

Luntbuild - 自动化构建与管理，用户手册

2005Luntbuild

目录

1. 介绍	1
2. 登录到Luntbuild	2
3. Luntbuild主页	3
4. 设置Luntbuild的系统属性	4
5. 添加Luntbuild用户	6
6. 创建一个项目	7
7. 为项目指定版本控制系统	9
7.1. 设置AccuRev连接信息	9
7.2. 设置Clearcase基本连接信息	9
7.3. Setting Cvs connection information. (设置Cvs连接信息)	10
7.4. Setting File system connection information. (设置文件系统的连接信息) ...	11
7.5. Setting Perforce connection information. (设置Perforce连接信息)	11
7.6. Setting Subversion connection information. (设置Subversion的连接信息) ..	12
7.7. Setting Clearcase UCM connection information. (设置Clearcase UCM的连接信息)	13
7.8. Setting Visual Sourcesafe connection information. (设置VSS连接信息)	14
7.9. Setting StarTeam connection information. (设置StarTeam连接参数)	16
7.10. Using multiple Version Control Adaptors. (同时使用多个VCS连接)	16
8. 新建VCS模块	17
8.1. 设置AccuRev模块信息	17
8.2. 设置cvs模块信息	17
8.3. 设置Perforce模块信息。	18
8.4. 设置Subversion模块信息	18
8.5. 设置VSS模块信息	20
8.6. 设置StarTeam模块信息	20
9. 创建项目的Builder	21
9.1. 配置Ant Builder	21
9.2. 配置命令行Builder	23
9.3. 配置Maven Builder	23
10. 创建项目的构建计划	26
11. 定义项目的登录映射表	33
12. 创建/修改项目的Ant Builder	34
13. 创建修改项目的Maven Builder	35
14. 构建计划的快照	36
15. 项目快照	39
16. 使用Luntbuild远程调用接口	40
17. 调试构建过程中的问题	41
18. Luntbuild安全性	42
18.1. 整体安全概念一览	42
18.2. 怎样配置Luntbuild的安全	42
19. 数据导出和导入	44
19.1. 数据导出	44
19.2. 数据导入	44
19.3. 通过导出和导入功能来进行数据迁移	44

A. 构建时支持的OGNL 表达式	45
-------------------------	----

第 1 章 介绍

原著: Luntbuild Team, 翻译: Melthaw Zhang [RedSaga]

Luntbuild是一种基于流行的构建工具— Apache Ant [<http://ant.apache.org>]- 的自动化构建和管理工具。通过Luntbuild, 可以很容易做到日构建和持续集成。如果您对日构建和持续集成并不熟悉, 请参考一下关于日构建和持续集成的文章。

Continuous Integration [<http://www.martinfowler.com/articles/continuousIntegration.html>]

Daily Builds Are Your Friend [<http://www.joelonsoftware.com/articles/fog0000000023.html>]

首先, Luntbuild团队感谢您选择Luntbuild作为您持续集成的工具, 同时, 我们注意到, 现在市场上还有很多 现成的持续集成的工具, 到底该选择哪一种, 需要您自己做出判断。下面这篇文章对现有的持续集成工具 进行了很好的横向比较, 也许这篇文章对您的抉择有所帮助: Continuous Integration Server Feature Matrix. [<http://damagecontrol.codehaus.org/Continuous+Integration+Server+Feature+Matrix>]

您可以通过 Luntbuild 示例 [<http://www.pmease.com/luntbuild/luntbuild-demo.html>] 来了解 Luntbuild的功能。也可以通过阅读FAQ [[../faq/index.html](http://www.pmease.com/luntbuild/faq/index.html)] 进一步学习Luntbuild。

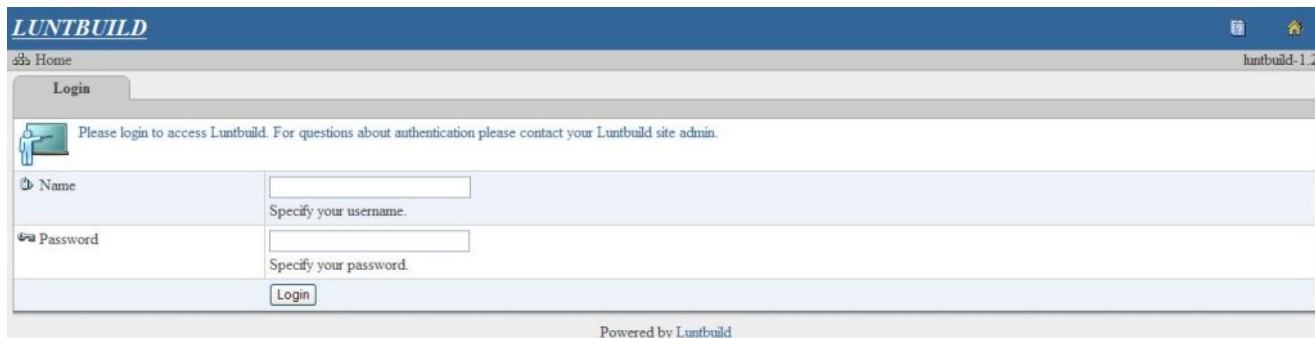
在Luntbuild里面最基本的操作单位是一次构建(build)。构建通过构建计划(Schedule)触发, 也可以通过手工启动。在Luntbuild里面一次构建会经过以下几个步骤:

1. 从版本控制系统(VCS)中获取源码。
2. 对当前源码打上标签, 标签值就是当前构建的版本号。
3. 在源码目录下运行相关的Ant/Maven/Command构建脚本(build script)。
4. 在源码目录下运行相关的Ant/Maven/Command构建后处理脚本(post build script)。
5. 发布构建的日志以及其他构建制品(build artifacts)。

构建的配置, 监视, 以及访问构建制品这些操作全部都通过web这种直观的方式来完成。开发和测试团队集中式的获取这些构建信息。

第 2 章 登录到Luntbuild



激动的时间来到了，我们现在要做第一次登录Luntbuild的表演。Luntbuild的登录页面如下：



登录页面要求您输入Name(用户名) 和 Password(口令)。第一次登录请用默认的管理员帐号 luntbuild/luntbuild(分别作为用户名和口令)，如果您修改了安全配置，那么请使用您在 applicationContext.xml文件里面指定的口令(详情请参考

Luntbuild安全

)。登录以后您可以创建新的用户。

登录页面右上角有两个图标。这两个图标在所有的Luntbuild页面里都有，其中  链接到Luntbuild 用户指南。而  链接到Luntbuild官方web站点。

输入用户名和口令之后，点击login(登录)按钮(或者直接按回车键)即可登录到Luntbuild系统。

第 3 章 Luntbuild主页

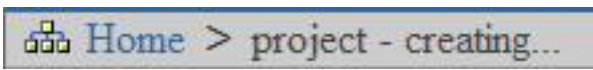
成功登录后将会进入Luntbuild主页：




在主页上有五栏，分别是：

1. Builds
(构建清单) - 概括显示当前的Luntbuild构建
2. Projects
(项目列表) - 显示所有的Luntbuild项目
3. Users
(用户列表) - 显示所有的Luntbuild用户
4. Properties
(系统属性) - 显示Luntbuild的一些系统属性
5. Administration
(系统管理) - 显示Luntbuild管理任务，例如导入/导出(import/export)

点击各栏就可以进入相应的页面。

每个页面的顶端是导航区 ，导航区将帮助您在Luntbuild的不同页面间进行快速切换。例如：如果您正在创建一个新的项目(通常是点击Project(项目)页上的New(新建)图标)，您可以通过点击导航区的Home(主页)快速返回主页面。

如果您在运行Luntbuild的时候遇到什么问题了，点击每一页右上角的system log(系统日志)，然后就会进入Luntbuild的系统日志页面。关于调试构建过程中遇到的问题，详情请参考调试构建中的问题

每页的右上角还包含了刷新按钮 ，该按钮可以设置页面自动刷新功能为启动或者关闭状态。如果您正在跟踪运行中构建的状态，那么设置为自动刷新状态是个不错的选择。

如果要退出Luntbuild，只需要点击右上角的logout(退出)链接。

第 4 章 设置Luntbuild的系统属性

在Properties(系统属性)页里面列出的属性将会影响所有的Luntbuild项目。Luntbuild的系统属性详细解释如下：

Url to access luntbuild servlet(访问luntbuild的Servlet的url)

在此指定访问Luntbuild的Servlet的url，在邮件通知里面会用到servlet url，因此设置正确的servlet url是很重要的。通常这个值的格式为http://<server>:<port>/luntbuild/app.do，其中<server> 是您的构建服务器的名字或者ip地址，而<port>是访问Luntbuild的端口号。如果这个属性没有设置，那么Luntbuild将会使用默认值http://<server_ip>:8080/luntbuild/app.do，其中<server_ip>是构建服务器的实际ip地址。

Work directory(顶层工作目录)

您可以根据需要指定Luntbuild的顶层工作目录，通常情况下，Lunbuild里面配置的各个项目会在该目录下建立子目录(以项目名称命名)作为本项目自己的工作目录，用来从版本控制系统中检出源码。如果没有指定该顶层工作目录，Luntbuild将使用Lunbuild安装目录下的work子目录作为顶层工作目录。

Publish directory(发布目录)

您可以根据需要指定Luntbuild的顶层发布目录。每个产生的构建都会在该目录下建立子目录作为自己的发布目录，用来存放该构建产生的制品，如日志和其他发布文件。该子目录格式为<project-name>/<schedule-name>/<build-version>。如果没有指定该顶层发布目录，Luntbuild将使用Luntbuild安装目录下的publish子目录作为顶成发布目录。

Page refresh interval(页面刷新时间间隔)

您可以以秒为单位指定页面刷新的时间间隔。如果没有设置该属性，那么默认值为15秒。

SMTP host(SMTP主机)

您可以根据需要指定SMTP邮件服务器，Luntbuild将通过它发送邮件通知。如果没有设置该属性，Luntbuild将使用本地机作为默认值。

SMTP user(SMTP用户)

该属性为可选属性，如果SMTP主机需要安全认证，那么您应该设置该属性来指定认证的用户名。

SMTP password(SMTP口令)

该属性为可选属性，该口令为上面的SMTP用户对应的口令。

Luntbuild Jabber account(Luntbuild Jabber帐号) - 如果您想通过Jabber通知用户，那么您在此需要设置Jabber有关的属性。

Luntbuild需要Jabber帐号用于发送Jabber消息。

注意

目前还不支持通过代理方式访问Jabber

Jabber server(Jabber服务器)

您可以根据需要指定Jabber服务器，用于发布Jabber消息。如果该属性没有设置，那么Luntbuild将使用本地机作为默认值。

Jabber server port(Jabber服务器端口号)

连接Jabber服务器的端口号，默认值5222。

Jabber user (Jabber用户)

在此设置Jabber帐号名，Luntbuild使用Jabber帐号来登录Jabber服务器并发送消息。

Jabber password (Jabber口令)

Jabber帐号对应的口令。

Luntbuild MSN account (Luntbuild MSN 帐号) - 如果您想通过MSN通知用户，在此设置有关的属性。

Luntbuild可以通过MSN帐号N来发送创建通知信息。 例如luntbuild@hotmail.com。

注意

目前还不支持通过代理方式访问MSN

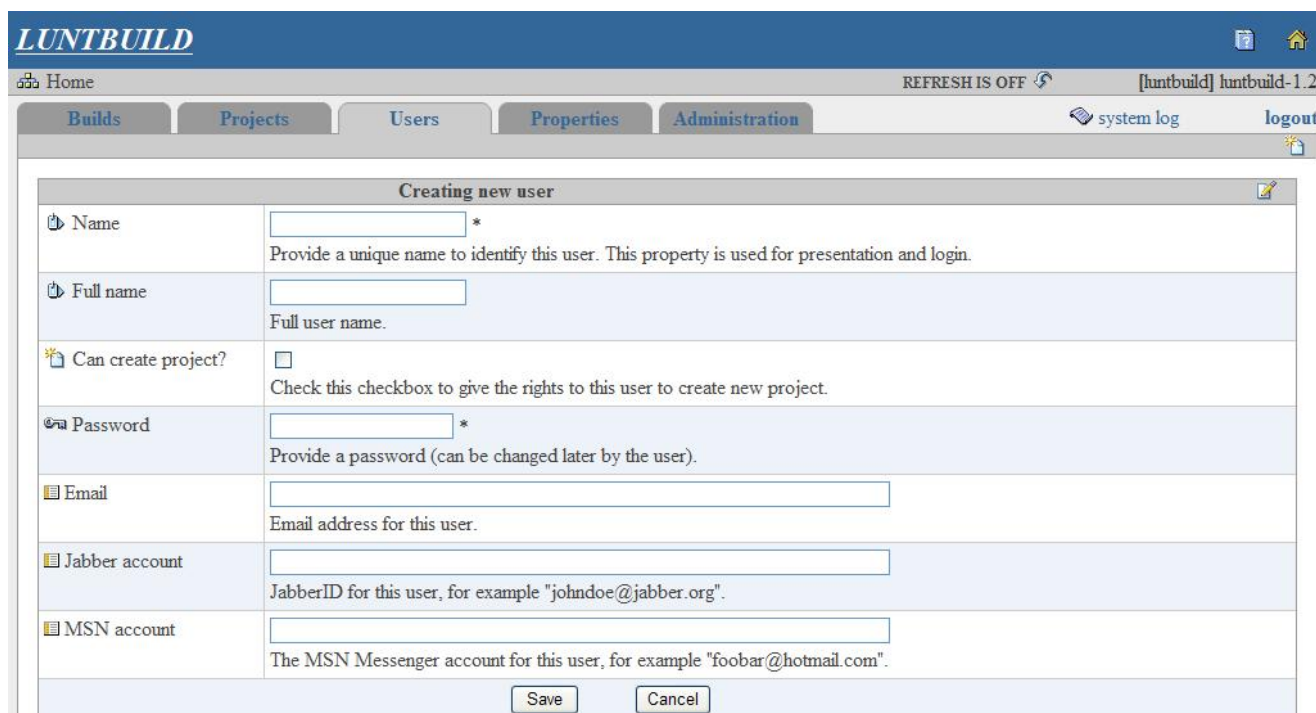
Luntbuild MSN password (Luntbuild MSN 口令)

MSN帐号的口令。

第 5 章 添加Luntbuild用户

在一个多人团队里使用Luntbuild，首先应该先创建好Luntbuild用户，并分配适当的权限，即使该团队只有您一个人。Luntbuild的用户可以收到构建状态的通知，同时被授权访问Luntbuild的不同部分。

创建Luntbuild用户的步骤：首先点击Users(用户)，然后点击new(新建)图标(该图标在该页的右上角)，然后进入以下页面：



The screenshot shows the 'Creating new user' form in the Luntbuild web interface. The form is titled 'Creating new user' and contains several input fields and a checkbox. The fields are: Name (required), Full name, Can create project? (checkbox), Password (required), Email, Jabber account, and MSN account. Each field has a description. At the bottom are 'Save' and 'Cancel' buttons.

