

Maven3 实战笔记 11Maven 集成 Hudson 进行持续集成

刘岩

Email:suhuanzheng7784877@163.com

1. 持续集成

持续集成其实是由敏捷开发这个概念衍生出来的，持续集成可以实现项目自动编译、自动运行单元测试、自动部署、自动进行回归测试。在设定的一段周期内，进行自动构建。

持续集成使很多大型项目敏捷化成为了可能。主要优点有以下

- 1): 集成速度比较快，按照一定的周期自动从 SVN 库 download 源码，编译，发布。
- 2): 集成周期越短，问题反馈越频繁，开发人员收到的反馈越多，修改越多，系统越稳定。被公司剥削的剩余价值就越彻底。
- 3): 所有工作都是自动的，包括搜索新源码、编译、测试、审核报告、部署、反馈信息，开发人员介入的极少，只需要接受反馈看看有没有自己的 problem，之后按照反馈报告进行修改 bug。
- 4): 最主要的还是提高开发人员的自主性，反馈报告发到有关的研发人员的邮箱中，之后研发人员看到后就会发挥主观能动性，修改自己的 bug。

由此可以看出从敏捷开发反映出来的概念就是让开发人员着重于核心、变化的业务开发。

而其他功能都通过自动工具进行，而自动工具又和开发人员进行某种关系的绑定，一旦出了问题，能够及时响应、及时反馈。OK，敏捷开发对于每一个 team 的研发人员要求是不低的。敏捷开发最适合资本家剥削劳动人民剩余价值了，当然对于创业性质的公司也是很好的借鉴思想。话又多了，偏离主题了，咱们来看 Hudson 是怎么进行持续集成的。

2. Hudson 介绍

我们假设一个场景吧，比如一个公司的项目使用 Hudson 进行项目的持续集成工具。首先 Hudson 需要和公司项目的 SVN 服务器 Subversion 进行集成，SVN 相应的账户信息也会集成到 Hudson 中，在 Hudson 中配置好项目的负责人信息（关键是 Email 信息）。设定好构建周期是 1 小时，每经过 1 小时，就进行一次构建，测试等等项目构建。如果遇到构建失败（哪怕一个测试用例和预期结果不一致），会将测试异常报告发到之前配置的人员邮箱中，具体再找原因、修改 bug、提交代码到 SVN 库。之后 Hudson 会按照之前的步骤重新再来一次~~~~如果遇到了问题还会进行反馈的。Hudson 是一个 war 工程，可以从网站 <http://hudson-ci.org/> 下载该 web 项目



笔者下载的版本是 2.0.1，在此版本中笔者觉得 Hudson 的国际化做得挺到位的，在笔者的中文机器上几乎得到了 70% 的汉化。下载后将其部署到 tomcat 下，运行 tomcat 后效果如下



