# general_ledger

Generated by Doxygen 1.8.1.2

Sat Jun 14 2014 20:56:46

# Contents

# Chapter 1

# General Ledger.

General Ledger will be a fully-featured, multi-user, open-source general ledger system. The project is in the early stages of development.

# Chapter 2

# Todo List

**File gluser_pass.cpp**

Implement a better form of password encryption. In particular, these functions are not re-entrant, and only use the first 8 characters of the password.

# Chapter 3

# Module Index

## 3.1 Modules

Here is a list of all modules:

# Chapter 4

# Class Index

## 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 5

# Class Index

## 5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 6

# File Index

## 6.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 7

# Module Documentation

## 7.1 General Ledger database module.

### Classes

- class genleg::GLDatabase

  *General ledger database class.*
- class genleg::GLDBException

  *Base general ledger database exceptionc class.*
- class genleg::GLUser

  *General ledger user class.*

### 7.1.1 Detailed Description

Module for interacting with the general ledger database model.

## 7.2 Database interaction module

### Classes

- class gldb::DBConnException

  *Base database connection exception class.*
- class gldb::DBConnCouldNotConnect

  *Could not connect to database exception class.*
- class gldb::DBConnCouldNotQuery

  *Could not execute database query exception class.*
- class gldb::DBConn

  *Database connection class.*
- class gldb::DBConnImp

  *Abstract database implementation base class.*
- class gldb::TableException

  *Base database connection exception class.*
- class gldb::TableNoSuchField

  *No such field exception class.*
- class gldb::TableNoSuchRecord

  *No such record exception class.*
- class gldb::TableMismatchedRecordLength

  *Mismatched record length exception class.*
- class gldb::TableBadInputFile

  *Could not connect to database exception class.*
- class gldb::TableCouldNotOpenInputFile

  *Could not connect to database exception class.*
- class gldb::Table

  *Database table class.*
- class gldb::TableField

  *Database table field class.*
- class gldb::TableRow

  *Database table row class.*
- class gldb::DBConnDummy

  *Dummy database implementation class.*
- class gldb::DBConnMySQL

  *MySQL database implementation class.*

### Functions

- DBConnImp * gldb::get_connection (const std::string database, const std::string hostname, const std::string username, const std::string password)

  *Creates and returns a pointer to a database implementation.*
- std::string gldb::get_database_type ()

  *Returns the name of the compiled-in database type.*

### 7.2.1 Detailed Description

Module for interacting with the database.

### 7.2.2 Function Documentation

#### 7.2.2.1 DBConnImp * gldb::get_connection ( const std::string *database*, const std::string *hostname*, const std::string *username*, const std::string *password* )

Creates and returns a pointer to a database implementation.

The implementation of this function is provided by the individual database implementations. One database implementation is compiled into the program at any one time. Multiple database systems are, or will be, supported, and not every system will possess the libraries and headers to compile every implementation. Therefore, only only implementation is compiled in at a time. The fact that each database implementation will implement this function to return the correct derived class prevents any attempt to compile unsupported library code. This would not be feasible if we were to simply provide each implementation as a subclass.

**Parameters**

| | |
|---|---|
| *database* | The name of the database to which to connect. |
| *hostname* | The hostname of the computer running the database. |
| *username* | The username with which to log into the database. |
| *password* | The password with which to log into the database. |

**Returns**

A pointer to the database implementation.

#### 7.2.2.2 std::string gldb::get_database_type ( )

Returns the name of the compiled-in database type.

**Returns**

The name of the compiled-in database type.

## 7.3 SQL statements module

### Classes

- class genleg::DBSQLMySQL

  *MySQL SQL statements class.*
- class genleg::DBSQLStatements

  *SQL statements class.*

### 7.3.1 Detailed Description

Module for producing SQL statements used by program.

## 7.4 Program configuration module

### Classes

- class genleg::ConfigException

  *Configuration module exception base class.*
- class genleg::ConfigOptionNotSet

  *Exception class for option not set.*
- class genleg::ConfigBadOption

  *Exception class for bad provided option.*
- class genleg::ConfigCouldNotOpenFile

  *Exception class for when conf file cannot be opened.*
- class genleg::ConfigBadConfigFile

  *Exception class for badly formed configuration file.*
- class genleg::Config

  *Configuration options class.*

### Enumerations

- enum genleg::Argument

  *Enumeration class for option argument specifications.*

### 7.4.1 Detailed Description

Module for getting options from the command line and configuration files.

## 7.5 General purpose helpers.

### Functions

- std::string & pgstring::trim_front (std::string &s)

  *Trims leading whitespace from a string.*
- std::string & pgstring::trim_back (std::string &s)

  *Trims trailing whitespace from a string.*
- std::string & pgstring::trim (std::string &s)

  *Trims leading and trailing whitespace from a string.*
- std::vector< std::string > pgstring::split (const std::string &s, const char delim)

  *Splits a delimited string into tokens.*
- std::vector< std::string > & pgstring::split (std::vector< std::string > &vec, const std::string &s, const char delim)

  *Splits a delimited string into tokens.*

### 7.5.1 Detailed Description

General purpose helper classes and functions.

### 7.5.2 Function Documentation

#### 7.5.2.1 std::vector< std::string > pgstring::split ( const std::string & *s*, const char *delim* )

Splits a delimited string into tokens.

**Parameters**

| | |
|---|---|
| *s* | The string to split. |
| *delim* | The delimiter character on which to split. |

**Returns**

A vector of tokens.

#### 7.5.2.2 std::vector< std::string > & pgstring::split ( std::vector< std::string > & *vec*, const std::string & *s*, const char *delim* )

Splits a delimited string into tokens.

**Parameters**

| | |
|---|---|
| *vec* | The vector into which to add the tokens. |
| *s* | The string to split. |
| *delim* | The delimiter character on which to split. |

**Returns**

A reference to `vec`.

#### 7.5.2.3 std::string & pgstring::trim ( std::string & *s* )

Trims leading and trailing whitespace from a string.

**Parameters**

| | |
|---|---|
| *s* | The string to trim. |

**Returns**

The trimmed string.

#### 7.5.2.4 std::string & pgstring::trim_back ( std::string & *s* )

Trims trailing whitespace from a string.

**Parameters**

| | |
|---|---|
| *s* | The string to trim. |

**Returns**

The trimmed string.

#### 7.5.2.5 std::string & pgstring::trim_front ( std::string & *s* )

Trims leading whitespace from a string.

**Parameters**

| | |
|---|---|
| *s* | The string to trim. |

**Returns**

The trimmed string.

## 7.6 Reporting program.

### Functions

- static void set_configuration (genleg::Config &config, int argc, char ∗argv[])

  *Sets program configuration options.*
- static void print_usage_message ()

  *Prints a program usage message.*
- static void print_version_message ()

  *Prints a program version message.*
- static void print_help_message ()

  *Prints a program help message.*
- static std::string login (void)

  *Gets a password from the terminal.*
- int main (int argc, char ∗argv[])

  *Main function.*

### Variables

- static const char ∗ progname = "gl_report"

  *Static variable for program name.*

### 7.6.1 Detailed Description

Administrative reporting program.

### 7.6.2 Function Documentation

#### 7.6.2.1 static std::string login ( void ) [static]

Gets a password from the terminal.

**Returns**

The password.

#### 7.6.2.2 int main ( int *argc*, char ∗ *argv[]* )

Main function.

**Parameters**

| | |
|---|---|
| *argc* | Number of command line arguments. |
| *argv* | Command line arguments. |

**Returns**

Exit status code.

#### 7.6.2.3 static void set_configuration ( genleg::Config & *config*, int *argc*, char * *argv[]* ) [static]

Sets program configuration options.

**Parameters**

| | |
|---|---|
| *config* | Reference to a Config object. |
| *argc* | `argc` passed to `main()`. |
| *argv* | `argv` passed to `main()`. |

## 7.7 Database program.

### Functions

- static void set_configuration (Config &config, int argc, char *argv[])
  *Sets program configuration options.*
- static bool check_help_and_version (const Config &config)
  *Prints help or version messages if requested.*
- static bool check_db_parameters (const Config &config)
  *Checks if database, hostname and username were provided.*
- static void print_usage_message ()
  *Prints a program usage message.*
- static void print_version_message ()
  *Prints a program version message.*
- static void print_help_message ()
  *Prints a program help message.*
- static std::string login (void)
  *Gets a password from the terminal.*
- int main (int argc, char *argv[])
  *Main function.*

### Variables

- static const char * progname = "gl_db"
  *Static variable for program name.*

### 7.7.1 Detailed Description

Administrative database management program.

### 7.7.2 Function Documentation

#### 7.7.2.1 static bool check_db_parameters ( const Config & *config* ) [static]

Checks if database, hostname and username were provided.

**Parameters**

| | |
|---|---|
| *config* | Reference to a Config object. |

**Returns**

`true` if the information was provided, `false` otherwise.

#### 7.7.2.2 static bool check_help_and_version ( const Config & *config* ) [static]

Prints help or version messages if requested.

**Parameters**

| | |
|---|---|
| *config* | Reference to a Config object. |

**Returns**

`true` if the help or version message was requested, `false` otherwise.

#### 7.7.2.3 static std::string login ( void ) [static]

Gets a password from the terminal.

**Returns**

The password.

#### 7.7.2.4 int main ( int *argc*, char * *argv[]* )

Main function.

**Parameters**

| | |
|---|---|
| *argc* | Number of command line arguments. |
| *argv* | Command line arguments. |

**Returns**

Exit status code.

#### 7.7.2.5 static void set_configuration ( Config & *config*, int *argc*, char * *argv[]* ) [static]

Sets program configuration options.

**Parameters**

| | |
|---|---|
| *config* | Reference to a Config object. |
| *argc* | `argc` passed to `main()`. |
| *argv* | `argv` passed to `main()`. |

# Chapter 8

# Class Documentation

## 8.1 genleg::Config Class Reference

Configuration options class.

`#include <config.h>`

### Public Member Functions

- Config ()
- ~Config ()
- void add_cmdline_option (const std::string option, const enum Argument arg)

  *Adds a supported command line option.*
- void populate_from_cmdline (const int argc, char *const *argv)

  *Populates options from the command line.*
- void populate_from_file (const std::string filename)

  *Populates options from a configuration file.*
- bool is_set (const std::string option) const

  *Checks is an option is set.*
- const std::string & operator[] (const std::string &option) const

  *operator[] overload.*

### Private Attributes

- std::map< std::string, std::string > m_opts_set
- std::list< std::pair < std::string, enum Argument > > m_opts_supp

### 8.1.1 Detailed Description

Configuration options class.

### 8.1.2 Constructor & Destructor Documentation

#### 8.1.2.1 Config::Config ( )

Constructor

#### 8.1.2.2 Config::~Config ( )

Destructor

### 8.1.3 Member Function Documentation

#### 8.1.3.1 void Config::add_cmdline_option ( const std::string *option*, const enum Argument *arg* )

Adds a supported command line option.

**Parameters**

| | |
|---|---|
| *option* | The name of the option. |
| *arg* | The argument specification for the option. |

#### 8.1.3.2 bool Config::is_set ( const std::string *option* ) const

Checks is an option is set.

**Parameters**

| | |
|---|---|
| *option* | The name of the option to check. |

**Returns**

`true` if the option has been set, `false` if it has not.

#### 8.1.3.3 const std::string & Config::operator[] ( const std::string & *option* ) const

operator[] overload.

Retrieves the value of a set option.

**Parameters**

| | |
|---|---|
| *option* | The name of the option. |

**Returns**

The value of the option.

**Exceptions**

| | |
|---|---|
| *ConfigOptionNotSet* | If the named option has not been set. |

#### 8.1.3.4 void Config::populate_from_cmdline ( const int *argc*, char *const * *argv* )

Populates options from the command line.

**Parameters**

| | |
|---|---|
| *argc* | `argc` supplied to `main()`. |
| *argv* | `argv` supplied to `main()`. |

**Exceptions**

| | |
|---|---|
| *ConfigBadOption* | If an unsupported option is specified, or if a required argument is missing, or if an unexpected argument is found. |

#### 8.1.3.5 void Config::populate_from_file ( const std::string *filename* )

Populates options from a configuration file.

**Parameters**

| | |
|---|---|
| *filename* | The name of the configuration file. |

**Exceptions**

| | |
|---|---|
| *ConfigCouldNotOpenFile* | If the configuration file cannot be opened. |
| *ConfigBadConfigFile* | If the configuration file is badly formed. |

### 8.1.4 Member Data Documentation

#### 8.1.4.1 std::map<std::string, std::string> genleg::Config::m_opts_set [private]

Map of options which have been set

#### 8.1.4.2 std::list<std::pair<std::string, enum Argument> > genleg::Config::m_opts_supp [private]

List of options which are supported

The documentation for this class was generated from the following files:

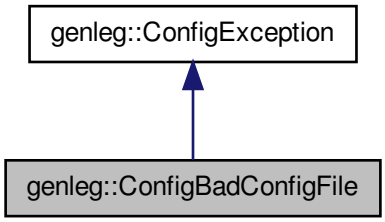
- lib/config/config.h
- lib/config/config.cpp
- lib/config/config_getopt.cpp

## 8.2 genleg::ConfigBadConfigFile Class Reference

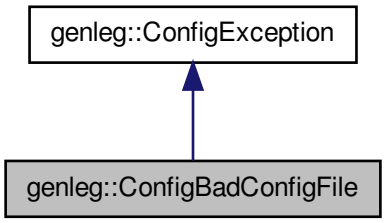
Exception class for badly formed configuration file.

`#include <config.h>`

Inheritance diagram for genleg::ConfigBadConfigFile:




Collaboration diagram for genleg::ConfigBadConfigFile:




### Public Member Functions

- ConfigBadConfigFile (const std::string &msg)
  *Constructor.*

### 8.2.1 Detailed Description

Exception class for badly formed configuration file.

### 8.2.2 Constructor & Destructor Documentation

#### 8.2.2.1 genleg::ConfigBadConfigFile::ConfigBadConfigFile ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

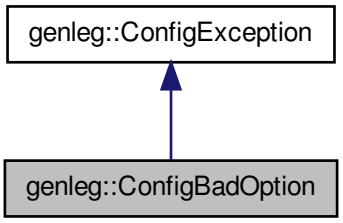
- lib/config/config.h

## 8.3 genleg::ConfigBadOption Class Reference

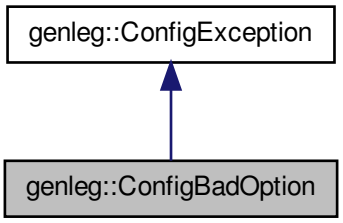
Exception class for bad provided option.

`#include <config.h>`

Inheritance diagram for genleg::ConfigBadOption:




Collaboration diagram for genleg::ConfigBadOption:




### Public Member Functions

- ConfigBadOption (const std::string &msg)

  *Constructor.*

### 8.3.1 Detailed Description

Exception class for bad provided option.

### 8.3.2 Constructor & Destructor Documentation

#### 8.3.2.1 genleg::ConfigBadOption::ConfigBadOption ( const std::string & *msg* ) [inline],[explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

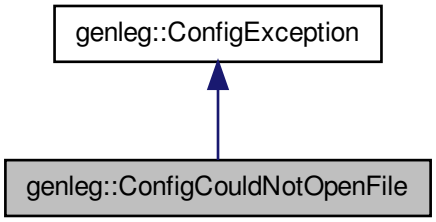
- lib/config/config.h

## 8.4 genleg::ConfigCouldNotOpenFile Class Reference

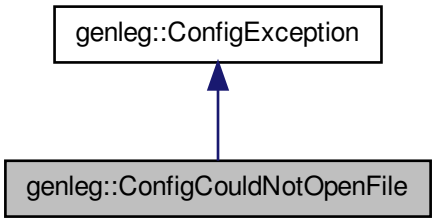
Exception class for when conf file cannot be opened.

`#include <config.h>`

Inheritance diagram for genleg::ConfigCouldNotOpenFile:




Collaboration diagram for genleg::ConfigCouldNotOpenFile:




### Public Member Functions

- ConfigCouldNotOpenFile (const std::string &msg)

  *Constructor.*

### 8.4.1 Detailed Description

Exception class for when conf file cannot be opened.

### 8.4.2 Constructor & Destructor Documentation

#### 8.4.2.1 genleg::ConfigCouldNotOpenFile::ConfigCouldNotOpenFile ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

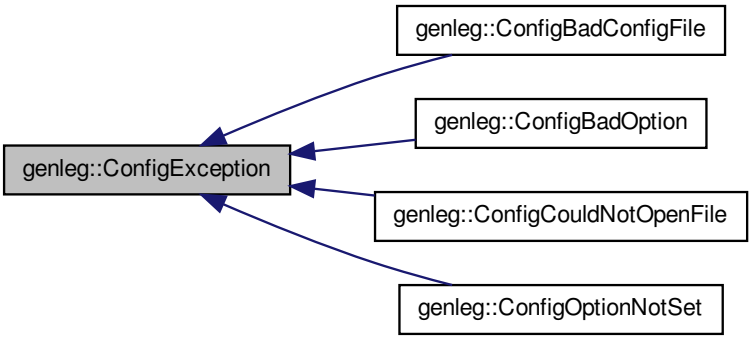
- lib/config/config.h

## 8.5 genleg::ConfigException Class Reference

Configuration module exception base class.

`#include <config.h>`

Inheritance diagram for genleg::ConfigException:




### Public Member Functions

- ConfigException (const std::string &msg)

  *Constructor.*

### 8.5.1 Detailed Description

Configuration module exception base class.

### 8.5.2 Constructor & Destructor Documentation

#### 8.5.2.1 genleg::ConfigException::ConfigException ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

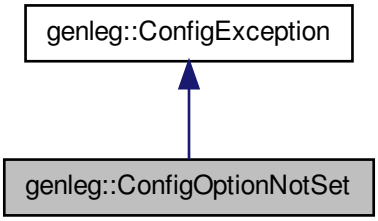
- lib/config/config.h

## 8.6 genleg::ConfigOptionNotSet Class Reference

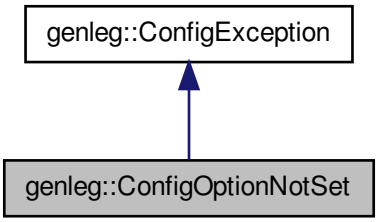
Exception class for option not set.

`#include <config.h>`

Inheritance diagram for genleg::ConfigOptionNotSet:




Collaboration diagram for genleg::ConfigOptionNotSet:




### Public Member Functions

- ConfigOptionNotSet (const std::string &msg)

*Constructor.*

### 8.6.1 Detailed Description

Exception class for option not set.

### 8.6.2 Constructor & Destructor Documentation

#### 8.6.2.1 genleg::ConfigOptionNotSet::ConfigOptionNotSet ( const std::string & *msg* ) [inline],[explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

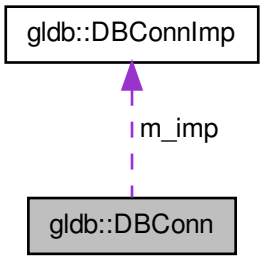
- lib/config/config.h

## 8.7 gldb::DBConn Class Reference

Database connection class.