填写以下信息：

Name(用户名)

在此提供一个唯一的名字用来确定一个用户。这个属性的值用于显示和登录。

Full name(全名)

用户的全名

Can create project?(是否具有创建项目的权限)

勾上checkbox既给用户赋予创建新项目的权限。

Password(口令)

在此提供一个初始化的口令(用户以后可以修改)。

Jabber account(Jabber帐号)

在此指定用户的Jabber帐号，例如： johndoe@jabber.org。 关于Jabber的详情请参考 jabber.org [http://www.jabber.org/about/overview.shtml]。

Email(电子邮箱)

在此指定用户的电子邮箱。

MSN account(MSN帐号)

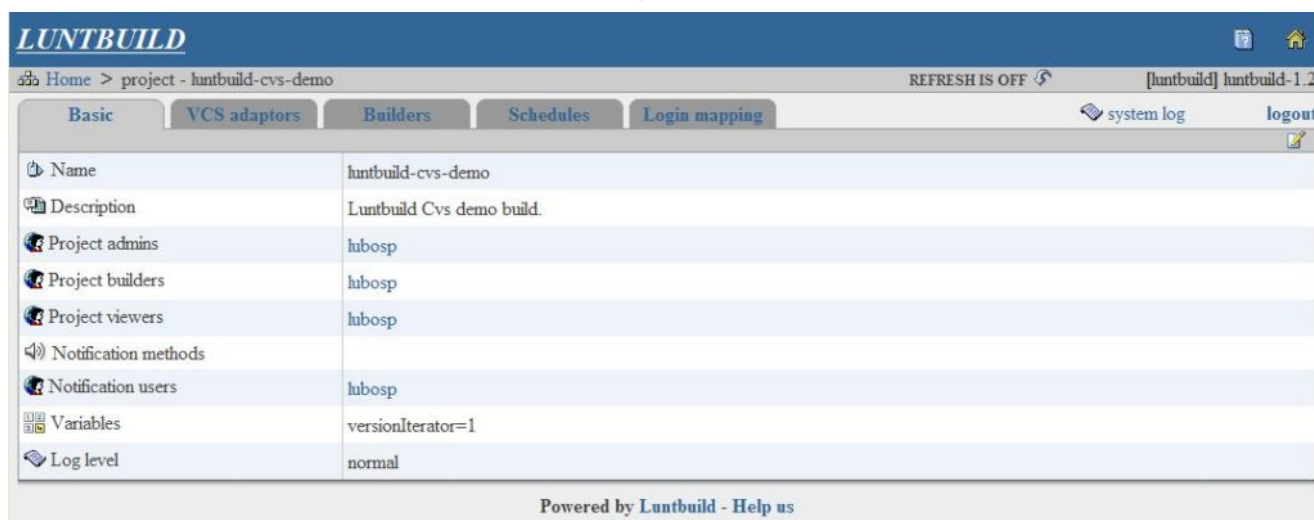
在此指定用户的MSN帐号，例如： foobar@hotmail.com

第 6 章 创建一个项目

点击Project(项目)栏。

项目页面显示了当前的Luntbuild实例下所有的项目配置信息。 一个项目就是一个可以构建的单元，在其中配置了和构建有关的信息，例如：版本控制系统，项目Builder，构建计划等等。

点击页面右上角的New Project(新建项目)图标 



Name(项目名)

提供一个唯一标识的项目名称。 请牢记对于每一个项目，都会在Luntbuild顶层工作目录和顶层发布目录下创建相应的子目录，而该子目录的名字就是项目名。

Description(描述)

项目的描述信息。

Project admins(项目管理员)

在此选择角色为'project admin'的用户

Project builders(项目构建者)

在此选择角色为'project builders'的用户

Project viewers(项目参与人员)

在此选择角色为'project viewers'的用户

Notification methods(通知方式)

在构建后以何种方式通知相关人员

Notification users(被通知的用户)

在项目构建完成后接收通知的用户

Variables(变量)

定义项目有关的变量，每行定义一个变量，例如

a=1

b=2

可以在其他的OGNL表达式中引用或者指定这里定义的变量，例如， 在设置构造构建计划的“next build version”属性的时候可以直接引用这里定义的变量。 数字型的变量可以递增或递减，例如， 如果您有两个构建计划分别为“nightly”和 “release”， 并希望发生在这两个计划的构建共享同一个递增的版本号，那么您可以定义下面这样的变量：

```
versionPart=foo-1.0.0
```

```
iterationPart=1
```

然后设置这两个构建计划的“next build version”属性。

```
${project.var["versionPart"]} (${project.var["iterationPart"].increaseAsInt())}
```

通过这种方式，这两个构建计划的构建版本号将会包括两个部分：第一部分为“versionPart”的值，第二部分为“iterationPart” 的值，而“iterationPart”会随着每次构建而递增。最终构建的版本号就是下面这个样子：

```
foo-1.0.0 (build 1)
```

```
foo-1.0.0 (build 2)
```

```
foo-1.0.0 (build 3)
```

```
...
```

您还可以定义许多其他类型的版本策略，详情参考构建计划的

next build version

属性。

Log level(日志级别)

定义项目的日志级别，将会影响到构建日志的详细程度。

第 7 章 为项目指定版本控制系统

首先选择“VCS Adaptors”控制面板标签

点击该面板右上角“New VCS Adaptor”图标。

选择版本控制系统

7.1. 设置AccuRev连接信息

以下地址可以下载 AccuRev: <http://www.accurev.com/download/index.htm> [<http://www.accurev.com/download/index.htm>]. 下面是该适配器的属性列表:

AccuRev port (AccuRev端口号)

AccuRev 端口号的格式为<servername>:<port>, 其中<servername> 和 <port> 应该用实际的 AccuRev的服务器名和端口号代替。该属性是可选的, 该属性的值会覆盖acclient.cnf值。

注意

AccuRev的默认配置信息定义在acclient.cnf 和wspaces 文件里面. 详情请参考AccuRev 用户手册。

Path for accurev executable (accurev的执行路径)

在该目录下保存accurev的可执行文件 如果系统目录下没有包含accurev的可执行文件, 那么应该放在 在该目录下。

Quiet period (等待周期)

从VCS check out代码来进行创建之前的等待周期(也就是多长时间没有用户check in), 以秒为单位。这是为了避免在其他人正在做check in的过程中check out代码。该属性为可选, 如果该属性没有设置, 那么在check out代码出来创建之前不会有等待周期。

7.2. 设置Clearcase基本连接信息

首先在用于创建的机器上必须安装了Clearcase客户端。同样您还应该保证启动您的应用服务器或者servlet容器的帐号能够访问Clearcase服务器, 而且能做视图快照。下面是该连接的属性列表。

Clearcase view stgloc name (Clearcase view stgloc名称)

Clearcase 服务器端视图存储的位置, 该属性用于创建当前项目的Clearcase视图的stgloc选项值。该属性和“Explicit path for view storage”只指定一个即可

Explicit path for view storage (明确的视图存储路径)

只有“Clearcase view stgloc name”属性没有指定的时候该属性才需要。如果指定了, 那么创建当前项目的Clearcase视图的时候, 该值将作为-vws选项值代替-stgloc选项

注意

在Windows平台上该值应该是可写的UNC路径。

Config spec(配置说明)

Config spec 用于创建Clearcase快照视图

Modification detection config(修改检测配置)

如果在上述的Config spec下指定要获取某些分支下的最新版本，这个时候该属性有效。Luntbuild用 该属性来决定自上一次创建后资料库是否发生了改变。该属性包括了多种条目，每个条目的格式为“<path>[:<branch>]”其中<path>是VOB下的路径，该值对于上述的config spec是可见的。对于该目录下的任何子目录，Luntbuild会检查clearcase服务器端内容是否发生了变化。如果<branch>指定了，那么Luntbuild将会检查指定的分支里的内容是否发生了变化。多个条目可以通过“;”符号或者多行来区分。

Extra options when creating snapshot view(创建快照视图的扩展选项)

当您使用Luntbuild为当前项目创建有关的Clearcase快照视图的时候，可以执行mkview sub命令的命令行选项。可以指定的命令行选项包括-tmode, -ptime, and -cachesize, 例如您可以指定“-tmode insert_cr”, 这样就使用 Windows的文本行终结符。

Path for cleartool executable(cleartool执行的路径)

cleartool可执行文件所在的路径。如果在系统的路径下找不到cleartool的可执行文件，就会在这里设置的目录下找。

Quiet period(等待周期)

在Luntbuild准备从VCS检出代码前，需要确保最近一次检入操作距离现在的最短时间(单位为秒)，在这段时间不能有新的检入操作。这样是为了避免有其他用户正在做检入的过程中Luntbuild做检出的操作，从而导致检出代码的不一致性。该属性为可选属性，如果为空，Luntbuild就会立刻检出代码，并执行构建，而不会检测当前VCS中是否有正在进行的检入操作。

7.3. Setting Cvs connection information. (设置Cvs连接信息)

如果您使用的是Windows平台，首先在您的平台上安装相应的CVS客户端，可以从下面的网站下载相应的安装包 <http://www.cvshome.org> 或者 <http://www.cvsnt.org>

注意

请保持构建服务器和CVS服务器在时间上的同步，这样构建服务器才能更精确的检测到cvs服务器上代码库的变化。还有就是记录在cvs变更日志里面的时间格式为UTC格式，而不是本地的时间格式。

下面是cvs连接的一些属性

Cvs root(CVS root)

例如:pserver:administrator@localhost:d:/cvs_repository。如果您的连接方式是ssh，请设置:ext:协议，并且要设置好ssh相应的环境，这些设置都在都在Luntbuild外进行。详情参考cvs用户指南。

Cvs password(Cvs口令)

使用pserver协议来连接的CVS口令

Is cygwin cvs?(是否使用的cygwin的cvs)

该属性指要使用的cvs是否是cygwin下的cvs? 可选的值为“yes”或“no”，默认值为“no”。

Disable “-S” option for log command? (禁止log命令的“-S”选项)

该选项用来设置log命令的“-S”选项是否禁止？ 可选的值为“yes”或“no”，默认值为“no”。 在log命令中使用-S选项可以加速对cvs代码库中发生变化的检测， 不过早期的cvs并不支持该选项。如果是早期的cvs版本，请输入“yes”来禁止该选项。

Disable history command?(禁止history命令)

该属性用来设置在检测CVS代码库的变化的时候是否禁止history命令。 可选的值为“yes”或“no”，默认值为“no”。 和log命令一起使用history命令可以加速对cvs代码库中发生变化的检测， 但是，有些cvs代码库并没有保存commit的历史信息，这种情况下，应该使用“yes”来禁止该命令。

Path for cvs executable(CVS执行的路径)

cvs可执行文件所在的路径。如果在系统路径下找不到cvs的可执行文件，就会在 这里设置的目录下找。

Quiet period(等待周期)

在Luntbuild准备从VCS检出代码前，需要确保最近一次检入操作距离现在的最短时间(单位为秒)，在这段时间不能有新的检入操作。 这样是为了避免有其他用户正在做检入的过程中Luntbuild做检出的操作，从而导致检出代码的不一致性。 该属性为可选属性，如果为空，Luntbuild就会立刻检出代码，并执行构建，而不会检测当前VCS中是否有正在进行的检入操作。

7.4. Setting File system connection information. (设置文件系统的连接信息)

Source directory(源目录)

