Subversion

CS 490MT/5555, Spring 2016, Yongjie Zheng

About Subversion

- Subversion
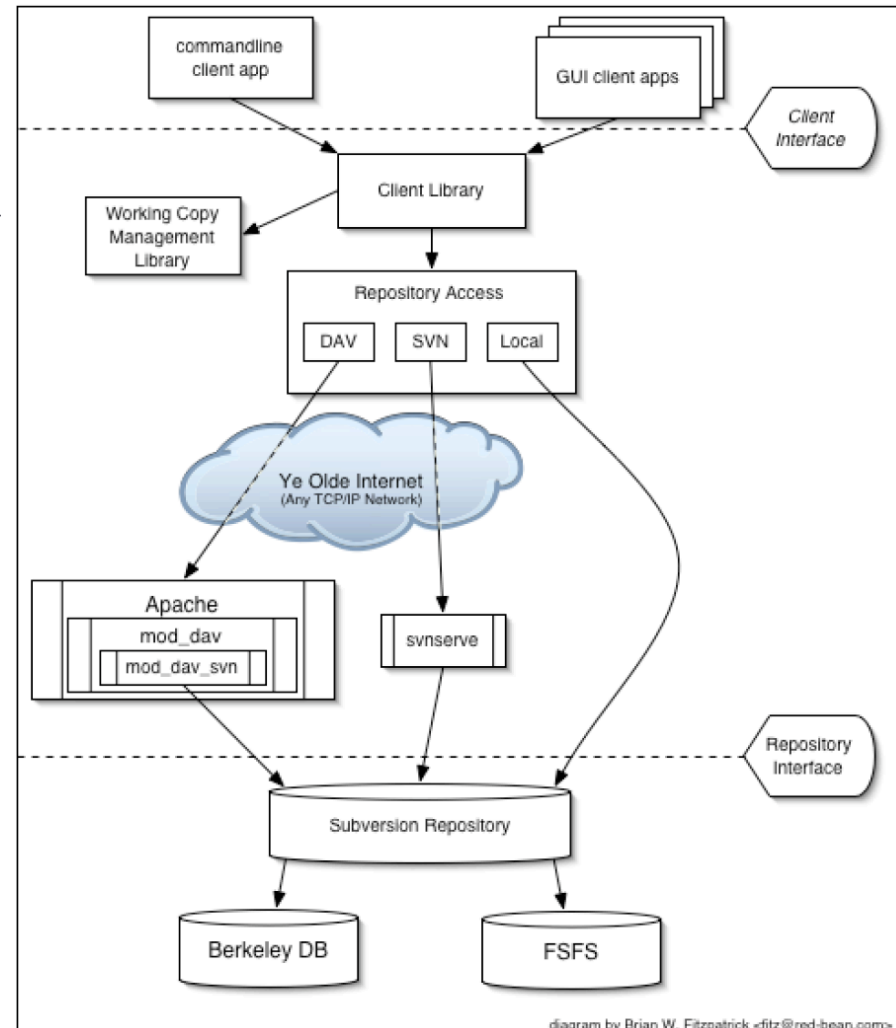
  - A free/open source version control system

  - A typical client-server model

  - Uses the copy-modify-merge strategy

- History

  - Founded in 2000 by CollabNet, Inc. as a successor to CVS

  - Became a top-level Apache project on February 17, 2010

  - http://subversion.apache.org/

CS 490MT/5555 Software Methods and Tools

Subversion's Architecture

▶ **A layered design**

- ▶ Repository Layer
- ▶ Repository Access Layer
- ▶ Client Layer

commandline client app

GUI client apps

Client Interface

Client Library

Working Copy Management Library

Repository Access

DAV   SVN   Local

Ye Olde Internet
(Any TCP/IP Network)

Apache
mod_dav
mod_dav_svn

svnserve

Repository Interface

Subversion Repository

Berkeley DB   FSFS

diagram by Brian W. Fitzpatrick <fitz@red-bean.com>

CS 490MT/5555 Software Methods and Tools

Subversion's Architecture

- ## A modular design
  - Implemented as a collection of libraries written in C
  - Each library has a well-defined purpose and application programming interface (API)
  - SVNKit vs. JavaHL
    - JavaHL is the Java language binding provided by the Subversion project. Based on it, the Subversion libraries can also be used by Java applications.
    - SVNKit is a pure Java implementation of the protocols used by Subversion, and does not use the Subversion libraries.

