

持续集成 jenkins

1、官网文档: <http://jenkins-ci.org/>, 参考资料: <http://pan.baidu.com/s/1o6MDYt0>

2、简单来说持续集成就是团队开发成员必须经常集成他们的工作, 甚至每天都可能发生多次集成。而每次的集成都是通过自动化的构建来验证, 包括自动编译、发布和测试, 从而尽快地发现集成错误, 让团队能够更快的开发内聚的软件。对于软件测试, 测试人员需要不断的回归测试case、输出测试报告, 持续集成的思想加上自动化case, 可以很好的实现质量回归。

3、术语和缩略语说明

CI Continuous Integration, 持续集成

Hudson/Jenkins 用于CI的开源项目

SVN SubVersion, 配置管理工具

Unittest 用于Python单元测试的开发框架

JRE Java Runtime Environment, Java运行时环境

Check-Style 开源项目, 用于Java代码的静态检查

Cobertura 开源项目, 用于统计单元测试对代码的覆盖率

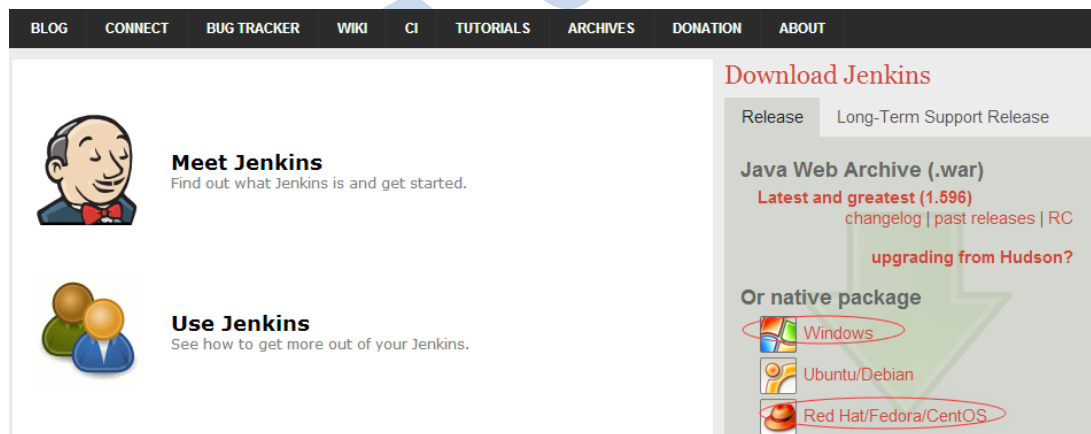
SMTP 简单邮件发送协议

1 环境搭建

1.下载和启动jenkins.war

网盘下载地址: <http://pan.baidu.com/s/1c0m0DoQ>

官网下载地址:



2.windows 镜像安装和 linux rpm 安装

上面官网下载的 jenkins 分别是 windows 镜像文件和 rpm 包, 在下载链接页面的上方, 都有详细的安装步骤或者命令。但这种安装方式不便于迁移, 所以我们一般选择直接启动 war 包的方式

3.直接启动 jenkins.war

在网盘下载 jenkins.war, 因为 Jenkins.war 内置了有 jetty, 所以我们可以直接在命令行运行: java -jar jenkins.war 启动。

在 windows 上, 我们可以在 jenkins.war 的同级目录新建个 bat 文件:

run.bat:

```
java -jar jenkins.war
```

在 linux 上, 新建一个 run.sh:

run.sh:

```
#!/bin/bash
```

```
java -jar jenkins.war
```

使用 nohup 打到后台运行: nohup ./run.sh &

启动成功之后, 访问 url:

<http://ip:8080/>

进入页面:



说明 jenkins 已经启动成功。

注意: 需要安装 jdk

4. 使用 tomcat 启动 jenkins

在网盘: Hudson 的持续集成指南.pdf 有详细的说明, 做过 web 部署的同学应该肯定比较熟悉了。

2 环境配置

1.用户注册: 默认情况下是不启用任何安全策略, 即任何人都可以访问页面且有读写权限。比较常用的安全配置方式即安全矩阵。

首先开启用户注册权限, 默认情况下, 我们在首页点击“系统管理”后, 页面上方会有这个提示:

管理Jenkins

⚠ Jenkins新版本 (1.596) 可点击 [download](#) ([变更说明](#)) 下载。 [或 自动升级版本](#)

⚠ 不安全的Jenkins允许网络上的任何人以你的身份访问程序。考虑至少启用身份验证来阻止滥用。

Because of a [security vulnerability](#) discovered earlier, we need to change the encryption key used to protect secrets in your configuration files on the disk. This process scans a large portion of your `$JENKINS_HOME` (C:\Users\yulong\.hudson), find encrypted data, re-key them, which will take some time. See [this document](#) for more implications about different ways of doing this (or not doing this.) This operation can be safely run in background, but cautious users are recommended to take backups.

[安全设置](#) [忽略](#)

[Re-key in background now](#) [Schedule a re-key during the next startup](#) [Dismiss this message](#)

点击“安全设置”, (对应 url: <http://ip:8080/configureSecurity/>) - “启用安全”, 进入下面页面并选择:



Configure Global Security

☒ 启用安全

JNLP节点代理的TCP端口 ☐ 指定端口: ☒ 随机选取 ☐ 禁用

Disable remember me ☐

Markup Formatter Escaped HTML

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

访问控制

安全域

☒ Jenkins专有用户数据库

☒ 允许用户注册

☐ LDAP

☐ Servlet容器代理

授权策略

☒ 任何用户可以做任何事(没有限制)

☐ 安全矩阵

☐ 登录用户可以做任何事

☐ 遗留模式

☐ 项目矩阵授权策略

☐ 防止跨站点请求伪造

保存后系统管理中就出现管理用户的选项。页面右上角也会出现登录/注册的选项。点击注册：

2.安全策略：

点击“系统管理-Configure Global Security”进入安全设置页面，简单常用的安全策略就是安全矩阵：

添加用户之后保存，这个安全矩阵即生效了。至于应该勾选哪些和不勾选哪些，则根据实际需求来决定了，前提是对这些选项对应的功能大家都摸透了。其他几个安全策略也可以尝试一下。如果要恢复到原始状态可以参考说明 3.

说明 1:

其中: Overall 是全局权限, slave 是集群权限, job,run,view,scm 是业务权限。

其中 overall 中的 read 要勾选, 否则用户登陆后什么也看不到。

overall:

Administer: 系统管理员权限

read:浏览框架

job:

read:查看 job

build:执行构建

cancel:取消构建

run:

Delete:删除某次构建

Update:编辑某次构建信息

SCM:

Tag:为某次构建在 scm 上打标签。

说明 2:

低版本的 jenkins 按上述配置后登录是会报错, 同时我们登录进去也就看不到所有的配置菜单了, 不过没有关系, 我们可以在系统主目录找到存放这个配置的文件: .jenkins/config.xml 文件 (默认情况下如果是 windows 环境则再用户目录下的 .jenkins, linux 上则是当前用户下的一个隐藏文件夹, 通过 ls -lart 查看)

替换为:

1、<authorizationStrategy class="hudson.security.AuthorizationStrategy\$Unsecured"/>

这个权限对应“任何用户可以做任何事(没有任何限制)”

2、<authorizationStrategy class="hudson.security.FullControlOnceLoggedInAuthorizationStrategy"/>

这个权限对应“登录用户可以做任何事”

3、<authorizationStrategy class="hudson.security.GlobalMatrixAuthorizationStrategy">

<permission>hudson.model.Hudson.Administer:test</permission>

<permission>hudson.scm.SCM.Tag:test</permission>

</authorizationStrategy>

这个权限对应 test 用户可以是管理员、打标签权限。

2、如果要配置连接微软 ldap, 需要安装 Active Directory plugin。

比如配置:

Domain Name: XXXX.net

Domain controller:192.168.0.112:3268

LDAP 全局目录: TCP 端口 3268 (如果 DC 保持着全局目录的操纵权)

2.主目录设置:

主目录存放了 jenkins 运行时的配置文件, 安全策略、插件等等, 是十分重要的工作。在“系统管理-系统设置”页面, 第一行即为系统主目录设置, 当 jenkins 运行时, 主目录是不允许更改, 停止 jenkins 后可以通过下面方式更改主目录:

1.如果是 tomcat 启动, 则需要修改 tomcat 的 catalina.sh(windows 下是 catalina.bat), 在# OS specific support. \$var _must_ be set to either true or false.上面添加: export JENKINS_HOME=""

2.修改环境变量: 用 root 用户登录编辑 profile 文件: vi /etc/profile

在最后加入: export JENKINS_HOME=/search/autoTest/bizsurpport/hudson/.hudson

保存, 退出后执行: source /etc/profile 让配置生效, 如果是 windows 则再高级管理-环境变量中, 添加对应的用户变量即可。

3. 更改 Jenkins.war(或者在展开的 Web 容器)内的 web.xml 配置文件:

用 winrar 打开 jenkins.war, 进入 WEB-INF 找到 web.xml:

<!-- if specified, this value is used as the Hudson home directory -->

<env-entry>

<env-entry-name>HUDSON_HOME</env-entry-name>

<env-entry-type>java.lang.String</env-entry-type>

<env-entry-value></env-entry-value>

</env-entry>

在红色节点中间填入路径。

说明 3:

安全矩阵和项目矩阵授权策略的配置是一模一样的,唯一的区别是项目矩阵授权策略支持在 Job 的配置页面再次配置授权策略。

3.修改用户密码

在用户管理界面可以按步骤操作修改用户的密码,这里的密码都是加密之后的,如果忘了密码可以采用下面策略。

Jenkins 专有用户的数据存放在 JENKINS_HOME/users 目录,打开 config.xml 这个文件如下截图:

```
<?xml version='1.0' encoding='UTF-8'?>
<user>
  <fullName>juanfei</fullName>
  <properties>
    <hudson.model.PaneStatusProperties>
      <collapsed/>
    </hudson.model.PaneStatusProperties>
    <jenkins.security.ApiTokenProperty>
      <apiToken>V49GXkBsMr0E6IH5RB1P4Xw254iXC5b6m1HINE02YrNiS5aF7KAdfqsbyth/lHFK</apiToken>
    </jenkins.security.ApiTokenProperty>
    <com.cloudbees.plugins.credentials.UserCredentialsProvider _UserCredentialsProperty plugin="credentials@1.9.4">
      <domainCredentialsMap class="hudson.util.CopyOnWriteMap$Hash"/>
    </com.cloudbees.plugins.credentials.UserCredentialsProvider _UserCredentialsProperty>
    <hudson.model.MyViewsProperty>
      <views>
        <hudson.model.AllView>
          <owner class="hudson.model.MyViewsProperty" reference="../../../../"/>
          <name>All</name>
          <filterExecutors>false</filterExecutors>
          <filterQueue>false</filterQueue>
          <properties class="hudson.model.View$PropertyList"/>
        </hudson.model.AllView>
      </views>
    </hudson.model.MyViewsProperty>
    <hudson.search.UserSearchProperty>
      <insensitiveSearch>false</insensitiveSearch>
    </hudson.search.UserSearchProperty>
    <hudson.security.HudsonPrivateSecurityRealm _Details>
      <passwordHash>#jbcrypt:$2a$10$5EblCzFqOZY910JYlR/b1.hFSVeuXGeeCAzun7E.L515107tI7x0e</passwordHash>
    </hudson.security.HudsonPrivateSecurityRealm _Details>
    <hudson.tasks.Mailer _UserProperty plugin="mailer@1.6">
      <emailAddress>juanfei@qq.com</emailAddress>
    </hudson.tasks.Mailer _UserProperty>
    <jenkins.security.LastGrantedAuthoritiesProperty>
      <roles>
        <string>authenticated</string>
      </roles>
      <timestamp>1420869530504</timestamp>
    </jenkins.security.LastGrantedAuthoritiesProperty>
  </properties>
</user>
```

仔细看一遍这个 xml 文件,其实密码就存放在:

<passwordHash>#jbcrypt:\$2a\$10\$5EblCzFqOZY910JYlR/b1.hFSVeuXGeeCAzun7E.L515107tI7x0e</passwordHash>

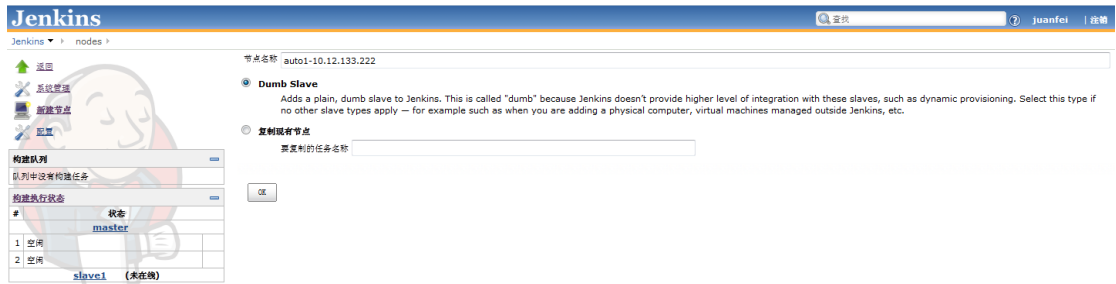
把忘记的密码替换为上面的字符串即可,这里对应密码是 1

3 配置 windows 和 linux 节点

所谓节点配置,就是 jenkins 支持分布式构建,分布式构建可以做到让同一套代码在不同的环境上编译、运行,对于测试来说可以指定主机来运行我们的自动化 case,对于运营同学来说可以将统一到代码发布到不同的机器上。

注意:不管是 windows 还是 linux 节点,都需要节点主机安装了 java 环境,即安装 jdk 并配置 JAVA_HOME。

点击“系统管理->管理节点->新建节点”:



输入节点名称，支持中文，一般情况下节点名称建议使用具有明确信息的关键字，否则将来节点多了查找会比较麻烦的事情。

例如我习惯 windows 节点带上 win 前缀区分 windows 环境。

如果是第一次新建一个节点，毫无疑问我们不能使用“已经现有节点”，但如果已经创建了一个节点当前创建的配置和之前的节点配置除了 ip 其他几乎一样，就可以使用复制的方式了。

选择 dumb slave 后：

说明：

Of executors: 最大同时构建数量（根据实际情况而定）

远程工作目录：如果目录不存在，会自动创建目录。你必须对该目录有读写权限，不然会报异常。

标签：用来对多节点分组

用法：下拉框有两个选项：尽可能使用该节点和只运行运行绑定到这台机器的 job。这里需要注意的是，如果我们运行的代码可以在所有节点上运行，那么不指定节点也没有问题，但如果这个段代码只能运行在 windows 或者只能运行在 linux 上，则需要正确设置该选项。

启动方法：该下拉框有四个选项，但本人使用的只是前两个，分别对应建立 linux 节点和 windows 节点：

1. Launch slave agents on Unix machines via SSH 在 Unix(包括 Linux)机器上通过 SSH 通道连接节点 (适用于 Unix 和 Linux

Host: 节点主机的 ip 地址

Credentials: 凭据(网上给的提示: 如果为空或者不可选择, 请在系统管理→Manage Credentials 中配置。Manage Credentials 的配置非常简单。Manage Credentials 配置完成后, 需刷新节点配置页面才会显示), 但实际使用当中, 这个凭据最好最这直接写入, uname 和 password 分别是使用 ssh 登录 linux 的用户名密码, description 可以随意。

说明：这里使用 ssh 登录时需要用户名密码验证（当然不用手动输入），还有另外一种方式是使用 linux 系统自带的信任机

制，想要学习同学百度一下关键字“linux 信任关系”

Port: 端口默认 22

JavaPath: [可选]JDK 路径，默认和 master 节点相同。路径必须指定到 Java 程序，如: /path/bin/java

JVM Options: [可选]JVM 可选参数

启动方法: Launch slave agents on Unix machines via SSH

Host: [redacted]

Credentials: [redacted] Add

Port: 22

JavaPath: /usr/local/jdk1.6.0_02/bin/java

JVM Options:

Prefix Start Slave Command:

Suffix Start Slave Command:

添加完成之后，在节点列表查看状态如下（注意如果有红色 X 则说明连接失败，可以在名字后面的下拉框查看系统消息）

linux132	8.00 GB	2848ms	In sync	1022.48 MB	Linux (amd64)	3.01 GB
master	7.22 GB	0ms	In sync	1.84 GB	Windows 7 (amd64)	7.22 GB
win-189	N/A	Time out for last 3 try	N/A	N/A		N/A

说明：建立 linux 到 linux 的连接、windows 到 linux 连接可以直接使用

2. Launch slave agents via Java Web Start 通过 Java Web Start 连接节点 (适用于所有支持 Java 程序的系统)

Tunnel connection through: [可选]在端口转发这种情况下使用

JVM options: [可选]JVM 可选参数

这种方法的缺点：如果该节点宕机了，主节点无法自动重启它。

启动方法: Launch slave agents via Java Web Start

Tunnel connection through:

JVM options:

Availability: Keep this slave on-line as much as possible

该几点配置好以后，进入节点 lanuch 页面：

Jenkins > nodes > win-189

Slave win-189 (rfs 自动化)

Connect slave to Jenkins one of these ways:

- Launch Launch agent from browser on slave
- Run from slave command line
- Or if the slave is headless

Created by juanfei

标签

关联到 win-189 的项目







该页面告诉我们，有三种方式可以启动这个节点：直接点击【lanuch】按钮或者进入命令行输入对应的命令

个人做法是创建一个 win189.bat 文件，即第三种启动方式：

win189.bat:

java -jar slave.jar -jnlpUrl <http://localhost:8080/computer/win-189/slave-agent.jnlp>

按任意一种方式启动后，如下：

 master	8.00 GB	Linux (amd64)	In sync	126.27 GB	1022.48 MB	0ms	
 slave	N/A		N/A	N/A	N/A	Time out for last 4 try	
 win-189	7.21 GB	Windows 7 (x86)	In sync	185.28 GB	N/A	2579ms	

同时会有这么一个 java 窗口：



说明：一般这种情况下，是 jenkins 部署在 linux 主机上，但是自动化代码只能运行在 windows 环境，这就需要建立一个 linux 到 windows 的连接。而两个 windows 之间也只能使用这种方式。

4 创建项目

首页左上角最显眼的位置就是“新建”：



输入项目名称后，选择第一个选项：构建一个自由风格的软件项目，同样，按语义理解“复制已有的 Item”只适用于已经创建了类似的项目。

☐ 丢弃旧的构建
☐ 参数化构建过程
☐ Promote builds when...
☐ 关闭构建 (重新开启构建前不允许进行新的构建)
☐ 在必要的时候并发构建
☐ Restrict where this project can be run

