# JOBBY Installation & Development Notes

# (On HostingRails.com)

# Setting up Database / MySQL Stuff – for a new environment

## Create Backup

1. Create backup from LOCAL Dev environment
   1. MySQL Administrator
   2. “Backup”
   3. New Project
      1. Name it “Backup for HostingRails.com”
   4. Select “testdb1\_development”
   5. Save Project
   6. Execute Backup Now
   7. “C:\Applications\Ruby\projects\jobby\db\backup”
   8. Rename dev database name to hosting database name
   9. Run “search and replace” on the .sql file generated (the backup file)
      1. Find “testdb\_development”
      2. Replace with “jobbyco\_main”
      3. Should be 3 occurrences
2. Create backup from REMOTE Hosted server
   1. “Files” section
   2. “Backups” option
   3. Under “Download a MySQL Database Backup”
   4. Click the appropriate database
      1. ‘jobbyco\_main’ for production
      2. ‘jobbyco\_test’ for test
   5. File will be downloaded as:
      1. Production: ‘main.sql.gz’
      2. Test: ‘test.sql.gz’

## Restore DB into appropriate environment

1. Extract the sql script from the .zip file into same directory:
   1. Production: ‘main.sql’
   2. Test: ‘test.sql’
2. Login to the server
   1. Currently using Putty
   2. Login/password
3. Create rails environment (if the this is for a new environment)
   1. Go to home (/home/jobbyco)
   2. You should see ‘jobby\_prod’ folder
   3. Create a new rails environment with:
      1. Test environment

**rails jobby\_test**

* + 1. This will create all the folders necessary for the application and database to be setup
  1. Go into the new directly:

**cd jobby\_test**

* 1. Copy the DB backup (‘**main.sql**’ for production or ‘**test.sql**’ for test) from the local hard-drive to the ‘**/home/jobbyco/jobby\_test/db**’ folder
  2. Enter the following command to restore the database into this environment:

**mysql --user=jobbyco\_test --pass=tomer66 jobbyco\_test < db/main.sql**

# SVN Setup & Configuration

## Install SVN support for Eclipse - Subclipse

* Download subclipse from Eclipse
  1. “Help” -> “Software Updates”
  2. “Add Site”
  3. Add the following line:

[**http://subclipse.tigris.org/update\_1.6.x**](http://subclipse.tigris.org/update_1.6.x)

* 1. Tick the parent option box “Subclipse 1.6.x Update Site” to ensure all Subclipse components will be installed
  2. Click “Install”
  3. Select to ‘run in background’
  4. A restart of Eclipse may be required
  5. Install Subclipse updates (if required)
  6. Configure Eclipse to connect to the repository using SVN-SSH
     1. Windows -> Properties -> Team -> SVN
     2. At “SVN Interface”, “Client”
     3. Select: "SVNKit (Pure Java)..."

## Create an SVN repository – on HostingRails

1. Create new repository on server
2. Login (using putty)
   1. Jobbyco / tomer66
3. Execute script from site to create an SVN repository
4. Follow instructions at HostingRails.com:
   1. Subversion & Rails - Implement Your Own Version Control System

<http://www.hostingrails.com/Subversion-Rails-Implement-Your-Own-Version-Control-System>

1. Create a linux batch file to create the SVN repository
   1. Appendix A has the contents of the file:

**setupsvn.sh**

1. Copy the file to the root folder

**/home/jobbyco**

1. Allow execute permissions:

**chmod 755 setupsvn.sh**

1. Execute the script from Putty:

**./setupsvn.sh <projectname>**

1. This will create the necessary SVN folders and the following folder structure within the repository:

**…/svn/<projectname>/trunk**

**…/svn/<projectname>/tags**

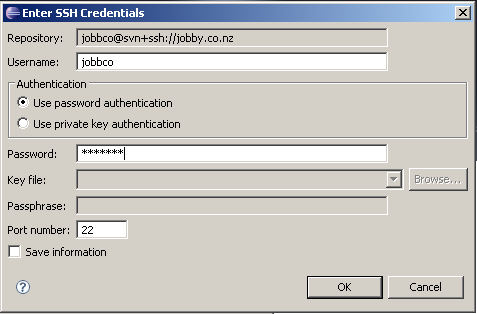
**…/svn/<projectname>/branches**

## Creating a Repository Connection

1. "Open Perspective" from top right nav menu
2. Select "SVN Repository Explorer"
3. Right-click in "SVN Repositories" on the left
4. "New" -> "Repository Location"
5. Enter the following URL:

**svn+ssh://jobbyco@jobby.co.nz/home/jobbyco/svn/flyc**

1. The following screen should appear
2. The **username** should be populated
3. Enter the password: **tomer66**
4. Keep “Use password authentication” selected

****

1. If prompted for 'Author Name'
   1. Click "OK" leaving **jobbyco** and other information as they are

## Creating a new Eclipse project and checking out a project from SVN

These are important links I used to create this tutorial

* Visual representation and great explanation of an SVN branch/tag life-cycle
  1. <http://www.pushok.com/help/svnscc/index.php?redirect=adv_tagsbranches.htm>
* Standard SVN operations (check-in, check-out etc)
  1. <http://agile.csc.ncsu.edu/SEMaterials/tutorials/subclipse/index.html>
* Branching & Tagging
  1. <http://www.saltycrane.com/blog/2007/03/how-to-setup-subclipse-project-to/>

### Create new Eclipse “Rails” project

You could create a new project and ‘checkout’ from SVN using two methods:

* Create a project and then ‘Share’ with SVN
* Create a project using ‘Checkout from SVN’ option

It’s better to create a ‘Rails Project’ and then share with SVN (1st option) as it will setup all the Rails specific functions and capabilities (e.g. app server). So…

1. Go to “RadRails” perspective
2. Right-click on the white space on the left
3. “New” -> “Rails Project”
4. “Project Name”:

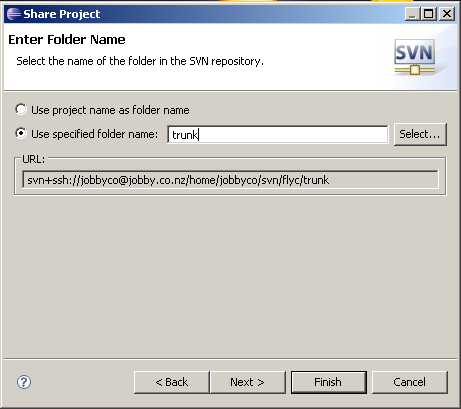
<projectname>

1. Keep “Use default location” ticked as it points to the current workspace
2. Untick “Generate Rails application skeleton” as it will create all the standard rails folders (i.e. ‘apps’, ‘config’ etc. SVN will create all these folders for us
3. Untick “Automatically start server after project is created” as we’ll start it manually when we need to
4. Click “Finish”

### Checking-out from SVN

By selecting an existing project, the project will be checked-out from SVN instead of being checked-in

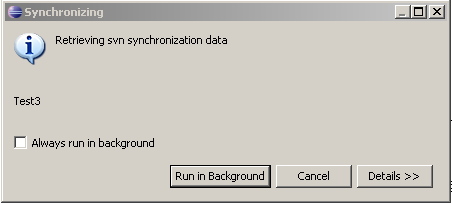
1. From “RadRails” perspective
2. Right-click **<projectname>**
3. “Team” -> “Share Project”
4. “SVN” -> Next
5. Keep “Use existing repository location” selected -> Next
6. Select “Use specified folder name”
7. Click “Select”
8. Select “trunk” folder -> OK
   1. This means that the project will be checked out and not checked in
9. Remove the auto-added project name at the tail end of the ‘folder name’ field and have



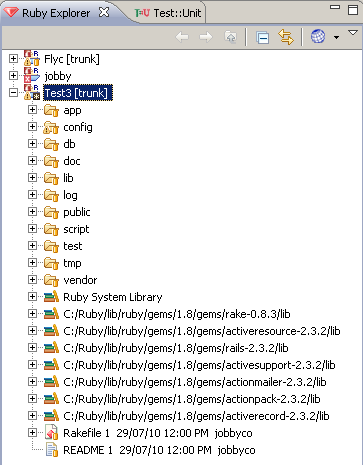
1. Click “Next”
2. Click “Finish”
3. This screen will appear to indicate a check-out will occur instead of a check-in



1. Eclipse will now synchronize the local and remote files



1. Select “No” to change perspective to “Team Synchronization Perspective”
2. Assuming a project “Test3” was created, this is the folder structure that would appear in the left panel:



## Disconnecting a local project from SVN

1. Right click the project (with SVN)
2. Select “Team” -> “Disconnect”
3. Tick the “Delete .svn information if you’re …” to ensure the SVN folders locally are removed to

## Synchronizing and Committing Resources to the Repository

1. Any resources that you have modified will have a black asterisk in front of the resource name, and any added will have a question mark in front of the name.
2. It is always a good idea to see what needs to be done before committing files.
3. To synchronize resources, right click on a project and select

**Team > Synchronize with Repository ...**.

1. You will be switched to the Team Synchronizing view. From there you can look at your outgoing changes, and others incoming changes, and synchronize your code before committing to the repository. Double-clicking on a file will open a side-by-side view of the file and its changes.
2. If a file has a two-way red arrow on it, then there is a conflict. Someone else has changed that file while you were working on it. Double-click the file to get a compare editor, and copy-paste what you need to resolve onto your local copy. When you have the local copy finished, right-click on the file and choose Mark as Merged. Note: PLEASE USE THIS FEATURE RESPONSIBLY!! Think very hard before selecting "Mark as Merged", as it will override what is in the repository. An entire history is kept in case you make a mistake, but always pause and check your work before marking as merged.

## Create a Branch

1. Commit any changes you want in the branch.
2. In the "Navigator" window, right-click your project and select "Team" > "Update"
3. Right-click your project and select "Team" > "Branch/Tag..."
4. In the "Copy (Branch/Tag)" dialog, in the "To URL:" textbox, enter

**svn+ssh://jobbyco@jobby.co.nz/home/jobbyco/svn/flyc/branches/branch2**

1. “Next”
2. Leave the "HEAD revision in the repository" option selected -> Next
3. Enter a comment
4. Ticking the “Switch working copy to new branch/tag” will change the local copy to the branch folder. Leave it unticked for now
5. Click "OK"

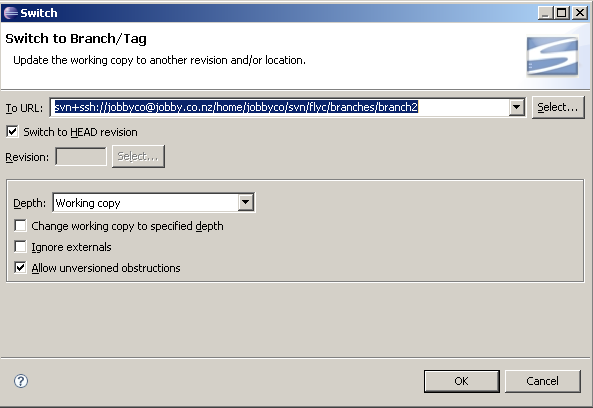
## Switching working copy to a branch

This operation will JUST switch the working copy and won’t change local / remote copies. This will not destroy local copies if changes were made to them. For that see next 2 sections.

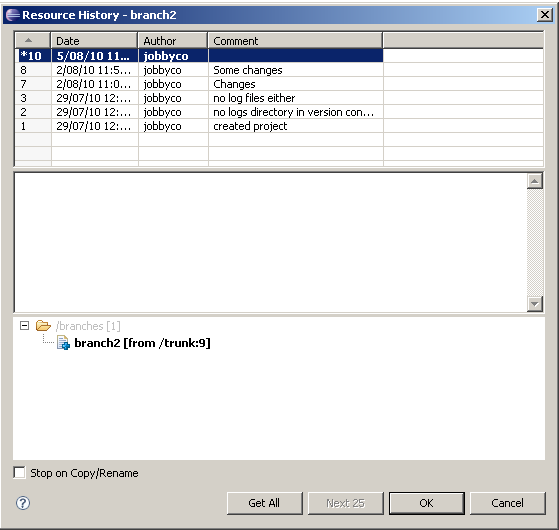
1. Right-click your project and select "Team" > "Switch to another Branch/Tag..."
2. In the "To URL:" textbox, enter:

**svn+ssh://jobbyco@jobby.co.nz/home/jobbyco/svn/flyc/branches/branch2**

1. Make sure the “Switch to HEAD revision” is ticked



1. Click "OK"
2. \* You could select a specific revision from any folder (e.g. ‘trunk’, ‘branch2’ etc) by unticking the “Switch to HEAD revision” and clicking the “Select” button. This screen would appear
   1. Clicking on a specific revision from the list will indicate what files have been changed
   2. If it’s a branch/tag, it will show the correct folder as well



