

Contents

1	NAME	2
2	SYNOPSIS	2
3	DESCRIPTION	3
4	Special Concepts	3
5	Methods	4

1 NAME

OpenSLX::MetaDB::Base - the base class for all MetaDB drivers

2 SYNOPSIS

```
package OpenSLX::MetaDB::coolnewDB;

use vars qw(@ISA $VERSION);
@ISA = ('OpenSLX::MetaDB::Base');
$VERSION = 1.01;

my $superVersion = $OpenSLX::MetaDB::Base::VERSION;
if ($superVersion < $VERSION) {
    croak _tr('Unable to load module <%s> (Version <%s> required)',
            'OpenSLX::MetaDB::Base', $VERSION);
}

use coolnewDB;

sub new
{
    my $class = shift;
    my $self = {};
    return bless $self, $class;
}

sub connectConfigDB
{
    my $self = shift;

    my $dbName = $openslxConfig{'db-name'};
    vlog(1, "trying to connect to coolnewDB-database <$dbName>");
    $self->{'dbh'} = ... # get connection handle from coolnewDB
}

sub disconnectConfigDB
{
    my $self = shift;

    $self->{'dbh'}->disconnect;
}

# override all methods of OpenSLX::MetaDB::Base in order to implement
# a full MetaDB driver
...
```

*The synopsis above outlines a class that implements a MetaDB driver for the (imaginary) database **coolnewDB***

3 DESCRIPTION

This class defines the MetaDB interface for the OpenSLX.

Aim of the MetaDB abstraction is to make it possible to use a large set of different databases (from CSV-files to a fullblown Oracle-installation) transparently.

While `OpenSLX::ConfigDB` represents the data layer to the outside world, each implementation of `OpenSLX::MetaDB::Base` provides a backend for a specific database.

This way, the different OpenSLX-scripts do not have to burden themselves with any DB-specific details, they just request the data they want from the ConfigDB-layer and that in turn creates and communicates with the appropriate MetaDB driver in order to connect to the database and fetch and/or change the data as instructed.

The MetaDB interface contains of four different parts:

- **basic methods (connection handling and utilities)**
- **data access methods (getting data)**
- **data manipulation methods (adding, removing and changing data)**
- **schema related methods (migrating between different DB-versions)**

In order to implement a MetaDB driver for a specific database, you need to inherit from `OpenSLX::MetaDB::Base` and implement the full interface. As this is quite some work, it might be wiser to actually inherit your driver from `OpenSLX::MetaDB::DBI`, which is a default implementation for SQL databases.

If there is a DBD-driver for the database your new MetaDB driver wants to talk to then all you need to do is inherit from `OpenSLX::MetaDB::DBI` and then reimplement `connectConfigDB` (and maybe some other methods in order to improve efficiency).

4 Special Concepts

Filters

A filter is a hash-ref defining the filter criteria to be applied to a database query. Each key of the filter corresponds to a DB column and the (hash-)value contains the respective column value.

[At a later stage, this will be improved to support a more structured approach to filtering (with boolean operators and hierarchical expressions)].

5 Methods

Basic Methods

The following basic methods need to be implemented in a MetaDB driver:

`connectConfigDB()`

Tries to establish a connection to the DBMS that this MetaDB driver deals with. The global configuration hash `%config` contains further info about the requested connection. When implementing this method, you may have to look at the following entries in order to find out which database to connect to:

`$config{'db-spec'}`

Full specification of database, a special string defining the precise database to connect to (this allows connecting to a database that requires specifications which aren't cared for by the existing `%config`-entries).

`$config{'db-name'}`

The precise name of the database that should be connected (defaults to 'openslx').

`disconnectConfigDB()`

Tears down the connection to the DBMS that this MetaDB driver deals with and cleans up.

`quote(string)`

Returns the given string quoted such that it can be used in SQL-statements (with respect to the corresponding DBMS).

This usually involves putting single quotes around the string and escaping any single quote characters enclosed in the given string with a backslash.

Data Access Methods

The following methods need to be implemented in a MetaDB driver in order to allow the user to access data:

`fetchVendorOSByFilter([%$filter], [$resultCols])`

Fetches and returns information about all vendor-OSes that match the given filter.

Param filter

A hash-ref containing the filter criteria that shall be applied - default is no filtering. See §?? for more info.

Param resultCols

A string listing the columns that shall be returned - default is all columns.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchVendorOSByID(@$ids, [$resultCols])`

Fetches and returns information the vendor-OSes with the given IDs.

Param ids

An array of the vendor-OS-IDs you are interested in.

Param resultCols

A string listing the columns that shall be returned - default is all columns.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchExportByFilter([%$filter], [$resultCols])`

Fetches and returns information about all exports that match the given filter.

Param filter

A hash-ref containing the filter criteria that shall be applied - default is no filtering. See §?? for more info.

