，会报错，所以需要把权限打开。

sudo chmod  -R a+w .（更改文件夹及其子文件夹） 最好是在自己的文件下面，如果修改系统文件的读写权限，会引起错误。

6.下载maven，方便管理。

7.配置tomcat，用已下载的淘宝的tomcat。

open . 打开当前文件的界面，可以方便操作。

8.熟悉了一下iwant项目的大体架构。

下一步：

熟悉源码，为项目做准备

（很感谢师兄们的帮助！）

2017.5.17

1.今天上午办完卡10.50到办公室，参加了会议。对菜鸟的项目开发流程有了大概的了解，作为技术人员，也要有做产品的思维，这样才能读懂用户的想法，开发更好的代码。

2.在起应用的时候，遇到一些困难，tomcat没有成功启动起来。按照大概的思路进行排查，首先查看tomcat的配置，发现vm选项配置文件有错误；然后排查过程中发现，maven的settings.xml文件也出错了。在拷贝的过程中漏掉了小部分。这样的话，没法用maven把整个应用启动起来；最后，发现ali-tomcat可能下载有误，重新下载了一下。

3.跑起来应用之后，需要对JRebel进行配置。提示找不到rebel.xml,因为还没有勾选需要用到JRebel的文件。相关的东西都在新人导航里做了修改，方便以后的同学。

4.下午的时候，师兄给我讲解了整个鸟瞰的架构。包括全栈学习，前端，后端开发，测试都需要自主完成。还有代码的大致讲解。对SpringMvc+Hibernate框架没有做过实例，后期需要自己研究代码走通流程。

2017.5.18

1.上午是对项目的大体流程进行了熟悉，知道了MVC三者之间代码的联系。

2.师兄布置了任务，给我讲解了需要做的需求，总的来说是做一个项目总览，把项目的一些信息用图表等展示出来。目前新建了一个前端project­\_overview,打算从学做前端开始，带着做DO。

3.下午是对前端vm文件格式的研究。后来在写页面的时候，发现，虽然启动成功，但是页面却出现404错误，在师兄的帮助下，逐步查错，（1）404发现是url写错的问题。（2）500的错误，提示是内部服务器的问题。用maven打包编译，还是出错；Git查看代码的提交，发现没有代码修改的误操作；怀疑是Jrebel问题，用普通debug还是错；最后觉的最大的错误可能是数据库访问没有权限，被拒绝。权限问题有点繁琐。

2017.5.19

周报：

1. 总览：

从本周二开始，正式的开始工作。

1. 基础环境的搭建，相关工具的熟悉。
2. 参与项目会议，对鸟瞰整体方向和项目内容有了大致的了解
3. 着手项目准备，学习框架，研究需求，准备开发。
4. 工作内容
5. 针对新人手册的内容，一边实施，一边对其做一些改进，以免后面来的新人走弯路。
6. 一边学习源码，一边着手写需求代码。但是由于数据库权限问题，开启tomcat服务器之后，访问不到页面。需要尽快解决这个问题，否则会耽误后续工作。
7. 分配的项目需求是一个项目总览页面，现在的设想是先把第一个需求，基本信息，先画出来。
8. 感想：

项目的需求需要快速的熟悉相关内容。在这一周里我学习了很多知识，更重要的是，也慢慢培养了快速分析错误，解决问题的能力。每一个出错都是一个学习完善的机会，不能错过。最后很感谢各位师兄的帮助，耐心的帮我讲解问题！

2017/5/25

结束了三天的百技培训，终于回来了。比起听课，宁愿坐办公室写代码。。。

1. 培训期间，在晚上抽空解决了一些零碎的问题。今天新写了一个projectoverviewDO，后来发现AKprojectDO已经把所有数据都添加好了，没必要再写，就当练手了。
2. 运行的时候，发现日志提示找不到万鸟归巢的layout，主要是对新生Tips有些误解，文件夹有重叠。师兄修改了相关内容。另外git部分，当时不确定邮箱写自己的github，还是新的，所以没有设置，我添加了相关内容，推荐新生用集团域帐号和菜鸟的邮箱。
3. 下午的时候学了一会bootstrap规则，把页面粗略的搭建起来。另外写了需要的controller，向页面传值。一个雏形大概出来了。熟悉了jrebel的使用，方便了修改页面。
4. 晚上对git的分支实操了一下，理解的更深入了。

最后，仍然是感谢各位师兄的耐心答惑，加快了我的进度和学习速度。十分感激。

2017/5/26

今天进一步解决了第二个模块。第二个模块主要是表的展示问题。比较顺利的完成了。

今天主要卡在了第三模块，里程碑的展示。这个前端页面并没有接触过，所以上手有些吃力。问了几个师兄都没有做过这块，问了下师姐，提供了一些思路。之后我的打算是：

1. 自己写一个css文件。因为样式太多，全写在一个vm里，不太合适。
2. 仿照aone的样式，来写这个里程碑。

出现了两个问题：

1. css文件位置放错。不小心放到target文件夹。Target是编译输出的文件夹。
2. aone里的“里程碑源码“的嵌套层数很多，没有点时间理解不了他定义的类的样式，所以有些焦急。

晚上开了会之后，反思下自己。应该是昨天比较顺，今天卡了一下，觉得很突然。其实想一想很正常，这个模块对现在的我来说，确实是有难度的。如果把这个模块解决掉了，之后在前端这块就会很轻松了。

另外，从学习的状态跨入到工作的状态，是一种很宝贵的体验。我会珍惜这段经历，并快速转变成工作状态。

2017.5.31

今天写第四模块，一开始以为只是简单的表的注入，把表的查询值直接在前端展示就好了。后来发现要涉及到三个表。项目表，bug表，需求表。梳理了思路：

1. 先用项目号得到项目成员。
2. 得到每个项目成员的bug信息
3. 得到每个项目成员的需求信息。

后来，打算用返回的DO的方式，用Criteria直接查询，得到相应的个数，但是这样用到的sql语句会很多。效率会比较低。还是用map的方式查询效率比较高。磨刀不误砍柴工，还是得花点时间钻研一下。

2017.6.1

今天把第四模块实现了，虽然看起来只是一个表的展示，其实后台还是数据处理。把思路缕一下：

1. 首先，在DAO层，我添加了若干个方法，用来帮我收集数据。这个地方我采用的结构是List<Map<>>,把每个人的姓名及对应的信息保存起来。
2. 在Service层（为了让代码逻辑更清晰，分层写），我把得到的数据通过姓名合起来。并且通过json的方式，填写表头和行。
3. 在Controller层，我把json对象传到前端。
4. 在前端页面，我写了一个js，定义表格的形式。

问题：

1. 首先，在DAO层，花了点时间确定使用什么样的数据结构来存储我要的数据。这里还有优化的空间，打个tag（待优化）
2. 在Service层合并数据的时候，会遇到null或size()==0的问题，我的思路是先把第一个List填充完整，默认是0，然后其他list向第一个填充，null的话，就默认为0了。但是这样还会出现问题，因为有的数据会填充，有的为null，就不会填充。这就使得不同的数据，数据类型不一样。造成的现象很有迷惑性。另外，对数据的输出格式做了标准，保留小数点后两位。
3. 最后在前端构造表的时候，因为不熟，所以有格式错误。另外，与引用css文件类似，这里也要引用js文件。

总结：从无到有是一个比较难的过程。但是坚持下去，总能得到不错的结果；

在走通了整个流程后，对框架有了更深入的认识；

感谢师兄们的提点。

2017.6.2 周报

这周挺短的，就三天。但是成果还是蛮不错的。主要还是通过表的展示，把整个流程走通了，这是这周最大的收获。在后台处理数据的时候，一定要注意数据结构的选择，以及传入参数合理性的判断。

1.数据结构如果选择不好，会造成大量的计算。在一开始，我打算用List<Do>的方式，然后直接一个list.size()，就能得到数量，满足了我的需求，但是这样会花费大量的sql语句，当数据量大的时候，页面打开会非常的慢。另外就是参数的判断。对null和size()==0 ,要有一个判断，保证程序的健壮性。

