xapian-core

1.2.8

Generated by Doxygen 1.5.9

Wed Dec 14 05:37:34 2011

Contents

Chapter 1

Deprecated List

Member Xapian::Enquire::get_mset(Xapian::doccount first, Xapian::doccount maxitems, Xapian::doccount checkat
this parameter is deprecated - use the newer MatchSpy class and add_matchspy() method instead.

Class Xapian::MultiValueSorter This class is deprecated - you should migrate to using MultiValueKeyMaker instead. Note that MultiValueSorter::add() becomes MultiValueKeyMaker::add_value(), but the sense of the direction flag is reversed (to be consistent with Enquire::set_sort_by_value()), so:

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Namespace Documentation

6.1 Xapian Namespace Reference

The Xapian namespace contains public interfaces for the Xapian library.

Namespaces

- namespace Auto

  Database factory functions which determine the database type automatically.

- namespace Brass

  Database factory functions for the brass backend.

- namespace Chert

  Database factory functions for the chert backend.

- namespace Flint

  Database factory functions for the flint backend.

- namespace InMemory

  Database factory functions for the inmemory backend.

- namespace Remote

  Database factory functions for the remote backend.

- namespace Unicode

  Functions associated with handling Unicode characters.

Classes

- class Error

  All exceptions thrown by Xapian are subclasses of Xapian::Error.

- class LogicError

  The base class for exceptions indicating errors in the program logic.

- class RuntimeError

  The base class for exceptions indicating errors only detectable at runtime.

- class AssertionError

  AssertionError is thrown if a logical assertion inside Xapian fails.

- class InvalidArgumentError

  InvalidArgumentError indicates an invalid parameter value was passed to the API.

- class InvalidOperationError

  InvalidOperationError indicates the API was used in an invalid way.

- class UnimplementedError

  UnimplementedError indicates an attempt to use an unimplemented feature.

- class DatabaseError

  DatabaseError indicates some sort of database related error.

- class DatabaseCorruptError

  DatabaseCorruptError indicates database corruption was detected.

- class DatabaseCreateError

  DatabaseCreateError indicates a failure to create a database.

- class DatabaseLockError

  DatabaseLockError indicates failure to lock a database.

- class DatabaseModifiedError

  DatabaseModifiedError indicates a database was modified.

- class DatabaseOpeningError

  DatabaseOpeningError indicates failure to open a database.

- class DatabaseVersionError

  DatabaseVersionError indicates that a database is in an unsupported format.

- class DocNotFoundError

  Indicates an attempt to access a document not present in the database.

- class FeatureUnavailableError

  Indicates an attempt to use a feature which is unavailable.

- class InternalError

  InternalError indicates a runtime problem of some sort.

- class NetworkError

  Indicates a problem communicating with a remote database.

- class NetworkTimeoutError

  Indicates a timeout expired while communicating with a remote database.

- class QueryParserError

  Indicates a query string can't be parsed.

- class SerialisationError

  Indicates an error in the std::string serialisation of an object.

- class RangeError

  RangeError indicates an attempt to access outside the bounds of a container.

- class Compactor

  Compact a database, or merge and compact several.

- class Database

  This class is used to access a database, or a group of databases.

- class WritableDatabase

  This class provides read/write access to a database.

- class Document

  A handle representing a document in a Xapian database.

- class MSet

  A match set (MSet).

- class MSetIterator

  An iterator pointing to items in an MSet.

- class ESet

  Class representing an ordered set of expand terms (an ESet).

- class ESetIterator

  Iterate through terms in the ESet.

- class RSet

  A relevance set (R-Set).

- class MatchDecider

  Base class for matcher decision functor.

- class Enquire

  This class provides an interface to the information retrieval system for the purpose of searching.

- class ErrorHandler

  Decide if a Xapian::Error exception should be ignored.

- class ExpandDecider

  Virtual base class for expand decider functor.

- class ExpandDeciderAnd

  ExpandDecider subclass which rejects terms using two ExpandDeciders.

- class ExpandDeciderFilterTerms

  ExpandDecider subclass which rejects terms in a specified list.

- class KeyMaker

  Virtual base class for key making functors.

- class MultiValueKeyMaker

  KeyMaker subclass which combines several values.

- class Sorter

  Virtual base class for sorter functor.

- class MultiValueSorter

  Sorter subclass which sorts by a several values.

- class MatchSpy

  Abstract base class for match spies.

- class ValueCountMatchSpy

  Class for counting the frequencies of values in the matching documents.

- class PositionIterator

  An iterator pointing to items in a list of positions.

- class PostingIterator

  An iterator pointing to items in a list of postings.

- class PostingSource

  Base class which provides an "external" source of postings.

- class ValuePostingSource

  A posting source which generates weights from a value slot.

- class ValueWeightPostingSource

  A posting source which reads weights from a value slot.

- class DecreasingValueWeightPostingSource

  Read weights from a value which is known to decrease as docid increases.

- class ValueMapPostingSource

  A posting source which looks up weights in a map using values as the key.

- class FixedWeightPostingSource

  A posting source which returns a fixed weight for all documents.

- class Query

  Class representing a query.

- class Stopper

  Base class for stop-word decision functor.

- class SimpleStopper

  Simple implementation of Stopper class - this will suit most users.

- struct ValueRangeProcessor

  Base class for value range processors.

- class StringValueRangeProcessor

  Handle a string range.

- class DateValueRangeProcessor

  Handle a date range.

- class NumberValueRangeProcessor

  Handle a number range.

- class QueryParser

  Build a Xapian::Query object from a user query string.

- class Registry

  Registry for user subclasses.

- struct StemImplementation

Class representing a stemming algorithm implementation.

- class Stem

  Class representing a stemming algorithm.

- class TermGenerator

  Parses a piece of text and generate terms.

- class TermIterator

  An iterator pointing to items in a list of terms.

- class Utf8Iterator

  An iterator which returns Unicode character values from a UTF-8 encoded string.

- class ValueIterator

  Class for iterating over document values.

- class ValueSetMatchDecider

  MatchDecider filtering results based on whether document values are in a user-defined set.

- class Weight

  Abstract base class for weighting schemes.

- class BoolWeight

  Class implementing a "boolean" weighting scheme.

- class BM25Weight

  Xapian::Weight subclass implementing the BM25 probabilistic formula.

- class TradWeight

  Xapian::Weight subclass implementing the traditional probabilistic formula.

Typedefs

- typedef unsigned doccount

  A count of documents.

- typedef int doccount_diff

  A signed difference between two counts of documents.

- typedef unsigned docid

  A unique identifier for a document.

- typedef double doclength

A normalised document length.

- typedef int percent

  The percentage score for a document in an MSet.

- typedef unsigned termcount

  A counts of terms.

- typedef int termcount_diff

  A signed difference between two counts of terms.

- typedef unsigned termpos

  A term position within a document or query.

- typedef int termpos_diff

  A signed difference between two term positions.

- typedef unsigned timeout

  A timeout value in milliseconds.

- typedef unsigned valueno

  The number for a value slot in a document.

- typedef int valueno_diff

  A signed difference between two value slot numbers.

- typedef double weight

  The weight of a document or term.

Functions

- bool operator== (const MSetIterator &a, const MSetIterator &b)

  Equality test for MSetIterator objects.

- bool operator!= (const MSetIterator &a, const MSetIterator &b)

  Inequality test for MSetIterator objects.

- bool operator== (const ESetIterator &a, const ESetIterator &b)

  Equality test for ESetIterator objects.

- bool operator!= (const ESetIterator &a, const ESetIterator &b)

  Inequality test for ESetIterator objects.