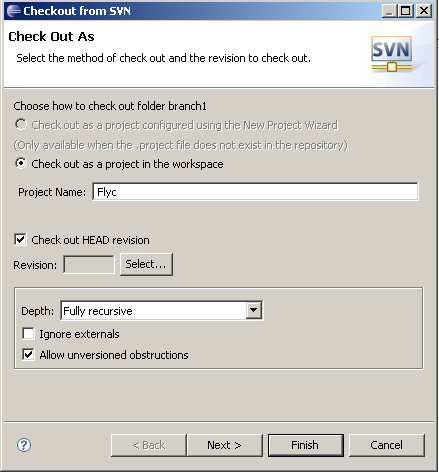
### Replacing local copy with remote SVN copy

1. Right-click project
2. “Replace With” -> “Base Revision”
3. A message will indicate if you’re about to override local changes
4. “OK”
5. The new branch should in the local environment now

### Checking out a project / branch from SVN repository

This will only work on ‘Empty’ / disconnected projects from SVN. It will throw an error if you’re trying to check-out a project which is already ‘attached’ to another SVN repository.

1. Change to “SVN Repository Explorer”
2. Locate the folder / branch you wish to check-out
3. Right-click the branch (as an example) -> “Checkout”
4. Enter an existing project in the workspace (to override an existing project) or define a new project name to create a new local project to hold the selected branch



1. Click “Next”
2. Leave “Use default workspace location” ticked to checkout the project into the project in the current workspace
3. Click “Finish”
4. A confirm message will appear if you’re about to override an existing project in the workspace. Click “OK”

# Setting up a new environment on HostingRails.com

## Creating test environment

### Install Capistrano

Follow instructions at

* <http://www.hostingrails.com/Capistrano-2-x-Deploying-Your-Rails-Application-on-HostingRails-com>
* <http://www.capify.org/index.php/Deploying_to_Multiple_Stages>

1. On the local dev machine run

gem install -y Capistrano

1. Then run

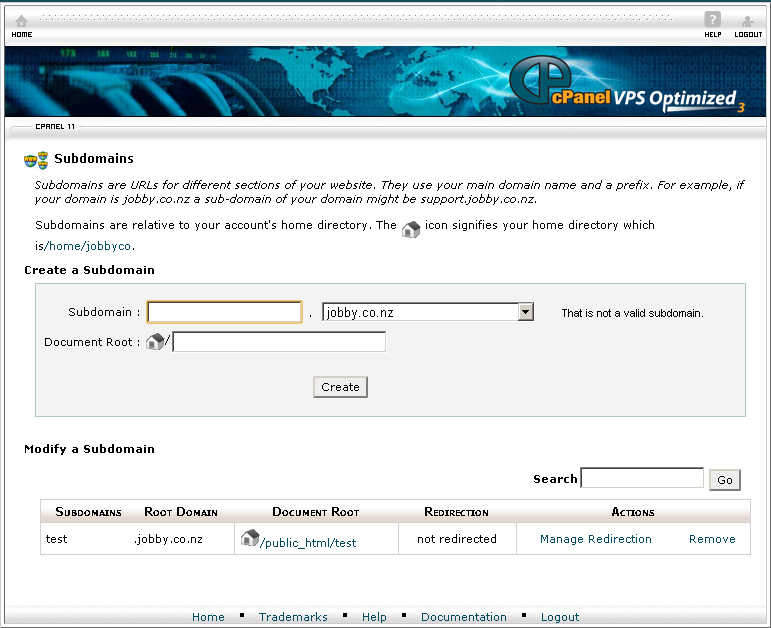
capify .

