Source Management Tool Server Environment Operation

Summary

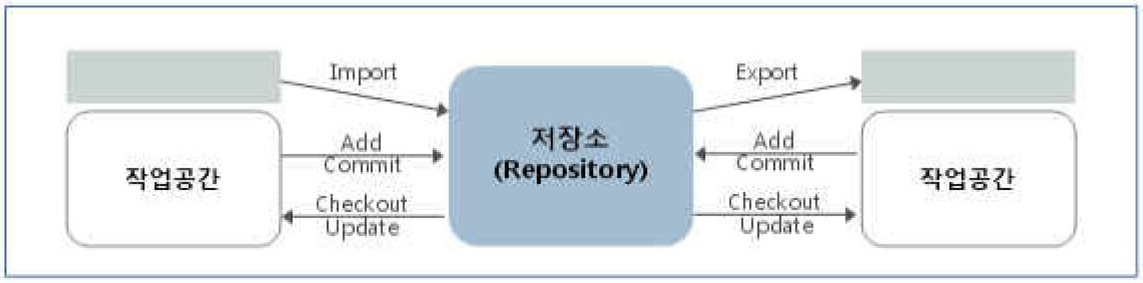
In the e- government standard framework, Subversion, an open source, is used as the source management tool for development environment. Subversion (SVN, version management system) is a system for the management of files used throughout a project.
Subversion allows a history- based version management when multiple developers work for a software and an consistent management of common modules for a team.
As of May 2009, the subject of this manual and the installation version of Subversion included in the e- government standard framework development environment are as follows.

- Server environment: Subversion 1.4.6

The above specified version of Subversion may change in the future due to any improvement made to the continuous improvement efforts on the e- government standard framework development environment.

Description

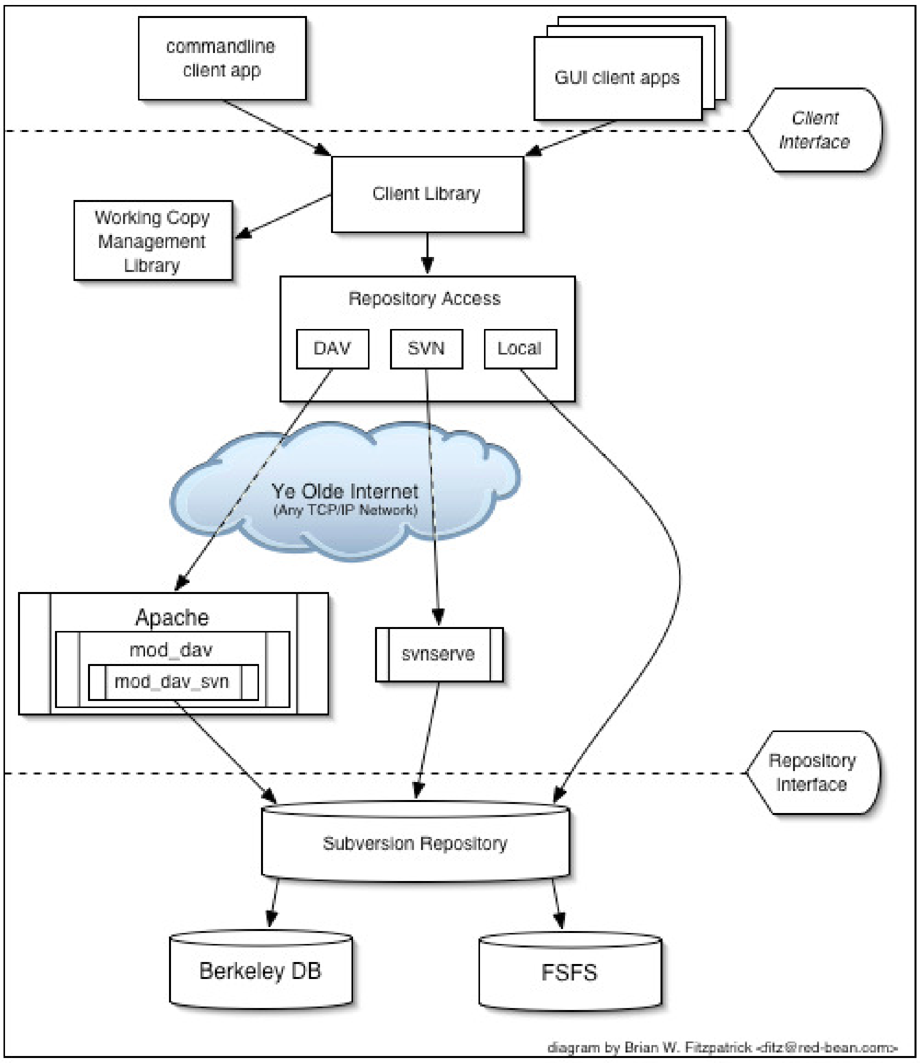
In addition to the actual work directory (workspace), Subversion stores the first data, its update history and change information in the domain called repository.



