IOS项目持续集成环境配置

**目录：**

**一、集成测试环境**

**二、Jenkins的安装与权限**

**1.介绍**

**2.下载与安装**

**3.权限**

**三、项目构建与Plugins**

**1.源码管理**

**2.构建触发器**

**3.Xcode的构建**

**4.邮件报告**

**5.邮件增强插件**

**四、CodeStyle**

**五、小结**

注：Jenkins安装包放在221的“Mac开发工具及软件”里面。

**一、集成测试环境**

系统环境：Mac OS X 10.7.3

IDE：Xcode 4.3.2

工具：jenkins-1.465

Mac OS X 系统上必须安装了JDK插件，不知道机器上有没有安装的话，在终端输入

$ java -version

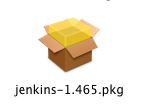
如果输出NotFoundException则说明没有安装，输出的是JDK版本则说明已经安装。没有安装的话，在安装Jenkins的时候系统会提示安装。

**二、Jenkins的安装与权限**

**1.介绍**

Jenkins原名Hudson，是SUN公司用Java语言开发的一个持续集成工具。后来甲骨文收购了Java后，SUN公司被迫将Hudson改名为Jenkins。

**2.下载与安装**

在Jenkins官方网站<http://jenkins-ci.org>上有提供各种平台下的Jenkins包下载，这里我们只需要用到Mac OS X平台下的。下载下来之后会是一个.pkg的java包

双击安装之。

在安装到这一步的时候，可以选择“自定”。Jenkins提供2种用户运行模式，一种是使用Daemon守护程序运行，一种是使用它自己名称Jenkins作为账户运行，如图：

由于没有User的用户选择，因此选择哪种都对后面的权限问题没有正面作用，因此默认选择就好了（默认是daemon）。安装完之后Jenkins会自动运行并且弹出网页<http://localhost:8080/>，这是Jenkins的操作界面，会占用8080端口（请确认8080端口未被使用）。如果打不开，请到Application里面双击运行Jenkins。

如果你已经能正常进去Jenkins的操作界面，那么恭喜你，安装成功了。

**3.权限**

安装虽然成功了，Jenkins现在不是运行在root权限下，也不是运行在你当前用户下，因此无论如何操作都会诸多阻挠，各种报错，因此我们首先要解决的就是权限问题（事实证明，后面遇到的很多问题都是权限有关）。

既然我们知道权限是账户的，那么我们把Jenkins的进程移到我们自己的账户下就好了。 Jenkins安装完毕后会在/Library/LaunchDaemons/ 目录下创建一个org.jenkins-ci.plist文件，这个文件决定了Jenkins进程运行的用户组和用户，如图：



注：这图我已经修改了用户组和用户，没修改之前string都是daemon

我们只需要更改这个配置文件，就可以让Jenkins进程挪到另外一个用户。但是直接编辑这个文件是无效的，必须要用到root权限来修改，那么我们使用命令来修改：

首先，我们要停止这个文件的使用，停止后Jenkins也会停止

$ sudo launchctl unload /Library/LaunchDaemons/org.jenkins-ci.plist

进入org.jenkins-ci.plist文件所在目录

$ cd /Library/LaunchDaemons

然后我们就可以修改了，那么使用vim打开该文件进行修改

$ sudo vim +1 +/daemon +'s/daemon/staff/' +/daemon +'s/daemon/twer' +wq org.jenkins-ci.plist

（注释：这里用的是vim的命令，+代表一个命令的开始。/daemon是在文本中搜索daemon关键字逐一匹配；s是替换命令，将搜索到的daemon关键字替换为staff；最后面的wq 是保存并且退出，最后面的是保存的文件名。staff和twer对应的是用户组和用户名，可以输入id命令查看）