`#include <dbconn.h>`

Collaboration diagram for gldb::DBConn:




### Public Member Functions

- DBConn (DBConnImp *imp)

  *Constructor.*
- ~DBConn ()

  *Destructor..*
- void query (std::string sql_query)

  *Runs an SQL query.*
- Table select (std::string query)

*Runs an SQL SELECT query.*

- DBConn (const DBConn &)
- DBConn & operator= (const DBConn &)

### Private Attributes

- DBConnImp * m_imp

## 8.7.1 Detailed Description

Database connection class.

## 8.7.2 Constructor & Destructor Documentation

### 8.7.2.1 DBConn::DBConn ( DBConnImp * *imp* ) [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *imp* | Pointer to database implementation object. |

### 8.7.2.2 gldb::DBConn::DBConn ( const DBConn & )

Deleted copy constructor

## 8.7.3 Member Function Documentation

### 8.7.3.1 DBConn& gldb::DBConn::operator= ( const DBConn & )

Deleted assignment operator

### 8.7.3.2 void DBConn::query ( std::string *sql_query* )

Runs an SQL query.

**Parameters**

| | |
|---|---|
| *sql_query* | The query. |

**Returns**

A Table object containing the results.

### 8.7.3.3 Table DBConn::select ( std::string *query* )

Runs an SQL SELECT query.

**Parameters**

| | |
|---|---|
| *query* | The query. |

**Returns**

A Table object containing the results.

### 8.7.4 Member Data Documentation

#### 8.7.4.1 DBConnImp* gldb::DBConn::m_imp [private]

Pointer to database implementation object.

The documentation for this class was generated from the following files:

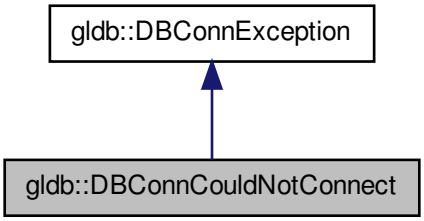
- lib/database/dbconn.h
- lib/database/dbconn.cpp

## 8.8 gldb::DBConnCouldNotConnect Class Reference

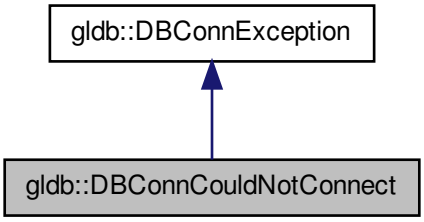
Could not connect to database exception class.

`#include <dbconn.h>`

Inheritance diagram for gldb::DBConnCouldNotConnect:




Collaboration diagram for gldb::DBConnCouldNotConnect:



### Public Member Functions

- DBConnCouldNotConnect (const std::string &msg)

  *Constructor.*

### 8.8.1 Detailed Description

Could not connect to database exception class.

### 8.8.2 Constructor & Destructor Documentation

#### 8.8.2.1 gldb::DBConnCouldNotConnect::DBConnCouldNotConnect ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

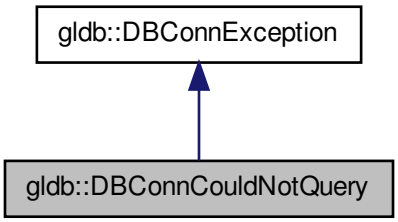
- lib/database/dbconn.h

## 8.9 gldb::DBConnCouldNotQuery Class Reference

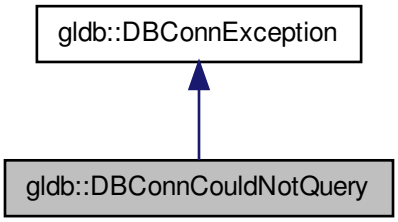
Could not execute database query exception class.

`#include <dbconn.h>`

Inheritance diagram for gldb::DBConnCouldNotQuery:



Collaboration diagram for gldb::DBConnCouldNotQuery:




### Public Member Functions

- DBConnCouldNotQuery (const std::string &msg)
  *Constructor.*

### 8.9.1 Detailed Description

Could not execute database query exception class.

### 8.9.2 Constructor & Destructor Documentation

#### 8.9.2.1 gldb::DBConnCouldNotQuery::DBConnCouldNotQuery ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

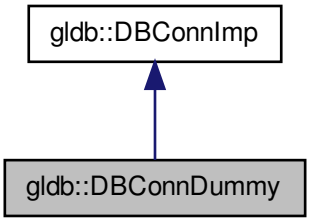
- lib/database/dbconn.h

## 8.10 gldb::DBConnDummy Class Reference

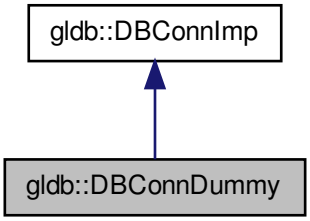
Dummy database implementation class.

`#include <dbconn_dummy_imp.h>`

Inheritance diagram for gldb::DBConnDummy:




Collaboration diagram for gldb::DBConnDummy:




### Public Member Functions

- DBConnDummy (const std::string database, const std::string hostname, const std::string username, const std::string password)

  *Constructor.*
- DBConnDummy (const DBConnDummy &)
- virtual ~DBConnDummy ()
- DBConnDummy & operator= (const DBConnDummy &)
- Table select (std::string query)

  *Fakes running of an SQL SELECT query.*

## 8.10.1 Detailed Description

Dummy database implementation class.

## 8.10.2 Constructor & Destructor Documentation

### 8.10.2.1 DBConnDummy::DBConnDummy ( const std::string *database*, const std::string *hostname*, const std::string *username*, const std::string *password* )

Constructor.

**Parameters**

| | |
|---|---|
| *database* | The name of the Dummy database. |
| *hostname* | The hostname of the server. |
| *username* | The username to log into the database. |
| *password* | The password to log into the database. |

#### 8.10.2.2 gldb::DBConnDummy::DBConnDummy ( const DBConnDummy & )

Deleted copy constructor

#### 8.10.2.3 DBConnDummy::~DBConnDummy ( ) [virtual]

Destructor

### 8.10.3 Member Function Documentation

#### 8.10.3.1 DBConnDummy& gldb::DBConnDummy::operator= ( const DBConnDummy & )

Deleted assignment operator

#### 8.10.3.2 Table DBConnDummy::select ( std::string *query* ) [virtual]

Fakes running of an SQL SELECT query.

**Parameters**

| | |
|---|---|
| *query* | Any query. |

**Returns**

A Table object containing dummy results.

Implements gldb::DBConnImp.

The documentation for this class was generated from the following files:

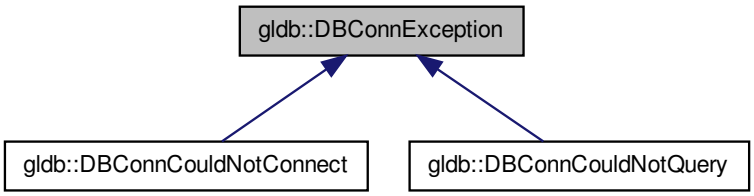
- lib/database_imp/dummy/dbconn_dummy_imp.h
- lib/database_imp/dummy/dbconn_dummy_imp.cpp

## 8.11 gldb::DBConnException Class Reference

Base database connection exception class.

`#include <dbconn.h>`

Inheritance diagram for gldb::DBConnException:




### Public Member Functions

- DBConnException (const std::string &msg)

  *Constructor.*

### 8.11.1 Detailed Description

Base database connection exception class.

### 8.11.2 Constructor & Destructor Documentation

#### 8.11.2.1 gldb::DBConnException::DBConnException ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

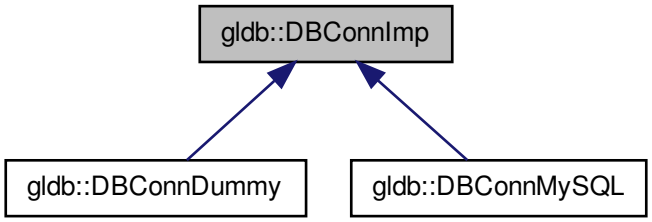
- lib/database/dbconn.h

## 8.12 gldb::DBConnImp Class Reference

Abstract database implementation base class.

`#include <dbconnimp.h>`

Inheritance diagram for gldb::DBConnImp:




### Public Member Functions

- DBConnImp ()
- virtual ~DBConnImp ()
- virtual void query (std::string sql_query)=0

  *Runs an SQL query.*
- virtual Table select (std::string query)=0

  *Runs an SQL SELECT query.*

## 8.12.1 Detailed Description

Abstract database implementation base class.

## 8.12.2 Constructor & Destructor Documentation

### 8.12.2.1 gldb::DBConnImp::DBConnImp ( ) [inline]

Constructor

### 8.12.2.2 virtual gldb::DBConnImp::~DBConnImp ( ) [inline], [virtual]

Destructor

## 8.12.3 Member Function Documentation

### 8.12.3.1 virtual void gldb::DBConnImp::query ( std::string *sql_query* ) [pure virtual]

Runs an SQL query.

**Parameters**

| | |
|---|---|
| *sql_query* | The query. |

Implemented in gldb::DBConnMySQL.

#### 8.12.3.2 virtual Table gldb::DBConnImp::select ( std::string *query* ) [pure virtual]

Runs an SQL SELECT query.

**Parameters**

| | |
|---|---|
| *query* | The query. |

**Returns**

A Table object containing the results.

Implemented in gldb::DBConnMySQL, and gldb::DBConnDummy.

The documentation for this class was generated from the following file:

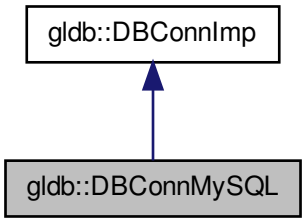
- lib/database/dbconnimp.h

## 8.13 gldb::DBConnMySQL Class Reference

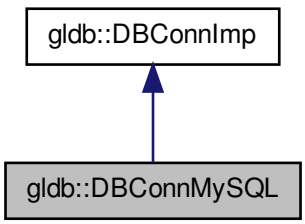
MySQL database implementation class.

`#include <dbconn_mysql_imp.h>`

Inheritance diagram for gldb::DBConnMySQL:




Collaboration diagram for gldb::DBConnMySQL:



### Public Member Functions

- DBConnMySQL (const std::string database, const std::string hostname, const std::string username, const std::string password)

  *Constructor.*
- DBConnMySQL (const DBConnMySQL &)
- virtual ~DBConnMySQL ()
- DBConnMySQL & operator= (const DBConnMySQL &)
- virtual void query (std::string sql_query)

  *Runs an SQL query.*
- virtual Table select (std::string query)

  *Runs an SQL SELECT query.*

### Private Attributes

- MYSQL * m_conn

## 8.13.1 Detailed Description

MySQL database implementation class.

## 8.13.2 Constructor & Destructor Documentation

### 8.13.2.1 DBConnMySQL::DBConnMySQL ( const std::string *database*, const std::string *hostname*, const std::string *username*, const std::string *password* )

Constructor.

**Parameters**

| | |
|---|---|
| *database* | The name of the MySQL database. |
| *hostname* | The hostname of the server. |
| *username* | The username to log into the database. |
| *password* | The password to log into the database. |

**Exceptions**

| | |
|---|---|
| *DBConnCouldNotConnect* | If could not connect to database. |

### 8.13.2.2 gldb::DBConnMySQL::DBConnMySQL ( const DBConnMySQL & )

Deleted copy constructor

### 8.13.2.3 DBConnMySQL::~DBConnMySQL ( ) [virtual]

Destructor

## 8.13.3 Member Function Documentation

#### 8.13.3.1 DBConnMySQL& gldb::DBConnMySQL::operator= ( const DBConnMySQL & )

Deleted assignment operator

#### 8.13.3.2 void DBConnMySQL::query ( std::string *sql_query* ) [virtual]

Runs an SQL query.

**Parameters**

| | |
|---|---|
| *sql_query* | The query. |

**Exceptions**

| | |
|---|---|
| *DBConnCouldNotQuery* | If could not successfully execute query. |

Implements gldb::DBConnImp.

#### 8.13.3.3 Table DBConnMySQL::select ( std::string *query* ) [virtual]

Runs an SQL SELECT query.

**Parameters**

| | |
|---|---|
| *query* | The query. |

**Returns**

A Table object containing the results.

**Exceptions**

| | |
|---|---|
| *DBConnCouldNotQuery* | If could not successfully execute query. |

Implements gldb::DBConnImp.

### 8.13.4 Member Data Documentation

#### 8.13.4.1 MYSQL* gldb::DBConnMySQL::m_conn [private]

The initialized MySQL handle.

The documentation for this class was generated from the following files:

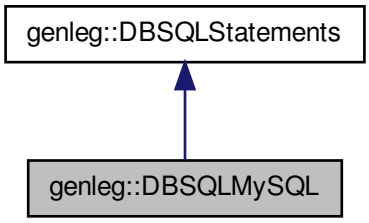
- lib/database_imp/mysql/dbconn_mysql_imp.h
- lib/database_imp/mysql/dbconn_mysql_imp.cpp

## 8.14 genleg::DBSQLMySQL Class Reference

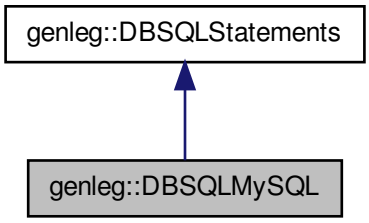
MySQL SQL statements class.

`#include <dbsql_mysql.h>`

Inheritance diagram for genleg::DBSQLMySQL:




Collaboration diagram for genleg::DBSQLMySQL:




### Additional Inherited Members

#### 8.14.1 Detailed Description

MySQL SQL statements class.

The documentation for this class was generated from the following file:

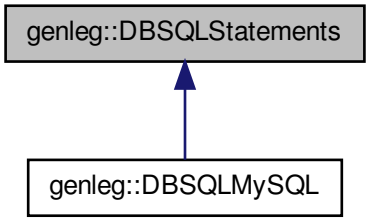
- lib/dbsql/dbsql_mysql.h

## 8.15 genleg::DBSQLStatements Class Reference

SQL statements class.

`#include <dbsqlstatements.h>`

Inheritance diagram for genleg::DBSQLStatements:




### Public Member Functions

- DBSQLStatements ()
- virtual ~DBSQLStatements ()
- virtual std::string create_table (const std::string table_name) const

  *Returns a SQL statement for creating a table.*
- virtual std::string drop_table (const std::string table_name) const

  *Returns a SQL statement for dropping a table.*
- virtual std::string create_view (const std::string view_name) const

  *Returns a SQL statement for creating a view.*
- virtual std::string drop_view (const std::string view_name) const

  *Returns a SQL statement for dropping a view.*
- virtual std::string user_by_id (const std::string user_id) const

  *Returns a SQL statement to select a user by ID.*
- virtual std::string user_by_username (const std::string user_name) const

  *Returns a SQL statement to select a user by username.*
- virtual std::string update_user (const GLUser &user) const

  *Returns a SQL UPDATE statement to update a user.*

## 8.15.1 Detailed Description

SQL statements class.

## 8.15.2 Constructor & Destructor Documentation

### 8.15.2.1 DBSQLStatements::DBSQLStatements ( )

Constructor

### 8.15.2.2 DBSQLStatements::~DBSQLStatements ( ) [virtual]

Destructor

### 8.15.3 Member Function Documentation

#### 8.15.3.1 std::string DBSQLStatements::create_table ( const std::string *table_name* ) const [virtual]

Returns a SQL statement for creating a table.

**Parameters**

| | |
|---|---|
| *table_name* | The table to create. |

**Returns**

The SQL statement to create the table.

#### 8.15.3.2 std::string DBSQLStatements::create_view ( const std::string *view_name* ) const [virtual]

Returns a SQL statement for creating a view.

**Parameters**

| | |
|---|---|
| *view_name* | The view to create. |

**Returns**

The SQL statement to create the view.

#### 8.15.3.3 std::string DBSQLStatements::drop_table ( const std::string *table_name* ) const [virtual]

Returns a SQL statement for dropping a table.

**Parameters**

| | |
|---|---|
| *table_name* | The table to drop. |

**Returns**

The SQL statement to drop the table.

#### 8.15.3.4 std::string DBSQLStatements::drop_view ( const std::string *view_name* ) const [virtual]

Returns a SQL statement for dropping a view.

**Parameters**

| | |
|---|---|
| *view_name* | The view to drop. |

**Returns**

The SQL statement to drop the view.

#### 8.15.3.5 std::string DBSQLStatements::update_user ( const GLUser & *user* ) const [virtual]

Returns a SQL UPDATE statement to update a user.

**Parameters**

| | |
|---|---|
| *user* | A user object. |

**Returns**

The SQL statement.

#### 8.15.3.6 std::string DBSQLStatements::user_by_id ( const std::string *user_id* ) const [virtual]

Returns a SQL statement to select a user by ID.

**Parameters**

| | |
|---|---|
| *user_id* | The user_id |

**Returns**

The SQL statement.

#### 8.15.3.7 std::string DBSQLStatements::user_by_username ( const std::string *user_name* ) const [virtual]

Returns a SQL statement to select a user by username.

**Parameters**

| | |
|---|---|
| *user_name* | The username. |

**Returns**

The SQL statement.

The documentation for this class was generated from the following files:

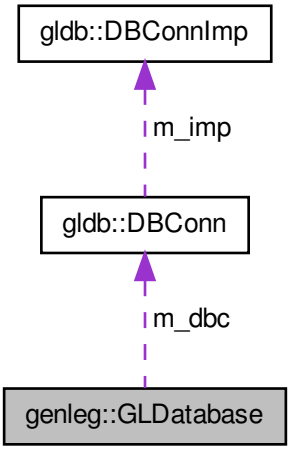
- lib/dbsql/dbsqlstatements.h
- lib/dbsql/dbsqlstatements.cpp

## 8.16 genleg::GLDatabase Class Reference

General ledger database class.

`#include <gldatabase.h>`

Collaboration diagram for genleg::GLDatabase:




## Public Member Functions

- GLDatabase (const std::string database, const std::string hostname, const std::string username, const std::string password)

  *Constructor.*
- ~GLDatabase ()
- void create_structure ()

  *Creates the database structure.*
- void destroy_structure ()

  *Destroys the database structure.*
- void load_sample_data (const std::string &dir)

  *Loads sample data into the database.*
- GLUser get_user_by_id (const std::string &user_id)

  *Returns a user from an ID.*
- GLUser get_user_by_username (const std::string &user_name)

  *Returns a user from a user name.*
- void update_user (const GLUser &user)

  *Updates a user's details.*

## Static Public Member Functions

- static std::string backend ()

  *Returns the backend database implementation.*

## Private Attributes

- gldb::DBConn m_dbc
- std::shared_ptr< DBSQLStatements > m_sql
- const std::vector< std::string > m_tables
- const std::vector< std::string > m_views

### 8.16.1 Detailed Description

General ledger database class.