Using Subversion

- Subversion Repository URLs

- Working Copies

- Revisions

- Repository Administration

- Basic Usage

- Subclipse

Subversion Repository URLs

Schema	Access method
file:///	Direct repository access (on local disk)
http://	Access via WebDAV protocol to Subversion-aware Apache server
https://	Same as http://, but with SSL encryption.
svn://	Access via custom protocol to a svnserve server
svn+ssh://	Same as svn://, but through an SSH tunnel.

- A Subversion URL uses forward slashes on all platforms, including Windows.
  - file:///var/svn/repos
  - file:///C:/var/svn/repos
  - svn://host/svn/repos

Working Copies

▸ A private copy of the project for you

▸ Also contains some extra files, created and maintained by Subversion: the .svn folder

  ▸ Which files contain unpublished changes

  ▸ which files are out of date with respect to others' work

  ▸ The .svn folder only exists in the topmost directory of a working copy from Subversion 1.7. In previous versions, it exists in every versioned subdirectory.

▸ Created by the checkout operation

Revisions

- Each time the repository accepts a commit, this creates a new state of the file system tree, called a *revision.*

- Global Revision Numbers
  - Subversion's revision numbers apply to entire trees, not individual files
  - Revision N represents the state of the repository file system after the Nth commit

- Mixed Revision Working Copies
  - Every commit only increases the revision of the committed file.
  - Your working copy may contain files from several different revisions after several commits.
  - Revisions N and M of a file do not necessarily differ.

CS 490MT/5555 Software Methods and Tools

Repository Administration

- **Choose a Data Store**
  - Berkley DB vs. FSFS
  - Prior to Subversion 1.2, the default was Berkeley DB; the default is now FSFS.

- **Recommended Repository Layout**
  - Three folders: trunk, branches, tags

- **Administrator's Utilities**
  - svnadmin
    - We will be using this one in our assignment
  - svnlook
  - svndumfilter
  - svnsync

Repository Administration

- ### svnadmin Usages

  - svnadmin SUBCOMMAND REPOS_PATH [ARGS & OPTIONS ...]

  - Type 'svnadmin help <subcommand>' for help on a specific subcommand.

- ### Examples

  - svnadmin create /home/tom/svnrepository

  - svnadmin create --fs-type bdb /home/tom/svnrepository

- ### Note: svnadmin is a server-side utility.

  - It is used on the machine where the repository resides.

  - The path argument to svnadmin is a regular filesystem path and not a URL.

CS 490MT/5555 Software Methods and Tools

Basic Usage: svn

- svn: the command-line client program

- General Usages

  - svn <subcommand> <options>

  - The order of the subcommand and the options may vary

  - Subcommands: add, checkout, commit, revert, …

  - Options can have two different forms

    - Long option: two hyphens followed by several letters and hyphens. (E.g. --fs-type, --verbose)

    - Short option: a single hyphen followed by a single letter (E.g. -v).

- Help!

  - svn help <subcommand>

  - Describes the syntax, options, and behavior of subcommand

CS 490MT/5555 Software Methods and Tools

Basic Usage: please note that

- svn subcommands presented in the following slides are some most popular ones.

- It is not an exhaustive list.

- The sequence of presentation follows a basic work cycle of Subversion usage.

- Refer to the svn manual (link is provided in the last slide) if you have question about how to use these subcommands.

- You should be able to use these commands after this class.

Basic Usage

- Getting data into your repository

- Getting a working copy

- Basic work cycle

  - Update your working copy

  - Make your changes

  - Review your changes

  - Fix your mistakes

  - Resolve any conflicts

  - Commit your changes

Getting data into your repository

▸ **Command: svn import**

    ▸ Copy an unversioned tree of files into a repository and create intermediate directories as necessary.

    ▸ Recursively **commit** a copy of PATH to URL.

    ▸ Usage: svn import [PATH] URL

    ▸ Example: svn import myproj http://svn.com/repos

    ▸ To start working, you still need to svn checkout a fresh working copy of the tree.

