Multi-agent Environment for Complex SYstems COsimulation (MECSYCO) - User Guide: Debug system

Benjamin Camus[1,2], Julien Vaubourg[2], Yannick Presse[2],
Victorien Elvinger[2], Thomas Paris[1,2], Alexandre Tan[2]
Vincent Chevrier[1,2], Laurent Ciarletta[1,2], Christine Bourjot[1,2]
[1]Universite de Lorraine, CNRS, LORIA UMR 7503,
Vandoeuvre-les-Nancy, F-54506, France.
[2]INRIA, Villers-les-Nancy, F-54600, France.
mecsyco@inria.fr

April 6, 2016

Contents

Introduction

The debug system is a tool adapt to MECSYCO-java only. It helps to debug since it allows the user to follow the proceedings of the simulation (data send, received, by who etc...). It relies on SLJF4J[1]. SLF4J is a facade or abstraction for various logging frameworks. Thereby the choice of the debug system is a decision from the user.

[1] `http://www.slf4j.org/`

Chapter 1

Installation

The debug system is not a jar but a folder that interact with MECSYCO properties. You can download it on the website in the "Download" section. You will also need to install beforehand the dependencies: Logback-core-1.1.3.jar, Logback-classic-1.1.3.jar, janino-2.7.8.jar and commons-compiler-2.7.8.jar.

After the download, you will have a folder named **conf**. This folder need to be placed at the root of your Eclipse project where you need to use the debug system (Figure 1.1).

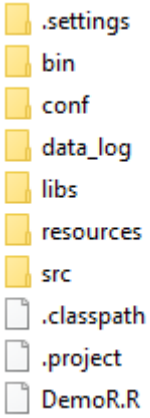

Figure 1.1: The folder to placed "conf" (Getting Started 2 example)

Then, you need to add it to the build path of the project. To do so, use Eclipse interface (Project– >Properties– >Java Build Path– >Add Folder...– >select the conf folder)

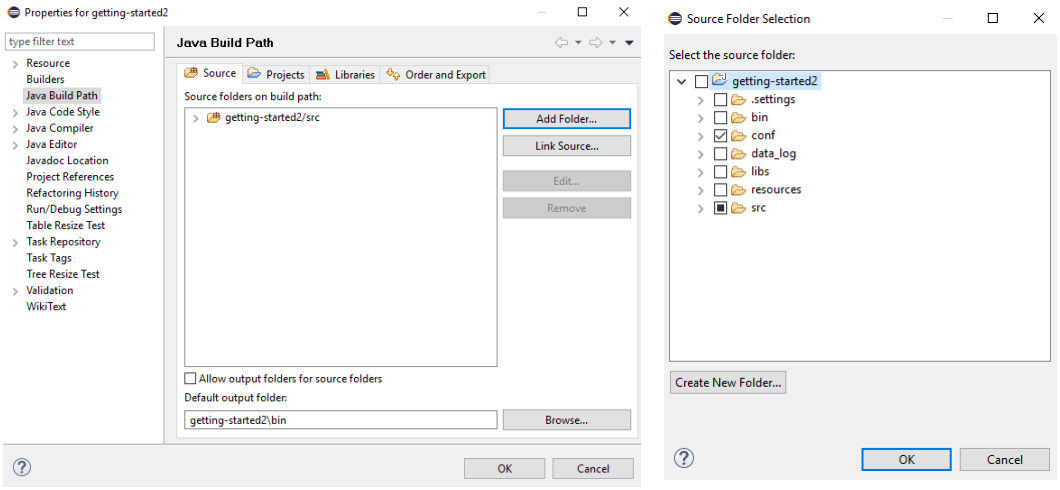

Figure 1.2: Adding "conf" to the build path of the getting started 2 project.

Your project is now able to use the debug system.

Chapter 2

Use

The debug system allows you to follow the proceedings of the simulation by printing in the console all kind of information. Having a lot of information printed could slow down the simulation, so we made it activation and disabling easy.

2.1 Levels

SLF4J provides five message levels. A level adds a specific semantic to the message:

- **ERROR:**for error and exception messages which can produce the stopping of the system
- **WARN:** for warning messages which reports unusual or unexpected behaviors
- **INFO:** for messages which reports useful information in dev and production mode (Remote connection establishment, ...)
- **DEBUG:**for debugging messages which can be useful in the development process
- **TRACE:** is similar to DEBUG. It can be useful for printing extra details.

In Logback, the set of message levels is ordered. Logback enables to set a level. It allows that the message with this level and the lower levels are printed.

The next table exhibits the printed messages according to the configured level ($OFF < ERROR < DEBUG < INFO < DEBUG < TRACE$).