2.其次，周五学会了制作图的过程。首先要写指标，但是在我的需求里，没法直接通过sql语句的方式得到我需要的指标，师兄给我介绍了javabean的方式。这种方式跟我周四写后台的方法很类似，得到的是List<Map<>>形式的结果。需要注意的是，指标系统找的是别名，且方法名与参数都要按格式来。然后要在controller添加相应的页面容器service. 最后写前端的时候，注意reportTool源码中，已经对格式做了设置，不要跟自己的格式冲突了。

3.最后，在写代码的时候，注意结构层次要分明。比如我在写指标的时候，把得到指标的javabean，写在了service，虽然也是可以跑通的，但是逻辑就很混乱。代码也要有个美观与整洁，全写在一起，肯定是不好看的。

4.感谢师兄们在工作和生活上的帮助！

2017/6/5

今天主要做了过程效率模块和需求分布模块。接受师兄的意见，先暂时没有处理三维图。

有了前几个星期的经验之后，感觉独立解决问题的能力大幅上升，这是最近感受的自身的一个变化。对框架和业务慢慢熟悉，在和师兄讨论并确定细节之后，可以很快的理清思路，思考出相应的对策，不再像之前一样束手无策。

比如今天和师兄讨论风险模块后，首先我发现一个项目是有多个risk的，之前的思考有些漏洞。很自然的会想到用兵力模块的表的模式来完成，但是不用照搬复制，因为不涉及到数据的拼接。直接选择用List<Do>. 接着就是“风险等级”和“状态”这两个指标，需要用枚举类来得到具体的名字。在service里面我直接引用并添加到json。传到前端的流程就很熟练了。

要注意的一点就是，代码是要不断优化的，修修改改才能越来越好。

2017/6/6

今天对里程碑的细节再次确认，把逻辑关系理清楚。根据状态的不同，分成4种情况。具体的细节不在这细说了。主要说下思考问题的流程。

1. 首先思路是数据库里按日期顺序得到DO，这里遇到一个新问题，当存在null，会默认放在前面（升序），此时可以倒序排列，也可以order by .. **is null, ..来实现。**
2. 然后在处理“今天”这个符号的时候，一开始我是想在前端直接用逻辑来判断。即，在遍历DO的时候，找到第一个大于当前时间的DO，然后直接画“今天”。但是在前端不是很好处理，而且在前端写复杂的逻辑是不合适的，它还需要从后端拿数据。最后采纳师兄的建议，把“今天”也作为一个DO，一起判断，然后画前端。
3. 不得不说，前端比较琐碎，需要点时间来调试。



2017/6/7

今天已经把基本的页面完成了。

1. 上午的时候遇到一个问题，每当我重启服务器的时候，我的一些用简单指标做的图都显示不出来，而横纵坐标都可以显示。说明数据没有拿到。最后发现是传参的问题。因为我现在做的页面没有入口，所以我统一留了一个入口，用户可以通过下拉框选择，或者url跳转的方式到达“项目全景”的页面，并传给我projectId这个参数，作为数据的来源。目前为了显示，我写死了一个projectId，所以说这个问题，如果是javabean形式画的图，则不会出现这种情况，因为后台可以拿到数据。
2. 下午的时候对前端的风格进行了调整，与项目的大体风格形成一致。
3. 下一步要解决的问题：
4. 一个是优化的问题，感觉线页面的打开速度有点慢，大概是2～3秒这样。
5. 另一个是入口的问题，确定projectId的入口，融入项目。
6. 项目提交的问题。写了太多代码，肯定是有写地方是冲突的，最好还是写一点，提交一点，及时解决问题，否则留到后面就复杂了。

2017/6/12

1.今天上午主要系统学习了一下JavaScript，反正都是要学的，还是抽时间学习学习，以备不时之需。另外，顺便改了几个小bug。

2.在和师兄讨论了之后，师兄又给我提了几个需求：

（1）项目健康度要调整一下，思路是根据porjectDO里第一个未完成的里程碑来判断，如果时间没有逾期就显示正常，逾期了就显示逾期多久。

（2）“兵力“里的表格，显示的开发人员的信息，不是所有人的信息。其实在”需求分布“里，也是开发人员的信息。但是团队成员分布肯定是所有成员的分布。所以根据需要，新写了一个listMemeber，得到的是开发人员的列表。这里遇到了criteria的问题，得灵活使用。

（3）新补充了一个bug状态分布图。

（4）个别的表格格式不对齐，纠正完成。

3.制作指标还是没有权限，目前没有更好的方法，只能修改本地代码。等提交的时候再修改过来。

2017/6/13

今天是很有收获的一天。上午师兄又提了几个要求，改了之后感觉细节更清楚了。晚上跟师兄们讨论过代码之后，发现了很多自己忽略掉的东西，很多都是细节方面的问题。所以工作学习都要精益求精，慢一点未必不是好事。

总结了一下，主要有几个典型问题。

1. 里程碑的几个标志要和Aone的保持一致。从用户的角度出发，如果第一次看到我的这个圆的图标，可能会产生奇异，所以保持一致是好的选择。
2. 数据为空的问题。这个问题在我的代码里面普遍存在。有的地方是注意到了，但是很多地方都忽视了，觉得不会有大问题。但是要时刻考虑到极端的情况，这样才能最大限度的模拟意外情况。
3. 另外一个细节问题就是，不同分辨率情况下，缩放不能有特别不规整的情况。这个确实没有考虑到。
4. 还有一个业务问题，延期的天数计算。首先最好使用库函数去计算，因为自己写的逻辑会有各种边界问题。另外，（1）若预期结束时间默认都是0时0分0秒，那么逾期几秒也算是延期1天。（2）若预期时间没有默认格式，要根据是否超过当天来计算，单纯的计算差的毫秒数确实是不合适的。
5. 最后是BO的应用问题和JavaBean的分布。之前没有了解过BO，所以有的地方可能绕了弯子，需要了解下BO的应用。因为大部分图都是用JavaBean写的，所以可以把我这个“项目全景”下所有的javabean放在一起，方便管理。

总结，多从用户的角度考虑问题，做一个合格的“产品经理“和“测试人员“。

2017/6/14

今天把自己的代码进行了一次整体优化，优化后的代码更加清晰，而且健壮度很好。不过，以后要注意，在写代码的时候就随时优化，处理边界，养成一个好习惯。

(1)在处理空的问题上，参考师兄的解决办法，同时自己也添加了几个方法帮助处理。首先，不仅是get值的时候需要判断null，通过projectId得到DO的时候很有可能得到的是null，即使这种情况不太可能发生，也得考虑到。数值判断null，除0都得考虑到，师兄推荐的几个类，都很有用。

(2)数据结构的优化。主要针对的是兵力表，这个表稍微有点复杂，涉及到两个表，而且每个字段都是需要另外计算的。之前是把几个List<Map<>>合并成一个大list。为了解决空的问题，当时是以第一个list为基准，做扩展，然后再把其他list加到第一个list。这样的话，必须得用二层循环，另外代码冗余，不好操作。

现在我根据字段的需要，写了自己的BO，然后采用Map<String，AkBugBO>的数据结构。首先，用我得到的项目成员list，根据name新建BO，然后由map建立<name, BO>的映射。这样就是接近O(N)的复杂度。

由sql语句的count操作得到的值默认是BigInteger，avg得到的是 BigDecimal类型。

(3)另外，师兄推荐的velocity的DateTool并不能解决我的实际问题，所以我重新写了一个函数，用Calendar的函数来操作，基本满足了需求。

现在看来，改bug，优化代码也是个大工程。所以最好的办法就是把优化分布到平时，每天都解决一点，不要堆起来。

2017/6/15 (周报)

这周的工作主要集中在对“项目全景”页面细节的优化改进上面。

1. 项目的健康度在经过讨论之后，由第一个未完成的里程碑来判断，如果时间没有逾期就显示正常，逾期了就显示逾期多久。而不是由里程碑表的数据来决定。
2. 每个模块的显示的人群不一样，要细化。有的模块是项目所有成员，有的则是针对开发人员。
3. 补充了一个projectOverViewBean, 存储当前页面图形所涉及到的javabean。为了方便管理，都统一写在了一起。新增了bug状态分布图。
4. 非空判断。程序的健壮性要达到很强的状态，随便输入一个不存在的项目编号，会有相应的处理，而不是直接挂掉。重新理了一遍各个模块，确保空值异常都要相应处理，且特殊的，如“正常”，“未填写”，要有颜色显示，增强用户体验。
5. 数据结构优化。对兵力表的数据结构进行了优化，采用Map<String，BO>的形式，复杂度会降低，另外方便使用库函数。
6. 修改了bug关闭时长分布模块，固定天数，方便用户浏览。
7. 里程碑的修改。前端的东西并不熟悉，所以是边学边做了。为了实现与AONE的对接，最好是采用用户熟悉的界面。之前套用的icon不太合适，在仔细研究了AONE页面之后，发现关键的因素在于伪元素的使用。例如，:after它的逻辑是在当前元素管辖范围最后加上设定的元素。前端的东西是不难的，但是很琐碎，仔细研究之后觉得还是很有意思的。