- bool operator== (const PositionIterator &a, const PositionIterator &b)

  Test equality of two PositionIterators.

- bool operator!= (const PositionIterator &a, const PositionIterator &b)

  Test inequality of two PositionIterators.

- bool operator== (const PostingIterator &a, const PostingIterator &b)

  Test equality of two PostingIterators.

- bool operator!= (const PostingIterator &a, const PostingIterator &b)

  Test inequality of two PostingIterators.

- std::string sortable_serialise (double value)

  Convert a floating point number to a string, preserving sort order.

- double sortable_unserialise (const std::string &value)

  Convert a string encoded using sortable_serialise *back to a floating point number.*

- bool operator== (const TermIterator &a, const TermIterator &b)

  Equality test for TermIterator objects.

- bool operator!= (const TermIterator &a, const TermIterator &b)

  Inequality test for TermIterator objects.

- bool operator== (const ValueIterator &a, const ValueIterator &b)

  Equality test for ValueIterator objects.

- bool operator!= (const ValueIterator &a, const ValueIterator &b)

  Inequality test for ValueIterator objects.

- const char * version_string ()

  Report the version string of the library which the program is linked with.

- int major_version ()

  Report the major version of the library which the program is linked with.

- int minor_version ()

  Report the minor version of the library which the program is linked with.

- int revision ()

  Report the revision of the library which the program is linked with.

Variables

- const int DB_CREATE_OR_OPEN = 1

  Open for read/write; create if no db exists.

- const int DB_CREATE = 2

  Create a new database; fail if db exists.

- const int DB_CREATE_OR_OVERWRITE = 3

  Overwrite existing db; create if none exists.

- const int DB_OPEN = 4

  Open for read/write; fail if no db exists.

- const valueno BAD_VALUENO = static_cast<valueno>(-1)

  Reserved value to indicate "no valueno".

6.1.1 Detailed Description

The Xapian namespace contains public interfaces for the Xapian library.

6.1.2 Typedef Documentation

6.1.2.1 typedef unsigned Xapian::doccount

A count of documents.

This is used to hold values such as the number of documents in a database and the frequency of a term in the database.

6.1.2.2 typedef int Xapian::doccount_diff

A signed difference between two counts of documents.

This is used by the Xapian classes which are STL containers of documents for "difference_type".

6.1.2.3 typedef unsigned Xapian::docid

A unique identifier for a document.

Docid 0 is invalid, providing an "out of range" value which can be used to mean "not a valid document".

6.1.2.4 typedef double Xapian::doclength

A normalised document length.

The normalised document length is the document length divided by the average document length in the database.

6.1.2.5 typedef int Xapian::percent

The percentage score for a document in an MSet.

6.1.2.6 typedef unsigned Xapian::termcount

A counts of terms.

This is used to hold values such as the Within Document Frequency (wdf).

6.1.2.7 typedef int Xapian::termcount_diff

A signed difference between two counts of terms.

This is used by the Xapian classes which are STL containers of terms for "difference_-type".

6.1.2.8 typedef int Xapian::termpos_diff

A signed difference between two term positions.

This is used by the Xapian classes which are STL containers of positions for "difference_type".

6.1.2.9 typedef unsigned Xapian::timeout

A timeout value in milliseconds.

There are 1000 milliseconds in a second, so for example, to set a timeout of 5 seconds use 5000.

6.1.2.10 typedef unsigned Xapian::valueno

The number for a value slot in a document.

Value slot numbers are unsigned and (currently) a 32-bit quantity, with Xapian::BAD_-VALUENO being represented by the largest possible value. Therefore value slots 0 to 0xFFFFFFFE are available for use.

6.1.2.11 typedef int Xapian::valueno_diff

A signed difference between two value slot numbers.

This is used by the Xapian classes which are STL containers of values for "difference_-type".

6.1.2.12 typedef double Xapian::weight

The weight of a document or term.

6.1.3 Function Documentation

6.1.3.1 int Xapian::major_version ()

Report the major version of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN_MAJOR_-VERSION) if shared libraries are being used.

6.1.3.2 int Xapian::minor_version ()

Report the minor version of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN_MINOR_-VERSION) if shared libraries are being used.

6.1.3.3 int Xapian::revision ()

Report the revision of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN_REVISION) if shared libraries are being used.

6.1.3.4 std::string Xapian::sortable_serialise (double *value*)

Convert a floating point number to a string, preserving sort order.

This method converts a floating point number to a string, suitable for using as a value for numeric range restriction, or for use as a sort key.

The conversion is platform independent.

The conversion attempts to ensure that, for any pair of values supplied to the conversion algorithm, the result of comparing the original values (with a numeric comparison operator) will be the same as the result of comparing the resulting values (with a string comparison operator). On platforms which represent doubles with the precisions specified by IEEE_754, this will be the case: if the representation of doubles is more precise, it is possible that two very close doubles will be mapped to the same string, so will compare equal.

Note also that both zero and -zero will be converted to the same representation: since these compare equal, this satisfies the comparison constraint, but it's worth knowing this if you wish to use the encoding in some situation where this distinction matters.

Handling of NaN isn't (currently) guaranteed to be sensible.

Parameters:

value The number to serialise.

6.1.3.5 double Xapian::sortable_unserialise (const std::string & *value*)

Convert a string encoded using *sortable_serialise* back to a floating point number.

This expects the input to be a string produced by *sortable_serialise()*. If the input is not such a string, the value returned is undefined (but no error will be thrown).

The result of the conversion will be exactly the value which was supplied to *sortable_serialise()* when making the string on platforms which represent doubles with the precisions specified by IEEE_754, but may be a different (nearby) value on other platforms.

Parameters:

value The serialised string to decode.

6.1.3.6 const char* Xapian::version_string ()

Report the version string of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN_VERSION) if shared libraries are being used.

6.1.4 Variable Documentation

6.1.4.1 const valueno Xapian::BAD_VALUENO = static_cast<valueno>(-1)

Reserved value to indicate "no valueno".

6.1.4.2 const int Xapian::DB_CREATE = 2

Create a new database; fail if db exists.

6.1.4.3 const int Xapian::DB_CREATE_OR_OPEN = 1

Open for read/write; create if no db exists.

6.1.4.4 const int Xapian::DB_CREATE_OR_OVERWRITE = 3

Overwrite existing db; create if none exists.

6.1.4.5 const int Xapian::DB_OPEN = 4

Open for read/write; fail if no db exists.

6.2 Xapian::Auto Namespace Reference

Database factory functions which determine the database type automatically.

Functions

- Database open_stub (const std::string &file)

  Construct a Database object for a stub database file.

- WritableDatabase open_stub (const std::string &file, int action)

  Construct a WritableDatabase object for a stub database file.

6.2.1 Detailed Description

Database factory functions which determine the database type automatically.

6.2.2 Function Documentation

6.2.2.1 WritableDatabase Xapian::Auto::open_stub (const std::string & *file*, int *action*)

Construct a WritableDatabase object for a stub database file.

The stub database file must contain serialised parameters for exactly one database.

Parameters:

file pathname of the stub database file.

action determines handling of existing/non-existing database:

- Xapian::DB_CREATE fail if database already exist, otherwise create new database.
- Xapian::DB_CREATE_OR_OPEN open existing database, or create new database if none exists.
- Xapian::DB_CREATE_OR_OVERWRITE overwrite existing database, or create new database if none exists.
- Xapian::DB_OPEN open existing database, failing if none exists.

6.2.2.2 Database Xapian::Auto::open_stub (const std::string & *file*)

Construct a Database object for a stub database file.

The stub database file contains serialised parameters for one or more databases.

Parameters:

file pathname of the stub database file.

