LoggingDetails

TCK20 Logging

A TCK run involves several tools/components. Each of these components use standard or proprietary logging implementations:

- **Derby** uses proprietary logging.
- **JPOX** uses Log4J.
- **SpringFramework** (called by TCK) uses Apache commons-logging.
- **TCK testcases** use Apache commons-logging.
- **TCK JUnit** result logging uses proprietary logging.
- **TCK result summary** uses proprietary logging.

Log files are written under a common root directory `tck20/target/logs`. All involved tools/components above are configured to write to a separate subdirectory. When you run the TCK, the following directories are created:

- `.../logs/database/`
- `.../logs/enhancer/`
- `.../logs/<timestamp>/`

Each of these directories contains one or more log files. The log files in the database and enhancer subdirectories are configured to append log messages for consecutive TCK runs. In contrast, the `<timestamp>` directory is always created for each TCK run. The naming pattern is `yyyyMMdd-HHmmss`.

Below, log files for all involved tools/components are listed:

- Derby writes log messages to `.../logs/database/derby.txt`.
- JPOX enhancer writes log messages to directory `.../logs/enhancer/`. Two log files are written, one for each identity type. The naming pattern is `<identity type>-jpox.txt`.
- JPOX runtime writes log messages to the `<timestamp>` directory. Several log files are written for each TCK configuration, one per database and one per identity type. The naming pattern is `<database>~~<identity type>~~<TCK configuration>-jpox.txt`.
- TCK test cases write log messages to the `<timestamp>` directory. Several log files are written for each TCK configuration, one per database and one per identity type. The naming pattern is `<database>~~<identity type>~~<TCK configuration>-tck.txt`.
- TCK writes JUnit result output to the `<timestamp>` directory. Several log files are written for each TCK configuration, one per database and one per identity type. The naming pattern is `<database>~~<identity type>~~<TCK configuration>-junit.txt`.
- TCK writes a result summary file for each run. This file contains information about each TCK configuration per database and per identity type. It is written to the `<timestamp>` directory. The file name is `TCK-results.txt`.

TCK20 Logging Configuration Files

Each of the involved logging implementations use different properties files configuring logging. These properties files are located in directory `tck20/test/conf`:

- Derby uses `derby.properties`.
- Log4J uses `log4j.properties`.
- Apache commons-logging uses `common-logging.properties`.

The TCK junit result logging and the TCK result summary logging are not configurable.

There is another properties file configuring JDK 1.4 logging. This file may be used by implementations under test (iut). It is also used if Apache commons-logging is configured to run with JDK 1.4 logging.

TCK20 Logger Instances

The TCK uses the following logger instances:

Logger Instance	Log Level	Component
org.apache.jdo.tck	INFO	TCK test cases
org.apache.jdo.tck.pc.company.Company{{`Model}} {{Reader	ERROR	Spring}} `Framework
org.springframework	ERROR	SpringFramework

The log level of logger instance `org.apache.jdo.tck` defaults to `INFO`. Both remaining logger instances are used by Spring{{`Framework. The log levels of these logger instances default to }}ERROR`.

***Note**: Spring{{`Framework classes write log messages using log level }}INFO{{. Since we do not want to see Spring}}Framework }}INFO{{ log messages, we set the log level of SpringFramework logger instances to }}ERROR{{. Due to the fact that TCK class }}org.apache.jdo.tck.pc.company.Company{{`ModelReader extends a Spring{{`Framework class which retrieves a logger instance calling }}LogFactory.getLog(getClass()), we have to define a logger instance on class }}org.apache.jdo.tck.pc.company.Company{{`ModelReader.*

TCK20 Logging Configuration for JDO Vendors

JDO implementations using Log4J or JDK 1.4 logging may use specific file appender or file handler implementations of tck20 in order to write logging output to directory `tck20/target/logs/<timestamp>/`. This may be achieved by editing file `tck20/test/log4j.properties`:

```
# log4j vendor-specific appender
log4j.appender.<vendor> = org.apache.jdo.tck.util.TCKFileAppender
log4j.appender.<vendor>.File = <vendor>.txt
```

or by editing file `tck20/test/logging.properties`:

```
# JDK 1.4 vendor-specific handler
handlers = org.apache.jdo.tck.util.TCKFileHandler
org.apache.jdo.tck.util.TCKFileHandler.fileName = <vendor>.txt
org.apache.jdo.tck.util.TCKFileHandler.level = FINEST
```

JDO implementations using other logging implementations may use a static public tck20 method to retrieve the name of the logging file to be generated in directory `tck20/target/logs/<timestamp>/`:

```
    org.apache.jdo.tck.util.BatchTestRunner#changeFileName(String fileName)
```

This method returns a file name which is constructed by values of some system properties appended by the given file name. The system properties are:

- `jdo.tck.log.directory`: Specifies the directory for the file.
- `jdo.tck.database`, `jdo.tck.cfg`: The values of these properties are used to construct the file name.
- `jdo.tck.identitytype`: The value of this property is replaced by `app` if it equals `applicationidentity`, else it is replaced by `dsid`.

The returned file name is constructed as follows:

```
    <jdo.tck.log.directory>/<jdo.tck.database>-<jdo.tck.identitytype>-<jdo.tck.cfg>-<given file name>
```

Values of properties which do not exist default to `""`.

Apache commons-logging Configuration

Apache commons-logging allows switching between different logging implementations (including JDK1.4 logging, Log4J and Apache's simple logging implementation). There are three properties files to configure logging:

Properties File	Description
common-logging. properties	Specifies the logging implementation to use.
logging.properties	Logger configuration when using JDK 1.4 logging.
log4j.properties	Logger configuration when using Log4J logging.
simplelog.properties	Logger configuration when using Apache simple logging.

Log Level Mapping between JDK 1.4 and Apache commons-logging

The following table describes the mapping between the log level of JDK 1.4 logging and Apache commons-logging:

JDK 1.4	Apache commons-logging
FINEST	trace
FINE, FINER	debug
INFO, CONFIG	info
WARNING	warn
SEVERE	error, fatal

TCK11 and RI11 Logger Instances

The ri11 implementation uses the following logger instances:

Logger Instance	Log Level	Component
org.apache.jdo.impl.fostore	File Object Store implementation	
org.apache.jdo.impl.jdoql	JDOQL query runtime	
org.apache.jdo.impl.jdoql.jdoqlc	JDOQL query compiler	
org.apache.jdo.impl.model.jdo	JDOModel implementation	
org.apache.jdo.impl.model.jdo.xml	XML parser for JDO metadata files	
org.apache.jdo.impl.pm	PM and PMF implementation	
org.apache.jdo.impl.sco	SCO implementation	
org.apache.jdo.impl.state	State manager implementation	
org.apache.jdo.store	Generic store manager implementation	
org.apache.jdo.util	Utility classes	

The ri11 test classes use the following logger instance:

```
org.apache.jdo.test
```

The tck11 test classes use the following logger instance:

```
org.apache.jdo.tck
```

Log Level Mapping between RI11 Logging and Apache commons-logging

The following table maps the log level of the former JDORI util.Logger class to Apache commons-logging log level:

JDORI Logger	Apache commons-logging
TIME, BUF	trace
DEBUG, TEST	debug
INFO	info
WARN	warn
ERROR	error