今天我的《The Pragmatic Programmer》到了，看了第一章，感觉很应景，

“Provide Options, Don’t Make Lame Excuse”. 提供选择，而不是找蹩脚的理由。作为新手程序员，难免会因为经验不足而犯一些比较低级的错误，这时候要认真的思考如何更好的解决问题或优化代码，而不是说一些理由。很庆幸，这一点我做的还算不错。

2017/6/19

今天完成了项目描述模块的内容。

1. 首先，要解决的是提交项目描述的模块。前端画好之后，我的需求是传送项目号，用户工号，项目描述。有两种方式，一种是form把text和button包裹起来。但是这样是在url带过去的，不太合适。选择用AJAX的传值方式。在DAO里，用hibernate的save 函数写进数据库。
2. 另外一块是描述展示模块。首先要确定逻辑，当前用户只能看到记录的编辑按钮。所以，按时间先后顺序读数据库之后，要确定哪一条记录与页面传进来的工号相等，然后再设置编辑按钮show。
3. 编辑框我采用的是悬浮框的模式，没有采用再打开连接的方式，那样有点浪费。这里涉及到一些前端展示，调试了一段时间。更新之后，有两个问题，一个是提示对象引用的问题，解决了之后，用saveOrUpdate会出现重复插入的现象，改成了update，细节我需要再看看，这边标记一下。之后应该会再添加一个删除的按钮，方便操作。

今天学到的东西还是非常多的，都是之前没用过的。感觉很好，又学到了很多新知识。

2017/6/20

做完项目描述模块之后，师兄又提了几个新需求。

1. 确认逻辑。本来我的思路是每个用户只能插入一条项目描述，要是有变更的话，就使用“修改”按钮，直接对数据库进行update操作。完成之后，与师兄讨论，变更为每个人可以有多条项目描述记录。之后我错误的想使用花名和时间作为确定记录的条件，其实是可以用getId()方法的。在我的潜意识，以为id是自增的，不应该有get，set方法。其实这是想当然的，以后一定要避免。
2. 一些前端格式优化。要考虑用户体验
3. 输入框，设计成收起模式，我的逻辑是通过要点击的span的内容来判断收起或展开。点击之后，若提示留下描述，就显示输入框，并且设置span内容为收起，这样就可以通过一个控件来首先展开收起。

今天发现最大的问题就是沟通问题。尤其是和PM的沟通，最好是在一开始想好之后，尽可能的和PM讨论细节，越细越好。理清楚要做的有哪些，少走弯路。

2017/6/21

解决项目描述的分页功能

（1）首先，分页的逻辑并不是隐藏，而是根据当前的page，去重新进行分页查询，然后在前端展示。

（2）所以，我先在前端画了paginator，并且写了相应的事件，把分页改变的页数传到设置好的一个input里面，再传到后端。在后端我写了一个projectDescriptionQuery<>，可以存放查询结果和设置分页信息。根据传过来的当前页，和每页的记录数，来用criteria筛选某个范围的页数，并放在Query中，一起再带到前端展示。

这里出现了一个小问题，每条记录的“修改”按钮，在点击之后就自动刷新页面，这是因为“修改按钮”设置为submit type，之前没有form，所以不会触发，而现在会触发表单，从而进行刷新。

（3） 小技巧，若是想传值而不想展示，可以用隐式文本域<input>；若出现嵌套的div等元素，其内容溢出的情况，可以设置style的clear:both，清楚浮动。

2017/6/22

今天的主要任务还是改bug，并且帮师兄做了一个需求，人员组织分布表。

1. 涉及到次数的字段，如果为空的话，默认为0，不是以前的未填写；而涉及到计算的字段，如果分子分母有一个为null，显示“暂时无法计算”。另外，关于健康度的计算，一开始是用数据库的firstUnfinishedMilestone字段，但是由于默认的字段是1970-01-01，所以得做个判断。其实还是有些疑问，需要再讨论讨论。
2. 人员组织分布表的话，思路还是很清晰的。传进来的入口是工号，然后查询当前该用户名下所有的TL，对于每个TL，查询其团队的基本信息和人员的层级（P，M）。然后，非0的字段添加链接，可以点击去查看人员明细。第一次用javabean的形式做表，学到了一个操作，就是在map操作的时候，可以直接添加拼接的<a href=’’></a> ,就省的前端再判断，再去画了。
3. 今天干了一件很无语的事。晚上在idea把新做的需求commit的时候，提交信息是上一次的，忘了改了。本着专业的原则，我reset了一下，要命的是，我加了hard参数。结果，强制把工作的代码也reset了，本地的修改没了。浪费了我一个小时，又重构了一遍代码。

Git reset commit\_id（重置版本库和暂存区）

Git reset --soft commit\_id （重置版本库）

Git reset --hard commit\_id （重置版本库，暂存区，工作区，危险操作）

我以为做了几个实验，算是比较熟悉git了。然而有的细节我并不熟悉。教训啊。

但是不能因噎废食，该学的还是要学的。只是要避免过大的损失。

（PS:还是有解决方案的。可以在git reflog 里面查到reset的记录，然后checkout就可以了）

2017/6/26

今天上午改了一个需求，当项目没有成员，bug总数为0，需求为0时，对应的图要隐藏起来。另外，跟师兄配合，把图表发布到预发环境下。最后成功把项目全景发布成功。

另外，解决了指标系统，点击其他页没有带值的问题。解决办法比较简单，把checkBox改为radio。因为逻辑上说，不应该同时选择简单指标和复合指标。在radio的属性里做判断，如果pointType为0，就设置该radio为checked。

单选框，复选框的问题。单选框的name一定要一样（id不一样），否则就没有单选框的效果了。

例如：我用getParameter(“radioName”)时，因为两个单选框的name一样，所以get到的是被选中的那个radio的value。

晚上师兄给我讲解了一下应用配置项的展示问题。还需要在消化一下。

类的序列化和反序列化。

2017/6/27

今天完成了应用配置信息的初级页面。写了对应的vm，DO，conroller。总的来说是比较常规的操作。但是往下走的时候遇到了问题。

1. 昨天的讨论之后，按照我的理解，做出来初级页面之后，点击url“diamond配置”，是通过应用名称，查询之后得到的dataId，group等相关信息。但是跟冲越师兄讨论之后，发现没有这种API。而且，之前讨论的需求是所有的应用都需要有配置一致性检查。冲跃师兄的方案跟昨晚上讨论的有点不一样。所以，需求还是要进一步明确。
2. 改了一下“视图-图标”的高亮小bug。它的原理是这样的，通过模式匹配来寻找最相似的那一栏，并设置为高亮。所以，在编辑的时候，view\_config匹配的是view，所以“视图-卡片”会被高亮。把整体的url调整一下即可。顺便学了一手git grep –in “report/view\_config” 查询当前文件所在的位置

人总是不太愿意改变，喜欢在自己熟悉的领域，很顺畅的把代码写出来。但是每一个不熟悉，比较痛苦的过程，都是一次宝贵的提升机会。今天涉及到的diamond，有些地方的模式是我没有接触过的，又体会到了刚来时候那种陌生的感觉。心态要摆正，这是一次好的机会。