6.3 Xapian::Brass Namespace Reference

Database factory functions for the brass backend.

Functions

- Database open (const std::string &dir)

  Construct a Database object for read-only access to a Brass database.

- WritableDatabase open (const std::string &dir, int action, int block_size=8192)

  Construct a Database object for update access to a Brass database.

6.3.1 Detailed Description

Database factory functions for the brass backend.

6.3.2 Function Documentation

6.3.2.1 WritableDatabase Xapian::Brass::open (const std::string & *dir*, int *action*, int *block_size* = `8192`)

Construct a Database object for update access to a Brass database.

Parameters:

dir pathname of the directory containing the database.

action determines handling of existing/non-existing database:

- Xapian::DB_CREATE fail if database already exist, otherwise create new database.
- Xapian::DB_CREATE_OR_OPEN open existing database, or create new database if none exists.
- Xapian::DB_CREATE_OR_OVERWRITE overwrite existing database, or create new database if none exists.
- Xapian::DB_OPEN open existing database, failing if none exists.

block_size the Btree blocksize to use (in bytes), which must be a power of two between 2048 and 65536 (inclusive). The default (also used if an invalid value if passed) is 8192 bytes. This parameter is ignored when opening an existing database.

6.3.2.2 Database Xapian::Brass::open (const std::string & *dir*)

Construct a Database object for read-only access to a Brass database.

Parameters:

dir pathname of the directory containing the database.

6.4 Xapian::Chert Namespace Reference

Database factory functions for the chert backend.

Functions

- Database open (const std::string &dir)

  Construct a Database object for read-only access to a Chert database.

- WritableDatabase open (const std::string &dir, int action, int block_size=8192)

  Construct a Database object for update access to a Chert database.

6.4.1 Detailed Description

Database factory functions for the chert backend.

6.4.2 Function Documentation

6.4.2.1 WritableDatabase Xapian::Chert::open (const std::string & *dir*, int *action*, int *block_size* = `8192`)

Construct a Database object for update access to a Chert database.

Parameters:

dir pathname of the directory containing the database.

action determines handling of existing/non-existing database:

- Xapian::DB_CREATE fail if database already exist, otherwise create new database.
- Xapian::DB_CREATE_OR_OPEN open existing database, or create new database if none exists.
- Xapian::DB_CREATE_OR_OVERWRITE overwrite existing database, or create new database if none exists.
- Xapian::DB_OPEN open existing database, failing if none exists.

block_size the Btree blocksize to use (in bytes), which must be a power of two between 2048 and 65536 (inclusive). The default (also used if an invalid value if passed) is 8192 bytes. This parameter is ignored when opening an existing database.

6.4.2.2 Database Xapian::Chert::open (const std::string & *dir*)

Construct a Database object for read-only access to a Chert database.

Parameters:

dir pathname of the directory containing the database.

6.5 Xapian::Flint Namespace Reference

Database factory functions for the flint backend.

Functions

- Database open (const std::string &dir)

  Construct a Database object for read-only access to a Flint database.

- WritableDatabase open (const std::string &dir, int action, int block_size=8192)

  Construct a Database object for update access to a Flint database.

6.5.1 Detailed Description

Database factory functions for the flint backend.

6.5.2 Function Documentation

6.5.2.1 WritableDatabase Xapian::Flint::open (const std::string & *dir*, int *action*, int *block_size* = `8192`)

Construct a Database object for update access to a Flint database.

Parameters:

dir pathname of the directory containing the database.

action determines handling of existing/non-existing database:

- Xapian::DB_CREATE fail if database already exist, otherwise create new database.
- Xapian::DB_CREATE_OR_OPEN open existing database, or create new database if none exists.
- Xapian::DB_CREATE_OR_OVERWRITE overwrite existing database, or create new database if none exists.
- Xapian::DB_OPEN open existing database, failing if none exists.

block_size the Btree blocksize to use (in bytes), which must be a power of two between 2048 and 65536 (inclusive). The default (also used if an invalid value if passed) is 8192 bytes. This parameter is ignored when opening an existing database.

6.5.2.2 Database Xapian::Flint::open (const std::string & *dir*)

Construct a Database object for read-only access to a Flint database.

Parameters:

dir pathname of the directory containing the database.

6.6 Xapian::InMemory Namespace Reference

Database factory functions for the inmemory backend.

Functions

- WritableDatabase open ()

  Construct a WritableDatabase object for a new, empty InMemory database.

6.6.1 Detailed Description

Database factory functions for the inmemory backend.

6.6.2 Function Documentation

6.6.2.1 WritableDatabase Xapian::InMemory::open ()

Construct a WritableDatabase object for a new, empty InMemory database.

Only a writable InMemory database can be created, since a read-only one would always remain empty.

6.7 Xapian::Remote Namespace Reference

Database factory functions for the remote backend.

Functions

- Database open (const std::string &host, unsigned int port, Xapian::timeout timeout=10000, Xapian::timeout connect_timeout=10000)

  Construct a Database object for read-only access to a remote database accessed via a TCP connection.

- WritableDatabase open_writable (const std::string &host, unsigned int port, Xapian::timeout timeout=0, Xapian::timeout connect_timeout=10000)

  Construct a WritableDatabase object for update access to a remote database accessed via a TCP connection.

- Database open (const std::string &program, const std::string &args, Xapian::timeout timeout=10000)

  Construct a Database object for read-only access to a remote database accessed via a program.

- WritableDatabase open_writable (const std::string &program, const std::string &args, Xapian::timeout timeout=0)

  Construct a WritableDatabase object for update access to a remote database accessed via a program.

6.7.1 Detailed Description

Database factory functions for the remote backend.

6.7.2 Function Documentation

6.7.2.1 Database Xapian::Remote::open (const std::string & *program*, const std::string & *args*, Xapian::timeout *timeout* = `10000`)

Construct a Database object for read-only access to a remote database accessed via a program.

Access to the remote database is done by running an external program and communicating with it on stdin/stdout.

Parameters:

program the external program to run.

args space-separated list of arguments to pass to program.

timeout timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then Xapian::NetworkTimeoutError is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).

6.7.2.2 Database Xapian::Remote::open (const std::string & *host*, unsigned int *port*, Xapian::timeout *timeout* = `10000`, Xapian::timeout *connect_timeout* = `10000`)

Construct a Database object for read-only access to a remote database accessed via a TCP connection.

Access to the remote database is via a TCP connection to the specified host and port.

Parameters:

host hostname to connect to.

port port number to connect to.

timeout timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then Xapian::NetworkTimeoutError is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).

connect_timeout timeout to use when connecting to the server. If this timeout is exceeded then Xapian::NetworkTimeoutError is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).

6.7.2.3 WritableDatabase Xapian::Remote::open_writable (const std::string & *program*, const std::string & *args*, Xapian::timeout *timeout* = `0`)

Construct a WritableDatabase object for update access to a remote database accessed via a program.

Access to the remote database is done by running an external program and communicating with it on stdin/stdout.

Parameters:

program the external program to run.

args space-separated list of arguments to pass to program.

timeout timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then Xapian::NetworkTimeoutError is thrown. (Default is 0, which means don't timeout).

6.7.2.4 WritableDatabase Xapian::Remote::open_writable (const std::string & *host*, unsigned int *port*, Xapian::timeout *timeout* = `0`, Xapian::timeout *connect_timeout* = `10000`)

Construct a WritableDatabase object for update access to a remote database accessed via a TCP connection.

Access to the remote database is via a TCP connection to the specified host and port.

Parameters:

host hostname to connect to.

port port number to connect to.

timeout timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then Xapian::NetworkTimeoutError is thrown. (Default is 0, which means don't timeout).

connect_timeout timeout to use when connecting to the server. If this timeout is exceeded then Xapian::NetworkTimeoutError is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).

