Using CVS Repositories with SAS® webAF™ 3.0

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *Using CVS Repositories with SAS® webAF™ 3.0*. Cary, NC: SAS Institute Inc.

Using CVS Repositories with SAS® webAF™ 3.0

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, August 2004

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Abstract

Team-based development, collaboration, version control and related features are very important in software development.

Source Code Control (SCC) and the Concurrent Versioning System (CVS) are the most popular source code control systems available on the market. Many SAS customers asked us to support these tools and, starting with SAS® AppDev Studio™ 3.0, webAF™ software supports Microsoft SCC.

CVS is becoming more and more important, especially in the Java community, and even though webAF software doesn't support CVS natively at this point in time, there are tools available that bridge between SCC and CVS. One of those tools is the CVS SCC proxy plug-in from PushOK Software. This document will introduce you to CVS and how webAF 3.0 makes use of CVS via the PushOK bridge.

SAS webAF 3.0, SCC, and CVS

Starting with SAS AppDev Studio 3.0, webAF software supports the Microsoft Source Code Control (SCC) interface.

Microsoft development tools (starting with Visual C++ 4.0) use the "Common Source Code Control" interface to talk to source code control. Other implementations are Microsoft Visual Studio, Powerbuilder, and so on.

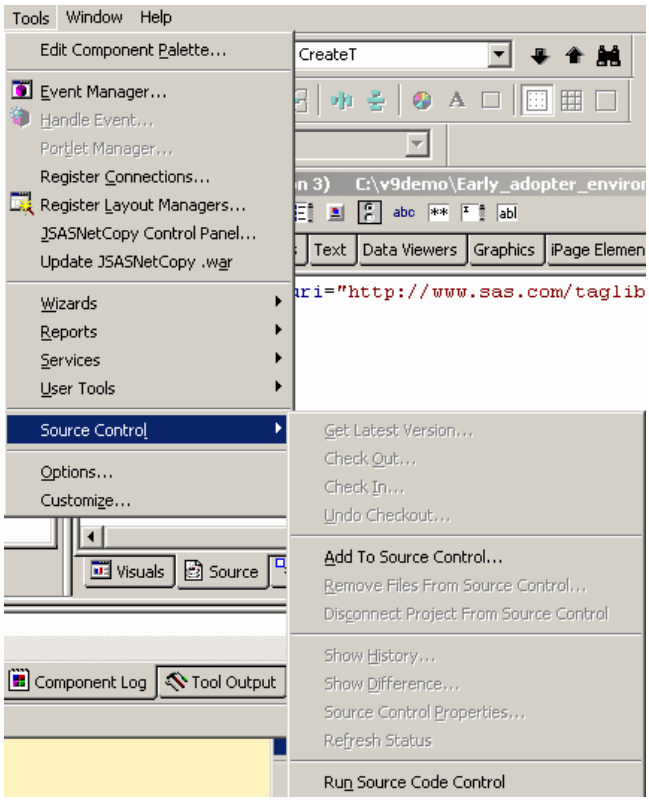
SCC is one of the control versioning systems. All SCC systems can be divided into three groups:

- SourceSafe from Microsoft (Used from Integrated Development Environments [IDEs], such as Microsoft Visual Basic and .NET)
- Serious commercial systems
- CVS.

All of these systems provide the possibility for *team-based application programming*, which enables the following:

- Support for multiple users developing within the same project
- Support for versioning
- Change management.

Through the SCC interface you can access version control tasks directly from the webAF environment. In this screenshot you can see how SCC is integrated in SAS AppDev Studio 3.0.


Tools
Window
Help
Edit Component Palette...
Event Manager...
Handle Event...
Portlet Manager...
Register Connections...
Register Layout Managers...
JSASNetCopy Control Panel...
Update JSASNetCopy .war
Wizards
Reports
Services
User Tools
Source Control
Options...
Customize...
CreateT
Text
Data Viewers
Graphics
Visuals
Source
Component Log
Tool Output
Get Latest Version...
Check Out...
Check In...
Undo Checkout...
Add To Source Control...
Remove Files From Source Control...
Disconnect Project From Source Control
Show History...
Show Difference...
Source Control Properties...
Refresh Status
Run Source Code Control

CVS (Concurrent Versioning System)

CVS is a very popular, low-cost source code control system that has been around for many years. Our users want CVS support from within SAS webAF 3.0, because CVS contains many functions for code merging, branching, notifications and other, and is almost free!

Normally CVS is installed on a server. The idea is to store the latest source code on the server and not modify it directly. Each developer gets their own copy instead.

When a developer wants to modify a file, a checkout is performed which informs others that the file is being modified, and when the modifications are done, the developer commits or checks the file back in to the server.

The server resolves conflicts automatically at the line level, such as when two users modify the same file. An example would be if one developer modifies lines near the top of the source and another developer modifies lines near the bottom, then in this case the two files will be automatically merged together.

If two developers modify the same lines, then the last user who commits the changes will have to resolve the conflict manually.

Apart from this main functionality, the source control system also provides many other interesting and useful functions such as a visual comparison of versions, commit and rollback, notifications, tags, branches, history, locks, and others.

Due to these functions the use of the source control is very useful for single developer work, too, because it is not necessary to backup different versions of a source code.

CVS is a command line tool ported from UNIX, and because using it from command line is not very convenient, a graphical user interface was created. The most popular interface is WinCVS.

This is the most powerful tool to work with CVS repositories from Windows. It covers practically all the functionality of CVS, but it is not integrated in an IDE.

PushOK CVS SCC Proxy Plug-in

The PushOK CVS SCC proxy plug-in is software that bridges Microsoft's SCC interface and the CVS SCC command interface.

Using this plug-in along with SAS webAF software's support for the SCC interface enables webAF users to control CVS source code control repositories.

Download and Installation

The software can be downloaded from here: http://www.pushok.com/soft_short_info.php

You have to register to get a generated key for a 30-day trial version. The full version is available at a low cost. The installation is typical for a Windows application, and after the installation there is no further configuration required.

After the installation you will find the files in this directory:

```
C:\Program Files\PushOk Software\CVSSC
```

Configuration

Start the "configure plugin" from **`C:\Program Files\PushOk Software\CVSSC`** to configure the plugin.