2017/6/28

今天实现了堆叠折线图的功能。一直想做来着，但是一直分配其他任务，没来得及做，正好趁这个机会把流程走一遍。以后若是有其他图表引进，就有经验了。

1. 首先得在图标模版中新建一个模版。这个模版要参照highchart的官网模版进行一些修改。这个地方要稍微注意，json的解析是要求双引号的，而官网的模版则是无要求的。所以下次遇到不确定时候，要用jsoneditor之类的方法进行检验。
2. 之后就是在前端添加“堆叠折线图”的选项，再在view\_config中加入堆叠图的编号。之前以为是随意改的，其实是有规可循的。
3. 最后在ChartOptionsGeneratorFactory中添加对应的堆叠折线图的处理。总体来说，思路是很清晰的。
4. 在测试该功能的时候，学到一个知识点，逻辑删除，留一个回滚的机会。在iwant\_report\_view表里面，删除只是修改了status。但是这样有一个小问题，当我删除一个图表，然后再建一个重名图表时，会出错。明天要检查检查。

编程的过程还是要多思考，多总结，多积累。为什么师兄一看到com.alibaba.fastjson.JSONException: syntax error,一下子就想到是json解析的问题。虽然只是一个简单的错误，但这是不断的积累过程培养出来的直觉，可以快速定位到问题，提高效率。

2017/6/29

这周的事情是把“项目全景”成功发布，并且自己发现了一些bug并纠正。另外就是完成师兄分配给我的一些需求。

1. 上周在发布的过程中发现了一个小bug，很尴尬。这个bug很隐秘，本地数据库没有相关的测试数据，我也没有重点对这块进行检测。所以这周对“项目全景”页面又扫了一遍，这让我了解到测试的重要性。程序的健壮性是很重要的一点。
2. 在检查的过程中，发现项目以前留下的bug，挺影响用户操作的。一个是指标系统中，单选，复选框的问题，一个是编辑图标高亮显示错误的问题，还有一个是reportView逻辑删除的问题。写程序的过程中，多留心一点就会发现bug，好的态度是不要错过他们，在纠正的过程中，可以学到东西，同时也增强了用户体验。
3. 因为diamond的需求没有很快速的确定，所以我终于有空把之前想做的堆叠折线图模版做好了，以后大家可以用新图展示了。计划赶不上变化，需求是死的，人是活的，所以需求总是会变，要接受这种时有发生的现象。在主需求不确定的情况下，可以主动找别的需求做，保持一个很好的状态。

总的来说这周还是很充实的，是主动找问题和解决问题的过程。目前diamond需求已定，接下来就是尽快完成这个需求，不耽误业务的进度。

2017/7/3

今天完成了应用配置项的增删改查功能。其实是个挺普通的功能，但是有很多细节的东西需要注意。

（1）首先，为了实现层次，逻辑上的清晰，最好还是根据操作去选择requestMapping中的method。原则上说，GET，POST，PUT，DELETE分别对应了查数据，插入数据，更新数据，删除数据。虽然POST也能写更新的操作，但是不符合常用做法。要保持一个好的代码风格。

（2）在Ajax里面做页面跳转的时候，没有起效果。原因是表单提交的时候，其默认的行为阻止了页面的跳转。所以需要在函数里面添加e.preventDefault();阻止默认行为。

（3）一般情况下，增，删，改这种需要返回数据的操作，在controller中对应的方法前面都要加上加上@Responsebody，这样返回结果不会被解析为跳转路径，而是直接写入HTTP response body中。另外，接受数据的时候，也应该在Ajax里面JSON.stringift(),转化成JSON格式，再在方法里面用@RequestBody得到数据。

（4）删除操作的时候遇到了麻烦。因为采用的是指标操作，且指标对应的javabean里面，添加了click事件。所以在前端会出现同步冲突的现象。可以暴力点，直接用json写表，或者延时操作，但是并不优雅。师兄提示我在javabean里面用onclick=“方法名（this）”的思路，把元素带到vm中的js函数。然后用

$(element).data('config-id')，就可以了。

总结来说，还是对js不够熟悉，写的太少了。得多练练，实践中积累经验。

2017/7/5

今天完成了所有应用，和已配置应用的diamond检验。

1. 一致性检验主要是调用diamond提供的接口，通过我们手动输入的dataId和groupId，得到相应的配置信息，最后得到MD5，检查预发和阿里云的环境与线上中心是否一致。
2. 增加了“详情”选项，方便用户查看细节；调整了入口，增加了一个“查看已配置”的按钮，app表数据太多，这样方便查找。

感想：

1. 正确的定义问题：遇到错误的时候，不能只看表面。比如今天在写“详情”的时候，前端展示的表格内容顺序错了，中心预发不在第一位。按着直觉，开始搜索关键词“json 顺序”，其实一步步调bug，发现到前端之后，顺序依然是对的，其实是明细表格自带的排序没有取消。
2. 设计合理的架构：本来讨论之后，是想在“application\_configuration”页面添加radio单选框的。但是考虑到，“全部应用”和“已配置应用”不是出自同一个表，所以不太合适在同一个页面出现。所以改成跳转到另一个页面。

2017/7/6

本周完成需求：

1. 应用配置项的增删改查：表的结构选择中，留有逻辑删除的字段status。然后，根据操作去选择requestMapping中的method。原则上说，GET，POST，PUT，DELETE分别对应了查数据，插入数据，更新数据，删除数据。
2. 所有应用，和已配置应用的diamond检验：一致性检验主要是调用diamond提供的接口，通过我们手动输入的dataId和groupId，得到相应的配置信息，最后得到MD5，检查预发和阿里云的环境与线上中心是否一致。
3. 调用aone接口，根据入参查询相关信息：实现一个接口，功能是根据项目id，状态信息得到相应的结果。目前还在调试。

感想：

1. 设计表结构的时候，要考虑细节。一般不在表结构里面限制非空等操作。这样的话，遇到空的情况数据库会直接报错，而我们则不好介入。所以好的办法是把空的问题留给自己处理，不要在表里设置非空。

另外，为了保证误删除的情况下，数据还可以回滚，最好是在涉及到删除的数据库中添加status，逻辑删除则是把该字段设置为-1。

2. 正确的定义问题：遇到错误的时候，不能只看表面。比如在写“详情“的时候，前端展示的表格内容顺序错了，中心预发不在第一位。按着直觉，开始搜索关键词“json 顺序”，其实一步步调bug，发现到前端之后，顺序依然是对的，其实是明细表格自带的排序没有取消。

还有，今天碰到的servlet-HTTP Status 500的错误，错误信息提示并不是真正的问题根源。

3.要有很好的阅读api的习惯。最近经常会调用其他平台的接口。如diamond，Aone。着手写代码之前，得仔细的阅读提供的接口，其入参，返回结果是怎样的。这是很常见的应用场景，熟练掌握技巧对以后的操作会很有帮助。

2017/7/10

需求开发详情：

1. 应用配置页面的筛选功能，已完成。根据团队，已配置和未配置选项来选择相应的配置。可以筛选当前项目下的应用配置。两个radio单选框都不选的时候，选择全部的；选择已配置，未配置的，则显示相应配置项。
2. AoneSearchServiceImpl接口方法修改完毕。并且通过写AoneSearchServiceTest测试了相应的功能。在写了一些测试用例之后，得到了对应的结果。

感悟：

1. 写程序比较简单，要改的时候就觉得有点麻烦了，因为框架已经摆在那了。但是需求会经常变化，这时候只能思考有没有优美的解决方案。应用配置页面的筛选功能其实很简单，但是因为涉及到两个表，所以不能套用现成的detail模版。然后和pm讨论之后，将应用的下拉框取消了。因为留着会增加选择的复杂度，且有些多余。另外，根据应用场景，把已配置，未配置设置为单选框，且刷新后不传值。
2. 最近的需求会经常用到别的平台，所以跟师兄请教了TestBase的用法。这样就可以在本地测试，在连接别的平台后得到的结果是否正确。又多了一个测试技巧，很有帮助。

2017/7/11

需求开发详情：

1. 项目进展页面开发。目前已完成提交描述功能。可以在页面的输入框提交描述，并执行数据库的插入操作。
2. 后端数据的处理。给数据库增加了一个parent\_comment\_id字段，用来描述其父评论是哪一条记录。编写递归程序，得到当前评论的所有子评论，作为二级评论，列在当前评论的下面。

