Specifications of Data Repository V2

The Data Repository manages data sets. A data set is either a file or a folder with all its files and subfolders. The Data Repository should be able to handle large data sets. A large data set can be a file of size several GB and/or a folder with thousands of small files or both.

The Data Repository is a command line tool written in Java 7. It is the front end to the repository of data sets. The basic commands are:

- add: Adds a data set to the repository.
- replace: Replaces a data set in the repository.
- delete: Deletes a data set in the repository.
- export: Exports a data set of the repository to a specified directory.
- list: Lists meta data of data sets in a tabular form.
- server: Runs in server mode: Moving data sets from an incoming directory into the repository.

All commands have the following behavior in common:

- They do not read from standard input. This allows to use the tool in any shell script.
- Exit value will be 0 in case of success and 1 in case of an error.
- Informations are printed on standard output.
- Error messages are printed on standard error. They are human-readable and explain the most likely reason of failure.
- Commands with no explicit output on the console should print a success message summarizing what they performed.
- Commands with `-verbose` option print progress information onto the standard output if the option is set.

1 Add Command

1.1 Synopsis

```
data-repository add [--description <description>] [--move] [--verbose] \
                        <repository path> <file/folder>
```

1.2 Parameter and Options

`<repository path>` Absolute or relative path to the repository.

`<file/folder>` Absolute or relative path to the file or folder defining the data set to be added.

`--description <description>` A description of the data set. The maximum length is restricted to 1000 characters. It does not contain any ISO control character like TAB, CR, or LF.

`--move` The data set is *moved* into the repository. Otherwise it is *copied*.

`--verbose` Shows progress information in case of copying mode. The number of bytes already copied versus the total number of bytes to be copied will be shown.

1.3 Description

The specified file/folder will be added to the repository. Without the `--move` option a copy will be created. With the `--move` option the file/folder is moved completely into the repository. This allows to handle large data sets without copying the data which can lead to large time and memory consumption. A unique identifier is created for the data set. It is part of the success message.

The repository folder will be created if it does not exist.

If the same file/folder is added twice to the same repository it will be stored as two different data sets.

2 Replace Command

2.1 Synopsis

```
data-repository replace  [--description <description>] [--move] [--verbose] \
                         <repository path> <data set identifier> <file/folder>
```

2.2 Parameter and Options

`<repository path>` Absolute or relative path to the repository.

`<data set identifier>` Identifier of the data set to be replaced.

`<file/folder>` Absolute or relative path to the file or folder defining the data set to be added.

`--description <description>` A description of the data set. The maximum length is restricted to 1000 characters. It does not contain any ISO control character like TAB, CR, or LF.

`--move` The data set is *moved* into the repository. Otherwise it is *copied*.

`--verbose` Shows progress information in case of copying mode. The number of bytes already copied versus the total number of bytes to be copied will be shown.

2.3 Description

Replaces the data set of specified identifier completely by the specified file/folder. If no description has been specified the old description is kept. This command behaves like a sequence of a `delete` and `add` command.

3 Delete Command

3.1 Synopsis

```
data-repository delete [--id <identifier>] [--name <name>] [--text <text snippet>] \
                       [--before <time stamp>] [--after <time stamp>] \
                       <repository path> [<data set identifier>]
```

3.2 Parameter and Options

`--id <identifier>` Data set identifier.

`--name <name>` Data set name.

`--text <text snippet>` Text contained in name or description.

`--before <time stamp>` Time stamp in the format `YYYY-MM-DD` or `YYYY-MM-DD HH:MM:SS`.

`--after <time stamp>` Time stamp in the format `YYYY-MM-DD` or `YYYY-MM-DD HH:MM:SS`.

`<repository path>` Absolute or relative path to the repository.

`<data set identifier>` Identifier of the data set to be deleted. This parameter has to appear if no option has been specified. If an option has been specified it is not allowed.

3.3 Description

Delete data sets completely from the repository.

There are three ways to specify the set of data sets to be deleted:

1. No options have been specified: The parameter `<data set identifier>` is mandatory and specifies the data set to be deleted. An error occurs if no data set with specified identifier exists.
2. Option `--id` has been specified: The specified data set will be deleted. All other options and the parameter `<data set identifier>` are not allowed. An error occurs if no data set with specified identifier exists.
3. One or several of the following options restrict the set of data sets to be deleted:

`--name` Name of the data set.

