**CVS基本原理**

在通过CVS进行版本控制时，应该首先搭建CVS服务器。目前，有很多的CVS服务器，例如CVSNT等。本书将带领读者搭建CVSNT服务器，并使用NetBeans连接此服务器讲解CVS的使用。

CVS的工作原理如图10-1所示。CVS服务器用于保存项目源（包括源代码、项目文件、文档等任何需要进行版本控制的对象）。但是CVS服务器中的项目源***不能直接被使用。要将其导出到客户端，在客户端建立的副本才能对其进行编辑修改***。

在进行团队开发时，需要对代码进行共享。通常开发人员首先将自己的代码保存（也叫导入或者签入，英文为Check in）到CVS服务器，这样团队中的其他成员可以把这些代码导出（也叫签出，英文为Check out）到本地计算机上。这样就可以在多个客户端上保存同一项目的副本，实现了代码的共享。

需要说明的是，CVS服务器端和客户端可以在同一台机器上，也可以是在局域网中的不同机器上，或是在互联网中的不同机器上。

|  |
| --- |
| <http://book.51cto.com/files/uploadimg/20080525/1507460.gif> |
| 图10-1  CVS基本原理示意图 |

客户端可以使用多种方法与服务器建立连接。在建立连接时，需要给出服务器的地址，就像用IE浏览器访问Web网站时要输入网站的地址一样。CVS服务器连接地址的格式定义如下所示：

[:访问方法:][[[用户名][:密码]@]主机名[:[端口]]]/路径  
其中，连接地址各个部分的说明如下。  
访问方法：类似于连接协议，NetBeans支持以下访问方法。  
local：连接到本地服务器。  
ext：使用系统外部的方法（如rsh）连接。  
pserver：使用pserver方法连接。  
fork：将本地机器作为远程机器来连接（通常用于测试）。  
用户名：在CVS服务器上注册的用户名。  
密码：注册用户时使用的密码。  
主机名：CVS服务器的主机名称或IP地址。  
端口：连接服务器时使用的端口号。如果为空表示使用默认端口。  
路径：在CVS服务器中注册的用于存放项目的仓库路径。

例如，要使用用户名wind9494访问路径为cvs.dev.java.net/cvs的仓库，那么连接地址应该为：pserver:wind9494@cvs.dev.java.net:/cvs。

2.

update：从库（cvs-repository）中更新代码到本地

update只有在本地代码版本低于cvs-repository中的项目代码文件版本时，才能update成功。如果版本相同，update了本地代码不会改变，因为版本相同cvs认为你的代码为最新，不要更新。如果本地代码版本低于库中的，那么说明不是最新需要更新，并且此时，如果不更新，是无法commit的，目的是为了保证每次commit的版本最新，确保版本递增，代码有序更新。(***其中一个例子：如果现在拿下来的版本是1.4,本地修改编辑后。如果有其他人commit了变为1.5,那么等我修改完了。再commit之前我会update,由于1.4比1.5少，所以cvs会将1.5中变化的内容融合入我修改过的1.4版本。并且此时本地文件版本变为1.5,如果没有冲突，我上传本地文件成为1.6.如果有冲突，我要手动改动。需要注意的是，cvs允许带着冲突的文件上传，也可以不update直接commit,但是这些都很容易使cvs服务器中的文件被破坏，带着冲突符号，导致不可用，所以必须的做法是，先update没冲突以后再上传***)

syncronize：同步比较代码。这个仅仅是比较代码而已，开发者不能修改代码。但是通过这个操作，可以找出和主线不同的代码文件，方便比较而已。

commit：提交本地代码到库。此时的操作其实是，本地项目代码文件覆盖库该项目相应代码。此时库和你自己刚上传的本地代码文件版本会变高一个级别（如1.4变成1.5）

commit大家都会，但是commit只有在本地版本和库版本相同（即最新版本）的时候才生效

3.

CVS 基础

cvs 并不是 eclipse 独有, 而是 eclipse 支持 cvs. cvs 是用于几个程序员之间共享程序避免冲突的一个工具, 是 C/S 架构.

cvs 是 java 的一个开源项目.

cvs 存储库是开发人员存储要共享代码文件的地方. (repository)

存储库有两种模式:

pessimistic locking(悲观锁定) 目前服务器采用这种模式, 即同时只能有1个开发人员修改某个source.

optimistic locking (乐观锁定) 同时可以有多个开发人员一同修改某个source, 提交修改时, 存储库会自动合并所有修改.