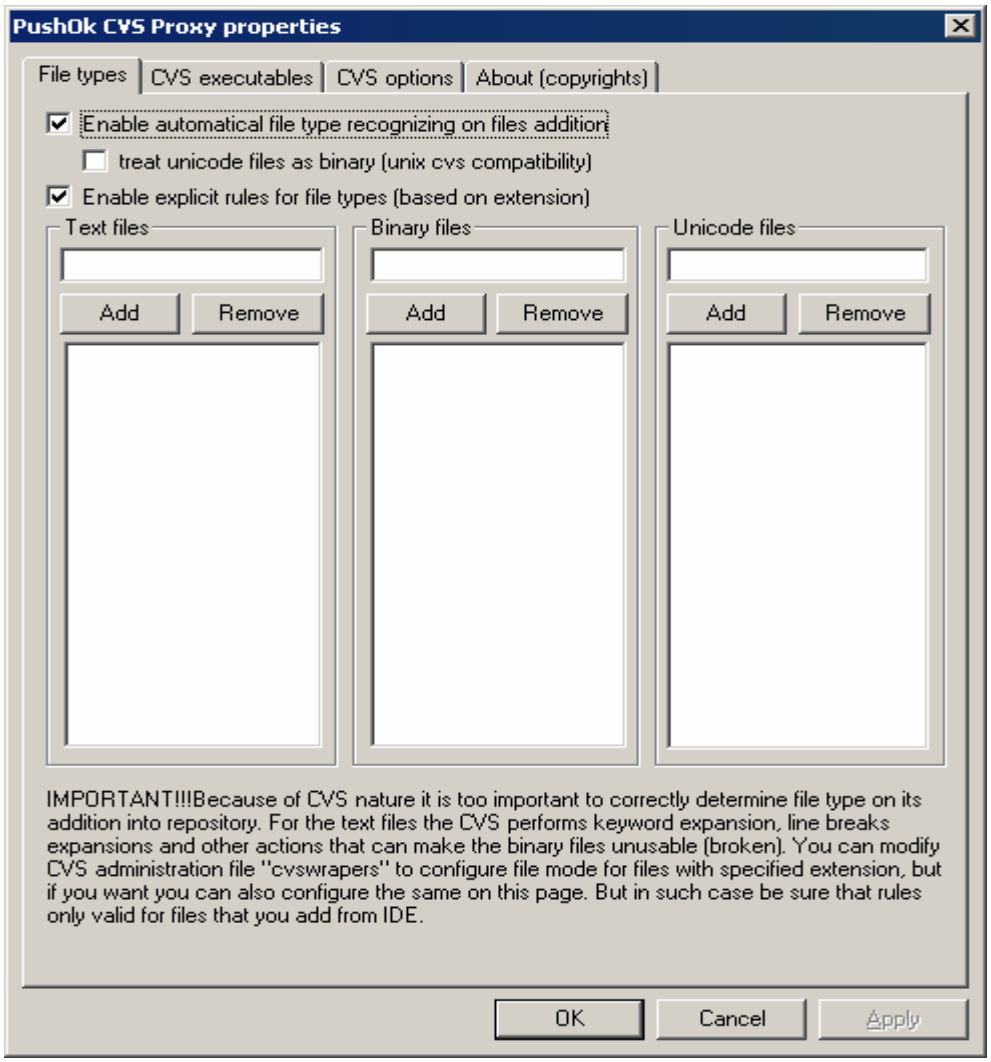
The File types tab enables you to customize the plug-in behaviors when adding files (and only) in CVS.

Normally there are no changes necessary. However, some interesting settings can be:

Option: Enable automatic file type recognition on file addition:
The option is on by default, and lets the plug-in test the file content automatically in order to define its type. This option solves such problems with 95% success.

Option: Treat Unicode files as binary (UNIX CVS compatibility):
Only CVSNT servers support the processing of UNICODE files; therefore, if you are using a UNIX server you should turn this option on.

Option: Enable explicit rules for file types (based on extension):
If automatic type definition does not work well, then you can obviously specify the file types by adding the relevant extension to one of three groups.



The **CVS executables** tab enables you to customize the application that will be started by the "Start Source control" command from the IDE.

For each task there is a default built-in tool available. But sometimes external tools are more powerful.

The **CVS options** tab enables you to specify the plug-in behavior. Normally there are no changes necessary. For further information about these options see the PushOk CVS SCC plug-in documentation: http://www.pushok.com/help/cvsscc/index.php.

Using the PushOK CVS SCC Proxy Plugin within webAF 3.0

The following section will demonstrate the use of the PushOk CVS SCC proxy plug-in within webAF 3.0.

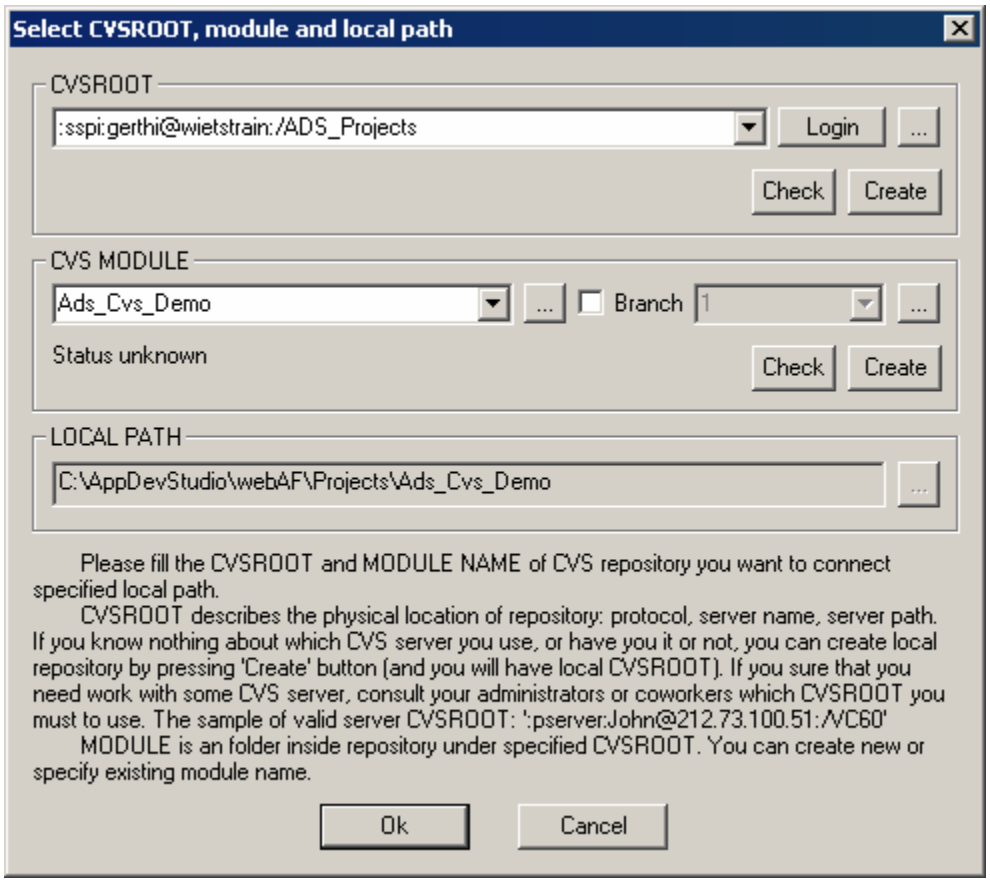
The assumption is that a new development project has started. The first developer has just generated a SAS AppDev Studio Project with some rudimentary classes and interfaces.

