

JAVA CoG KIT CODING GUIDELINES

Gregor von Laszewski

Software Version: 4.1.2

Manual version: 02/18/05

Url: <http://www.cogkit.org/release/4.1.2/manual/coding.pdf>

Url: <http://www.cogkit.org/release/4.1.2/manual/coding/coding.html>

Last update: July 11, 2005

CONTENTS

| | |
|--|-----------|
| 1 About this Document | 3 |
| 2 Registration | 3 |
| 3 Introduction | 4 |
| 4 The Java CoG Kit Module Concept | 4 |
| 5 Versioning | 4 |
| 6 Directory names | 4 |
| 7 Imports | 4 |
| 8 Indentation | 4 |
| 9 Brackets | 4 |
| 10 Variables | 5 |
| 11 Instance Variables | 5 |
| 12 One-Liners | 5 |
| 13 Logging | 5 |
| 14 Testing | 5 |
| 15 Internationalization | 5 |
| 16 Library Reuse | 5 |
| 17 Exceptions | 6 |
| 18 System.out | 6 |
| 19 Logging | 6 |
| 20 Tools | 7 |
| 21 Eclipse | 9 |
| Appendix | 12 |

| | |
|---|-----------|
| A Java CoG Kit Guides | 12 |
| B Java CoG Kit Guides Under Construction | 12 |
| C Available Downloads | 12 |
| D Availability of the Document | 13 |
| E Bugs | 13 |
| F Administrative Contact | 13 |

1. ABOUT THIS DOCUMENT

This document includes basic information about the different Java CoG Kit distributions in order to help you chose the one that is most suitable to your needs.

1.1. Reproduction


The material presented in this document can not be published, mirrored, electronically or otherwise reproduced without prior written consent. As you can link to this document, this should not pose much of a restriction.

1.2. Viewing

The best way to read this document is with Adobe Acrobat Reader. Please make sure you configure Adobe Acrobat Reader appropriately so you can follow hyperlinks. This is the case if you follow the default installation. Acrobat Reader is available at <http://www.adobe.com/products/acrobat/readermain.html>. Because the hyperlinks are not available in the printed form of this manual and we support saving our environment we strongly discourage printing this document.

We recommend that you save this manual locally on your machine and use Acrobat Reader. This has the advantage that you do not lose your anchor points while switching back and forth between different hyperlinks. An HTML version of this manual is planed, but not available yet.

1.3. Format

We have augmented the document with some comments at places where we found issues. Our intend is to address these issues in a future release. The comments are marked by the icon  and the name of the person that will work on the removal of the issue.

2. REGISTRATION

Please be a team player and support us indirectly by registering with us or reporting your use of the Java CoG Kit. Although this software is free, we still need to justify to our funders the usefulness of the projects. If you want to help us with our efforts please take a few seconds to complete this information. We do not use this information for other purposes. If you have special needs or concerns please contact gregor@mcs.anl.gov. The registration form can filled out in a variety of formats. The online form can be found at

<http://www.cogkit.org/register>

This form is available also as ASCII text at

<http://www.cogkit.org/register/form.txt>

which you can FAX to

Gregor von Laszewski, Fax: 630 252 1997

3. INTRODUCTION

In case you like to contribute to the Java CoG Kit we have established these simple coding guidelines.

The Java CoG Kit follows the basic coding conventions given in the “Sun Coding Conventions for the Java Programming Language” [\[HTML\]](#).

Additionally we have the following rules given in this guide.

4. THE JAVA COG KIT MODULE CONCEPT

◆ *Kaizar: Describe it.*

◆ Kaizar

5. VERSIONING

5.1. Java CoG Kit

The version for the CoG Kit can be found in the cog/VERSION.txt. It follows the same rules as for module versions. Please, note that the Java CoG Kit version number and the version numbers for modules differ. Consider the Java CoG Kit as one big module.

5.2. Modules

Java CoG Kit modules must include an accurate versioning. Every time you make a change in a module and commit it to CVS, it needs to be associated with a version that can be figured from just the jar file.

The version of the module is to be integrated in the file

VERSION.txt

This version reflects a per module version number. In addition you must include an entry into *two* CHANGES.txt file. One, that you maintain as part of your module in which you can include a lot of details about changes, and one in the Java CoG Kit main directory in which you write a summary of the changes. Many times, you can just write the same changes in both files.

The version numbers you will use between versions are minor versions.

Modules must not have an “_” in their directory name. Instead all modules must use an “-” as this will simplify our process that we use to autogenerate documentation from the modules.