默认情况下是使用 “乐观锁定”, 但是个人觉得还是”悲观锁定”好一点. 不会出现矛盾冲突.

eclipse 采用 “乐观锁定”

每次提交时, cvs都会为每一个文件分配一个版本号, history 管理.

cvs 还支持同一个模块同时进行多个开发流, 称为” 分支” (branch) 一个模块中的主开发流称为” 头” (head), 分支是从主开发流中分出的叉, 例如, 一个分支代表项目的一个测试版, 或者添加到代码中首次进行测试的某项新功能.

我们服务器没有使用分支,

1. 首先要创建一个存储库, repository. 共享代码位置

2. 将 eclipse 连接到存储库. (在eclipse中打开 Repository Exploring 视图-> new-> Repository Location-> add cvs.

   这里要输入 cvs 服务器名称, 主机名, cvs存储库, 提供用户名和密码, 等等.

如果你有一个项目想共享, 只需要点这个项目, 右键->Team->share project.

commit 的意思, 是提交文件, 即将文件上传到 CVS 存储库中.

update 获得更新, 即将服务器中的别人更新的内容(别人已经commit), 更新到自己本地文件.

orveride and update, 如果你对某个source进行了修改, 想回退到跟server同步状态, 则使用这个命令.

blog

团队开发中的要求

提供方法, 能够协调对源代码的更改, 并能继承这些更改

团队提交工作的历史记录

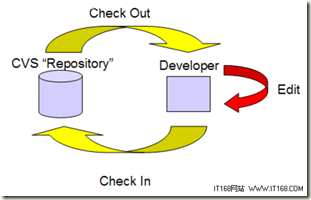
cvs 使用分支(branch) 模型来支持彼此独立而又高度相互依赖的多个工作流程. 这些分支是开发团队用来共享和集成正在进行中的工作的地方. 可以认为分支是一个共享的工作台, 当团队成员对源代码进行更改时就更新这个工作台, 这个模型准许从事CVS团队项目开发的每个人在进行更改时与其他成员共享其工作, 以及在项目进展期间访问其他成员的工作.

个人理解,

CVS 全称 current versions system (并发版本系统)

cvs 允许多个开发人员通过一个中心版本控制系统来记录文件版本, 从而达到保证文件同步的目录.

cvs 工作原理 copy-modify-merge （拷贝, 修改, 合并) 具体来说: cvs服务器创建一个源代码库 (cvs repository), 苦力可以存放许多不同的项目源程序, 由管理员统一管理. cvs 客户端支持不同平台. 在使用cvs任何源代码库之前, 开发人员需要将项目文件检出到本地(check out), 然后 edit, 最后提交文件(check in)到cvs服务器.

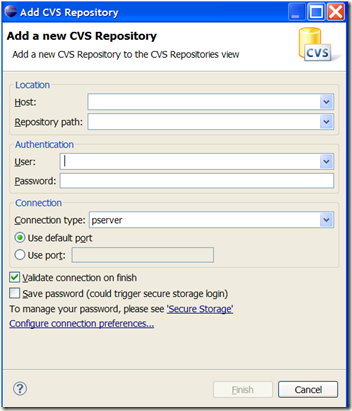
[](http://images.cnitblog.com/blog/370445/201401/07135246-af4dba410bc64cd6aa9f3909788583ed.png)

cvs Repository

cvs 源代码库 指的是cvs 存储所有修订版本历史记录的地方, 每个项目都有自己的确定的源代码库, 注意: 创建一个新的Repository需要管理员权限.

eclipse 创建新的 cvs repostority 方法如下:

cvs repository视图->右键new->Repository Location->弹出 Add CVS Repository对话框.