高级项目选项

源码管理

☐ CVS
☐ CVS Projectset
☒ None
☐ Subversion

构建触发器

☐ Build after other projects are built
☐ Build periodically
☐ Build when another project is promoted
☐ Poll SCM

构建

增加构建步骤

构建后操作

重点看一下这几个选项都该怎么设置:

丢弃旧的构建: 一般默认为不勾选

参数化构建过程: 参数化构建是 **jenkins** 和外界交互的利器, 使用参数化构建可以控制 **jenkins** 的构建过程, 也就是说我们可以通过将测试集以参数的形式传递给 **jenkins**, 指定运行哪些 **case**, 而不是每次运行所有 **case**。这样做显得更灵活实用。

☒ 参数化构建过程

String Parameter

名字: testcasesfileurl

默认值: http://localhost:8090/testcasefile

描述: 测试用例集合文件

[Escaped HTML] 预览

一般情况下, 测试集生成肯定是人为判断哪些用例需要执行之后, 指定用例生成的一个测试集合, 这个测试集合生成的方式不能在 **jenkins** 上实现, 最好的控制办法是在调度 **jenkins** 的时候指定构建过程。

Restrict where this project can be run: 指定当前项目运行的机器, 如果不指定, 则由 **jenkins** 自动安排一台机器运行

☒ Restrict where this project can be run

Label Expression: w|

win-189

源码管理:

源码管理

☐ None
☐ CVS
☐ CVS Projectset
☒ Subversion

Modules

Repository URL: http://[redacted]/AUTO/bizsurpport/crmapietest

Local module directory (optional): crmapitestcase

Repository depth: infinity

Ignore externals: ☐

Add more locations...

Check-out Strategy: Use 'svn update' as much as possible, with 'svn revert' before update

Do 'svn revert' before doing 'svn update'. This slows down builds a bit, but this prevents files from getting modified by builds.

Local module directory 设置路径为相对路径，即相对于节点 workspace+itemName/ 的路径，如果设置，则源码将会从 svn 上 checkout 到该目录下，不设置则 checkout 到节点 workspace+itemName/ 工程目录下。

Check-out Strategy: 尽量选择每次构建时都更新代码，确保运行的是最新的 case

增加构建步骤：

该步骤是我们自动化测试项目的核心步骤，case 运行调度在这里进行。例如我们事先已经写了一个 bat 脚本，该脚本运行我们的前端自动化 case：



addnewcustomer.bat:

```
@echo off
```

```
cd D:\Python27\Scripts\sample\Adminpc\客户管理
```

```
call pybot D:\Python27\Scripts\sample\Adminpc\客户管理\BO 添加新客户.txt
```

实际上配置了这一步，就创建成功了一个可运行的项目了。

5 运行项目

创建项目成功之后，会在“ALL”列表下有该项目



Name 后面会有一个按钮很醒目，点击即可执行项目里配置的内容。

点击 name 进入项目详情，在页面左上角也会有一个立刻构建入口，同样可以触发构建。

6 添加视图

1. 添加视图：在首页 ALL 后一个(+)号，点击可以按照提示添加视图，视图的作用是给众多项目进行分类，便于管理和查找。

All	+			
S	W	Name ↓	上次成功	
		test new account audit	没有	

新建项目的时候默认是创建在 ALL 下，新建视图的时候，可以使用 list view 选项，勾选想要分类的视图，“列”项目则显示该视图要显示的对应列信息，例如我们添加了 robot 插件后，如果要显示则需要在这个地方添加：

Job过滤器

状态过滤

所有选择的Jobs

Recurse in subfolders

☐

Jobs

☒ csdntest
☐ test new account audit

☐ 使用正则表达式在视图中显示Jobs

列

Status

删除

Weather

删除

Name

删除

Last Success

删除

Last Failure

删除

Last Duration

删除

分类之后会有下面效果：

All	besttest	csdn	+					
S	W	Name ↓	上次成功	上次失败	上次持续时间	Robot Results		
		csdntest	没有	无	无			
		test new account audit	没有	9月13 days - #9	3分50秒		0 / 15 passed	

图标：S M L

2. 用户视图：

当 team 比较大，项目比较多的时候，靠视图来分类查找也很快到瓶颈儿，因为视图也会很多了，这时候可以设置用户权限，不同的用户可以关注自己想要关注的视图。

以注册用户身份登录，点击 “My Views”：

Jenkins

新建

用户

任务历史

系统管理

Credentials

My Views

All

besttest

csdn

+

S	W	Name ↓
		csdntest
		test new account audit

图标：S M L

后选择 “Include a global view”：

名称	mycsdn
描述	<div>[Escaped HTML] 预览</div>
过滤构建队列	<input type="checkbox"/>
过滤构建执行器	<input type="checkbox"/>
View name	<div>All All besttest csdn</div>
<div>保存 应用</div>	

在 viewname 里选择自己要关注的视图即可。

目前来看，分用户使用视图的效果并不是很好，应该是 jenkins 在这个地方做得不是很精致。

7 配置管理工具 svn

新版本 svn 插件已经自动集成进来了，可以像 ssh 一样，配置用户名密码，按提示设置即可。


8 插件管理


其中标红部分是常用插件，需要重点关注学习

源代码管理

Hudson 本身支持 Subversion、CVS 以及下列插件：

-  BitKeeper Plugin — 为 Hudson 添加 BitKeeper(Linux 内核开发人员在全球使用的主要源代码工具)支持。
-  Visual SourceSafe Plugin — 该插件集成微软的 VSS 到 Hudson。
-  Git Plugin — 该插件允许使用 GIT 作为一个构建 SCM(源代码控制管理系统)，但必须使用 Git 1.3.3 及以上。
-  Team Foundation Server Plugin — 该插件集成了 Microsoft Team Foundation Server 源码控制到 Hudson 中。
-  CMVC Plugin — 该插件集成 CMVC(IBM 和许多跨国公司的缺陷管理工具。)到 Hudson。
-  File System SCM — Use File System as SCM.使用文件系统作为 SCM。
-  StarTeam — 该插件把 StarTeam(StarTeam 是一个集合了版本控制和缺陷控制两种功能的软件，并且具有 CVS 没有的强大的图形界面，易学易用。2002 年底被 Borland 公司收购。)和 Hudson 集成在一起。
-  **Subversion Plugin** — 该插件增加 Hudson 对 svn(通过 SVNKit)的支持。
-  Accurev Plugin — 该插件允许您在 Hudson 中使用 AccuRev 作为 SCM。
-  Template Project Plugin — 该插件可以让您使用另一个项目中的构建人、发布人和 SCM 设置。
-  URL SCM — 该插件允许您使用 URLs 作为 SCM。
-  Bazaar Plugin — 该插件集成 Bazaar 到 Hudson，该插件需要确保 Bazaar 库(bzr)已安装到目标机器上。
-  PVCS SCM — 该插件集成了 Serena 提供的 PVCS SCM。

 Mercurial Plugin — 该插件集成 Mercurial version control system 到 Hudson 中。

 Perforce Plugin — 该插件集成 Perforce 到 Hudson 中。

 Synergy Plugin — 该插件把 CM/Synergy 版本管理系统集成在 Hudson。

 Dimensions — 该插件集成 Dimensions SCM 到 Hudson 中。

 ClearCase Plugin — 把 ClearCase(IBM 提供的版本控制系统)集成到 Hudson。

跟 SCM 有关联的其它插件：


 SVNCompat14 Plugin — 该插件强制内置 SVNKit 库使用 Subversion 1.4 工作拷贝格式(替代它最近支持的 svn 版本。)


 Subversion Tagging Plugin — 该插件在成功的构建中执行 svn 标签(也就是使用 svn copy)。

 CVS Tagging Plugin — 该插件将在一个作业构建成功后执行 cvs 标签(也就是 cvs rtag)。

构建触发

 Ivy Plugin — 该插件通过配置 Apache Ivy 自动化地配置一次构建并触发其有构建依赖的其他构建。


 Join Plugin — 该插件允许一个作业在所有它下游的作业结束以后才开始运行它本身。


 Log-Action plugin — 该插件在作业输出中按选定模式开始一系列动作，但这是一个统一的配置。

 URL Change Trigger — 该网址变更触发器插件允许您当网址的内容发生变更时 Hudson 触发一次构建。

 Locks and Latches plugin — 该插件允许你管理作业的并发执行。

 IRC Plugin — 该插件在您选择的 IRC 频道中安装 Hudson IRC 机器人，您可以通过 IRC 获得通知，并通过 IRC 与 Hudson 互动。

 Downstream-Ext Plugin — 该插件为下游触发器支持扩展配置。当前它增加一个选项，如果它们发生 SCM 改变时，仅仅触发下游的构建。

 Jabber Plugin — 把 Jabber 即时消息协议集成在 Hudson 中。注意您也需要安装 instant-messaging 插件。

 Naginator Plugin — 该插件允许您在一次构建失败后自动重建一次构建。

构建工具

Hudson 本身支持 Maven、Ant、Shell 脚本和 Windows 批处理命令。

 Grails Plugin — 该插件允许 Hudson 调用 Grails 任务作为一个构建步骤。


 PowerShell Plugin — 在 Hudson 中集成 Windows PowerShell。

 Jython Plugin — 在 JVM 中增加执行 Jython 脚本的能力。

 Post build task — 该插件允许用户依据构建日志的输出执行一个 shell/批处理任务。

 Ruby Plugin — 该插件允许用户在构建脚本中使用 Ruby。

 Gant Plugin — 该插件允许 Hudson 调用 Gant 构建脚本作为主体构建的一个步骤。

 Seleniumhq Plugin — 该插件允许您从 Seleniumhq 中运行和加载通过 Selenium 服务器生成的 HTML Selenese 套件结果。Hudson 将以此生成测试结果的趋势报告。