6. DIRECTORY NAMES

Directory names must not have an “_” in their directory name. Instead all modules must use an “-” as this will simplify our process that we use to autogenerate documentation from the modules.

7. IMPORTS

All imports must be single class and explicit. That is, `import <package>.*` is not allowed.

8. INDENTATION

All indentation levels should be 4 spaces. No editor tabs are allowed unless they are converted to 4 spaces before saving the file.

9. BRACKETS

All brackets must follow the Java Coding guidelines. E.g.

```
for (index = 0; index < length; index++) {  
    <code>  
}
```

Having the bracket below the “for” should be avoided. The reason for this is that our automatic code checking tools have an easier time to analyze the code.

10. VARIABLES

No acronyms or abbreviations should be used. For example, `a = b + mVarLen` should be avoided. Instead, use: `totalLength = partLength + newLength`.

11. INSTANCE VARIABLES

Use “this.” prefix when referencing instance variables, for example:

```
public MyClass (ServicePropertiesInterface properties) {  
    this.properties = properties;  
}  
  
public int foo () {  
    int localInt = 3;  
    return this.instanceInt + localInt;  
}
```

12. ONE-LINERS

Even single line statements should be inside brackets, for example:

```
if (isEmpty) {  
    return;  
}
```

13. LOGGING

Log4J should be used exclusively. `System.out.println` and `System.err.println` is not allowed. Further, exceptions should be logged.

14. TESTING

Each component or class should have a JUnit test The tests should be put in `test/` directory under each package directory.

15. INTERNATIONALIZATION

The core framework should be fully internationalized. The samples may be internationalized. The Java I18n/L10n Toolkit may be used to verify whether code is international. However as we do not have funding to support internationalization in our ongoing efforts, we treat it at this time with low priority.

16. LIBRARY REUSE

Treat all code as a library, and as a reusable component. Calls to `System.exit()` are disallowed (except the main method)

17. EXCEPTIONS

Use chained exceptions. Java CoG Kit provides two simple generic exception classes for chaining multiple exceptions together. Look at `ChainedException` and `ChainedIOException`.

18. SYSTEM.OUT

The call of `System.out` is to be avoided as it terminates the JVM. Only commandline tools may call `system.out`. However it is better to assume that even command line tools should be written first with an exception and another wrapper should call `System.out` dependent on the exception. This way the command can be reused within other programs without terminating the JVM based on an error (see Logging).

19. LOGGING

We use the following conventions for using the log4j calls.

fatal: Something from which the application cannot recover. Application must be terminated.

Error: An exceptional state was encountered, and the application is very likely to not behave correctly. Continue at your own risk

Warn: An exceptional state was encountered, but the effects will not be on the functional side (say, an icon is missing).

Info: Overall flow

Debug: Detailed flow and data dumps

More information about how to use log4j can be found at

<http://logging.apache.org/log4j/docs/manual.html>

To use log4j you need to include the following import statement

```
import org.apache.log4j.Logger;
\end{Verbatim}
```

If you use loggers in your code, they must be `private`, `final`, and `static`.

```
private final static Logger logger = Logger.getLogger(HTTPPost.class);
```

```
\TODO{Robert}{The parameter to getLogger is unclear.}
```

In the code you can use statements such as

```
\begin{Verbatim}
logger.info("An example for a Information statement: I entered this method");
logger.fatal("An example for a fatal error related to a URL", urlException);
logger.error("An example for an error", ioException)
logger.debug("An example for a debug statement");
logger.warn("An example for a warning statement");
\end{Verbatim}
```

In case you use a string concatenation or other relevant lines to prepare the log message you must use an if statement before printing the log message to increase performance.

```
if(logger.isDebugEnabled()) {
    logger.debug("variable="+variable);
}
```