代表 Hudson 部署成功。

3. Hudson 的配置

部署成功后还需要进行一些预配置。点击左边的系统管理-》系统设置

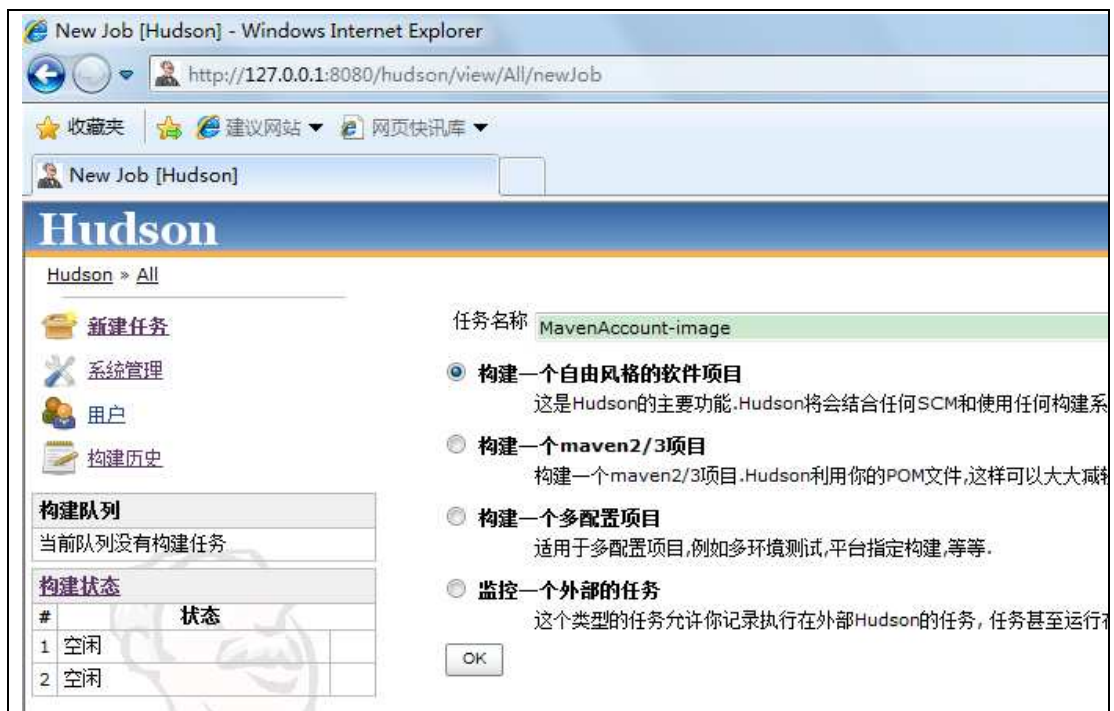
JDK 和 Maven 的 HOME 路径在笔者本机配置如下



4. 配合 Maven3 进行持续集成

配置信息配置好了后就可以建立一个 Hudson 任务了。左边菜单栏新建一个自由风格的

持续集成任务



点击 ok 按钮后出现如下界面

Discard Old Builds 是抛弃历史构建的策略，Days to keep builds 是维护最近几天内的构建，Max # of builds to keep 是维护最近几次构建文件。下面是配置 SVN 项目位置

SVN 的 Server 配置不是本文章的重点，可以参考 Blog 文章

<http://suhuanzheng7784877.iteye.com/blog/926667> 查看详细的 SVN 服务安装。

下面的 Build Triggers 是构建触发器，提供了三种策略来触发“构建”任务。

Build after other projects are built : 在一个别的任务构建成功后才触发此项目的构建，开始构建。

Build periodically : 周期性的构建此项目，按照配置时间无论 svn 是否有所更新，直接触发构件此项目的事件。

Poll SCM : 周期性检测 SVN 仓库，下载，对比发现有新版本后再进行构建。

一般采用 Poll SCM 模式，这个时候就需要配置频率时间了。时间配置类似于 quartz 的时间频率配置

**** : 每分钟

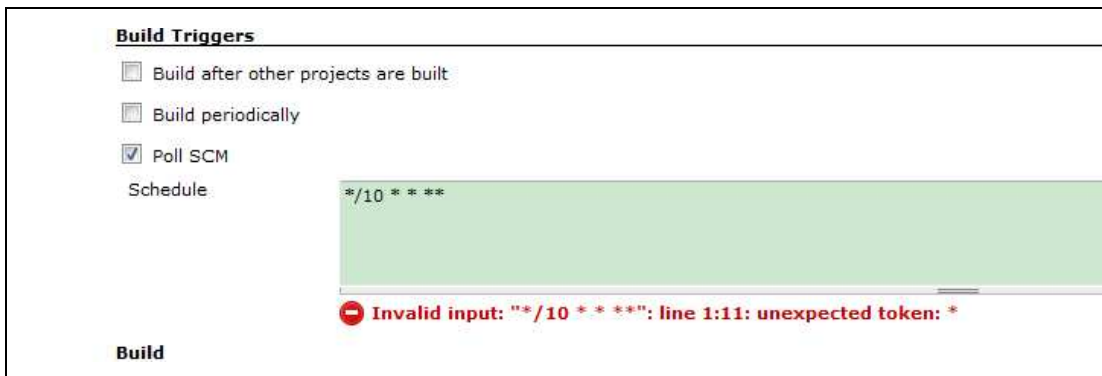
5 **** : 每小时的第五分钟，如：12:05、13:05

*/10 **** : 每隔十分钟

45 10 ** 1-5 : 每周一到周五的上午 10:45

0,30 * 13 * 5: 每月 13 号的没半个小时构建一次，或者每周五的每半个小时

这个配置在网上有很多如何配置的资源，用到的时候查询就好了。如果字符串的配置有误，格式不正确会出现如下提示



The screenshot shows the 'Build Triggers' configuration window. On the left, there are three checkboxes: 'Build after other projects are built' (unchecked), 'Build periodically' (unchecked), and 'Poll SCM' (checked). Below these is a 'Schedule' label. To the right of 'Schedule' is a text input field containing the cron expression '*/10 * * * *'. Below the input field, a red error message is displayed: 'Invalid input: "*/10 * * * *": line 1:11: unexpected token: *'. At the bottom left of the window is a 'Build' button.