 Groovy plugin — 该插件允许 Hudson 直接执行 Groovy 代码。

 Rake plugin — 该插件允许 Hudson 调用 Rake 任务作为构建步骤。




 NAnt Plugin — 该插件允许您在 Hudson 中使用 NAnt 构建.NET 项目。

 Template Project Plugin — 该插件可以让您在 Hudson 中使用另一个项目中的构建人、发布人和 SCM 设置。













 Buckminster Plugin — 该插件把 Eclipse Buckminster 做为一个新的构建步骤集成在 hudson 中。

 Python Plugin — 添加执行 Python 脚本作为 Hudson 的构建步骤。

 SCons Plugin — 该插件允许 Hudson 调用 SCons 构建脚本作为主体构建的步骤。









-
-  Selenium AES Plugin — 该插件能让 Hudson 调用 Selenium Auto Exec Server(AES) 来测试。
 -  Kundo Plugin — 该插件允许你调用 Kundo 构建作为一个 Hudson 构建步骤。
 -  Gradle Plugin — 该插件允许 Hudson 调用 Gradle 构建脚本作为主体构建的步骤。
 -  EasyAnt Plugin — 该插件允许 Hudson 调用 EasyAnt 构建脚本作为主体构建的步骤。
 -  RAD Builder Plugin — 该插件允许你调用 IBM RAD7.0/7.5 作为一个 Hudson 构建步骤。
 -  Phing Plugin — 该插件允许你在 Hudson 中使用 Phing 构建 PHP 项目。
 -  SCTMExecutor — 该插件允许您在 Hudson 中使用 Borland 的 SilkCentral Test Manager 2008 R2 及以上版本。
 -  MSBuild Plugin — 该插件允许您在 Hudson 中使用 MSBuild 构建.NET 项目。
 -  Batch Task Plugin — 该插件增加一些不规则执行的批处理任务到项目中，诸如打包、集成、归档等等。

构建封装


-  Xvnc Plugin — 该插件可以让您在一次构建时运行 Xvnc 会话。如果您的构建包括用户界面测试时需要一个可暴露的显示值这就非常方便。
-  ZenTimestamp Plugin — 该插件允许您在 Hudson BUILD_ID 变量中自定义日期和时间模式。
-  VMware plugin — 该插件允许您在一次构建开始前启动 VMware 虚拟机，构建完成后又停止该虚拟机。
-  Build Secret Plugin — 让你通过一次构建上传可用的秘密文件。
-  M2 Release Plugin — 这是一个能让您在 Hudson 中使用 maven-release-plugin 来执行发布功能的插件。
-  M2 Extra Steps Plugin — 该插件为 Hudson 加入 pre- 和 post- 构建步骤到 Maven 2 类型的项目的能力。
-  Template Project Plugin — 该插件可以让您使用另一个项目中的构建人、发布人和 SCM 设置。
-  Locks and Latches plugin — 该插件允许您控制作业的并发执行。
-  Hudson Distributed Workspace Clean plugin — 该插件允许您在同一个隶属组中未使用的隶属机器来清理工作区。
-  Release Plugin — 该插件允许您在当一次发布构建是手动触发时，配置预前构建或后构建的执行动作。
-  Copy To Slave Plugin — This plugin allows to copy, to slave nodes running a job, a set of files that are required for the good execution of the job.
-  Setenv Plugin — 为一个项目设置环境变量，以备构建步骤引用。


构建通知


Hudson 本身支持电子邮件通知，但也有一个可扩展的电子邮件插件来支持扩展功能。


-  Status Monitor Plugin — 该插件能让您在 Hudson 单独的屏幕上直观地显示您选定作业的状态。
-  SameTime Plugin — 该插件允许您在 Hudson 中使用 SameTime 作为构建通知。
-  Nabaztag Plugin — 该插件允许您在 Hudson 中发布一次构建的结果到 Nabaztag。
-  Google Calendar Plugin — 该插件允许 Hudson 发布当前作业状态到谷歌日历。
-  hudsonTracker — A cross-platform application that sits in your system tray and monitors Hudson builds via its RSS feeds. See hudsonTracker for more details. No Hudson server config required!
-  TuxDroid Plugin — 该插件允许您在 Hudson 发布一个构建的结果到 TuxDroid(Tux Droid 是一个强大的 Linux 玩具，它可以通过配套的软件与你电脑中的很多应用程序相关联)。
-  IRC Plugin — 该插件在您选择的 IRC 频道中安装 Hudson IRC 机器人，您可以通过 IRC 获得通知，并通过 IRC 与 Hudson 互动。
-  Email-ext plugin — This plugin allows you to configure every aspect of email notifications. You can customize when an


email is sent, who should receive it, and what the email says.

 Instant Messaging Plugin — This plugin provides generic support for IM notifications. This plugin itself is of no use. Please use one of the derived plugins like (at the moment only) Jabber !

 Blame Upstream Committers Plugin — This is a very simple plugin that adds a post build action to mail upstream committers when a build fails.

 Campfire Plugin — This plugin allows your team to setup build notifications to be sent to Campfire rooms.

 The new EMailer — Merge of the Core EMailer and email-ext plugin

 Twitter Plugin — This plugin posts build results to Twitter.该插件支持 Hudson 发送构建报告到 Twitter。

 Jabber Plugin — 把 Jabber 即时消息协议集成在 Hudson 中。注意您也需要把安装 instant-messaging plugin 。


隶属(机器)激活和控制

Hudson 本身支持通过 JNLP 或者命令启动隶属, 以及支持尽可能保持隶属连接, 或者按需连接。

 SSH Slaves plugin — 该插件允许您在 SSH 外的隶属* nix 机器上运行和管理 Hudson。

构建报告


Hudson 本身支持 Junit 报告和 JavaDoc。


 Robot Framework Plugin — 发布 robotframework 测试报告。


 Serenitec Plugin — 在您的项目中执行 Serenitec 重构方案。

 Violations — 该插件为 checkstyle、pmd、cpd、findbugs、fxcop、stylecop 和 simian 等静态代码分析工具生成报告。


 NUnit Plugin — 该插件允许你发布 NUnit 测试结果。

 WebTest Presenter Plugin — This plugin publishes the reports generated by the Canoo WebTest tool for each build.

 MSTest Plugin — 该插件允许您发布 MSTest 的测试结果。

 Crap4J Plugin — This plugin reads the "crappy methods" report from Crap4J. Hudson will generate the trend report of crap percentage and provide detailed information about changes.

 FindBugs Plugin — 该插件主要收集项目模块中 FindBugs(静态分析源代码中可能会出现的 Bug 的 Eclipse 插件工具)的分析结果, 并以视图的方式呈现已发现的警告。

 Seleniumhq Plugin — 该插件允许您从 Seleniumhq 中运行和加载通过 Selenium 服务器生成的 HTML Selenese 套件结果。Hudson 将以此生成测试结果的趋势报告。


 Cppcheck Plugin — 该插件为 CppCheck(静态的 C/C++ 代码分析工具)生成趋势报告。


 NCover Plugin — 该插件允许 Hudson 从 NCover 中归档和发布.NET 代码覆盖率的 HTML 报告。

 JDepend Plugin — 该 JDepend 插件是一个为构建生成 JDepend 报告的插件。


 Plot Plugin — 该插件为 Hudson 提供通用的测绘(或图表)的能力。



 Checkstyle Plugin — 该插件主要收集项目模块中 Checkstyle (自动化代码检查工具)的分析结果, 并以视图的方式呈现已发现的警告。

 JavaNCSS+Plugin — 该插件允许您使用 JavaNCSS 构建报告工具。







 SLOCCount Plugin — 该插件能为 SLOCCount 生成趋势报告, 它是一个开源程序, 能为超过 25 种不同的语言统计代码行的数量, 包括 C/C++、Ada、COBOL、Fortran、SQL、Ruby、Python 等等。

 JavaTest Report Plugin — 该插件把 JavaTest(一个通过 Sun 公司发布的 TCK 应用的 框架)中解析成 XML 结果文件, 并以此方式显示它们。




 Emma Plugin — 该插件在 Hudson 中集成 EMMA code coverage reports (检测和报告 JAVA 代码覆盖率的开源工具)。Hudson will generate the trend report of coverage.