`--text` Text snippet appearing in either the name or the description of the data set.

`--before` Data sets with a timestamp before the specified timestamp.

`--after` Data sets with a timestamp after the specified timestamp.

Data sets fulfilling all conditions are deleted. The parameter `<data set identifier>` is not allowed. The success message tells how many data sets have been deleted. It is valid that no data set is deleted which lead also to a success message.

4 Export Command

4.1 Synopsis

```
data-repository export [--id <identifier>] [--name <name>] [--text <text snippet>] \
                       [--before <time stamp>] [--after <time stamp>] [--verbose] \
                       <repository path> [<data set identifier>] <destination folder>
```

4.2 Parameter and Options

`--id <identifier>` Data set identifier.

`--name <name>` Data set name.

`--text <text snippet>` Text contained in name or description.

`--before <time stamp>` Time stamp in the format `YYYY-MM-DD` or `YYYY-MM-DD HH:MM:SS`.

`--after <time stamp>` Time stamp in the format `YYYY-MM-DD` or `YYYY-MM-DD HH:MM:SS`.

`--verbose` Shows on the standard output progress information. The number of bytes already copied versus the total number of bytes to be copied will be shown every second.

`<repository path>` Absolute or relative path to the repository.

`<data set identifier>` Identifier of the data set to be exported. This parameter has to appear if no option has been specified. If an option has been specified it is not allowed.

`<destination folder>` Path to the destination folder.

4.3 Description

Copies data sets to the destination folder specified by an absolute or relative path. The exported files/folders have the original name. An error occurs if there is already a file/folder of the same name in the destination folder.

There are three ways to specify the set of data sets to be deleted:

1. No options have been specified: The parameter `<data set identifier>` is mandatory and specifies the data set to be exported. An error occurs if no data set with specified identifier exists.
2. Option `--id` has been specified: The specified data set will be exported. All other options and the parameter `<data set identifier>` are not allowed. An error occurs if no data set with specified identifier exists.
3. One or several of the following options restrict the set of data sets to be deleted:

`--name` Name of the data set.

`--text` Text snippet appearing in either the name or the description of the data set.

`--before` Data sets with a timestamp before the specified timestamp.

`--after` Data sets with a timestamp after the specified timestamp.

Data sets fulfilling all conditions are exported if all data sets have different names. If at least two data sets have the same name an error occurs. The error message lists the identifiers of all data sets which could not be exported because of same name. The parameter `<data set identifier>` is not allowed. The success message tells which data sets (by identifier and name) have been exported. It is valid that no data set is exported which lead also to a success message.

5 List Command

5.1 Synopsis

```
data-repository list [--id <identifier>] [--name <name>] [--text <text snippet>] \
                        [--before <time stamp>] [--after <time stamp>] \
                        <repository path>
```

5.2 Parameter and Options

`--id <identifier>` Data set identifier.

`--name <name>` Data set name.

`--text <text snippet>` Text contained in name or description.

`--before <time stamp>` Time stamp in the format `YYYY-MM-DD` or `YYYY-MM-DD HH:MM:SS`.

`--after <time stamp>` Time stamp in the format `YYYY-MM-DD` or `YYYY-MM-DD HH:MM:SS`.

`<repository path>` Absolute or relative path to the repository.

5.3 Description

Lists the meta data as a TAB-separated table onto the standard output. This is a tabular text format where each row is separated by a newline character. Cells in a row are separated by TAB characters.

- The first row contains always the following header names:

```
ID
Name
Timestamp
Number of Files
Size
Description
```

- The name is the original file/folder name.
- The table is sorted in accordance to the timestamp.
- If no description has been specified in the add command an empty string is printed.
- The number of files also includes directories.
- The size is the sum of the sizes of all files in bytes.
- The timestamp is defined by the date and time of adding/replacing the data set. It is shown in the following format: `YYYY-MM-DD HH:MM:SS`.
- In case of an empty or non-existing repository only the header line is printed.
- The result will be filtered by the provided options:

`--id` Shows only the data set with specified identifier.