### 8.16.2 Constructor & Destructor Documentation

#### 8.16.2.1 GLDatabase::GLDatabase ( const std::string *database*, const std::string *hostname*, const std::string *username*, const std::string *password* )

Constructor.

**Parameters**

| | |
|---|---|
| *database* | Database name. |
| *hostname* | Hostname of database machine. |
| *username* | Username to log into database. |
| *password* | Password to log into database. |

**Exceptions**

| | |
|---|---|
| *GLDBException* | on error. |

#### 8.16.2.2 GLDatabase::~GLDatabase ( )

Destructor

### 8.16.3 Member Function Documentation

#### 8.16.3.1 std::string GLDatabase::backend ( ) [static]

Returns the backend database implementation.

This may be called to discover which database platform support has been compiled into the application.

**Returns**

A string containing the database platform name.

#### 8.16.3.2 void GLDatabase::create_structure ( )

Creates the database structure.

**Exceptions**

| | |
|---|---|
| *GLDBException* | on error. |

#### 8.16.3.3 void GLDatabase::destroy_structure ( )

Destroys the database structure.

**Exceptions**

| | |
|---|---|
| *GLDBException* | on error. |

#### 8.16.3.4 GLUser GLDatabase::get_user_by_id ( const std::string & *user_id* )

Returns a user from an ID.

**Parameters**

| | |
|---|---|
| *user_id* | The user ID. |

**Returns**

The user.

**Exceptions**

| | |
|---|---|
| *GLDBException* | if the user cannot be found. |

#### 8.16.3.5 GLUser GLDatabase::get_user_by_username ( const std::string & *user_name* )

Returns a user from a user name.

**Parameters**

| | |
|---|---|
| *user_name* | The user name. |

**Returns**

The user.

**Exceptions**

| | |
|---|---|
| *GLDBException* | if the user cannot be found. |

#### 8.16.3.6 void GLDatabase::load_sample_data ( const std::string & *dir* )

Loads sample data into the database.

**Parameters**

| | |
|---|---|
| *dir* | The directory containing the sample data. Individual files in that directory should be named after the table they are intended to poplate. |

**Exceptions**

| | |
|---|---|
| *GLDBException* | on error. |

#### 8.16.3.7 void GLDatabase::update_user ( const GLUser & *user* )

Updates a user's details.

**Parameters**

| | |
|---|---|
| *user* | The user object. |

### 8.16.4 Member Data Documentation

#### 8.16.4.1 gldb::DBConn genleg::GLDatabase::m_dbc [private]

Database connection

#### 8.16.4.2 std::shared_ptr<DBSQLStatements> genleg::GLDatabase::m_sql [private]

SQL statements object

#### 8.16.4.3 const std::vector<std::string> genleg::GLDatabase::m_tables [private]

Vector containing database table names

#### 8.16.4.4 const std::vector<std::string> genleg::GLDatabase::m_views [private]

Vector containing database view names

The documentation for this class was generated from the following files:

- lib/gldb/gldatabase.h
- lib/gldb/gldatabase.cpp

## 8.17 genleg::GLDBException Class Reference

Base general ledger database exceptionc class.

`#include <glexception.h>`

### Public Member Functions

- GLDBException (const std::string &msg)

  *Constructor.*

### 8.17.1 Detailed Description

Base general ledger database exceptionc class.

### 8.17.2 Constructor & Destructor Documentation

#### 8.17.2.1 genleg::GLDBException::GLDBException ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

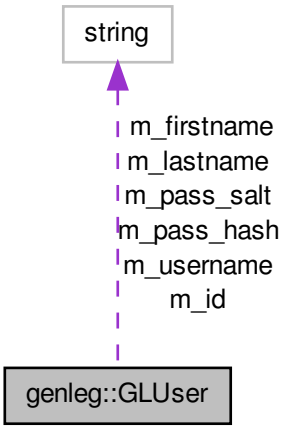
- lib/gldb/glexception.h

## 8.18 genleg::GLUser Class Reference

General ledger user class.

`#include <gluser.h>`

Collaboration diagram for genleg::GLUser:




### Public Member Functions

- GLUser (const std::string id, const std::string username, const std::string firstname, const std::string lastname, const std::string pass_hash, const std::string pass_salt, const bool enabled)

  *Constructor.*
- ~GLUser ()
- std::string id () const

  *Returns the user ID.*
- std::string username () const

  *Returns the username.*
- std::string firstname () const

  *Returns the user's first name.*
- std::string lastname () const

  *Returns the user's last name.*
- std::string pass_hash () const

  *Returns the user's hashed password.*
- std::string pass_salt () const

  *Returns the user's password salt.*
- bool enabled () const

  *Returns the user's enabled status.*
- void set_username (const std::string &new_username)

  *Sets a user's username.*
- void set_firstname (const std::string &new_firstname)

  *Sets a user's first name.*
- void set_lastname (const std::string &new_lastname)

*Sets a user's last name.*

- void set_enabled (const bool new_enabled)

  *Sets a user's enabled status.*

- void set_password (const std::string &new_pass)

  *Sets a user's password hash and salt.*

- bool check_password (const std::string &check_pass)

  *Checks a password against the user's hash.*

### Private Attributes

- std::string m_id
- std::string m_username
- std::string m_firstname
- std::string m_lastname
- bool m_enabled
- std::string m_pass_hash
- std::string m_pass_salt

## 8.18.1 Detailed Description

General ledger user class.

## 8.18.2 Constructor & Destructor Documentation

### 8.18.2.1 GLUser::GLUser ( const std::string *id*, const std::string *username*, const std::string *firstname*, const std::string *lastname*, const std::string *pass_hash*, const std::string *pass_salt*, const bool *enabled* )

Constructor.

**Parameters**

| | |
|---|---|
| *id* | User ID |
| *username* | Username |
| *firstname* | First name |
| *lastname* | Last name |
| *pass_hash* | The hashed password |
| *pass_salt* | The salt for the hashed password |
| *enabled* | `true` if user is enabled, `false` otherwise. |

### 8.18.2.2 GLUser::~GLUser ( )

Destructor

## 8.18.3 Member Function Documentation

### 8.18.3.1 bool GLUser::check_password ( const std::string & *check_pass* )

Checks a password against the user's hash.

**Parameters**

| | |
|---|---|
| *check_pass* | The password to check, must be $>$ 8 characters. |

**Returns**

`true` is the password matches, `false` otherwise.

#### 8.18.3.2 bool GLUser::enabled ( ) const

Returns the user's enabled status.

**Returns**

The user's enabled status.

#### 8.18.3.3 std::string GLUser::firstname ( ) const

Returns the user's first name.

**Returns**

The user's first name.

#### 8.18.3.4 std::string GLUser::id ( ) const

Returns the user ID.

**Returns**

The user ID.

#### 8.18.3.5 std::string GLUser::lastname ( ) const

Returns the user's last name.

**Returns**

The user's last name.

#### 8.18.3.6 std::string GLUser::pass_hash ( ) const

Returns the user's hashed password.

**Returns**

The user's hashed password.

#### 8.18.3.7 std::string GLUser::pass_salt ( ) const

Returns the user's password salt.

**Returns**

The user's password salt.

#### 8.18.3.8 void GLUser::set_enabled ( const bool *new_enabled* )

Sets a user's enabled status.

**Parameters**

| | |
|---|---|
| *new_enabled* | The user's new enabled status. |

#### 8.18.3.9 void GLUser::set_firstname ( const std::string & *new_firstname* )

Sets a user's first name.

**Parameters**

| | |
|---|---|
| *new_firstname* | The user's new first name. |

#### 8.18.3.10 void GLUser::set_lastname ( const std::string & *new_lastname* )

Sets a user's last name.

**Parameters**

| | |
|---|---|
| *new_lastname* | The user's new last name. |

#### 8.18.3.11 void GLUser::set_password ( const std::string & *new_pass* )

Sets a user's password hash and salt.

**Parameters**

| | |
|---|---|
| *new_pass* | The new password, must be $>$ 8 characters. |

#### 8.18.3.12 void GLUser::set_username ( const std::string & *new_username* )

Sets a user's username.

**Parameters**

| | |
|---|---|
| *new_username* | The user's new username. |

#### 8.18.3.13 std::string GLUser::username ( ) const

Returns the username.

**Returns**

The username.

### 8.18.4 Member Data Documentation

#### 8.18.4.1 bool genleg::GLUser::m_enabled [private]

User's enabled status

#### 8.18.4.2 std::string genleg::GLUser::m_firstname [private]

User's first name

#### 8.18.4.3 std::string genleg::GLUser::m_id [private]

User ID

#### 8.18.4.4 std::string genleg::GLUser::m_lastname [private]

User's last name

#### 8.18.4.5 std::string genleg::GLUser::m_pass_hash [private]

User's hashed password

#### 8.18.4.6 std::string genleg::GLUser::m_pass_salt [private]

User's password salt

#### 8.18.4.7 std::string genleg::GLUser::m_username [private]

Username

The documentation for this class was generated from the following files:

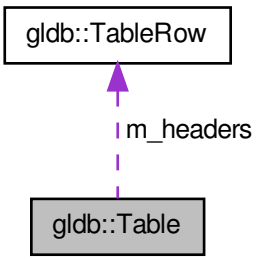
- lib/gldb/gluser.h
- lib/gldb/gluser.cpp
- lib/gldb/gluser_pass.cpp

## 8.19 gldb::Table Class Reference

Database table class.

`#include <table.h>`

Collaboration diagram for gldb::Table:



### Public Member Functions

- Table (const TableRow &headers)

  *Constructor.*
- ~Table ()
- size_t num_fields () const

  *Returns the number of fields in each row.*
- size_t num_records () const

  *Returns the number of record in the table.*
- void set_quoted (std::vector< bool > &vec)

  *Sets the quote flags for the records.*
- const TableRow & get_headers () const

  *Returns the field names.*
- const TableRow & operator[] (const size_t idx) const

  *Overloaded index operator.*
- void append_record (const TableRow &new_record)

  *Appends a record to the table.*
- std::string insert_query (const std::string table_name, const size_t idx)

  *Creates an SQL INSERT query from a table record.*
- std::string get_field (const std::string field_name, const size_t row_index)

  *Gets a field from a record by field name.*

### Static Public Member Functions

- static Table create_from_file (const std::string filename, const char delim)

  *Creates a table from an input file.*

### Private Attributes

- TableRow m_headers
- std::vector< TableRow > m_records
- std::vector< bool > m_quoted

## 8.19.1 Detailed Description

Database table class.

## 8.19.2 Constructor & Destructor Documentation

### 8.19.2.1 Table::Table ( const TableRow & *headers* ) [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *headers* | Table row containing field names. |

### 8.19.2.2 Table::~Table ( )

Destructor

### 8.19.3 Member Function Documentation

#### 8.19.3.1 void Table::append_record ( const TableRow & *new_record* )

Appends a record to the table.

**Parameters**

| | |
|---|---|
| *new_record* | The record to append. |

#### 8.19.3.2 Table Table::create_from_file ( const std::string *filename*, const char *delim* ) [static]

Creates a table from an input file.

**Parameters**

| | |
|---|---|
| *filename* | The name of the input file. |
| *delim* | The delimiting character. |

**Returns**

The table.

**Exceptions**

| | |
|---|---|
| *TableBadInputFile* | on badly formed input file. |
| *TableCouldNotOpenInputFile* | on bad filename. |

#### 8.19.3.3 std::string Table::get_field ( const std::string *field_name*, const size_t *row_index* )

Gets a field from a record by field name.

**Parameters**

| | |
|---|---|
| *field_name* | The name of the field. |
| *row_index* | The index of the row. |

**Returns**

The contents ofthe field.

**Exceptions**

| | |
|---|---|
| *TableNoSuchField* | if `field_name` is not a valid field name. |
| *TableNoSuchRecord* | if there is no record at index `row_index`. |

#### 8.19.3.4 const TableRow & Table::get_headers ( ) const

Returns the field names.

**Returns**

The field names.

#### 8.19.3.5 std::string Table::insert_query ( const std::string *table_name,* const size_t *idx* )

Creates an SQL INSERT query from a table record.

**Parameters**

| | |
|---|---|
| *table_name* | The name of the table into which to INSERT. |
| *idx* | The index of the record. |

**Returns**

A string containing the query.

#### 8.19.3.6 size_t Table::num_fields ( ) const

Returns the number of fields in each row.

**Returns**

The number of fields in each row.

#### 8.19.3.7 size_t Table::num_records ( ) const

Returns the number of record in the table.

**Returns**

The number of records in the table.

#### 8.19.3.8 const TableRow & Table::operator[ ] ( const size_t *idx* ) const

Overloaded index operator.

**Parameters**

| | |
|---|---|
| *idx* | The zero-based index of the record. |

**Returns**

The selected record.

#### 8.19.3.9 void Table::set_quoted ( std::vector< bool > & *vec* )

Sets the quote flags for the records.

**Parameters**

| | |
|---|---|
| *vec* | A vector of bools. The size must match the size of the records. |

### 8.19.4 Member Data Documentation

#### 8.19.4.1 TableRow gldb::Table::m_headers [private]

The names of the fields

#### 8.19.4.2 std::vector<bool> gldb::Table::m_quoted [private]

A vector to show if fields should be quoted for INSERT

#### 8.19.4.3 std::vector<TableRow> gldb::Table::m_records [private]

A vector of the records

The documentation for this class was generated from the following files:

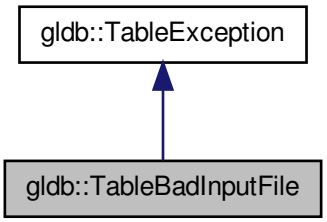
- lib/database/table.h
- lib/database/table.cpp

## 8.20 gldb::TableBadInputFile Class Reference

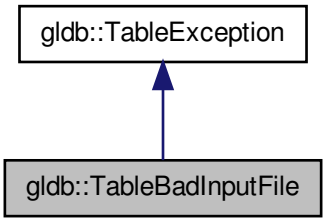
Could not connect to database exception class.

`#include <table.h>`

Inheritance diagram for gldb::TableBadInputFile:



Collaboration diagram for gldb::TableBadInputFile:




### Public Member Functions

- TableBadInputFile (const std::string &msg)

  *Constructor.*

### 8.20.1 Detailed Description

Could not connect to database exception class.

### 8.20.2 Constructor & Destructor Documentation

#### 8.20.2.1 gldb::TableBadInputFile::TableBadInputFile ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

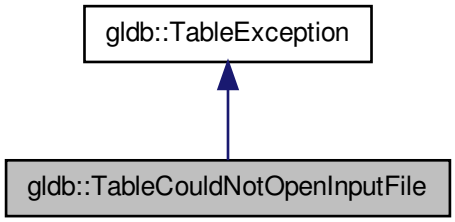
- lib/database/table.h

## 8.21 gldb::TableCouldNotOpenInputFile Class Reference

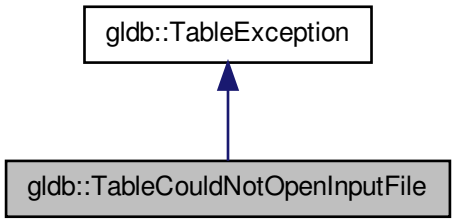
Could not connect to database exception class.

`#include <table.h>`

Inheritance diagram for gldb::TableCouldNotOpenInputFile:




Collaboration diagram for gldb::TableCouldNotOpenInputFile:




## Public Member Functions

- TableCouldNotOpenInputFile (const std::string &msg)

  *Constructor.*

### 8.21.1 Detailed Description

Could not connect to database exception class.

### 8.21.2 Constructor & Destructor Documentation

#### 8.21.2.1 gldb::TableCouldNotOpenInputFile::TableCouldNotOpenInputFile ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

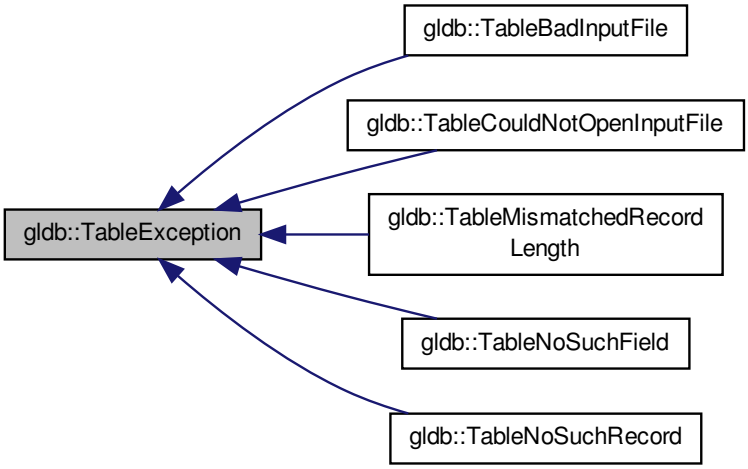
- lib/database/table.h

## 8.22 gldb::TableException Class Reference

Base database connection exception class.

`#include <table.h>`

Inheritance diagram for gldb::TableException:




### Public Member Functions

- TableException (const std::string &msg)

  *Constructor.*

### 8.22.1 Detailed Description

Base database connection exception class.

### 8.22.2 Constructor & Destructor Documentation

#### 8.22.2.1 gldb::TableException::TableException ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

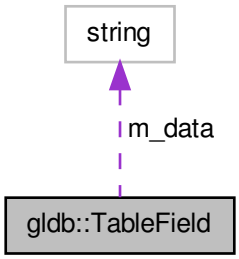
- lib/database/table.h

## 8.23 gldb::TableField Class Reference

Database table field class.

`#include <tablefield.h>`

Collaboration diagram for gldb::TableField:




### Public Member Functions

- TableField (const char *data)
  *Constructor accepting `const char *` data.*
- TableField (const std::string &data)
  *Constructor accepting `std:string` data.*
- ~TableField ()
- size_t length () const
  *Returns the length of the field.*
- operator std::string () const
  *Overridden conversion operator.*
- TableField & operator= (const char *data)
  *Overridden assignment operator for `const char *`.*
- TableField & operator= (const std::string &data)
  *Overridden assignment operator for `std::string`.*
- char & operator[] (const size_t idx)
  *Overridden index operator.*
- const char & operator[] (const size_t idx) const
  *Overridden index operator.*
- TableField & operator+= (const char &c)
  *Overridden compound assignment operator.*
- TableField & operator+= (const std::string &data)
  *Overridden compound assignment operator.*

### Private Attributes

- std::string m_data

### Friends

- std::ostream & operator<< (std::ostream &out, const TableField &field)

  *Overridden << operator for printing a field.*

## 8.23.1 Detailed Description

Database table field class.

## 8.23.2 Constructor & Destructor Documentation

### 8.23.2.1 TableField::TableField ( const char * *data* ) [explicit]

Constructor accepting `const char *` data.

**Parameters**

| | |
|---|---|
| *data* | The initial contents of the field. |

### 8.23.2.2 TableField::TableField ( const std::string & *data* ) [explicit]

Constructor accepting `std:string` data.

**Parameters**

| | |
|---|---|
| *data* | The initial contents of the field. |

### 8.23.2.3 TableField::~TableField ( )

Destructor

## 8.23.3 Member Function Documentation

### 8.23.3.1 size_t TableField::length ( ) const

Returns the length of the field.