-
-  Warnings Plugin — This plugin generates the trend report for compiler warnings in the console log or in log files.
 -  Gallio Plugin — 该插件允许您发布 Gallio/MbUnit 的测试结果。
 -  Doxygen Plugin —该插件可发布通过 Doxygen 工具生成的报告。
 -  Testability Explorer Plugin —该插件为 Testability Explorer 生成趋势报告，这是一个能使用字节码分析以在 Java 代码中寻找可测试性缺陷的开放源码程序。
 -  DRY Plugin — 该插件为像 CPD 那样的重复代码检查器生成趋势报告。
 -  Japex Plugin — 该插件为 Hudson 增加了 Japex 支持，以使 Hudson 能够显示其趋势报告和其他有用的数据。
 -  PMD Plugin — 该插件主要收集项目模块中 PMD(程序代码检查工具)的分析结果，并以视图的方式呈现已发现的警告。
 -  Selenium AES Plugin — 该插件调用 Selenium Auto Exec Server(AES) 来测试。
 -  Task Scanner Plugin — 该插件为开放任务扫描工作区文件，并生成一个趋势报告。
 -  PureCoverage plugin — PureCoverage plugin reports coverage results from Rational PureCoverage tool (c++ coverage tool).
 -  Gnat Plugin — 该插件允许 Hudson 集成 Gnat 的功能来作为 ADA(Ada 是一种表现能力很强的通用程序设计语言，它是美国国防部为克服软件开发危机，耗费巨资，历时近 20 年研制成功的。)语言。
 -  Grinder Plugin — 该插件从 Grinder 执行测试中读取输出结果，并生成报告，其中包括显示每次构建的测试结果和对应的构建显示执行结果的趋势报告。
 -  JSUnit plugin — This plugin allows you publish JSUnit test results
 -  Cpptest Plugin — This plugin allows you publish Parasoft C++test test results.
 -  Clover Plugin — 该插件是在 Hudson 中集成 Clover code coverage reports(代码覆盖测试分析工具)。Hudson will generate and track code coverage across time. This plugin can be used without the need to modify your build.xml.
 -  Cobertura Plugin — 该插件允许您从 Cobertura 中获取代码覆盖率报告。Hudson 将生成覆盖率趋势报告。
 -  eXtreme Feedback Panel Plugin — This plugin provides an eXtreme Feedback Panel that can be used to expose the status of a selected number of Jobs.
 -  CppUnit Plugin — 该插件允许您发布 CppUnit 测试结果。
 -  Ruby metrics plugin — 该插件添加 Ruby 统计报告(Rcov, Saikuro, Rails stats...)的捆绑包到 Hudson 中。

构件上传

- 
-  SCP plugin — 该插件允许你使用 SFTP (SSH)协议上传一些构件到仓库站点。
 -  FTP-Publisher Plugin — 该插件能上传项目构件和整个目录到一个 FTP 服务器。
 -  SFEE Plugin — 依赖于 Collabnet Source Forge Enterprise Edition (SFEE 是 SourceForge 发布的项目协作管理软件) 服务器验证用户并发布构件。
 -  java.net uploader Plugin — 该插件使用 java.net 任务库，以使 Hudson 有能力发送构件到 java.net。
 -  SVN Publisher — This plugin allows you to upload artifacts to a subversion repository. This is done via a delete/import of the items requested.

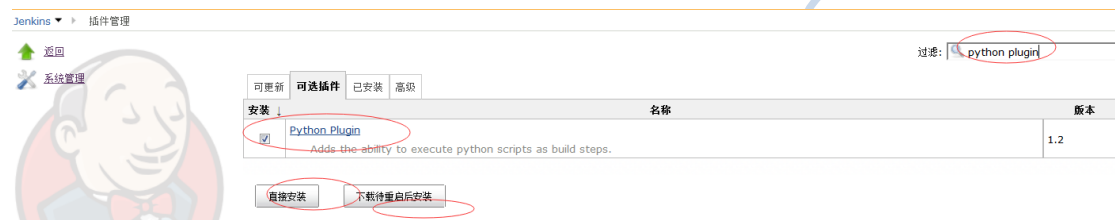
其他构建后操作

-  Parameterized Trigger Plugin — 该插件可以让你在构建完成后触发新的构建，并为这次新构建使用不同的方式指定一些参数。
-  Build Publisher Plugin — 该插件允许您把一个 Hudson 的记录发布到另一个 Hudson 中。
-  Post build task — 该插件允许用户依据构建日志的输出执行一个 shell/批处理任务。Java regular expression are allowed.

-  **Description Setter Plugin** — 该插件为每次构建设置描述信息，它是一个基于正则表达式校验的构建日志文件。
-  **Deploy Plugin** — This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build
-  **DocLinks Plugin** — 该插件允许您发布在构建步骤中创建的文档。
-  **Subversion Tagging Plugin** — This plugin performs subversion tagging (technically speaking svn copy) on successful build.
-  **CVS Tagging Plugin** — 该插件将在一个作业构建成功后执行 cvs 标签(也就是 cvs rtag)。
-  **CopyArchiver Plugin** — 该插件的目标是从几个作业中把已归档的构件集中到一个共享目录。每个作业中只有最后一次成功构建的归档构件才会被复制。
-  **Text-finder Plugin** — 该插件是用来在工作区文件中搜索字符串。这个搜索结果可以用来标记该构建是正常或者失败。

插件安装方式，以 python plugin 为例：

在“系统管理-插件管理”页面过滤条件输入 python：



什么情况下选择直接安装或者下载待重启后安装？

“直接安装”和“下载待重启后安装”都是jenkins服务器直连互联网，下载插件并安装，即前提条件是jenkins服务器能够联网，直接点击即可安装。

但如果jenkins服务器没有连到互联网，这种情况下我们需要下载该插件到本地，然后在“高级”选项里导入：

点击上面的名称链接，即进入下载页面：



Python Plugin

Added by [R. Tyler Croy](#), last edited by [Larry Shalzer](#) on Aug 05, 2013 ([view change](#))

Jenkins

Home
Mailing lists
Source code
Bugtracker
Security Advisories
Events
Donation
Commercial Support
Wiki Site Map

Documents

Meet Jenkins
Use Jenkins
Extend Jenkins
Plugins
Servlet Container Notes

Plugin Information

Plugin ID	python	Changes	In Latest Release Since Latest Release
Latest Release	1.2 (archives)	Source Code	GitHub
Latest Release Date	Nov 16, 2011	Issue Tracking	Open Issues
Required Core	1.409	Maintainer(s)	R. Tyler Ballance (id:)
Dependencies	正在打开 python.hpi		
Usage	ns 2014-Jan 3464 2014-Feb 3537 2014-Mar 3708 2014-Apr 3703 2014-May 3829 2014-Jun 3817 2014-Jul 4031 2014-Aug 4103 2014-Sep 4274 2014-Oct 4426 2014-Nov 4406 2014-Dec 4381		

Adds the al

ks pretty much like the stand

Change Log

点击 Latest Release 后的链接，保存 hpi 文件到本地

然后在“高级”选项卡里，找到“上传插件”：

上传插件

您可以通过上传一个 .hpi 文件来安装插件。

文件: 浏览...

上传

将下载下来的 hpi 上传接口安装成功

9 Unittest 框架与 jenkins 集成，运行并生成测试报告

1、安装 xmlrunner 插件（xmlrunner 是 unittest 运行的插件，不是 jenkins 插件）

安装 xmlrunner:

下载地址: <https://github.com/xmlrunner/unittest-xml-reporting>

wget <https://github.com/xmlrunner/unittest-xml-reporting/archive/master.zip>

unzip master.zip

cd unittest-xml-reporting

sudo python setup.py install

修改为 xmlrunner，只需要将我们 run 入口修改两行代码：

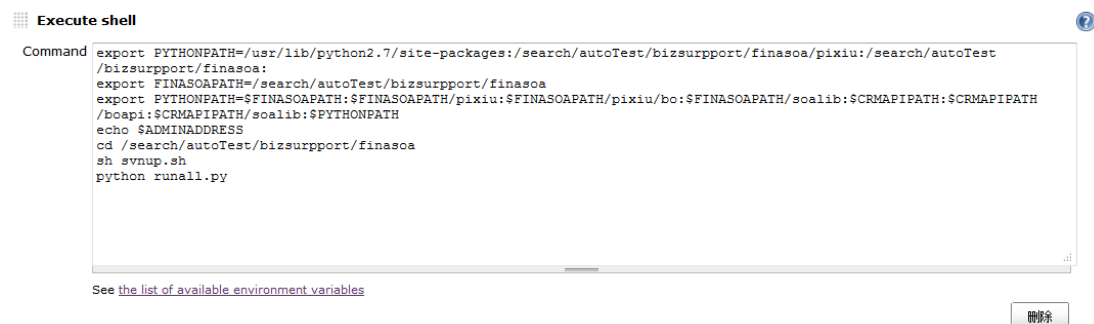
```
if __name__ == '__main__':  
    xmlrunner.XMLTestRunner(output='test-reports', verbose=1)  
    runner.run(t1.suite())
```

说明：

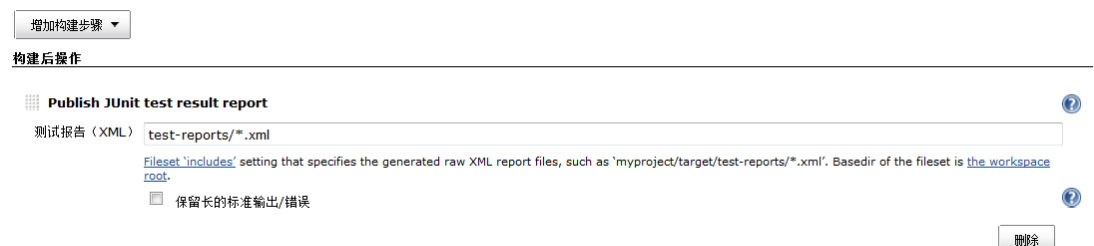
Output 属性设置 xml 文件路径

2、创建 job，增加构建步骤

构建内容如下：



设置构建后操作，将 xmlrunner 生成的 junit 报告输出到 report 中：



运行后报告效果如下：

Test Result

0次失败

3个测试
花了
添加说明

所有的测试

Package	花的时间	失败	(区别) 跳过	(区别) Pass	(区别) 总数	(区别)
case.CooperationServiceTest	35 毫秒	0	0	1	+1	1 +1
case.CustomerServiceTest	1.3 秒	0	0	2	+2	2 +2

3、运行报告和自动发送报告

在“增加构建后操作步骤”选择 Email-Notification (建议使用 Editable Email Notification，使用方式参见自动发送测试报告)

Editable Email Notification

Disable Extended Email Publisher ☐

Project Recipient List

Project Reply-To List

Content Type

Default Subject

Default Content

Allows the user to disable the publisher, while maintaining the settings

Comma-separated list of email address that should receive notifications for this project.