The properties of the logging facilities can also be customised during runtime while modifying the log4j.properties file. An example is listed below.

```
#####
#Root category
#####
log4j.rootCategory=WARN, STDOUT, ERROR-DIALOG,LOG-FILE
# uncomment this one to use the ERROR-DIALOG and change root to DEBUG level
# log4j.rootCategory=DEBUG, STDOUT, ERROR-DIALOG, LOG-FILE

#####
# ERROR-DIALOG is an error dialog for the desktop that allows errors to be emailed
#####
log4j.appender.ERROR-DIALOG=org.globus.cog.gridface.impl.util.ErrorDialogAppender
# override the value for ERROR-DIALOG to WARN
log4j.appender.ERROR-DIALOG.level=WARN
log4j.appender.ERROR-DIALOG.headerPattern=%d %-5p [%t] %C{2} (%F:%L) - %m%n
log4j.appender.ERROR-DIALOG.layout=org.apache.log4j.HTMLLayout

#####
# STDOUT is set to be a ConsoleAppender using a PatternLayout
#####
log4j.appender.STDOUT=org.apache.log4j.ConsoleAppender
log4j.appender.STDOUT.layout=org.apache.log4j.PatternLayout
log4j.appender.STDOUT.layout.ConversionPattern=%-5p [%c] %x - %m%n

#####
# LOG-FILE appends to a log file and can be read using chainsaw and the chainsaw-config.xml
#####
log4j.appender.LOG-FILE=org.apache.log4j.RollingFileAppender
log4j.appender.LOG-FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.LOG-FILE.layout.ConversionPattern=%d %-5p [%t] %C{2} (%F:%L) - %m%n
log4j.appender.LOG-FILE.MaxFileSize=20KB
log4j.appender.LOG-FILE.Append=true
log4j.appender.LOG-FILE.File=example.log

#####
# customize classes
#####
log4j.logger.org.globus.cog.gridfaces=INFO
```

20. TOOLS

We are using findbugs and PMD to verify the coding standard. Tools for to check or manipulate the code quality are currently located in

/cogkit/src/cog/tools

⚠ *Kaizar: These tools may move to cogkit/tools in a seperate CVS that only developers have access to.*

There is a special directory called

```
./cogkit/src/cog/qualitycontrol
```

in that contains various configuration files for findbugs and pmd.

⚠ *TBD: In future we may include different configuration files for other tools such as deriving UML diagrams.*

These files are

```
./cogkit/src/cog/qualitycontrol/pmd.xml
./cogkit/src/cog/qualitycontrol/findbugs.fb
```

The files contain our current defaults. The appropriate quality control programs an be installed as follows.

```
cd ./cogkit/src/cog
make install-qc
```

⚠ *Kaizar: This makefile should at one point become an ant file.*

Now you can invoke the quality control programs.

20.1. Findbugs

Find bugs can be run by

```
cd ./cogkit/src/cog
make findbugs
```

Findbugs in Cygwin. Findbugs will not work in Cygwin. In order to invoke it on a Windows machine, you should type in the command findbugs command in a Windows CMD.EXE window.

```
cd ./cogkit/src/cog
cd qualitycontrol; \
'../../tools/findbugs-0.8.6/bin/findbugs.bat' \
    -textui -low -html -project findbugs.fb > findbugs.html
```

Publishing the Results. On Windows findbugs should be called in the Windows CMD.exe program, it does not work under cygwin.

The results of findbugs may be posted on the release distribution page as an <http://www.cogkit.org/release/4-> document. This can be done by calling

```
make publish
```

Please note that not all errors are relevant. We will improve the code quality gradually. We recommend that that developers and contributors use PMD (<http://pmd.sourceforge.net>) to check their code. Many of the complaints that PMD generates should be taken seriously. Still, there are instances when PMD rules do not apply for a good reason and create false positives.

To use pmd, you need to download it and add all its jar files to the cog/tools/pmd-2.3/lib directory. Afterwards, just run 'ant pmd' in the module you want to check. It will generate both an on-screen report and an html report (pmd-report.html). For our core developers we have a makefile in the ./cog directory that will install pmd appropriately by calling "make install-qc".

20.2. Automatic Cron scripts

◆ *Kaizar: write the cron scripts for pmd and findbug and install them on gong or wiggum as a nightly run at around 3:00am*

◆ Kaizar

21. ECLIPSE

◆ *Deepti: Improve the documentation. I have for example not put headers on the figures.*

◆ Deepti

We do not recommend to use the CVS feature form eclipse at this time. Instead we recommend the following procedure.

1. Create directory "cogkit" under your cygwin home.

```
> mkdir cogkit
> cd cogkit
```

2. Checking out from CVS:

```
from>thespdrvrserver:anonymous@cvs.cogkit.org:/cvs/cogkit login
> cvs -d :pserver:anonymous@cvs.cogkit.org:/cvs/cogkit checkout src/cog
```

with a login account (We assume you use bash). In your bashrc file set

```
alias ncvs='cvs -d <youruserlogin>@cvs.cogkit.org/cvs/cogkit
```

In a new shell you can no say

```
ncvs checkout -P src/cog
```

3. Open Eclipse.
4. Switch workspace if a workspace from a previous project is opened. Create a new Project Workspace directory "CogKit". It would be better to make the workspace location different from the "cogkit" directory where you checked out the code. (see [Figure 1](#))
5. Create a new "Java Project". The screen below is shown when a new project is to be created. (see [Figure 2](#))
6. Enter a Project Name "Cog" and choose "Create Project at external location". Browse to find the "cogkit" directory within cygwin where the cogkit was checked out from the cvs. (see [Figure 3](#))
7. Click on "Configure Defaults" in the above screen and choose Project as the source and output folder.