Getting a working copy

- svn checkout
  - Check out a working copy from a repository.
  - Usage: svn checkout URL[@REV]... [PATH]
  - E.g. svn checkout file:///var/svn/repos/test mine
  - If "PATH" information is not provided, svn checkout will create a working copy in a directory named for the final component of the checkout URL ("test" in the above example).
  - If the local directory you specify does not yet exist, svn checkout will create it for you.

Basic work cycle

- Update your working copy
  - Involved command: **svn update**.

- Make your changes
  - Involved commands: **svn add**, **svn delete**, **svn mkdir**.

- Review your changes
  - Involved commands: **svn status**, **svn diff**.

- Fix your mistakes
  - Involved command: **svn revert**.

- Resolve any conflicts (merge others' changes)
  - Involved commands: **svn update**, **svn resolve**.

- Commit your changes: **svn commit**.

Update your working copy

- It is a good practice to keep in touch with the repository (update often).

- Command: svn update

  - Bring your working copy into sync with the latest version in the repository.

  - Resolves mixed revision working copies.

  - Usage: svn update [PATH...]

  - E.g. svn update

Making your changes

▸ Two kinds of changes: file changes and directory changes.

▸ You must tell Subversion about every directory change that you do by using the provided commands.

▸ Command: svn add

   ▸ Schedule the file or directory to be added to the repository.

   ▸ Will be reflected to the repository in next commit.

   ▸ Usage: svn add PATH... (e.g. svn add testdir)

▸ Command: svn delete (E.g. svn delete testdir)

▸ Command: svn copy (E.g. svn copy testdir testdirCopy)

▸ Command: svn move (E.g. svn move testdir newdir)

▸ Command: svn mkdir (E.g. svn mkdir testdir)

   ▸ Same as: mkdir testdir; svn add testdir

Review your changes

- See an overview of your changes: svn status – one of most popular svn commands
  - svn status –verbose (-v): shows the status of every item in your working copy
    - One column shows the working revision of the item.
    - Another column shows the revision in which the item last changed.

- Examine the details of your local modifications: svn diff
  - The svn diff command displays differences in file content, with each line of text pre-fixed with a single-character code.
    - A space: unchanged.
    - A minus sign: how the line looked **before** the modifications.
    - A plus: how the line looked **after** the modifications..

Fix your mistakes

- **svn revert**
  - Undo all local edits, and revert the file to its pre-modified state by overwriting it with the *pristine* version of the file.
    - The pristine version is the version that the user checked out from the repository without making any changes.
  - Usage: svn revert PATH...
  - E.g. svn revert foo.c

Resolve any conflicts

▸ When you run svn update, you may see conflict information displayed.

  ▸ "Conflict discovered in xxx …"

▸ If conflict exists, you will be given several options to resolve the conflict. Overall, they can be classified as

  ▸ Viewing conflicts

    ▸ Display all changes or all conflicts (df or dc)

  ▸ Resolving conflicts

    ▸ Edit merged file in an editor (e)

    ▸ Accept my version for all conflicts or entire file. (mc or mf)

    ▸ Accept their version for all conflicts or entire file. (tc or tf)

  ▸ Postponing conflicts resolution

    ▸ Mark the conflict to be resolved later (p)

CS 490MT/5555 Software Methods and Tools

Resolve any conflicts

▸ When postponing conflict resolution, three things will happen.

- ▸ Subversion prints a C during the update
- ▸ Subversion places conflict markers into the file to highlight the overlapping areas.
- ▸ For every conflicted file, Subversion creates three temporary files for it
  - ▸ Filename.mine, filename.rOLDREV, filename.rNEWREV

▸ Revolve by hand

▸ After the conflicts are resolved

- ▸ svn resolve –accept working filename
- ▸ As a result, the temporary files will be removed and the file becomes commit-able.

Commit your changes

▸ It is a good practice to explain your commits completely.

▸ Command: svn commit

  ▸ Send changes from your working copy to the repository.

  ▸ Usage: svn commit [PATH...]

  ▸ E.g. svn commit -m "added howto section."