该属性为可选属性。如果指定了，那么会以文件的修改时间为检测对象来检测源目录内容的变化。所有检测到的已修改的文件将会复制到项目工作目录下，并在此基础上进行构建。

Quiet period(等待周期)

在Luntbuild准备从VCS检出代码前，需要确保最近一次检入操作距离现在的最短时间(单位为秒)，在这段时间不能有新的检入操作。 这样是为了避免有其他用户正在做检入的过程中Luntbuild做检出的操作，从而导致检出代码的不一致性。 该属性为可选属性，如果为空，Luntbuild就会立刻检出代码，并执行构建，而不会检测当前VCS中是否有正在进行的检入操作。

7.5. Setting Perforce connection information. (设置Perforce连接信息)

首先应该在构建的机器上安装Perforce客户端。关于许可证信息请联系<http://www.perforce.com>。下面是该连接的属性列表：

Perforce port(Perforce端口)

Perforce端口格式<port>，或者 <servername>:<port>，其中<servername> 和 <port> 将被实际的Perforce服务器名和端口代替。

User name(用户名)

在此指定访问Perforce服务器的用户名。该用户应该有创建和修改客户端描述符(client specifications)的权限， 还应该检出代码以及给代码打上标签的权限。

Password(口令)

Perforce用户对应的口令。如果Perforce没有使用基于口令的安全机制，那么这里口令可以为空。

Line end(行结束符)

设置客户文本文件的行结束符。目前可设置的值如下：

```
local: use mode native to the client
unix: UNIX style
mac: Macintosh style
win: Windows style
share: writes UNIX style but reads UNIX, Mac or Windows style
```

该属性为可选属性。如果没有指定，那么默认值为“local”

Path for p4 executable(p4执行的路径)

P4可执行文件所在的路径。如果在系统的路径下找不到cvs的可执行文件，就会在 这里设置的目录下找。

Quiet period(等待周期)

在Luntbuild准备从VCS检出代码前，需要确保最近一次检入操作距离现在的最短时间(单位为秒)，在这段时间不能有新的检入操作。 这样是为了避免有其他用户正在做检入的过程中Luntbuild做检出的操作，从而导致检出代码的不一致性。 该属性为可选属性，如果为空，Luntbuild就会立刻检出代码，并执行构建，而不会检测当前VCS中是否有正在进行的检入操作。

7.6. Setting Subversion connection information. (设置 Subversion的连接信息)

首先请在您的构建服务器上安装好 Subversion 的客户端。 subversion 的下载地址为 <http://subversion.tigris.org> [<http://subversion.tigris.org/>].

注意

请保持构建服务器和Subversion服务器在时间上的同步，这样构建服务器才能更精确的检测到 Subversion服务器上代码库的变化。

下面是Subversion连接的属性列表：

Repository url base(代码库的“根”url)

Subversion 的 “ 根 ” url ， 例如 ： “svn://buildmachine.foobar.com/” 或者 “file:///c:/svn_repository” 或 “svn://buildmachine.foobar.com/myproject/othersubdirectory”，等等。 其他的例如标签的目录，分支的目录，模块等等都是相对于该“根”url的。

注意

如果使用https的方式访问subversion服务器，那么您需要配置Luntbuild的构建机器， 让它事先永久接受subversion服务器颁发的安全证书(具体可以参见subversion的用户说明)。

Directory for trunk(trunk目录)

指定在上述根url下用于存放主分支代码(trunk)的目录。该目录是根url的相对目录。 如果在根url下没有定义任何trunk目录，那么该属性的值为空值。

Directory for branches(分支目录)

指定在上述根url下用于存放分支代码的目录。该目录是根url的相对目录。 如果不设置该属性，那么Luntbuild将采用默认值，即“branches”。

Directory for tags(标签目录)

指定在上述根url下用于存放标签的目录。该目录是根url的相对目录。 如果不设置该属性，那么Luntbuild将采用默认值，即“tags”。

Username(用户名)

登录Subversion的用户名。

Password(口令)

登录Subversion的用户的口令。

Path for svn executable(svn执行的目录)

svn可执行文件所在的路径。如果在系统的路径下找不到svn的可执行文件，就会在 这里设置的目录下找。

Quiet period(等待周期)

在Luntbuild准备从VCS检出代码前，需要确保最近一次检入操作距离现在的最短时间(单位为秒)，在这段时间不能有新的检入操作。 这样是为了避免有其他用户正在做检入的过程中Luntbuild做检出的操作，从而导致检出代码的不一致性。 该属性为可选属性，如果为空，Luntbuild就会立刻检出代码，并执行构建，而不会检测当前VCS中是否有正在进行的检入操作。

7.7. Setting Clearcase UCM connection information. (设置 Clearcase UCM的连接信息)

首先需要在构建服务器上安装Clearcase客户端。 同时还要确保运行应用服务器和servlet容器的帐户能够访问Clearcase服务器，并且能够创建快照视图。 下面是该连接的一些属性：

Clearcase view stgloc name(Clearcase视图存储位置的名字)

在此指定Clearcase视图存储位置的名字，其值将作为创建Clearcase视图的stgloc选项的值。

Project VOB tag(项目VOB标签)

在此指定项目vob的标签，例如\pvob1

Explicit path for view storage(视图存储位置的路径)

如果“Clearcase view stgloc name” 属性的值为空，那么本属性的值就必须设置。 如果指定了本属性的值，其值作为创建Clearcase视图的-vws选项的值，并且会代替原来使用的-stgloc选项。

注意

This value should be a writable UNC path on Windows platform.

UCM stream name(UCM流)

在此指定UCM流的名字。

What to build(构建对象)

指定在UCM流内部要构建的基线。如果要指定多个基线，那么中间用空格分开。 下面的值具有特殊的意义：

<latest>: 基于每个组件最新的代码进行构建。
<latest baselines>: 基于每个组件最新的基线进行构建。
<recommended baselines>: 基于推荐的基线进行构建。
<foundation baselines>: 基于所有的最基础的基线进行构建。

Modification detection config(检测变化的配置)

当“`What to build`”属性的值为“`latest`”时，本属性的值有效。Luntbuild用该属性来决定自上一次创建后资料库是否发生了改变。该属性包括了多种条目，每个条目的格式为“`<path>[:<branch>]`”其中`<path>`是VOB下的路径，该值对于上述的`config spec`是可见的。对于该目录下的任何子目录，Luntbuild会检查clearcase服务器端内容是否发生了变化。如果`<branch>`指定了，那么Luntbuild将会检查指定的分支里的内容是否发生了变化。多个条目可以通过“;”符号或者多行来区分。

Extra options when creating snapshot view(创建快照视图的扩展选项)

在通过Luntbuild来创建当前项目有关的clearcase的快照视图的时候，您可以有选择性的指定`mkview sub`命令的扩展选项。目前可以指定选项限定为`-tmode`，`-ptime`，和`-cachesize`。例如，您可以指定“`-tmode insert_cr`”，这样可以使用Windows的行结束文本模式。

Path for cleartool executable(clearcase工具执行的路径)

在此指定clearcase工具可执行文件所在路径，如果这些可执行文件不在系统路径下，就应该在此指定其所在的目录。

Quiet period(等待周期)

在Luntbuild准备从VCS检出代码前，需要确保最近一次检入操作距离现在的最短时间(单位为秒)，在这段时间不能有新的检入操作。这样是为了避免有其他用户正在做检入的过程中Luntbuild做检出的操作，从而导致检出代码的不一致性。该属性为可选属性，如果为空，Luntbuild就会立刻检出代码，并执行构建，而不会检测当前VCS中是否有正在进行的检入操作。

7.8. Setting Visual Sourcesafe connection information. (设置VSS连接信息)

首先在构建服务器上安装VSS软件，下载地址<http://download.microsoft.com>。下面是VSS连接的属性：

注意

为了VSS的`history`命令的准确性，所有开发机，构建服务器的时间设置要保持同步。

Sourcesafe path(Sourcesafe路径)

该目录是`srcsafe.ini`文件所在目录。例如：`\\machine1\directory1` 这个路径名要么用主机名要么用主机的ip地址，而且要确保运行应用服务器或者servlet容器的帐号事先已经登录到这台机器上，否则在构建时Luntbuild会汇报一个“No VSS database found”的错误。

Username(用户名)

登录该VSS的用户名。

Password(口令)

上面指定的用户的口令。

Datetime format(日期时间格式)

指定VSS的history命令的日期时间格式。该属性为可选属性，如果该属性的值为空，那么Luntbuild将使用默认值“M/dd/yy;h:mm:ssa”。默认值只适合使用US的英文操作系统。对于其他的使用英语的国家，例如英国，澳大利亚，加拿大，VSS使用的时间格式为(假设VSS的本地语言已经设置为相应的值)：

```
'd/M/yy;H:mm'
```

如果Luntbuild运行的操作系统为非英文的操作系统，可以通过以下方法决定使用的日期时间格式：

打开安装在您的构建服务器上的VSS，选择一个现有的VSS数据库，然后查看里面的项目和文件。在文件列表框里面有几栏，其中一栏是“Date-Time”。可以使用这个位置相应的“datetime format”(日期时间格式)。例如，如果其中一个的值为“04-07-18 20:19”，那么相应的“datetime format”(日期时间格式)为“yy-MM-dd;HH:mm”。注意在日期和时间之间的semicolon(分号)一定要指定。在这里鼓励大家将该属性设置为“yy-MM-dd;HH:mm:ss”，这样可以提高精确度。例如，如果VSS里面的日期时间为“7/18/04 8:19p”，那么相应的“datetime format”(日期时间格式)为“M/dd/yy;h:mm:ssa”。这个时候如果设置为“M/dd/yy;h:mm:ssa”，那么可以增加准确性。下面是从JDK文档中节选的部分格式化字符的信息：

表 7.1. Date/Time format characters(日期时间格式化字符)

字符	解释	例子
y	年	1996 ; 96
M	(一年中的)月	July ; Jul ; 07
d	(一个月中的)日	10
a	Am/pm(上下午)标志	p
H	(一天中的)小时，范围(0-23)	0
h	(上午或下午的)小时，范围(1-12)	12
m	(小时中的)分钟	30
s	(分钟中的)秒	55

详情参考 <http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html>
[<http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html>]

Path for ss.exe(ss.exe文件所在路径)

在此指定ss.exe文件所在的路径，如果ss.exe文件不在系统路径下面，Luntbuild就会使用这个属性的值作为寻找该ss.exe文件的目录。

Quiet period(等待周期)

在Luntbuild准备从VCS检出代码前，需要确保最近一次检入操作距离现在的最短时间(单位为秒)，在这段时间不能有新的检入操作。这样是为了避免有其他用户正在做检入的过程中Luntbuild做检

出的操作，从而导致检出代码的不一致性。该属性为可选属性，如果为空，Luntbuild就会立刻检出代码，并执行构建，而不会检测当前VCS中是否有正在进行的检入操作。

7.9. Setting StarTeam connection information. (设置StarTeam连接参数)

首先在Windows平台上完全安装StarTeam SDK的运行时套件(把运行时dll安装在Windows系统目录下)。通常这是StarTeam客户端安装程序的一部分。许可信息请参考<http://www.borland.com>。下面是该连接的一些属性：

Project location(项目位置)

StarTeam项目的位置，格式<servername>:<portnum>/<projectname>，其中<servername>是StarTeam服务器名字，<portnum>是服务器使用的端口号，默认值49201。<projectname>是运行在StarTeam服务器上的StarTeam的项目名。

User(用户)

登录StarTeam服务器的用户名。

Password(口令)

登录StarTeam服务器的口令。

Convert EOL?

该属性可选的值如下：

all: 所有ASCII码文件将其行结束符调整为和检出的客户机的行结束符一致。

no: 检出的文件的行结束符以服务器为准。

该属性为可选属性。如果没有指定，默认值为yes。

Quiet period(等待周期)

在Luntbuild准备从VCS检出代码前，需要确保最近一次检入操作距离现在的最短时间(单位为秒)，在这段时间不能有新的检入操作。这样是为了避免有其他用户正在做检入的过程中Luntbuild做检出的操作，从而导致检出代码的不一致性。该属性为可选属性，如果为空，Luntbuild就会立刻检出代码，并执行构建，而不会检测当前VCS中是否有正在进行的检入操作。

7.10. Using multiple Version Control Adaptors. (同时使用多个VCS连接)

对于一个项目可以定义多个VCS连接。当该项目的构建开始运行的时候，所有有关的VCS代码库的相关代码检出到项目的工作目录。例如，某个项目，客户端模块是在Cvs代码库里面，服务器端的模块是在VSS代码库里面。这种多连接的情况还适合于那种使用同一种VCS，当时不同的模块存放在不同的代码库里面的情况。例如，某个项目，客户端模块在某一个Cvs代码库中，而服务器端模块在另一个Cvs代码库中。

第 8 章 新建VCS模块

点击某个VCS定义页面里面的New Module(新建模块)图标。如果定义了多个VCS模块，那么Luntbuild会按序依次检出这些模块。如果后面模块目标路径和前一个模块有部分重叠，那么这部分重叠的路径将会被后一个模块覆盖。例如：假设您定义了一个模块，其目标路径为“/foo/bar”，然后又定义了第二个模块，其目标路径为“/foo”，那么从第二个模块检出的内容将完全覆盖第一个模块。反过来，如果第一个模块的目标路径为“/foo”，第二个模块的目标路径为“/foo/bar”，那么第一个模块检出的“/foo/bar”下的内容会被第二个模块覆盖，而在检出代码中的其他路径比如/foo/another_bar，将不会被覆盖。

8.1. 设置AccuRev模块信息

Label(标签)

标签在AccuRev里面是用于同步的事务编号。在此指定您希望基于哪个事务编号进行构建。

Depot(存储位置)

用于检出代码 Accurev储存位置。

Backing stream(回溯流)

构建模块的回溯流的名字。The backing stream should be able to have streams created from it by the build user.

