持续集成及CruiseControl

技术交流

罗时飞
宇信易诚广州研究所
http://www.yuchengtech.com

适合群体

- 具有一定的软件研发经验
- 熟悉研发工作生命周期中的各个阶段
- 对持续集成感兴趣的人士

自我介绍

- 将近10年的软件研发经验，主攻Java/Java EE平台、BI、敏捷方法、开源技术
- 最近几年，参与到银行信息化建设工作当中
- 出版过10本技术书籍。近期已出版《敏捷持续集成(CruiseControl版)－高效研发之道》一书

交流内容

- 各方的苦恼
- 区别计算机与人类智慧
- 何谓持续集成
- CruiseControl介绍
- 案例研究
- 总结
- 回答问题

各方的烦恼

- 软件研发中存在的角色：DBA、开发经理、架构师、部署工程师、功能及负载测试人员、……
- DBA:
- 开发经理:
- 架构师:
- 部署工程师:
- 功能及负载测试人员:

区别计算机与人类智慧

- 划清重复（体力）劳动与智力劳动间的界线
- 智力劳动，人类智慧（软件研发团队）不可用计算机替代
- 重复劳动，应该让计算机去完成，因为它会真正任劳任怨地工作
  - 既然如此，软件研发团队应该让计算机的价值最大化！
- 可回归性
  - 即尽可能将重复劳动交给计算机去完成，也就是说研发人员（软件从业人员，包括CI集成人员）需要最大化计算机的价值。自从人类发明计算机以来，它们最擅长的，就是完成重复劳动，而且毫无怨言地工作。与此同时，我们要善于将软件研发期间涉及到的各种重复劳动提炼出来，并将这类劳动托付给计算机完成，进而使得我们能够尽可能地把宝贵的时间投身于智力劳动上。这也体现了现阶段计算机与人类间社会分工的差异性

提炼重复任务—实现可回归性的重要前提

- 在软件研发的整个生命周期中，不同阶段的大量工作在经过一定的分析后，其中的很大一部分工作内容都可以抽取为重复任务，具体如下：
  - 对于数据库相关任务而言，DBA（数据库管理员或设计师）可能要经常重建数据库，并初始化数据库。如果把用于数据库重建的DDL脚本外在化管理，并作为配置管理系统（Software Configuration Management，SCM）的配置项，则数据库的重建工作将不再是只有DBA才能够完成的工作。同理，用于初始化数据库的相关DML脚本也可以单独整理出来，并存放到SCM中，比如ClearCase、VSS、CVS、Subversion。此后，这类DDL和DML脚本的执行便可丢给计算机完成，继而DBA能够把自身宝贵的时间和精力投身于数据库本身的设计、优化上，而不用再去浪费时间重复执行它们

提炼重复任务—实现可回归性的重要前提(续)

- 接上:
  - 对于开发人员而言，他们可能要经常手工运行自身写好的单元和集成测试代码。而且，随着项目的往前推进，这类测试代码的数量往往是惊人的，如果要开发人员或项目组派专职人员来手工运行它们，则效果可能而知。这类测试代码的运行完全有理由托付给计算机去自动完成
  - 对于架构设计师而言，他们要经常检查开发人员完成的开发工作是否符合设计初衷，比如JavaDoc注释是否符合要求、各个Java包是否遵循了设计约定、Java包间的使用是否符合设计规范、等。这些工作都应该属于重复任务的范畴，比如借助JDK内置的javadoc.exe命令行工具能够自动生成JavaDoc文档、通过JDepend评估静态代码的设计和开发质量。类似的工具还非常多，比如商业软件Fortify，开源的Checkstyle、PMD、FindBugs、等等

提炼重复任务—实现可回归性的重要前提(续)

