Table of Contents

Setting Up Your Software Project

Software Tools and External Packages available

At CERN there are a set of software development tools available and centrally available, installed and managed.

For more information look at the following web sites:

- SDT service for commercial Software Development Tools for all phases of software development
- LCG External Software Service for the most commonly used open source packages used in HEP
- LCG Software Download Service to retrieve all the version of the packages developes in the LCG Applications Area.
- Linux software Repository for all Linux packages that can be downloaded on your desktop machine. Also groupe by category.
- Windows Applications Database for the application available on the WIndows platform at CERN.

The Savannah bug tracker

The Savannah portal offers facilities for development, distribution and maintenance of software projects. Savannah is an open source derivative of SourceForge. The main features available are a set of trackers that allow the managament of bugs, tasks, patche, etc.

In order to use the Savannah service register yourself and then register your project.

Access to CERN CVS servers

General documentation

The LCG Applications Area CVS service is now based on the IT CVS service. All LCG project previously hosted by the SPI CVS service were migrated to the centrally-managed IT CVS service. All the details are available on the LCG IT CVS web site.

Everything about CVS at CERN (including the summary reported on this very page) can be found in the CERN CVS documentation. Refer to that page for more precise and complete informations.

A few useful CVS commands

- Checking out (downloading) the last revision (last version) of a module:
  ```
  $ cvs checkout < modulename >
  ```
- Checking out a given revision of a module:
  ```
  $ cvs checkout -r < revision > < modulename >
  ```
- Updating a module:
  ```
  $ cvs update -PAd < modulename >
  ```
  The `PAd` option (refers to man cvs for exact info) ensure that everything appends correctly on a recursive checkout (creates new dirs, remove or ignore empty ones).
    - Updating a single file:
      ```
      $ cvs update < filename >
      ```

    - Updating a directory (can be " `.` "):
      `$ cvs update < directory >`
- Tagging a release:
  `$ cvs tag < tagname >`
- Adding a file (Note: added files have to be commited, see below):
  `$ cvs add < filename >`
- Removing a file (Note: added files have to be commited, see below):
  `$ cvs remove < filename >`
- Commiting (registering changes in the CVS database) a file:
  `$ cvs commit -m "my comment" < filename >`
- Differences between revisions:
  `$ cvs diff < filename >`
    - Recursive diff through a directory:
      `cvs diff < directory >`
    - Diff between two explicits revisons:
      `$ cvs diff -r < v1 > -r < v2 >` (ommiting `< v2 >` will diff local file content against `< v1 >`)
    - Diff against a date:
      `$ cvs diff -D "< date >" < filename >` (date can be `"2005-03-01"`, `"1 month ago"`, ...)
- Developpers logs:
  `$ cvs rdiff < filename >`
- Checking freshness:
  `$ cvs release < modulename >`
  This print the list of non up to date files compared with the CVS repository.

-- AlbertoAimar - 05 Aug 2005

This topic: LCGAAWorkbook > SettingUpSwProject
Topic revision: r1 - 2005-09-06 - AlbertoAimar