搜狗实验室技术交流文档

Vol3:2 使用 hudson 搭建 daily build 系统

摘要

每日构建，Daily Build 是指周期性地（每天）、全自动地、完整地对整个项目的代码进行编译和集成。本文以 miscsearch 组搭建 hudson 服务器的实践过程为例，介绍了 daily build 系统的搭建过程。

每日构建与持续集成

每日构建至少有如下两点好处：

- 保证有需求的人（用户或者关联的项目的开发人员）随时可以得到项目的一个最新的稳定版本；
- 对程序员向代码库中所提交的代码的质量做早期的检查，以便尽可能早地发现问题，减小传统项目开发流程中系统集成阶段的难度。

在规模较大的项目中，一次集成的过程往往相当耗资源；为了避免影响日间工作，很多项目将每日构建放到夜间进行，因此也称 nightly build。

每日构建是持续集成的基础。持续集成是在每日构建的基础上，强调集成发生的频率，以及鼓励开发者在提交代码之后尽快进行集成。这样可以在集成验收测试之中尽早发现一些基础性的 bug，减小这部分 bug 的反馈周期，使得测试人员可以更多地关注新功能的测试。

每日构建在实施上由如下要素构成：

1. 集成的自动化。整个集成的过程，从获取代码到集成结束，不应该有人的参与。
2. 构建的完整性。需要将每一个模块从源代码编译开始构建，直到整个项目完成。
3. 构建的周期性。构建需要保证稳定的周期。

以上三点中，2 和 3 通常由持续集成服务器来完成；1 除了持续集成服务器支持外，还需要由项目的开发人员提供相应的脚本来完成。在本文的例子中，以上两者分别由 hudson 和 ant 脚本来实现。

使用 ant, junit 实现构建自动化

对于 java 代码的项目，使用 ant 的<junit>任务以及 junit 工具可以很容易的实现构建自动化。以下主要说明一下自动测试的实现。

假设项目中有数个模块，典型的代码组织结构如下：

```
/
|- build.xml
```

```
|- +postlogic
   |- build.xml
|- +mysqllogic
   |- build.xml
|- +front_post
   |- build.xml
```

其中，postlogic, mysqllogic, front_post 分别是项目的 3 个模块，有各自的 build.xml 脚本；整个项目有一个总的 build.xml 脚本。

在模块 build.xml 中，除了通常的编译任务以外，需要如下的任务：

- **运行测试。**

```
<target name="run-test" depends="compile-test">
    <mkdir dir="${reports.tests}" />

    <junit fork="yes">
        <classpath refid="libs.path" />
        <classpath>
            <pathelement path="${conf.dir}" />
        </classpath>

        <formatter type="xml" extension=".xml" />

        <batchtest todir="${reports.tests}">
            <fileset dir="${src_test.dir}">
                <include name="**/*Test.java" />
                <exclude name="**/AllTests.java" />
            </fileset>
        </batchtest>
    </junit>

</target>
```

以上<junit>任务的作用是，找出${src_test.dir}目录下所有以 Test.java 为结尾的源码所对应的类，将他们作为 TestCase 全部运行一遍。<formatter>元素指定了一个输出测试信息的格式。所以，项目中编写的 test case 最好也遵循这个**约定：Test Case 以"Test"作为类名的结尾；非 test case 源码则不要以"Test"作为类名结尾。**

同时，上述<junit>运行完成后会将测试的结果保存在${reports.tests}目录中，这些结果是一系列的 xml 文件，形如：

TEST-com.sogou.saybar.postlogic.remote.PostLogicAccessTest.xml。

这些文件的内容包括测试运行的结果、stdout/stderr 记录、运行时间等等信息。这些结果将被 hudson 收集和整理，见下文。ant 也提供测试报告整理的功能，有兴趣请参考<junitreport>。

- **集成接口。**

```
<target name="cit" depends="run-test" description="Continuous Integration
Test">
  <echo message="CIT done." />
</target>
```

