Working with CVS using Eclipse

Hoang Huu Hanh

(11.10.2004)
Contents

• Introduction: CVS basics
• Setting up
• Import/Checkout a project
• Team working
CVS Basics

• CVS, stands for Concurrent Version System, is the leading open source versioning system.
• CVS manages files by noting changes in them.
• The information is saved in a so-called repository (directory tree on the hard drive, managed by the CVS server).
• The data is always put in a branch of the repository (the main branch is named HEAD).
• Each change will be given a unique revision number. New files start with 1.1.
• A version is a set of files with certain revision numbers. We can create a version if the resources in your project have a state that seems worthy to be saved explicitly.
Setting Up

• Prepare for CVS connection
  – Location
    • Host: nemesis ifs tuwien ac at
    • Repository path:
      /system/apps/cvs/CVSROOT[/semanticlife]
  – Authentication
    • User: <your account at IFS>
    • Password: ******
  – Connection
    • Connection type: extssh
    • Port (use default port: 22)
Setting Up –
Create a New CVS Repository

  You could choose CVS Repository Exploring command in the dialog appearing as Screenshot

  See ‘CVS Repositories’ perspective

- Right click on this perspective to bring the short-cut (context) menu out, then choose New – Repository Location (SS)

- After that, enter parameters for the new repository location in following dialog (SS)
Problem in Connection (?)

- If you have any problem when accessing to CVS server, please do as follow to overcome:
  - Access to your account on nemesis ifs tuwien ac at (or thor intra at)
  - Open the file ".bashrc", the content of file looks like as below:
    ```
    MAIL=/usr/mail/${LOGNAME:?}
    EDITOR=vi
    export PATH=${PATH:/system/apps/cvs/bin
    export CVSROOT=/system/apps/cvs/CVSROOT
    ```
  - Then append following statement into that file
    ```
    export CVSROOT=/system/apps/cvs/CVSROOT
    ```
  - Save and try to access the CVS sever again.
Putting a Project into a CVS Repository

• In Eclipse, open an existing project or create a new project
• Choose the active project, and select ‘Share Project…’ command under ‘Project’ menu or ‘Team’ option in shortcut menu
  Screenshot
• If you did not create any CVS repository, you would be asked to create a CVS repository location. SS
• Otherwise, you can choose an existing CVS repository location created in Eclipse before. SS
Putting… (2/5)

• Enter Module project on CVS:
  – Use project name as module name
  – Enter a new name for the module
  – Choose an existing module in the CVS Repository
Putting… (3/5)

- Review and commit the project resources:
  - Select or deselect some resources to commit to CVS repository
Putting… (4/5)

- After pressing the “Finish” button, Eclipse will ask you to commit all new changes in the project into the CVS repository.
  
  ![Commit Changes dialog]

- If you choose “Yes”, you will be prompted to give a ‘commit comment’; after that, changes will be committed to the CVS repository and a version control will be added.

- If you choose “No”, you can commit changes later using the ‘Commit…’ command under the ‘Team’ option.

- In addition, you can add resources of the projects to Version Control by choosing the command ‘Add to Version Control’ in the context menu ‘Team’.
• Finish
Check Out a Project from a CVS Repository

• In Eclipse, choose to create new project
• Select wizard “Check out Projects from CVS” under option “CVS”. (SS)
  This wizard help to create a new project by checking out an existing project from a CVS repository
• After that, we choose or create a CVS repository location to check the project out. (SS)
• Select the way to checked out a project:
  – Enter name of the module
  – Browse the CVS repository to choose the module (recommended) (SS)
Checkout...

- Select the method to check out (SS)
  - As a new project
  - Into an existing project
- And project location in local storage. (SS)
- Finally, choose a branch (HEAD is recommended) to start checking out.
Team Working

• Synchronizing: *(SS)*
  – *Synchronize with Repository* feature helps developers to synchronize theirs local resource with the base resource on CVS repository and remote resource variants as well (Three-way comparison).
  – This type of comparison involves the local resource, a remote resource variant and a base resource variant.
    • Outgoing Change/Deletion/Addition
    • Incoming Change/Deletion/Addition
    • Conflicting Change/Deletion/Addition
  – The base resource variant represents a common ancestor for the local and remote resources
Team Working (2/7)

Three types of events

• Outgoing events: there are changes in the local resource
• Incoming events: there are changes in the base resource
• Conflicting events: there are change in the local resource and the remote resource variants
Team Working (3/7)

• Seeing Differences
  – To see the differences between local resources and CVS repository, you just double click on the resource in Synchronize perspective and see the differences.