[](http://images.cnitblog.com/blog/370445/201401/07135248-7d0883b1a63143c1b2e4a2a6427f8f39.png)

1. Location

Location区描述的是CVS Repository的定位信息，包括主机(Host)与存储路径(Repository)。CVS Repository支持local与remote两种方式。对于本地，主机名为localhost或者127.0.0.1，存储路径前缀为 “:local:”。例如，:local:/usr/local/cvsroot表示CVS Repository位于本地/usr/local/cvsroot目录。其URL的写法也因[操作系统](http://product.it168.com/list/b/0501_1.shtml)而有所差别。对于Linux，:local:/usr/local/cvsroot等价于/usr/local/cvsroot。对于Windows，:local:c:/src/cvsroot等价于c:\src\cvsroot。

如果服务器在远端, 以上host可以输入IP地址, 然后存储路径是在这台机器本地的路径, 例如/home/cvs.

Repository的目录分为两部分：$CVSROOT/CVSROOT包含的是CVS的管理文件，而其余部分为用户自定义模块。我们除了使用Eclipse CVS客户端的方式指定Repository之外，还可以通过CVS命令行的方式，如下：

cvs -d /usr/local/cvsroot checkout yoyodyne/tc #–d选项表示direcotry，即CVS Repository对应的目录。

它等价于：

setenv CVSROOT /usr/local/cvsroot

export CVSROOT

checkout yoyodyne/tc

我们来分析一下CVS Repository对应的数据结构。我们假设当前的CVS Repository为/usr/local/cvsroot，如图。CVS Repository目录包含两部分，一部分为administrative files, 给CVS系统管理员使用，记录一些Repository相关的元数据等。另一部分就是源代码目录结构，这里源代码项目的根目录为yoyodyne。

我们服务器, 全部在 cvsroot 目录下

2. Authentication

authentication 描述认证信息

3. Connection

我们是采用默认方式

Connection区描述的是CVS客户端与CVS 服务器端的Repository之间的通信协议。当然，CVS客户端与服务器端可以是同一台机器，此时的主机名为localhost或者 127.0.0.1。按照类型与需求的不同，又分为以下三类协议，分别为：pserver，ext/extssh，pserverssh2。具体含义如 下：

pserver协议：指CVS客户端向服务器发送的密码以明文的方式传送。对于匿名用户，URL为cvs -d :pserver:fun.example.com:/usr/local/cvsroot; 而对于密码用户，按照显式与隐式划分，URL分别表示为：

cvs -d :pserver:doublelife@fun.example.com:/usr/local/cvsroot login

CVS password:

或者

cvs -d :pserver:doublelife:p4ss30rd@fun.example.com:/usr/local/cvsroot login

注意：cvs –d 命令表示指定CVS Repository。

ext/extssh：指使用SSH建立CVS客户端/服务器间的安全连接。因此，从应用场景上说，Pserver通常适用于普通用户或匿名用户，而对 于要求安全性较高的开发人员，则推荐使用ext/extsssh。ext与extssh的区别在于extssh只支持SSH1，而ext支持SSH1与 SSH2。换句话说，extssh是ext的子集，推荐使用ext。需要说明的是，Eclipse CVS Repository透视图本身已经内置有SSH客户端，无需额外的插件。下面是演示的是ext协议对应的CVS命令：

export CVS\_RSH=ssh

cvs -d :ext:doublelife:p4ss30rd@fun.example.com:/usr/local/cvsroot

pserverssh2：指的是pserver over SSH2。我们简单的理解为pserverssh2是SSH2与pserver两种协议的叠加。相比pserver与ext/extssh来说，并不常用，这里就不给出示例了。

对于CVS客户端/服务器间的通信协议，除连接类型外，还需要指定服务器端监听端口。服务器既可以使用默认端口，也可以根据需要，指定端口。比 如，pserver协议对应的默认端口为2401，如果该端口被其他的服务器占用，处于侦听状态，此时CVS 服务器可以为Repository指定可用端口。

在完成所有字段的填充后，我们推荐选中“是否在向导完成后验证连接有效”单选框，确保所填信息完整、有效。

作为示例，我们演示一下使用Eclipse来连接著名开源网站sourceforge的FTP软件filezilla的CVS Repository，如下图所示。这里我们采用的是匿名用户，对应的连接协议为pserver，端口默认值。

[](http://images.cnitblog.com/blog/370445/201401/07135250-8cf8ae715b2b41e7a01df461ac295c55.png)

如之前所述，CVS Repository为树状层级结构。在正式开始copy->modify->merge三部曲之前，有必要花些时间熟悉上图CVS的几个关键概念：Module，HEAD，Branches，Versions，Dates。