It is assumed that you have already installed a CVS Server and you have a user ID and the rights to create a new CVS Module. The CVS Module in this case will be equivalent to a SAS AppDev Studio Project.

After installing the CVS SCC Proxy, you can add a project to the Source Control and make it available for other developers by selecting **Tools > Source Control > Add to Source Control**.

If this is the first time you are using the CVS SCC plug-in, then you will be asked to register your version of the software.

Add to Source Control



To add your project to the Source Control you first need to connect to the CVS Server. Therefore, you need to specify CVSROOT as a repository location and the type of connection. You must

enter this data manually the first time you connect to the CVS Server. Later you can reuse this data to reconnect.

The CVS repository is data storage under control. Mostly the repository is located on a server. The repository itself is a usual folder on the disk having a specific structure and storing special sorts of files. All the files have a suffix of .v.

The repository always has a folder named CVSROOT containing basic administrative files necessary for CVS work. In addition, the repository may contain other folders containing user data.

To be sure that you entered the correct data, click the **Check** button under the input field of CVSROOT. Then you need to specify a module that is a folder inside the repository, where you will add your project. Usually the project needs to have a nonexistent module; therefore, you should create it by clicking the **Create** button.

By default SAS AppDev Studio gives the project name as a module, so more often than not there is nothing to change. In order to avoid errors click the **Check** button before and after you create the module. This ensures that the module did not exist first, and that you have created it.

CVSROOT

CVSROOT is a specific line with special format, specifying the address of the repository which you connect to.

CVSROOT has the following general format:
[[:*protocol*:][[*user*][:*password*]]][@*servername*][:*serverpath*]

Example:

:sspi:gerthi@wietstrain://ADS_Projects

You can get this information from the CVS server administrator.

:*protocol*:

is a type of connection to CVS. It must be enclosed by ":" and can contain the following values:

- :local: indicates that a repository is located in local or network _file_ system. This variant of access is not recommended because this way CVS degrades to client-file system (like SourceSafe). There can be problems if too many users try to use it.
- :pserver: is the most common access protocol according to client-server scheme. This is the easiest and the fastest one. A drawback is that it transfers all the data in plain text. If you need to secure your source code and user passwords then do not use this protocol over public networks.
- :ssh: is the access protocol using SSH (Secure Shell).
- :sspi: is the access protocol for Windows server with support for data encoding.

There are another access protocols also, but their use is extremely specific and as such they are seldom used. See the CVS documentation for further information.

user
is the user login necessary for protocols that need authorization.

:*password*
is the user password. It is used in the protocols :pserver: and :sspi:. However, the direct use of this field is not recommended. Instead it is assumed that you login, and your password is remembered for the duration of your working session.

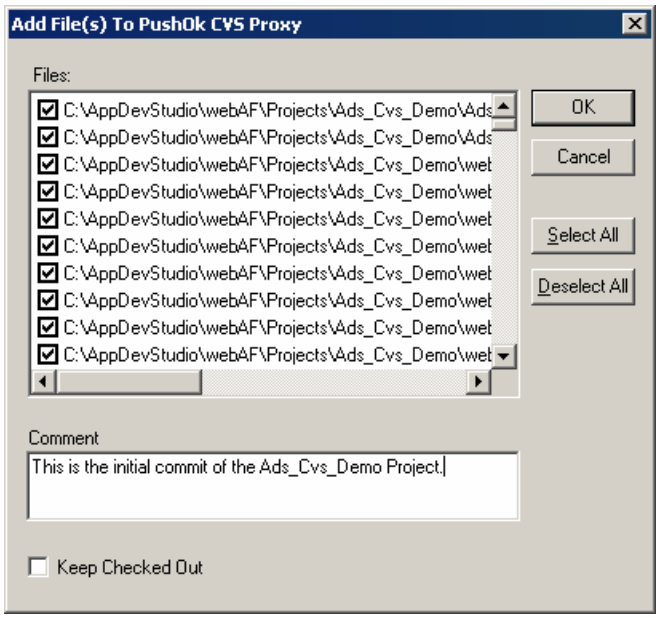
@servername
is the server name for client-server protocols. The character "@" is required at the beginning of the server name.

:*serverpath*
is the path to the repository on server. The character ":" is required at the beginning of the path. The path itself can be in either UNIX (**`/usr/local/cvs`**), or in Windows format (**`C:\cvs\repository`**).

Module

Module is a folder inside the repository. Its name is a path to it, beginning from the repository root. For instance, if your repository is located in the **`C:\cvs\repository`** directory, the MyTestProject module will be physically located in the **`C:\cvs\repository\MyTestProject`** directory. By default, the module name is the same as the SAS AppDev Studio project.

Next a list of all files in the project appears.

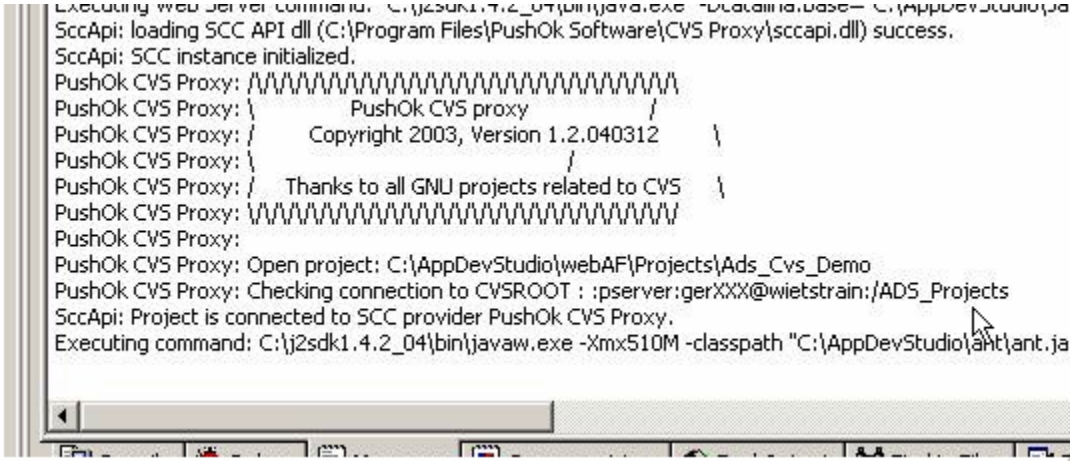


If you want other users to be able to check out the complete project from the CVS Server, then you have to check in all your project files. These files are already marked by default.