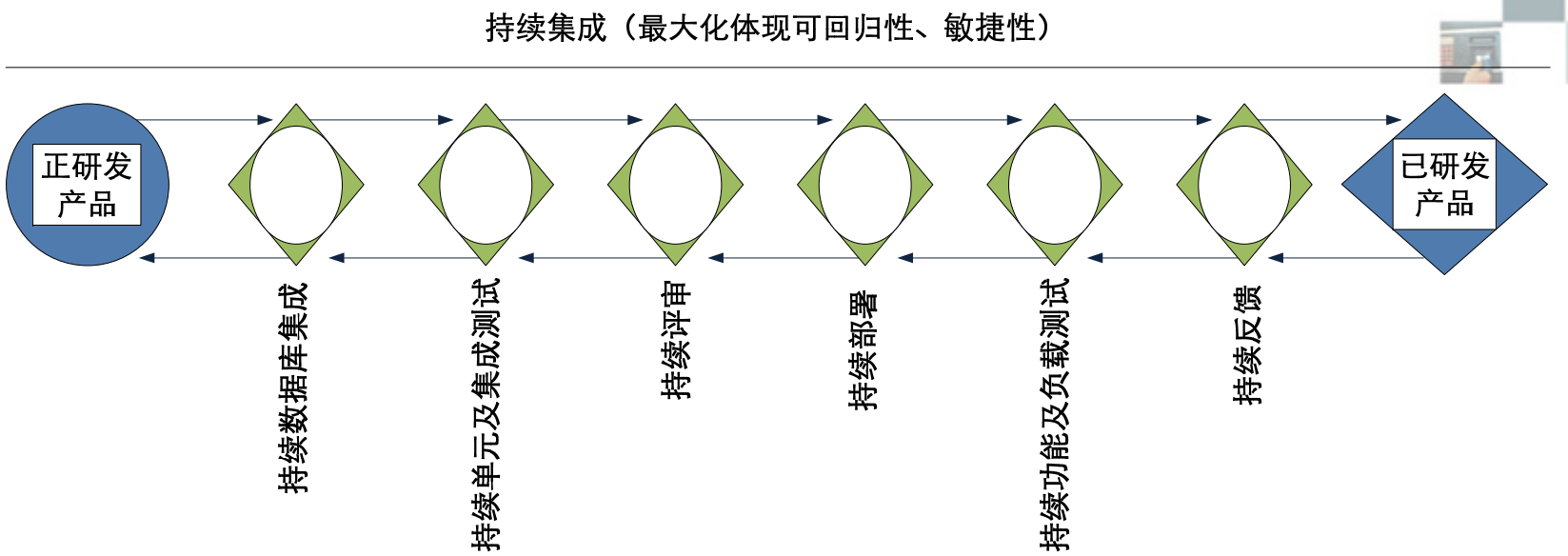
- 接上:
  - 对于部署工程师而言，完成目标应用的打包和发布工作往往是他们的重要工作内容之一。Apache Ant和Maven内置了一流的JAR/WAR/EAR打包支持，何不把应用的打包工作丢给它们呢
  - 对于功能及负载测试工程师而言，他们可以借助于OpenQA Selenium、Apache Jakarta JMeter等自动化工具完成功能及负载测试工作。就工作性质而言，功能测试及负载测试是较为乏味的工作内容，而且许多软件研发团队的这类测试工作局限于手工层面，因此我们急需改进现有的功能及负载测试工作方式
  - 最头疼的问题是，如何快速地将上述各种重复工作的执行结果反馈到项目涉及人员手里，比如借助电子邮件（E_mail）、聚合内容（RSS）、FTP、Socket、HTTP等途径
  - 等等，这里还没有列举出来的。各位可以举一反三、再不断补充

敏捷性—奔向成功的致胜法宝

- 敏捷性：
  - 即在保证目标任务完成质量的前提下，以最快的速度交付出工作成果。这实际上体现了效率与质量的并重。借用现有成熟的技术或工具提炼重复任务，这就是一种敏捷性的体现。值得注意的是，软件研发期间，除重复任务的提炼外，那些不能够作为重复任务的工作的完成也可以用敏捷性来衡量。比如，软件项目的架构设计本身，这项工作是需要经过富有经验的业务和技术架构师的深思熟虑。即使如此，在很多场合，架构设计工作实际上也可以交给一些成熟的开源框架，比如Spring Framework，这类框架已经几乎把目标应用的架构设计确定下来了
  - 可回归性和敏捷性相辅相成，可回归性是基础，没有可回归性就没有敏捷性；敏捷性是提升可回归能力的重要法宝。为了使得团队和软件研发项目的可回归性、敏捷性达到最优，必须实施持续集成。借助于持续集成（Continuous Integration，CI），重复任务的执行能够变得更加敏捷

何谓持续集成

- 下图展示了持续集成:

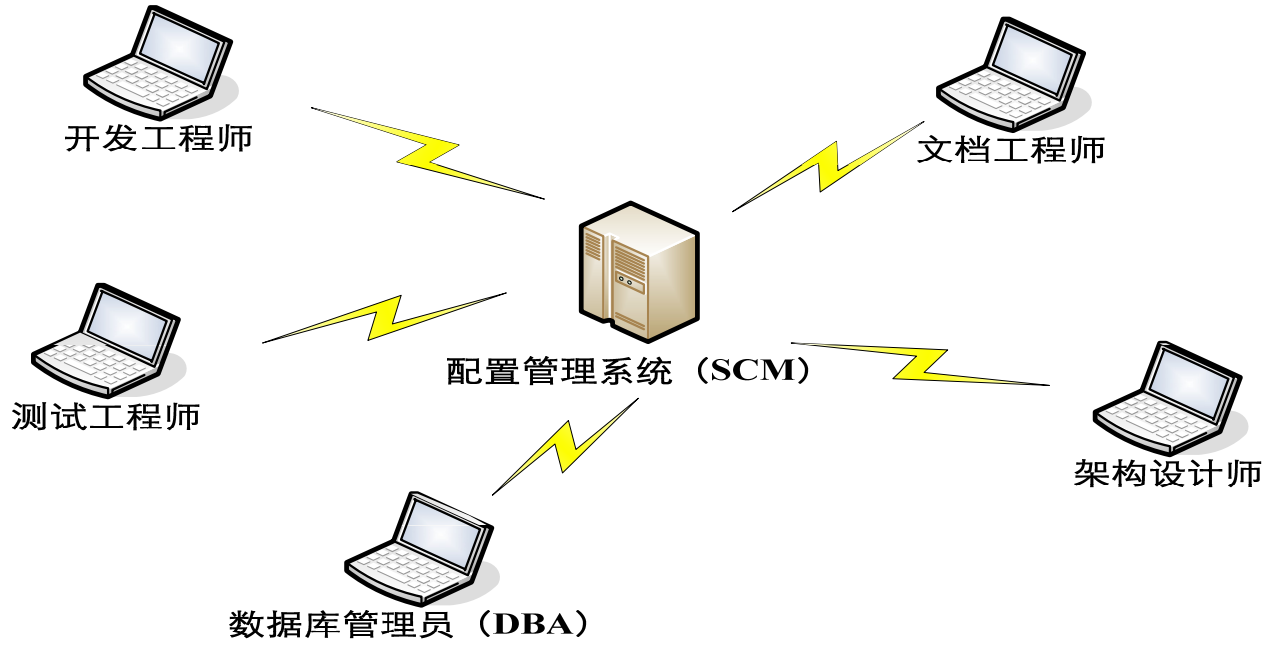


持续……

- 持续数据库集成：定期基于SCM中的最新SQL脚本（包括DDL和DML语句）完成数据库的重建和数据初始化工作。比如，借助Apache Ant内置的<sql/>支持
- 持续单元及集成测试：从SCM中检出最新的Java代码，并自动执行其中的单元及集成测试代码，最后给出相应的测试结果。如有需要，还可以自动给出代码覆盖度分析结果，比如Clover、Cobertura、EclEmma、TPTP、Coverlipse
- 持续评审：基于SCM中的最新代码进行各方面的自动评审工作，比如代码风格和规范评审、架构设计评审、详细设计（JavaDoc）评审。比如，基于Checkstyle、PMD、JDepend、JavaNCSS、Fortify、javadoc.exe等自动进行
- 持续部署：将构建工作的产出物持续部署到目标测试环境，比如借助Apache Ant、SmartFrog
- 持续功能及负载测试：比如，对刚部署到测试环境的目标应用进行自动化功能测试及负载测试。借助Selenium RC、Apache JMeter等技术可以自动完成这类工作
- 持续反馈：通过各种渠道将上述各种持续性工作的具体结果分发出去。比如，邮件服务器、RSS、FTP服务器、HTTP服务器、Socket、等等

敏捷配置管理

- 很明显，SCM工具在实施持续集成期间扮演了非常重要的角色。如果整个研发团队中的各个成员没有将各自的工件（代码、脚本、DDL语句、等等）提交到SCM配置库中，则持续集成服务器将没有办法进行构建工作。即使是产品研发（包括项目研发）不实施持续集成，SCM工具也是不可或缺的