修改完配置文件后，我们需要修改Jenkins的owner（其中路径是Jenkins的安装目录）

$ sudo chown -R twer:staff /Users/Shared/Jenkins/

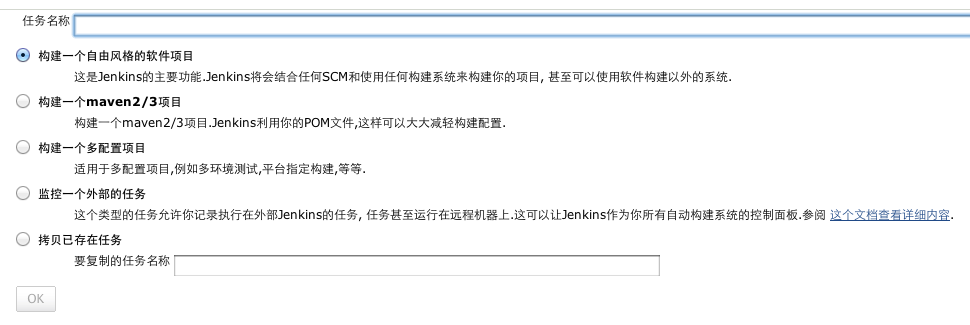
OK，一切准备就绪，启动Jenkins

$ sudo launchctl load /Library/LaunchDaemons/org.jenkins-ci.plist

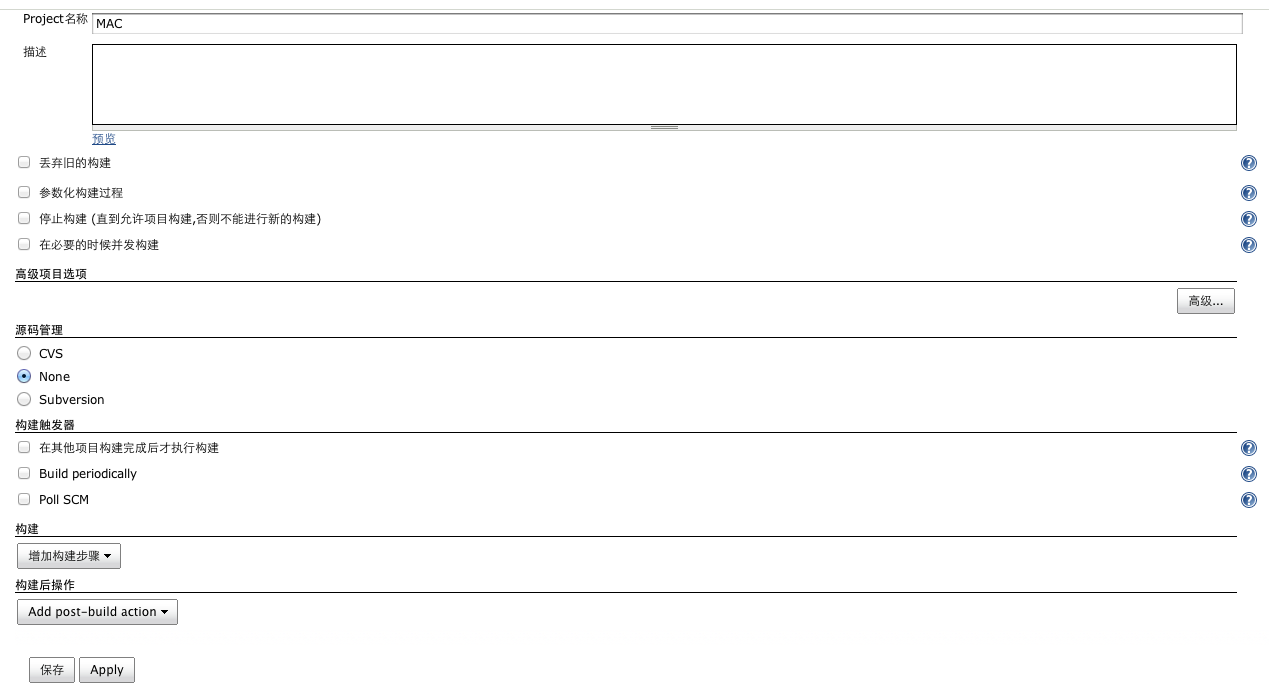
启动Jenkins之后刷新一个Jenkins的操作界面，等待一会之后就可以正常操作了。这之后打开活动监视器，Jenkins就是运行在你的用户下了。