修改后如下

Build Triggers

☐ Build after other projects are built

☐ Build periodically

☒ Poll SCM

Schedule

*/10 * * * *

Build

Add build step ▼

之后就是任务构建的配置

Build

Add build step ▼

Execute shell

Invoke top-level Maven targets

Execute Windows batch command

Invoke Ant

☐ Aggregate downstream test results

选择 Invoke top-level Maven targets，配置相应的 Maven 构建命令。

Build

Invoke top-level Maven targets

Maven Version

maven3.0.3

Goals

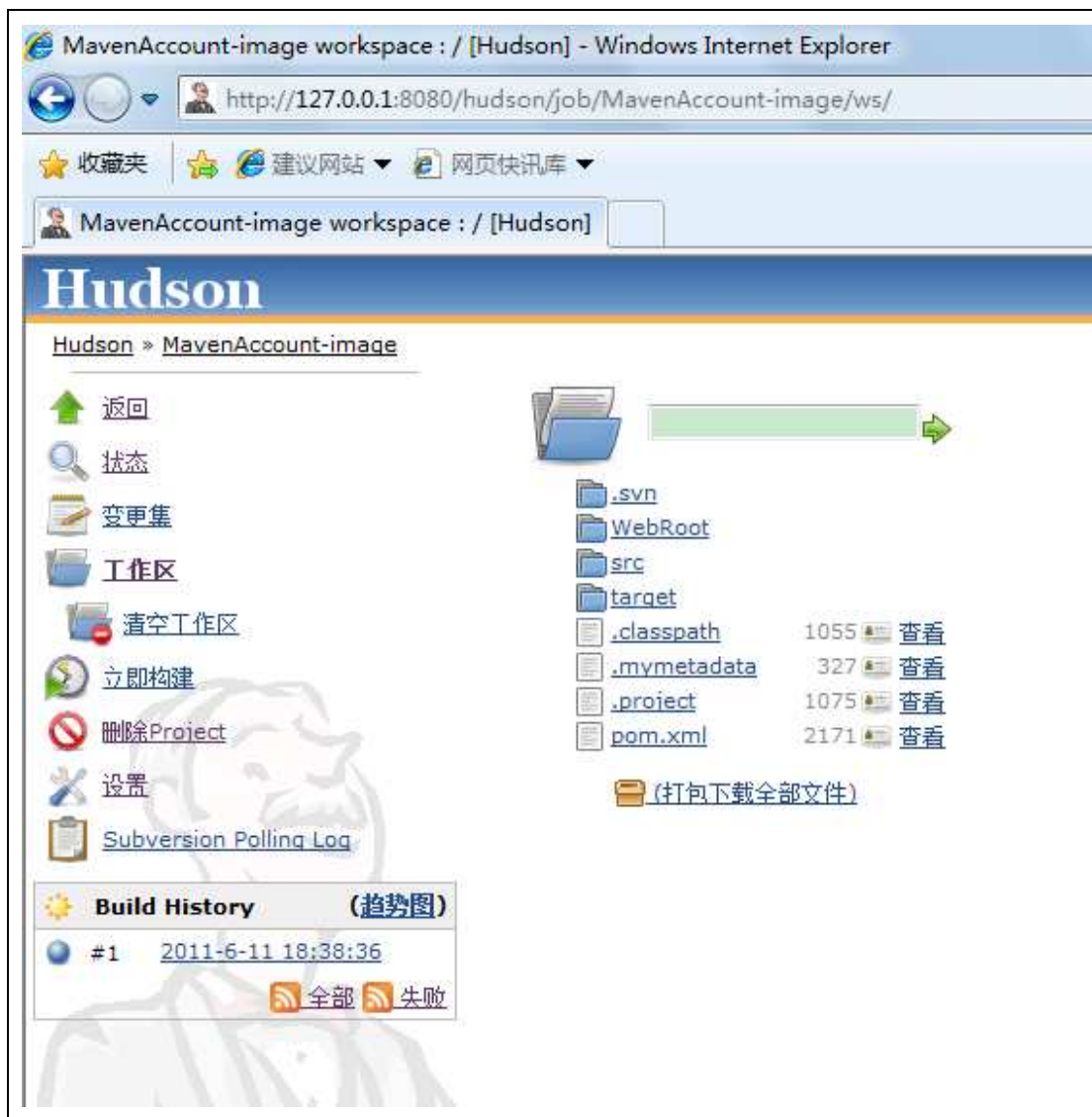
clean test

一切 OK 后保存即可。

之后点击立即构建，效果如下



点击工作区可以看到我们集成测试工具 Hudson 下载、构建项目的文件夹信息



构建历史如下

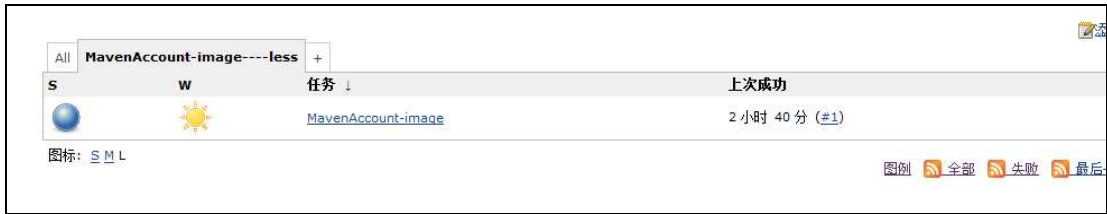
集成测试入门

All +		任务 ↓	上次成功	上次失败	上次持续时间	
S	W	MavenAccount-image	2 小时 36 分 (±1)	N/A	1 分 6 秒	

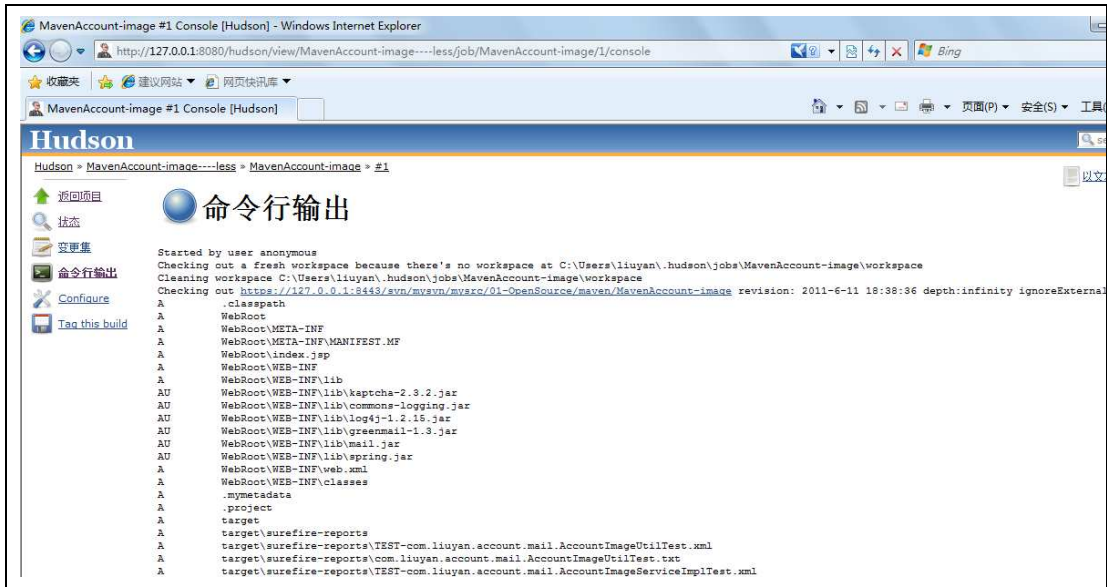
图例: S M L

图例 全部 失败 最后

S 列代表构建状态，W 则反映了该任务的长期统计构建状况。具体可以参考图例。希望 Hudson 下一个版本的国际化做得更好。当然，单词也都很简单。假如只关心其中一个任务的几个字段，而不是全部信息，可以点击+选项卡添加自己感兴趣的项目任务。