感想：

1. 这个功能要考虑的东西很多，想要做好还是挺有挑战性的。首先是数据结构的选择。评论是树形结构，每个人都可以互相评论。综合考虑了一下，觉得用二级结构比较合适。如果有两个人在聊天式的互相评论，那层级结构就很麻烦了。所以用A回复B的形式是比较合适的。
2. 后端最终得到的数据结构是JSONArray的形式，里面存放JSONObject，JSONObject里面嵌套全部子评论的JSONArray。在前端遍历的时候没有看到结果，最后发现是JSONArray没有用阿里的，JAVA自带的JSONArray没有实现迭代方法。这个完全没有意识到，还是得细心一点。

2017/7/12

需求开发详情：

1. 已经实现评论展示功能和评论回复功能。大概页面如下：



采用二级结构。一级评论为父评论，修改了递归程序，把父评论下所有的子评论都归位二级评论，然后通过A回复B的形式来区分孙子评论。这样扁平化比较好看。

1. 为了更接近商业性网站的评论模式，把之前projectOverview里面的架构推翻重构。下一步要把@的功能添加进去，这样就更完美了。

感悟：

1. 昨天用了合适的数据结构得到了数据。今天调试前端的时候，废了很大的功夫。看似简单的页面，其实用了很多的堆叠样式，确实很考验耐心。在传值的时候遇到了一个麻烦。因为是用双重循环画的评论样式，且父评论，子评论的回复设计为共用一个评论框，所以点击“回复”，要对应同一个回复框。
2. 且回复框的“回复”按钮必须要带其回复对象的id，作为parentId。最终的解决方案，就是在各个元素里面添加属性，利用属性进行传值。理清楚之后，回复功能成功完成了。在传userId的时候，直接用$!userId,并没有得到正确的值，但是通过元素的属性传过来就成功了，有点疑惑，待解决的问题。

2017/7/13

需求进展情况：

1. 应用配置项的增删改查：表的结构选择中，留有逻辑删除的字段status。然后，根据操作去选择requestMapping中的method。原则上说，GET，POST，PUT，DELETE分别对应了查数据，插入数据，更新数据，删除数据。
2. 所有应用，和已配置应用的diamond检验：一致性检验主要是调用diamond提供的接口，通过我们手动输入的dataId和groupId，得到相应的配置信息，最后得到MD5，检查预发和阿里云的环境与线上中心是否一致。
3. 应用配置页面的筛选功能，已完成。根据团队，已配置和未配置选项来选择相应的配置。可以筛选当前项目下的应用配置。两个radio单选框都不选的时候，选择全部的；选择已配置，未配置的，则显示相应配置项。
4. AoneSearchServiceImpl接口方法修改完毕。并且通过写AoneSearchServiceTest测试了相应的功能。在写了一些测试用例之后，得到了对应的结果。
5. 项目进展页面开发。已完成提交描述功能。可以在页面的输入框提交描述，并执行数据库的插入操作。
6. 回复功能的实现。给数据库增加了一个parent\_comment\_id字段，用来描述其父评论是哪一条记录。编写递归程序，得到当前评论的所有子评论，作为二级评论，列在当前评论的下面。整体页面比较简洁。
7. 实现了@用户的功能。根据师兄的推荐，在github上研究了几个@用户的插件Mention.js , At.js. Mention的实验之后发现不支持中文,最后选用At。现在还在完善的阶段。

感想：

1. 正确的定义问题：遇到错误的时候，不能只看表面。比如在写“详情“的时候，前端展示的表格内容顺序错了，中心预发不在第一位。按着直觉，开始搜索关键词“json 顺序”，其实一步步调bug，发现到前端之后，顺序依然是对的，其实是明细表格自带的排序没有取消。
2. 后端最终得到的数据结构是JSONArray的形式，里面存放JSONObject，JSONObject里面嵌套全部子评论的JSONArray。在前端遍历的时候没有看到结果，最后发现是JSONArray没有用阿里的，JAVA自带的JSONArray没有实现迭代方法。这个完全没有意识到，还是得细心一点。
3. 回复框的“回复”按钮必须要带其回复对象的id，作为parentId。最终的解决方案，就是在各个元素里面添加属性，利用属性进行传值。理清楚之后，回复功能成功完成了。
4. 一定要注意，工号默认是6位数，所以不足六位的是默认前面补0，例如“031545”，当后端定义一个String类型的userId的时候，若是需要在AJAX里面传回后端，会发现var userId接受到的是数值类型，所以031545自动转成8进制。解决这个问题很简单，转成String就行了。另外，如果先赋值给属性，会自动转成字符串。

2017/7/17

需求进展：

“@用户”功能的改进，楼层设置改进；“@用户”之后的邮箱通知功能。

1. 上周讨论过之后，飞泰师兄提出了几点很有建设性的建议。首先，@功能不能是只@当前项目组的人，项目进度描述这个大页面，其目的是让其他相关的人也能参与进来，所以暂时把@列表设置为公司所有成员。
2. 楼层的设计。本来我的理解是，评论区是溯源性质的，所以要把最早的评论放在最上面。但是PD的意思是，把最新的评论或者汇报摆在最上面，方便查看。可以接受。
3. @用户之后邮件通知功能。发邮件的功能有接口，难的是如何把@到的人提取出来，传到后端。并且在邮件里用html元素展示。

遇到的问题：

1.得到当前元素下面的所有span子元素。Span只是个例子，可以用map的方式遍历每个子元素，然后做function处理。师兄提醒了我一点，会有不带@的元素出现，所以要判断一下。

2.从前端传递数组到后端。我的想法是用param的方式传，但是没有成功。最后采用JSON.stringify()的方式先把数组转成JSON字符串，然后放到{}中传到后端，然后再用parseArray转成jsonArray。jsonObject也是一个道理。

3.得到用户数据，进行邮件发送时，因为要用html元素展示通知信息。所以要进行拼接。这时候会遇到一个问题。从评论输入框传过来的html元素，其””是没有经过转义的，所以传到邮箱的页面，无法解释。这个问题需要进一步研究一下。

2017/7/18

需求进展情况：

已经完成输入框和回复框的@功能，及发邮件给被@的人的功能。至此，评论区的功能已经基本完成，并且已经上线。当前用户（实习生排除在外）可以在评论框和回复框添加文字并@所有员工，后台会处理要@的userList，然后群发邮件给他们。

感悟：

1.在提交区域是可以把html元素也存入数据库的。但是现在有一个场景，在邮件中显示信息的时候，要显示一小段评论的内容，所以我需要对内容的长度做个限制。但是这些@的元素是span元素，每一个转成字符串后长度为80左右。如果用字符串过滤，是可行的，但是不可控，不容易扩展。因为你没法限制用户@的人数。

所以说，不能为了解决问题而解决问题，多思考思考问题的根源。我可以用text()方法直接得到文字部分的内容，而不需要得到span元素再切分。（html,text,val,attr的区别）

2测试代码的时候发现一些意想不到的漏洞。从前端得到projectId的时候，我把它的默认值设为0L。但是后面的判断逻辑只有判断null的。在项目全景的页面并没有这种问题，因为绝大部分的内容都是从projectID作为入口。

相同的代码在看起来相似的场景下，也是有细节是不同的，要注意。

2017/7/19

需求进展：

1. 需求介绍：新发生高风险预警。打通与Aone的信息交流，当Aone项目中新建了等级高的风险，或者由低，中等级的风险转变成高风险，就需要在鸟瞰发工单信息给对应的跟进人。另外，每30分钟检查一次；数据需要保存数据库。
2. 解决思路，首先我需要得到A one那边的风险数据，这个问题很好解决，用我之前写的接口就可以完成，添加了一些字段；定时任务，这个需要建立相应的processor和task，已完成；得到数据后，在判断为高风险后，需要入库，所以设计并新建了iwant\_report\_project\_alerts. 及对应的DO，DAO。已完成；在入库之前，要调用表单系统，新建表单。返回id后，再一起入库，尚在测试中。
3. 感想：需要调用的接口很多，Aone系统的，工单系统的，这些都需要去认真阅读API。另外，慢慢由原先的大部分时间写代码，到现在的参与讨论需求。这是另外一个阶段，主动的与PD进行交流，而不是按照需求做。另外，这个需求的代码方面出错的较少，侧重业务逻辑。