**Returns**

The length of the field.

### 8.23.3.2 TableField::operator std::string ( ) const

Overridden conversion operator.

Returns the field contents as a string.

### 8.23.3.3 TableField & TableField::operator+= ( const char & *c* )

Overridden compound assignment operator.

**Parameters**

| | |
|---|---|
| *c* | The character to append to the field. |

**Returns**

A reference to the same field.

#### 8.23.3.4 TableField & TableField::operator+= ( const std::string & *data* )

Overridden compound assignment operator.

**Parameters**

| | |
|---|---|
| *data* | The string to append to the field. |

**Returns**

A reference to the same field.

#### 8.23.3.5 TableField & TableField::operator= ( const char * *data* )

Overridden assignment operator for `const char *`.

**Parameters**

| | |
|---|---|
| *data* | The new contents of the field. |

**Returns**

A reference to the same field.

#### 8.23.3.6 TableField & TableField::operator= ( const std::string & *data* )

Overridden assignment operator for `std::string`.

**Parameters**

| | |
|---|---|
| *data* | The new contents of the field. |

**Returns**

A reference to the same field.

#### 8.23.3.7 char & TableField::operator[] ( const size_t *idx* )

Overridden index operator.

**Parameters**

| | |
|---|---|
| *idx* | The desired index. |

**Returns**

A reference to the character at the specified index.

#### 8.23.3.8 const char & TableField::operator[] ( const size_t *idx* ) const

Overridden index operator.

**Parameters**

| | |
|---|---|
| *idx* | The desired index. |

**Returns**

A const reference to the character at the specified index.

### 8.23.4 Friends And Related Function Documentation

#### 8.23.4.1 std::ostream& operator<< ( std::ostream & *out,* const TableField & *field* ) [friend]

Overridden << operator for printing a field.

**Parameters**

| | |
|---|---|
| *out* | The ostream to which to print. |
| *field* | A reference to the field. |

**Returns**

A reference to `out`.

### 8.23.5 Member Data Documentation

#### 8.23.5.1 std::string gldb::TableField::m_data [private]

The field contents

The documentation for this class was generated from the following files:

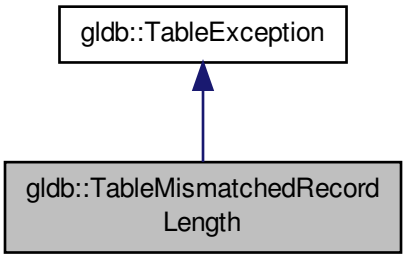
- lib/database/tablefield.h
- lib/database/tablefield.cpp

## 8.24 gldb::TableMismatchedRecordLength Class Reference

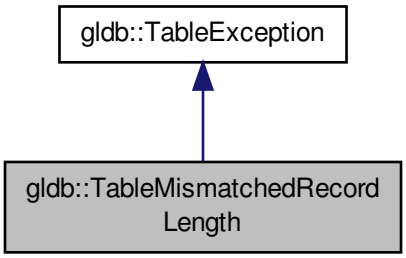
Mismatched record length exception class.

`#include <table.h>`

Inheritance diagram for gldb::TableMismatchedRecordLength:




Collaboration diagram for gldb::TableMismatchedRecordLength:




## Public Member Functions

- TableMismatchedRecordLength (const std::string &msg)

  *Constructor.*

### 8.24.1 Detailed Description

Mismatched record length exception class.

### 8.24.2 Constructor & Destructor Documentation

#### 8.24.2.1 gldb::TableMismatchedRecordLength::TableMismatchedRecordLength ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

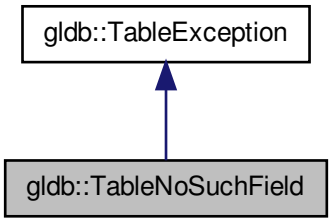
- lib/database/table.h

## 8.25 gldb::TableNoSuchField Class Reference

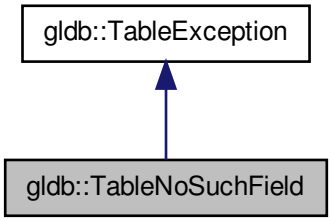
No such field exception class.

`#include <table.h>`

Inheritance diagram for gldb::TableNoSuchField:




Collaboration diagram for gldb::TableNoSuchField:




### Public Member Functions

- TableNoSuchField (const std::string &msg)

  *Constructor.*

### 8.25.1 Detailed Description

No such field exception class.

### 8.25.2 Constructor & Destructor Documentation

#### 8.25.2.1 gldb::TableNoSuchField::TableNoSuchField ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

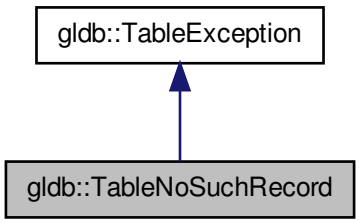
- lib/database/table.h

## 8.26 gldb::TableNoSuchRecord Class Reference

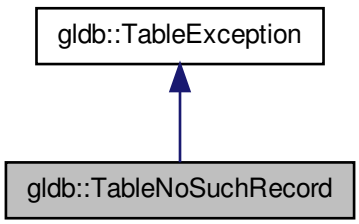
No such record exception class.

`#include <table.h>`

Inheritance diagram for gldb::TableNoSuchRecord:




Collaboration diagram for gldb::TableNoSuchRecord:




### Public Member Functions

- TableNoSuchRecord (const std::string &msg)

*Constructor.*

### 8.26.1 Detailed Description

No such record exception class.

### 8.26.2 Constructor & Destructor Documentation

#### 8.26.2.1 gldb::TableNoSuchRecord::TableNoSuchRecord ( const std::string & *msg* ) [inline], [explicit]

Constructor.

**Parameters**

| | |
|---|---|
| *msg* | Database error message |

The documentation for this class was generated from the following file:

- lib/database/table.h

## 8.27 gldb::TableRow Class Reference

Database table row class.

`#include <tablerow.h>`

### Public Member Functions

- TableRow ()
- TableRow (const size_t size)
  *Constructor with initial number of fields.*
- TableRow (std::vector< std::string > &vec)
  *Constructor with string vector.*
- ~TableRow ()
- size_t size () const
  *Returns the number of fields.*
- TableField & operator[] (const size_t idx)
  *Overridden index operator.*
- const TableField & operator[] (const size_t idx) const
  *Overridden index operator.*
- void append_field (const char *new_field)
  *Appends a field to the row.*
- void append_field (const std::string &new_field)
  *Appends a field to the row.*
- void append_field (const TableField &new_field)
  *Appends a field to the row.*
- void print (std::ostream &stream) const
  *Prints a row.*
- std::string record_string (const std::vector< bool > &quoted)
  *Creates a comma separated string of fields.*
- std::string record_string ()
  *Creates an unquoted comma separated string of fields.*

### Private Attributes

- std::vector< TableField > m_fields

## 8.27.1 Detailed Description

Database table row class.

## 8.27.2 Constructor & Destructor Documentation

### 8.27.2.1 TableRow::TableRow ( )

Default constructor

### 8.27.2.2 TableRow::TableRow ( const size_t *size* ) [explicit]

Constructor with initial number of fields.

**Parameters**

| | |
|---|---|
| *size* | The initial number of fields. |

### 8.27.2.3 TableRow::TableRow ( std::vector< std::string > & *vec* ) [explicit]

Constructor with string vector.

**Parameters**

| | |
|---|---|
| *vec* | The vector. |

### 8.27.2.4 TableRow::~TableRow ( )

Destructor

## 8.27.3 Member Function Documentation

### 8.27.3.1 void TableRow::append_field ( const char * *new_field* )

Appends a field to the row.

**Parameters**

| | |
|---|---|
| *new_field* | The contents of the new field. |

### 8.27.3.2 void TableRow::append_field ( const std::string & *new_field* )

Appends a field to the row.

**Parameters**

| | |
|---|---|
| *new_field* | The contents of the new field. |

#### 8.27.3.3 void TableRow::append_field ( const TableField & *new_field* )

Appends a field to the row.

**Parameters**

| | |
|---|---|
| *new_field* | A field from which to copy. |

#### 8.27.3.4 TableField & TableRow::operator[] ( const size_t *idx* )

Overridden index operator.

**Parameters**

| | |
|---|---|
| *idx* | The zero-based index of the field. |

**Returns**

A reference to the field at the specified index.

#### 8.27.3.5 const TableField & TableRow::operator[] ( const size_t *idx* ) const

Overridden index operator.

**Parameters**

| | |
|---|---|
| *idx* | The zero-based index of the field. |

**Returns**

A const reference to the field at the specified index.

#### 8.27.3.6 void TableRow::print ( std::ostream & *stream* ) const

Prints a row.

**Parameters**

| | |
|---|---|
| *stream* | The ostream to which to print. |

#### 8.27.3.7 std::string TableRow::record_string ( const std::vector< bool > & *quoted* )

Creates a comma separated string of fields.

**Parameters**

| | |
|---|---|
| *quoted* | A vector of `bool`, for each field `true` means that field will be enclosed in single quotes in the comma separated string, `false` means it will not be. |

**Returns**

The comma separated string.

#### 8.27.3.8 std::string TableRow::record_string ( )

Creates an unquoted comma separated string of fields.

**Returns**

The unquoted comma separated string.

#### 8.27.3.9 size_t TableRow::size ( ) const

Returns the number of fields.

**Returns**

The number of fields.

### 8.27.4 Member Data Documentation

#### 8.27.4.1 std::vector<TableField> gldb::TableRow::m_fields [private]

A vector of fields

The documentation for this class was generated from the following files:

- lib/database/tablerow.h
- lib/database/tablerow.cpp

# Chapter 9

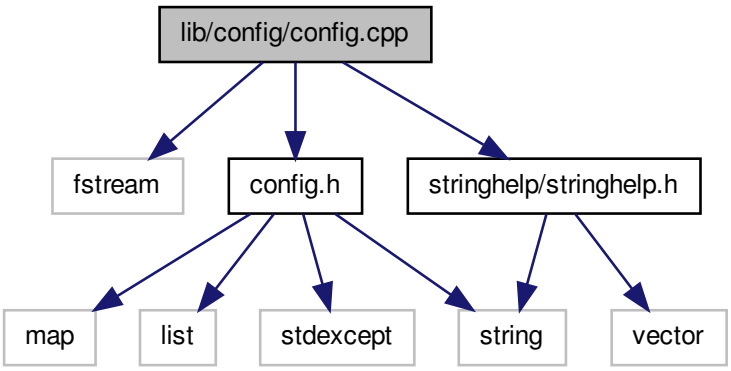
# File Documentation

## 9.1 lib/config/config.cpp File Reference

Implementation of program configurations class.

```
#include <fstream>
#include "config.h"
#include "stringhelp/stringhelp.h"
```

Include dependency graph for config.cpp:




### 9.1.1 Detailed Description

Implementation of program configurations class.

**Author**

Paul Griffiths

**Copyright**

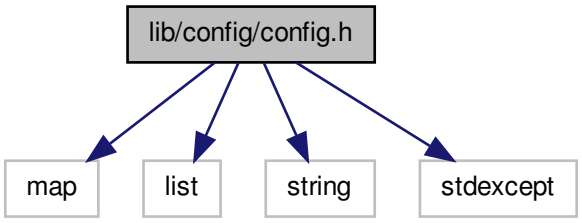
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.2 lib/config/config.h File Reference

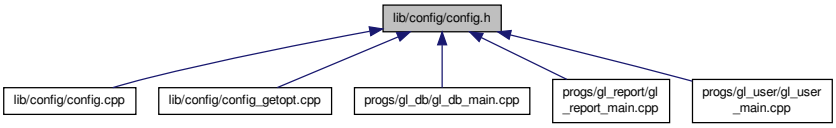
Interface to program configurations class.

```
#include <map>
#include <list>
#include <string>
#include <stdexcept>
```

Include dependency graph for config.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class genleg::ConfigException

  *Configuration module exception base class.*
- class genleg::ConfigOptionNotSet

  *Exception class for option not set.*
- class genleg::ConfigBadOption

  *Exception class for bad provided option.*
- class genleg::ConfigCouldNotOpenFile

  *Exception class for when conf file cannot be opened.*
- class genleg::ConfigBadConfigFile

  *Exception class for badly formed configuration file.*
- class genleg::Config

  *Configuration options class.*

### Enumerations

- enum genleg::Argument

  *Enumeration class for option argument specifications.*

### 9.2.1 Detailed Description

Interface to program configurations class.

**Author**

Paul Griffiths

**Copyright**

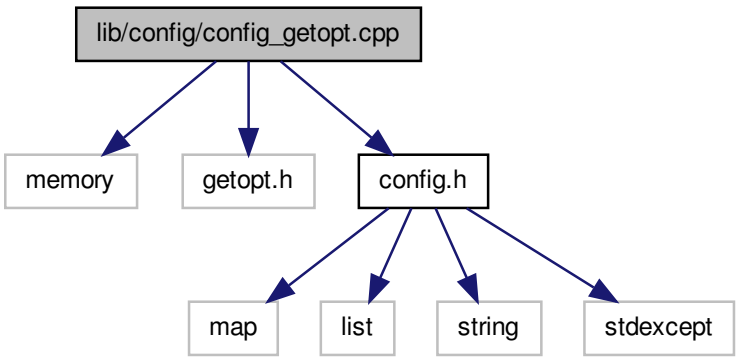
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.3 lib/config/config_getopt.cpp File Reference

Implementation of command line functionality.

```
#include <memory>
#include <getopt.h>
#include "config.h"
```

Include dependency graph for config_getopt.cpp:




### Macros

- #define _XOPEN_SOURCE 600

### 9.3.1 Detailed Description

Implementation of command line functionality. Included in separate file to isolate usage of non-standard getopt library.

**Author**

Paul Griffiths

**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

### 9.3.2 Macro Definition Documentation

#### 9.3.2.1 #define _XOPEN_SOURCE 600

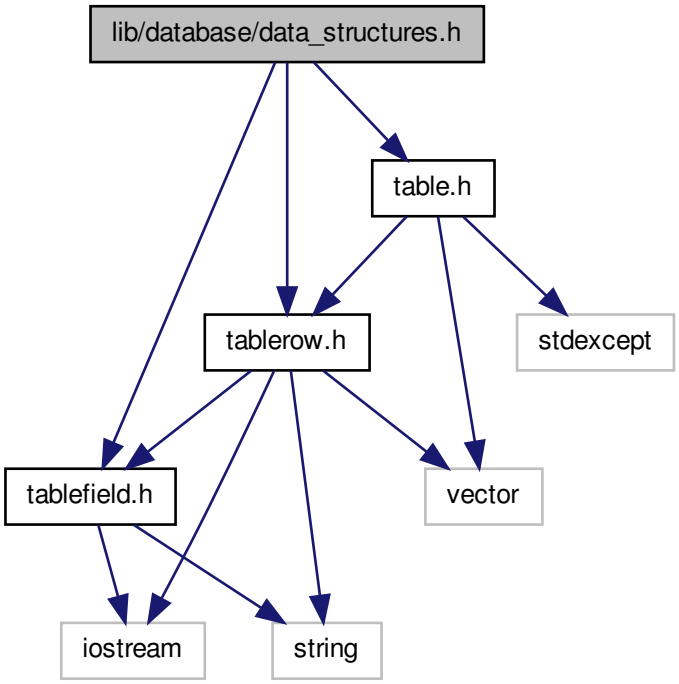
UNIX feature test macro for getopt library

## 9.4 lib/database/data_structures.h File Reference

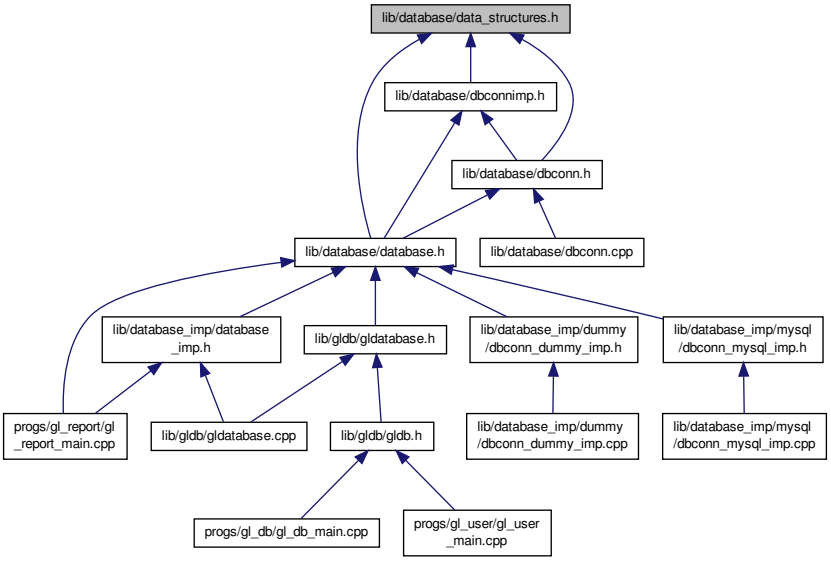
Main interface to database data structures.

```
#include "tablefield.h"
#include "tablerow.h"
#include "table.h"
```

Include dependency graph for data_structures.h:



This graph shows which files directly or indirectly include this file:




### 9.4.1 Detailed Description

Main interface to database data structures.