Enter a comment. If you use Javadoc comments in your source files, this comment will be added to your source code.

Click **OK** and wait until all files are committed. The first commitment will take some time because in this case all JAR files and GIF files are also committed.

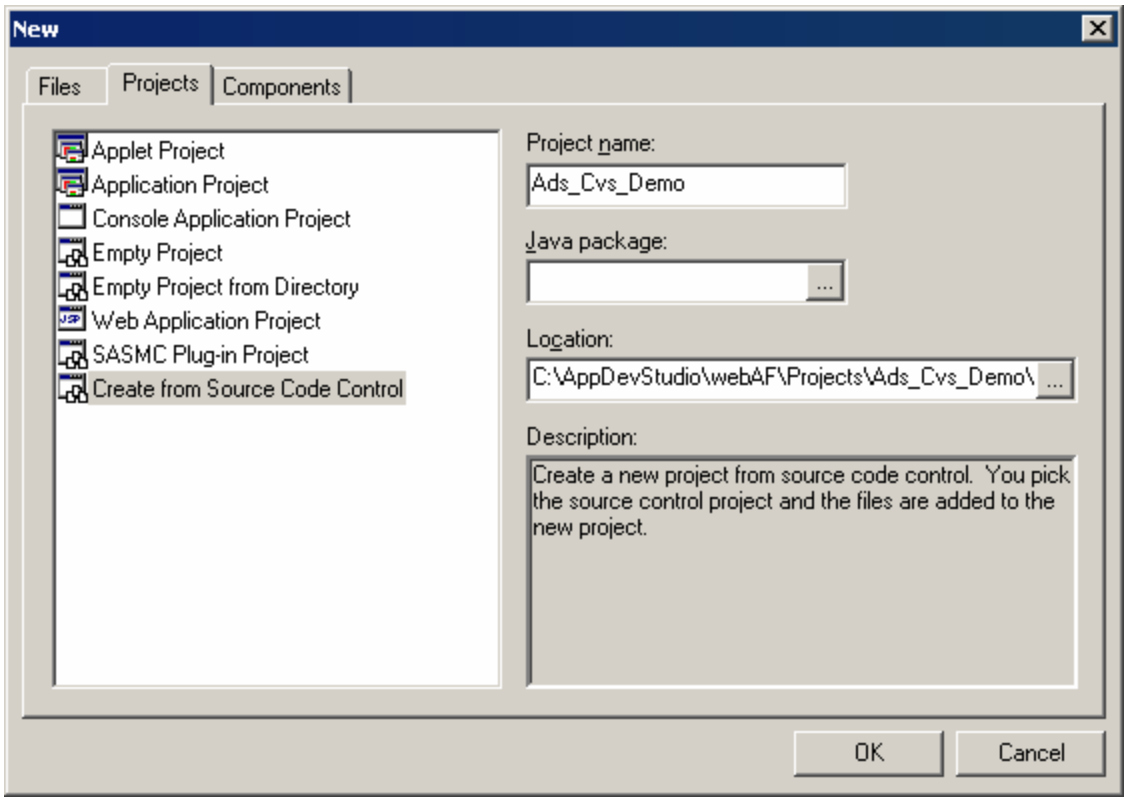
Now your project is connected to the CVS Server. After the project is connected to the SCC you should see the following messages in the message window each time you load your project in SAS AppDev Studio:



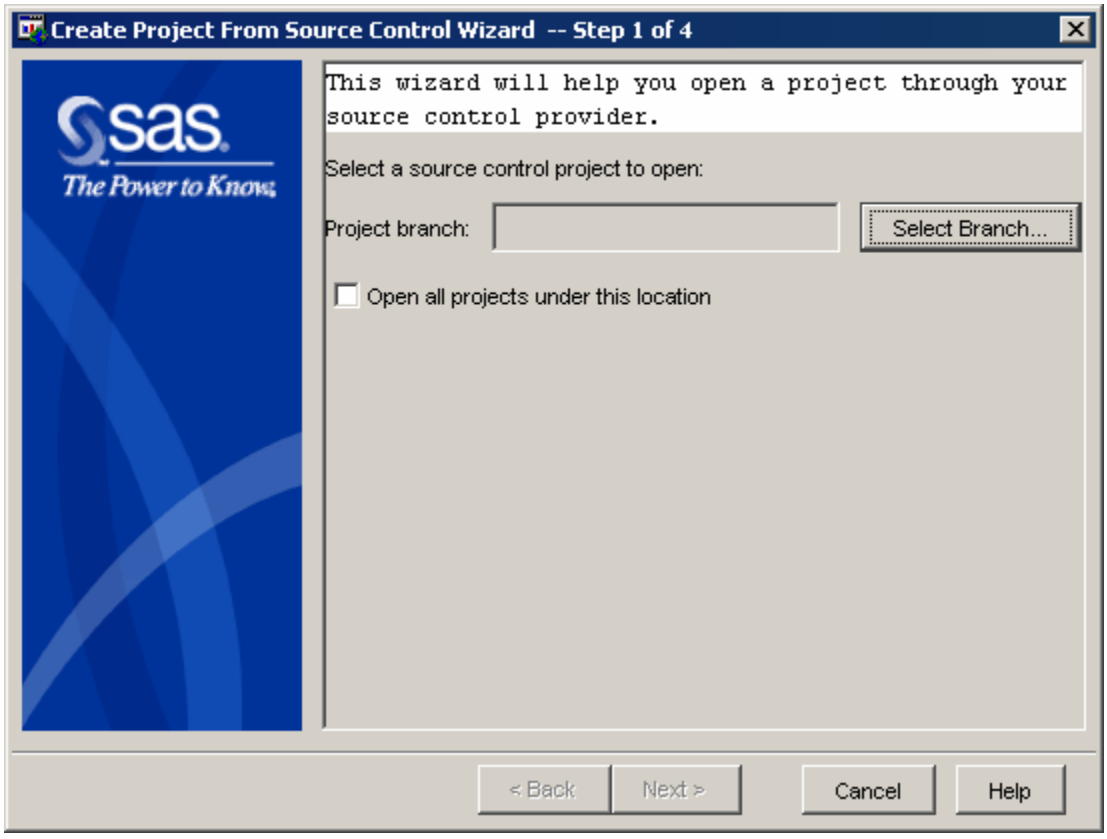
Create from Source Code Control

The following example is used to demonstrate this functionality.
A second developer, gerXXX, performs a checkout of the complete project from the CVS server and both developers change the same source file.