2017/7/20

需求进展：

1. 主要是对预警功能的测试。因为这个功能涉及到的接口比较多，Aone那边的的api一般比较笼统，其他的东西还需要自己去测试。在阅读了工单系统提供的文档后，发现参数分配者id和提交者id不明确，与师兄讨论了之后，确定了相关字段。并且重构了接口，分成多个模块。
2. 另外，明确什么范围内的项目风险会参与每30分钟一次的检查。首先，项目类型为“项目”，且该项目的创建时间是6个月前到现在。另外风险等级为95。创建时间的计算，本来是想先求出6个月前的日期，然后用criteria筛选。但是这样计算过于麻烦，且6个月只是个大概时间，所以我先得到所有的projectDO，再筛选创建时间与当前时间差小于180天的。

感想：

1.今天最大的收获是程序结构的设计。本来我的预警相关代码都是写在一起的，但是这样有个很大的问题，等以后有其他类型的操作时，会有大量重复的代码。而有的代码抽出来，可以方便别人复用。而且设计类似工厂模式的设计模式也可以方便后来的用户理解。设计一个好的程序架构和结构，还是很考验人的，要多思考思考这块的东西。

2017/7/24

需求进展：

1. 完成了新需求变更的预警功能。从需求变更表里面查找最新生成的需求变更，然后做预警，发送钉钉到指定的丁丁群。取需求变更数据的原则是，每30分钟查找最新的，也就是与当前时间差小于等于30分钟的需求变更数据。在DateUtil写了一个计算相差分钟数的函数，可以复用。
2. 在diamond写了token的配置项。暂时作为测试用。因为，一方面，token这种秘钥性质的东西，最好不要放在diamond配置里面；另一方面，之后会在另外一个页面把token传过来，所以先暂时这样。
3. 修改了里程碑的样式，添加了一个“超时完成”的选项。虽然里程碑已经完成，但是属于超时完成，要在里程碑上方做个“超时”标记。另外，修改了原先的换行模式，让里程碑都在一行显示。

4. git pull 的时候，如果没有产生冲突的话，会自动merge commit。而如果有冲突，会中断commit的过程，等待冲突被解决后，可以手动commit。感想：

前端调试真的很琐碎，尤其是调试扒过来的前端代码，很痛苦。之前为了与Aone的里程碑保持一致，避免用户在鸟瞰看到里程碑后对应不上，就仿照Aone的形式写的里程碑。但是它的源码肯定是没法得到的，所以改起样式会不理解他们的思路。所以我觉得，以后再涉及到对接，保持一致的情况，可以画出它的样式，但尽量不仿照它的结构。

2017/7/25

需求进展：

（1）上午在调试的时候发现了项目描述部分的bug，因为现在会有@，等html元素，所以在传递描述给悬浮框的时候，出错了。原因是我把描述放在了属性里面，这样转义会出现问题。最后采用每条评论对应一个悬浮框的解决方案，比解决转义要节约时间。

（2）下午先把“新增高风险”预警的方式改为了钉钉通告，然后与师兄讨论了相关的业务问题。这部分的业务逻辑比较繁杂，最后的结论是，对于风险的“钉钉通告”，通过风险项id和handleMethod来查找数据库，如果有，则不发送；没有则发送，且根据钉钉传回来的参数决定是否入库，入库时，默认status“已完成”。

而对于风险的“工单驱动”，每三天检查一次，如果该风险下面的评论满足一定规则，则判断该风险为待发工单风险。因为这种情况下有可能会有多条工单生成，所以规则是得到最新的那条工单信息，如果该工单已完成，或已取消，则新建工单；若未完成，则不操作。

（3） 针对上述预警表中，status的问题，下一步决定采用监听器的形式， 监听工单系统的工单处理状态，如果更新了，就修改预警表中的status字段。

感想：

最近接触的业务逻辑比较多，技术上的问题基本都可以解决。遇到不清晰的地方，还是要多沟通沟通，讨论出最佳的解决方案，沟通很重要，有效率的沟通可以提高效率。

2017/7/26

需求进展：

1.实现了监听工单系统的处理状态，然后实时更新预警表中风险status字段的值。

metaQ的接口已经写好，研究了一下，可以得到我需要的工单id和tags。根据不同的tag，会给status字段设置相应的状态值。

问题和感想：

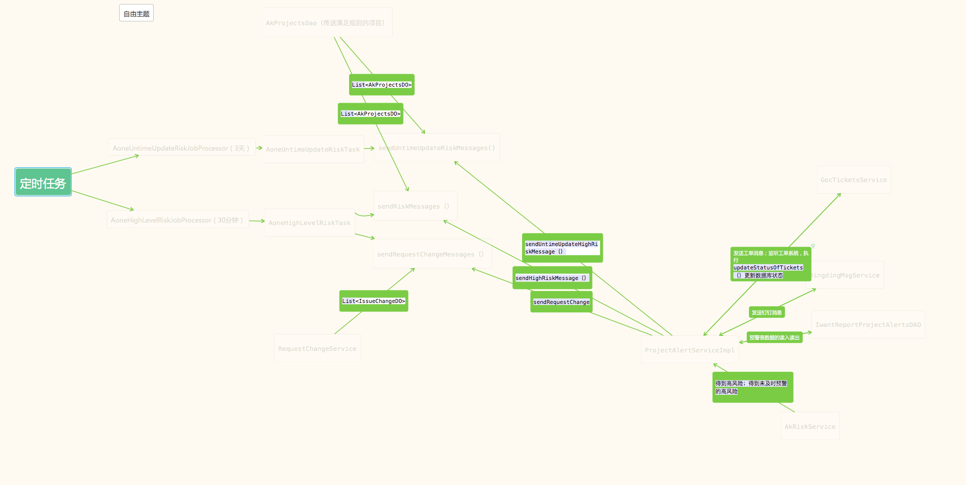
需求思路还是有的，但是遇到了很多麻烦。

1. 业务场景。工单系统的状态是比较多的，而我们自己的数据库，为了减少不必要的误解，把这些状态归为五类，（1）已完成： 工单完结；（2）已取消：工单退回，工单撤销；（3）未处理：是入库的默认状态，对应工单的create，但是不做监听；（4）处理中：工单受理，工单转交；（5）已处理：工单解决 这样cover了工单的所有状态，也避免了误解。
2. init()函数作为metaQ的监听接收端口，是唯一的，不能在别的service里面再建init()函数，否则metaQ就产生歧义，找不到位置。然后在测试metaQ消息发送的时候，发现没有接收到，其实是控制台没有选择all，一直以来都默认只显示异常，没想到这一点。
3. 之后的工单创建异常，发现提示信息是提交者没有在服务组里。之前和师兄讨论，工单系统在发现提交者不在服务组后，后自动添加此人；但是今天和师兄聊到这个时，才知道已经改为手动添加了。保持信息实时更新很重要。
4. 更新入数据库的遇到了小麻烦，但是很快就解决了。

2017/7/27

需求进展：

1.上午帮科怀新增了表字段，并且修改了所有相关的代码。因为项目进展那边需要用到父项目id，方便操作。然后把预警这一块的逻辑整理了一下，因为之后肯定是要模块，可配置化的，所现在先考虑哪边可以抽象出来用。大概的思路图是这样的：



还是有很多优化空间的，因为当前item是和发送方式绑定的，未来打算把四种发送类型定义成tag，用户可以随意选择发送方式。

2.下午接了一个新需求，发送超时任务的预警信息。形式和风险的差不多，注意下，比较的规则以天为标准，在同一天就不算超时。主要的卡点在如何得到预期完结时间。一开始以为是expectedAt字段，但是试了很多次，每条的该字段都是null。最后咨询了下Aone的师兄，他解释说这是自定义字段，都放在了cf\_list里面了。

这样的话，有两点疑惑：（1）我理解这样做，是因为有的issue用不到预期完结时间这个字段，所以作为自定义属性。为什么不采用覆盖的方式呢？如果加的字段不多的话，是可以接受的吧。（2）cf\_list是一个list<class>形式。这个就不太合理了吧，自定义类型很多，我想得到“截止日期”，还是得遍历整个list。所以就很矛盾了，如果字段少，那就覆盖；如果字段多，传个map，方便查找。

3.晚上听到一个好消息，Aone那边提供了监听方法，不用我们定时去拿数据了。

4.今天听了科怀的分享，还是很有收获的，每个人遇到的问题都不尽相同，多听听可以帮助自己填补漏洞。

2017/7/28 周报

需求进展：

1.完成了新需求变更的预警功能。从需求变更表里面查找最新生成的需求变更，然后做预警，发送钉钉到指定的钉钉群。取需求变更数据的原则是，每30分钟查找最新的，也就是与当前时间差小于等于30分钟的需求变更数据。在DateUtil写了一个计算相差分钟数的函数，可以复用。

