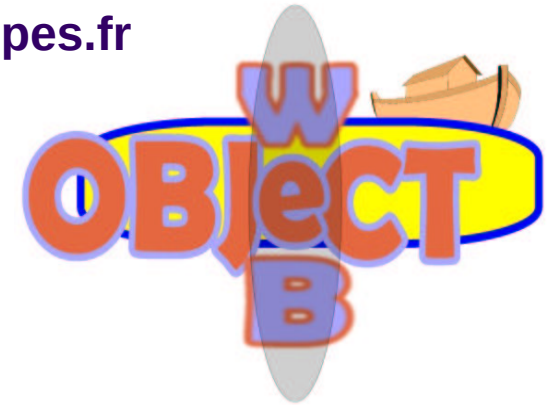
MonoLog

Monitoring & Logging

Sebastien.chassande@inrialpes.fr



Summary

I. Goals

II. Architecture

III. Intrumentation

IV. Log4j & JavaLog wrapper

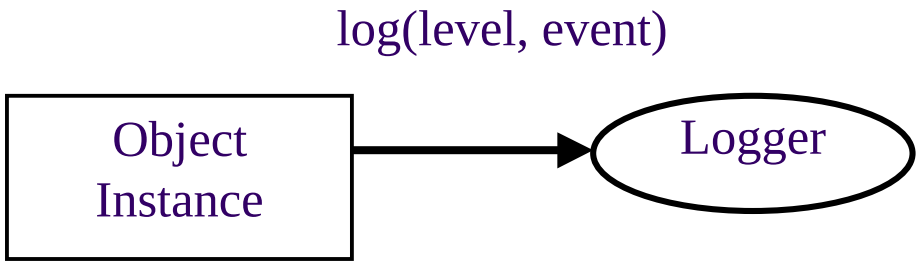
V. Packaging

I. Goals

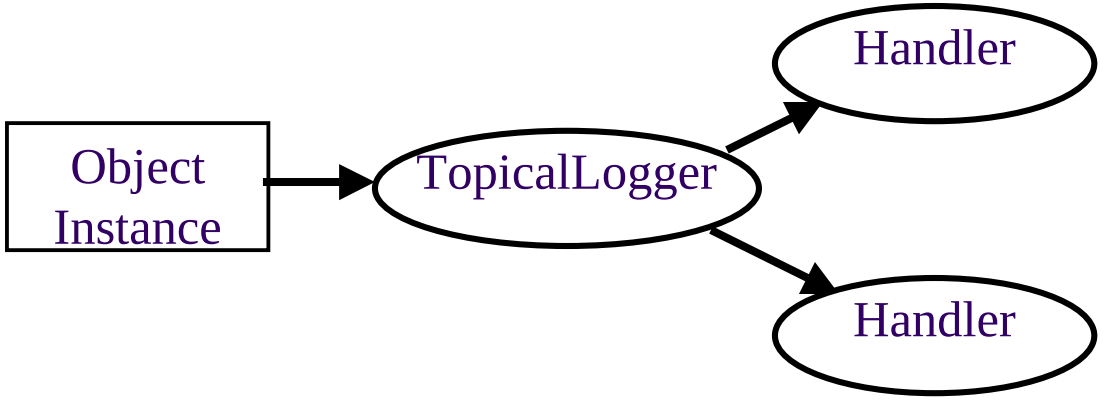
- Dissociate source code instrumentation to a logging implementation.
- Support a component architecture.
- Same API for logging and monitoring.
- Efficient implementation of logging.
- Support internationalisation

II. Architecture: Overview

- Objects make logging calls



- TopicalLogger
  = Logger extension
  = event router



- Handler ~ output

II. Architecture: Level

- Level Interface
- Represents the event importance
- Five predefined levels (in BasicLevel class):
  - FATAL
  - ERROR
  - WARN
  - INFO
  - DEBUG
- = Log4j priority
- Flyweight pattern to reduce cost (Integer)

II. Architecure: Handler

- **Represents an output**
  (Can be a file output, a console output,..)

- **Empty interface ~ tag**

II. Architecure: Logger

- **Extends Handler**
- **Interface for the instrumentation**
- **Log methods:**
  - log(Level l, Object event)
  - log(Level l, Object event, Throwable t)
  - log(Level l, Object event, Object instance, Object method)
  - log(Level l, Object event, Throwable t, Object instance, Object method)
  - + Same method with int instead Level
- **Level assessors : get/set/isLoggable (Level & int)**

II. Architecture: TopicalLogger

- Interface, extends Logger (Handler too)
- Linked to the configuration aspect
- Manages Handler list (add/remove)
- Is a named entity (dotted java.lang.String)
  => Hierachical namespace
  => inherits parent Level & parent Handlers
- Multiple names (add/remove)
  => support the sharing component