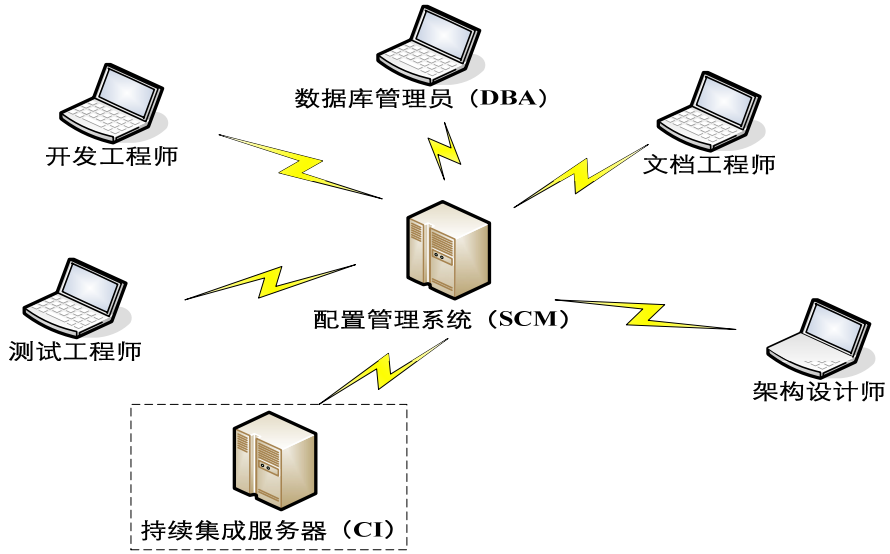


衍生于CI

- 针对不同客户、产品、项目，不同研发团队可能会采用不同配置管理工具（Subversion、ClearCase、CVS、VSS）。传统的SCM工作，仅仅包含常用的基线管理、配置状态报告、配置审计报告、变更管理等。至于配置库中的工件是否有问题，上述工作内容并不能告知，比如代码是否能够正常编译、产品文档是否能够正常打开。敏捷配置管理（Agile CM，ACM）克服了这些缺陷。ACM，不仅存储静态工件（黑盒），还动态地、集成地将配置库中工件本身（白盒）存在的问题不定期暴露出来

驱动CI的引擎—CI服务器

- 引入的CI服务器将持续完成各种重复任务，比如持续数据库集成、持续单元及集成测试、持续评审、持续部署、持续功能及负载测试、持续反馈
- 一旦项目实施持续集成后，被研发产品的集成工作便经常发生，而且随时都能够构建出最新的可供部署和测试的产出物（或JAR或WAR或EAR包，甚至是单独的可执行文件）。由于上述各项工作都将持续发生，各种潜在的问题将被持续地暴露和跟进，进而能够逐渐改善软件产品的质量，并有效地降低项目风险