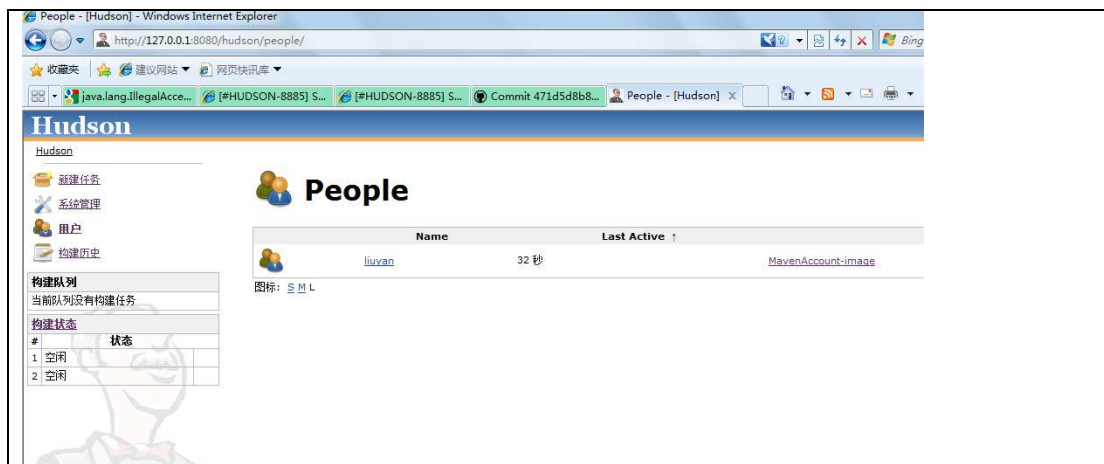
还可以查看项目的实际命令行输出



修改源代码后通过 SVN 提交，稍等一会儿 Hudson 界面如下



在用户那一栏也已经更改了，和 SVN 用户信息同步了



以下是更改后的自动构建效果



这里有一个小问题，就是使用原始的 Hudson2.0.1 的 svn 插件会报错

```
java.lang.IllegalAccessError:      tried      to      access      method
hudson.scm.ChangeLogSet$Entry.setParent(Lhudson/scm/ChangeLogSet;)V      from      class
hudson.scm.SubversionChangeLogSet
```

需要从他的官网上下载新的 subversion 插件放到 tomcat 下的 hudson 项目的 plugin 下

面。证据在

<http://issues.hudson-ci.org/browse/HUDSON-8885?page=com.atlassian.jira.plugin.system.issuetabpanels%3Aall-tabpanel> 处。

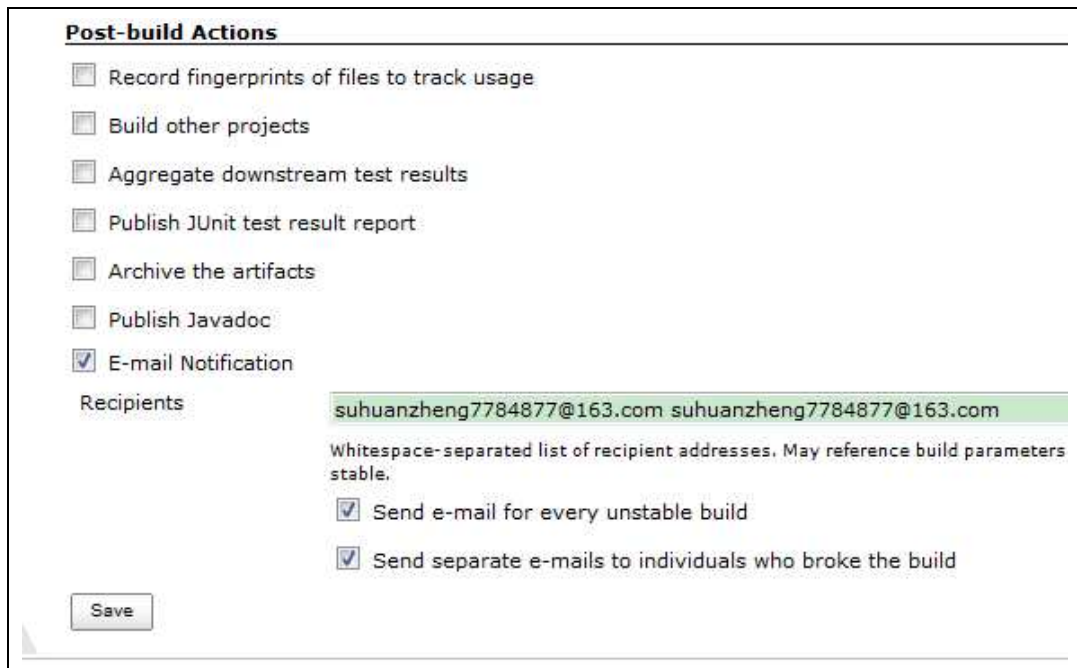
5. 报告通知机制

在用户设置那边是自动与 SVN 账户进行同步的，最重要的就是设置用户的 Email 信息，以便能够追踪问题到个人。



The screenshot shows the Hudson user settings page for a user named 'liuyan'. The page has a blue header with the 'Hudson' logo. On the left, there is a sidebar with navigation links: '用户' (User), '状态' (Status), '构建' (Build), '我的视图' (My Views), and '设置' (Settings). The main content area is divided into sections: 'Your name' (liuyan), 'Description' (半涉流席青, 倚箫闲吟广陵文。寒剑默听君子意, 傲视人间笑红尘。), 'E-mail' (Email address: suhuanzheng7784877@163.com), and 'My Views' (Default View: All). A 'Save' button is at the bottom.