6.8 Xapian::Unicode Namespace Reference

Functions associated with handling Unicode characters.

Enumerations

- enum category

  Each Unicode character is in exactly one of these categories.

Functions

- unsigned nonascii_to_utf8 (unsigned ch, char *buf)

  Convert a single non-ASCII Unicode character to UTF-8.

- unsigned to_utf8 (unsigned ch, char *buf)

  Convert a single Unicode character to UTF-8.

- void append_utf8 (std::string &s, unsigned ch)

  Append the UTF-8 representation of a single Unicode character to a std::string.

- category get_category (unsigned ch)

  Return the category which a given Unicode character falls into.

- bool is_wordchar (unsigned ch)

  Test if a given Unicode character is "word character".

- bool is_whitespace (unsigned ch)

  Test if a given Unicode character is a whitespace character.

- bool is_currency (unsigned ch)

  Test if a given Unicode character is a currency symbol.

- unsigned tolower (unsigned ch)

  Convert a Unicode character to lowercase.

- unsigned toupper (unsigned ch)

  Convert a Unicode character to uppercase.

- std::string tolower (const std::string &term)

  Convert a UTF-8 std::string to lowercase.

- std::string toupper (const std::string &term)

  Convert a UTF-8 std::string to uppercase.

6.8.1 Detailed Description

Functions associated with handling Unicode characters.

6.8.2 Enumeration Type Documentation

6.8.2.1 enum Xapian::Unicode::category

Each Unicode character is in exactly one of these categories.

6.8.3 Function Documentation

6.8.3.1 unsigned Xapian::Unicode::nonascii_to_utf8 (unsigned *ch*, char * *buf*)

Convert a single non-ASCII Unicode character to UTF-8.

This is intended mainly as a helper method for to_utf8().

Parameters:

ch The character (which must be > 128) to write to *buf*.

buf The buffer to write the character to - it must have space for (at least) 4 bytes.

Returns:

The length of the resultant UTF-8 character in bytes.

Referenced by to_utf8().

6.8.3.2 unsigned Xapian::Unicode::to_utf8 (unsigned *ch*, char * *buf*) [inline]

Convert a single Unicode character to UTF-8.

Parameters:

ch The character to write to *buf*.

buf The buffer to write the character to - it must have space for (at least) 4 bytes.

Returns:

The length of the resultant UTF-8 character in bytes.

References nonascii_to_utf8().

Referenced by append_utf8().

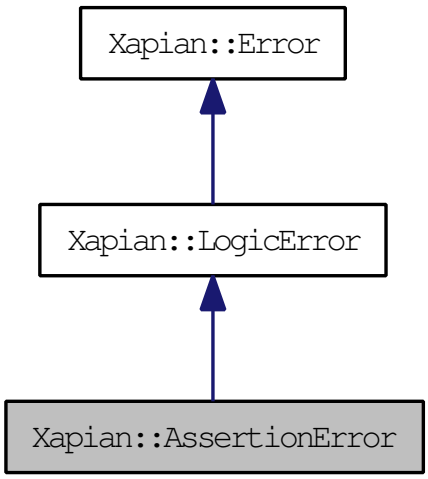
Chapter 7

Class Documentation

7.1 Xapian::AssertionError Class Reference

AssertionError is thrown if a logical assertion inside Xapian fails.

Inheritance diagram for Xapian::AssertionError:



Public Member Functions

- AssertionError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- AssertionError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.1.1 Detailed Description

AssertionError is thrown if a logical assertion inside Xapian fails.

In a debug build of Xapian, a failed assertion in the core library code will cause AssertionError to be thrown.

This represents a bug in Xapian (either an invariant, precondition, etc has been violated, or the assertion is incorrect!)

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Xapian::AssertionError::AssertionError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.1.2.2 Xapian::AssertionError::AssertionError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

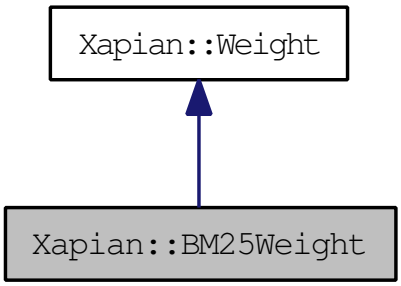
The documentation for this class was generated from the following file:

- xapian/error.h

7.2 Xapian::BM25Weight Class Reference

Xapian::Weight subclass implementing the BM25 probabilistic formula.

Inheritance diagram for Xapian::BM25Weight:



Public Member Functions

- BM25Weight (double k1, double k2, double k3, double b, double min_-normlen)

  Construct a BM25Weight.

- std::string name () const

  Return the name of this weighting scheme.

- std::string serialise () const

  Return this object's parameters serialised as a single string.

- BM25Weight * unserialise (const std::string &s) const

  Unserialise parameters.

- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::termcount do-clen) const

  Calculate the weight contribution for this object's term to a document.

- Xapian::weight get_maxpart () const

  Return an upper bound on what get_sumpart() can return for any document.

- Xapian::weight get_sumextra (Xapian::termcount doclen) const

  Calculate the term-independent weight component for a document.

- Xapian::weight get_maxextra () const

  Return an upper bound on what get_sumextra() can return for any document.

7.2.1 Detailed Description

Xapian::Weight subclass implementing the BM25 probabilistic formula.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Xapian::BM25Weight::BM25Weight (double *k1*, double *k2*, double *k3*, double *b*, double *min_normlen*) [inline]

Construct a BM25Weight.

Parameters:

k1 A non-negative parameter controlling how influential within-document-frequency (wdf) is. k1=0 means that wdf doesn't affect the weights. The larger k1 is, the more wdf influences the weights. (default 1)

k2 A non-negative parameter which controls the strength of a correction factor which depends upon query length and normalised document length. k2=0 disable this factor; larger k2 makes it stronger. (default 0)

k3 A non-negative parameter controlling how influential within-query-frequency (wqf) is. k3=0 means that wqf doesn't affect the weights. The larger k3 is, the more wqf influences the weights. (default 1)

b A parameter between 0 and 1, controlling how strong the document length normalisation of wdf is. 0 means no normalisation; 1 means full normalisation. (default 0.5)

min_normlen A parameter specifying a minimum value for normalised document length. Normalised document length values less than this will be clamped to this value, helping to prevent very short documents getting large weights. (default 0.5)

7.2.3 Member Function Documentation

7.2.3.1 Xapian::weight Xapian::BM25Weight::get_maxextra () const [virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.2.3.2 Xapian::weight Xapian::BM25Weight::get_maxpart () const [virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.2.3.3 Xapian::weight Xapian::BM25Weight::get_sumextra (Xapian::termcount *doclen*) const [virtual]

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters:

doclen The document's length (unnormalised).

Implements Xapian::Weight.

7.2.3.4 Xapian::weight Xapian::BM25Weight::get_sumpart (Xapian::termcount *wdf*, Xapian::termcount *doclen*) const [virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters:

wdf The within document frequency of the term in the document.

doclen The document's length (unnormalised).

Implements Xapian::Weight.

7.2.3.5 std::string Xapian::BM25Weight::name () const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend, you can use the default implementation which simply returns an empty string.

Reimplemented from Xapian::Weight.

7.2.3.6 std::string Xapian::BM25Weight::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::Weight.

7.2.3.7 BM25Weight* Xapian::BM25Weight::unserialise (const std::string & *s*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Parameters:

s A string containing the serialised parameters.

Reimplemented from Xapian::Weight.