### Create subdomain

1. Login to cPanel
   1. Jobbyco
   2. tomer66
2. Domains -> Subdomains
3. Add the subdomain ‘test’. The following screen will display:



1. After clicking “Create”, the screen will show



1. Go into Putty
2. “rails jobby\_test” to create new rails application
3. **mv ~/public\_html/test ~/public\_html/test\_backup**
4. Create a symbolic link for the Test environment, the 2nd application
   1. **ln -s ~/jobby\_test/public ~/public\_html/test**
5. Add the following lines

# Lines for 'Test' subdomain

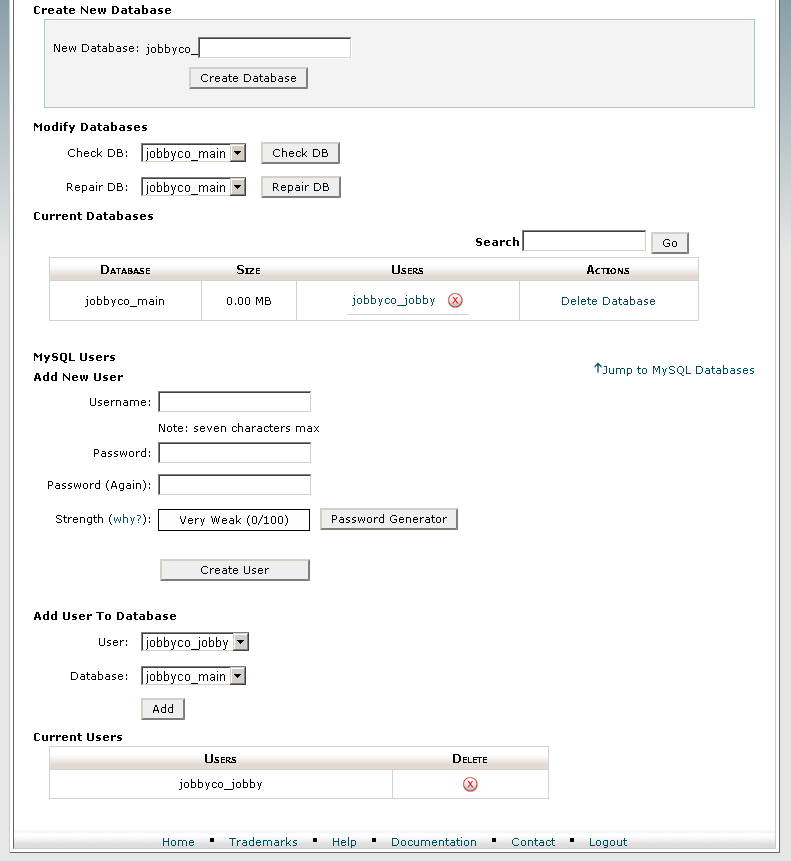
RewriteCond %{REQUEST\_URI} ^/test.\*

RewriteRule .\* - [L]

1. Above the line

RewriteRule ^(.\*)$ dispatch.fcgi [QSA,L]

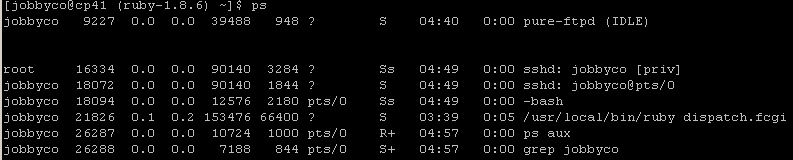
1. Restart the application
2. SVN checkout using Capistrano
3. Copy across the .htaccess file from “jobby\_prod”
4. Copy across the environment.rb file from “jobby\_prod”
   1. Hosting web-site
      1. Login to system
      2. “Databases” section
      3. “MySQL Databases”
      4. Create database
         1. “main” for production
         2. “test” for test / staging
      5. Create user
         1. Production
            * “jobby” => “jobbyco\_jobby”
            * Password “\*0jobby0\*”
         2. Test
            * “test” => “jobbyco\_test”
            * Password “tomer66”
         3. Click “Create User” button
      6. “Add user to database”
         1. Production
            * Select User “jobbyco\_jobby”
            * Select Database “jobbyco\_main”
         2. Test
            * Select User “jobbyco\_test”
            * Select Database “jobbyco\_test”
         3. Select privileges
            * Tick “All Privilages”



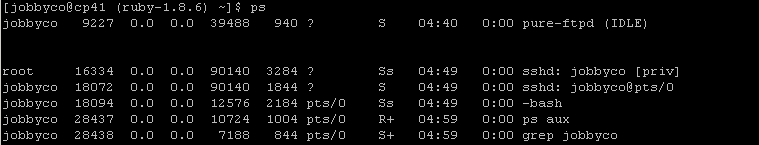
* 1. .htaccess file stuff
     1. Create a temporary rails app on the hosting site (this will generate the following 4 files):
        1. **.htaccess**
        2. **dispatch.fcgi**
        3. **dispatch.rb**
        4. **dispatch.cgi**