8. Do not press enter since it will save the project. Click on the “Next” button instead. This will take you to the project Properties screen shown below.
(see Figure 4)
- Make sure you don’t have the source folders on your build path since this will lead to a different kind of display of the directories within the eclipse “Package Explorer”. They can be added later on when a build needs to be done.
9. To add modules to the build path: Right click on Project -> Properties -> Java Build Path. Here you could add folders to the build path.
10. Commit to CVS: Log on to cygwin and go to the appropriate directory to do the commits. If adding a new file use cvs add before committing to cvs. Similarly, for to delete files from cvs remove the file from local directory and use cvs remove before committing.

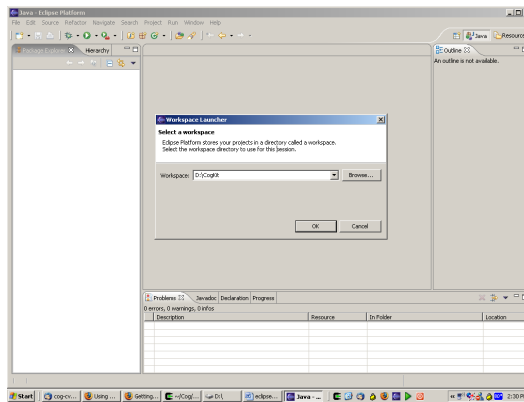


Figure 1: TBD

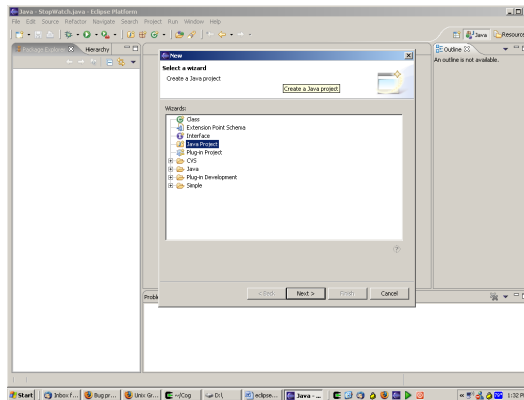


Figure 2: TBD

21.1. Automatic Cron scripts

⚠ *Kaizar: write the cron scripts for pmd and findbug and install them on gong or wiggum as a nightly run at around 3:00am*

⚠ Kaizar

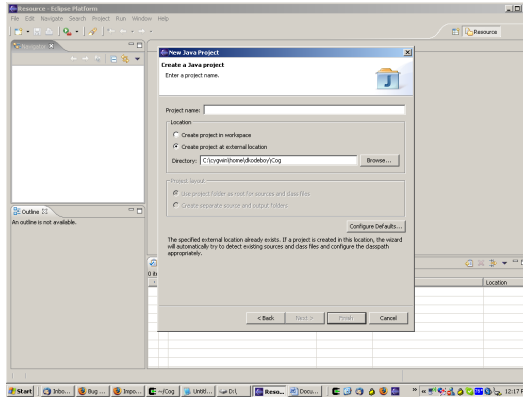


Figure 3: TBD

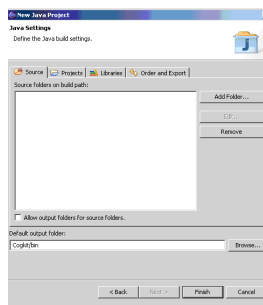


Figure 4: TBD

REFERENCES

- [1] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, “A Java Commodity Grid Kit,” *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: <http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cog-cpe-final.pdf>
- [2] “Java CoG Kit Wiki,” 2004. [Online]. Available: <http://www.cogkit.org/wiki>
- [3] “Java CoG Kit Registration,” 2004. [Online]. Available: <http://www.cogkit.org/register>

Additional publications about the Java CoG Kit can be found as part of the vita of Gregor von Laszewski <http://www-unix.mcs.anl.gov/~laszewsk/vita.pdf>. Most documents are available online if you follow the links. In future we intend to provide this information without Gregors vita data.

If you need to cite the Java CoG Kit, please use [1].