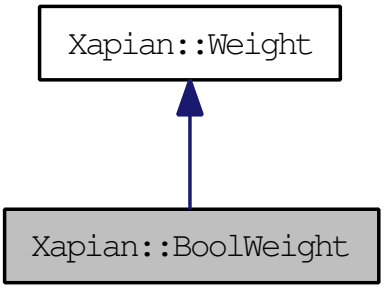
The documentation for this class was generated from the following file:

- xapian/weight.h

7.3 Xapian::BoolWeight Class Reference

Class implementing a "boolean" weighting scheme.

Inheritance diagram for Xapian::BoolWeight:



Public Member Functions

- BoolWeight ()

  Construct a BoolWeight.

- std::string name () const

  Return the name of this weighting scheme.

- std::string serialise () const

  Return this object's parameters serialised as a single string.

- BoolWeight * unserialise (const std::string &s) const

  Unserialise parameters.

- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const

  Calculate the weight contribution for this object's term to a document.

- Xapian::weight get_maxpart () const

  Return an upper bound on what get_sumpart() can return for any document.

- Xapian::weight get_sumextra (Xapian::termcount doclen) const

  Calculate the term-independent weight component for a document.

- Xapian::weight get_maxextra () const

  Return an upper bound on what get_sumextra() can return for any document.

7.3.1 Detailed Description

Class implementing a "boolean" weighting scheme.

This weighting scheme gives all documents zero weight.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Xapian::BoolWeight::BoolWeight () [inline]

Construct a BoolWeight.

7.3.3 Member Function Documentation

7.3.3.1 Xapian::weight Xapian::BoolWeight::get_maxextra () const [virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.3.3.2 Xapian::weight Xapian::BoolWeight::get_maxpart () const [virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.3.3.3 Xapian::weight Xapian::BoolWeight::get_sumextra (Xapian::termcount *doclen*) const [virtual]

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters:

doclen The document's length (unnormalised).

Implements Xapian::Weight.

7.3.3.4 Xapian::weight Xapian::BoolWeight::get_sumpart (Xapian::termcount *wdf*, Xapian::termcount *doclen*) const [virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters:

wdf The within document frequency of the term in the document.

doclen The document's length (unnormalised).

Implements Xapian::Weight.

7.3.3.5 std::string Xapian::BoolWeight::name () const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend, you can use the default implementation which simply returns an empty string.

Reimplemented from Xapian::Weight.

7.3.3.6 std::string Xapian::BoolWeight::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::Weight.

7.3.3.7 BoolWeight* Xapian::BoolWeight::unserialise (const std::string & *s*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Parameters:

s A string containing the serialised parameters.

Reimplemented from Xapian::Weight.

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

- xapian/weight.h

7.4 Xapian::Compactor Class Reference

Compact a database, or merge and compact several.

Public Member Functions

- void set_block_size (size_t block_size)

  Set the block size to use for tables in the output database.

- void set_renumber (bool renumber)

  Set whether to preserve existing document id values.

- void set_multipass (bool multipass)

  Set whether to merge postlists in multiple passes.

- void set_compaction_level (compaction_level compaction)

  Set the compaction level.

- void set_destdir (const std::string &destdir)

  Set where to write the output.

- void add_source (const std::string &srcdir)

  Add a source database.

- void compact ()

  Perform the actual compaction/merging operation.

- virtual void set_status (const std::string &table, const std::string &status)

  Update progress.

- virtual std::string resolve_duplicate_metadata (const std::string &key, size_t num_tags, const std::string tags[ ])

  Resolve multiple user metadata entries with the same key.

7.4.1 Detailed Description

Compact a database, or merge and compact several.

7.4.2 Member Function Documentation

7.4.2.1 void Xapian::Compactor::add_source (const std::string & *srcdir*)

Add a source database.

Parameters:

srcdir The path to the source database to add.

7.4.2.2 virtual std::string Xapian::Compactor::resolve_duplicate_metadata (const std::string & *key*, size_t *num_tags*, const std::string *tags*[ ]) [virtual]

Resolve multiple user metadata entries with the same key.

When merging, if the same user metadata key is set in more than one input, then this method is called to allow this to be resolving in an appropriate way.

The default implementation just returns tags[0].

For multipass this will currently get called multiple times for the same key if there are duplicates to resolve in each pass, but this may change in the future.

Parameters:

key The metadata key with duplicate entries.

num_tags How many tags there are.

tags An array of num_tags strings containing the tags to merge.

7.4.2.3 void Xapian::Compactor::set_block_size (size_t *block_size*)

Set the block size to use for tables in the output database.

Parameters:

block_size The block size to use. Valid block sizes are currently powers of two between 2048 and 65536, with the default being 8192, but the valid sizes and default may change in the future.

7.4.2.4 void Xapian::Compactor::set_compaction_level (compaction_level *compaction*)

Set the compaction level.

Parameters:

compaction Available values are: - Xapian::Compactor::STANDARD - Don't split items unnecessarily. - Xapian::Compactor::FULL - Split items whenever it saves space (the default). - Xapian::Compactor::FULLER - Allow oversize items to save more space (not recommended if you ever plan to update the compacted database).

7.4.2.5 void Xapian::Compactor::set_destdir (const std::string & *destdir*)

Set where to write the output.

Parameters:

destdir Output path. This can be the same as an input if that input is a stub database (in which case the database(s) listed in the stub will be compacted to a new database and then the stub will be atomically updated to point to this new database).

7.4.2.6 void Xapian::Compactor::set_multipass (bool *multipass*)

Set whether to merge postlists in multiple passes.

Parameters:

multipass If true and merging more than 3 databases, merge the postlists in multiple passes, which is generally faster but requires more disk space for temporary files. By default we don't do this.

7.4.2.7 void Xapian::Compactor::set_renumber (bool *renumber*)

Set whether to preserve existing document id values.

Parameters:

renumber The default is true, which means that document ids will be renumbered - currently by applying the same offset to all the document ids in a particular source database.

If false, then the document ids must be unique over all source databases. Currently the ranges of document ids in each source must not overlap either, though this restriction may be removed in the future.

7.4.2.8 virtual void Xapian::Compactor::set_status (const std::string & *table*, const std::string & *status*) `[virtual]`

Update progress.

Subclass this method if you want to get progress updates during compaction. This is called for each table first with empty status, And then one or more times with non-empty status.

The default implementation does nothing.

Parameters:

table The table currently being compacted.

status A status message.

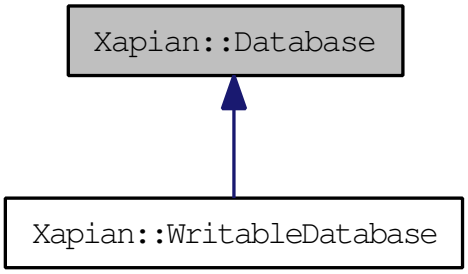
The documentation for this class was generated from the following file:

- xapian/compactor.h

7.5 Xapian::Database Class Reference

This class is used to access a database, or a group of databases.

Inheritance diagram for Xapian::Database:



Public Member Functions

- void add_database (const Database &database)

  Add an existing database (or group of databases) to those accessed by this object.

- Database ()

  Create a Database with no databases in.

- Database (const std::string &path)

  Open a Database, automatically determining the database backend to use.

- virtual ~Database ()

  Destroy this handle on the database.

- Database (const Database &other)

  Copying is allowed.

- void operator= (const Database &other)

  Assignment is allowed.

- void reopen ()

  Re-open the database.

- virtual void close ()

  Close the database.

- virtual std::string get_description () const

  Return a string describing this object.

- PostingIterator postlist_begin (const std::string &tname) const

An iterator pointing to the start of the postlist for a given term.