Create a new Project. There is a new entry available: Create from Source Code Control.
Enter the project name **Ads_Cvs_Demo**.

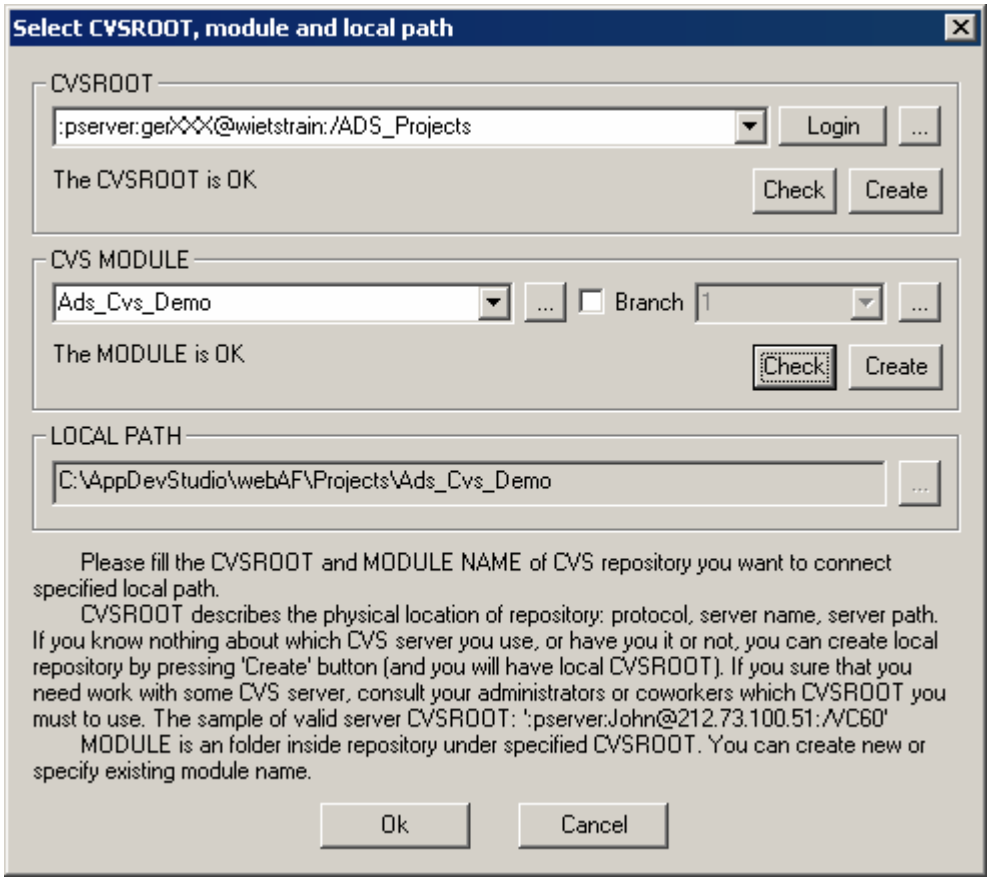


The Create Project From Source Control Wizard opens as shown:



Click **Select Branch**.

In the **Select CVSROOT, module and local path** dialog box enter your CVSROOT and the CVS Module, just as the first developer did. Check CVSROOT and CVS Module. The result should be as follows:

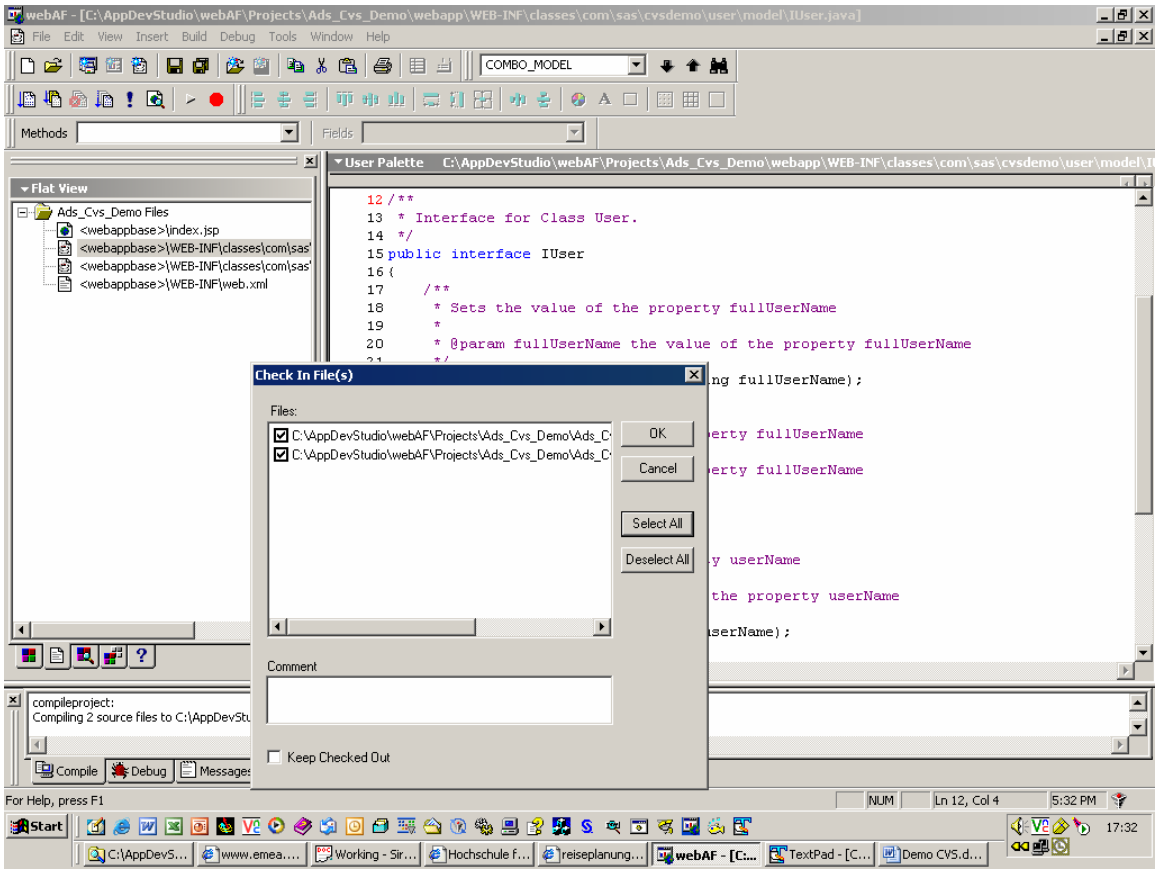


Use the default values for all further steps of the wizard.

Check In

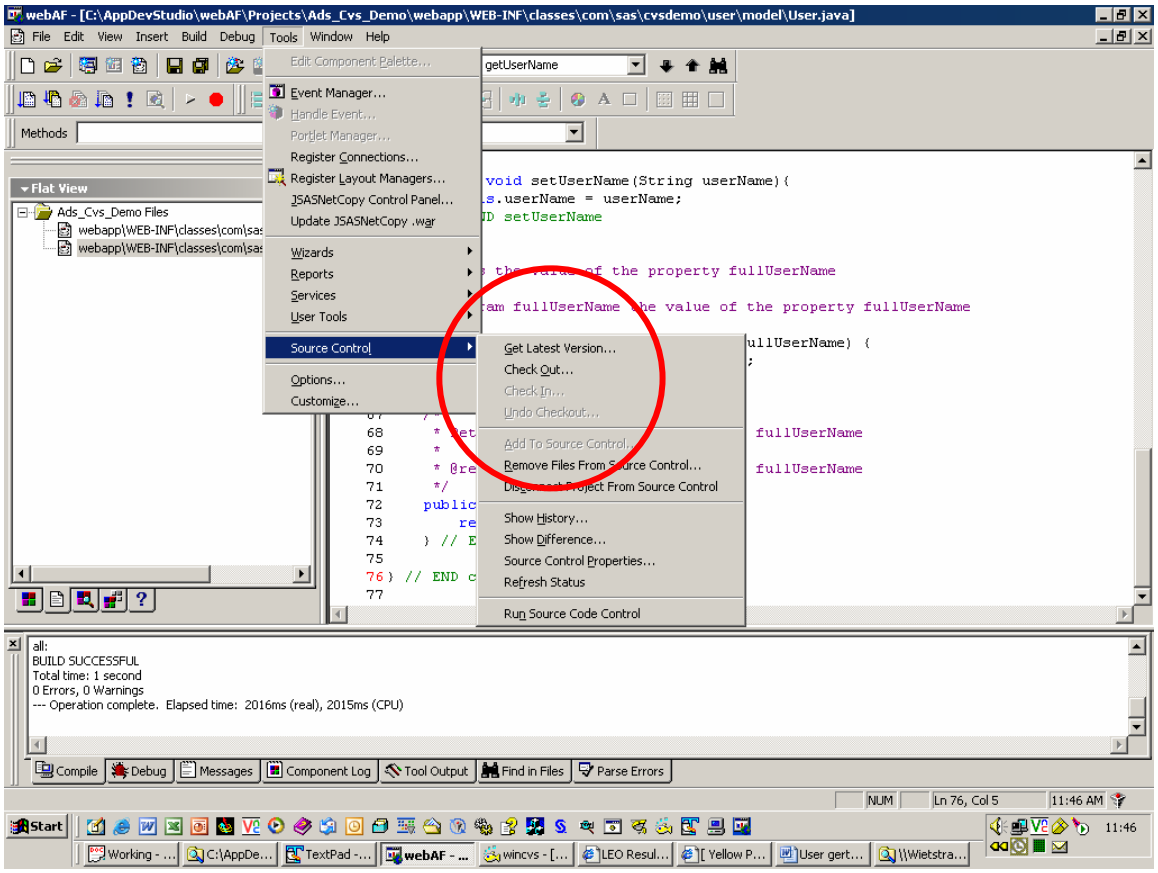
Assumption:
The second developer changes the source and adds new methods. He tested his changes locally and now wants to commit these changes.

Select **Tools > Source Control > Check In**. All modified files will be checked in the file list. Enter a comment about your changes and click **OK**.



Conflicts

Assumption:
The first developer did not notice this change and also changed the same file. He added the methods getUserId() and setUserId() to the interface and the class. Now he wants to commit his changes, but the **Check in** command is grayed out as shown below:

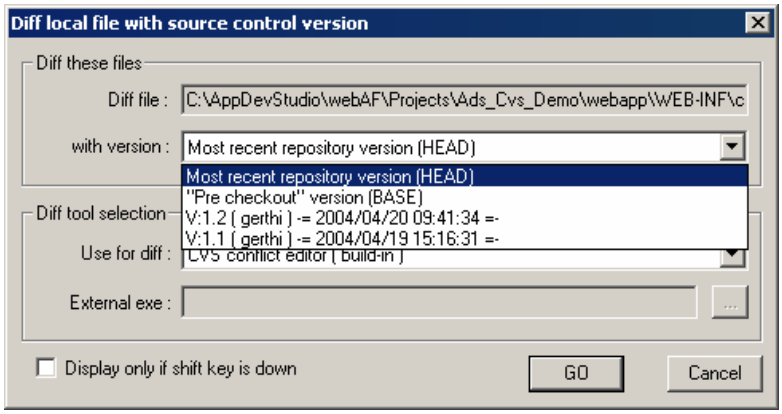


The next step is to see whether there are any changes in the CVS Repository.

What to do in this case depends on the situation. Sometimes you will check out the actual version and use the automatic merge function. Sometimes you want to see what the differences are beforehand. In this case, check the differences first.