#### Author

Paul Griffiths

#### Copyright

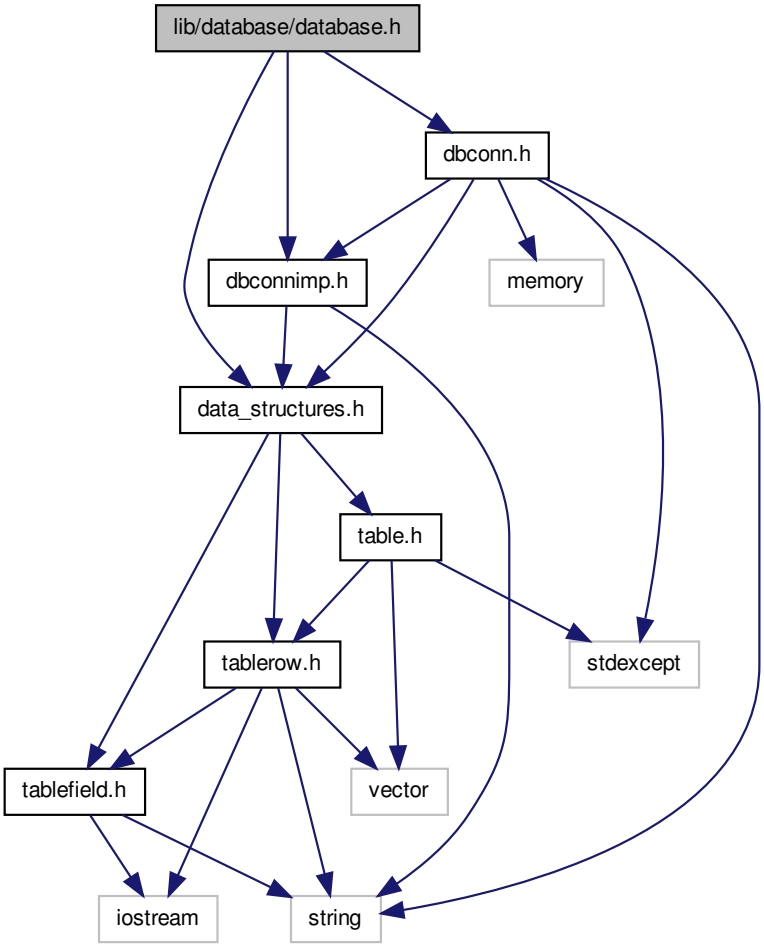
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.5 lib/database/database.h File Reference

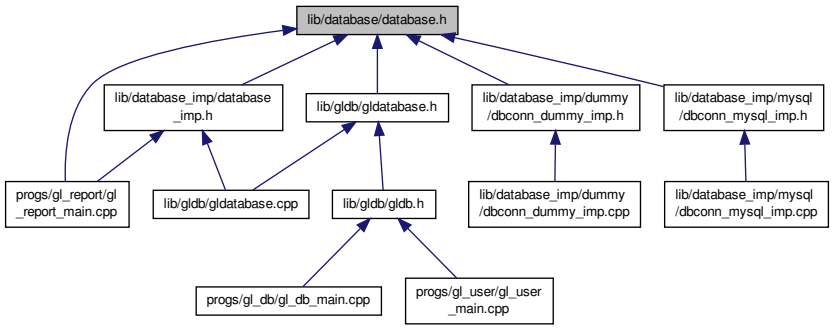
User interface to database functionality.

```
#include "data_structures.h"
#include "dbconnimp.h"
#include "dbconn.h"
```

Include dependency graph for database.h:




This graph shows which files directly or indirectly include this file:



### 9.5.1 Detailed Description

User interface to database functionality.

**Author**

Paul Griffiths

**Copyright**

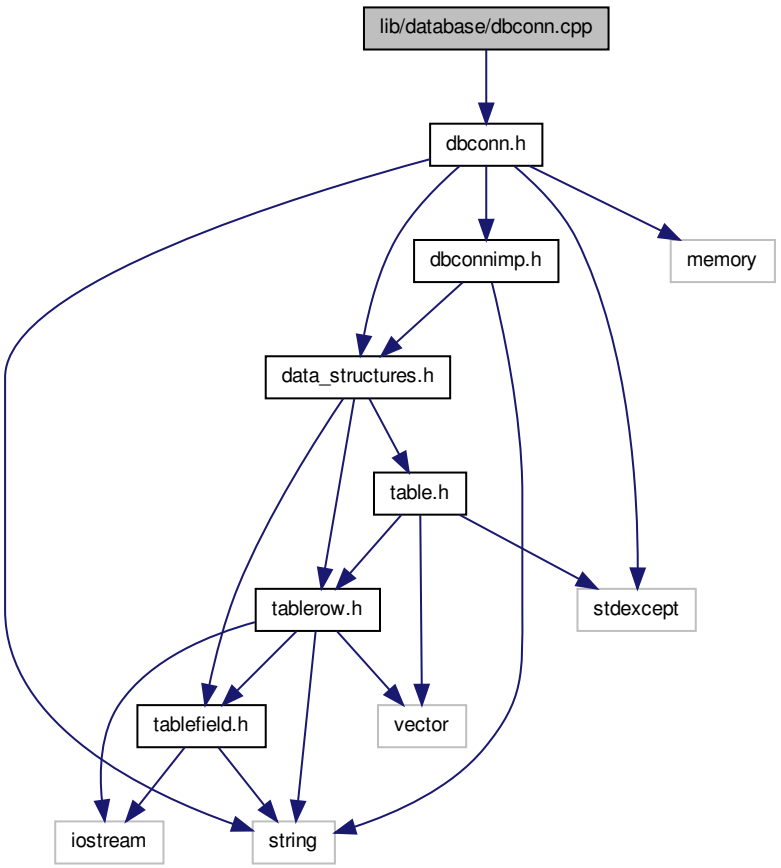
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.6 lib/database/dbconn.cpp File Reference

Implementation of database connection class.

```
#include "dbconn.h"
```

Include dependency graph for dbconn.cpp:




### 9.6.1 Detailed Description

Implementation of database connection class.

**Author**

Paul Griffiths

**Copyright**

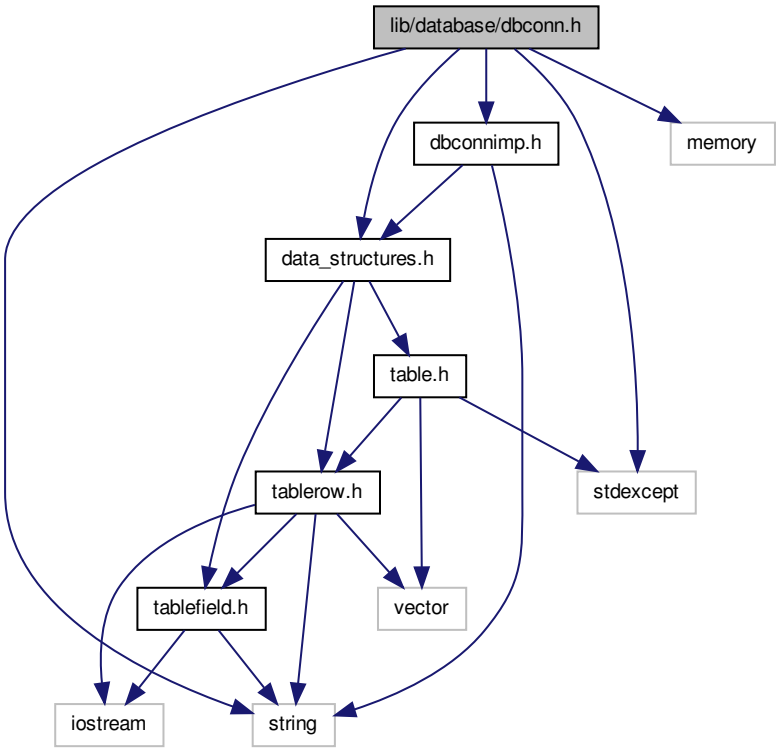
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.7 lib/database/dbconn.h File Reference

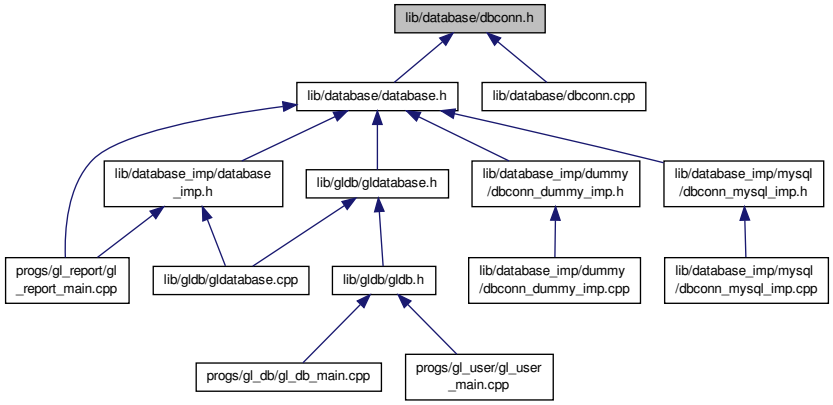
Interface to database connection base class.

```
#include <string>
#include <memory>
#include <stdexcept>
#include "data_structures.h"
#include "dbconnimp.h"
```

Include dependency graph for dbconn.h:



This graph shows which files directly or indirectly include this file:




## Classes

- class gldb::DBConnException

  *Base database connection exception class.*
- class gldb::DBConnCouldNotConnect

  *Could not connect to database exception class.*
- class gldb::DBConnCouldNotQuery

  *Could not execute database query exception class.*
- class gldb::DBConn

  *Database connection class.*

### 9.7.1 Detailed Description

Interface to database connection base class.

#### Author

Paul Griffiths

#### Copyright

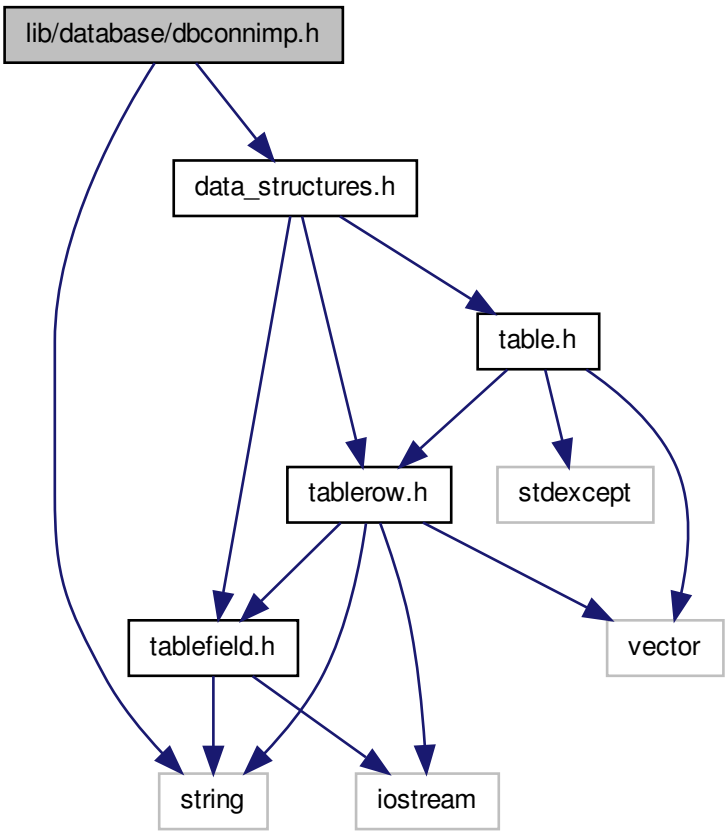
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.8 lib/database/dbconnimp.h File Reference

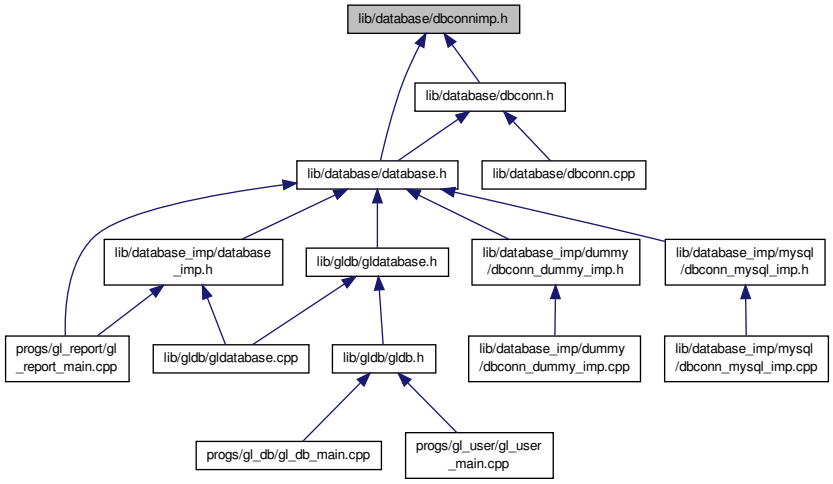
Interface to abstract database implementation base class.

```
#include <string>
#include "data_structures.h"
```

Include dependency graph for dbconnimp.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class gldb::DBConnImp

  *Abstract database implementation base class.*

### 9.8.1 Detailed Description

Interface to abstract database implementation base class.

#### Author

Paul Griffiths

#### Copyright

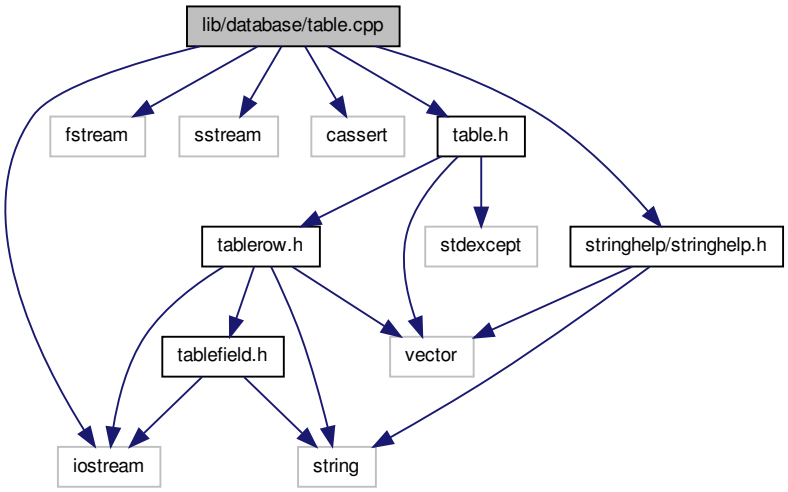
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.9 lib/database/table.cpp File Reference

Implementation of database table data structure.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <cassert>
#include "table.h"
#include "stringhelp/stringhelp.h"
```

Include dependency graph for table.cpp:




### 9.9.1 Detailed Description

Implementation of database table data structure.

**Author**

Paul Griffiths

**Copyright**

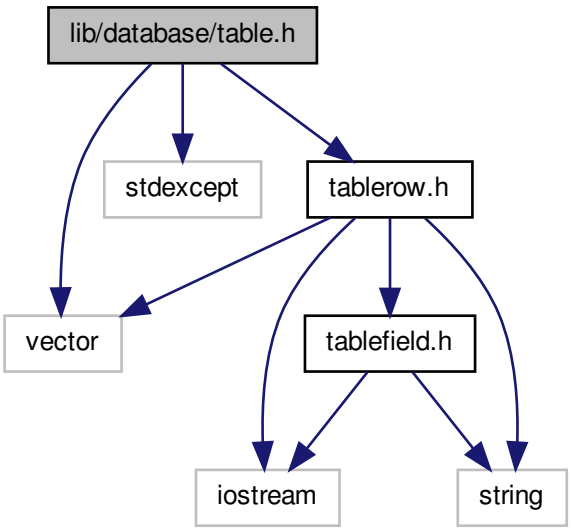
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.10 lib/database/table.h File Reference

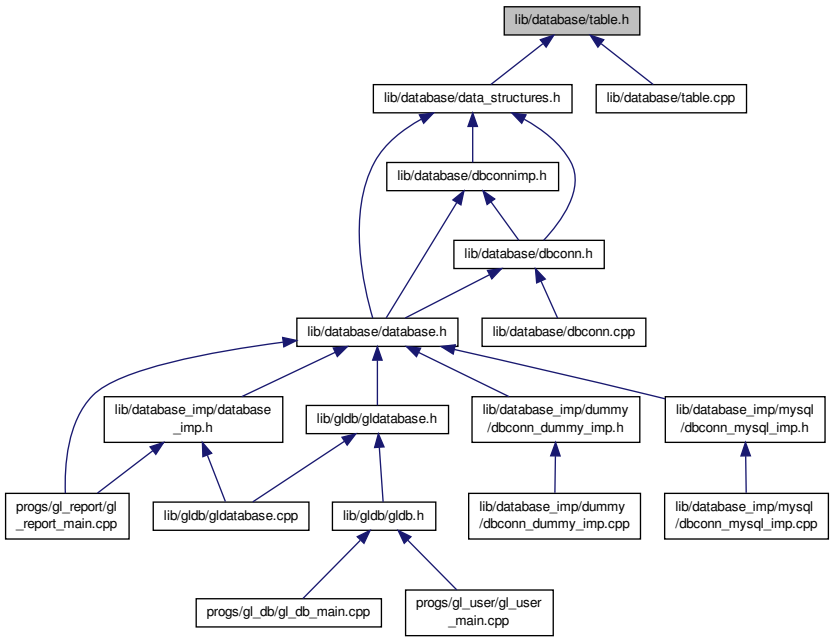
Interface to database table data structure.

```
#include <vector>
#include <stdexcept>
#include "tablerow.h"
```

Include dependency graph for table.h:



This graph shows which files directly or indirectly include this file:




## Classes

- class gldb::TableException

  *Base database connection exception class.*
- class gldb::TableNoSuchField

  *No such field exception class.*
- class gldb::TableNoSuchRecord

  *No such record exception class.*
- class gldb::TableMismatchedRecordLength

  *Mismatched record length exception class.*
- class gldb::TableBadInputFile

  *Could not connect to database exception class.*
- class gldb::TableCouldNotOpenInputFile

  *Could not connect to database exception class.*
- class gldb::Table

  *Database table class.*

### 9.10.1 Detailed Description

Interface to database table data structure.

#### Author

Paul Griffiths

#### Copyright

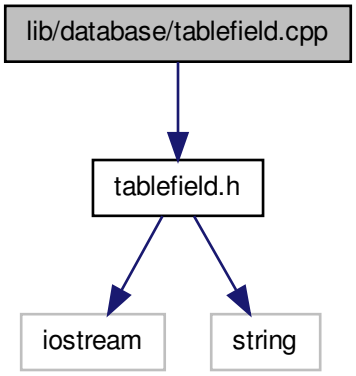
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.11 lib/database/tablefield.cpp File Reference

Implementation of database table field class.

```
#include "tablefield.h"
```

Include dependency graph for tablefield.cpp:




### 9.11.1 Detailed Description

Implementation of database table field class.

**Author**

Paul Griffiths

**Copyright**

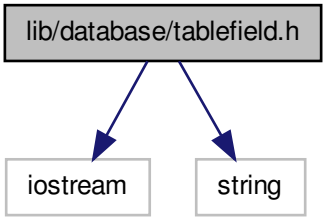
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.12 lib/database/tablefield.h File Reference

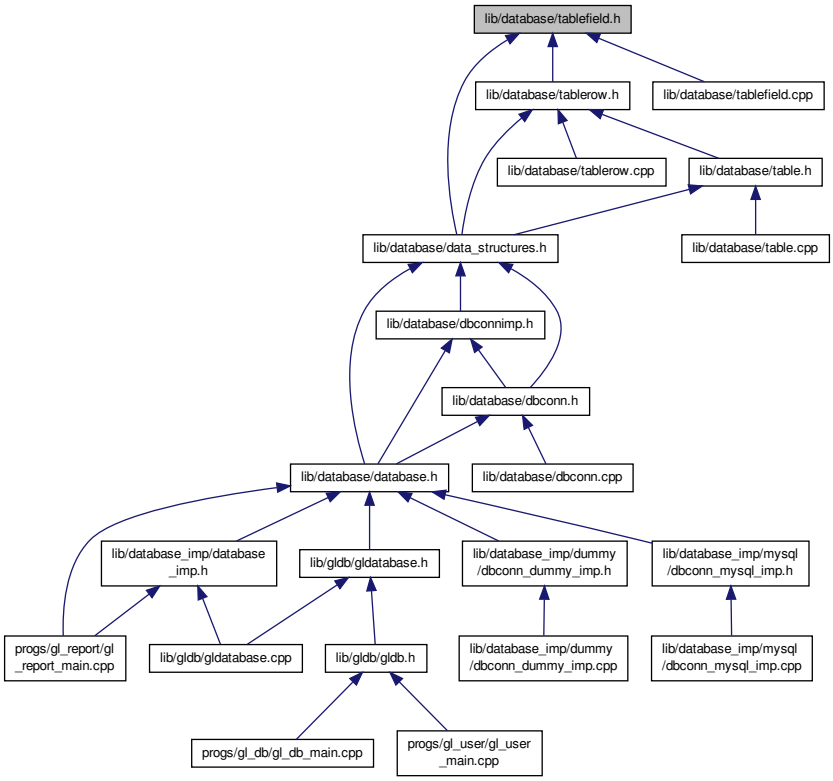
Interface to database table field class.