- PostingIterator postlist_end (const std::string &) const

  Corresponding end iterator to postlist_begin().

- TermIterator termlist_begin (Xapian::docid did) const

  An iterator pointing to the start of the termlist for a given document.

- TermIterator termlist_end (Xapian::docid) const

  Corresponding end iterator to termlist_begin().

- bool has_positions () const

  Does this database have any positional information?

- PositionIterator positionlist_begin (Xapian::docid did, const std::string &tname) const

  An iterator pointing to the start of the position list for a given term in a given document.

- PositionIterator positionlist_end (Xapian::docid, const std::string &) const

  Corresponding end iterator to positionlist_begin().

- TermIterator allterms_begin () const

  An iterator which runs across all terms in the database.

- TermIterator allterms_end () const

  Corresponding end iterator to allterms_begin().

- TermIterator allterms_begin (const std::string &prefix) const

  An iterator which runs across all terms with a given prefix.

- TermIterator allterms_end (const std::string &) const

  Corresponding end iterator to allterms_begin(prefix).

- Xapian::doccount get_doccount () const

  Get the number of documents in the database.

- Xapian::docid get_lastdocid () const

  Get the highest document id which has been used in the database.

- Xapian::doclength get_avlength () const

  Get the average length of the documents in the database.

- Xapian::doccount get_termfreq (const std::string &tname) const

  Get the number of documents in the database indexed by a given term.

- bool term_exists (const std::string &tname) const

  Check if a given term exists in the database.

- Xapian::termcount get_collection_freq (const std::string &tname) const

  Return the total number of occurrences of the given term.

- Xapian::doccount get_value_freq (Xapian::valueno slot) const

  Return the frequency of a given value slot.

- std::string get_value_lower_bound (Xapian::valueno slot) const

  Get a lower bound on the values stored in the given value slot.

- std::string get_value_upper_bound (Xapian::valueno slot) const

  Get an upper bound on the values stored in the given value slot.

- Xapian::termcount get_doclength_lower_bound () const

  Get a lower bound on the length of a document in this DB.

- Xapian::termcount get_doclength_upper_bound () const

  Get an upper bound on the length of a document in this DB.

- Xapian::termcount get_wdf_upper_bound (const std::string &term) const

  Get an upper bound on the wdf of term term.

- ValueIterator valuestream_begin (Xapian::valueno slot) const

  Return an iterator over the value in slot slot *for each document.*

- ValueIteratorEnd_ valuestream_end (Xapian::valueno) const

  Return end iterator corresponding to valuestream_begin().

- Xapian::termcount get_doclength (Xapian::docid did) const

  Get the length of a document.

- void keep_alive ()

  Send a "keep-alive" to remote databases to stop them timing out.

- Xapian::Document get_document (Xapian::docid did) const

  Get a document from the database, given its document id.

- std::string get_spelling_suggestion (const std::string &word, unsigned max_-edit_distance=2) const

  Suggest a spelling correction.

- Xapian::TermIterator spellings_begin () const

  An iterator which returns all the spelling correction targets.

- Xapian::TermIterator spellings_end () const

  Corresponding end iterator to spellings_begin().

- Xapian::TermIterator synonyms_begin (const std::string &term) const

  An iterator which returns all the synonyms for a given term.

- Xapian::TermIterator synonyms_end (const std::string &) const

  Corresponding end iterator to synonyms_begin(term).

- Xapian::TermIterator synonym_keys_begin (const std::string &prefix=std::string()) const

  An iterator which returns all terms which have synonyms.

- Xapian::TermIterator synonym_keys_end (const std::string &=std::string()) const

  Corresponding end iterator to synonym_keys_begin(prefix).

- std::string get_metadata (const std::string &key) const

  Get the user-specified metadata associated with a given key.

- Xapian::TermIterator metadata_keys_begin (const std::string &prefix=std::string()) const

  An iterator which returns all user-specified metadata keys.

- Xapian::TermIterator metadata_keys_end (const std::string &=std::string()) const

  Corresponding end iterator to metadata_keys_begin().

- std::string get_uuid () const

  Get a UUID for the database.

7.5.1 Detailed Description

This class is used to access a database, or a group of databases.

For searching, this class is used in conjunction with an Enquire object.

Exceptions:

InvalidArgumentError will be thrown if an invalid argument is supplied, for example, an unknown database type.

DatabaseOpeningError may be thrown if the database cannot be opened (for example, a required file cannot be found).

DatabaseVersionError may be thrown if the database is in an unsupported format (for example, created by a newer version of Xapian which uses an incompatible format).

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Xapian::Database::Database (const std::string & *path*) [explicit]

Open a Database, automatically determining the database backend to use.

Parameters:

path directory that the database is stored in.

7.5.2.2 virtual Xapian::Database::~Database () [virtual]

Destroy this handle on the database.

If there are no copies of this object remaining, the database(s) will be closed.

7.5.2.3 Xapian::Database::Database (const Database & *other*)

Copying is allowed.

The internals are reference counted, so copying is cheap.

Parameters:

other The object to copy.

7.5.3 Member Function Documentation

7.5.3.1 void Xapian::Database::add_database (const Database & *database*)

Add an existing database (or group of databases) to those accessed by this object.

Parameters:

database the database(s) to add.

7.5.3.2 TermIterator Xapian::Database::allterms_begin (const std::string & *prefix*) const

An iterator which runs across all terms with a given prefix.

This is functionally similar to getting an iterator with allterms_begin() and then calling skip_to(prefix) on that iterator to move to the start of the prefix, but is more convenient (because it detects the end of the prefixed terms), and may be more efficient than simply calling skip_to() after opening the iterator, particularly for remote databases.

Parameters:

prefix The prefix to restrict the returned terms to.

7.5.3.3 virtual void Xapian::Database::close () [virtual]

Close the database.

This closes the database and releases all file handles held by the database.

This cannot be undone - in particular, calling reopen() after closing a database will not reopen it, but will instead throw a Xapian::DatabaseError exception.

Calling close() again on a database which has already been closed has no effect (and doesn't raise an exception).

After close() has been called, calls to other methods of the database, and to methods of other objects associated with the database, will either:

- behave exactly as they would have done if the database had not been closed (this can only happen if all the required data is cached)
- raise a Xapian::DatabaseError exception indicating that the database is closed.

The reason for this behaviour is that otherwise we'd have to check that the database is still open on every method call on every object associated with a Database, when in many cases they are working on data which has already been loaded and so they are able to just behave correctly.

7.5.3.4 Xapian::termcount Xapian::Database::get_collection_freq (const std::string & *tname*) const

Return the total number of occurrences of the given term.

This is the sum of the number of occurrences of the term in each document it indexes: i.e., the sum of the within document frequencies of the term.

Parameters:

tname The term whose collection frequency is being requested.

7.5.3.5 Xapian::termcount Xapian::Database::get_doclength_lower_bound () const

Get a lower bound on the length of a document in this DB.

This bound does not include any zero-length documents.

7.5.3.6 Xapian::Document Xapian::Database::get_document (Xapian::docid *did*) const

Get a document from the database, given its document id.

This method returns a Xapian::Document object which provides the information about a document.

Parameters:

did The document id of the document to retrieve.

Returns:

A Xapian::Document object containing the document data

Exceptions:

Xapian::DocNotFoundError The document specified could not be found in the database.

Xapian::InvalidArgumentError did was 0, which is not a valid document id.

7.5.3.7 std::string Xapian::Database::get_metadata (const std::string & *key*) const

Get the user-specified metadata associated with a given key.