自动反映报告机制是在系统设置上的 Email-notification 设置的，多个电子邮箱以空格分别。



The screenshot shows the 'Post-build Actions' configuration page in Hudson. It lists several actions with checkboxes: 'Record fingerprints of files to track usage', 'Build other projects', 'Aggregate downstream test results', 'Publish JUnit test result report', 'Archive the artifacts', 'Publish Javadoc', and 'E-mail Notification' (which is checked). Below the 'E-mail Notification' section, there is a 'Recipients' field containing the email address 'suhuanzheng7784877@163.com'. A description below the field states: 'Whitespace-separated list of recipient addresses. May reference build parameters stable.' There are two more checkboxes: 'Send e-mail for every unstable build' (checked) and 'Send separate e-mails to individuals who broke the build' (checked). A 'Save' button is at the bottom.

修改一下测试代码

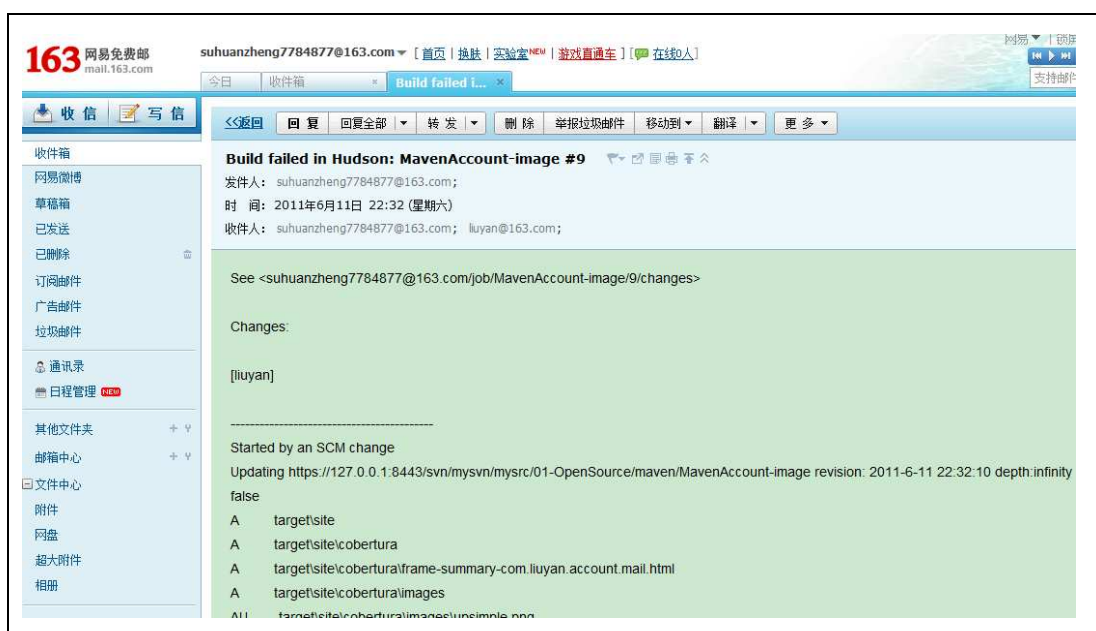
```
package com.liuyan.account.mail;
import static org.junit.Assert.assertFalse;
import org.junit.Test;
public class AccountImageUtilTest {
```

```

@Test
public void testGenerateText() {
    String value = AccountImageUtil.generateText();
    assertFalse("1111".equals(value));
    assertFalse(!"1111".equals(value));
}
}

```

构建、测试之后会发现一个测试用例和预期结果不符。我的邮箱收到了一封来自 Hudson 的测试报告



内容基本上就是 Maven 的命令行输出信息，邮箱是要在全局系统配置中设置的，在此不再赘述。

是得一提的是在用户的临时文件夹下，例如：C:\Users\liuyan\.hudson\jobs (其中 liuyan 是登录系统用户名，linux 是/home/用户名/.hudson/jobs) 下面包含了构建历史信息，如果项目构建次数过多，那么此文件夹的历史信息会很多。及早清除无用的存档历史可以节省不少空间。