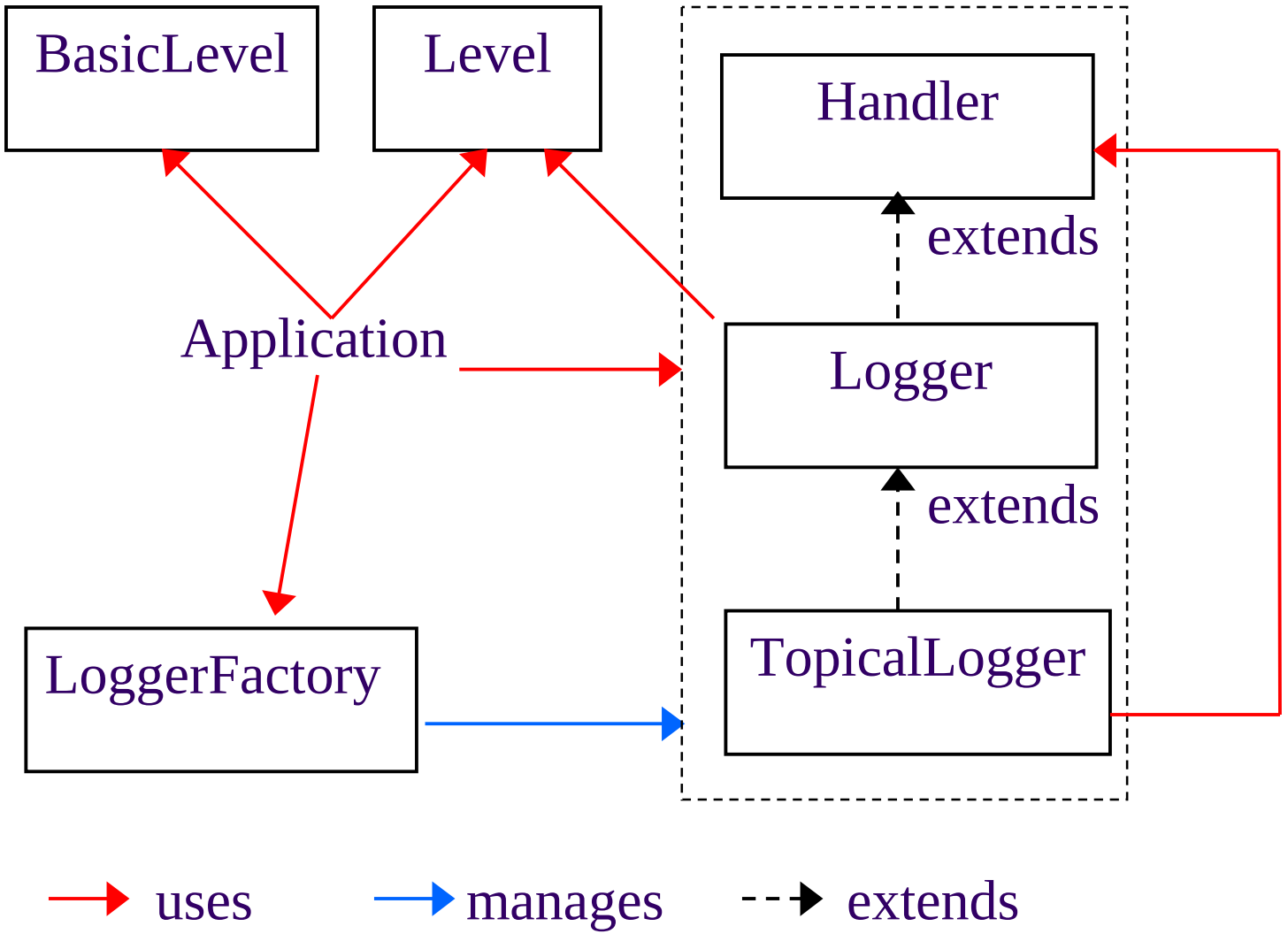
II. Architecture: LoggerFactory

- **Interface to fetch a Logger (TopicalLogger too)**

- **Methods:**
  - Logger getLogger(String key)
    key = initial topic (package or class name)
  - Logger getLogger(String key, String resourceBundleName)
    to support internationalisation

- **How to found a LoggerFactory ?**
  - Component initialisation
  - static variable in near class ~ LoggerFactory Manager
    Provide a Default LoggerFactoryManager

II. Architecture: Summary



Roles:

	Role
Handler	Empty
Logger	Log methods
TopicalLogger	Manages Handler & topic
LoggerFactory	Provides Logger

Aims:

- Application uses the interface for the instrumentation
- Interface names represent clearly the inheritance structure
- Respect the java log terminology

II. Architecture: comparaison with java log & log4j

monolog	Java log	Log4j
Handler	Handler	Appender
Logger	Logger	Category
TopicalLogger		
LoggerFactory	LogManager	CategoryFactory

Aspect separation

III. Intrumentation(1/2): header

Imports

```
import org.objectweb.util.monolog.api.BasicLevel;
import org.objectweb.util.monolog.api.Logger;
import org.objectweb.util.monolog.api.LoggerFactory;
```

Get the Monitor

```
LoggerFactory lf = … // component initialisation
static Logger l = lf.getLogger("org.ow.toto");
```

Boolean to delete instrumentation code at compilation time

```
static boolean trace = true; //false
```

III. Instrumentation (2/2)

In source code

```
if ( trace && l.isLoggable(BasicLevel.DEBUG) )
    l.log(BasicLevel.DEBUG, ...);
```

or

```
if ( m.isLoggable(BasicLevel.DEBUG) )
    l.log(BasicLevel.DEBUG, ...);
```

or

```
l.log(BasicLevel.DEBUG, ...);
```

IV. Log4j & JavaLog wrapper

- Easy levels mapping
- Easy methods mapping
- Easy unique name mapping

- One LoggerFactory by wrapper
- Must implement multiple name property

V. Packaging

- **Package name**
  - org.objectweb.util.monolog.api
  - org.objectweb.util.monolog.wrapper.log4j
  - org.objectweb.util.monolog.wrapper.javalog
- **Jars**
  - ow_util_log_api.jar
  - ow_util_log_wrp_log4j.jar
  - ow_util_log_wrp_ javaLog.jar

do not integrated to objectweb project