User-specified metadata allows you to store arbitrary information in the form of (key,tag) pairs. See *WritableDatabase::set_metadata()* for more information.

When invoked on a Xapian::Database object representing multiple databases, currently only the metadata for the first is considered but this behaviour may change in the future.

If there is no piece of metadata associated with the specified key, an empty string is returned (this applies even for backends which don't support metadata).

Empty keys are not valid, and specifying one will cause an exception.

Parameters:

key The key of the metadata item to access.

Returns:

The retrieved metadata item's value.

Exceptions:

Xapian::InvalidArgumentError will be thrown if the key supplied is empty.

7.5.3.8 std::string Xapian::Database::get_spelling_suggestion (const std::string & *word*, unsigned *max_edit_distance* = 2) const

Suggest a spelling correction.

Parameters:

word The potentially misspelled word.

max_edit_distance Only consider words which are at most *max_edit_distance* edits from *word*. An edit is a character insertion, deletion, or the transposition of two adjacent characters (default is 2).

7.5.3.9 std::string Xapian::Database::get_uuid () const

Get a UUID for the database.

The UUID will persist for the lifetime of the database.

Replicas (eg, made with the replication protocol, or by copying all the database files) will have the same UUID. However, copies (made with copydatabase, or xapian-compact) will have different UUIDs.

If the backend does not support UUIDs or this database has no subdatabases, the UUID will be empty.

If this database has multiple sub-databases, the UUID string will contain the UUIDs of all the sub-databases.

7.5.3.10 Xapian::doccount Xapian::Database::get_value_freq (Xapian::valueno *slot*) const

Return the frequency of a given value slot.

This is the number of documents which have a (non-empty) value stored in the slot.

Parameters:

slot The value slot to examine.

Exceptions:

UnimplementedError The frequency of the value isn't available for this database type.

7.5.3.11 std::string Xapian::Database::get_value_lower_bound (Xapian::valueno *slot*) const

Get a lower bound on the values stored in the given value slot.

If there are no values stored in the given value slot, this will return an empty string.

If the lower bound isn't available for the given database type, this will return the lowest possible bound - the empty string.

Parameters:

slot The value slot to examine.

7.5.3.12 std::string Xapian::Database::get_value_upper_bound (Xapian::valueno *slot*) const

Get an upper bound on the values stored in the given value slot.

If there are no values stored in the given value slot, this will return an empty string.

Parameters:

slot The value slot to examine.

Exceptions:

UnimplementedError The upper bound of the values isn't available for this database type.

7.5.3.13 void Xapian::Database::keep_alive ()

Send a "keep-alive" to remote databases to stop them timing out.

Has no effect on non-remote databases.

7.5.3.14 Xapian::TermIterator Xapian::Database::metadata_keys_begin (const std::string & *prefix* = `std::string()`) const

An iterator which returns all user-specified metadata keys.

When invoked on a Xapian::Database object representing multiple databases, currently only the metadata for the first is considered but this behaviour may change in the future.

If the backend doesn't support metadata, then this method returns an iterator which compares equal to that returned by metadata_keys_end().

Parameters:

prefix If non-empty, only keys with this prefix are returned.

Exceptions:

Xapian::UnimplementedError will be thrown if the backend implements user-specified metadata, but doesn't implement iterating its keys (currently this happens for the InMemory backend).

7.5.3.15 void Xapian::Database::operator= (const Database & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

Parameters:

other The object to copy.

7.5.3.16 PostingIterator Xapian::Database::postlist_begin (const std::string & *tname*) const

An iterator pointing to the start of the postlist for a given term.

Parameters:

tname The termname to iterate postings for. If the term name is the empty string, the iterator returned will list all the documents in the database. Such an iterator will always return a WDF value of 1, since there is no obvious meaning for this quantity in this case.

7.5.3.17 void Xapian::Database::reopen ()

Re-open the database.

This re-opens the database(s) to the latest available version(s). It can be used either to make sure the latest results are returned, or to recover from a Xapian::DatabaseModifiedError.

Calling reopen() on a database which has been closed (with *close()*) will always raise a Xapian::DatabaseError.

7.5.3.18 Xapian::TermIterator Xapian::Database::spellings_begin () const

An iterator which returns all the spelling correction targets.

This returns all the words which are considered as targets for the spelling correction algorithm. The frequency of each word is available as the term frequency of each entry in the returned iterator.

7.5.3.19 Xapian::TermIterator Xapian::Database::synonym_keys_begin (const std::string & *prefix* = `std::string()`) const

An iterator which returns all terms which have synonyms.

Parameters:

prefix If non-empty, only terms with this prefix are returned.

7.5.3.20 Xapian::TermIterator Xapian::Database::synonyms_begin (const std::string & *term*) const

An iterator which returns all the synonyms for a given term.

Parameters:

term The term to return synonyms for.

7.5.3.21 bool Xapian::Database::term_exists (const std::string & *tname*) const

Check if a given term exists in the database.

Parameters:

tname The term to test the existence of.

Returns:

true if and only if the term exists in the database. This is the same as (get_-termfreq(tname) != 0), but will often be more efficient.

7.5.3.22 TermIterator Xapian::Database::termlist_begin (Xapian::docid *did*) const

An iterator pointing to the start of the termlist for a given document.

Parameters:

did The document id of the document to iterate terms for.

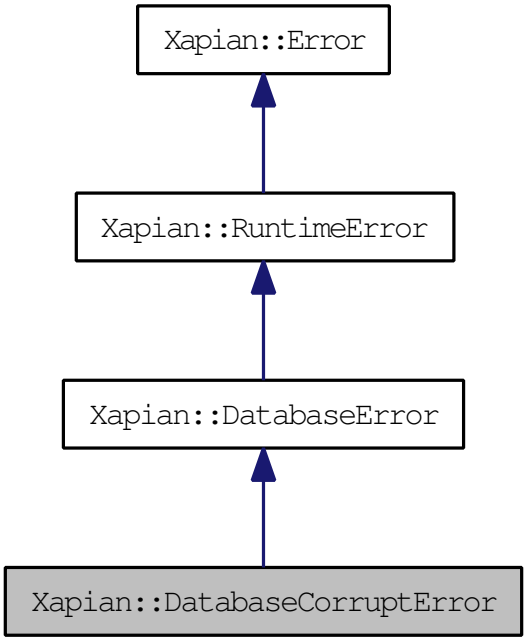
The documentation for this class was generated from the following file:

- xapian/database.h

7.6 Xapian::DatabaseCorruptError Class Reference

DatabaseCorruptError indicates database corruption was detected.

Inheritance diagram for Xapian::DatabaseCorruptError:



Public Member Functions

- DatabaseCorruptError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- DatabaseCorruptError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.6.1 Detailed Description

DatabaseCorruptError indicates database corruption was detected.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 Xapian::DatabaseCorruptError::DatabaseCorruptError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.6.2.2 Xapian::DatabaseCorruptError::DatabaseCorruptError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

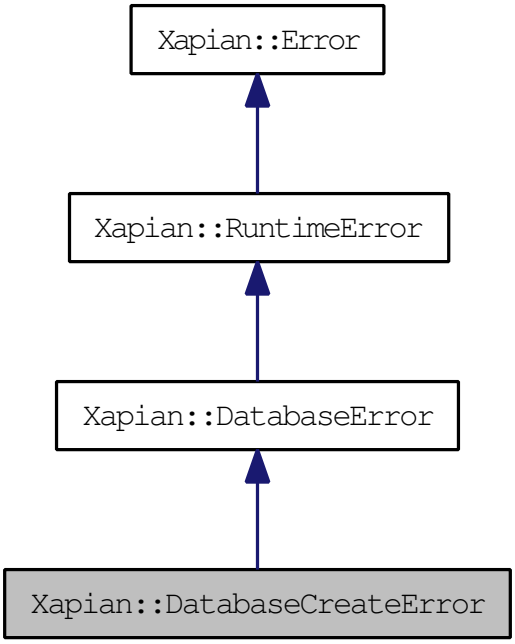
The documentation for this class was generated from the following file:

- xapian/error.h

7.7 Xapian::DatabaseCreateError Class Reference

DatabaseCreateError indicates a failure to create a database.