Comma-separated list of email address that should be in the Reply-To header for this project.

HTML (text/html)

\$DEFAULT_SUBJECT

<hr/>
(本邮件是程序自动下发的，请勿回复！)

项目名称: \$PROJECT_NAME

构建编号: \$BUILD_NUMBER

svn版本号: \${SVN_REVISION}

构建状态: \$BUILD_STATUS

触发原因: \${CAUSE}

构建日志地址: <a href=\${BUILD_URL}console=\${BUILD_URL}console

构建地址: <a href=\${BUILD_URL}\${BUILD_URL}

变更集:\${JELLY_SCRIPT,template="html"}

10 robot framework 与 jenkins 集成，运行并生成测试报告

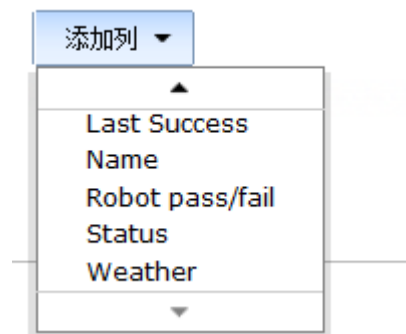
1. 安装 Robot Framework Plugin 插件

在插件管理的 filter 中搜索 robot，选择下面插件：

Robot Framework Plugin

This publisher stores [Robot Framework](#) test reports for builds and shows summaries of them in project and build views along with trend graph.

安装方式可以参考插件管理，安装成功后在“编辑视图”-“列”，添加列下拉框会出现“Robot pass/fail”



添加进来之后，在视图 job 列表会出现 robot result 列：

S	W	Name ↓	上次成功	上次失败	上次持续时间	Robot Results
		test	没有	无	无	
		finaweb-allautocase	没有	7 月 18 days - #7	0.76 秒	
		money	没有	1 day 20 小时 - #56	47 秒	0 / 2 passed

11 unittest 框架参数化运行

参数化运行的案例比较少，这几个参数化使用的优劣也不是很好比较，这里只介绍本人使用的参数化运行的方式，供大家参考。

1. 首先参数化构建过程选择 String Parameter。这个参数要求接收一个 url，该 url 对应的是生成好的一个测试集合文件

☒ 参数化构建过程

String Parameter

名字testcasesfileurl

默认值http://localhost:8090/testcasefile

描述测试用例集合文件

构建过程里，我们需要使用该 url：

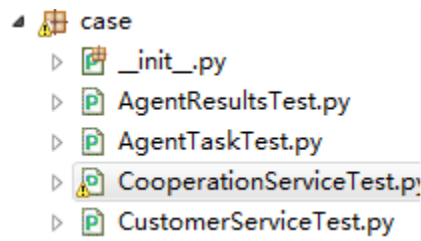
Execute shell

Commandsource ~/.bash_profile
source /etc/profile
if [\$testcasesfileurl]; then
 wget \$testcasesfileurl -O testcases
fi
/usr/local/bin/python runall12.py

首先导入环境变量（有时候不导入会报错），然后根据传入的参数，wget 到我们测试用例集文件，命名为 testcases。

2.介绍一下我们的 case 结构：

测试用例文件集合：



测试用例文件里的测试用例

```
46 class CooperationServiceTest(unittest.TestCase):
47     def setUp(self):
48         pass
49     def tearDown(self):
50         pass
51     def testAddCommunicationItemResult(self):
59     def testQueryCommunicationItemsByAccountId(self):
71     def testQueryLastCommunicationItemByAccountIdResult(self):
```

所有 case 保存在 svn 指定目录。

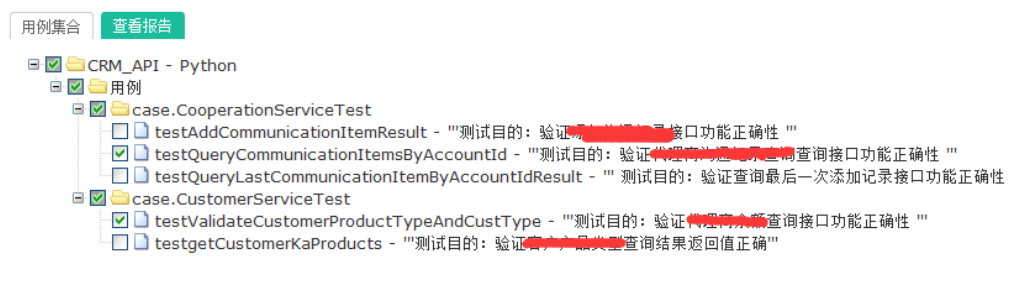
3.测试集合文件的格式:

testcases:

case.CustomerServiceTest,testValidateCustomerProductTypeAndCustType

case.CustomerServiceTest,testGetCustomerKaProducts

该文件是由测试用例管理系统，在前端勾选要运行的测试用例，然后将测试用例完整路径写入文件，生成的测试集合：



这是测试用例集合界面，实现原理也就是读取用例文件，然后按指定格式展示用例的名字和文档注释，生成可读性好的测试用例集合。

4.测试集合文件传过来的是测试用例名字的集合，是不能直接放在 unittest 里运行的，这就需要做一下反射，将字符串对象转换为可以执行的测试用例，同时生成测试集合 suite，交给 unittest 运行：

runall2.py:

```
from case import CooperationServiceTest, CustomerServiceTest
import sys
import unittest
import xmlrunner
if __name__ == '__main__':
    runner=xmlrunner.XMLTestRunner(output='./test-reports',verbose=1)
    suite=unittest.TestSuite()
    with open('testcase.conf') as f:
        for line in f.readlines():
            #从文件读取出类名和 case 方法名
```

```

classname=line.split(',')[0].strip()

casename=line.split(',')[1].strip()

#将类名做一次反射，由字符串转换为 class

amod=sys.modules[classname]

aclass=getattr(amod,classname.split('.')[1])

suite.addTest(aclass(casename))

runner.run(suite)

```

5.输出测试报告:

Publish JUnit test result report

测试报告 (XML):

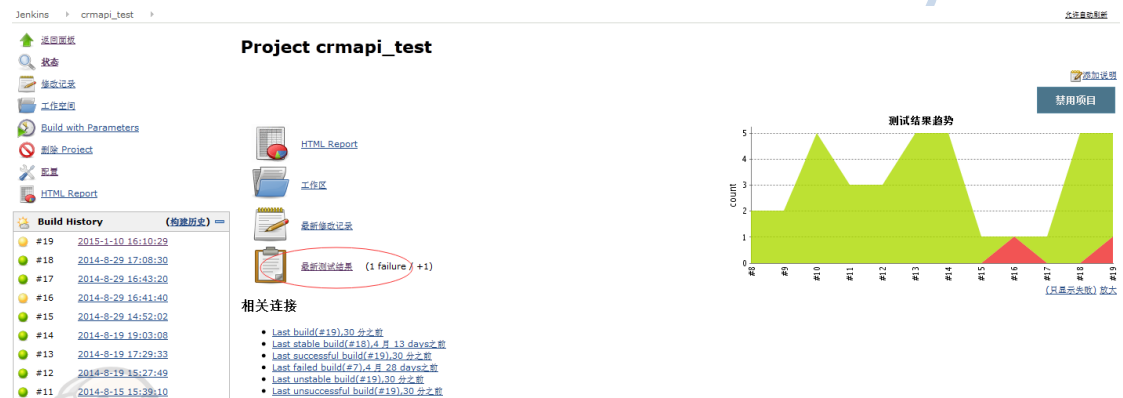
Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is [the workspace root](#).

☐ 保留长的标准输出/错误

删除

注意这里的 XML 路径，也是相对于工作空间的路径，前提条件是使用 unittest 的 xmlrunner 扩展包来运行这些 case，即上面代码中的：runner=xmlrunner.XMLTestRunner(output='./test-reports',verbose=1)

最终测试一次构建之后的界面如下：



点击“最新测试结果”：

Test Result

1次失败 (+1)

5个测试 (+0)

添加测试

所有失败的测试

测试的名称	花的时间	寿命
case_CooperationServiceTest_CooperationServiceTest.testQueryLastCommunicationItemByAccountIdResult	0.85 秒	1

所有的测试

Package	花的时间	失败	跳过	通过	Pass	总数	失败率
case_CooperationServiceTest	2.5 秒	1	+1	0	2	-1	3
case_CustomerServiceTest	1.7 秒	0	0	2	2	2	2

12 RobotFramework 框架参数化运行

1. 首先参数化构建过程选择 String Parameter。这个参数要求接收一个 url，该 url 对应的是生成好的一个测试集合文件



在增加构建步骤里里，我们需要使用该 url:

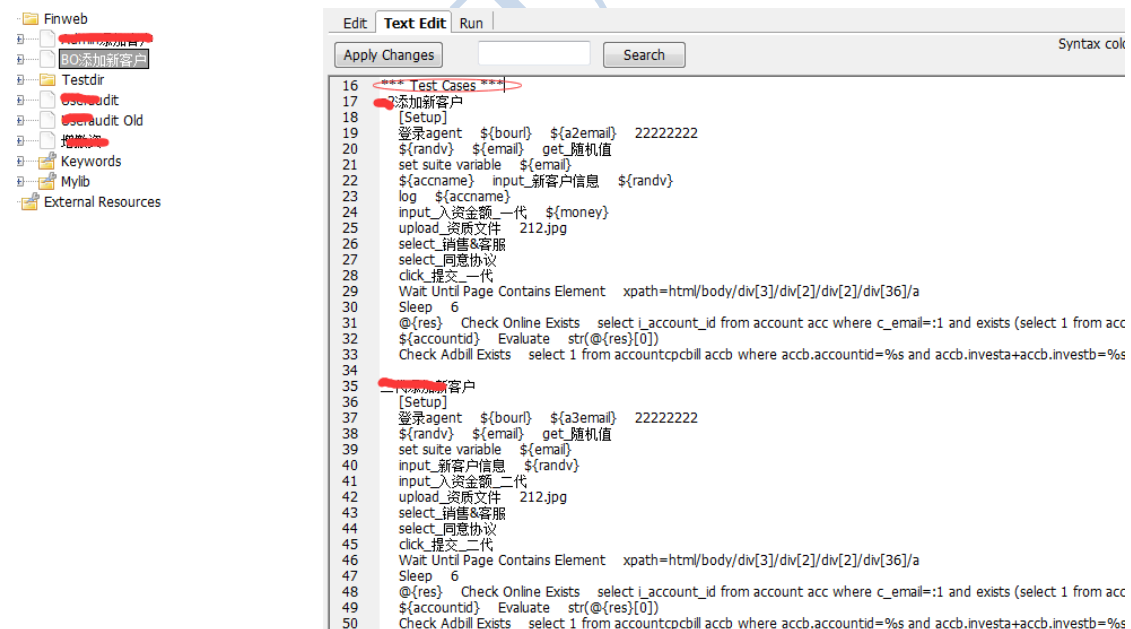


在 windows 下没有 wget 命令，需要去网上下载一个（下载地址：<http://users.ugent.be/~bpuype/wget/>），不清楚怎么使用的请百度“windows wget”

根据传入的参数，wget 到我们测试用例集文件，命名为 case.txt。

2.介绍一下我们的 case 结构:

测试用例文件和测试用例内容集合:



建议将所有 case 保存在 svn 指定目录，通过 svn 插件将 case 下载到本地运行:

源码管理

CVS

CVS Projectset

None

Subversion

Modules

Repository URL

http://[redacted]@10.129.148.126:8001/bizsurpport/webautocase/money

Local module directory (optional)

.

Repository depth

infinity

Ignore externals

☐

Add more locations...

Check-out Strategy

Use 'svn update' as much as possible

Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

源码库浏览器

(自动)

3.测试集合文件的格式:

```
--suite
Money.Useraudit Old
--test
Money.Useraudit Old.add_and_pass
--suite
Money.Useraudit Old
--test
Money.Useraudit Old.add_and_refuse
```

该文件是由测试用例管理系统，在前端勾选要运行的测试用例，然后将测试用例完整路径写入文件，生成的测试集合：

10.129.148.126:8001/caselist/

用例集合

查看报告

资金前端用例 - Robot

case

Money.Useraudit Old

Money.Useraudit Old.add_and_pass - 资金前端

Money.Useraudit Old.add_and_refuse - 资金前端

Money.Useraudit Old.KA_pass - 资金前端

Money.Useraudit Old.KA_refuse - 资金前端

Money.Useraudit

Money.增撤资

Money.Admin添加客户

Money.BO添加新客户


这是测试用例集合界面，实现原理也就是读取用例文件，然后按指定格式展示用例的名字和文档注释，生成可读性好的测试用例集合。

4.测试集合文件传过来的是测试用例名字的集合，同时也可以指定运行时的其他变量参数，这样可以更精细化控制 case 执行，包括运行时使用的变量内容等

5.输出测试报告：

在构建后操作步骤增加“Publish Robot Framework test result”

构建后操作

 Publish Robot Framework test results

Directory of Robot output

Path to directory containing robot xml and html files (relative to build workspace)

高级...

Thresholds for build result

%

50.0

%

100.0

☒ Use thresholds for critical tests only


删除

增加构建后操作步骤

其中 Directory of Robot output, 和运行时 argumentfile 里指定 outputdir 属性一致, 如果默认 outputdir 则该位置也是用默认值。


Build result 对应的两个天气标记可根据喜好填写。

最终测试一次构建之后的界面如下:



构建 #56 (2015-1-18 16:28:37)


启动时间
构建用时




修正版本号: 9847

Changes

1. web应用前端自动化 ([detail](#))



由匿名用户触发



Robot Test Summary:

	Total	Failed	Passed	Pass %
Critical tests	2	2	0	0.0
All tests	2	2	0	0.0

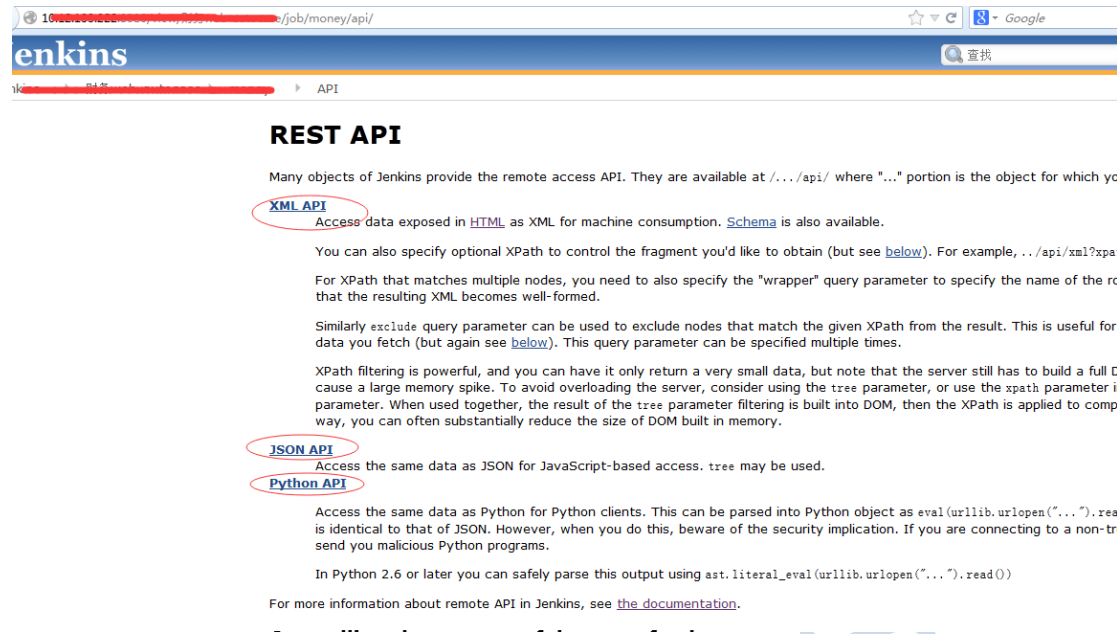
> [Browse results](#)

> [Open report.html](#)

> [Open log.html](#)

13 Api 介绍和使用方法

在 job 对应的 url 后面直接输入 api:



系统会给出三个 api 调用方式。其中 xml 和 jsonapi 是多种语言通用的方式

1、xml api, 点击 XML API 超链接可以获取到 xml 文件, 常用的方式是使用 Pycurl 操作这个 xml 文件, Pycurl 用法可以参考文章 (<http://www.cnblogs.com/lonelycatcher/archive/2012/02/09/2344005.html>)

2、最简单的 api 调用方法 Python API:

Python 扩展包 jenkinsapi 下载安装: <https://pypi.python.org/pypi/jenkinsapi/>

使用方式:

```
if __name__ == '__main__':
    jsrv=Jenkins('http://10.12.133.222:8080')
    print(jsrv.keys())
    print([jsrv.get_job('money')])
    try:
        jsrv.build_job('finweb',{'url':'http://www.sogou.com'})
    except:
        pass
    print(jsrv['finweb'].get_last_build().baseurl)
```

在 pydev 里, 我们可以通过自动提示, 查看所有 Jenkin 对象和 job 对象都有哪些方法可供调用。

14 自动发送测试报告

1、Jenkins 自带邮件发送:

自带邮件发送功能配置比较简单, 进入“系统管理-系统设置”, 点开高级:

邮件通知

SMTP服务器	smtp.126.com	?
用户默认邮件后缀		?
<input checked="" type="checkbox"/> 使用SMTP认证		?
用户名	yuleng662@126.com	
密码	*****	
使用SSL协议	<input type="checkbox"/>	?
SMTP端口		?
Reply-To Address		
字符集	UTF-8	
<input type="checkbox"/> 通过发送测试邮件测试配置		

Smtp 服务器需要去网上百度一下自己邮箱注册的域名提供的 smtp 服务地址。

配置好以后就可以使用“通过发送测试邮件中测试配置”验证是否能发送成功。

注意：有些邮件服务器会验证发件人设置和登录服务器的账户是否一致，所有邮件默认发件人是系统管理员地址，设置位置：

Jenkins Location

Jenkins URL	http://10.10.123.33:8080/	?
系统管理员邮件地址	yuleng662@126.com	?