```
#include <iostream>
#include <string>
```

Include dependency graph for tablefield.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class gldb::TableField

  *Database table field class.*

## Functions

- std::ostream & **gldb::operator**<< (std::ostream &out, const TableField &field)

*Overridden << operator for printing a field.*

### 9.12.1 Detailed Description

Interface to database table field class.

**Author**

Paul Griffiths

**Copyright**

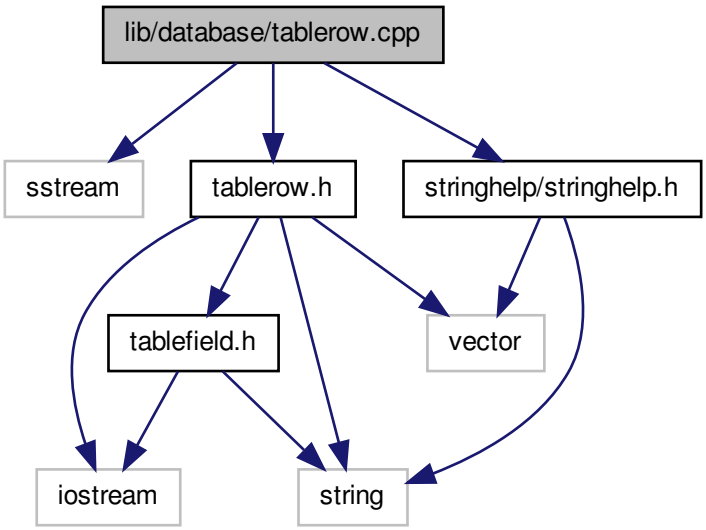
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.13 lib/database/tablerow.cpp File Reference

Implementation of database table row data structure.

```
#include <sstream>
#include "tablerow.h"
#include "stringhelp/stringhelp.h"
```

Include dependency graph for tablerow.cpp:




### 9.13.1 Detailed Description

Implementation of database table row data structure.

**Author**

Paul Griffiths

**Copyright**

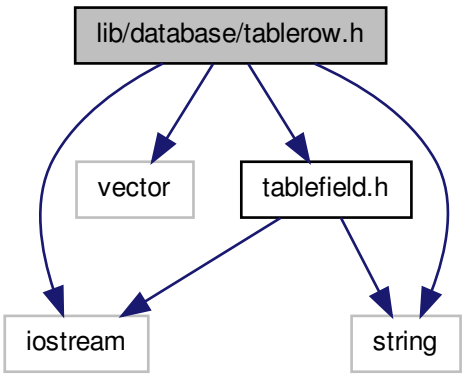
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.14 lib/database/tablerow.h File Reference

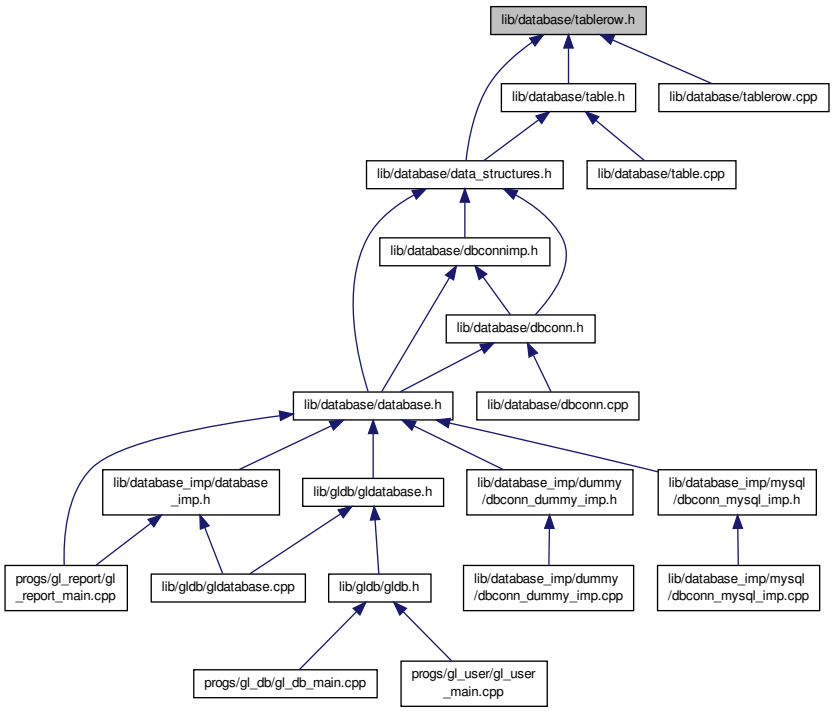
Interface to database table row data structure.

```
#include <iostream>
#include <vector>
#include <string>
#include "tablefield.h"
```

Include dependency graph for tablerow.h:



This graph shows which files directly or indirectly include this file:




## Classes

- class gldb::TableRow

  *Database table row class.*

### 9.14.1 Detailed Description

Interface to database table row data structure.

#### Author

Paul Griffiths

#### Copyright

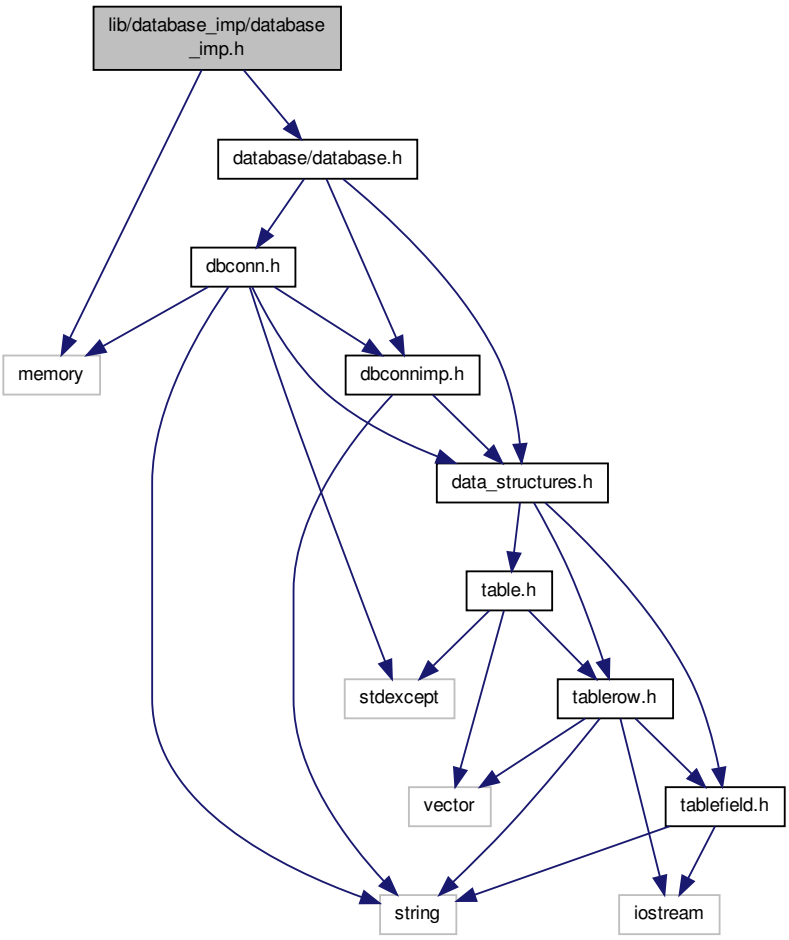
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.15 lib/database_imp/database_imp.h File Reference

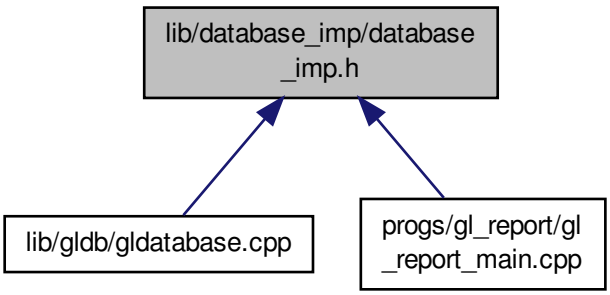
Interface to database implementation factory function.

```
#include <memory>
#include "database/database.h"
```

Include dependency graph for database_imp.h:




This graph shows which files directly or indirectly include this file:



### Functions

- DBConnImp * gldb::get_connection (const std::string database, const std::string hostname, const std::string username, const std::string password)

  *Creates and returns a pointer to a database implementation.*

- std::string gldb::get_database_type ()

  *Returns the name of the compiled-in database type.*

#### 9.15.1 Detailed Description

Interface to database implementation factory function.

**Author**

Paul Griffiths

**Copyright**

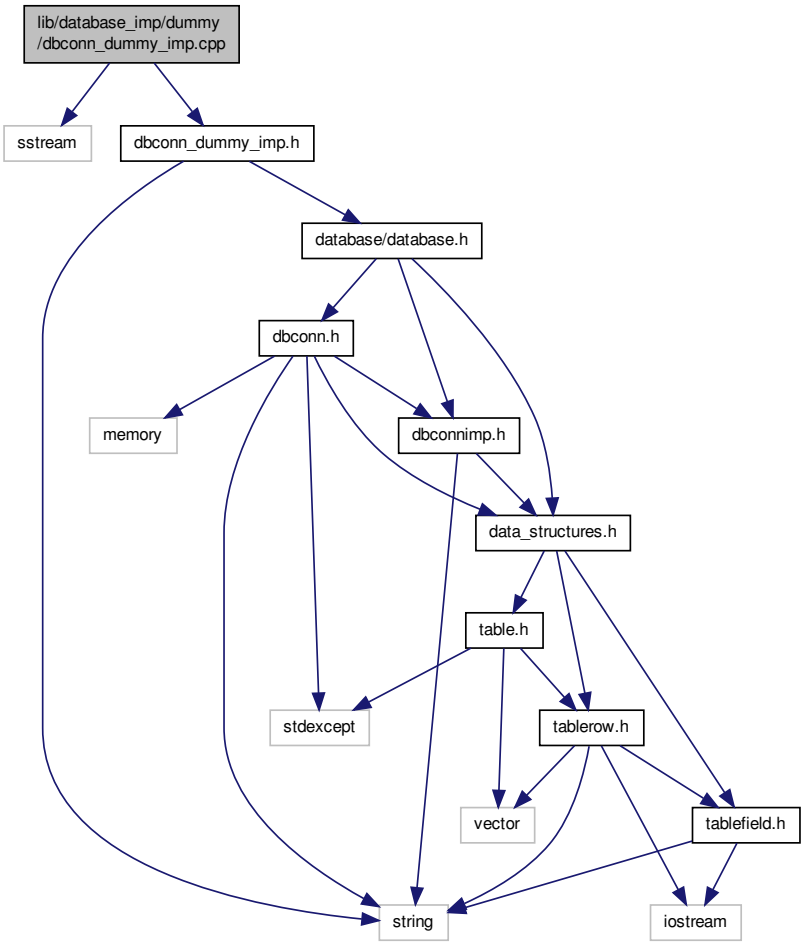
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.16 lib/database_imp/dummy/dbconn_dummy_imp.cpp File Reference

Implementation of Dummy database connection implementation class.

```
#include <sstream>
#include "dbconn_dummy_imp.h"
```

Include dependency graph for dbconn_dummy_imp.cpp:




### 9.16.1 Detailed Description

Implementation of Dummy database connection implementation class.

**Author**

Paul Griffiths

**Copyright**

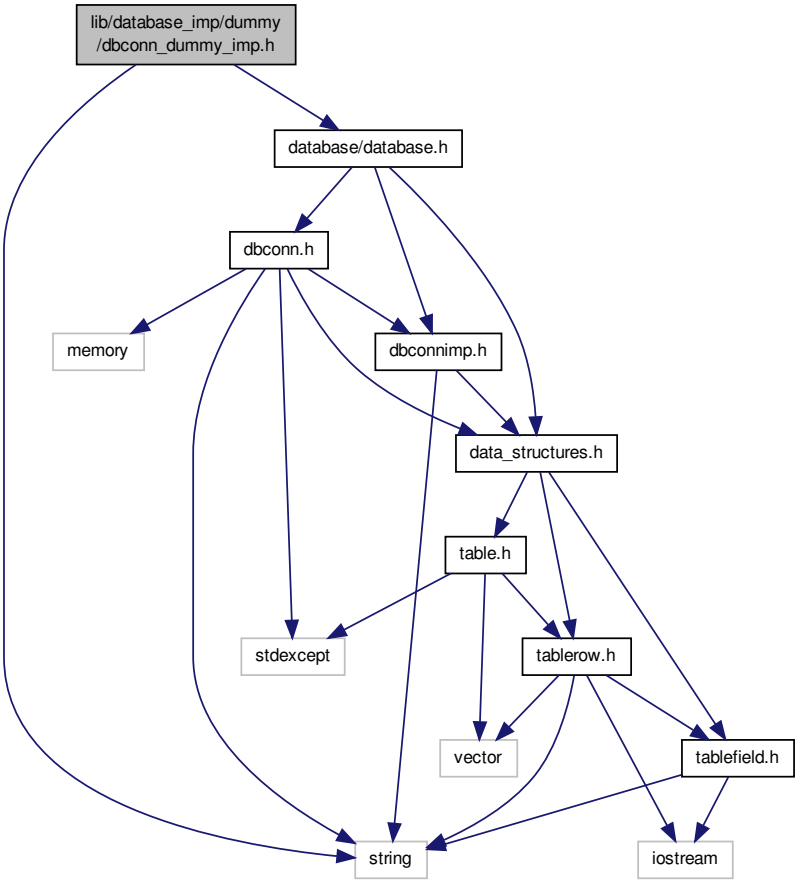
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.17 lib/database_imp/dummy/dbconn_dummy_imp.h File Reference

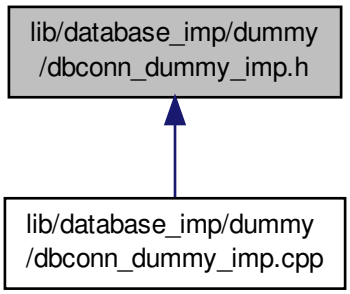
Interface to dummy database connection implementation class.

```
#include <string>
#include "database/database.h"
```

Include dependency graph for dbconn_dummy_imp.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class gldb::DBConnDummy

  *Dummy database implementation class.*

### 9.17.1 Detailed Description

Interface to dummy database connection implementation class.

**Author**

Paul Griffiths

**Copyright**

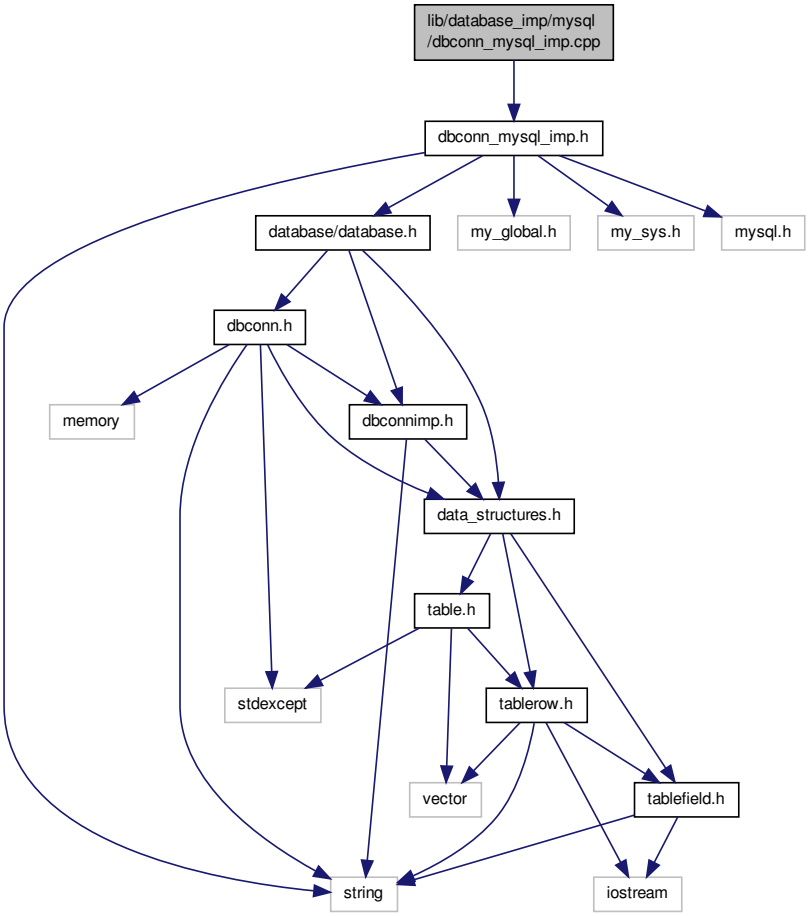
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.18 lib/database_imp/mysql/dbconn_mysql_imp.cpp File Reference

Implementation of MySQL database connection implementation class.

```
#include "dbconn_mysql_imp.h"
```

Include dependency graph for dbconn_mysql_imp.cpp:




### 9.18.1 Detailed Description

Implementation of MySQL database connection implementation class.

#### Author

Paul Griffiths

#### Copyright

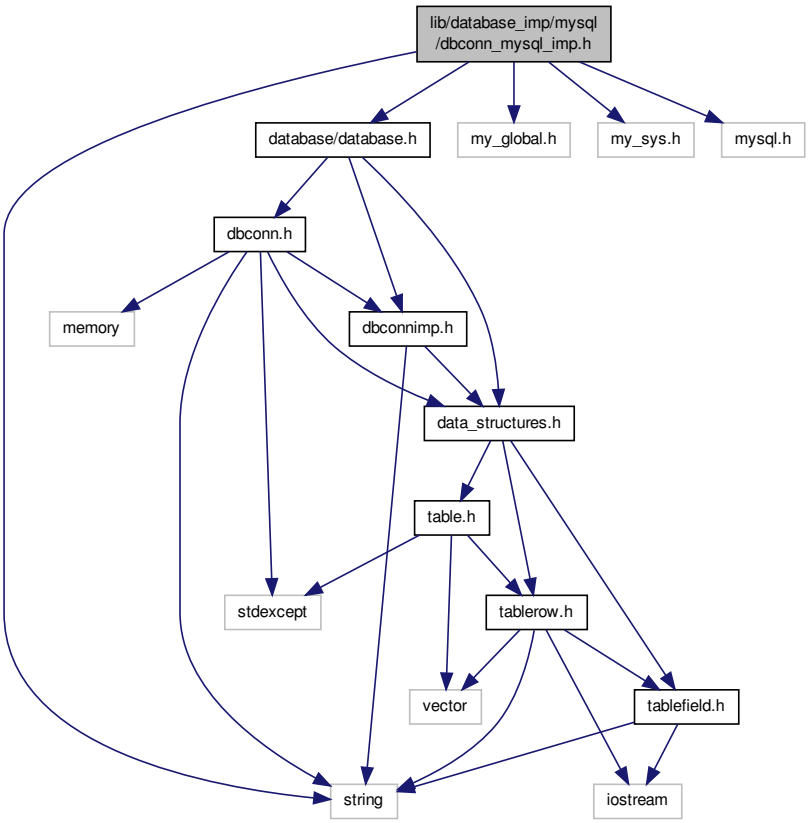
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.19 lib/database_imp/mysql/dbconn_mysql_imp.h File Reference

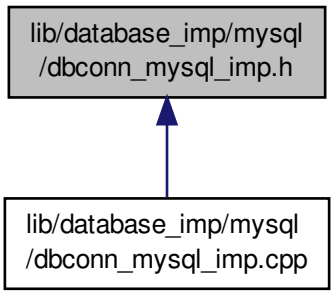
Interface to MySQL database connection implementation class.