2.里程碑样式优化。修改了里程碑的样式，添加了一个“超时完成”的选项。虽然里程碑已经完成，但是属于超时完成，要在里程碑上方做个“超时”标记。另外，修改了原先的换行模式，让里程碑都在一行显示。

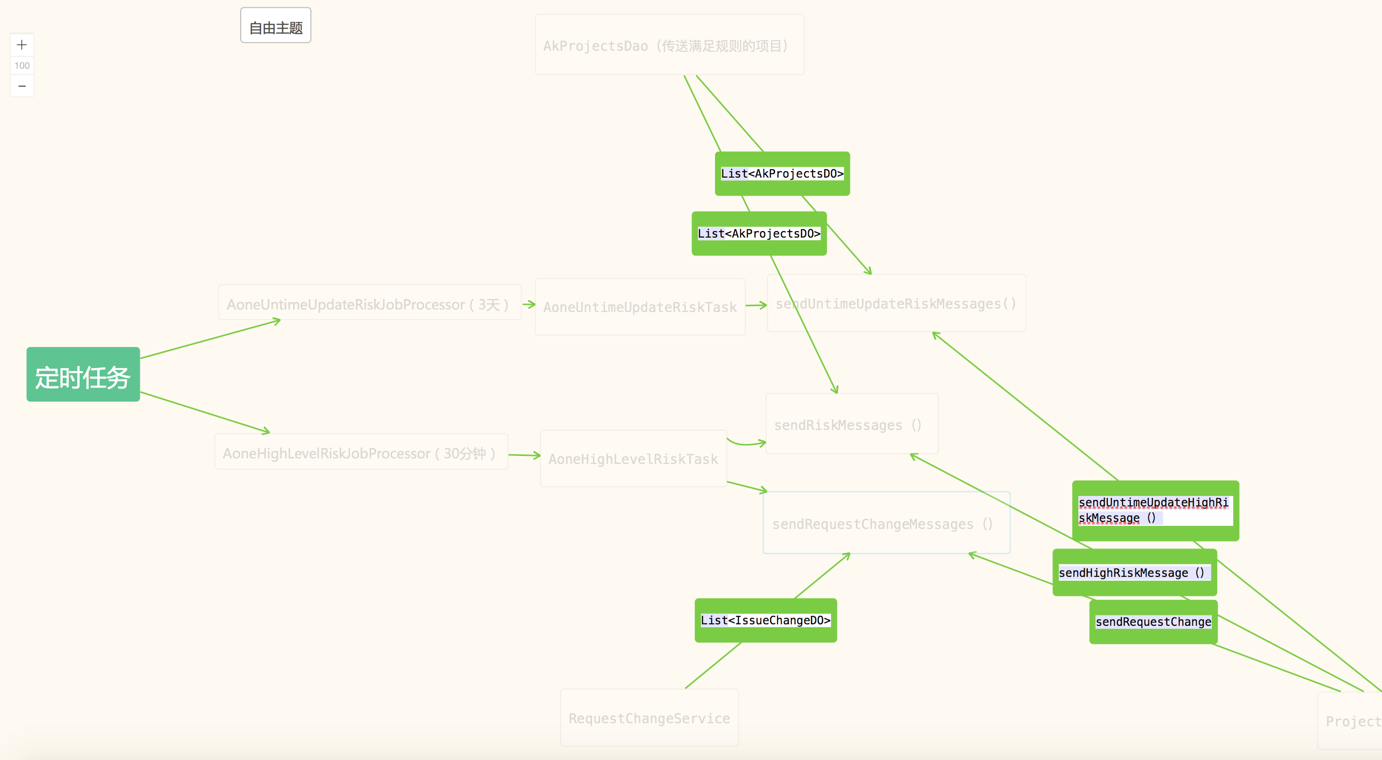
3. 实现了监听工单系统的处理状态，然后实时更新预警表中风险status字段的值。metaQ的接口已经写好，研究了一下，可以得到我需要的工单id和tags。根据不同的tag，会给status字段设置相应的状态值。

4. 发送超时任务的预警信息。形式和风险的差不多，注意下，比较的规则以天为标准，在同一天就不算超时。主要的卡点在如何得到预期完结时间。一开始以为是expectedAt字段，但是试了很多次，每条的该字段都是null。最后咨询了下Aone的师兄，他解释说这是自定义字段，都放在了cf\_list里面了。

5.目前Aone提供了新的监听接口，对于新增高风险，需要走新的接口。Dts相关的先暂时保留。

感想：

很充实的一周，基本做完了一整个预警模块：



未来希望把这块优化的更好，能用监听方式的尽量用监听，几种发送方式要模块化，可配置化。另外，昨天的疑惑，今天Aone新发布的接口中，返回结果已经改为Json了，而且预期时间也写成expectedAt字段了。看来这个问题Aone早就注意到了，很棒！

新人接入文档：

1. 前端

(1)样式定义：一般会在前端的style里面定义。但是如果样式非常多，就不适合写在html里面了，这时候可以单独写出来，并在html里面添加链接，如：

<**link rel="stylesheet" href="/static/css/iwatch/projectOverViewStyle.css" type="text/css"**/>

（引用外部jsp文件与之类似。例如，常用的**jquery.dataTables**）

(2)样式层叠：会经常遇到这样一种情况：设置好样式之后，页面却没有反应， 没有到达自己预期的效果。这时候一般有两种情况：1.缓存。 2.定义的样式重叠了。可以右击，用页面的“检查”，查看起作用的是哪一个样式。详见：<http://fullstack.cainiao-inc.com/study/detail/javascript/debug?spm=a2d1e.8232163.0.0.uj82GE>

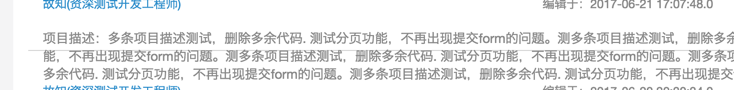
（javaScript中的前端调试技巧）

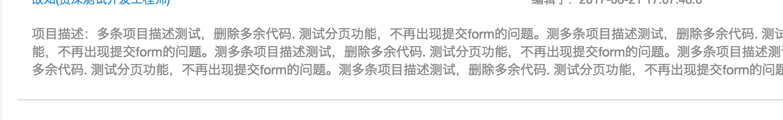
(3)velocity的简单逻辑：常用的“#if #else #end #foreach”,使用方式灵活，可以在标签内部使用。另外注意, **$!是不过滤输出，$是过滤输出，!$用来判断null. 处理简单逻辑的时候，可以使用上述逻辑。不推荐在前端使用大段的复杂逻辑。$velocityCount输出当前迭代的次数。**

(4)如果只是想前后端传值，而不需要页面展示，可以使用隐式文本域。

<**input type="hidden" name="projectId" id="projectId" value="$!projectID"**/>

(5)清楚浮动：例如，在div中填充数据时，出现文字溢出的情形，这时候可以添加 <**div style="clear**:**both"**></**div**> ，清除浮动。





(6)画图：画图有两种方式，一种是sql查询方式，一种是javaBean的方式。javaBean的方式要注意格式（“bean的注解名”，“方法名”）。另外，在前端引用图时，reportTool源码中，已经对格式做了设置，（默认是占用一半）不要跟自己的格式冲突了。

2.后端

(1)sql语句：用sql语句查询数据时，count操作得到的值默认是BigInteger，avg得到的是BigDecimal。如果数据需要二次处理，一定要注意数值类型转换的问题。

(2)sessionFactory在循环里面添加DO，有的东西就失效。

(3)hotswap的热部署范围：

3.1目前的 Java 虚拟机只能实现方法体的修改热部署，对于整个类的结构修改，仍然需要重启虚拟机。（类的结构是指哪些呢）

(4)查询的三种方式，sql，hql，criteria。

区别：sql的查询是直接走的数据库。Hql有自己的缓存机制，但是最终也是生成sql取查询。

(5) 八进制：有前缀0o或0O的数值，或者有前导0、且只用到0-7的八个阿拉伯数字的数值。如果不全是0-7，并不会转成八进制，但是前面的0会自动去掉，对于userId来说是不合适的，因为它是String类型，这样就查不到该DO。