Param resultCols

A string listing the columns that shall be returned - default is all columns.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchExportByID(@$ids, [$resultCols])`

Fetches and returns information the exports with the given IDs.

Param ids

An array of the export-IDs you are interested in.

Param resultCols

A string listing the columns that shall be returned - default is all columns.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchExportIDsOfVendorOS($id)`

Fetches the IDs of all exports that make use of the vendor-OS with the given ID.

Param id

ID of the vendor-OS whose exports shall be returned.

Return Value

An array of system-IDs.

`fetchSystemByFilter([%$filter], [$resultCols])`

Fetches and returns information about all systems that match the given filter.

Param \$filter

A hash-ref containing the filter criteria that shall be applied - default is no filtering. See §?? for more info.

Param \$resultCols [Optional]

A comma-separated list of column names that shall be returned. If not defined, all available data must be returned.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchSystemByID(@$ids, [$resultCols])`

Fetches and returns information the systems with the given IDs.

Param ids

An array of the system-IDs you are interested in.

Param resultCols

A string listing the columns that shall be returned - default is all columns.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchSystemIDsOfExport($id)`

Fetches the IDs of all systems that make use of the export with the given ID.

Param id

ID of the export whose systems shall be returned.

Return Value

An array of system-IDs.

`fetchSystemIDsOfClient($id)`

Fetches the IDs of all systems that are used by the client with the given ID.

Param id

ID of the client whose systems shall be returned.

Return Value

An array of system-IDs.

`fetchSystemIDsOfGroup($id)`

Fetches the IDs of all systems that are part of the group with the given ID.

Param id

ID of the group whose systems shall be returned.

Return Value

An array of system-IDs.

`fetchClientByFilter([%$filter], [$resultCols])`

Fetches and returns information about all clients that match the given filter.

Param \$filter

A hash-ref containing the filter criteria that shall be applied - default is no filtering. See §?? for more info.

Param \$resultCols [Optional]

A comma-separated list of column names that shall be returned. If not defined, all available data must be returned.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchClientByID(@$ids, [$resultCols])`

Fetches and returns information the clients with the given IDs.

Param ids

An array of the client-IDs you are interested in.

Param resultCols

A string listing the columns that shall be returned - default is all columns.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchClientIDsOfSystem($id)`

Fetches the IDs of all clients that make use of the system with the given ID.

Param `id`

ID of the system whose clients shall be returned.

Return Value

An array of client-IDs.

`fetchClientIDsOfGroup($id)`

Fetches the IDs of all clients that are part of the group with the given ID.

Param `id`

ID of the group whose clients shall be returned.

Return Value

An array of client-IDs.

`fetchGroupByFilter([%$filter], [$resultCols])`

Fetches and returns information about all groups that match the given filter.

Param `$filter`

A hash-ref containing the filter criteria that shall be applied - default is no filtering. See §?? for more info.

Param `$resultCols` **[Optional]**

A comma-separated list of column names that shall be returned. If not defined, all available data must be returned.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchGroupByID(@$ids, [$resultCols])`

Fetches and returns information the groups with the given IDs.

Param `ids`

An array of the group-IDs you are interested in.

Param `resultCols`

A string listing the columns that shall be returned - default is all columns.

Return Value

An array of hash-refs containing the resulting data rows.

`fetchGroupIDsOfClient($id)`

Fetches the IDs of all groups that contain the client with the given ID.

Param id

ID of the client whose groups shall be returned.

Return Value

An array of client-IDs.

`fetchGroupIDsOfSystem($id)`

Fetches the IDs of all groups that contain the system with the given ID.

Param id

ID of the system whose groups shall be returned.

Return Value

An array of client-IDs.

Data Manipulation Methods

The following methods need to be implemented in a MetaDB driver in order to allow the user to access change the underlying:

`addVendorOS(@$valRows)`

Adds one or more vendor-OS to the database.

Param valRows

An array-ref containing hash-refs with the data of the new vendor-OS(es).

Return Value

The IDs of the new vendor-OS(es), `undef` if the creation failed.

`removeVendorOS(@$vendorOSIDs)`

Removes one or more vendor-OS from the database.

Param vendorOSIDs

An array-ref containing the IDs of the vendor-OSes that shall be removed.

Return Value

1 if the vendorOS(es) could be removed, `undef` if not.

`changeVendorOS(@$vendorOSIDs, @$valRows)`

Changes the data of one or more vendor-OS.

Param vendorOSIDs

An array-ref containing the IDs of the vendor-OSes that shall be changed.

Param valRows

An array-ref containing hash-refs with the new data for the vendor-OS(es).

Return Value

1 if the vendorOS(es) could be changed, **undef** if not.

addExport(@\$valRows)

Adds one or more export to the database.

Param valRows

An array-ref containing hash-refs with the data of the new export(s).

Return Value

The IDs of the new export(s), **undef** if the creation failed.

removeExport(@\$exportIDs)

Removes one or more export from the database.