Build stream(构建流)

指定同回溯流关联的构建流名称（如果该构建流 存在，它将被自动创建）。基于这个流，一个相关的引用树将会被创建(树的名称为流的名称加上“_reference”后缀)。

8.2. 设置cvs模块信息

Source path(源路径)

指定从CVS代码库中获取代码的路径，例如：testcvs/src

Branch(分支)

指定前面的源路径的分支。该选项是可选的。如果没有设置相应的值，那么将把主分支作为默认值。

Label(标签)

指定前面的源路径的标签。该选项也是可选的。如果指定了该值，那么该值将优先于分支的值。如果没有设置相应的值，就会检出上述指定分支里面的最新代码。

“Source path”(源路径)代表了CVS代码库中模块的意思，例如“/testcvs”，“/testcvs/web”，或者“testcvs”。但是不能用“/”或者“\”这两个字符来定义源路径。“Branch”(分支)就是CVS分支(branch)，而“Label”(标签)就是CVS的标签(tag)。对于某个模块，这二个属性(branch和label)中只有一个的值有效。如果都不为空，那么标签将优于分支。如果都为空，Luntbuild则从主分支获取最新的代码。

8.3. 设置Perforce模块信息。

Depot path(存储位置路径)

指定Perforce存储位置的路径, 例如`“//depot/testperforce/...”`。

Label(标签)

指定前面的存储位置路径上的标签。该属性是可选的。 如果为空, 那么将会检出该存储路径上最新的版本。

Client path(客户端路径)

指定客户端的路径, 例如`“//myclient/testperforce/...”`。

注意

某些文件和目录要排除在外, 那么对于需要排除在外的每一个文件和目录都需要创建一个独立的模块, 而且 在Depot路径的属性前需要加上减号((-))作为前缀:

```
Depot path: -//depot.side  
Client path: //client.side
```

Perforce的模块定义把代码库的Depot路径(“Depot path”)映射到客户端路径(“Client path”)。Luntbuild也支持Perforce的标签属性。 Depot路径(“Depot path”)表示Perforce代码库中的路径, 例如 `“//depot/testperforce/...”` 客户端路径(“Client path”)表示客户端的路径(也就是Depot路径下的内容检出之后保存在客户端的什么目录下), 例如`“//myclient/testperforce/...”` “Label”是Perforce标签, 如果您想获取指定的Depot路径下的内容的某个时刻的快照, 那么就需要使用标签。 如果标签为空, 那么获得的Depot路径下的内容为head版本。 客户端路径对应的目录不一定要存在。 如果不存在那么Luntbuild会自动创建相应的目录。 在Perforce的连接信息中指定的用户应该有足够的权限来创建或者修改Perforce的参数。

8.4. 设置Subversion模块信息

Source path(源路径)

即 Subversion 代码库中的路径, 例如`“testsvn”, “testsvn/web”, 或者 “/testsvn”`。 当 `“branch”`(分支) 或者 `“label”`(标签)属性指定的时候, 本路径会映射到svn代码库的另一个路径。

Branch(分支)

指定以上源路径的分支。该属性为可选项, 如果不设置, 那么将使用默认值trunk。

注意

在subversion里面没有分支的概念。在此指定的值会被Luntbuild映射到相应的源路径, 效果就像cvs的分支一样。

Label(标签)

指定以上源路径的标签。该属性为可选的。如果设置了该属性, 其优先级高于分支。如果不设置, 就采用指定分支的head version。

注意

同样，subversion也没有标签的概念。在此指定的值会被Luntbuild映射到相应的源路径，效果就像cvs的标签一样。

Destination path(目标路径)

该属性为可选的。目标路径相对于项目的工作目录，如果设置了相应的值，那么从subversion代码库中获取的代码将保存在该目录下。否则将会在项目的工作目录下创建一个以源路径命名的目录，从subversion获取的代码将保存在该目录下。

“Source path”(源路径)即Subversion代码库中的路径，例如“testsvn”，“testsvn/web”，或者“/testsvn”。Luntbuild在其他的属性值基础上将该路径映射到Svn代码库中的路径。我们定义了以下属性来描述这种映射方式：

Repository url base: svn://localhost

Directory for trunk: trunk

Directory for branches: branches

Directory for tags: tags

可以试验一下下面的模块设置和路径映射：

Source path: testsvn/web, branch: <empty>, label: <empty>, destination path: <empty>

Luntbuild从ulr“svn://localhost/trunk/testsvn/web”下检出代码，并保存在“<project work directory>/testsvn/web”下面。

Source path: testsvn/web, branch: simplified-chinese, label: <empty>, destination path: <empty>

Luntbuild从ulr “svn://localhost/branches/simplified-chinese/testsvn/web”下检出代码，并保存在“<project work directory>/testsvn/web”下面。

Source path: testsvn/web, branch: <empty>, label: v1_0, destination path: <empty>

Luntbuild从ulr “svn://localhost/tags/v1_0/testsvn/web”下检出代码，并保存在“<project work directory>/testsvn/web”下面。

Source path: testsvn/web, branch: simplified-chinese, label: v1_0, destination path: testsvn/web/simplified-chinese

Luntbuild从ulr “svn://localhost/tags/v1_0/testsvn/web”下检出代码，并保存在“<project work directory>/testsvn/web/simplified-chinese”下面。

注意

这里定义的分支无效，因为同时定义了分支和标签，那么标签优先。

当Luntbuild给检出到目录“<project work directory>/testsvn/web”下的代码打上标签，例如是“v1_0”，会执行下面的代码：“svn copy <project work directory>/testsvn/web svn://localhost/tags/v1_0/testsvn/web”

如果将“Directory for trunk”，“Branch”和“Label”三个属性都设置为空值，就可以避免上面的url映射。通过这种方式，只设置“Source path”和“Destination path”属性，您就可以控制从什么地方检出代码，并将检出的代码保存在什么地方。这种情况下，源路径只会把“repository url base”属性的值作为前缀。

8.5. 设置VSS模块信息

Source path(源路径)

在此指定在VSS代码库中的路径，例如“testvss”，或者“/testvss”。

注意

不用在路径前添加\$符号，如果指的是整个代码库，只需要输入“/”就可以了。

Label(标签)

在此指定上面设置的源路径上的标签，该属性为可选的，如果为空，就使用最新的版本。

Destination path(目标路径)

指定相对于项目工作目录的目标目录，从源目录下获取的代码就保存在这个目录下。该属性为可选的，如果该属性为空，Luntbuild会在项目工作目录下建一个和源目录的目录结构一致的目录，并将检出的代码保存在这个目录下。

源路径其实就是相对于VSS代码库根目录的项目路径，例如“testvss”，“/testvss”，或者“/testvss/web”等等。使用“/”或者“\”，都可以获取整个代码库的内容。标签指VSS标签。VSS通过创建一个新的共享的项目来实现分支功能。因此您需要配置不同的模块，这样才能从不同的分支获取代码。如果某个模块的标签为空，Luntbuild则从VSS获取该模块的最新代码。如果定义目标路径属性，那么从VSS获取的代码将会保存在相对于项目工作目录的目标目录下。否则会保存在相对于项目工作目录并且和源目录的目录结构一样的目录下。

8.6. 设置StarTeam模块信息

StarTeam view(StarTeam视图)

在此指定StarTeam视图。该属性为可选属性。如果为空，Luntbuild使用当前的StarTeam项目的根view。

Source path(源路径)

在此指定相对于StarTeam视图的路径。“/”代表根路径。

Label(标签)

指定以上StarTeam视图的标签。该属性为可选属性。如果为空，就以指定视图上的最新代码为准。


Destination path(目标路径)

指定相对于项目工作目录的目标目录，从源目录下获取的代码就保存在这个目录下。该属性为可选的，如果该属性为空，Luntbuild会在项目工作目录下建一个和源目录的目录结构一致的目录，并将检出的代码保存在这个目录下。

“StarTeam view”指StarTeam的视图，“Label”指StarTeam视图的标签。如果“StarTeam view”为空，Luntbuild就使用StarTeam的根视图。“Source path”是相对指定的StarTeam视图根目录的路径。如果定义了“Destination path”性，那么从StarTeam代码库获取的代码将会保存在相对于项目工作目录的目标目录下。否则会保存在相对于项目工作目录并且和源目录的目录结构一样的目录下。

第 9 章 创建项目的Builder

Builder根据项目特定的构建计划执行相应的创建活动。

新建Builder的步骤：首先点击“Builders”，然后点击该页右上角的“New”图标  然后就会进入Builders编辑页面。

LUNTBUILD

Home > project - luntbuild-cvs-demo REFRESH IS OFF [luntbuild] luntbuild-1.2

Basic VCS adaptors **Builders** Schedules Login mapping system log logout

Editing builder information (Fields marked with the * are required)

builder type	Ant builder Select the type of builder
Name	default * Provide a name to identify this builder, this name can be changed later.
Command to run Ant	d:\apache-ant-1.6.2\bin\ant.bat Specify the command to run Ant (normally path to ant.bat or ant shell script) here. For example: /path/to/ant -DbuildVersion="\${build.version}" -DartifactsDir="\${build.artifactsDir}". String enclosed by \${...} will be interpreted as OGNL expression, and it be evaluated before execution. For valid OGNL expressions in this context, please refer to the User's Guide. NOTE. A single argument that includes spaces should be quoted in order not to be interpreted as multiple arguments.
Build script path	luntbuild/build/build.xml * The path for the Ant build script. If this path is not an absolute path, it is assumed that it is relative to the schedule work directory. Refer to the User's Guide for details about how to write a new Ant build file or how to modify your existing Ant build script.
Build targets	clean package

首先选择适当的Builder类型。 目前可以选择的Builder包括：

Ant Builder

Command Builder

Maven Builder

对于指定的项目，根据不同任务的需要，可以创建所需的builders，创建的数目没有限制，这个根据需求而定。 然后针对项目的每个构建计划，从这里定义的一套builder里面挑选出相应的builder或者post-builder。

9.1. 配置Ant Builder

Name (名称)

提供一个用于标识该builder的名称，该名称以后可以修改。

Command to run Ant (运行Ant的命令)

指定运行Ant的命令(通常是ant.bat或者ant的shell文件及其所在的目录) 例如：/path/to/ant 包含在\${...}里面的字符串被当作是OGNL表达式，而且在构建前被替换成实际值。 每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [../javadoc/com/luntsys/luntbuild/builders/Builder.html]对象。

注意

包含空格的单个参数应该用引号括起来，这样才不会被解释为多个参数。

注意

对于Ant命令行选项，下面的选项不需要指定“-buildfile”和“-logfile”，因为这两个选项由Luntbuild内部提供。其他选项允许使用。

您可以修改Ant命令以添加命令行选项和属性，例如-Ddebug=_debug.

