Integrating Jalopy code formatting in CVS repositories

A NOW guide

Dr. Wolfgang Thronicke

C-LAB Short Report

Vol. 1 (2004) No. 2

Cooperative Computing & Communication Laboratory

ISSN 1614-1172

C-LAB ist eine Kooperation
der Universität Paderborn und der Siemens Business Services GmbH & Co OHG
www.c-lab.de
info@c-lab.de

C-LAB Short Report

Herausgegeben von
Published by

Dr. Wolfgang Kern, Siemens Business Services GmbH & Co OHG

Prof. Dr. Franz-Josef Rammig, Universität Paderborn

Das C-LAB - Cooperative Computing & Communication Laboratory - leistet Forschungs- und Entwicklungsarbeiten und gewährleistet deren Transfer an den Markt. Es wurde 1985 von den Partnern Nixdorf Computer AG (nun Siemens Business Services GmbH & Co OHG) und der Universität Paderborn im Einvernehmen mit dem Land Nordrhein-Westfalen gegründet.
Die Vision, die dem C-LAB zugrunde liegt, geht davon aus, dass die gewaltigen Herausforderungen beim Übergang in die kommende Informationsgesellschaft nur durch globale Kooperation und in tiefer Verzahnung von Theorie und Praxis gelöst werden können. Im C-LAB arbeiten deshalb Mitarbeiter von Hochschule und Industrie unter einem Dach in einer gemeinsamen Organisation an gemeinsamen Projekten mit internationalen Partnern eng zusammen.

C-LAB - the Cooperative Computing & Cooperation Laboratory - works in the area of research and development and safeguards its transfer into the market. It was founded in 1985 by Nixdorf Computer AG (now Siemens Business Services GmbH & Co OHG) and the University of Paderborn under the auspices of the State of North-Rhine Westphalia.
C-LAB's vision is based on the fundamental premise that the gargantuan challenges thrown up by the transition to a future information society can only be met through global cooperation and deep interworking of theory and practice. This is why, under one roof, staff from the university and from industry cooperate closely on joint projects within a common research and development organization together with international partners. In doing so, C-LAB concentrates on those innovative subject areas in which cooperation is expected to bear particular fruit for the partners and their general well-being.

ISSN 1614-1172

C-LAB
Fürstenallee 11
33102 Paderborn
fon: +49 5251 60 60 60
fax: +49 5251 60 60 66
email: info@c-lab.de
Internet: www.c-lab.de

Integrating Jalopy code formatting in CVS repositories

A NOW[1] guide

Dr. Wolfgang Thronicke

Siemens Business Services

2004

This report documents the steps necessary to integrate a tool into CVS to achieve automated formatting while checking in new documents and source code.

A short note about CVS repositories

CVS repositories are perhaps the most commonly used tools to manage source code and documents in development projects. They form the back-store to manage versions and releases. Especially the open-source community would not function without it, with Sourceforge as its most prominent CVS repository provider.

One issue of CVS is to process files before they are stored in the repository. This interface allows the integration of tools and scripts to augment the functionality.

The Task

For software projects with many participants it is desirable to establish a coding convention that is reflected by a uniform formatting of the sources. This allow developers and reviewers to grasp alien code more quickly and thus contributes to overall coding quality.

For JAVA there is an open-source code formatter “jalopy” which can be found on Sourceforge. The task has been to integrate it into CVS in such a way that its use remains flexible. This report assumes that the console version of jalopy[2] is properly installed.

Leverage

Script and tool execution in CVS

CVS[3] allows to integrate scripts and tools so that they are applied to the committed files. The scripts specified in the `commitinfo` file which is found in the CVSROOT directory of the repository. By adding a line like

<pattern> <script>

the script is invoked on all commits to files whose path matches the pattern. Note that you cannot match for the file itself! This would have greatly simplified the task since you could restrict the script execution to something like *.java.