Param exportIDs

An array-ref containing the IDs of the exports that shall be removed.

Return Value

1 if the export(s) could be removed, **undef** if not.

changeExport(@\$exportIDs, @\$valRows)

Changes the data of one or more export.

Param vendorOSIDs

An array-ref containing the IDs of the exports that shall be changed.

Param valRows

An array-ref containing hash-refs with the new data for the export(s).

Return Value

1 if the export(s) could be changed, **undef** if not.

addSystem(@\$valRows)

Adds one or more systems to the database.

Param valRows

An array-ref containing hash-refs with the data of the new system(s).

Return Value

The IDs of the new system(s), **undef** if the creation failed.

removeSystem(@\$systemIDs)

Removes one or more systems from the database.

Param systemIDs

An array-ref containing the IDs of the systems that shall be removed.

Return Value

1 if the system(s) could be removed, **undef** if not.

`changeSystem(@$systemIDs, @$valRows)`

Changes the data of one or more systems.

Param systemIDs

An array-ref containing the IDs of the systems that shall be changed.

Param valRows

An array-ref containing hash-refs with the new data for the system(s).

Return Value

1 if the system(s) could be changed, **undef** if not.

`setClientIDsOfSystem($systemID, @$clientIDs)`

Specifies all clients that should offer the given system for booting.

Param systemID

The ID of the system whose clients you'd like to specify.

Param clientIDs

An array-ref containing the IDs of the clients that shall be connected to the system.

Return Value

1 if the system/client references could be set, **undef** if not.

`setGroupIDsOfSystem($systemID, @$groupIDs)`

Specifies all groups that should offer the given system for booting.

Param systemID

The ID of the system whose groups you'd like to specify.

Param clientIDs

An array-ref containing the IDs of the groups that shall be connected to the system.

Return Value

1 if the system/group references could be set, **undef** if not.

`addClient(@$valRows)`

Adds one or more clients to the database.

Param valRows

An array-ref containing hash-refs with the data of the new client(s).

Return Value

The IDs of the new client(s), **undef** if the creation failed.

`removeClient(@$clientIDs)`

Removes one or more clients from the database.

Param `clientIDs`

An array-ref containing the IDs of the clients that shall be removed.

Return Value

1 if the client(s) could be removed, **undef** if not.

`changeClient(@$clientIDs, @$valRows)`

Changes the data of one or more clients.

Param `clientIDs`

An array-ref containing the IDs of the clients that shall be changed.

Param `valRows`

An array-ref containing hash-refs with the new data for the client(s).

Return Value

1 if the client(s) could be changed, **undef** if not.

`setSystemIDsOfClient($clientID, @$clientIDs)`

Specifies all systems that should be offered for booting by the given client.

Param `clientID`

The ID of the client whose systems you'd like to specify.

Param `systemIDs`

An array-ref containing the IDs of the systems that shall be connected to the client.

Return Value

1 if the client/system references could be set, **undef** if not.

`setGroupIDsOfClient($clientID, @$groupIDs)`

Specifies all groups that the given client shall be part of.

Param `clientID`

The ID of the client whose groups you'd like to specify.

Param `groupIDs`

An array-ref containing the IDs of the groups that the client should be part of.

Return Value

1 if the client/group references could be set, **undef** if not.

`addGroup(@$valRows)`

Adds one or more groups to the database.

Param `valRows`

An array-ref containing hash-refs with the data of the new group(s).

Return Value

The IDs of the new group(s), `undef` if the creation failed.

`removeGroup(@$groupIDs)`

Removes one or more groups from the database.

Param `groupIDs`

An array-ref containing the IDs of the groups that shall be removed.

Return Value

1 if the group(s) could be removed, `undef` if not.

`changeGroup(@$groupIDs, @$valRows)`

Changes the data of one or more groups.

Param `groupIDs`

An array-ref containing the IDs of the groups that shall be changed.

Param `valRows`

An array-ref containing hash-refs with the new data for the group(s).

Return Value

1 if the group(s) could be changed, `undef` if not.

`setClientIDsOfGroup($groupID, @$clientIDs)`

Specifies all clients that should be part of the given group.

Param `groupID`

The ID of the group whose clients you'd like to specify.

Param `clientIDs`

An array-ref containing the IDs of the clients that shall be part of the group.

Return Value

1 if the group/client references could be set, `undef` if not.

`setSystemIDsOfGroup($groupID, @$groupIDs)`

Specifies all systems that should be offered for booting by the given group.

Param `groupID`

The ID of the group whose systems you'd like to specify.

Param systemIDs

An array-ref containing the IDs of the systems that shall be connected to the group.

Return Value

1 if the group/system references could be set, **undef** if not.

Schema Related Methods

The following methods need to be implemented in a MetaDB driver in order to be able to automatically adjust to new database schema versions (by adding and/or removing tables and table-columns).