Build script path(构建脚本所在路径)

Ant的构建脚本文件(buildfile)所在的路径，如果该目录非绝对路径，Luntbuild认为该目录是相对项目工作目录的。

Build targets(构建目标)

指定需要构建的目标，使用空格来分隔不同的目标(包括空格符的目标名需要用引号括起来，这样是为了避免被当成是多个目标来处理)。如果没有指定目标，那么将使用ant构建文件的默认目标。同样地，您可以使用OGNL表达式(\${...})来传递变量，该变量作为构建的目标名。例如，您可以使用\${build.schedule.name}变量，该变量的作用是可以根据不同的构建计划使用不同的构建目标。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [../javadoc/com/luntsys/luntbuild/builders/Builder.html]对象。

Build properties(构建属性)

在此定义需要传递给ant构建脚本的属性，例如：

```
buildVersion=${build.version}
scheduleName=${build.schedule.name}
```

每一行只能设置一个变量。这里支持使用OGNL表达式格式，即\${...}。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [../javadoc/com/luntsys/luntbuild/builders/Builder.html]对象。

环境变量

设置运行builder的环境变量，例如：

```
MYAPP_HOME=${build.schedule.workingDir}
SCHEDULE_NAME=${build.schedule.name}
```

每一行只能设置一个变量。可以在变量里插入OGNL表达式(即\${...})。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [../javadoc/com/luntsys/luntbuild/builders/Builder.html]对象。

Build success condition(成功构建条件式)

构建成功条件式是一个用来判断当前项目是否成功构建的OGNL表达式 每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [../javadoc/com/luntsys/luntbuild/builders/Builder.html]对象。如果不设置该属性，那么会使用默认值——result==0 and logContainsLine(“BUILD SUCCESSFUL”)。当这个表达式的值为true的时候，就认为本次构建是成功的。下面的一些例子演示了OGNL表达式的格式。

result==0, 此处的“result”表示ant构建结果的返回值。

logContainsLine(“^ERROR.*”), 如果构建的日志里面的某一行匹配正则表达式“^ERROR.*”, 那么就认为该表达式为true。关于正则表达式的格式请参考<http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html>
[<http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html>]

以上的表达式可以使用'!'符号，该符号的作用是用来求相反的值。例如，`!logContainsLine("^ERROR.*")`，如果构建日志里没有包括匹配指定模式的行，那么该表达式的值为true。

以上的表达式可以通过"and"和"or"逻辑运算符组合成新的表达式。例如，表达式`result==0 and !logContainsLine("^ERROR.*")`，如果ANT执行构建的结果为0，且构建的日志不包括以"ERROR"字符串打头的行，那么该表达式的值为true。

9.2. 配置命令行Builder

Name(名称)

builder的标识名，设置后可以修改。

Build command(构建命令)

在此指定构建命令。例如：`/path/to/command.bat "${build.version}" "${build.artifactsDir}"`。`${...}`所包括的部分被当作是OGNL表达式来处理，在构建前会被实际值代替。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [[../javadoc/com/luntsys/luntbuild/builders/Builder.html](#)]对象。

注意

包含了空格的参数需要用引号括起来，否则会被当作是多个参数来处理。

Run command in directory(运行构建命令的目录)

用于运行构建命令的目录。如果该目录非绝对路径，Luntbuild认为该目录是相对项目工作目录的。

Environment variables(环境变量)

在构建前需要先设置环境变量，例如：

```
MYAPP_HOME=${build.schedule.workingDir}
```

```
SCHEDULE_NAME=${build.schedule.name}
```

每行只能指定一个环境变量。OGNL表达式可以嵌入在变量的值里面，OGNL表达式需要用`${...}`括起来。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [[../javadoc/com/luntsys/luntbuild/builders/Builder.html](#)]对象。

Build success condition(成功构建表达式)

构建成功条件式是一个用来判断当前项目是否成功构建的OGNL表达式。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [[../javadoc/com/luntsys/luntbuild/builders/Builder.html](#)]对象。如果不设置该属性，那么会使用默认值——`result==0 and logContainsLine("BUILD SUCCESSFUL")`。当这个表达式的值为true的时候，就认为本次构建是成功的。

9.3. 配置Maven Builder

Name(名字)

提供一个用于标识该builder的名称，该名称以后可以修改。

Command to run Maven(运行Maven的命令)

指定运行Maven的命令(一般来说是maven.bat，或者maven的shell脚本)。例如：/path/to/maven。
。 \${...} 中间的部分是OGNL表达式，在命令执行前会计算出该表达式的值。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder
[../javadoc/com/luntsys/luntbuild/builders/Builder.html]对象。

注意

如果要在Maven中引用Luntbuild提供的构建版本号，请参考下面的project.xml：

```
<project>
...
<!--Use value of variable "buildVersion" as current version, this variable is defined in Luntbuild's Maven
builder configuration page-->
<currentVersion>${buildVersion}</currentVersion>
...
</project>
```

注意

包含了空格的参数需要用引号括起来，否则会被当作是多个参数来处理。

Directory to run Maven in (运行Maven的目录)

指定要在哪个目录下运行Maven。如果该目录非绝对路径，Luntbuild认为该目录是相对项目工作目录的。

Goals to build(构建的目标)

指定要构建的目标。如果要指定多个目标，请用空格来分开多个目标(带空格的目标应该使用引号括起来，否则会被当作是多个目标来处理。)在目标名字里面可以使用\${...}，\${...}中间的部分会被当作是OGNL表达式来处理。例如，您可以使用\${build.schedule.name}，该表达式可以根据不同的目标使用不同的构建计划。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder
[../javadoc/com/luntsys/luntbuild/builders/Builder.html]对象。

Build properties(构建的属性)

定义传递给maven构建脚本的属性值。例如：

```
buildVersion=${build.version}
scheduleName=${build.schedule.name}
```

每一行只能设置一个变量。也可以在变量中使用OGNL表达式，OGNL表达式需要用\${...}括起来。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder
[../javadoc/com/luntsys/luntbuild/builders/Builder.html]对象。

Environment variables(环境变量)

在这里设置构建的时候要使用的环境变量。例如：

```
MYAPP_HOME=${build.schedule.workingDir}
```

```
SCHEDULE_NAME=${build.schedule.name}
```

每一行只能设置一个变量。也可以在变量中使用OGNL表达式，OGNL表达式需要用`${...}`括起来。每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [[../javadoc/com/luntsys/luntbuild/builders/Builder.html](#)]对象。

Build success condition(成功构建表达式)

构建成功条件式是一个用来判断当前项目是否成功构建的OGNL表达式 每个OGNL表达式都需要有一个对应的JAVA根对象来对表达式求值(通过反射机制)，这里的根对象就是当前的Builder [[../javadoc/com/luntsys/luntbuild/builders/Builder.html](#)]对象。 如果不设置该属性，那么会使用默认值——`result==0` 和 `logContainsLine("BUILD SUCCESSFUL")`。 当这个表达式的值为true的时候，就认为本次构建是成功的。

第 10 章 创建项目的构建计划

构建计划用于启动或触发构建活动，包括了后台自动触发和手工触发两种方式。

构建需要一个工作目录，该工作目录用来保存从VCS代码库中检出的代码。下面是Luntbuild构造工作目录的一些准则：

1. Luntbuild
顶层工作目录
作为所有的Luntbuild项目工作目录的根目录。
2. 每个构建计划都允许定义自己的工作目录。默认情况下，该目录位于Luntbuild顶层工作目录下的以项目名命名的子目录下。
3. 定义了源目录的VCS模块在检出之后，Luntbuild自动在构建计划的工作目录下建相应的子目录，通常情况下子目录结构和 VCS模块的源目录结构一致（除非在定义VCS模块时指定了目标目录）。

例如，如果Luntbuild的顶层工作目录为/luntbuild-install-dir/work，项目名为myproject，而构建计划子目录定义为myscheduleworkdir，VCS的某个模块的源路径为source，那么通常情况下source下的内容将会被检出到：
/luntbuild-install-dir/work/myproject/myscheduleworkdir/source

为什么构建计划的工作目录这么重要？原因如下：

构建计划的工作目录可以在同一个项目的多个构建计划间共享。这样这些构建计划的构建就会同一个工作目录，从而可以节省很多磁盘空间。Luntbuild保证如果多个构建计划共享同一个工作目录，那么这些构建计划不会同时运行。事实上，如果使用该共享工作目录的第一个构建启动了，那么其他的使用该共享目录的构建就会进入一个等待构建的队列，等待前一个构建完成之后才能启动下一个构建。

如果构建计划的工作目录没有和同一个项目的其他构建计划共享，那么项目对应的VCS模块的内容会被多次检出(到多个工作目录)，这样会消耗大量的磁盘空间，而且也会花更多的时间来执行检出VCS模块的操作。但是这样有个好处就是，(同一个项目里面)不同的构建计划使用不同的工作目录，这样不同的构建计划里的构建可以同时进行。

每个构建实例使用发布目录来输出构建的制品，例如构建日志，版本变更日志，要发布的文件等等。下面是Luntbuild构建发布目录的一些准则：

1. Luntbuild顶层
发布目录
是所有的Luntbuild构建发布目录的根目录。
2. 在顶层
发布目录
下是以项目名命名的子目录。
3. 在项目子目录下是以构建计划名字命名的子目录
4. 在构建计划子目录下则是以构建版本号为名字来命名子目录。该目录下包括了构建日志 build_log.txt，版本变更日志 revision_log.txt，还有两个子目录 artifacts和 junit_html_report。其中artifacts用于存储您的其他输出制品，junit_html_report用于存储

单元测试的结果。

例如：如果Luntbuild的顶层发布目录为 /luntbuild-install-dir/publish，项目名myproject，构建计划名为myschedule，当前构建的版本号为myapp-1.2.0，那么版本myapp-1.2.0的构建发布目录的绝对路径为 /luntbuild-install-dir/publish/myproject/myschedule/myapp-1.2.0。

点击Schedules(构建计划)标签，然后点击该页右上角的New Schedule(新建构建计划)图标，即可新建一个构建计划。

LUNTBUILD

Home > project - luntbuild-build REFRESH IS OFF [luntbuild] luntbuild-1.2

Basic VCS adaptors Builders Schedules Login mapping system log logout

Creating a new schedule (Fields marked with the * are required)

Name *
Provide a name for this schedule. This name will be used to identify this schedule, and cannot be changed later.

Description
Provide a description for this schedule.

Next build version *
Next build version for this schedule. The version increments as follows:
luntbuild-1.9 will be increased to luntbuild-1.10
luntbuild-1.5 (build 1000) will be increased to luntbuild-1.5 (build 1001)
You can also insert variables(enclosed by \${...}) to the version string to make it more flexible. For example, the version string can be defined as:
luntbuild-\${#currentDay=system.(year+"-"+month+"-"+dayOfOfMonth), #lastDay=project.var["day"].setValue(#currentDay), #dayIterator=project.var["dayIterator"].intValue, project.var["dayIterator"].setIntValue(#currentDay==#lastDay?#dayIterator+1:1), #currentDay}.\${project.var["dayIterator"]}
Then the actual version string for a build will include the build date and iterations for that date. Or you can specify the version string as:
luntbuild-1.0.\${project.var["versionIterator"].increaseAsInt()}
In this way, last digit of the version will take the increased value of a project variable named by "versionIterator".
For details, please refer to the User's Guide.

Work directory

Name(名字)

构建计划的名字，该名字用于标识构建计划。

Description(描述)

在此提供对该构建计划的一些描述信息。

Next build version(下一个构建版本)

在此指定下一个构建版本的字符串。Luntbuild会在构建的过程中自动增量式修改该字符串的版本号部分。

luntbuild-1.0 会增加到 luntbuild-1.1

luntbuild-1.2.9 增加到 luntbuild-1.2.10

luntbuild-1.5(1000) 增加到 luntbuild-1.5(1001)

通常情况下，每次构建下一个构建版本的最后一个数字都会增加。例如三次构建之后“luntbuild-1.2.0”会增加到“luntbuild-1.2.3”。但是，如果在该字符串中嵌入了OGNL表达式(以\${...}括起来)，那么最后一个数字就不会自动增加了。取而代之的是，Luntbuild会根据特定的构建计算出每一个嵌入的OGNL表达式的实际值。在执行计算的过程中，当前的Schedule[../javadoc/com/luntsys/luntbuild/db/Schedule.html]对象会作为该OGNL表达式的根(root)对

象。下面的例子展示了使用OGNL表达式获取不同的版本号的策略：

场景1：将当前的日期和迭代次数作为构建版本号的一部分 Scenario 1: Put current date and iteration of this date as the part of the build version
将每个构建计划的下一个版本号定义为：

```
foo-${#currentDay=system. (year+"-"+month+"-"+dayOfMonth), #lastDay=project. var["day"]. setValue(#currentDay, #lastDay), #dayIterator=project. var["dayIterator"]. intValue, project. var["dayIterator"]. \
setIntValue(#currentDay==#lastDay?#dayIterator+1:1), #currentDay}. ${project. var["dayIterator"]}
```

最终，某个构建的实际版本的字符串将包括构建的日期和当天迭代的次数。

场景2：测试和发布构建计划共享和共同增加相同的版本号字符串，而持续集成构建则使用另一个独立的版本号。

测试和发布构建计划的下一个构建版本都设置为：

```
foo-${project. var["majorVersionPart"]}. ${project. var["minorVersionPart"]. increaseAsInt ()}
```

对于持续集成构建计划，将下一个构建版本设置为：

```
foo-1
```

场景3：发布构建计划增加版本号的发布有关的部分，而每夜构建计划增加版本好的迭代有关的部分。当发布有关的部分改变的同时，迭代部分重置为1。

定义以下项目的变量：

```
fixPart=foo-1.1
```

```
releasePart=1
```

```
iterationPart=0
```

定义每夜构建计划的下一个版本号为：

```
${project. var["fixPart"]}. ${project. var["releasePart"]} build ${project. var["iterationPart"]. increaseAsInt ()}
```

定义发布构建计划的下一个版本号为：

```
${project. var["fixPart"]}. ${project. var["iterationPart"]. setValue(1), project. var["releasePart"]. (increaseAsInt (), value)}
```

This way, builds in "release" schedule will get versions like: foo-1.1.1, foo-1.1.2, foo-1.1.3, ..., and builds in "nightly" schedule will get versions like: foo-1.1.1 build 1, foo-1.1.1 build 2, foo-1.1.1 build3,, foo-1.1.2 build 1, foo-1.1.2 build2, ...