**三、项目构建与Plugins**

预备工作都好了以后，我们可以操作Jenkins了。新建一个Job，选择第一个——自由风格的软件项目，填入任务名称，如图：



点击OK进入项目构建配置界面，如图：

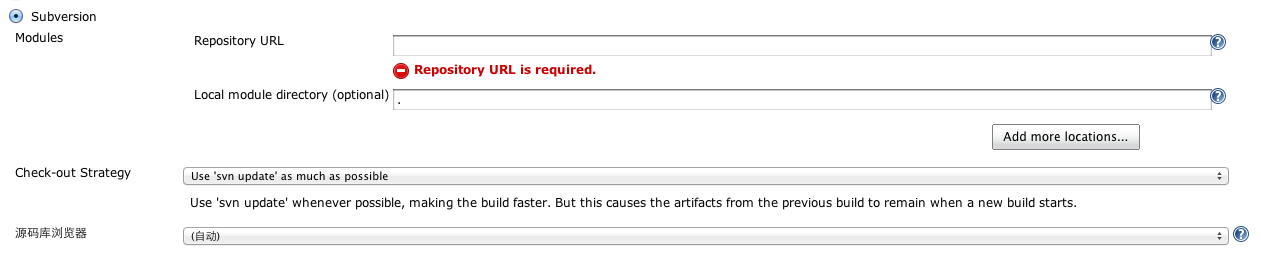


这是一开始的配置界面。可以设置你需要的配置，这里我用到SVN的源码和定时构建，因此源码管理和构建触发器。

Tips：几乎每一个配置后面都有？提示，点击提示按钮会有该配置项的说明。

**1.源码管理**

选择Subversion，



填入SVN地址，其他默认就好了，至于高级，如果有需要的话可以去配置一下。在输入框失去焦点的时候，Jenkins会根据你输入的SVN地址尝试连接，如果连接不到你会得到相关的提示。如果你的SVN需要账号密码，则初次配置的时候会提示你需要配置，



点击



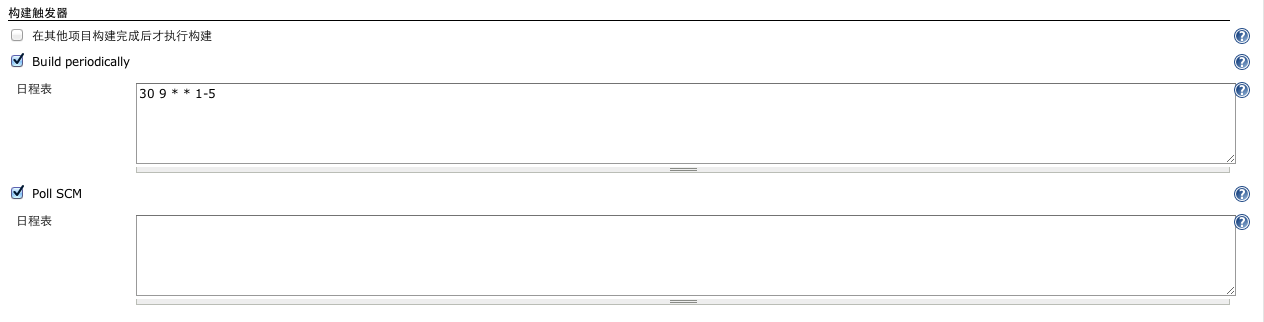
后可以设置账号密码。配置正确后确认保存。如果你的工程需要多个文件，可以增加签出内容。

SVN配置好后，在Jenkins的Home目录下会产生一个配置文件，记录了SVN地址和你的账号密码，在Jobs目录下能看到你的Job，里面有一个SVN的证书，用UItraEdit打开的话会看到这是一个xml格式的文件，里面也保存了你的账号密码，只是密码是经过加密的。