# Deploying a new version to an existing site

1. Use Putty
   1. Jobbyco
   2. tomer66
2. in the root directory
   1. /home/jobbyco
3. Type ‘ps’ to see all processes
   1. > ps



1. Kill the running process by typing
   1. > kill 21826
2. Type ‘ps’ again to make sure the process is killed
   1. > ps



1. Backup live application
   1. TBD
2. Backup live database
   1. TBD
3. Make sure email server configurations are set correctly
   1. ‘C:\Applications\Ruby\projects\jobby\_online\config’
   2. ‘environment.rb’
   3. Make sure the plugins are installed correctly

Rails::Initializer.run do |config|

…

config.plugins = 'calendar\_date\_select', 'enum-column', 'fckeditor'

…

end

* 1. Make sure the bottom of the file shows this

ActionMailer::Base.smtp\_settings = {

:address => "localhost",

:domain => "jobby.co.nz"

}

1. Override the routes file with the local ‘online’ version
   1. and make sure the file has the following

map.with\_options(:controller => 'main') do |app|

app.home '', :action => 'home'

end

…

map.root :controller => 'welcome'

1. Make sure the Database configuration is set correctly
   1. ‘database.yml’

# SQLite version 3.x

# gem install sqlite3-ruby (not necessary on OS X Leopard)

development:

adapter: sqlite3

database: db/development.sqlite3

pool: 5

timeout: 5000

# Warning: The database defined as "test" will be erased and

# re-generated from your development database when you run "rake".

# Do not set this db to the same as development or production.

test:

adapter: sqlite3

database: db/test.sqlite3

pool: 5

timeout: 5000

production:

adapter: mysql

database: jobbyco\_main

username: jobbyco\_jobby

password: \*0jobby0\*

pool: 5

timeout: 5000

# Fixing Eclipse not starting after forced restart

* + - 1. Delete all *.manager* and all .??? files from the following folders:
         * org.eclipse.osgi
         * org.eclipse.update
         * org.eclipse.core.runtime

# Using Rails validation without a ‘DB’ dependency

1. **18th Apr 2010**: Classes with ‘Validation’ capability (for easy error handling and message display on screen) but without the db table
   1. Rails 3.0 has ActiveModule which has the functionality needed but I couldn’t get it installed yet
   2. ActiveRecord::BaseWithoutTable seems to be a plug-in that provides this functionality, still looking for it
      1. <http://agilewebdevelopment.com/plugins/activerecord_base_without_table>
2. Installing a plugin
   1. Current direcetory:
      1. C:\Applications\Ruby\projects\jobby\vendor\plugins>
   2. Command line:
      1. ruby ../../**script/plugin install** http://svn.viney.net.nz/things/rails

# Creating an SVN Repository on HostingRails.com

#!/bin/bash  
# derived from <http://www.hostingrails.com/forums/wiki_thread/4>  
echo  
echo "Setup a new SVN repository under ~/svn/<projectname>"  
echo "Usage: $0 <projectname>"  
echo  
  
if [ $# == 0 ]  
then  
exit  
fi  
  
PROJECT=$1  
  
echo "Creating SVN repository"  
mkdir -p ~/svn/$PROJECT  
svnadmin create ~/svn/$PROJECT  
mkdir ~/tmp/$PROJECT  
cd ~/tmp/$PROJECT  
mkdir branches  
mkdir tags  
mkdir trunk  
cd trunk  
  
rails .  
  
echo "Creating initial import of temp project into SVN"  
svn import ~/tmp/$PROJECT file:///home/$USER/svn/$PROJECT -m "created project"  
  
cd ~  
rm -rf ~/tmp/$PROJECT  
  
echo "Checking out a working copy on the server"  
mkdir ~/tmp/workingcopy  
cd ~/tmp/workingcopy  
svn co file:///home/$USER/svn/$PROJECT/trunk $PROJECT  
  
echo "Cleaning up some rails-specific things..."  
cd $PROJECT  
svn remove log/\*  
svn commit -m 'no logs directory in version control please'  
svn propset svn:ignore "\*.log" log/  
svn update log/  
svn commit -m 'no log files either'