`--name` Shows only data sets with specified name.

`--text` Shows only data sets where the name or the description contains the specified text snippet.

`--before` Shows only data sets with a timestamp before the specified timestamp.

`--after` Shows only data sets with a timestamp after the specified timestamp.

- In case of option `--id` other options will lead to an error.
- Each option restricts the result. That is, only data sets fulfilling all options are shown.
- If no option is present all data sets are listed.

6 Server Command

6.1 Synopsis

```
data-repository server <repository path> <properties file>
```

6.2 Parameter and Options

`<repository path>` Absolute or relative path to the repository.

`<properties file>` Java properties file with configuration parameters as described below.

6.3 Description

Starts the application in a mode where it is constantly observing an incoming directory for new data sets. If a new one is detected it will be added to the repository similar to the add command with option `--move`.

In addition an HTML file with an HTML table of same content as the table produced by the list command (without options) will be updated. This file is always recreated at server start up and contains data sets added previously.

All output is redirected to a log file after successful start up of the server. It contains log messages in the format: `<time stamp> <log level> <log message>`. Where `<time stamp>` is the actual date and time in the same format as for the list command. The `<log level>` is either `[INFO]` or `[ERROR]`. The log entries are on one line except error logs with stack traces. The first log entry shows the version of the Data Repository followed by entries for each value of the configuration parameters.

If the server is successfully started a success message is printed onto standard output. Otherwise an error message is printed and the server is stopped. The running server is stopped by pressing ctrl-c on the console.

Because dropping a data set into the incoming directory might not be instantaneous for a large data set (many files and/or large files) an algorithm is needed to detect completeness of a data set file/folder. The application provides three completeness detection algorithms (described below) from which one has to be chosen.

A little API allows to add customized completeness detections later on. Classes for those algorithms are expected in an additional JAR file. It can be dropped into some defined location inside the installed application (e.g. in the same folder as the JAR file of the application). After restart of the server the new completeness detection is available.

The configuration parameters are read from a properties file (for format specifications see Javadoc of the load method of class Properties). The following properties are expected:

`incoming-directory:` Absolute or relative path to the incoming directory. Existence will be

checked at server start up.

`html-overview:` Absolute or relative path to the file which contains a complete static HTML page with an HTML table with the same content produced by the optionless list command. If not specified no overview will be created.

`log-file:` Absolute or relative path to the log-file. If not specified it will be `server.log` inside the repository folder. If the log file doesn't exist it will be created. If it already exists new log entries are append.

`checking-interval-in-seconds:` The time interval (in seconds) the incoming directory will be scanned again after the last scan.

`completeness-detection.class-name:` Fully qualified name of a Java class implementing a completeness detection algorithm (a Java interface).

If a mandatory property is missing or invalid the server doesn't starts up but an error message is printed.

6.3.1 Completeness Detection

All properties starting with '`completeness-detection.`' are properties for the completeness detection algorithm. The three build-in algorithms and the properties they know are the following:

No Detection: This algorithm assumes that the data set folder/file is already complete when it appears in the incoming directory. Thus, no detection of completeness is needed.

Marker File Detection: A certain (empty) marker file found in the incoming directory indicates that the data set is complete. The name of the marker file is of the form `<prefix><file/folder name>`. The prefix is determined by the mandatory property `completeness-detection.prefix`. This marker file will be removed by the completeness checker. If there is no file/folder named `<file/folder name>` an error will be logged.

Unchanged Modification Date Detection: A data set is assumed to be complete if for any file of the data set the last-modified time stamp is older than the current time minus a quiet period specified by the mandatory property `completeness-detection.quiet-period-in-seconds`.

7 Help Command

7.1 Synopsis

```
data-repository [help] [command name]
```

7.2 Description

Prints onto standard output the version of the software in the first line followed by a list of all commands if the parameter `[command name]` is missing. For each command the name and a short description (should fit in one line) is printed. The user is also informed how to get more information for a command.

If the command name is specified a complete synopsis and brief description of the command is printed onto standard output.

In case of no command, options, or parameters the short help information will also be printed.