Another effect of this processing is the control of the commit process. If the script return a non-zero value the commit process is aborted. This can be used to ensure that only files passing a test can be checked in. If the file is altered by this script (but not its filename!) the

1 The NOW project is a public funded German project investigating the benefits of using open-source in industrial and commercial contexts (see: http://now.c-lab.de).
2 As of this writing the open-source version has been 1.0b10.
3 I will not discuss the details of CVS here. Please consult a CVS manual or book.

altered version is checked in.

Placing configuration files in CVS

Some tools allow to be parametrized and jalopy is no exception to that. Hence it is desirable to allow this specification to be altered by the repository users in a convenient way. The first idea to simply put the configuration files into the repository is not bad at all. However, files stored in CVS are versioned and need to be checked out before use. Of course a script could compensate for this but it adds to the complexity of the integration. Luckily in CVS there is a place where is a group of files which is automagically checked out inside(!) the repository after a new version has been checked in. This mechanism is restricted to the CVSROOT directory but it can be augmented to include additional files. Just insert the names of the files into the file `checkoutlist`.

The integration

The first step of course has to be the installation of jalopy which itself depends on a working JAVA environment. Since simply calling jalopy is a real bad idea (it would try to format all kinds of files), a driver-script is necessary which adds the necessary intelligence for the process.

In this script the following functionality is realized:

- Filtering of JAVA files for jalopy-processing
- Commit control: A file `jalopy_mode` controls if JAVA files are processed and if not conformant files are rejected:
    - off: no jalopy processing of JAVA files
    - on: processing of JAVA files, but continues if jalopy throws an error
    - strict: enforces successful formatting by jalopy, rejects commit otherwise.
- Check for the configuration file and apply it if available.

Since CVS is driven by the pserver access method the environment variable CVSROOT is set, so all necessary files can be located easily.

Tips

- CVS starts the script for every commit with an arbitrary number of files to be committed. If you are dealing with a high traffic situation on a CVS server it is sensible to restrict the use of the script to certain paths of the repository.
- Jalopy 1.10 depends on an integrated JAVA parser and requires a JAVA 1.3 installation. It can handle JAVA 1.4, but I am not sure with JAVA 1.5. Since jalopy is open-source the alterations in the source of jalopy can be made quite easily.

The driver script

This script is given as an example for the integration. Some lines helpful for debugging and diagnosis have been commented out.

```
#!/bin/bash
#
# This is the cvs integration driver for jalopy
# (c) Siemens Business Services 2004
#
# Autor: W. Thronicke
# Date: 2003-04-07
#
```

```
# echo "Commit processing start `date`" >> /tmp/cvslog
# DEBUG
# set >> /tmp/cvslog
#

export JALOPYHOME=/home/cvs/jalopy
export JAVA_HOME=/usr


# CVSROOT is set from the server! (we are lucky!)
JALOPY_ACTIVATION_FILE=$CVSROOT/CVSROOT/jalopy_mode
JALOPY_CONFIGURATION=$CVSROOT/CVSROOT/jalopyconv.xml

if [ -r $JALOPY_ACTIVATION_FILE ]; then
  val=`cat $JALOPY_ACTIVATION_FILE`
else
  val='off'
fi
if [ "$val" = "off" ]; then
 # exiting processing
 exit 0
fi

confopt=""
if [ -r $JALOPY_CONFIGURATION ]; then
  confopt="--convention=$JALOPY_CONFIGURATION"
fi


# skip directory argument
shift
for thefile in $*
do
   # if file does not exist it is probably deleted ...
  # so don't process it
  if [ -f $thefile ]; then
  case $thefile in
  *.java ) $JALOPYHOME/bin/jalopy.sh $confopt $thefile
           ret=$?
           #echo "Retcode: $ret" >> /tmp/cvslog
           # in case of "strict" we abort in every other case
           # we commit silently
           if [ $ret = 1 ] && [ $val = "strict" ]; then
              exit 1
           fi
           ;;
  esac
  fi
done
exit 0
```

The creation of parts of this document has been funded by the German NOW project under grant no. 01 ISB 04D.