

Allow redirection to alternate logging API

Motivation:	Allow redirection to alternate logging API
Contact:	Andrea Aime Martin Desruisseaux
Tracker:	http://jira.codehaus.org/browse/GEOT-1545
Tagline:	You are debugging GeoTools, in a J2EE application, do you know where your logs are?

Description

GeoTools currently uses Java Logging directly and allows for redirection to other logging systems using a handler, `CommonHandler`, redirecting log statements to commons logging, but not allowing to control the log level accurately. The downsides of this approach are discussed in these two threads:

- [Considerations on GeoServer logging](#)
- [Why our current logging approach will never work in Jetty](#)

The proposal here is to extend the `org.geotools.util.Logging` class with the following method:

```
public class Logging {
    /**
     * Convenience method that redirects the
     * call to the current log factory
     */
    public static Logger getLogger(String
name);
}
```

And provide a global configuration option to change the kind of Logger produced:

```

public class GeoTools {
    /**
     * Sets a logger factory. By default
     * GeoTools starts using the Java logging.
     */
    public void
    setLoggerFactory(LoggerFactory factory);
}

```

Where the `LoggerFactory` interface generates the user desired `Logger`:

```

public interface LoggerFactory {
    public Logger getLogger(String name);
}

```

This factory class will return the most appropriate `Logger` implementation depending on the redirection set up:

- when no redirection is in place, a standard `java.util.logging.Logger` will be returned.
- when asking for redirection onto Log4j an special subclass of `Logger` will be returned that delegates everything to a wrapped Log4j logger, bypassing completely the java logging subsystem, and thus using directly handlers and logging levels as configured in Log4j. This is similar to providing an implementation of `org.apache.commons.logging.Log` interface except that instead of implementing every methods in an Apache's interface, we override every methods in a core Java class. More subclasses may be created in order to setup other redirections.

In order to have the above work, all `GeoTools` classes will have to switch to use `org.geotools.util.Logging.getLogger(name)` (this can be done by performing a search and replace on files). Applications depending on `GeoTools` will be able to use the same redirection mechanism.

The initial implementation will introduce two log factories:

- `CommonsLogger.Factory` that will generate Commons-logging `Log` instances. These will delegate to Commons-logging;
- `Log4JLogger.Factory` that will generate Log4J `Logger` instances. These will delegate directly to Log4J;
- (not really an implementation) `null`, which is supposed to be the default, means to generate java logging loggers.

Users are then free to add their own redirection factories mimicking the code provided in `CommonsLogger.Factory` and `Log4JLogger.Factory`

Status

This proposal is ready for discussion.

Voting:

- [Andrea Aime](#) +1
- [Ian Turton](#)
- [Justin Deoliveira](#) -1 [Alternative Use Commons Logging](#)
- [Jody Garnett](#) +1
- [Martin Desruisseaux](#) +1
- [Simone Giannecchini](#)

Tasks

	no progress	✓	done	✗	impeded	⚠	lack mandate /funds/time	?	volunteer needed
--	-------------	---	------	---	---------	---	--------------------------	---	------------------

GeoTools 2.5 and 2.4:

- ✓ [Martin Desruisseaux](#) [Andrea Aime](#) Create Loggers and Log4jLogger
- ✓ [Martin Desruisseaux](#) [Andrea Aime](#) | API changed based on BEFORE / AFTER
- ✓ [Martin Desruisseaux](#) Deprecate `CommonLogger` and Logging methods associated to it!
- ✓ [Martin Desruisseaux](#) Update developers guide

API Changes

BEFORE

This example shows how redirection to Log4J is setup and how logging is used.

The example talks about commons logging, but since commons logging by default can redirect to java logging (no point in doing that) or to Log4J, we assume that commons logging is really used to redirect logging to Log4J.

```
// on gt2 startup
Logging.GEOTOOLS.redirectToCommonsLogging();

// generic java class
import java.util.logging.logger;
...
private static final Logger LOGGER =
    Logger.getLogger("org.geotools.rendering");
```

AFTER

```
// on gt2 startup
Logging.setLogFactory(new
Log4JLogFactory());

// generic gt2 class
import java.util.logging.Logger;
import org.geotools.util.Logging;
...
private static final Logger LOGGER =
Logging.getLogger("org.geotools.rendering");
```

Documentation Changes

- [02 Integration Basics](#) - including logging redirection in the steps needed to run GeoTools inside a J2EE environment
- [Logging](#) - example of how to configure logging.
Optional: provide a note explaining the why of the GeoTools logger factory with a link to Integration Basics
- [5.1.3 Logging](#) - code example of what must be done inside the library.
Optional: provide a note explaining the why and how of the GeoTools logger factory.