```
#include <string>
#include "database/database.h"
#include <my_global.h>
#include <my_sys.h>
#include <mysql.h>
```

Include dependency graph for dbconn_mysql_imp.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class gldb::DBConnMySQL

  *MySQL database implementation class.*

### 9.19.1 Detailed Description

Interface to MySQL database connection implementation class.

#### Author

Paul Griffiths

#### Copyright

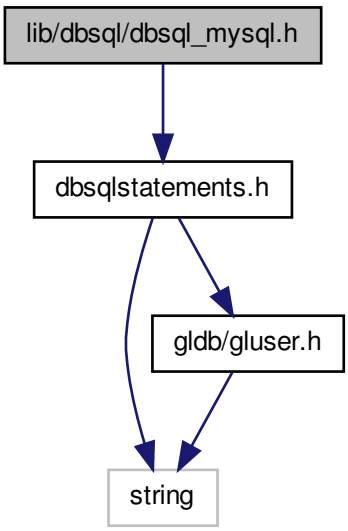
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.20 lib/dbsql/dbsql_mysql.h File Reference

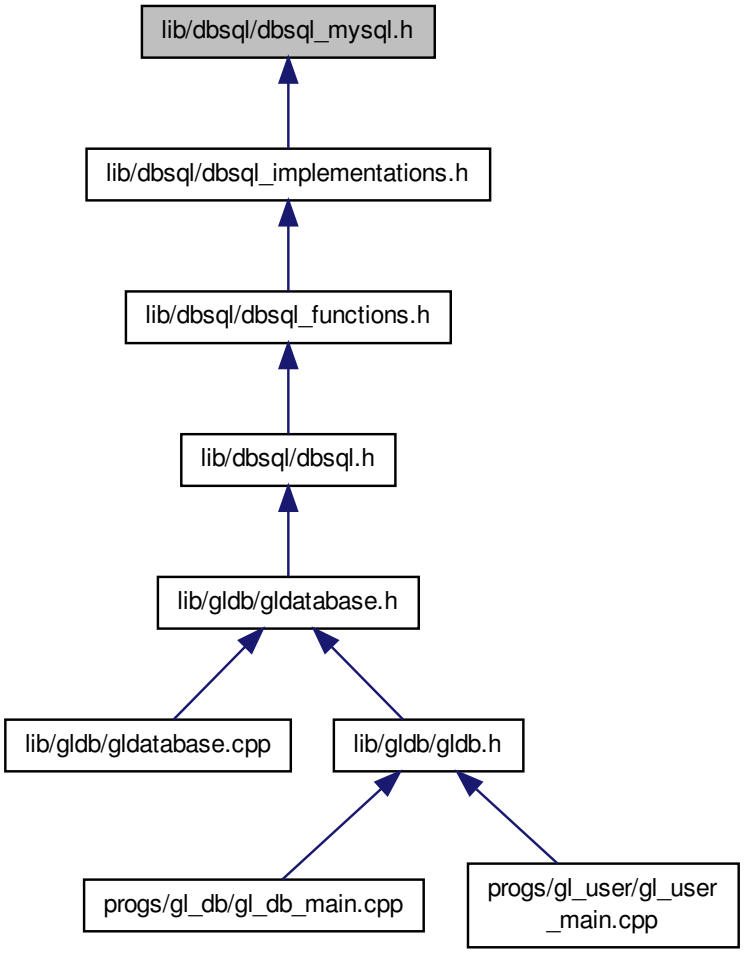
Interface to MySQL SQL statement class.

```
#include "dbsqlstatements.h"
```

Include dependency graph for dbsql_mysql.h:



This graph shows which files directly or indirectly include this file:




## Classes

- class genleg::DBSQLMySQL

  *MySQL SQL statements class.*

### 9.20.1 Detailed Description

Interface to MySQL SQL statement class. Interface to MySQL SQL statement class

**Author**

Paul Griffiths

**Copyright**

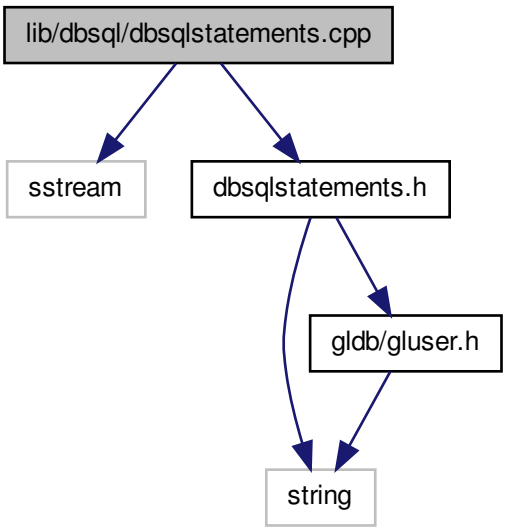
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.21 lib/dbsql/dbsqlstatements.cpp File Reference

Implementation of SQL statement class.

```
#include <sstream>
#include "dbsqlstatements.h"
```

Include dependency graph for dbsqlstatements.cpp:




### 9.21.1 Detailed Description

Implementation of SQL statement class. Implementation of SQL statement class

**Author**

Paul Griffiths

**Copyright**

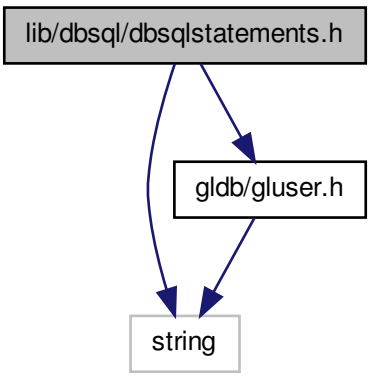
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.22 lib/dbsql/dbsqlstatements.h File Reference

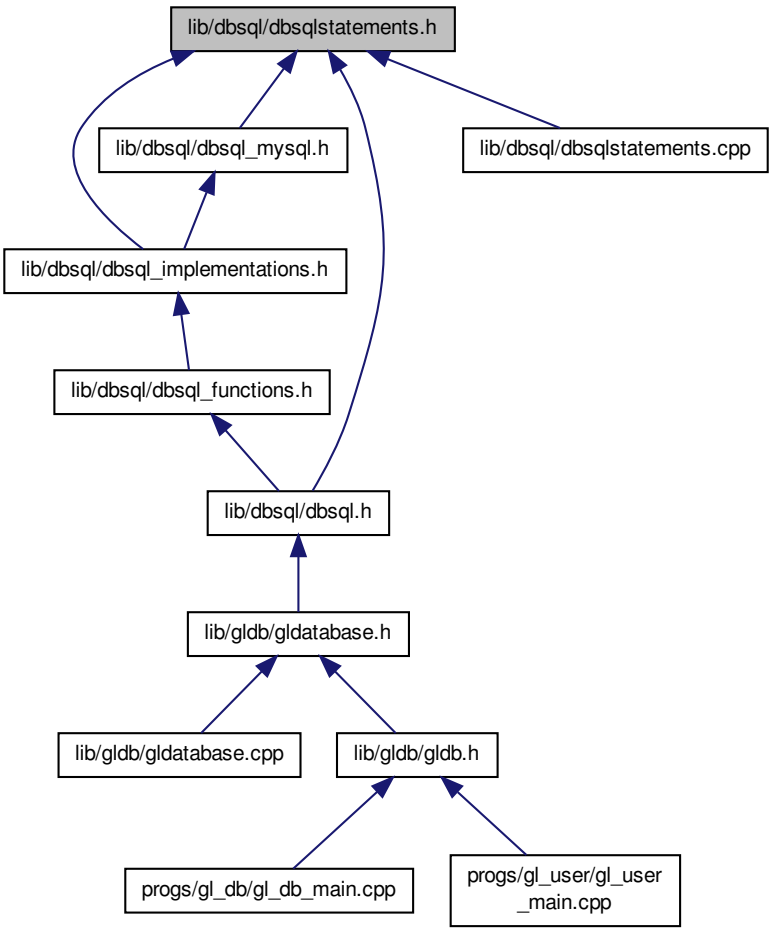
Interface to SQL statement class.

```
#include <string>
#include "gldb/gluser.h"
```

Include dependency graph for dbsqlstatements.h:



This graph shows which files directly or indirectly include this file:




## Classes

- class genleg::DBSQLStatements

  *SQL statements class.*

### 9.22.1 Detailed Description

Interface to SQL statement class.

**Author**

Paul Griffiths

**Copyright**

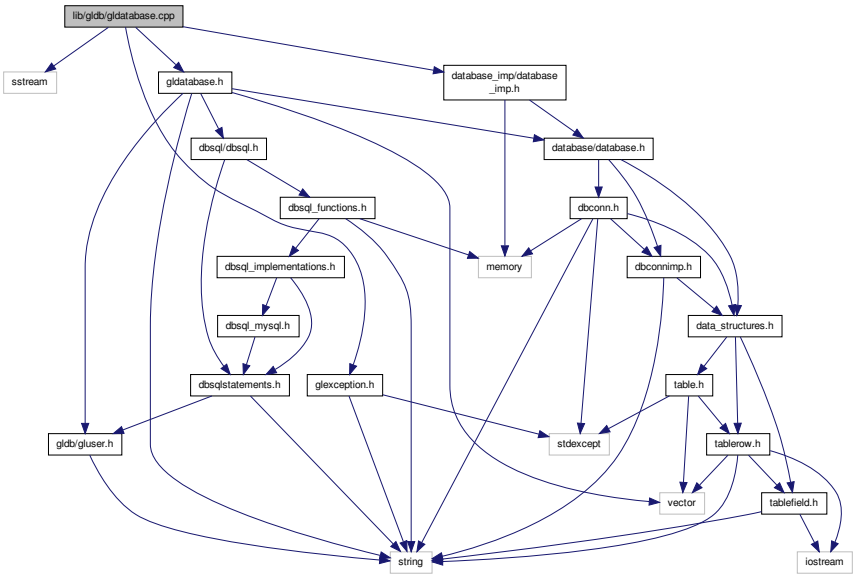
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.23 lib/gldb/gldatabase.cpp File Reference

Implementation of General Ledger database class.

```
#include <sstream>
#include "gldatabase.h"
#include "glexception.h"
#include "database_imp/database_imp.h"
```

Include dependency graph for gldatabase.cpp:




### Functions

- **m_views** ({"current_trial_balance","check_total","all_jes"})

### 9.23.1 Detailed Description

Implementation of General Ledger database class.

#### Author

Paul Griffiths

#### Copyright

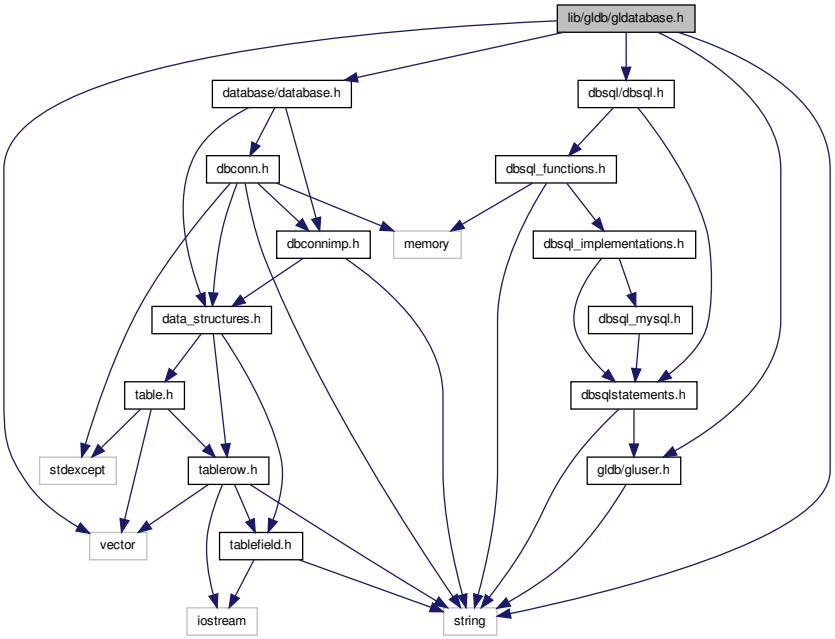
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.24 lib/gldb/gldatabase.h File Reference

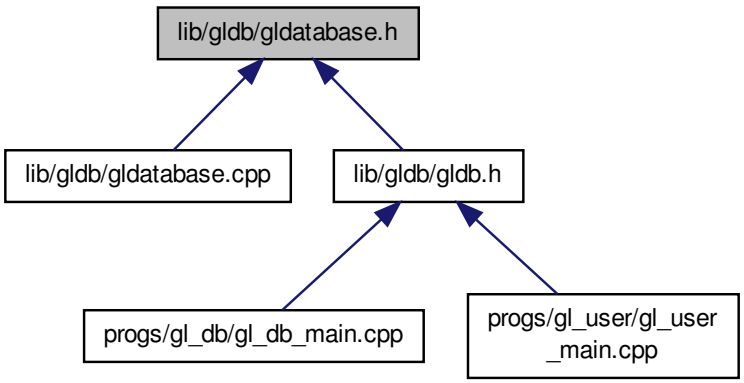
Interface to General Ledger database class.

```
#include <vector>
#include <string>
#include "database/database.h"
#include "dbsql/dbsql.h"
#include "gluser.h"
```

Include dependency graph for gldatabase.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class genleg::GLDatabase

*General ledger database class.*

### 9.24.1 Detailed Description

Interface to General Ledger database class.

#### Author

Paul Griffiths

#### Copyright

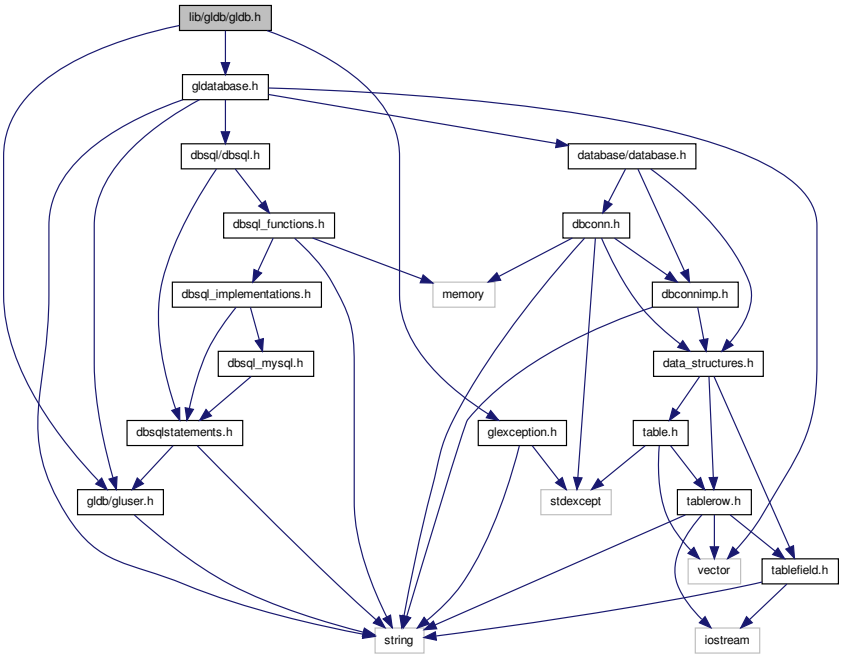
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.25 lib/gldb/gldb.h File Reference

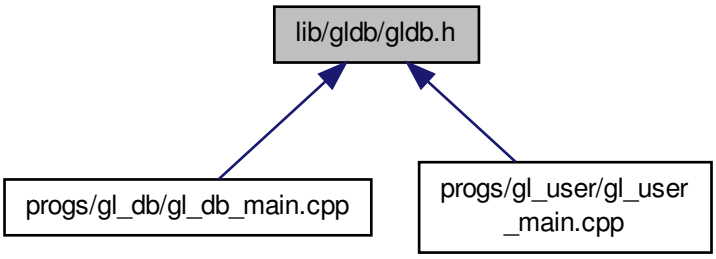
User interface to General Ledger database module.

```
#include "glexception.h"
#include "gldatabase.h"
#include "gluser.h"
```

Include dependency graph for gldb.h:



This graph shows which files directly or indirectly include this file:




### 9.25.1 Detailed Description

User interface to General Ledger database module.

**Author**

Paul Griffiths

**Copyright**

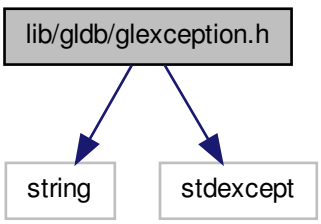
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.26 lib/gldb/glexception.h File Reference

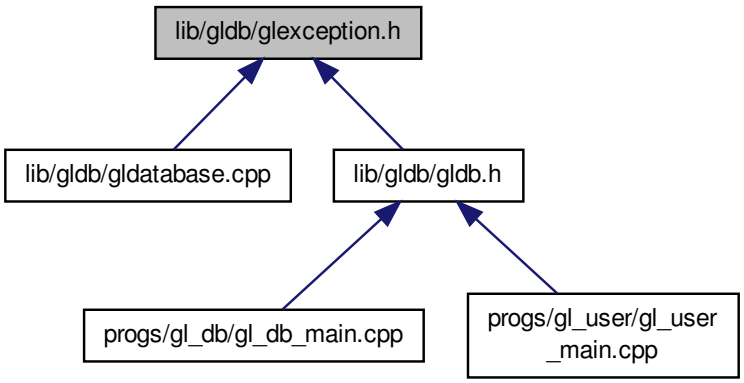
Interface to General Ledger base exception class.

```
#include <string>
#include <stdexcept>
```

Include dependency graph for glexception.h:



This graph shows which files directly or indirectly include this file:




## Classes

- class genleg::GLDBException

  *Base general ledger database exceptionc class.*

### 9.26.1 Detailed Description

Interface to General Ledger base exception class.

**Author**

Paul Griffiths

**Copyright**

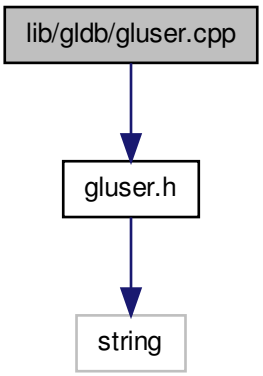
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.27 lib/gldb/gluser.cpp File Reference

Implementation of user class.