(6)数据库设计问题，除了id，其他最好全设置为可空，由后端来处理空值的问题。留言类的数据，设置为text，因为没法限制用户输入多少字数。（但是为了防止用户误操作，应该设置一定的长度。）

1.System.arraycopy方法：如果是数组比较大，那么使用System.arraycopy会比较有优势，因为其使用的是内存复制，省去了大量的数组寻址访问等时间

如果是复制一个一维数组，那么改变复制后的数组并不影响原数组。但是如果复制一个二维数组，那么改变其中任何一个数组，那么另一个的值也发生了变化

因为二维数组实质是，装着一维数组引用的一维数组。既然是引用，拷贝之后指向的是同一片区域，所以修改新数组的值，原数组的值也会被修改。

备注：

在项目构建时，利用Maven的“约定大于配置”的思想，可以比Ant脚本构建项目省去不少配置文件的内容。而且一个项目可能依赖于其他的项目和第三方的组件才能顺利完成，Maven提供了仓库的概念

，让这些依赖项放进仓库中，项目想要从仓库中去取

json格式问题

com.alibaba.fastjson.JSONException: syntax error, pos 364, json : {

"options": {

"title": {

"text": ""

},

"xAxis": {

"categories": []

},

"yAxis": {

"title": {

"text": ""

}

},

"legend": {

"layout": 'vertical',

"align": 'right',

"verticalAlign": 'middle'

},

"plotOptions": {

"series": {

"pointStart":

}

},

"series": []

}

}

1. 需求：设计一块评论区，可以实现基本的评论，回复功能。类似下图所示：



2.解决思路：

* 评论框负责插入评论功能，走最基本的save路线。
* 不涉及评论的更改，删除功能。
* 从数据库中拿评论数据，并进行展示。考虑到回复的功能，所以需要在数据库中增加parentId字段。同时在父评论下面展示所有子评论。
* 回复区。需要在每一层评论下面添加评论区，方便用户回复。且考虑到冗余度，每层只设置一个回复区。
* @用户的功能需要寻找插件。手写的话没有必要，github有很多开源的插件。
* @用户之后要给用户发送邮件，内容要用html元素来展示。

3.数据的生成

     为了得到评论数据，需要先设计一种好的数据结构来存放。和师兄讨论后，我采用了JSONArray里嵌套JSONObject的方式。类似于以前的List<Map< , >> 形式。优点是使用方便，且传到前端不用再做解析。（import com.alibaba.fastjson.JSONArray; 用公司的JSONArray，java自带的没有实现遍历功能）  另外，具体到收集数据，有两种方式：

（1）多叉树形结构：这种方式是以前的评论区经常用的方式。父评论下面有子评论，子评论下面又有子评论，通过每层缩进的方式来表达父子关系。递归时，每个子评论带List属性，存放它的子孙评论。

（2）二级结构：这种方式现在比较普遍。每层只有一个父评论，其所有涉及到的子孙评论都作为一个层级放在子评论区，通过 “A 回复 B”的形式来表达关系。递归时，只有父评论带List属性，存放所有子孙评论。（ProjectCommentServiceImpl）

4.前端的展示

  数据处理完成后，就要构思前端的展示设计。可以参考别的网站的，也可以自己写，但是要注意层次。

（1）设计时，把每一层的评论分为三层。“评论区”，“回复区”，“回复框”。（**comment，reply-box，release-reply**）

（2）为了减少冗余，每一层的评论，他们的回复功能共享一个”回复框“。为了实现这个功能，用父评论的id作为统一的标准。

（3）效果的渲染。

5.发邮件的格式。

 为了增强用户的体验，发邮件的时候可以用html元素包裹后来发送。



1. 前端

* 样式定义：一般会在前端的style里面定义。但是如果样式非常多，就不适合写在html里面了，这时候可以单独写出来，并在html里面添加链接，如：<link rel="stylesheet" href="/static/css/iwatch/projectOverViewStyle.css"type="text/css”/>（引用外部jsp文件与之类似。例如，常用的jquery.dataTables）
* 样式层叠：会经常遇到这样一种情况：设置好样式之后，页面却没有反应，    没有到达自己预期的效果。这时候一般有两种情况：1.缓存。 2.定义的样式重叠了。可以右击，用页面的“检查”，查看起作用的是哪一个样式。详见：<http://fullstack.cainiao-inc.com/study/detail/javascript/debug?spm=a2d1e.8232163.0.0.uj82GE>

         （javaScript中的前端调试技巧）

          评判条件：

           （1）由于继承而发生样式冲突时，最近祖先获胜

                 例如：body {color:black;} p {color:blue;}

<body>

 <p>

<strong></strong>

</p>

</body>

（2）继承的样式和直接指定的样式冲突时，直接指定的样式获胜

            strong {color:red;}

（3）直接指定的样式发生冲突时，样式权值高者获胜（相同则选后者）

标签选择器：1。

        类选择器：10 (‘.XXX')

               ID选择器：100 (‘#XXX')

                内联样式：1000 (外部样式表，内部样式表，内联样式，例：style)

                伪元素（：first ：after）：1

伪类（：link）：10

                 例如：#release-reply .textarea-wrap a   权值为111

            （4）!important 强制选择某个属性。（一般不用）

* velocity的简单逻辑：常用的“#if #else #end #foreach”,使用方式灵活，可以在标签内部使用。另外注意, $!是不过滤输出，$是过滤输出，!$用来判断null. 处理简单逻辑的时候，可以使用上述逻辑。不推荐在前端使用大段的复杂逻辑。
* $velocityCount输出当前迭代的次数。(例如：留言板，评论楼层数)velocity的加，减法#set($floor=$!{jsonArrayOfComment.size()}-$!{velocityCount}+1)                                <a  class="com-floor">${floor}F</a>
* 如果只是想前后端传值，而不需要页面展示，可以使用隐式文本域。<input type="hidden"  name="projectId" id="projectId" value="$!projectID"/>
* 清楚浮动：例如，在div中填充数据时，出现文字溢出的情形，这时候可以添加 <div style="clear:both"></div> ，清除浮动。
* reportTool源码中，已经对格式做了设置，不要跟自己的格式冲突了。
* 缩放页面的时候会暴露一些问题，写完一个页面可以通过缩放的方式来检查效果。
* 前端效果：fadeIn(), fadeOut(), slideDown(), slideUp()，动画animate

2.后端：

* 天数计算。首先最好使用库函数去计算，因为自己写的逻辑会有各种边界问题。例如，（1）若预期结束时间默认都是0时0分0秒，那么逾期几秒也算是延期1天。（2）若预期时间没有默认格式，要根据是否超过当天来计算，单纯的计算差的毫秒数确实是不合适的。(AKProjectsDaoImpl,DateUtil,DateTool)
* 由sql语句的count操作得到的值默认是BigInteger，avg得到的是BigDecimal类型。数据处理的时候注意类型转换的问题。(例如：BigInteger，Integer的相互转换)
* ToolUtil，BaseDetailsServiceImpl的使用。计算百分比。（ProjectOverViewService）
* 数据库操作。有三种方式（1）sql （2）Hql （3）Criteria。（使用sql的时候要注意，sql是直接按数据库中的表名去查询，会跳过DO。另外无法使用高级sql，如视图。）
* JSONArray,JSONObject的使用。前端用stringify传到后端。后端再根据其类型，转换成array或object。（应用：数组从前端传到后端）
* map函数。遍历元素，然后得到想要的结果。 user= $.map( $('.wmd-input').find("span"), function(elem) {   if($(elem).text().toString().startsWith('@'))          return $(elem).text().slice(1);       else   return $(elem).text();});  且不要做完美假设。要考虑到用户误操作的情况。
* StringEscapeUtils.*escapeJava().*当需要对一个字符串中的一些字符进行转义时，会用得到这个函数。（projectCommentService）

3.git：

* 慎用 git  reset  --hard  commit\_id ，会把工作区的内容也回退掉。万一用了这条命令，可以git  reflog , 查到commit id，然后checkout 就可以了。
* git grep –in “report/view\_config” ，查找文件所在位置。

测试一下git有没有成功。