	Configured level					
Message level	TRACE	DEBUG	INFO	WARN	ERROR	OFF
TRACE	PRINTED					
DEBUG	PRINTED	PRINTED				
INFO	PRINTED	PRINTED	PRINTED			
WARN	PRINTED	PRINTED	PRINTED	PRINTED		
ERROR	PRINTED	PRINTED	PRINTED	PRINTED	PRINTED	

Table 2.1: Behavior of the configured level

For example, if the level is set to **INFO** then the messages with **INFO** or **WARN** or else **ERROR** level are printed.

2.2 Loggers

SLF4J enables to print messages through named loggers. A logger should be created for each specific component or entity of the software. This is where the name parameter of the constructor

is mainly used. When printing, the name will be use and allow the user to know exactly which component is described.

```
410004 [agentZ] TRACE mecsyco.magent.simulation.agentZ -- at 99.99000000001425 from X process Tuple(2.58548295903819; X)
410005 [agentX] DEBUG mecsyco.artifact.coupling.centralized.nameless -- Link time update (100.00000000001425)
410006 [agentZ] TRACE mecsyco.magent.simulation.agentZ -- at 99.99000000001425 from Y process Tuple(3.1153759160666223; Y)
410007 [agentX] TRACE mecsyco.magent.simulation.agentX -- at 99.99000000001425 from Y process Tuple(3.1153759160666223; Y)
410008 [agentY] TRACE mecsyco.magent.simulation.agentY -- at 99.99000000001425 from X process Tuple(2.58548295903819; X)
410009 [obs] TRACE mecsyco.magent.observing.obs -- at 99.99000000001425 from X process Tuple(2.58548295903819; X)
410010 [obs] TRACE mecsyco.magent.observing.obs -- at 99.99000000001425 from Y process Tuple(3.1153759160666223; Y)
410011 [agentX] INFO mecsyco.artifact.interface.model.Xmodel -- end simulation at time 99.99000000001425
410012 [agentY] INFO mecsyco.artifact.interface.model.Ymodel -- end simulation at time 99.99000000001425
410013 [obs] TRACE mecsyco.magent.observing.obs -- at 99.99000000001425 from Z process Tuple(21.21287562541792; Z)
410014 [agentZ] INFO mecsyco.artifact.interface.model.Zmodel -- end simulation at time 99.99000000001425
410015 [obs] INFO mecsyco.artifact.interface.observing.comparator -- error rate: 0.0
410016 [obs] INFO mecsyco.artifact.interface.observing.comparator -- 30000.0 Data Compared
410017
```

Figure 2.1: Debug system level "all" and FILE_LOGGING_ENABLED for getting started 2

Logback uses a hierarchical naming system. Each division is separated by a dot. For example, **mecsyco** is a parent of **mecsyco.magent**. The set of logger is then partially ordered ($mecsyco.magent < mecsyco$)
Logback enables to set a level per logger. If a level is not specified then the logger inherits of the level of its parent. For example, if 'mecsyco.magent' has no level, then it inherits of the level of mecsyco.
The loggers without parents (the loggers without dot in their name) have a default parent called root.

Chapter 3

Configuration

A default configuration is present in the "conf" folder. Please don't change the file names.

Use textbfconf/mecsyco.properties to set basic printing configurations. Short comments are present in the configuration file as reminders.
By default the messages are only printed into the standard output (console). If FILE_LOGGING_ENABLED is uncommented, they are also printed in a file (don't forget to change the folder path, or to create a folder named "log" in your project root folder).

We propose the settings of the level of three set of loggers:

- **M_AGENT_LOGGING_LEVEL:** assigns a level of logging to mecsyco.magent
- **INTERFACE_LOGGING_LEVEL:** assigns a level of logging to mecsyco.artifact.interface
- **COUPLING_LOGGING_LEVEL:** assigns a level of logging to mecsyco.artifact.coupling

If a variable is not defined with a logging level, it inherits of ROOT_LOGGING_LEVEL.

A level can be assigned to a logger in the file **conf/logback-loggers.xml**. For example
$< loggername = "mecsyco.artifact.coupling.centralized" level = "debug" / >$
assigns the level **DEBUG** for all centralized coupling artifact.

conf/logback-test.xml is the plain Logback configuration. It loads the previous configuration files.

3.1 Create new messages

In the case of the message provided are not sufficient enough, you can add you own message for the level required. The message will be printed each time the algorithm go through (same behaviors than *System.out.print*).

Since MECSYCO is multi-threaded, you should avoid to use static field for a logger. Prefer create a logger per instance of the component that you wish to log. The following code creates a logger named "name":
Logger l = LoggerFactory.getLogger("name");
To log a message with the level 'INFO':
l.info("message");
Instead of concatenate strings, prefer use the built-in solution:
l.info("first: {} second: {}", 1, 2); (or with concatenation *l.info("first: "+1+"second: "+2);*)