CI开发与维护指导

|  |  |
| --- | --- |
| 修订记录 | |
| 2017-4-19 Ting YU | Init初稿 |
|  |  |

写给负责CI的人，他的职责包括：新功能的开发、现有功能在代码层面上的维护、协助维护CI的人定位问题等。

## 持续集成

持续集成continuous integration，简称CI，是一种软件开发实践，即团队开发成员经常集成他们的工作，每天至少集成一次。每次集成都通过自动化的构建（编译、出包、测试）来验证，从而尽早地发现错误。每个环节都是自动完成，无需太多的人工干预，较少重复过程以节省时间和工作量。CI的目的，就是让产品可以快速迭代，同时还能保持高质量。

1. 稳定性

包括构建的稳定和用例的稳定。必须保证构建的成功，如果失败，修复构建过程的错误是优先级最高的工作。如果构建常常因为未知原因随机失败，每次失败都需要投入人力去定位。其次是，用例的失败有可能是用例的问题，可能是产品的问题，尽可能得减少用例的问题。

1. 尽可能地自动化

支持更多类型的自动化，释放人力，将自动化测试的范围覆盖地更广泛，也可以更全面地测试和发现问题。

## 代码结构

db每个分支有各自的调度代码， pg也有自己的调度代码。

|  |  |
| --- | --- |
| 各分支 | 调度代码 |
| db1.12 | http://192.168.20.11/sequoiadb/trunk/misc/ci |
| db2.6 | http://192.168.20.11/sequoiadb/branches/engine\_1.12/misc/ci |
| db主干 | http://192.168.20.11/sequoiadb/ branches/sequoiadb\_2.6/misc/ci |
| pg | http://192.168.20.11/sequoiadb/otherdb/postgres/misc/ci |

db主干的代码重构过，CI添加新功能主要在主干上添加。db1.12、db2.6、pg的没有重构过，不会经常变动，只处于简单的维护状态。

现主要介绍db主干的代码结构：

-doc //放置文档

-mkrelease //放置打包的xml脚本

-lib //放置第三方库、jar包

-src //放置调度脚本

|-build.xml //入口

|-conf //放置配置文件

|-compile //编译阶段

|-install\_deploy //安装部署阶段

|-test //执行测试阶段

|-exec\_testcase //执行不同类型的用例

|-solve\_result //处理构建结果阶段

每日构建的流程：编译—>安装部署—>执行测试—>处理构建结果，对应的src目录也按照这个划分为compile、install\_deploy、test、solve\_result。

另外还有build.xml文件build.xml是整个src的入口，可以理解为main函数，ant默认找的build.xml文件，因此将它作为入口。在build.xml中可以找到所有的构建，例如compile\_build、normal\_test\_build等。

conf目录放置配置文件，包括ant的配置、安装部署的配置。把配置提取出来，而不是写在脚本里面的好处就是，需要修改配置时，不需要去翻代码，直接改conf方便很多。

## 工作日志

当CI出现一个问题，如jenkins曾经出现界面无法跳转的情况，负责人A花了时间后终于解决这个问题后，但没有留下文档记录问题和解决方法。可能过了一段时间，CI再次出现这个问题，负责人A很可能忘了如何解决的。甚至可能CI交给另外一个人B负责，A已经离职，这时再次出现界面无法跳转的问题，负责人B只能重新花时间解决问题。

工作日志就是为了防止这情况的发生。把CI维护时遇到的一些奇怪的复杂的问题，记录下来保存在misc/ci/doc/work\_log下。随着时间的增长，工作日志的内容也在增长，宝贵的经验可以不断累积下来。

工作日志的格式可以参考work\_log下已有的日志，主要记录问题详情、解决方法、解决过程。新手可以从前人写的“解决过程”中，学习到如何解决一个棘手的问题；也可以给写日志的人加以反思，走了哪些弯路，在哪里多花费了时间，以后遇到类似的问题是否可以少走弯路。

## 遵循规范

ant代码有编码规范misc\ci\doc\ci\_improve\_plan\ant\_code\_standard.docx，新增工程、slave节点、机器名也有各自的规范misc\ci\doc\name\_rule.txt。

一段时间内CI通常只有一个负责人，下一阶段换了另外一个负责人，新的人需要维护前人写的代码，统一的编码风格有利于长期维护。在维护过程中，发现编码中有什么坑，需要提醒以后的CI负责人，都可以加到编码规范中。

CI工程的命名须要遵守规范，这样让使用它的用户看一眼就知道这个工程的用途。