这个 target 作为项目集成的一个约定接口。

在项目的总 build.xml 脚本中，集成脚本的入口为：

```
<target name="cit" depends="init" description="Continous Integration Test">
    <ant dir="${postlogic.dir}" target="cit" inheritAll="false" />
    <ant dir="${mysqllogic.dir}" target="cit" inheritAll="false" />
    <ant dir="${front_post.dir}" target="cit" inheritAll="false" />
</target>
```

即依次将各个模块的集成测试任务调用了一遍。这里各个模块调用的顺序由模块之间的相互依赖关系决定，即**不允许出现模块间的循环依赖。**

至此，一个基本的自动集成脚本已经完成。

使用 hudson 搭建 daily build 系统

目前可供选择的商业/免费的持续集成引擎有不少，包括老牌的 CruiseControl，来自 apache 的 Continuum 等等。hudson 是 dev.java.net 社区的一个开源项目，至今已有 1 年的历史。与其他服务器相比，hudson 在易用和功能强大之间做了较好的取舍：hudson 配置非常方便，提供的功能也大都很实用。hudson 还通过 plugin 的方式保持了良好的可扩展性。开源项目 Lucene 已将其 nightly build 迁移至 hudson 来进行管理。

安装

下载

下载最新版的 hudson.war。

启动

在启动 hudson 之前，需要配置 hudson 的工作目录 HUDSON_HOME。这个目录用来保存所有的配置信息和数据。最直接的配置方式：创建一个空目录，并将这个目录路径赋给环境变量 HUDSON_HOME 即可。

hudson 可以作为 war 包部署在应用服务器上，也可以作为 jar 包直接运行。

作为 war 包部署的方式参考各个应用服务器，一般来说将这个 war 包拷贝至 $APP_SERVER/webapps 下即可。注意：hudson 不支持 resin，本文中例子均以 tomcat 作为应用服务器。

此时，可以通过 http://myhost:port/hudson/ 访问 hudson。

直接运行的方式为：

java -jar hudson.war

通过内置的轻量级应用服务器 winstone，hudson 在默认的 8080 端口上开了一个 http 服务。

此时，可以通过 http://myhost:8080/ 访问 hudson。

至此，以下所有配置均可在页面上完成。

配置

全局配置

首先，需要去 http://yourhost/hudson/configure 页面做一些全局的配置。这个页面的每一项都配有比较详细的说明文字。需要优先配置的项可能包括以下内容：

Security：配置 hudson 服务器的访问权限。在 security 选项默认关闭的状态下，匿名用户可以进行所有的操作。打开这个选项后，hudson 将采用应用服务器所配置的帐号权限策略。例如，tomcat 服务器需要在 $TOMCAT_HOME/conf/tomcat-users.xml 这个文件中配置用户的密码及权限。

Ant：添加一个 ant 的安装实例。将测试机上安装的 ant 路径填入，并起一个名字代表这个安装实例即可。

E-mail Notification：邮件通知。Hudson 通过邮件向开发人员通知某些事件的发生，如编译失败/测试失败/集成恢复正常，等等。在这里需要配置邮件的 SMTP 服务器和发件人。

创建项目

其次，需要在 http://yourhost/hudson/newJob 页面建立一个项目。填入一个项目名，选择 Build a free-style software project，创建即可。

项目配置

最后，需要在 http://yourhost/hudson/job/yourjob/configure 页面配置这个项目。包括：

Source Code Management：项目的源码获取方式。根据项目所采用的版本控制系统，选择 cvs 或者 svn 并填写 Repository URL 即可。这里有一个选项 Use update。为了保证每一次 build 不受前一次 build 结果的影响，这里**不推荐选择 Use Update 选项**。

Build Triggers：设置构建的触发器。这里的 Build periodically 选项支持类似 crontab 格式的任务周期定制。

Build：构建脚本设置。按照前文所述，自动集成脚本的入口为 ant 的 cit 任务。因此在这里选择 Invoke top-level Ant targets，并在 Targets 中填写“cit”。

Post-build Actions：构建完成后需要进行的处理。包括：

Archive the artifacts：将构建完成的结果存档打包，并提供下载。这里需要用 ant fileset 的形式指定哪些文件需要存档，如 trunk/*/debug/*.jar。请参考 ant 文档。