Module

如前所述，CVS Repository分为两部分，一部分是CVSROOT，描述CVS工程相关的元数据。另一部分即为Module，表示工程的模块。比如 filezilla分为三大模块，分别为FileZilla，Filezilla Server，Filezilla3。

Head

简单的说，HEAD表示CVS代码的主干与主体部分，正常情况下，我们对CVS的操作三部曲均发生在HEAD目录中。

Branch

我们没有使用 Branch

而对于一些特殊的情况，我们采用的Branch方式。Branch相对于HEAD而言，指的是代码的分支部分，我们简单的理解为补丁。举个例子，假设我 们的项目发布了第一个版本V1.0，第二版本V2.0正在开发当中，处于不稳定状态，随时有代码的改动。而与此同时，客户报告重大的bug，需要我们立即 修复。于是，我们check out V 1.0的代码，调试，并找到解决办法，并发布补丁。为了让补丁与开发代码隔离，我们可以创建一个新的branch，用于V 1.0的补丁，用户从branch中check out的是V1.0的补丁，而不会得到位于HEAD目录的尚在开发阶段V2.0代码。当然根据需要，开发人员可以将branch代码合并至HEAD中。

为更好地说明问题，我们从版本Revision控制的角度来对比HEAD与Branch的关系。一般说来，HEAD的Revision历史呈线性增长趋势，如下图:

由于Branch概念的介入，使得CVS并不局限于线性的开发，HEAD版本可以分为若干不同的Branch，每一个Branch是一个独立开发的自我 维护的开发线。如下图所示，出现了三个不同的分支，分支号的编排依赖于它分离出的主线版本。使用分支号允许一个特定版本分离出多个分支，图中 Revision 1.2同时派生出两个不同的Branch。一个分支也允许派生出多个子分子，Branch 1.2.2派生了一个sub branch。

一个Branch的变更可以很容易转移到HEAD中。可以通过CVS命令update配合-j选项实现合并。这里的-j表示join。

举个例子，我们当前的HEAD版本为1.4。现在需要将分支1.2.2合并到HEAD。

我们假设模块”mod”只包含一个文件”mod.c”。分支1.2.2分配了一个名字叫R1fix。

CVS checkout mod #检出最新版本1.4

CVS update –j R1fix m.c #合并所有分支中的变更，即1.2与1.2.2.2.2的合并

CVS commit –m “Included F1fix” #建立新版本1.5

下图就是合并后的Revision历史结构。

在合并过程中可能会发生冲突，可以通过手工地方式解决。

Versions/Tag

我们没有使用 Versions

Version是[标签](http://product.it168.com/list/b/0678_1.shtml)Tag的集合。所谓Tag指的是当项目达到某一个milestone时，对所有文件做一个[标签](http://product.pcpop.com/bq/10734_1.html)，记录历史记录。通常对一个release做一个tag，如下图所示。

类似 老李所说的 excel 管理, 有里程碑一样的修改完成后, 做一个 save as

为更好地理解tag的工作原理，我们可以将 tag 想象成为一条在由文件名和修订号组成的矩阵上穿过的“曲线”，如下图所示：

当我们将该 tag 曲线上的 \* 标记拉直后，你就得到了由所有 tag 修订号组成的如下图的水平线：

由于tag的目的是记录开发历史，因此人们通常不会删除或者改变[标签](http://product.it168.com/list/b/0678_1.shtml)。对tag的删除，移动，重命名多半是因为临时使用[标签](http://product.pcpop.com/bq/10734_1.html)或者不小心放错位置。警告，下列命令会永久删除历史信息，使用时应尽量小心。

cvs rtag -d rel-0-4 tc

Dates/Tags

与Versions一样，Dates也是标签Tag的集合。区别在于前者以版本作为tag的标记，而后者则以时间为标记。相比与Date类型，Version Tag更加常用。

为了更好的全面理解这些核心CVS概念，我们通过Show History视图将这些概念串联起来，如下图所示。

有对了相关概念的理解，接下来，我们将介绍CVS的常用操作。相信大家对日常的操作如update，commit非常熟悉，只做简单介绍。更多地是关注那些容易忽视但是特别使用的命令。