在URL下面的是文件存储路径。“.”表示默认路径，你修改该路径。

**2.构建触发器**

触发器有3个选项，是多选的，如图：



第一个说明得很清楚，是在某个Job完成之后再进行构建，勾选后填入你需要的Job名称就可以了。

第二个选项是定时build项目工程，第三个则是定时检测工程的版本。

日程表的填写格式在后面点击？后会看到，这里用的是cron的语法。我这里填写的是每周一到周五早上9点30分开始构建。注意，这里的时间是以本地时间来计算的。

**3.Xcode的构建**

Jenkins原生不支持Xcode构建项目，由于苹果开发者多了，于是有了Xcode插件。

进入Jenkins的系统管理界面，选择管理插件，



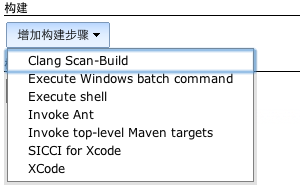
搜索xcode，会看见3个插件：





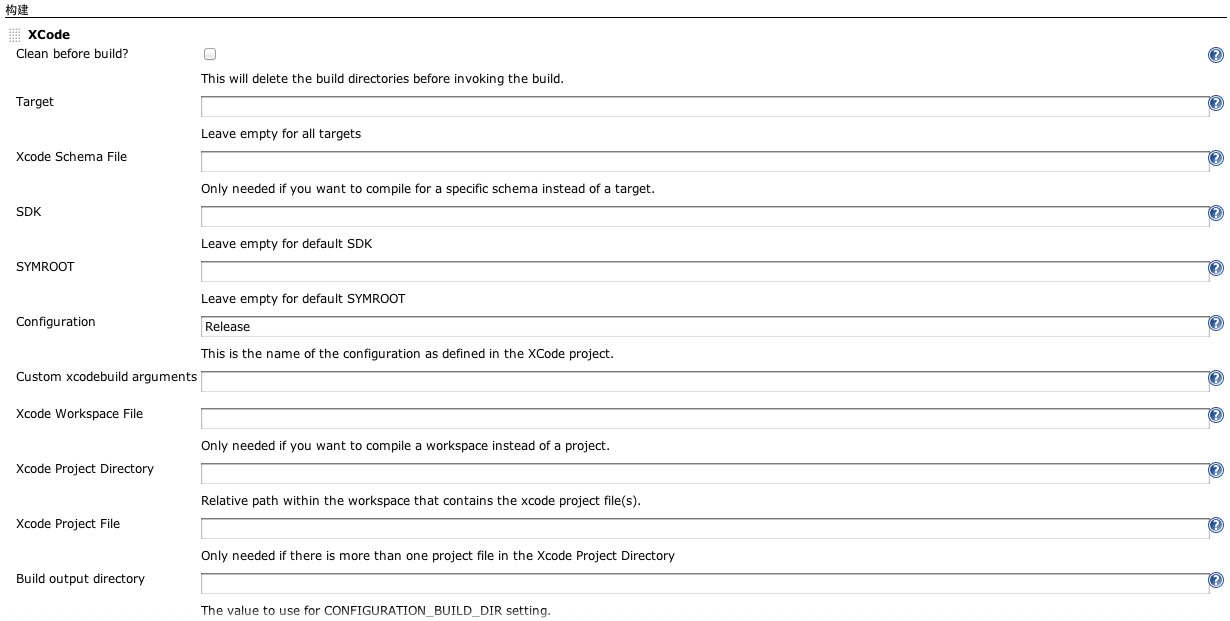


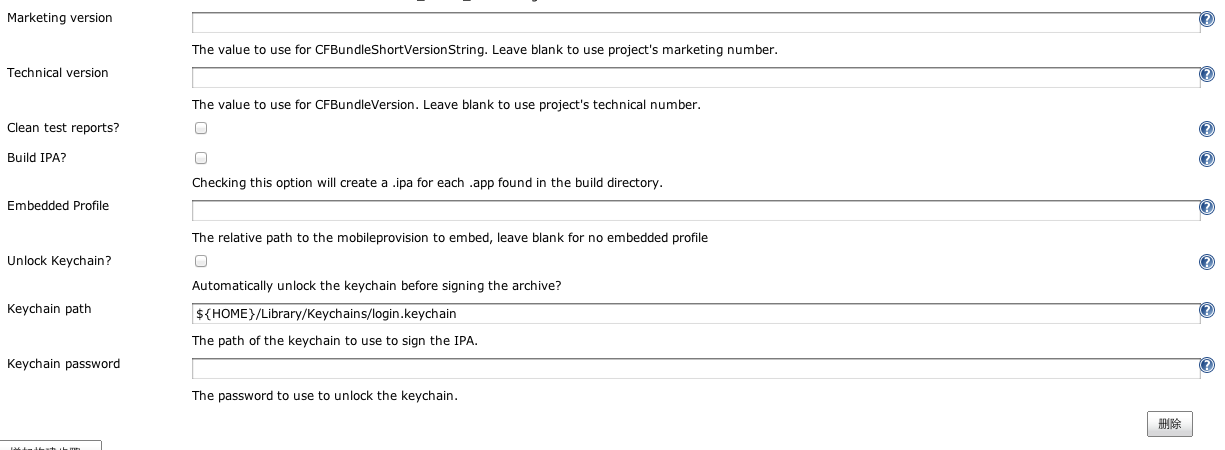
点击插件可以进入查看插件详情。这里最后一个插件是使用Xcode的Clang Static Analyzer 工具来检测代码的，这里暂时不使用。勾选XCode integration 和SICCI for Xcode Plugin插件，安装并且重启之后，在Job配置项中就可以选择插件了：



我们这里添加XCode。补充说明一下，SICCI for Xcode 这个插件是用于支持证书的，无需增加构建步骤，再且，你点击这个选项也无法增加一个构建步骤。

增加XCode构建步骤后，会看到插件配置如下图：





具体每一个项目可以点击后面的？查看，可以根据自己的工程项目来设置。我的工程是用的workspace，因此xcode workspace file 填写了工程workspace的本地目录（不是SVN的目录）：

/Users/zhangyj/Desktop/PMH/PMH