注意

在计算OGNL表达式的值的时候，Luntbuild会替换所有的"."字符串为"_"，所有的空格则被替换为"-". 该构建的版本号字符串会用作给VCS系统中相应的源码打标签的值。例如，如果构建的版本号是"v1.0 build256"，该构建对应的源码会被打上名为"v1_0-build256"的标签。

注意

Luntbuild基于版本号给源码打标签。如果在Luntbuild里面存在多个项目，或者配置了多个构建，那么您要确保没有重复定义的版本号字符串。例如，如果您给build1的下一个构建版本配置为"v1.0"，build2的下一个构建版本号为"v1.5"，如果两个构建使用同一个VCS和包括了同样的模块，那么可能在5次或者更多次构建之后，build1和build2的版本号就会相同了，这样我们就无法区分相应的源码，因为它们有相同的标签。

Work directory(工作目录)

在此设置构建计划的工作目录。如果设置的不是绝对路径，那么就假设是相对于该构建计划所属项目的工作目录。如果该属性为空，那么系统使用项目工作目录，即：
`<global_work_dir>/<project_name>`。其中`<global_work_dir>`是Luntbuild的顶层工作目录，`<project_name>`是构建计划对应的项目的名称。默认的工作目录可能造成同一个项目的多个构建计划使用同一个工作目录。

See build work directory 查看工作目录

.

Trigger type(触发器类型)

选择构建计划的触发器类型。如果选择“manual”(手工)那么意味着构建计划的构建任务只能通过手工触发。“simple”则周期性触发(定义一个时间间隔，每隔多少分钟触发一次)。“cron”则用来定义cron风格的触发器。关于怎么配置cron触发器的详情请参考<http://www.opensymphony.com/quartz/>

Cron expression(Cron表达式)

设置构建计划的cron表达式，格式为 `<seconds> <minutes> <hours> <day-of-month> <month> <day-of-week>` 例如

`0 1 * * ?`

的意思是每天的早上1点。关于该格式的详情，请参考Cron触发器指南[\[http://www.opensymphony.com/quartz/tutorial.html#cronTriggers\]](http://www.opensymphony.com/quartz/tutorial.html#cronTriggers)。

Repeat interval (minutes)(时间间隔(单位：分钟))

在此设置构建计划实施的时间间隔，单位为分钟。

Build necessary condition(构建必要条件)

该属性可选。如果为空，那么默认值为“vcsModified or dependencyNewer”。Luntbuild使用构建必要条件来决定当前构建是否有必要进行。构建必要条件是一个OGNL表达式。当该表达式的值为true的时候，就认为构建有必要进行。该OGNL表达式的根对象是当前的Schedule[\[../javadoc/com/luntsys/luntbuild/db/Schedule.html\]](http://../javadoc/com/luntsys/luntbuild/db/Schedule.html)对象。下面的例子展示了是OGNL表达式的格式：

1. `vcsModified` - 如果当前构建要访问的代码库的内容发生了改变那么认为该表达式的值为true。
2. `dependencyNewer` - 如果依赖的构建计划更新了，那么该表达式的值为true
3. `dependencySuccessful` - 如果依赖的构建计划里的最新构建都成功了，那么该表达式的值为true
4. `always` - 其值总是为true，用于强制运行构建
5. `never` - 其值总是为false，通过设置该表达式来停止构建
6. `alwaysIfFailed` - 如果上一次构建失败了，那么该表达式的值始终为true，如果上一次构建成功了，那么该表达式的值和“vcsModified or dependencyNewer”一致。
7. `project["testcvs"].vcsModified` - 如果项目“testcvs”对应的代码库的“development”视图的值改变了，那么该表达式的值为true。
8. `execute("/path/to/command.sh") == 0` - 如果指定的命令执行的返回值为0，那么该表达式

的值为true

注意

一些特殊的符号，例如‘\’，‘”’，应该在前面使用‘\’符号，这和java字符串语法是一致的。

以上的表达式都可以加上‘!’前缀，用来求相反的值， 例如如果当前项目代码库中的代码没有变化那么!vcsModified的值为true。

以上的表达式可以使用“and”和“or”逻辑运算符组合成新的表达式。例如表达式 `vcsModified or execute("/path/to/command.sh")==0`， 如果项目代码库中的代码发生了变化，或者指定的命令运行的返回值为0，那么该表达式为true。

关于OGNL表达式的语法请参考 <http://www ognl.org>

Associated builders(该构建计划使用的builder)

选择当前构建计划要使用的builder。执行顺序和选择的顺序一致。

Associated post-builders(该构建计划使用的post-builders)

选择当前构建计划要使用的post-builders。 在“post-build strategy”设置了合适的值（post build strategy属性）， 那么在所有选择的builder执行完 后就会执行这里选择的post-builders。

Build type(构建类型)

选择构建计划的构建类型，完整构建(clean build)相对可靠，但是速度更慢。 增量式构建(Incremental build)速度快，但是没那么可靠。 我们建议所有重要的构建计划例如日构建和发布构建都应该使用完整构建，而那些 频繁的构建计划，例如每小时构建使用增量式构建。

注意

该设置只有构建不是手工触发的时候才有效。

Post-build strategy(Post-build策略)

设置构建计划的Post-build策略，目前支持的策略如下：

do not post-build(不运行post-builders)

构建完成后不执行前述的post-builders。

post-build when success(成功后运行post-builders)

只有构建成功后才运行前述的post-builders。

post-build when failed(失败后运行post-builders)

只有构建失败后才运行前述的post-builders。

post-build always(无条件运行post-builders)

构建之后无条件运行post-builders

注意

该设置只有构建不是手工触发的时候才有效。

Label strategy(标签策略)

选择构建计划的标签策略。Luntbuld提供了以下策略：

label successful builds(给成功构建的代码打标签)

只给那些成功完成的构建对应的代码库打上相应的标签。

`do not label`(不打标签)

不论成功失败，构建结束后都不会打标签。

`label always`(无条件打标签)

不论成功失败，构建结束后都会打上标签。

注意

如果在第一次构建的时候不打上标签，那么产生的构建以后将不能够进行重新构建（因为 如果不打标签的话，Luntbuild无法断定特定构建所对应的源代码）。

注意

该设置只有构建不是手工触发的时候才有效。

Notify strategy(消息通知策略)

选择构建计划的消息通知策略。Luntbuild提供了以下策略： Choose the notify strategy for this schedule. There are following strategies:

`notify when status changed`(状态改变时发送通知)

如果当前构建和上一次构建的状态相比有变化的话就发送消息通知。 也就是说，如果上一次构建失败，而本次构建成功，或者上一次构建成功，本次构建失败，这些情况下都会发送通知。

`notify when failed`(构建失败时发送通知)

构建失败时发送通知

`notify when success`(构建成功时发送通知)

构建成功时发送通知

`do not notify`(不发送通知)

构建结束后不发送通知

`notify always`(无条件发送通知)

无论构建状态如何，结束后都发送通知。

注意

该设置只有构建不是手工触发的时候才有效。

Dependent schedules(依赖的构建计划)

选择当前构建计划所依赖的其他构建计划。 如果scheduleA依赖scheduleB，那么Luntbuild在触发scheduleA之前会先触发scheduleB。 构建计划之间的依赖关系间接定义了项目间的依赖关系。

Dependency triggering strategy(依赖性触发策略)

决定当前的构建计划被触发的时候怎样相应的触发其依赖构建计划。目前支持的策略有：

`trigger schedules this schedule depends on`(触发本构建计划所依赖的构建计划)

触发本构建计划所依赖的构建计划。 在当前的构建计划触发前会触发其依赖的构建计划。 例如，如果当前的构建计划依赖于其他的构建计划中的一些组件，那么您可以使用本策略，这样可以保证本构建计划使用到的组件是最新的。

`trigger schedules that depends on this schedule`(触发依赖本构建计划的其他构建计划)

触发依赖本构建计划的其他构建计划。在本构建计划触发后将触发依赖于本构建计划的其他构建计划

划。例如，如果本构建计划 要构建的组件要被其他的构建计划使用，那么您可以通过本策略来保证组件更新时，使用该组件的其他产品也被更新。

`trigger all dependent schedules`(触发所有依赖的构建计划)

本策略综合了上面两个策略，也就是说， 在本构建计划触发前先触发本构建计划依赖的其他构建计划，在本构建计划触发完后再触发依赖于本构建计划的其他构建计划。

`do not trigger any dependent schedules`(不触发任何依赖的构建计划)

不管本构建计划依赖于哪些构建计划，或者依赖于本构建计划的其他构建计划，触发本构建计划的时候都跟它们无关。

注意

该设置只有构建不是手工触发的时候才有效。

`Build cleanup strategy`(清除构建策略)

决定怎样清除本构建计划里面的历史构建。

`do not cleanup builds automatically`(不自动清除构建)：只有通过手工方式删除构建

`keep builds by day`(以天为单位保存构建成果)：保存指定的天数。

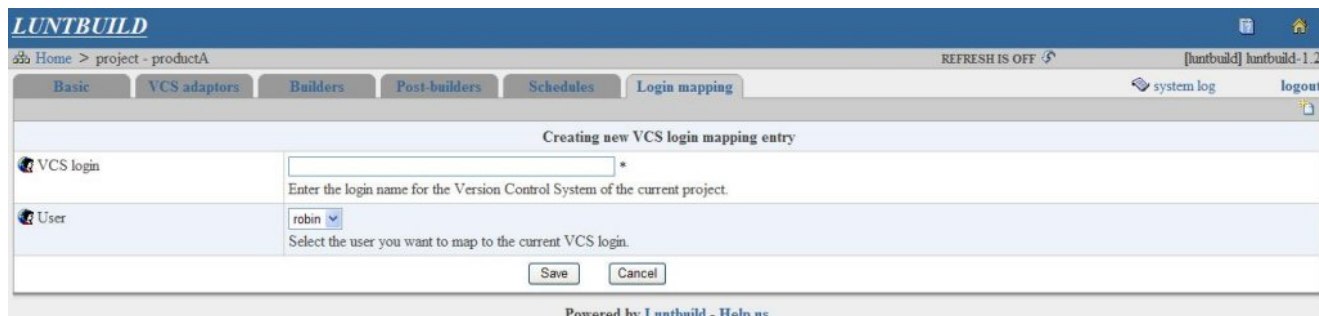
`keep builds by count`(以计数来保存构建成果)：保存指定的构建数。

注意

如果某计划的构建正在运行中，这个时候该计划中的新的构建被触发了，那么该构建会添加到这个构建计划的构建队列后面。在项目的构建计划标签页中会显示该等待执行的构建队列。

第 11 章 定义项目的登录映射表

本页向您展示从VCS用户到Luntbuild用户的映射表。当Luntbuild获得最近做check in操作的VCS用户清单的时候，会通过映射表获得相应的Luntbuild用户名，如果有必要就会向这些用户发送通知。如果没有针对某个特定的VCS用户做映射，那么Luntbuild会自动将VCS用户名映射到相同的Luntbuild用户名。



The screenshot shows the Luntbuild web interface. The page title is "LUNTBUILD". The breadcrumb is "Home > project - productA". The navigation bar includes "Basic", "VCS adaptors", "Builders", "Post-builders", "Schedules", and "Login mapping". The "Login mapping" tab is active. The main content area is titled "Creating new VCS login mapping entry". It contains two rows: "VCS login" with a text input field and a "*" character, and "User" with a dropdown menu showing "robin". Below the dropdown is a "Save" button and a "Cancel" button. The footer says "Powered by Luntbuild - Help us".

VCS login(VCS登录用户)

在此输入当前项目对应的版本控制系统的登录用户名。

User(项目用户)

选择对应的Luntbuild用户

第 12 章 创建/修改项目的Ant Builder

builder

buildVersion(构建版本号)

该属性值就是构建的当前版本号。

buildDate(构建日期)

该属性值就是构建的时间/日期。

artifactsDir(制品输出目录)

```
<propertyfile file="stage/buildInfo.properties">
  <entry key="buildVersion" value="${buildVersion}" />
  <entry key="buildDate" value="${buildDate}" />
</propertyfile>
```

```
<property name="buildVersion" value="luntbuild-1.0"/>
<property name="artifactsDir" value="distribute"/>
<property name="buildDate" value="" />
```

注意

ant构建文件的任何输出信息和错误信息都不需要您手工重定向到您的日志文件，这些事情留给ant去做，让ant将这些信息分别输出到stdout和stderr。Luntbuild能够捕获他们并将他们写到预先准备的日志文件里面，然后将日志文件发布到制品输出目录(artifactsDir)下。