作为三部曲之一的Copy操作，CVS对应的命令为check out。具体操作是，选择HEAD目录下的目标源文件夹，右键单击Check Out，如图：

Check Out

Check out之后的本地工作目录与CVS目录一致。如果想自定义文件夹的根目录，则可选择Check Out As命令。

Switch to another Branch or Version

作为三部曲之二的Modify操作，除正常的源代码变更操作外，对应一个非常重要的Eclipse CVS命令，即Switch to another Branch or Version，即切换到其它分支或者标签做源代码的修改，如图：

Merge

作为三部曲之三的Merge操作，除正常的update/commit外，也对应的一个非常重要的CVS命令merge，即将branch或tag与HEAD中的Base version进行合并，如图

Branch

此外，如果想创建一个新的分支，可通过右键Team -> Branch打开Create a new CVS Branch。

Tag

如果需要发布一个新的版本2.0.6，可以新创建一个Tag。步骤为右键源代码工程 –> Team -> Tag as

Version，如图。通过CVS Repository透视图可以查看相应的Tag。

Patch

补丁(patch)允许开发人员共享尚未提交到CVS的代码变更。补丁程序包含的是本地资源与CVS资源库的差别CVS Diff命令。补丁在许多场合都非常有用：

    · 由于权限的限制，需要将补丁程序发送给权限用户提交。

· 需要一个临时工作区间隔离待提交的代码。

· 在提交CVS之前，需要对更改的文件测试，可以将补丁程序发送给测试人员。

要创建一个补丁，使用右键Team –> Create Patch，开启Create Patch向导完成补丁文件的创建。

这里我们将补丁程序导出成文件patch.txt。按照默认方式完成向导。patch.txt内容如下：

### Eclipse Workspace Patch 1.0

#P source

Index: AsyncGssSocketLayer.cpp

===================================================================

RCS file: /cvsroot/filezilla/FileZilla/source/AsyncGssSocketLayer.cpp,v

retrieving revision 1.14

diff -u -r1.14 AsyncGssSocketLayer.cpp

--- AsyncGssSocketLayer.cpp 4 Mar 2005 10:35:04 -0000 1.14

+++ AsyncGssSocketLayer.cpp 28 Nov 2011 07:26:39 -0000

@@ -1,6 +1,6 @@

// GSSAsyncSocksifiedSocket.cpp: implementation of the CAsyncGssSocketLayer class.

//

-//////////////////////////////////////////////////////////////////////

+//////////////////////////////////////////////////////////////////////

// Part of this code is copyright 2001 Massachusetts Institute of Technology

#include "stdafx.h"

patch.txt实际上是CVS Diff命令的输出，从而实现了CVS Repository之外的资源共享。相应地，开发人员可以共享补丁程序。

Apply Patch命令提供了打补丁的方法，如图所示。

比较&替换

人生没有后悔药，但是Eclipse提供了，而且特别简单。快捷菜单操作Replace With和Compare With提供了本地历史记录与CVS Repository比较与替换的机会。需要说明的是，CVS Repository的资源既可以是HEAD也可以来自某一个分支，或者某一个tag。

重命名

一般情况下，要避免对CVS Repository作重命名操作。对于工程而言，CVS对待重命名后的工程为新工程。相应地，需要使用Team -> Disconnect操作解除项目与CVS的关联，然后再对项目重命名，最后，在重新连接到CVS之后，需要像其它新的工程一样定义。而对于文件而言，重命名会导致出现一个新的文件，不过幸运的是，原文件的内容会被拷贝到新的文件中。

CVS vs SVN

至此，相信读者对Eclipse CVS客户端有一个基本的了解。不过，CVS并非是完美的。下面我们来比较下两者的一些主要的不同点，如下表。

存储类型存取速度事务总体评价

CVS文件相比SVN慢没有实现4

SVN数据库相比CVS快完全实现5

实际上，CVS是一个古老的系统，其内部结构有许多改进之处。知道今天，仍有人想重头开始，重写CVS，但都未能成功，SVN的开发人员花了很多时间与心思在改进内部结构，使其变得简单、优雅，甚至有人认为SVN是CVS的接班人。

常用命令：

cvs -f -n update -dP

该命令并不实际update文件，但是列出来所有并不是up to date的文件

cvs remove file\_name

该命令从cvs 库中一处某个文件，然后还要cvs commit以后才生效。