

Subclipse Usage: Tips & Tricks

Table of contents

1 How to get started.....	2
2 Checking out a project.....	2
3 Committing code.....	3
4 Tagging a project.....	4

1. How to get started

For a general overview of how code repositories work, the *Team Programming with CVS* section of the Eclipse Help documentation (Workbench User Guide : Concepts : Team Programming with CVS) may be helpful, even for those who will use Subversion rather than CVS as their repository tool. Another source of overview information about repositories is [Eric Sink's Basics chapter](#) of his [Source Control HowTo](#). Both of these sources will require readers to look beyond the tool-specific terminology and focus on concepts. The exhaustive source of Subversion documentation is its manual - this manual is *not* recommended for those new to using code repositories. That said, version-specific on-line Subversion manuals are available at <http://svnbook.red-bean.com>.

Once the basics of what repositories are for and how they work are grasped, the best way to get an overview of how Subclipse works is to read its documentation, available in Eclipse Help once the plugin is installed, or [on-line on the collab.net site](#).

If you are familiar with CVS, you may want to check out differences between CVS and Subversion. The Subversion manual's appendix [Subversion for CVS Users](#) is a good place to start.

The rest of this page will offer simple guidance on a few common, day-to-day tasks a developer will perform. Before going any further, be sure you understand what Trunk, Branch and Tags refer to in a Subversion repository. These concepts are explained in the Subclipse documentation, in the `Tasks : Maintaining Branches` section.

2. Checking out a project

Normally, you will check out code from the *trunk* of a project. Assuming you have already configured your repositories (cf. [Subclipse Setup](#) on this site if you haven't), to use Subclipse to do this:

- open the SVN Repository Perspective
- open the contents of the repository where your project is located
- locate the trunk/ directory in the project
- right-click the trunk/ directory, and choose Checkout from the context-menu

SVN Repository Exploring Dialog - select project trunk

A checkout dialog will present itself, asking for the name of the project you want to use when it is checked out into your workspace. If you use the wizard, you will be offered an opportunity to specify which type of project you are checking out (unless that information is already found in the directory you are checking out); wizard dialogs are not shown in this

HowTo, as they are fairly self explanatory. Whether or not you use the wizard, you almost certainly to **not** want to use the default name presented for the project (it will be "trunk" if you don't use the wizard, and blank if you do), but instead will likely want to use the name of the project whose trunk you are checking out. Modify the dialog appropriately, as shown below (non-wizard example).

SVN Repository Exploring Checkout Dialog - specify name of project in workspace

You may click the `Finish` button to check out into your Eclipse workspace. Should you need for some reason to check out the code into a different location, click the `Next` button to specify.

The code will be checked out of the repository into your workspace (or other location, if you so specified):

Navigator perspective, after successful checkout

3. Committing code

You may commit one file at a time or multiple files at once. The guideline for how to group your commits is whether a single comment adequately describes the changes to a group of files. You are allowed one comment each time you commit to the repository, whether you commit one file or many.

As a rule, you will not commit generated files to the repository. Files created as part of a build process - even if the resultant files are code - are not source code. What can be generated from source code and build scripts should not be committed to a source code repository.

When a file is modified, Subclipse displays an icon next to the file name that signifies with an asterisk that it is different from what was in the repository when the file was checked out. [Note: a new file has a different icon, which shows a small question-mark - see the `.project` file in the example below.] The "changed" state is marked all the way up the directory tree in which a changed file exists.

Updated file prior to commit

To commit a file to the repository, right-click it, and choose `Team : Commit` from the context-menu. You will be presented with a dialog. Note the bottom part of the dialog. If the file has never been committed to the repository, its corresponding checkbox will be blank, and you must specify by checking it that you want to commit the file.

Commit dialog

Note:

You must provide a comment to commit!

Use a brief, informative comment when you commit. Too much detail makes it difficult to read the logs when viewing the file's history. Not enough detail makes the logs unhelpful. Remember that the repository provides tools that allow you to graphically view all the details of changes made to text files (not true for binaries), so there is no need to record the exact changes themselves - only a hint as to the type of changes made, as in the example above.

Updated file after commit

After committing the file, Subclipse shows it as one whose state is unchanged since the most recent synchronization with the repository.

4. Tagging a project

The concept of tagging in a repository is described in the Subclipse documentation, in the [Tasks : Maintaining Branches](#) page. You can also find a brief definition in the section on [Tags](#) in the Subversion manual.

Note:

You may only tag a single point on the directory tree: a single directory or a single file. This is different from CVS, and is a function of the fact that branches and tags in Subversion are implemented by copy operations, and only one source and one destination may be specified (Cf. [svn copy syntax](#) in the Subversion manual).

To tag files and directories in your working directory (workspace), they must first be committed to the repository. Then, right-click on the directory or file that you wish to tag, and choose **Team : Branch/Tag** from the context menu.

select files and/or directories to tag

In the example, only the `src/` directory is going to be tagged. After choosing **Team : Branch/Tag** from the context menu, a tagging dialog will appear:

copy branch/tag dialog

- Specify the location in the repository to which you wish to "copy" the tagged version. In accordance with convention, this is normally in a `tags/` directory that is a sibling of the `trunk/`. Notice in the example that the `tags/` section of the URL is something you need to change, manually, if you are committing from the `trunk/` to the `tags/` directory. The `TO URL:` field will contain a repository path that looks something like

this (first part of URL omitted for brevity):

```
svn+ssh://user@host/path/to/repository/testproj-a/tags/src/v0-1-0
```

- Choose whether you want to tag files from the latest revision in the repository, a specific revision in the repository, or files as they exist in your working copy (which you have already committed). *Most often, you'll want to tag the files as they exist in your working directory, as these are the files you have tested to assure they work in the way you expect.*
- Enter a comment describing the nature of the tag. Depending on your group's conventions, this may be as simple as a version number in a defined format (such as the tag shown in the example below).
- Click the Ok button to complete

copy branch/tag dialog

After the tag is committed, you can see it in the SVN Repository Exploring perspective, and can display the data from the SVN Resource History by right-clicking the resource of interest, and choosing Team : Show in Resource History.