  ▸ If you did not specify the message, Subversion will automatically launch your editor for composing a log message.

CS 490MT/5555 Software Methods and Tools

Additional useful commands

- Command: svn log
  - Shows information about the history of a working copy.
  - You can also examine the history of a single file or directory.
  - svn log foo.c
  - svn log http://foo.com/svn/trunk/code/foo.c

- Command: svn list
  - Shows what files are in a repository directory without actually downloading the files to your local machine.
  - svn list http://svn.example.com/repo/project
  - svn list –v http://svn.example.com/repo/project

CS 490MT/5555 Software Methods and Tools

Subclipse

▸ It is an Eclipse plug-in

▸ It aims to provide all Subversion functionality to the Eclipse development environment.

▸ It is an open source project: http://subclipse.tigris.org/

Install Subclipse

▸ **Follow the instructions here:**
  http://subclipse.tigris.org/servlets/ProjectProcess?pageID=p4wYuA

▸ **Installation Considerations**

  ▸ Update site

    ▸ http://subclipse.tigris.org/update_1.6.x

  ▸ Please, select all the components during the installation:

Install Subclipse - Verification

▸ To see the Subeclipse Perspective: go to Window->Open Perspective->Others->SVN Repository Exploring
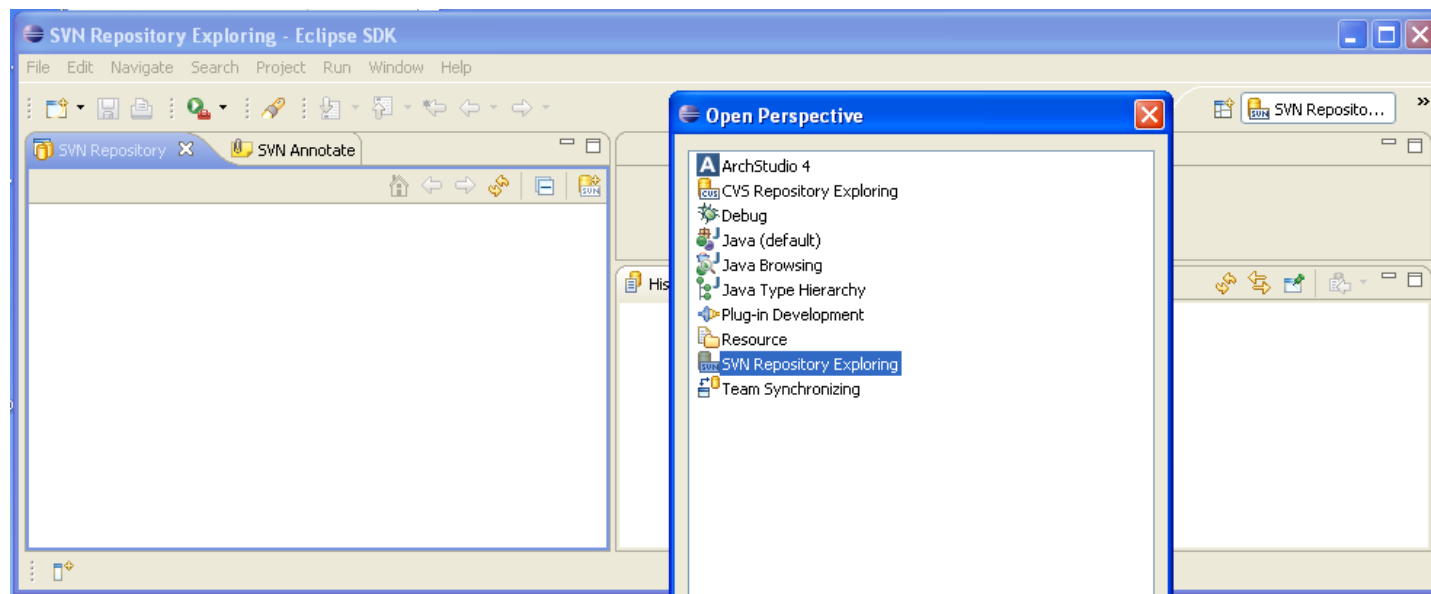
Reference

▸ Please read Chapter 2 of the book "Version Control with Subversion".