2. 扩展插件发送自定义邮件：由于 jenkins 自带的邮件发送通知，有很多局限性，邮件通知无法提供详细的邮件内容、无法定义发送邮件的格式、无法定义灵活的邮件接收配置等等，在这样的情况下，我们找到了 Jenkins Email Extension Plugin 这个插件。点击“系统管理-管理插件-可选插件”，在过滤文本框输入“email-ext”，

过滤: email-ext

可更新	可选插件	已安装	高级
安装 ↓			
<input type="checkbox"/>	Email-ext plugin This plugin allows you to configure every aspect of email notifications. You can customize when an email is sent, who should receive it, and what the email says.	2.39	
<input type="checkbox"/>	Email Ext Recipients Column Plugin This plugin allows you to add a column showing the recipients configured in email-ext-plugin for every project.	1.0	

这里我们用第一个 email-ext plugin

有网的情况下可以点击在线安装，也可以像安装 robot 扩展包一样，在网盘找到对应的插件然后导入。

配置扩展插件：

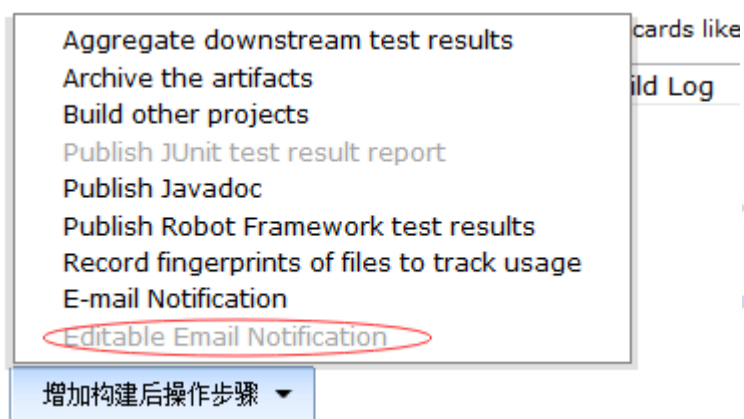
点击“系统管理-系统设置”下拉到页面最下方，找到标题为“Extended E-mail Notification”，主要需要的配置项如下：

Extended E-mail Notification

<input checked="" type="checkbox"/> Override Global Settings	
SMTP server	smtp.126.com
Default user E-mail suffix	
System Admin E-mail Address	yuleng662@126.com
<input checked="" type="checkbox"/> Use SMTP Authentication	
User Name	yuleng662@126.com
Password	*****
Use SSL	<input type="checkbox"/>
SMTP port	
Charset	UTF-8
Default Content Type	Plain Text (text/plain)
<input type="checkbox"/> Use List-ID Email Header	
<input type="checkbox"/> Add 'Precedence: bulk' Email Header	
Default Recipients	yuleng662@126.com

Reply To List	<input type="text"/>	?
Emergency reroute	<input type="text"/>	?
Excluded Recipients	<input type="text"/>	?
Default Subject	\$PROJECT_NAME - Build # \$BUILD_NUMBER - \$BUILD_STATUS!	?
Maximum Attachment Size	<input type="text"/>	?
Default Content	\$PROJECT_NAME - Build # \$BUILD_NUMBER - \$BUILD_STATUS: Check console output at \$BUILD_URL to view the results.	?
Default Pre-send Script	<input type="text"/>	?

然后，进入我们的项目，在“配置”里，下拉到最后，在“增加构建后步骤”中选择：



这是默认数据如下：

Editable Email Notification		?
Disable Extended Email Publisher	<input type="checkbox"/>	?
Allows the user to disable the publisher, while maintaining the settings		
Project Recipient List	<input type="text" value="example@company.com"/>	?
Comma-separated list of email address that should receive notifications for this project.		
Project Reply-To List	<input type="text" value="\$DEFAULT_REPLYTO"/>	?
Comma-separated list of email address that should be in the Reply-To header for this project.		
Content Type	<input type="text" value="Default Content Type"/>	?
Default Subject	<input type="text" value="\$DEFAULT_SUBJECT"/>	?
Default Content	<input type="text" value="\$DEFAULT_CONTENT"/>	?
Attachments	<input type="text"/>	?
Can use wildcards like 'module/dist/**/*.zip'. See the @includes of Ant files for the exact format. The base directory is the workspace .		
Attach Build Log	<input type="text" value="Attach Build Log"/>	?
Content Token Reference	<input type="text"/>	?
Advanced Settings...		

对于默认内容，我们可以稍加修饰：

(本邮件是程序自动下发的，请勿回复！)
<hr/>

项目名称：\$PROJECT_NAME
<hr/>

构建编号：\$BUILD_NUMBER
<hr/>

svn 版本号：\${SVN_REVISION}
<hr/>

构建状态：\$BUILD_STATUS
<hr/>

触发原因: \${CAUSE}
<hr/>

构建日志地址: \${BUILD_URL}console
<hr/>

构建地址: \${BUILD_URL}
<hr/>

变更集: \${JELLY_SCRIPT,template="html"}
<hr/>

<hr/>

再次构建我们的项目，将会看到发送的测试报告。

项目名称: finsoa-autocase

构建编号: 40

svn版本号: 400


构建状态: Still Failing

触发原因: Started by user anonymous

构建日志地址: <http://10.12.133.222:8080/job/finsoa-autocase/40/console>

构建地址: <http://10.12.133.222:8080/job/finsoa-autocase/40/>

变更集:

 **BUILD FAILURE**
Build URL: <http://10.12.133.222:8080/job/finsoa-autocase/40/>
Project: finsoa-autocase
Date of build: Sat, 10 Jan 2015 19:55:05 +0800
Build duration: 0.32 秒

CHANGES

No Changes

该插件使用到的全局属性详解:

1. Override Global Settings: 如果不选，该插件将使用默认的 E-mail Notification 通知选项。反之，您可以通过指定不同于(默认选项)的设置来进行覆盖。
2. Default Content Type: 指定构建后发送邮件内容的类型，有 Text 和 HTML 两种
3. Use List-ID Email Header: 为所有的邮件设置一个 List-ID 的邮件信头，这样你就可以在邮件客户端使用过滤。它也能阻止邮件发件人大部分的自动回复(诸如离开办公室、休假等等)。你可以使用你习惯的任何名称或者 ID 号，但是他们必须符合如下其中一种格式(真实的 ID 必须要包含在<和>标记里):
<ci-notifications.company.org>
Build Notifications <ci-notifications.company.org>
“Build Notifications” <ci-notifications.company.org>
4. Add 'Precedence: bulk' Email Header: 设置优先级
5. Default Recipients: 自定义默认电子邮件收件人列表。如果没有被项目配置覆盖,该插件会使用这个列表。您可以在项目配置使用\$ DEFAULT_RECIPIENTS 参数包括此默认列表，以及添加新的地址在项目级别。添加抄送: cc:电子邮件地址例如,CC:someone@somewhere.com
6. Reply To List: 回复列表
7. Emergency reroute: 如果这个字段不为空，所有的电子邮件将被单独发送到该地址
8. Excluded Committers: 防止邮件被邮件系统认为是垃圾邮件,邮件列表应该没有扩展的账户名(如:@domain.com),并且使用逗号分隔
9. Default Subject: 自定义邮件通知的默认主题名称。该选项能在邮件的主题字段中替换一些参数，这样你就可以在构建中包含指定的输出信息。
10. Maximum Attachment Size: 邮件最大附件大小。
11. Default Content: 自定义邮件通知的默认内容主体。该选项能在邮件的内容中替换一些参数，这样你就可以在构建中包含指定的输出信息。
12. Default Pre-send Script: 默认发送前执行的脚本
13. Enable Debug Mode: 启用插件的调试模式。这将增加额外的日志输出，构建日志以及 Jenkins 的日志。在调试时是有用的，但不能用于生产。

14. Enable Security: 启用时，会禁用发送脚本的能力，直接进入 Jenkins 实例。如果用户试图访问 Jenkins 管理对象实例，将抛出一个安全异常

15. Content Token Reference: 邮件中可以使用的变量，所有的变量都是可选的

全局变量详解:

`${BUILD_NUMBER}` 显示当前构建的编号。

`${JOB_DESCRIPTION}` 显示项目描述

`${SVN_REVISION}` 显示 svn 版本号。还支持 Subversion 插件出口的 `SVN_REVISION_n` 版本

`${CAUSE}` 显示谁、通过什么渠道触发这次构建

`${CHANGES}` -显示上一次构建之后的变化。

`${BUILD_ID}`显示当前构建生成的 ID。

`${PROJECT_NAME}` 显示项目的全名

`${PROJECT_DISPLAY_NAME}` 显示项目的显示名称

`${JENKINS_URL}` 显示 Jenkins 服务器的 url 地址（你可以再系统配置页更改）

`${BUILD_LOG_MULTILINE_REGEX}`按正则表达式匹配并显示构建日志。

`${BUILD_LOG}` 显示最终构建日志

`${PROJECT_URL}` 显示项目的 URL 地址

`${BUILD_STATUS}` -显示当前构建的状态(失败、成功等等)

`${BUILD_URL}` -显示当前构建的 URL 地址

`${CHANGES_SINCE_LAST_SUCCESS}` -显示上一次成功构建之后的变化

`${ENV}` - 显示一个环境变量。 `var` - 显示该环境变量的名称。如果为空，显示所有，默认为空。

`${FAILED_TESTS}` -如果有失败的测试，显示这些失败的单元测试信息。

`${JENKINS_URL}` -显示 Jenkins 服务器的地址。(你能在“系统配置”页改变它)。

`${PROJECT_URL}` -显示项目的 URL。

`${TEST_COUNTS}` -显示测试的数量。 `total` -所有测试的数量。 `fail` -失败测试的数量。 `skip` -跳过测试的数量。

另外我们可以尝试高级配置，主要实现在哪些情况下触发邮件发送以及发件人列表设置。这些设置将根据特殊需求来选择。