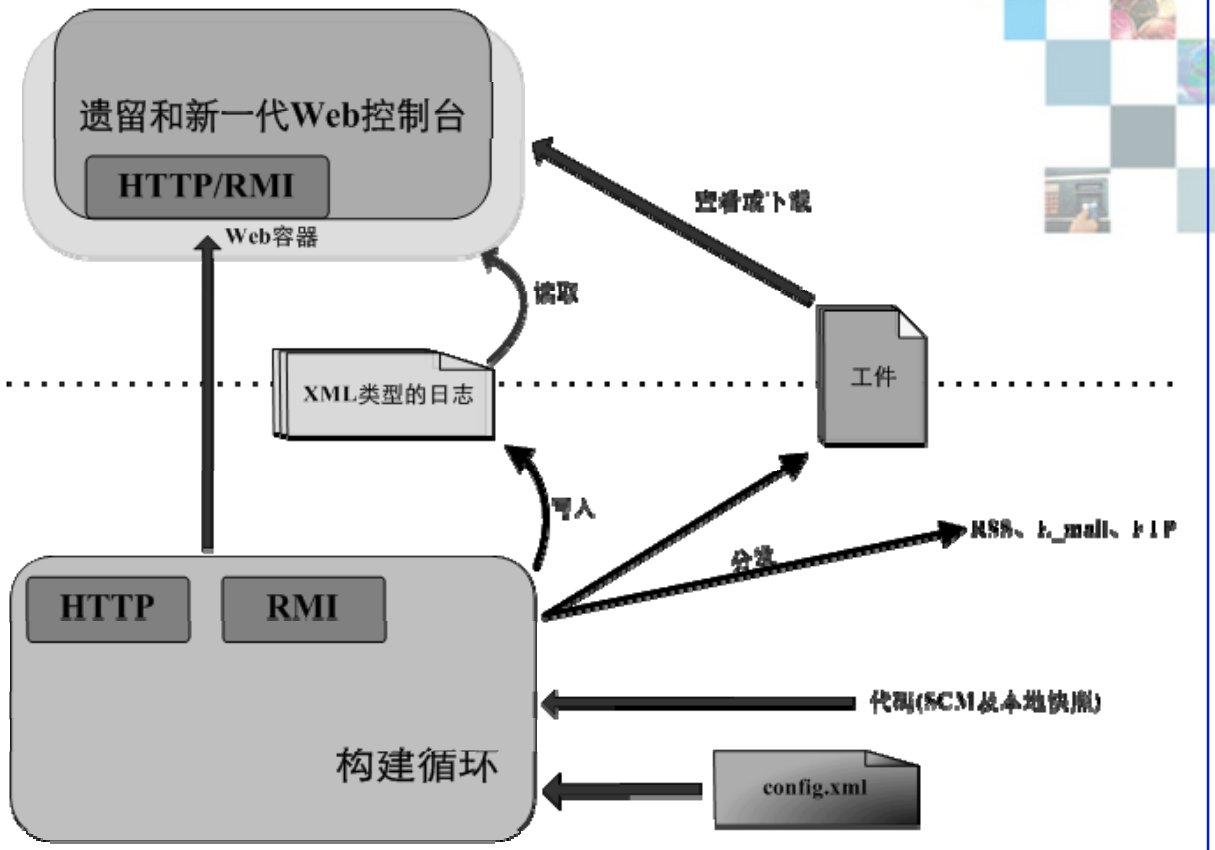


CI服务器介绍

- 迄今为止，如果算上各种商业和开源的CI服务器，则一共有20~30款产品。可见，持续集成已经深入到各个角落，包括对Java、.NET（C#）、Ruby、Perl、Python等语言和平台的支持。比如，CruiseControl、Hudson、CruiseControl.NET、Anthill、Apache Continuum、Luntbuild、等等。不同CI产品存在的差异还是较大的，但它们都是以解决持续集成问题为出发点的
- CruiseControl持续集成服务器历史悠久，可谓是现有各种CI服务器的鼻祖，其产品的成熟特质、使用面广、得到一流的开发团队支持（ThoughtWorks）。最重要的是，这是一款开源产品，其内置的持续集成特性非常丰富，而且可扩展性很强