这里不需要添加xcworkspace后缀，插件默认会帮你添加后缀。

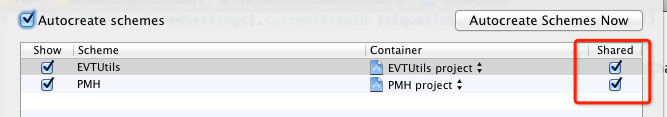
然后按照插件的意思，在你填写workspace之后必须提供一个编译器。一般情况下编译器的名字就是你的target名，我在Xcode Schema File上填写了

PMH

但是就这样的话，Jenkins无法访问你的编译器，会报错，报错内容大概如下：

xcodebuild: error: The workspace 'Moments.xcworkspace/' does not contain a scheme named 'Moments'.

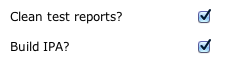
你需要用Xcode打开你的工程，进入Manager Schema 里面，share该Schema。



在Build output directory上填入输出的目录，由于涉及到文件读写权限（如果你在一开始设置了Jenkins的用户和用户组，那么你可以把输出目录设置到当前用户下），我把输出目录放到Jenkins里面：

/Users/Shared/Jenkins/Library/Developer/Xcode

填入后勾选如下2项：



至于是否需要build 之前先clean一下，就看你的需求了。

Keychain path默认的就好了。

这样，XCode 构建方面已经配置好了。现在可以构建一次。一般，如果没有任何权限问题，现在会报一个错：

You have not agreed to the Xcode license agreements, please run xcodebuild standalone from within a Terminal window to review and agree to the Xcode license agreements.