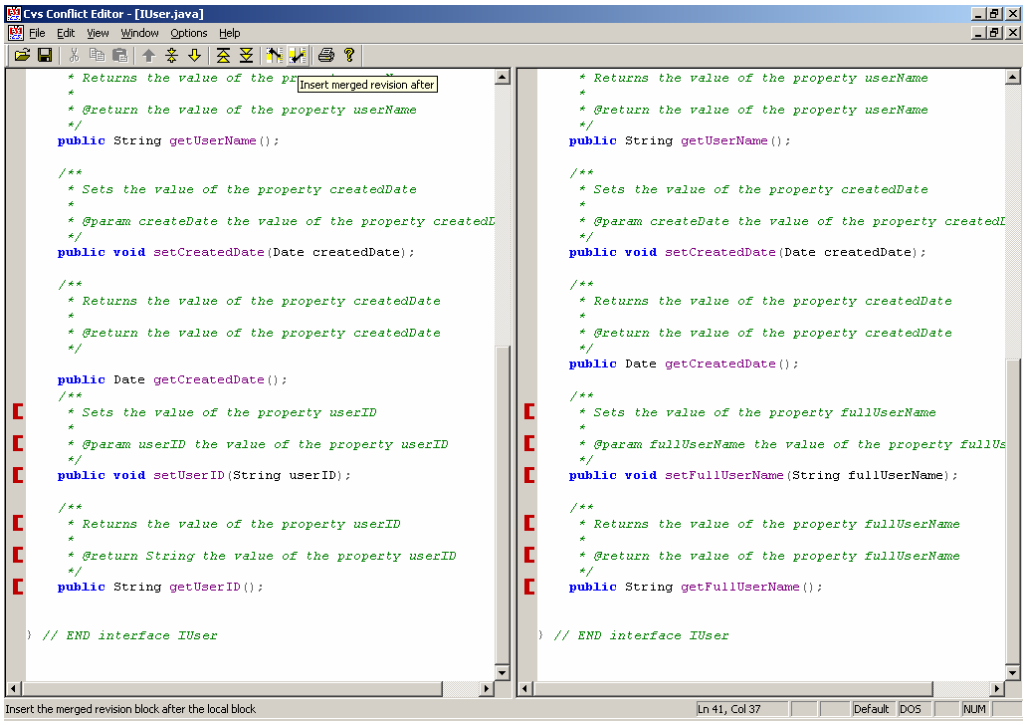
Show Difference

- Select **Tools > Source Control > Show Difference**.
- Select the file you want to compare with the source control version.
- Select the source control version you want to compare with your local file.



This feature enables you to compare the current version of the file with one of the versions in the repository. The necessary version can be chosen in the field **with version**, and the required compare program in the **Diff tool selection**. By default the last version is selected for comparison in the repository (HEAD).

Here is the CVS conflict editor.



The differences are marked red because the same (new) lines were modified. There are no conflicts so you can now merge the code. You can do this either manually here in the conflict editor or automatically by checking out the latest version.

Select **Tools > Source Control > Check Out**.

Notice the messages in the messages window:

PushOk CVS Proxy: RCS file: /ADS_Projects/Ads_Cvs_Demo/webapp/WEB-INF/classes/com/sas/cvsdemo/user/model/IUser.java,v
PushOk CVS Proxy: retrieving revision 1.1
PushOk CVS Proxy: retrieving revision 1.2
PushOk CVS Proxy: Merging differences between 1.1 and 1.2 into IUser.java
PushOk CVS Proxy: Merging IUser.java

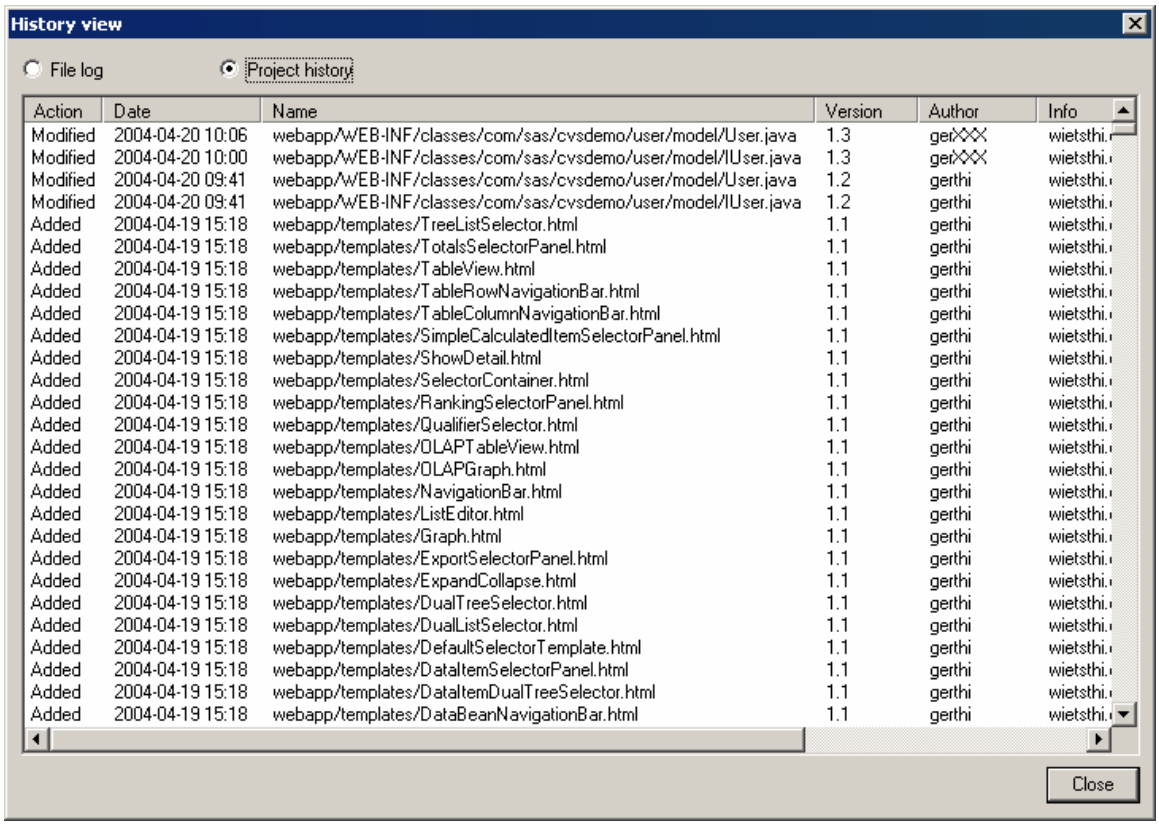
A real conflict would be the following scenario:

Both developers added a property userId with get- and set- methods. But one developer chooses type int and the other type String for the userID. This would be a conflict that cannot be solved automatically. The two developers would have to talk with each other about this in order to correct the conflict. If the second developer uses check out, both lines will be added to his local

copy of the file and the file will be marked in conflict. He will have to resolve this conflict to commit the new version.