```
#include "gluser.h"
```

Include dependency graph for gluser.cpp:




### 9.27.1 Detailed Description

Implementation of user class. Implementation of user class

**Author**

Paul Griffiths

**Copyright**

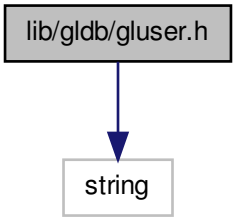
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.28 lib/gldb/gluser.h File Reference

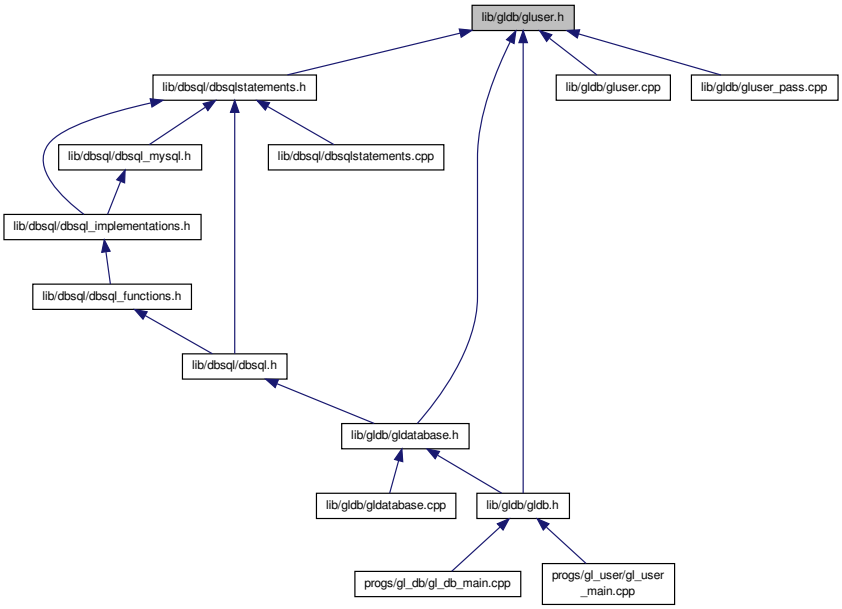
Interface to user class.

```
#include <string>
```

Include dependency graph for gluser.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class genleg::GLUser

  *General ledger user class.*

### 9.28.1 Detailed Description

Interface to user class. Interface to user class

**Author**

Paul Griffiths

**Copyright**

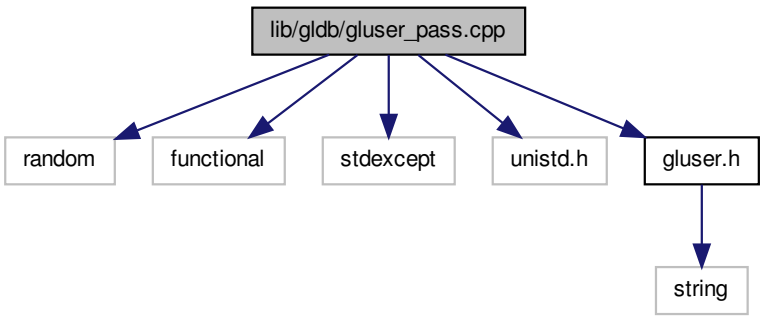
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.29 lib/gldb/gluser_pass.cpp File Reference

Implementation of password functions for user class.

```
#include <random>
#include <functional>
#include <stdexcept>
#include <unistd.h>
#include "gluser.h"
```

Include dependency graph for gluser_pass.cpp:




### Macros

- #define _XOPEN_SOURCE 600

### Functions

- static std::string generate_salt ()

  *Generates a random two-character salt for crypt()*

### 9.29.1 Detailed Description

Implementation of password functions for user class.

**Todo** Implement a better form of password encryption. In particular, these functions are not re-entrant, and only use the first 8 characters of the password.

**Author**

Paul Griffiths

**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

### 9.29.2 Macro Definition Documentation

#### 9.29.2.1 #define _XOPEN_SOURCE 600

UNIX feature test macro

### 9.29.3 Function Documentation

#### 9.29.3.1 static std::string generate_salt ( ) [static]

Generates a random two-character salt for crypt()

**Returns**

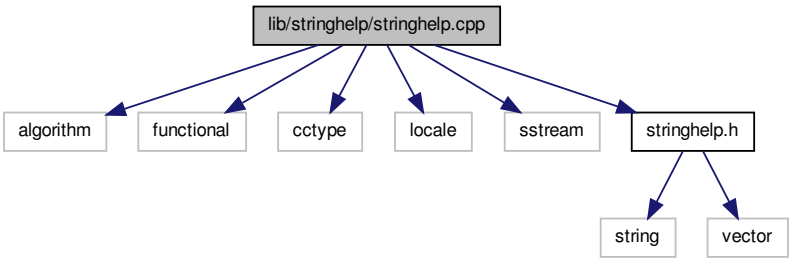
The two-character salt.

## 9.30 lib/stringhelp/stringhelp.cpp File Reference

Implementation of string helper functions.

```
#include <algorithm>
#include <functional>
#include <cctype>
#include <locale>
#include <sstream>
#include "stringhelp.h"
```

Include dependency graph for stringhelp.cpp:




### 9.30.1 Detailed Description

Implementation of string helper functions.

**Author**

Paul Griffiths

**Copyright**

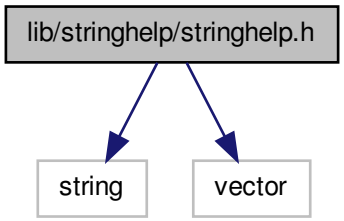
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.31 lib/stringhelp/stringhelp.h File Reference

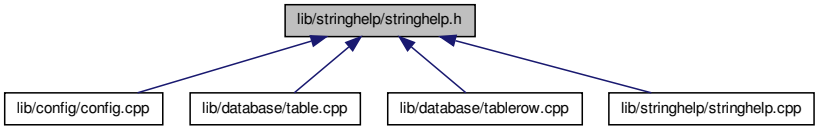
Interface to string helper functions.

```
#include <string>
#include <vector>
```

Include dependency graph for stringhelp.h:




This graph shows which files directly or indirectly include this file:




### Functions

- std::string & pgstring::trim_front (std::string &s)

  *Trims leading whitespace from a string.*
- std::string & pgstring::trim_back (std::string &s)

  *Trims trailing whitespace from a string.*
- std::string & pgstring::trim (std::string &s)

  *Trims leading and trailing whitespace from a string.*
- std::vector< std::string > pgstring::split (const std::string &s, const char delim)

  *Splits a delimited string into tokens.*
- std::vector< std::string > & pgstring::split (std::vector< std::string > &vec, const std::string &s, const char delim)

  *Splits a delimited string into tokens.*
- bool **pgstring::next_content_line** (std::istream &ifs, std::string &s)

  *Gets the next content line from a stream.*

- std::vector< std::string > & **pgstring::content_lines** (std::vector< std::string > &vec, std::istream &ifs)

  *Populates a vector of content lines from a stream.*
- std::vector< std::vector < std::string > > & **pgstring::split_lines** (std::vector< std::vector< std::string >> &vec, std::istream &ifs, const char delim)

  *Populates a vector of vectors of fields from a stream.*
- std::string & **pgstring::join** (std::vector< std::string > &vec, std::string &s, const char delim)

  *Joins a vector of strings into a delimited line.*
- bool **pgstring::replace** (std::string &str, const std::string &from, const std::string &to)

  *Replaces a substring with another string.*

### 9.31.1 Detailed Description

Interface to string helper functions.

#### Author

Paul Griffiths

#### Copyright

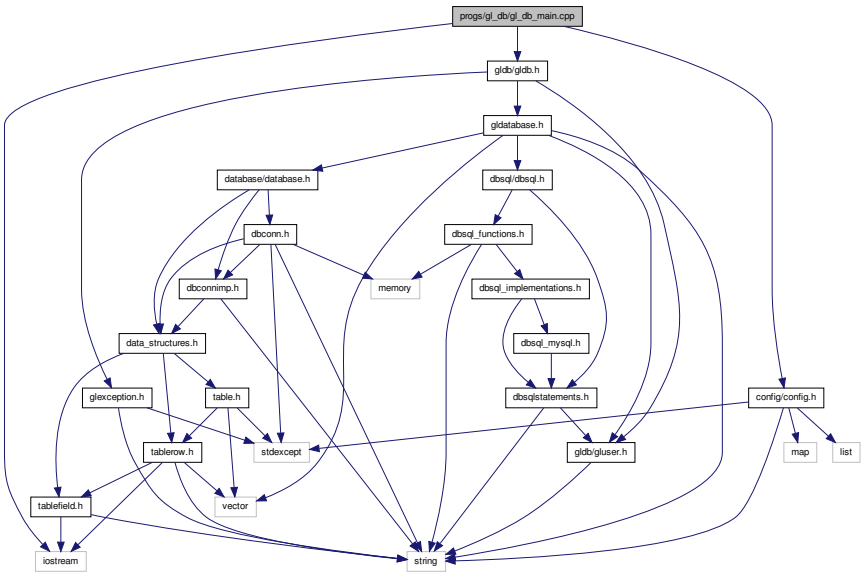
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.32 progs/gl_db/gl_db_main.cpp File Reference

Main functionality for gl_db program.

```
#include <iostream>
#include "gldb/gldb.h"
#include "config/config.h"
```

Include dependency graph for gl_db_main.cpp:



### Functions

- static void set_configuration (Config &config, int argc, char *argv[])

  *Sets program configuration options.*
- static bool check_help_and_version (const Config &config)

  *Prints help or version messages if requested.*
- static bool check_db_parameters (const Config &config)

  *Checks if database, hostname and username were provided.*
- static void print_usage_message ()

  *Prints a program usage message.*
- static void print_version_message ()

  *Prints a program version message.*
- static void print_help_message ()

  *Prints a program help message.*
- static std::string login (void)

  *Gets a password from the terminal.*
- int main (int argc, char *argv[])

  *Main function.*

### Variables

- static const char * progname = "gl_db"

  *Static variable for program name.*

#### 9.32.1 Detailed Description

Main functionality for gl_db program.

**Author**

Paul Griffiths

**Copyright**

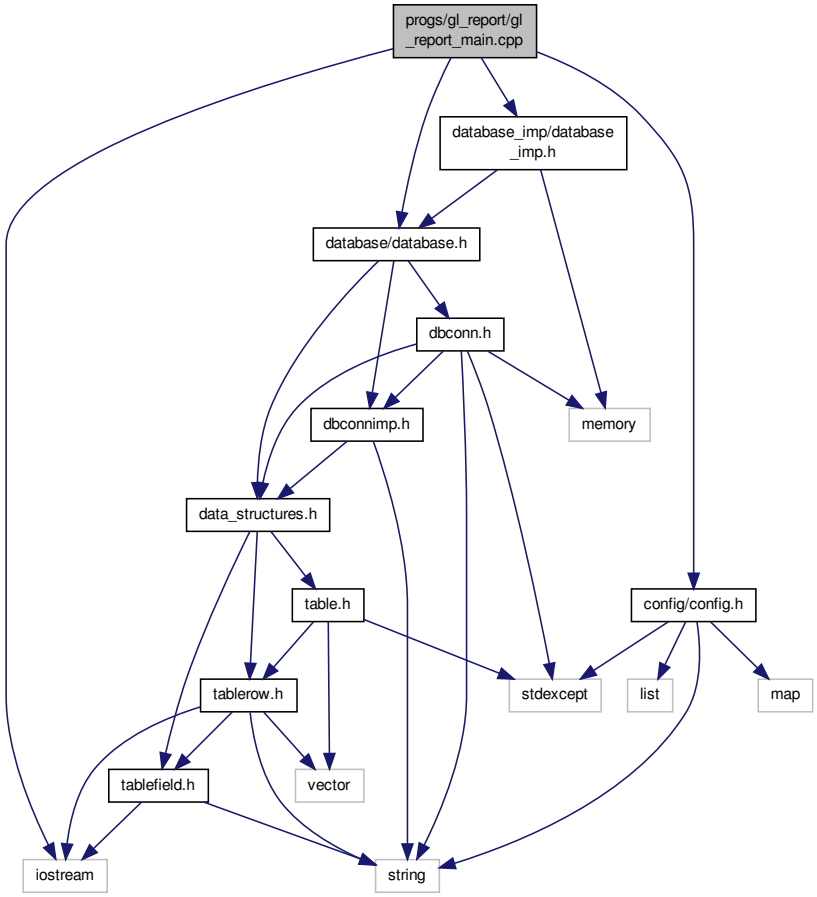
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.33 progs/gl_report/gl_report_main.cpp File Reference

Main functionality for gl_report program.

```
#include <iostream>
#include "database/database.h"
#include "database_imp/database_imp.h"
#include "config/config.h"
```

Include dependency graph for gl_report_main.cpp:




## Functions

- static void set_configuration (genleg::Config &config, int argc, char ∗argv[])

  *Sets program configuration options.*
- static void print_usage_message ()

  *Prints a program usage message.*
- static void print_version_message ()

  *Prints a program version message.*
- static void print_help_message ()

  *Prints a program help message.*
- static std::string login (void)

  *Gets a password from the terminal.*
- int main (int argc, char ∗argv[])

  *Main function.*

## Variables

- static const char ∗ progname = "gl_report"

  *Static variable for program name.*

### 9.33.1 Detailed Description

Main functionality for gl_report program.

**Author**

Paul Griffiths

**Copyright**

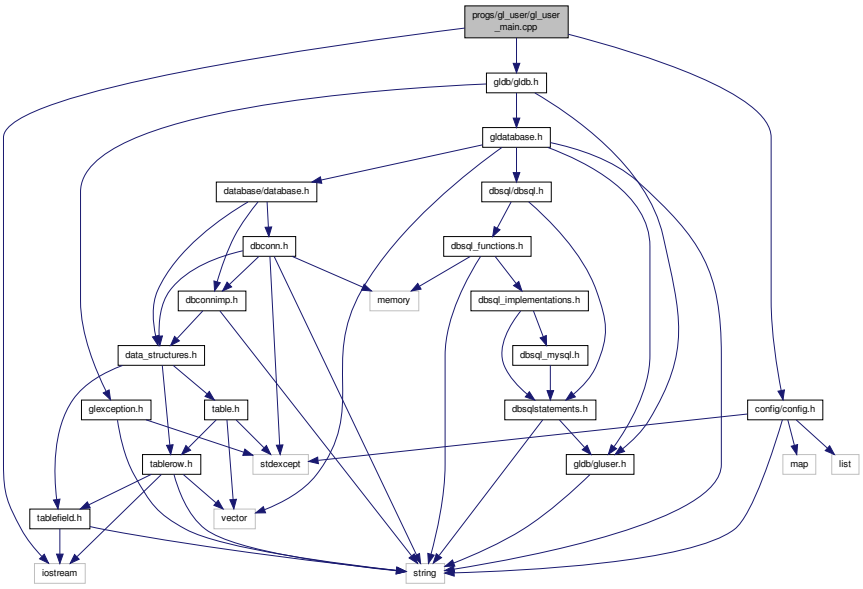
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

## 9.34 progs/gl_user/gl_user_main.cpp File Reference

Main functionality for gl_user program.

```
#include <iostream>
#include "gldb/gldb.h"
#include "config/config.h"
```

Include dependency graph for gl_user_main.cpp:




### Functions

- static void set_configuration (Config &config, int argc, char ∗argv[])

  *Sets program configuration options.*
- static bool check_help_and_version (const Config &config)

  *Prints help or version messages if requested.*
- static bool check_db_parameters (const Config &config)

  *Checks if database, hostname and username were provided.*
- static void show_user_details (const GLUser &user)

  *Outputs details for a user.*
- static void enable_user (GLUser &user, Config &config, GLDatabase &gdb)

*Enables or disables a user.*

- static void set_user_password (GLUser &user, Config &config, GLDatabase &gdb)

  *Sets a user's password.*
- static void check_user_password (GLUser &user, Config &config)

  *Checks a user's password.*
- static void print_usage_message ()

  *Prints a program usage message.*
- static void print_version_message ()

  *Prints a program version message.*
- static void print_help_message ()

  *Prints a program help message.*
- static std::string login (void)

  *Gets a password from the terminal.*
- int main (int argc, char *argv[])

  *Main function.*

## Variables

- static const char * progname = "gl_user"

  *Static variable for program name.*

### 9.34.1 Detailed Description

Main functionality for gl_user program.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. http://www.gnu.org/licenses/

### 9.34.2 Function Documentation

#### 9.34.2.1 static bool check_db_parameters ( const Config & *config* ) [static]

Checks if database, hostname and username were provided.

#### Parameters

| | |
|---|---|
| *config* | Reference to a Config object. |

#### Returns

`true` if the information was provided, `false` otherwise.

#### 9.34.2.2 static bool check_help_and_version ( const Config & *config* ) [static]

Prints help or version messages if requested.

**Parameters**

| | |
|---|---|
| *config* | Reference to a Config object. |

**Returns**

`true` if the help or version message was requested, `false` otherwise.

#### 9.34.2.3 static void check_user_password ( GLUser & *user,* Config & *config* ) [static]

Checks a user's password.

**Parameters**

| | |
|---|---|
| *user* | Reference to user. |
| *config* | Reference to program configuration options. |

#### 9.34.2.4 static void enable_user ( GLUser & *user,* Config & *config,* GLDatabase & *gdb* ) [static]

Enables or disables a user.

**Parameters**

| | |
|---|---|
| *user* | Reference to user. |
| *config* | Reference to program configuration. |
| *gdb* | Reference to database object. |

#### 9.34.2.5 static std::string login ( void ) [static]

Gets a password from the terminal.

**Returns**

The password.

#### 9.34.2.6 int main ( int *argc,* char * *argv[]* )

Main function.

**Parameters**

| | |
|---|---|
| *argc* | Number of command line arguments. |
| *argv* | Command line arguments. |

**Returns**

Exit status code.

#### 9.34.2.7 static void set_configuration ( Config & *config,* int *argc,* char * *argv[]* ) [static]

Sets program configuration options.

**Parameters**

| | |
|---|---|
| *config* | Reference to a Config object. |
| *argc* | `argc` passed to `main()`. |
| *argv* | `argv` passed to `main()`. |

#### 9.34.2.8 static void set_user_password ( GLUser & *user*, Config & *config*, GLDatabase & *gdb* ) [static]

Sets a user's password.

**Parameters**

| | |
|---|---|
| *user* | Reference to user. |
| *config* | Reference to program configuration. |
| *gdb* | Reference to database object. |

#### 9.34.2.9 static void show_user_details ( const GLUser & *user* ) [static]

Outputs details for a user.

**Parameters**

| | |
|---|---|
| *user* | Reference to user. |

# Index