A. JAVA CoG KIT GUIDES

| Short Title | Description | Format |
|--------------|---|---|
| Guide | A guide to help you finding out what guides have been written | [PDF] [HTML] |
| Install | A guide to the different ways of installing the Java CoG Kit | [PDF] [HTML] |
| Commands | A guide to the command line tools of the Java CoG Kit | [PDF] [HTML] |
| Workflow | A guide to the Gridant/Karajan Workflow | [PDF] [HTML] |
| Abstractions | A guide to the Java CoG Kit abstractions API | [PDF] [HTML] |
| JavaDoc | The Java API documentation to the Java CoG Kit | [HTML] |
| Coding | A guide to the Coding rules for the Java CoG Kit | [PDF] [HTML] |
| Overview | A future guide that will be an overview to the Java CoG Kit | [PDF] [HTML] |

B. JAVA CoG KIT GUIDES UNDER CONSTRUCTION

More guides are under development. The following guides are not yet completed, but are listed here to help us improving these guides. Please, explore them and send us e-mail about improvement suggestions. If you like to contribute a guide yourself, please contact gregor@mcs.anl.gov.

| Short Title | Description | Format |
|-----------------|---|---|
| MPI | A preliminary guide to execute MPI programs on the TeraGrid and alike | [PDF] [HTML] |
| Release Process | A preliminary guide to document the release process | [PDF] [HTML] |
| Guide | A preliminary guide to document writing guides | [PDF] [HTML] |
| Examples | A preliminary guide to examples alike | [PDF] [HTML] |

C. AVAILABLE DOWNLOADS

First time users of the Java CoG Kit should read the “Guide to Installing the Java CoG Kit” [\[PDF\]](#) [\[HTML\]](#). We hope that you will find this guide useful to decide which bundles you need. For the more experienced user, we provide the following table.

Binary Distributions

- Complete (all providers) [\[tar.gz\]](#)[\[zip\]](#)
Installation Guide [\[HTML\]](#) [\[PDF\]](#)
- Separate providers
Installation Guide [\[HTML\]](#) [\[PDF\]](#)
 - Main package (includes GT2 providers) [\[tar.gz\]](#)[\[zip\]](#)
 - Common GT 3.x.x package [\[tar.gz\]](#)[\[zip\]](#) (required for all GT 3.x.x providers)
 - GT 3.0.2 provider [\[tar.gz\]](#)[\[zip\]](#)
 - GT 3.2.0 provider [\[tar.gz\]](#)[\[zip\]](#)
 - GT 3.2.1 provider [\[tar.gz\]](#)[\[zip\]](#)

- GT 4.0.0 and 4.0.1 provider [[tar.gz](#)][[zip](#)]
- Condor provider [[tar.gz](#)][[zip](#)]
- SSH provider [[tar.gz](#)][[zip](#)]
- WebDAV provider [[tar.gz](#)][[zip](#)]
- Local provider [[tar.gz](#)][[zip](#)]

Source Distributions

- Complete source distribution
Installation Guide [[HTML](#)] [[PDF](#)]
Source Distribution [[tar.gz](#)][[zip](#)]

API Documentation

- Complete API [[tar.gz](#)][[zip](#)]

Module List

- Module list [[HTML](#)]

CVS

- Please consult the Installation Guide [[HTML](#)] [[PDF](#)]

D. AVAILABILITY OF THE DOCUMENT

The newest version of this document can be downloaded by the developers from

1. `cvs -d:pserver:anonymous@cvs.cogkit.org:/cvs/cogkit checkout manual/guide`

It is not allowed to reproduce this document or the source. This documentation is copyrighted and is not distributed under the CoG Kit license.

E. BUGS

This guide is constantly improved and your input is highly appreciated. Please report suggestion, errors, changes, and new sections or chapters through our [Bugzilla](#) system at <http://www-unix.globus.org/cog/contact/bugs/>

F. ADMINISTRATIVE CONTACT

The Java CoG Kit project has been initiated and is managed by Gregor von Laszewski. To contact him, please use the information below.

Gregor von Laszewski
Argonne National Laboratory
Mathematics and Computer Science Division
9700 South Cass Avenue
Argonne, IL 60439
Phone:(630) 252 0472
Fax: (630) 252 1997
gregor@mcs.anl.gov

INDEX

0 TASKS

To do

Deepti, [9](#)

Kaizar, [4](#), [8-10](#)

TBD, [8](#)

Administrative Contact, [13](#)

Contact, [13](#)

Registration, [3](#)