  See SS
  – You also can see the differences by choosing command ‘Compare with – Latest from HEAD | Another Branch or Version…’
Team Working (4/7)

• Commit changes
  – After synchronizing, Signs “>” appeared in front of some resources say that they should be committed to the CVS repository (SS)
  – Run command “Commit...” under “Team” to commit changes to the CVS repository.

• Update
  – update values of tags which could be chosen.

• Restore from Repository
  – to restore deleted files from CVS repository

• Disconnect
  – terminate CVS working mode.
Team Working (5/7)

- Revisions
  - Revisions are created when you commit changes with comments to CVS repository
  - Revisions Viewing
    - run command Team – Show in Resource History in the resources context menu
    - The view ‘CVS Resource History’ shows up. It lists the revisions of the chosen resource (SS)
- Replacing the current vision with another one
  - Choose ‘Get Sticky Revision’ of the revision that will replace the current revision.
  - A warning notice would appear, you choose to press OK to replace the current revision with chosen revision.
Team Working (6/7)

• Versioning
  – Versioning a project
    • Choose the project you want to version in ‘CVS Repositories’ view from HEAD tag (SS)
    • Choose command Tag as Version of the project context menu, then you would be prompted to enter a version tag, e.g. “Alpha1” as follow, and press OK

• You also can version projects from the workspace by running command Tag as version… in the ‘Team’ context menu of a chosen project
Team Working (7/7)

• Versioning (cont’d)
  – Checking out a Version
    • Select the version you want to load from CVS repository in ‘CVS Repositories’ perspective. (CVS Repositories: Versions - <project name> - <version name>)
    • Bring the context menu out; choose command ‘Check Out As…’
    • A dialog appears to prompt you enter a new name for the checkout project.
    • You can choose one of options of naming for the project and follow the guide of the wizard in order to complete of checking out a project from a version in CVS repository.
  – Working with Branches
    • Whenever you create a branch, a version is also created.
    • If a developer wants to share his changes, there is possibly a conflict with changes of another developer. It is the responsibility of the developer seeing the conflict to merge the changes and then try again
    • The merge will be done locally
That’s all. Thank you!
Add a new CVS Repository

Add a new CVS Repository to the CVS Repositories view

- **Host:** nemesis.ifs.tuwien.ac.at
- **Repository path:** /system/apps/cvs/CVSRoot

**Authentication**
- **User:** hhhanh
- **Password:** ************

**Connection**
- **Connection type:** extssh
- **Use Default Port**

**Options**
- **Validate Connection on Finish**
- **Save Password**

⚠️ Saved passwords are stored on your computer in a file that's difficult, but not impossible, for an intruder to read.

[Finish] [Cancel]
```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```
Create a new project by checking out an existing project from a CVS repository.

Wizards:

- Java Project
- Plug-in Project
- CVS
  - Checkout Projects from CVS
- Java
- Plug-in Development
  - Feature Patch
  - Feature Project
  - Fragment Project
  - Plug-in Project
  - Update Site Project
- Simple
Checkout Project from CVS Repository

Select an existing repository location or create a new location.

This wizard allows you to checkout projects from a CVS Repository.

- Create a new repository location
- Use existing repository location:

extssh:hhhanh@nemesis ifs.tuwien.ac.at:/system/apps/cvs/CVSROOT/semantclife
Select Module
Select the module to be checked out from CVS

- Use specified module name: HelloWorld
- Use an existing module (this will allow you to browse the modules in the repository)

Select Module
Select the module to be checked out from CVS

- Use specified module name: HelloWorld
- Use an existing module (this will allow you to browse the modules in the repository)

Directory Tree:
- CVSROOT
- HelloWorld
- RDQL
Choose how to check out folder 'HelloWorld'
- [ ] Check out as a project configured using the New Project Wizard
  (Only available when the .project file does not exist in the repository)
- [x] Check out as a project in the workspace

Project Name: HelloWorldnew

- [ ] Check out into an existing project

Select the project location

- [x] Use default workspace location

Location: D:\Eclipse3\workspace\HelloWorldnew
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

    int i = 12;
    while (i < 70) {
        System.out.print("Value of i = ");
        System.out.print(i);
        System.out.println("It will stop after");
        i++;
    }
}

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
        int i = 12;
        while (i < 70) {
            System.out.print("Value of i = ");
            System.out.print(i);
            System.out.println("It will stop after");
            i++;
        }
    }
}
The currently used revision is marked with "*"