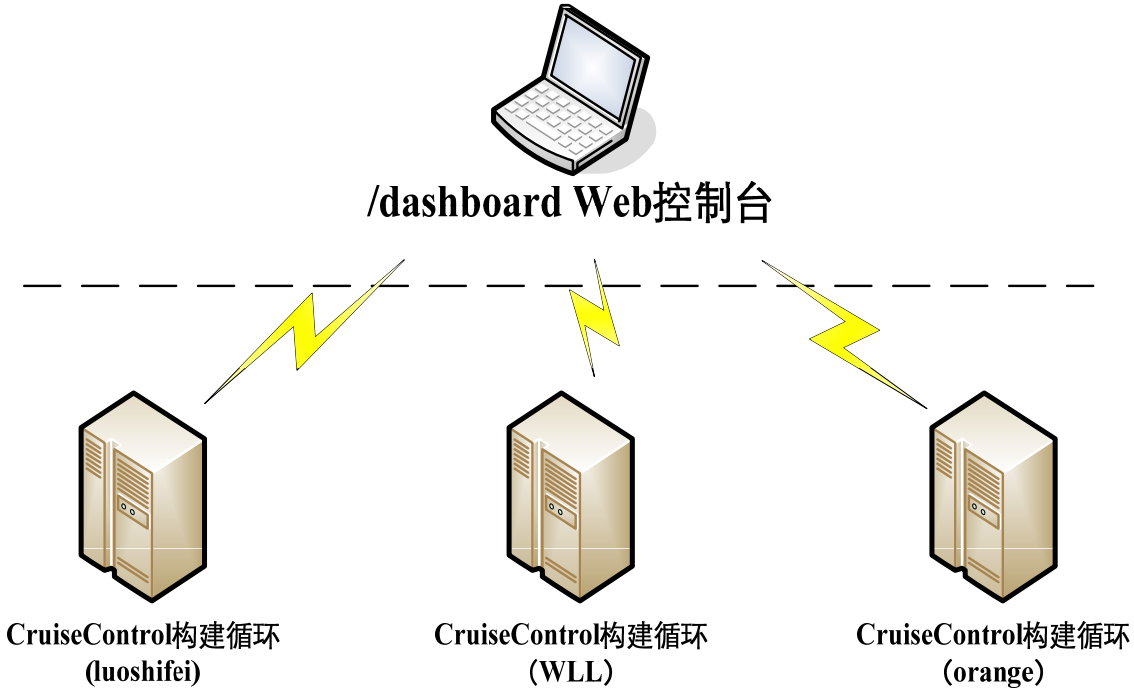
CruiseControl介绍

■ 架构图:



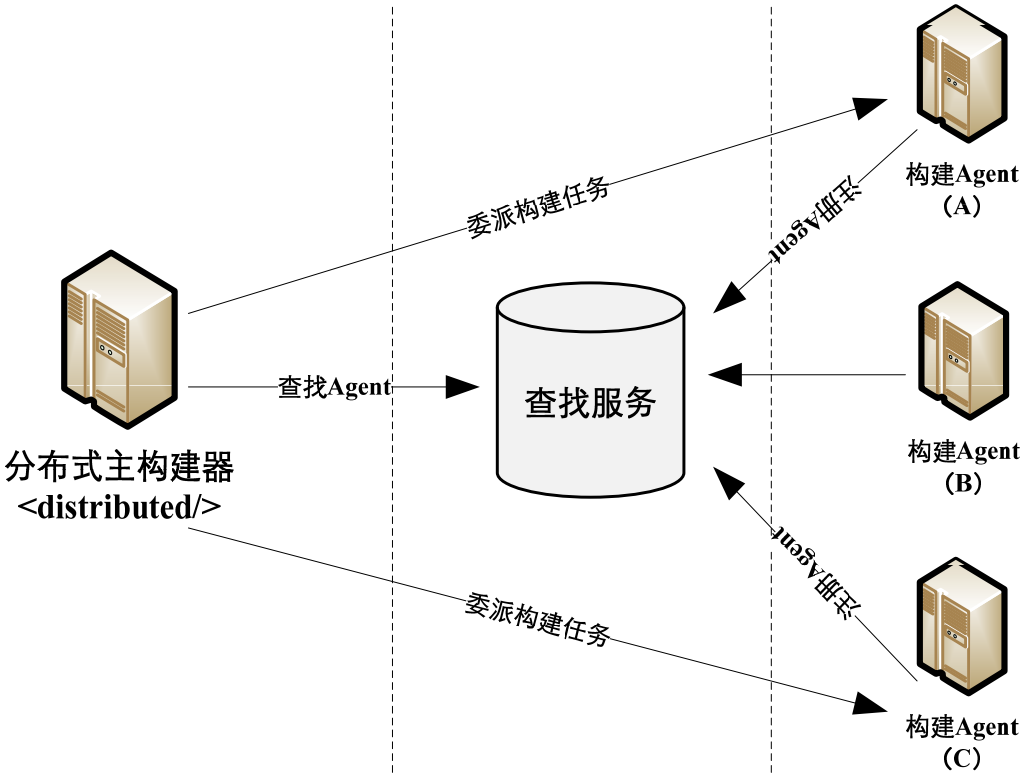
CruiseControl介绍

- 基于/dashboard控制台的构建网格:



CruiseControl介绍

■ 分布式集成构建架构图:



CruiseControl介绍

- config.xml配置文件示例:

```
<cruisecontrol>
    <project name="connectfour">

        <listeners>
            <currentbuildstatuslistener file="logs/${project.name}/status.txt"/>
        </listeners>

        <bootstrappers>
            <svnbootstrapper localWorkingCopy="projects/${project.name}" />
        </bootstrappers>

        <modificationset quietperiod="30">
            <svn localWorkingCopy="projects/${project.name}"/>
        </modificationset>

        <schedule interval="300">
            <ant anthome="apache-ant-1.7.0" buildfile="projects/${project.name}/build.xml"/>
        </schedule>

        <log>
            <merge dir="projects/${project.name}/target/test-results"/>
        </log>

        <publishers>
            <onsuccess>
                <artifactspublisher dest="artifacts/${project.name}"
                    file="projects/${project.name}/target/${project.name}.jar"/>
            </onsuccess>
        </publishers>

    </project>
</cruisecontrol>
```

CruiseControl介绍

■ 内置插件集合:

- ant(0..*)
- maven(0..*)
- maven2(0..*)
- pause(0..*)
- nant(0..*)
- phing(0..*)
- rake(0..*)
- exec(0..*)
- composite(0..*)

- accurevbootstrapper(0..*)
- alienbrainbootstrapper(0..*)
- antbootstrapper(0..*)
- clearcasebootstrapper(0..*)
- clearcaseviewstrapper(0..*)
- cmsynergybootstrapper(0..*)
- currentbuildstatusbootstrapper(0..*)
- currentbuildstatusftpbootstrapper(0..*)
- cvsbootstrapper(0..*)
- execbootstrapper(0..*)
- gitbootstrapper(0..*)
- harvestbootstrapper(0..*)
- lockfilebootstrapper(0..*)
- mercurialbootstrapper(0..*)
- p4bootstrapper(0..*)
- plasticscmbootstrapper(0..*)
- snapshotcmbootstrapper(0..*)
- starteambootstrapper(0..*)
- surroundbootstrapper(0..*)
- svnbootstrapper(0..*)
- tfsbootstrapper(0..*)
- vssbootstrapper(0..*)

- antpublisher(0..*)
- artifactspublisher(0..*)
- clearcasebaselinepublisher(0..*)
- cmsynergybaselinepublisher(0..*)
- cmsynergytaskpublisher(0..*)
- compoundpublisher(0..*)
- currentbuildstatuspublisher(0..*)
- currentbuildstatusftppublisher(0..*)
- email(0..*)
- execute(0..*)
- ftppublisher(0..*)
- htmlemail(0..*)
- http(0..*)
- jabber(0..*)
- onfailure(0..1)
- onsuccess(0..1)
- rss(0..*)
- sametimeannouncement(0..*)
- scp(0..*)
- sfeedocman(0..*)
- sfeefrs(0..*)
- sfeetracker(0..*)
- socket(0..*)
- weblog(0..*)
- x10(0..*)
- xsltlogpublisher(0..*)
- yahoopublisher(0..*)

- accurev(0..*)
- alienbrain(0..*)
- alwaysbuild(0..*)
- buildstatus(0..*)
- clearcase(0..*)
- cmsynergy(0..*)
- compound(0..*)
- cvs(0..*)
- darcs(0..*)
- filesystem(0..*)
- forceonly(0..*)
- git(0..*)
- harvest(0..*)
- httpfile(0..*)
- mavensnapshotdependency(0..*)
- maven2snapshotdependency(0..*)
- mercurial(0..*)
- mks(0..*)
- p4(0..*)
- plasticscm(0..*)
- pvcs(0..*)
- snapshotcm(0..*)
- starteam(0..*)
- store(0..*)
- surround(0..*)
- svn(0..*)
- tfs(0..*)
- timebuild(0..*)
- ucm(0..*)
- veto(0..*)
- vss(0..*)
- vssjournal(0..*)

案例研究

- petclinic: 一典型的示例应用

总结

- 在提升软件质量、降低研发风险、拒绝浪费方面，处于敏捷实践领域的持续集成起到重要作用
- 持续集成能够解决研发工作中的80%任务（日常），而剩下的20%任务（非日常）需要研发团队智力劳动的付出
- CI需要整个团队的配合，我们不能够把CI看成是单纯的技术实践。更多地，它是一种研发模式的转变
- 注意，持续集成本身也是一持续改进的过程。如何逐渐提升可回归性、敏捷性，这是一个永恒的话题

谢谢！

回答问题

- 技术的，非技术的，……
- 非“CruiseControl”