Build step 'XCode' marked build as failure

Finished: FAILURE

这个错误是说你的xcodebuild没有认证。你需要在终端里面输入

$ sudo xcodebuild –license

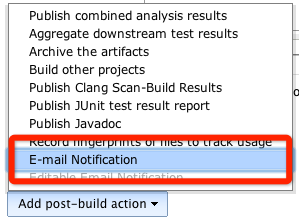
在看完一大串的文档之后，按照要求填写agree。

OK，这样问题都解决了。重新构建一次，大功告成！

**4.邮件报告**

我们好像忘记了什么？邮件报告！是的，Jenkins允许用户在构建之后发送邮件报告构建情况，默认情况下，连续的成功是不会发送报告的。

要发送邮件报告首先完成邮件的设置。首先在Job配置页面的最下面“构建后操作”选择邮件功能：



然后填入你要发送的目标邮箱地址，使用空格把每个邮箱地址分开。这里设置完之后，需要设置发送者的邮箱和服务器。这些设置在Jenkins的系统管理->系统设置 页面中。如下图：



配置好之后，在你构建完毕就会发送一份报告。上面也提到过，在连续构建成功的时候是不会发送邮件报告的。

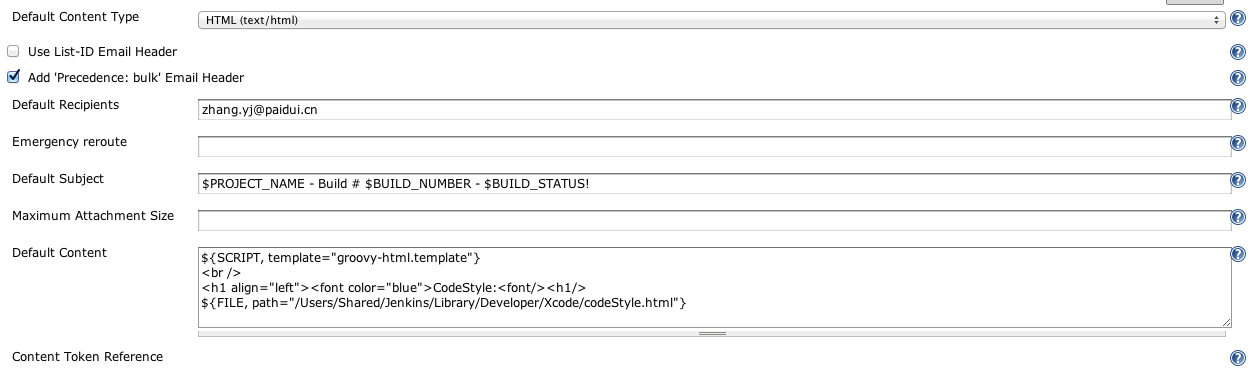
**5.邮件增强插件**

一般来说，Jenkins自带的邮件功能一般满足不了需求，那么我们需要一个更加强大的，能够自定义内容的邮件功能。去管理插件，找到插件



安装并且重启。

到系统设置里面，能看到Extended E-mail Notification 的分支项设置，设置好邮件的发送者和服务器。在这里也可以设置邮件内容格式，如果在这里设置的话，就会影响所有Jobs。由于需求中发送的邮件报告都一样，因此我在这里设置。



