How to Use the Logging and Tracing

General Rules:

- Keep in mind the difference between Logs and Traces. Logs are addressed to an administrator of a customer system, Traces to the developer and the support organization. An administrator typically is not interested in the details of our servers' architecture but in areas according to his administration tasks. A developer on the other hand wants to see the details of the control flow.
- Logs are written to Categories and Traces to Locations.
- During normal operation, developers and supporters are not looking at a running system. On the other hand the administrator has to check the system regularly. This implies that Traces do not have to be shown during normal operation, but Logs do.
- For performance reasons only traces of severity WARNING and logs of severity INFO or higher are written during normal operation.
- To emphasize it again: Write into **Logs** only such messages, which are important for an administrator, who supervises the system during normal operation. Write into **Traces** everything that might be important to trace erroneous behavior.
- In the case of an exception, a Trace should always be written immediately before the exception is thrown.
- If an exception is caught a Log must be written, if and only if the exceptional occurrence has to be noticed by an administrator. If the exception is a potential troublemaker, but the catching component proceeds anyway, a Trace record with WARNING severity is required

Preparation:

1. Log records are written to Categories. Use one of the following predefined categories:

   ```
   com.sap.engine.lib.logging.LoggingHelper.SYS_DATABASE;
   com.sap.engine.lib.logging.LoggingHelper.SYS_NETWORK;
   com.sap.engine.lib.logging.LoggingHelper.SYS_SERVER;
   com.sap.engine.lib.logging.LoggingHelper.SYS_SECURITY;
   ```

   (For the definition of further categories contact Gregor Frey**.)**

2. Trace records are written to Locations. In every class create a reference to a Location, which corresponds to the fully qualified name of this class:

   ```
     private static final Location LOCATION =
       Location.getLocation(Locking.class);
   ```

3. Use a static String object to hold the name of the method.

   ```
     String METHOD = "createLock(String)";
   ```

Usage:

1. If you have a message that is important for the customer or admin, write a log record:

   ```
   CATEGORY.infoT(LOCATION, METHOD,
      "User {0} logged in", new Object[] { user });
   ```

   For the other severities use the corresponding methods warningT, errorT, fatalT.

2. If you caught an Exception that is important for the customer or admin, write a log record:

   ```
   LoggingHelper.logThrowable(severity, CATEGORY, LOCATION, METHOD,
       throwable);
   ```

   The severity can be ERROR, FATAL.

3. If you have a message that might be useful for the developer or support, write a trace record:

   ```
   LOCATION.infoT(METHOD, "User {0} started transaction",
       new Object[] { user });
   ```

   For the other severities use the corresponding methods debugT, pathT, warningT, errorT.

4. If you throw an Exception that might be useful for the developer or support, write a trace record:

   ```
   LOCATION.throwing(METHOD, exception);
   ```

5. If you caught an Exception that might be useful for the developer or support, write a trace record:

   ```
   LoggingHelper.traceThrowable(severity, LOCATION, METHOD,
       throwable);
   ```

   The severity can be DEBUG, PATH, INFO, WARNING, ERROR.

6. For non-trivial methods it is often convenient to trace the entry and the exit of the method. If you do trace the entry it is urgently required that you trace the exit too! If your method might throw an exception, you have to trace the exit in a finally clause.

   ```
   LOCATION.entering(METHOD);
   LOCATION.exiting(METHOD);
   ```

Configuration:

1. By default logs are written if the severity is INFO, WARNING, ERROR, FATAL
2. By default traces are written if the severity WARNING, ERROR
   => if you have Exceptions, which you always want to trace, use WARNING, ERROR
3. You can change the defaults within a descriptors/<service>/log-configuration.xml file or during runtime using the log-configurator service.

Performance remarks:

1. If you have parameters, never build Strings yourself, always use the {n}-replacement
2. If you have to make calculations for traces, check first if the tracing is active …
   LOCATION.beLogged(severity)

Further readings:

1. For a general introduction:
   Tutorial on Logging and Tracing

Remarks:

The LogHelper class, the predefined Categories, the logThrowable and the traceThrowable methods are currently only available in the J2EE environment.