Publish JUnit test result report：收集整理 junit 的测试报告，并发布。这里也需要以 ant fileset 的形式指定 junit 报告文件的路径，如 trunk/*/reports/TEST-*.xml。

E-mail Notification：发送通知。在 hudson 中，一次 build 的结果有三种：集成失败(failed)、集成测试失败(unstable)、集成成功(successful)。Hudson 只有在 failed、unstable 以及从非 successful 恢复到 successful 的时候才发送通知，并且可以指定只发给引起集成失败的人。

功能

以上的配置完成以后，hudson 服务器将会按照 build trigger 中设定的日程，周期性地从代码库中下载整个项目，并编译、测试、发布结果。除此之外，hudson 还保留了每一次构建的结果，包括：

每次 build 的生成文件（根据项目配置）存档，可以在页面上下载；
每次 build 的测试结果，以网页形式发布；
每次 build 产生的 stdout/stderr 信息；
Build 持续的时间；
代码版本号；
代码版本间的 changelog，等等。

根据这些统计信息，hudson 还提供了测试结果以及构建时间的趋势图。

存在的问题

在建立 hudson daily build 服务器的初期是比较痛苦的，然而这个阶段同时也是对项目的代码进行整理的一个过程，完成了这个阶段后，项目的代码结构更加的清晰。

在搭建 hudson 的实践中，我们也遇到过一些问题，现在列举如下：

1. 如上文所述，hudson 不支持 resin 服务器，因为 resin 对 web.xml 的解析支持不够完善。如果需要在 resin 中运行 hudson，则需要根据 resin 的提示修改 WEB-INF/web.xml 中的部分内容，以通过 resin 的验证即可。
2. 在 hudson 中，可以配置使用操作系统环境已经配置好的 cvs 客户端，但不能配置 svn 客户端。Hudson 使用内置的 svnkit 工具包来进行 svn 代码库访问。但是目前版本的 svnkit（1.1.2）有一个 bug：它在 jdk1.5.0_06 下不能正常工作。目前，jdk1.5.0_02 和 jdk1.6.0_01 已经验证，可以正常运行 svnkit。
3. 如果采取系统环境变量的方式配置 HUDSON_HOME，则不要在同一台服务器上部署多个 hudson 实例，否则会在周期性 build 的时候产生冲突。如果确实需要部署多个 hudson 实例，可以通过 JNDI 的方式配置 HUDSON_HOME。
4. 如果需要配置 hudson 的邮件通知功能，则需要在可以连通 smtp 服务器的服务器上部署 hudson，即 192.168.11.*网段的服务器。如果发生连不上代码库服务器或 smtp 服务器的情况，可以注意查看服务器 DNS 的设置，或直接通过 IP 来访问。

下一步的工作

目前，还有一些比较有用的功能没有在 hudson 中提供。包括：

1. 和 issue tracker 的集成。目前只有开发者提供的 for JIRA 的 hudson plugin。如果 hudson 能够和我们的 butterfly 集成，则可以让代码的变动趋势更容易掌握。
2. code coverage 统计。测试代码的完善程度直接决定每日构建的效果。code coverage 将是对测试代码的最直接的评价标准。
3. 更丰富的邮件通知内容。目前的邮件通知内容比较简单，如果能在邮件中提供更丰富的内容，则可以节约大家一些时间。

参考资料

http://192.168.11.189:7070/hudson/ - miscsearch 组搭建的 hudson 服务器。注意：目前关闭了认证，匿名用户有所有的权限。

http://lucene.zones.apache.org:8080/hudson/ - Lucene 项目组的 nightly build 服务器。

让测试自动化 – Developerworks 上的一个介绍持续集成的系列文章。

Continuous Integration Server Feature Matrix – 主流的持续集成工具对照表。

https://hudson.dev.java.net/ – Hudson 首页。可惜 hudson 的 wiki 大概是在长城之外。

http://ant.apache.org/manual/index.html – Ant 手册。