第 13 章 创建修改项目的Maven Builder

buildVersion

该属性的值为构建的当前版本。

buildDate

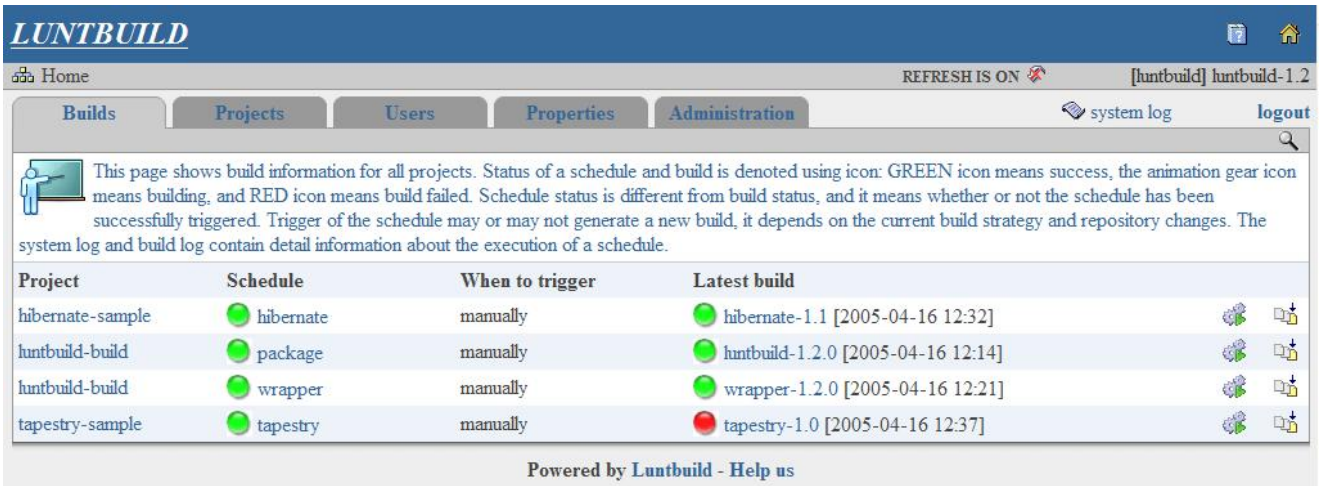
该属性的值为构建的日期时间。

artifactsDir

该属性指定了构建的制品输出目录。构建产生的最终的制品都会保存在“artifactsDir”属性对应的目录或者其子目录下。Luntnbuild在这个目录下只保存构建有关的信息。您可以导出构建的内部制品，如下：

```
<zip basedir="stage" destfile="${artifactsDir}/${buildVersion}.zip"/>
```




第 14 章 构建计划的快照



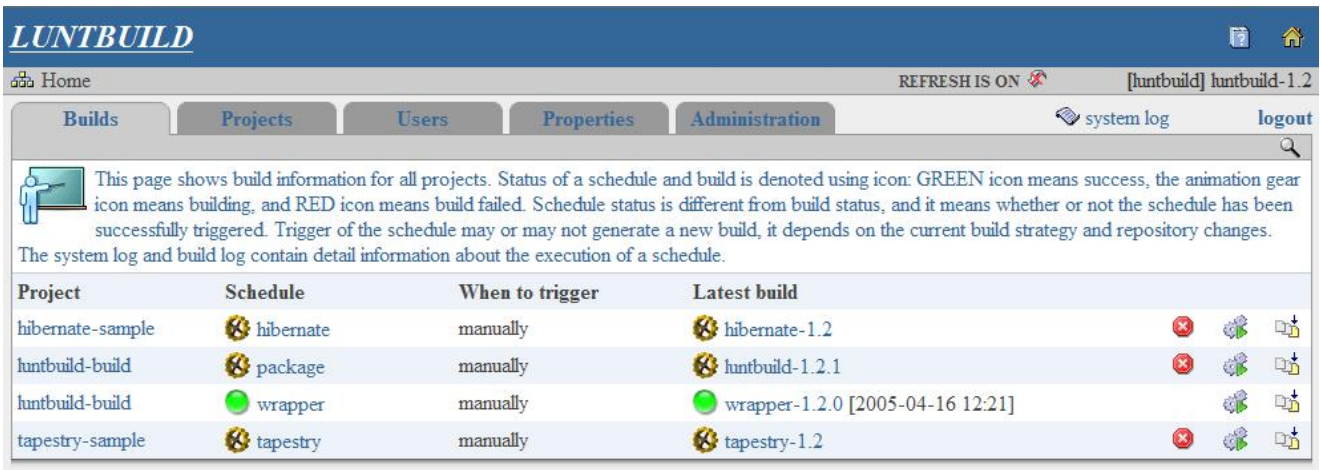
The screenshot shows the LUNTBUILD web interface. At the top, there's a navigation bar with tabs: Builds, Projects, Users, Properties, and Administration. The 'Builds' tab is selected. Below the navigation bar, there's a header area with a 'Home' link, a 'REFRESH IS ON' button, and a user/logout section. The main content area has a table with the following data:

Project	Schedule	When to trigger	Latest build
hibernate-sample	hibernate	manually	hibernate-1.1 [2005-04-16 12:32]
huntbuild-build	package	manually	huntbuild-1.2.0 [2005-04-16 12:14]
huntbuild-build	wrapper	manually	wrapper-1.2.0 [2005-04-16 12:21]
tapestry-sample	tapestry	manually	tapestry-1.0 [2005-04-16 12:37]

At the bottom of the table, there's a footer that says 'Powered by Luntbuild - Help us'.

该页展示了Luntbuild里所配置的所有构建计划。“Project”列对应的值代表了构建计划所关联的项目。“Schedule”列对应了该构建使用的构建计划。“When to trigger”列是触发构建计划的日程安排。“Latest build”列是该构建计划最后一次的构建实例。最后一列包括了两个图标。最右边的图标是“构建历史”(“history builds”)图标，用于访问对应构建计划的所有历史构建记录。这个图标左边的图标是代表“手工运行构建”(“run manually”)的意思。您可以通过点击该图标来手动启动构建。如果构建已经通过手工方式启动了，那么该按钮左边会出现一个“停止构建”图标。这个时候点击“停止构建”图标，就可以强制停止正在运行的构建。


接下来的例子是有部分构建已经在运行状态的构建计划列表。



The screenshot shows the LUNTBUILD web interface. At the top, there's a navigation bar with tabs: Builds, Projects, Users, Properties, and Administration. The 'Builds' tab is selected. Below the navigation bar, there's a header area with a 'Home' link, a 'REFRESH IS ON' button, and a user/logout section. The main content area has a table with the following data:

Project	Schedule	When to trigger	Latest build
hibernate-sample	hibernate	manually	hibernate-1.2
huntbuild-build	package	manually	huntbuild-1.2.1
huntbuild-build	wrapper	manually	wrapper-1.2.0 [2005-04-16 12:21]
tapestry-sample	tapestry	manually	tapestry-1.2


At the bottom of the table, there's a footer that says 'Powered by Luntbuild - Help us'.

在这一页的右上角还有一个“搜索”图标。可以通过该功能查找特定的构建实例，还可以对查询结果进行操作，例如可以删除查询结果里面列出的构建实例。

构建计划栏靠左的图标显示了构建计划的执行状态。构建计划的执行状态跟构建状态是有区别的。它代表了构建计划是否成功的触发。触发构建计划不一定会产生一个新的构建实例，这取决于当前的构建的策略和代码库是否已经改变。即使构建成功了，构建计划执行状态可能是“失败”(“fail”)，例如，可能是因为发送通知邮件出错了。相反的例子是，尽管构建失败了，但是构建计划执行状态是“成功”(“successful”)，因为构建计划成功触发，只是构建本身失败了。跟构建计划执行有关的细节可以查看系统日志，即每一页上面的“system log”链接。

有两种类型的构建，一种是clean(完整)构建，另一种是incremental(增量式)构建。执行clean(完整)


构建的时候，Luntbuild首先将目录清空，然后做一个对VCS模块的完整的检出操作。执行incremental(增量式)构建，Luntbuild只更新那些自上一次检出后发生改变的源文件。而且在新的构建开始之前增量式构建的文件不会被清除。增量式构建速度很快，但是没有完整构建可靠。例如，如果某人删除了VCS里面的一个文件，对于特定的VCS如CVS，其更新命令并不能将工作目录里对应的文件也进行删除。从而可能导致错误的构建。


每个构建计划的构建历史信息可以点击图标该图标(“history builds”)位于构建计划的每一行记录的右侧。点击该图标后会列出指定的构建计划的所有构建的历史信息。通过点击列表中某个构建版本对应的超链接，就可以查看该构建的详细信息。如下图：

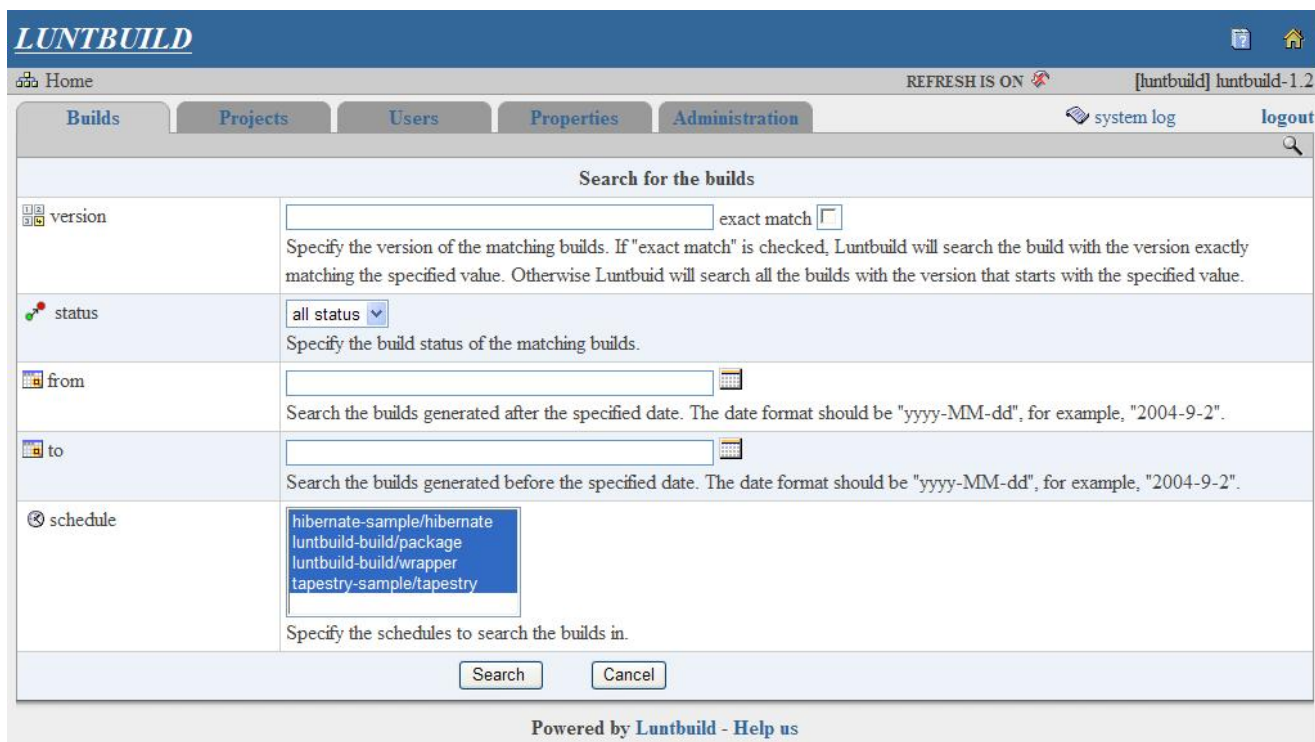


The screenshot shows the Luntbuild web interface. At the top, there's a navigation bar with tabs: Builds, Projects, Users, Properties, and Administration. Below the tabs, there's a table titled "2 builds found". The table has columns: project, schedule, Version, status, finish date, and minutes used. There are two rows of build data.

project	schedule	Version	status	finish date	minutes used
hibernate-sample	hibernate	hibernate-1.1	success	2005-04-16 12:32	2
hibernate-sample	hibernate	hibernate-1.0	failed	2005-04-16 12:27	0

在这个页面的“Build artifacts”(构建制品)这个区域，您可以下载相应的制品。同时您也可以上传制品，或者创建一个新的目录。如果您为某个构建提供补丁，那么这个功能就很有用了。该页面还可以访问构建的日志。日志文件可以帮助您分析构建失败的原因。在当前构建和上一次构建之间，代码库中的文件和目录的变更都会记录在版本变更日志文件里面。如果在生成构建的时候选择了“label build”，那么在该页面的顶端会显示一个“重新构建”(“rebuild”)图标。如果点击该图标，那么您可以重复该构建。重新构建的过程完全使用和上一次构建的一致的VCS设置。在进行重复构建的时候相应的VCS配置明细会记录在构建日志里面。点击“Builds”标签可以回到构建计划的页面。

在构建计划和历史构建信息的页面都可以通过点击查询构建图标(“search build icon”) 来查找构建历史信息。查询页面如下：



The screenshot shows the Luntbuild search interface. It has a search bar at the top with the text "Search for the builds". Below the search bar, there are several search criteria: version, status, from, to, and schedule. Each criterion has a corresponding input field and a description. At the bottom, there are "Search" and "Cancel" buttons.

Search for the builds

version exact match ☐
Specify the version of the matching builds. If "exact match" is checked, Luntbuild will search the build with the version exactly matching the specified value. Otherwise Luntbuild will search all the builds with the version that starts with the specified value.

status all status
Specify the build status of the matching builds.

from
Search the builds generated after the specified date. The date format should be "yyyy-MM-dd", for example, "2004-9-2".

to
Search the builds generated before the specified date. The date format should be "yyyy-MM-dd", for example, "2004-9-2".

schedule
Specify the schedules to search the builds in.