History View

Other interesting tools are available. In the project history of the history view you can find all actions performed on the project by the CVS server.

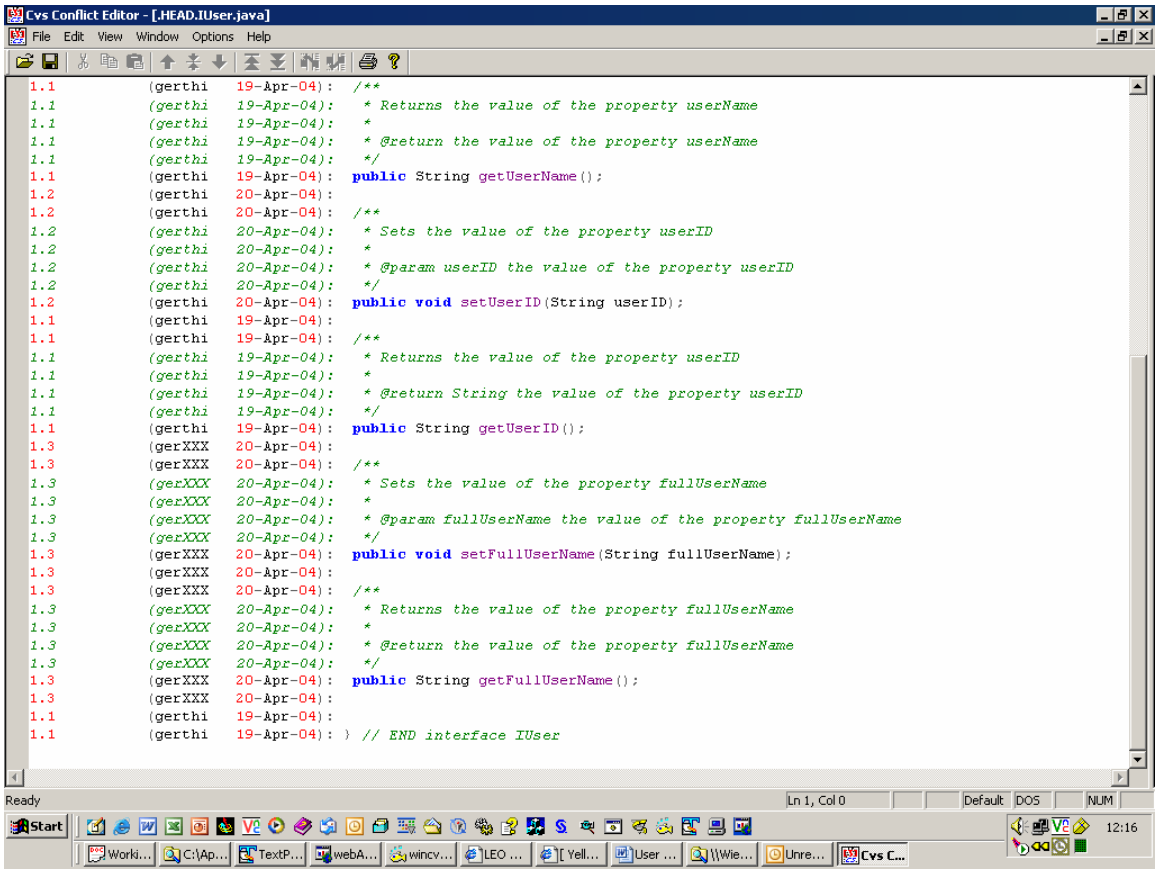


Action	Date	Name	Version	Author	Info
Modified	2004-04-20 10:06	webapp/WEB-INF/classes/com/sas/cvsdemo/user/model/User.java	1.3	gerXXX	wietsthi.
Modified	2004-04-20 10:00	webapp/WEB-INF/classes/com/sas/cvsdemo/user/model/IUser.java	1.3	gerXXX	wietsthi.
Modified	2004-04-20 09:41	webapp/WEB-INF/classes/com/sas/cvsdemo/user/model/User.java	1.2	gerthi	wietsthi.
Modified	2004-04-20 09:41	webapp/WEB-INF/classes/com/sas/cvsdemo/user/model/IUser.java	1.2	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/TreeListSelector.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/TotalsSelectorPanel.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/TableView.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/TableRowNavigationBar.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/TableColumnNavigationBar.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/SimpleCalculatedItemSelectorPanel.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/ShowDetail.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/SelectorContainer.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/RankingSelectorPanel.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/QualifierSelector.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/OLAPTableView.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/OLAPGraph.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/NavigationBar.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/ListEditor.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/Graph.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/ExportSelectorPanel.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/ExpandCollapse.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/DualTreeSelector.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/DualListSelector.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/DefaultSelectorTemplate.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/DataItemSelectorPanel.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/DataItemDualTreeSelector.html	1.1	gerthi	wietsthi.
Added	2004-04-19 15:18	webapp/templates/DataBeanNavigationBar.html	1.1	gerthi	wietsthi.

In the file log of the history view you can see the different versions and who has committed them. Here you have the ability to do the following:

- Get back to an older Version.
- Show the difference between miscellaneous versions such as the following: what are the differences between 1.1 and 1.3.
- Check the files you want to compare.

Using the **Annotate** tab in the History View window, you can see who has committed which code in which version:



Remove File

If you remove any files from your SAS webAF project, then you should also remove them from source control. To remove files from source control, use the **Tools > Source Control > Remove from Source Control** command.

Tips and Links

- If it is your first time working with CVS get some practice with a small sample project before using it for a customer project.
- For some actions a more powerful, external tool might be appropriate. Examples of such actions would be a complete checkout after a long time (vacation) or actions with multiple files.
- WinCVS is one of the best external tools to access the CVS repository. WinMerge is good for comparing files.
- You can use the CVS SCC plug-in together with WinCVS. Both tools complement one another.

PushOK CVS SCC plug-in

Download:
http://www.pushok.com/soft_download.php?idprogram=2

Reference Manual:
http://www.pushok.com/help/cvsscc/index.php

CVSNT

Download:
http://www.cvsnt.org/wiki/

Reference Manual:
http://www.cvsnt.org/manual/

Installation Tips:
http://www.cvsnt.org/wiki/InstallationTips

WinCVS: Standalone CvsGui client

Download:
http://www.wincvs.org/download.html

Documentation:
http://www.wincvs.org/doc.html