- Multiple repositories can be created.
- Usable files or directories are imported to the repository
- Running checkout will copy the repository content to the work space and create .svn directory which list management information. This work is needed only once.
- When a file or directory is modified, the changes are registered in the repository. To incorporate other changes existing in the repository to the work directory, run update.
- In order to bring the repository content without management information, run export.

Subversion's Architecture

The repository (Subversion Repository) manages all version data and the client program manages the copied data of the local environment. Various repository access (RA) layers are provided between these two.



Subversion Components

- svn: command line client program
- svnversion: program that manages the status of working copy
- svnlook: tool to inspect the Subversion's repository
- svnadmin: program to adjust and restore Subversion's repository, used by the system administrator
- svndumpfilter: filter program for dump file format data in Subversion's repository
- mod_dav_svn: plug- in module for Apache HTTP server, allowing other users on the network to use the repository
- svnserve: demon or an independent server program run by SSH, another method to use the repository on the network

Manual

Main functions

1. Version management of directory
   - Subversion provides a virtual version management that can check changes to the entire directory tree with time.
   - Version information is also added to the directory.
2. Version history enhanced
   - Subversion can add, delete, copy and rename both files and directories.
   - A newly added file has history that begins newly.
3. Atomic Commit

    - Gathering of change points is whether they are all incorporated in the repository or they are not at all.
    - The developer can produce a bundle of changes logically to carry out Commit, eliminating the problem that only some are incorporated in the repository.
4. Metadata version management
    - The files and directories are the combinations of their associated property keys and values. You can create you own key/value combinations.
    - The properties are version- managed in the way with the files.
5. Network layer selection
    - Subversion has an abstract layer for access in the repository so that it can implement a new network program easily.
    - Subversion is an extension module of the HTTP server and could be used as a plug- in. It is advantageous in terms of reliance and interoperability and is able to use the existing functions (authentication, permission, data compression, etc.) instantly.
    - You can use a standalone Subversion process which is more simple and easier to use. This server can use its own protocol to establish a SSH- based tunnel communication easily.
6. Consistency of data processing
    - Subversion uses the same algorithm to show the changes to the files.
    - It works in the same way to the texts (readable data) and binaries (unreadable data)
7. Production of efficient branches and tags
    - The cost to produce branches and tags is not proportional to the project size. Subversion produces branches and tags by using a method similar to the one known as hard link to just copy the project. Therefore it takes very little time to produce branches and tags.
8. Easiness to scale
    - Subversion is composed of sets of shared libraries written in C and completed with well designed APIs. This means that it is easier to maintain or interface them with other applications or languages.

Main commands

You can use the following commands via the system console to manage software configurations and repositories.

1. Checkout(import)
    - If you check out the repository, a copy of the project is produced in the local computer and this copy contains the latest revision of the designated repository.
    - You can check out your desired version, not the latest version
2. Commit
    - Commit is also called ‘check- in’. This modifies the checked- out sources and add/delete files and then update the changes to the local copy to the repository.
    - If you carry out Commit, the total revision increase by 1 in Subversion.
    - If Commit is carried out, the user can view the latest version of a certain file since the “update”
3. Update
    - Update synchronizes the latest changes to the work copy with the repository and moves the latest work coy to the local drive (changed parts only moved)
    - It is important to carry out Update before making a change to a file.
4. Import
    - This is to put the source into the repository which is empty.
5. Export
    - Unlike checkout, you can receive pure source files that don’t include version management files
    - In an open source project, this is used to compress and release sources.
6. Conflict
    - When some lines in a file are modified or files are updated in the repository by two or more users, a conflict occurs.
7. Revision
    - When a source file or something is modified and committed, the number increases according to a certain rule. This number is the file version of the file stored in the repository.