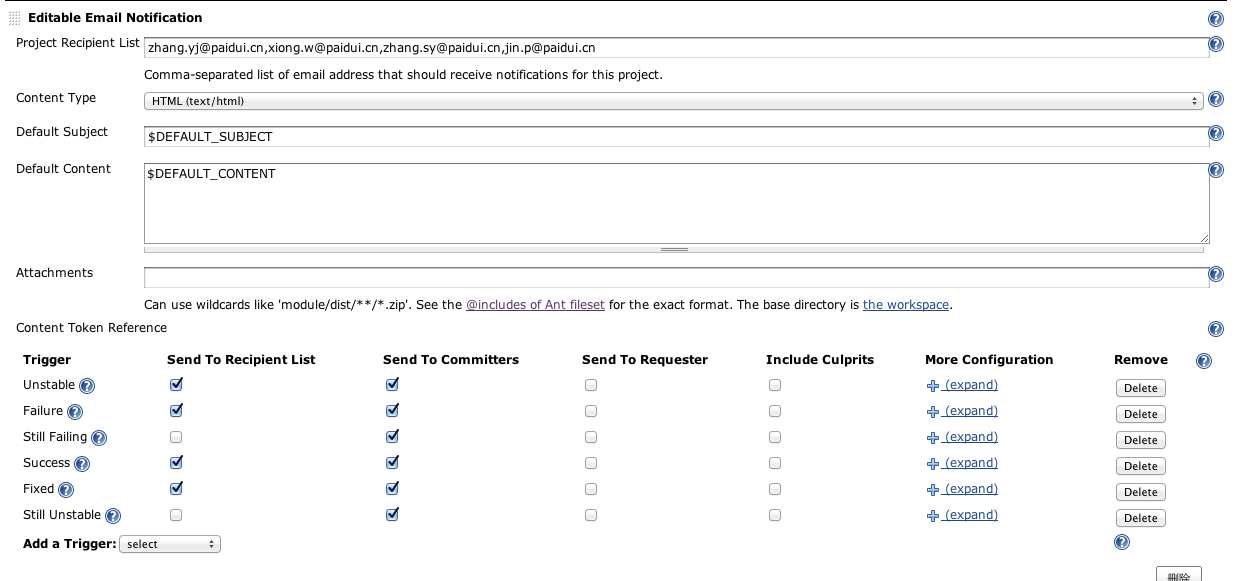
邮件内容包括以上设置。HTML格式会看起来更好看一点，当然也可以选择文本格式。

Default Subject是邮件的title，而默认就会有如上填入。$PROJECT\_NAME这些看上去是命令的东西是该插件提供的一些简便脚本，最下面的Content Token Reference ，点击？就可以看到所有的简便脚本和示例。由于现在的邮件是HTML格式的，因此支持HTML的一下标签。

插件除了提供简便脚本外，本身提供2套模板，你可以在Jenkins文件目录的Plugin目录下找到该插件，里面有模板。如果对此不明白，可以到管理插件处点击插件查看详细的插件介绍。

这里除了使用了一个模板之外，还读取了一个html文件。

设置完内容之后，再回去Job的配置里面，配置一下邮件应该在什么情况下发送和发送给谁。配置内设置界面如下：



由于在系统设置那边已经设置好内容，因此这里就不需要更改了。填入发送目标邮箱地址，用

“,”隔开。然后配置在什么情况下发送，发送给什么人。

如上，配置好后邮件功能也全了，那么重新构建一次，就可以看到邮件报告了。

**四、CodeStyle**

Jenkins有一个插件是checkStyle，但是不适合Objective-C检测，因此需要自己写一个相关的检测工具来满足需求上的代码检查项。例如：



该CodeStyle是用OC语言写的，最后生成一个执行文件。

有了执行后，到Job配置里面增加一个构建步骤，这次选择为



然后填入一段shell脚本，去执行这个CodeStyle文件，检查工程代码的规范性，并且在完毕之后在指定目录生成一个html文件，如：

/Users/zhangyj/Desktop/CodeStyle /Users/zhangyj/Desktop/PMH/ > /Users/Shared/Jenkins/Library/Developer/Xcode/codeStyle.html

该步骤在构建的时候进行，生成的html文件用于邮件输出。

**五、小结**

Jenkins持续集成的配置步骤说明完毕。由于受平台限制，挺多功能无法实现，或者说实现起来比较麻烦。在Jenkins的 FQA里面关于Mac 平台 Xcode构建的问题相对来说也比较少，大多情况下都需要google。

参考资料：

<http://www.cnitblog.com/luckydmz/archive/2012/01/03/77007.html>

<http://geeklu.com/2011/06/jenkins-ci-used-for-mac-app-dev/>

<http://www.printhelloworld.de/2012/04/resolve-jenkins-xcode-license-agreement-issue/>