EXT: SQL Connector Service

Extension Key: svconnector_sql

Language: en

Keywords: forDevelopers, forIntermediates

The content of this document is related to TYPO3
- a GNU/GPL CMS/Framework available from www.typo3.org

Table of Contents

Introduction

This extension implements a specific connector service for reading data from any database. It relies on ADODB to connect to any database system supported by that library (with some limits, due to the many weird syntaxes used for some systems). SQL queries can then be used.

Questions and support

If you have any questions about this extension, please ask them in the TYPO3 English mailing list, so that others can benefit from the answers. Please use the bug tracker on forge.typo3.org to report problem or suggest features (http://forge.typo3.org/projects/extension-svconnector_sql/issues).

Keeping the developer happy

If you like this extension, do not hesitate to rate it. Go the Extension Repository, search for this extension, click on its title to go to the details view, then click on the "Ratings" tab and vote (you need to be logged in). Every new vote keeps the developer ticking. So just do it!

You may also take a step back and reflect about the beauty of sharing. Think about how much you are benefiting and how much yourself is giving back to the community.

Installation

Install this extension and you can start using it's API for issuing SQL queries to any database inside your own code (although please see the limitations discussed in chapter "Configuration").

It requires extension “svconnector” which provides the base for all connector services and extension "adodb" to load the ADODB library. The "adodb" extension is a system extension provided with the TYPO3 Core.

Configuration

The various “fetch” methods of the SQL connector take the same parameters:

Parameter	Type	Description	Default
driver	string	Name of the database system to connect to, taken from the list of available ADODB drivers. See http://phplens.com/adodb/supported.databases.html	
server	string	Address of the server to connect to.	
user	string	User name to use to connect to the database.	
password	string	Password to use for the given user.	
database	string	Name of the database to connect to.	
query	string	The actual query to execute.	
init	string	SQL queries to be sent before the actual query (defined in parameter "query") is executed. This is typically used to temporarily change the encoding. **Example** `SET NAMES 'UTF8';`	

Connection methods

The current implementation of the SQL Connector only uses the classic way of setting up a connection:

```
$adodbObject = ADONewConnection($parameters['driver']);
$adodbObject->Connect($parameters['server'], $parameters['user'], $parameters['password'],
$parameters['database']);
```

that is, with the four standard parameters: server, user, password and database. This does not work for all database systems, but I currently had no use for more, nor a proper setup to test with other database system. The current version of the SQL connector was tested only with MySQL and PostgreSQL.

If you encounter problems connecting to some database system, please report an issue in the dedicated bug tracker and indicate as clearly as possible what connection syntax you would need.

Developer's manual

Getting data from another databse using the CSV connector service becomes a really easy task. The first step is to get the proper service object:

```
$services = t3lib_extMgm::findService('connector', 'sql');
if ($services === FALSE) {
      // Issue an error
} else {
      $connector = t3lib_div::makeInstanceService('connector', 'sql');
}
```

On the first line, you get a list of all services that are of type "connector" and subtype "sql". If the result if false, it means no appropriate services were found and you probably want to issue an error message.

On the contrary you are assured that there's at least one valid service and you can get an instance of it by calling `t3lib_div::makeInstanceService()`.

The next step is simply to call the appropriate method from the API – with the right parameters – depending on which format you want to have in return. For a PHP array:

```
$parameters = array(
            'driver' => 'postgres',
            'server' => '127.0.0.1',
            'user' => 'postgres',
            'password' => 'fgERvdbfR',
            'database' => 'some_db',
            'query' => 'SELECT * FROM foo ORDER BY bar'
      );
$data = $connector->fetchArray($parameters);
```

Obviously this is not limited to issuing SELECT queries, although it is what it was designed for, since connector services are really about getting data from some source. However other types of queries have not been tested.

The `fetchRaw()` method returns the same array as `fetchArray()`. The `fetchXML()` method returns the array created by `fetchArray()` transformed to XML using `t3lib_div::array2xml_cs()`.

Note that the connection is neither permanent, nor stored in the connector object (as one could imagine the object being called several times but for different connections), so it may not be ideal to use if you need to perform many queries on the same database in a given code execution run.

Known problems

See the discussion about setting up connections in the "Configuration" chapter. Please report problems in the dedicated bug tracker: http://forge.typo3.org/projects/extension-svconnector_sql/issues

To-Do list

There is a roadmap on Forge for the continuing development of this extension:

http://forge.typo3.org/projects/extension-svconnector_csv/roadmap

Changelog

Version	Changes:
1.0.0	First public release