- Though a single files is modified and committed, the version of all the files increases per project.

8. Trunk
    - This is a central directory for a project and all software development activities are made in the trunk directory.
    - Under the trunk directory, sources files and directories are placed.
9. Branch
    - While developing software in the trunk directory, sometimes you need to come out from it and work in a smaller category. At this time, you can put directories in the branch directory.
    - Branch directories are usually created to test a new function without being interrupted by compiler errors or bugs.
10. Tag
    - Tag directory is a space where sources are separately stored whenever announced in the regular release.
    - Tagging is to label each file regardless of the revision number. This can be applied to the work copy or the repository.
11. Hook
    - Hook is a program run by a repository event such as creation of new revision or modification of a property that is not version- managed.
    - Hook informs what the event is, with what objective it works and who the user is that ran the event.
12. Lock
    - This is a device for the user to demand an exclusive right to modify the work copy file.
13. Merge
    - Merge is to combine changes to a branch with another branch to incorporate them to the trunk (and vice versa)
    - At this time, another branch could be a trunk

Sever Operation Management

1. Producing a repository
   It is recommended Subversion repository be a multi- repository system for easy addition and management of repositories in the future.
    - Create a directory for Root Repository.

      ```
      > mkdir repository
      ```

    - Create repositories.

      ```
      > cd repository
      > svnadmin create --fs-type fsfs repo1
      > svnadmin create --fs-type fsfs repo2
      ```

2. Authentication/permission setup (conf/svnserve.conf)
    - In the svnserve.conf file in the repository conf folder (or directory), set up the permissions for the repositories
    - If not set up for anon- access (anonymous user) and auth- access (authenticated user), the anonymous user is granted the permission to read and the authenticated user is granted the permission to write by default.
    - The following is an example of svnserve.conf.

      ```
      [general]
      ### These options control access to the repository for unauthenticated
      ### and authenticated users.  Valid values are "write", "read",
      ### and "none".  The sample settings below are the defaults.
      # anon-access = read
      # auth-access = write
      ```

```
# The anonymous user is not allowed to access, but the authenticated user is granted the
#permission to write.
anon-access = none
auth-access = write
### The password-db option controls the location of the password
### database file.  Unless you specify a path starting with a /,
### the file's location is relative to the conf directory.
### Uncomment the line below to use the default password file.
# As the password file, the conf/passwd is used by default.
# password-db = passwd
### The authz-db option controls the location of the authorization
### rules for path-based access control.  Unless you specify a path
### starting with a /, the file's location is relative to the conf
### directory.  If you don't specify an authz-db, no path-based access
### control is done.
### Uncomment the line below to use the default authorization file.
# If permissions are not controlled for directories and user groups,
# no additional permission is set up.
# authz-db = authz
### This option specifies the authentication realm of the repository.
### If two repositories have the same authentication realm, they should
### have the same password database, and vice versa.  The default realm
### is repository's uuid.
# Designate the title that will appear when authenticating this repository.
realm = eGovFrame SVN Repository
```

3. Creating a repository
   Register users in the passwd file in each repository conf folder or directory. The format is 'id = password' and the password is a plaint text type.

```
### This file is an example password file for svnserve.
### Its format is similar to that of svnserve.conf. As shown in the
### example below it contains one section labelled [users].
### The name and password for each user follow, one account per line.
[users]
# harry = harryssecret
# sally = sallyssecret
developer01 = 123qwe
developer02 = 123qwe
```

4. Backup and recovery
   - Dump : Use the standard I/O to create a file from the content of repository. The svnadmin dump command is used. This command should be used outside the directory.

```
> ls
 repository
> svnadmin dump repository > repository.dump
```

   - Load: Use the reposiotry backup file to recover the repository. The svnadmin load command is used. Create an empty repository and use the backup file.

```
> svnadmin create repository
> ls
 repository   repository.dump
> svnadmin load repository < sample.dump
```

References

Subversion open source project site: http://subversion.tigris.org/