Search Cancel

该页面提供了以下查询条件：

Version(版本)

指定要查询的构建的版本号。如果选择了“exact match”(完全匹配)，Luntbuild的查询功能会根据指定的版本号进行精确的匹配。否则，只要以指定的版本号作为开头的版本都会纳入查询结果。

Status(状态)

指定要查询处于哪种状态的构建。目前提供了以下几个选择项。

```
all status
successful
failed
running
```

From(起始日期)

查询指定日期以后生成的构建。日期格式为“yyyy-MM-dd”，例如“2004-9-2”


To(终止日期)


查询指定日期之前生成的构建。日期格式为“yyyy-MM-dd”，例如“2004-9-2”

Schedule(构建计划)

指定查询哪个构建计划下生成的构建。

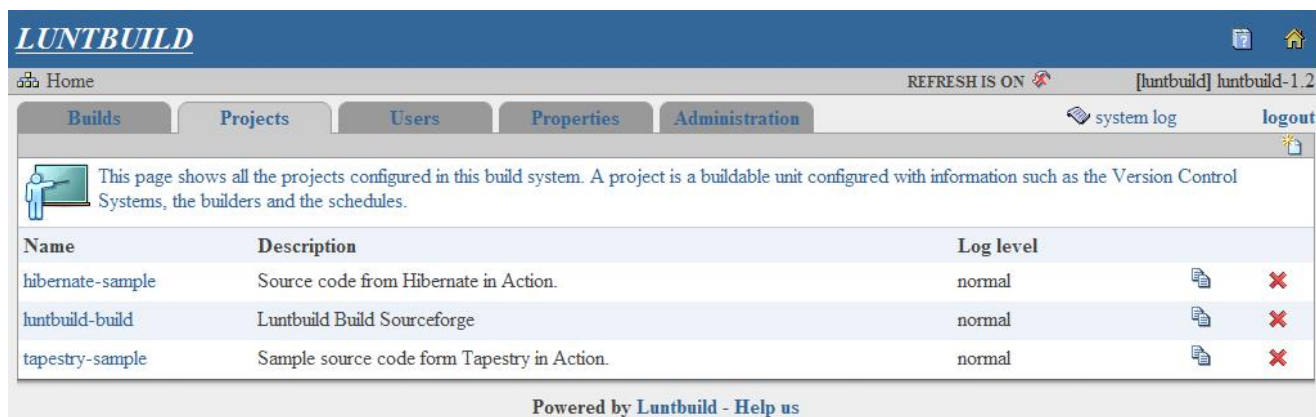
这一页显示匹配查询条件的构建历史信息列表。

点击删除按钮 .

可以删除列表里面相应的构建 也可以点击移动(或者叫提升)按钮  移动选定的构建，然后会出现移动构建(“Move builds”)页面。在这个页面您可以选择一个需要将构建移动到哪个目标构建计划(“Destination schedule”)。移动构建提供了以下功能：

1. 在删除构建计划或者项目前将构建保存到其他地方
2. 提升重要的构建。例如，我们可以将“nightly”(日构建)构建计划的某个构建提升到“release”(发布构建)构建计划，这样该构建就可以 准备作为对外发布。


第 15 章 项目快照



The screenshot shows the Luntbuild web application interface. The top navigation bar includes links for Home, Builds, Projects, Users, Properties, and Administration. The Projects page is active, displaying a list of configured projects. A message at the top explains that a project is a buildable unit configured with information such as the Version Control Systems, the builders and the schedules. The table below lists the projects:

Name	Description	Log level	Clone	Delete
hibernate-sample	Source code from Hibernate in Action.	normal		
luntbuild-build	Luntbuild Build Sourceforge	normal		
tapestry-sample	Sample source code form Tapestry in Action.	normal		

At the bottom of the interface, it says "Powered by Luntbuild - Help us".

本页显示了当前系统中配置的所有项目。创建项目需要输入版本控制系统信息，Builder信息，构建计划信息。还可以通过点击  进行项目克隆。这个功能对创建和现有项目类似的项目很有用。

第 16 章 使用Luntbuild远程调用接口

Luntbuild提供了一整套远程调用接口，这套API提供了以下功能：

1. 触发任何构建计划
2. 配置系统属性以及项目属性
3. 搜索构建任务，获得其构建的信息，例如其构建制品的url等等

通过利用Hessian的web service 协议，这些API用起来很容易。不过有两个jar文件需要加入classpath，这是最基本的，这两个文件是 hessian-3.0.8.jar 和 luntbuild-api.jar. 这两个文件位于“remoting”目录下。该目录下还有一些描述这些API用法的例子。例如，TriggerBuild这个例子可以用于实现实时的持续集成，也就是说，无论哪个时候，只要对代码库进行了检入(checkin)的操作，Luntbuild就可以立即触发构建任务。下面我们以一个cvs代码库的例子来演示一下：

1. 在项目里面创建一个手工触发的构建计划，用来实现实时的持续集成。为了让构建更加迅速，我们将构建配置为增量式构建。
2. 检出cvs代码库下CVSROOT目录下的“loginfo”文件，并在该文件内容后添加：

```
testcvs cmd /c d:/lunt/cvs/lunt/luntbuild/remoting/samples/trigger_build.bat
```

注意

在编辑前，您应该用cvs客户端检出该文件，就象您要编辑代码库中其他文件的方式一样

testcvs可以用cvs代码库下的任意目录名代替，所有在该目录下checkin的操作都会触发trigger_build.bat命令。trigger_build.bat位于“remoting/samples”目录下。当然，您也可以复制相关的文件到任何机器上（远程API的jar文件，TriggerBuild.class文件，trigger_build.bat文件），只要您在该机器上安装了JDK1.4或者更高的版本。您需要修改trigger_build.bat前面的目录为您的环境里的相应的目录。同样还需要修改trigger_build.bat文件里面和classpath有关的部分，将下面这些部分调整到您的实际环境一致，包括Luntbuild服务器的url，项目名，构建构建计划。在Unix平台下，基于trigger_build.bat文件，创建一个trigger_build.sh脚本文件是很容易的。

3. 检入(checkin)“loginfo”文件。从现在开始，在您在该文件中配置的目录下的每次检入(checkin)操作将会触发trigger_build命令，该命令会直接触发指定的构建计划，从而运行新的构建来验证当前VCS的健康状况。

注意

只要支持在做检入(checkin)的时候可以触发自定义的外部命令的版本控制系统都可以实现实时的持续集成。

第 17 章 调试构建过程中的问题

如果在项目构建的过程中遇到什么问题，您首先应该检查相应的构建日志(build log)。默认情况下，Luntbuild只将informational, warning, 和error这三类信息记录在日志。您可以修改项目配置信息，将“Log level”(日志级别)设置为“verbose”。Luntbuild的日志记录在“luntbuild installation directory”/logs目录下，“luntbuild installation directory”就是luntbuild的安装目录。日志里面记录了luntbuild检测到的VCS变更情况，还有其他跟特定的构建无关的信息。您可以查看这些日志来分析问题，比方说不知道什么原因构建计划或者手工构建没有执行。可以直接点击每一页右上端的“system log”来查看最近的日志。不过，对于那些旧的日志文件，您需要直接访问logs目录下的相应的日志文件。

第 18 章 Luntbuild安全性

Luntbuild内置了用户认证和授权处理的安全机制。Luntbuild的安全配置是独立于Servlet容器(或应用服务器)的,所有的配置都在Luntbuild应用程序里面完成。

18.1. 整体安全概念一览

只有通过认证的用户才可以访问Luntbuild。Luntbuild的功能可以指派给以下的几个基本角色:

站点管理员
项目管理员
项目构建者
项目参与者
角色描述

站点管理员

拥有该角色的用户代表了root用户。拥有“站点管理员”角色的用户可以访问所有的Luntbuild的功能,没有任何限制。下面是“站点管理员”有权执行的所有任务:

用户管理
系统属性管理
创建项目
管理构建计划
给不同的用户指派“项目管理员”角色
检查系统日志

项目管理员

拥有该角色的用户具有项目管理的所有功能。下面是“项目管理员”有权执行的所有任务:

修改项目配置
管理VCS模块
管理构建
给用户指派项目中的角色(项目管理员,项目构建者和项目参与者)

项目构建者

拥有该角色的用户只能管理项目构建有关的任务。下面是“项目构建者”有权执行的所有任务:

手工触发构建

项目参与者

这个角色是最受限制的角色,下面是“项目参与者”有权执行的所有任务:

查看构建结果
查看构建日志
下载构建的制品

18.2. 怎样配置Luntbuild的安全

Luntbuild的安全配置有两种方式。其中一种通过修改配置文件（例如修改<your app.server>/webapps/luntbuild/WEB-INF0/applicationContext.xml文件）基于安全方面的考虑，您首先应该修改站点管理员的口令。修改默认的站点管理员口令请参考inMemoryAuthenticationDAO这一节。

```
<bean id="inMemoryAuthenticationDAO" class="net.sf.acegisecurity.providers.dao.memory.InMemoryDaoImpl">
    <property name="userMap">
        <value>
            <!-- this is the build-in site admin user - please change the password for security reasons.
            However, name of the user, and its role should not be changed! -->
            luntbuild=luntbuild,ROLE_SITE_ADMIN,ROLE_AUTHENTICATED
            dummy=dummy,ROLE_AUTHENTICATED
        </value>
    </property>
</bean>
```

注意

在安装Luntbuild的时候也可以修改站点管理员的口令

Luntbuild还提供了一个dummy用户，该用户权限最小，仅用于测试。其他的用户都应该通过Users(用户)标签页来创建，详情请参考

添加Luntbuild用户

这一章。第二种是基于数据库的，可以在Project(项目)标签页定义项目有关的角色。详情请参考新建一个项目

这一章。

第 19 章 数据导出和导入

Luntbuid的数据(项目, 构建计划等等)可以通过“Administration”(管理)页进行导出和导入。

19.1. 数据导出

指定您要导出的文件的位置, 然后点击“Export”(导出)按钮, 数据就会被导出了。导出的数据以XML的格式保存, 虽然导出的文件可以手工编辑, 但是千万要小心。您可以将导出的数据再次导入到其他的Luntbuild实例。如果指定的目录为相对路径, 那么该目录将相对于Luntbuild的安装目录, 文件将会保存在该相对路径下。我们强烈推荐使用绝对路径, 这样可以很容易就定位文件的位置。

注意

注意在这里指定的文件将被创建在Luntbuild服务器上, 而不是运行web浏览器的客户机上。

注意

如果系统中存在大量的构建(build)实例, 该操作将会消耗较长时间。我们强烈建议删除一些没有用的构建(build)实例, 这样可以加速数据导出和导入的速度。

19.2. 数据导入

指定您要导入的文件的位置, 然后点击“Import”(导入)按钮, 数据就会被导入了。

注意

执行该操作时, Luntbuild将会先清空当前数据库, 然后再进行导入。导入时间的长短取决于被导入的文件的大小。

19.3. 通过导出和导入功能来进行数据迁移

使用导出和导入功能可以很容易实现数据迁移。例如您现在有一个运行在HSQLDB数据库上的Luntbuild实例, 因为某个原因要切换到MySQL数据库上。通过以下步骤可以完成数据迁移:

1. 首先启动基于HSQLDB的Luntbuild实例, 然后将数据导出到某个文件, 假设为data.xml文件。
2. 停止Luntbuild, 修改applicationContext.xml文件, 将数据库切换到MySQL。
3. 启动Luntbuild, 将data.xml文件导入系统。
4. 搞定!

附录 A. 构建时支持的OGNL 表达式

`build.version`

构建版本号

`build.artifactsDir`

构建副产品目录，它是`build.publishDir`的子目录。

`build.publishDir`

构建发布目录

`build.junitHtmlReportDir`

JUnit 测试的Html 报告目录

`build.schedule.name`

Build schedule 的名字。

`build.schedule.description`

Build schedule 描述

`build.schedule.workingDir`

Build schedule 工作目录。

`build.schedule.vcsModified`

VCS改变了吗？

`build.schedule.dependencyNewer`

是否检查依赖的schedule更新？

`build.schedule.project.name`

构建项目名。

`build.schedule.project.description`

构建项目描述。

下面是OGNL表达式的一些例子：

1. `build.schedule.vcsModified` - 假若当前构建所使用的VCS内容改变了，值为`true`
2. `build.schedule.project["testcvs"].vcsModified` - 假若“development”视角的“trstcvs”项目的VCS内容发生了改变，值为`true`
3. `ant("/path/to/command.xml", "targetA") == 0` - 假若执行ant脚本`/path/to/command.xml`的`targetA`，返回`success`的话，值为`true`。

注意

`'\','"` 这种特殊字符请按照Java字符串的规则前面加上`'\'`转义。 若不是绝对路径，Ant文件的路径会被假设为相对于当前项目工作目录，假若`target`是`"`或者`null`，会执行默认的`target`。

4. `execute("/path/to/command.sh") == 0` -

注意

假若执行此命令的返回值为0, 值为true ' \ ', ' ' ' 这种特殊字符请按照Java字符串的规则前面加上' \ ' 转义。

以上的表达式可以在前面加上'!' 来对值反转, 比如假若当前项目的VCS内容没有改变, !modified会是true.

以上的表达式可以使用 "and", "or" 进行组合。比如, modified or execute("/path/to/command.sh")==0的值, 在当前项目的VCS内容变化了, 或者执行特定命令的返回为0的时候, 会是 true.

请到<http://www.ognl.org>学习OGNL表达式的语法。