Inheritance diagram for Xapian::DatabaseCreateError:



Public Member Functions

- DatabaseCreateError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- DatabaseCreateError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.7.1 Detailed Description

DatabaseCreateError indicates a failure to create a database.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 Xapian::DatabaseCreateError::DatabaseCreateError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.7.2.2 Xapian::DatabaseCreateError::DatabaseCreateError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

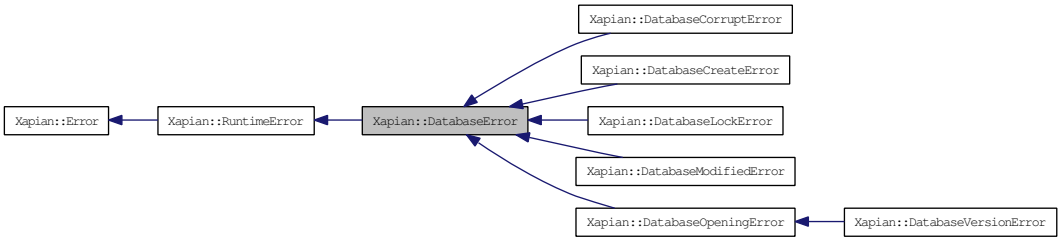
The documentation for this class was generated from the following file:

- xapian/error.h

7.8 Xapian::DatabaseError Class Reference

DatabaseError indicates some sort of database related error.

Inheritance diagram for Xapian::DatabaseError:



Public Member Functions

- DatabaseError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- DatabaseError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.8.1 Detailed Description

DatabaseError indicates some sort of database related error.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Xapian::DatabaseError::DatabaseError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.8.2.2 Xapian::DatabaseError::DatabaseError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

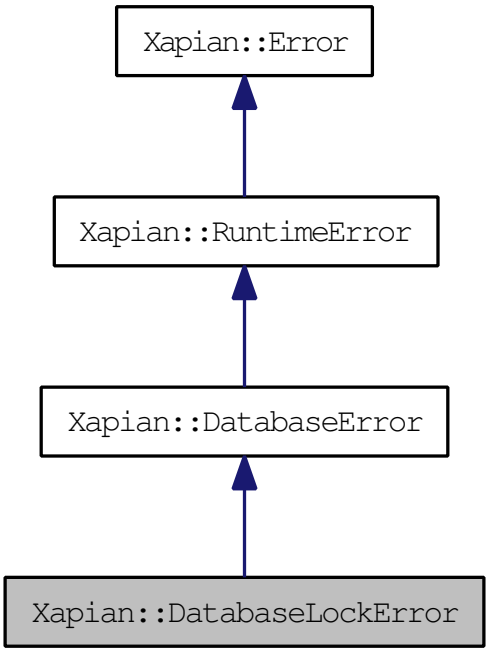
The documentation for this class was generated from the following file:

- xapian/error.h

7.9 Xapian::DatabaseLockError Class Reference

DatabaseLockError indicates failure to lock a database.

Inheritance diagram for Xapian::DatabaseLockError:



Public Member Functions

- DatabaseLockError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- DatabaseLockError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.9.1 Detailed Description

DatabaseLockError indicates failure to lock a database.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Xapian::DatabaseLockError::DatabaseLockError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.9.2.2 Xapian::DatabaseLockError::DatabaseLockError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

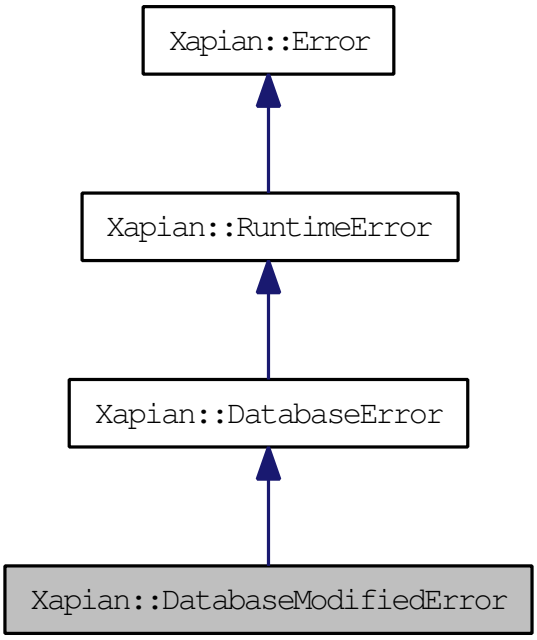
The documentation for this class was generated from the following file:

- xapian/error.h

7.10 Xapian::DatabaseModifiedError Class Reference

DatabaseModifiedError indicates a database was modified.

Inheritance diagram for Xapian::DatabaseModifiedError:



Public Member Functions

- DatabaseModifiedError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- DatabaseModifiedError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.10.1 Detailed Description

DatabaseModifiedError indicates a database was modified.

To recover after catching this error, you need to call Xapian::Database::reopen() on the Database and repeat the operation which failed.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 Xapian::DatabaseModifiedError::DatabaseModifiedError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.10.2.2 Xapian::DatabaseModifiedError::DatabaseModifiedError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

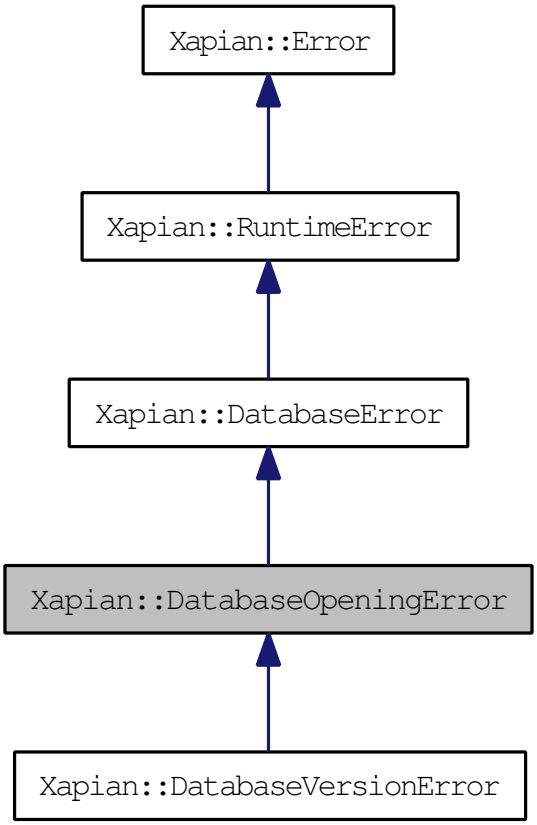
The documentation for this class was generated from the following file:

- xapian/error.h

7.11 Xapian::DatabaseOpeningError Class Reference

DatabaseOpeningError indicates failure to open a database.

Inheritance diagram for Xapian::DatabaseOpeningError:



Public Member Functions

- DatabaseOpeningError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- DatabaseOpeningError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.11.1 Detailed Description

DatabaseOpeningError indicates failure to open a database.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Xapian::DatabaseOpeningError::DatabaseOpeningError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.11.2.2 Xapian::DatabaseOpeningError::DatabaseOpeningError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

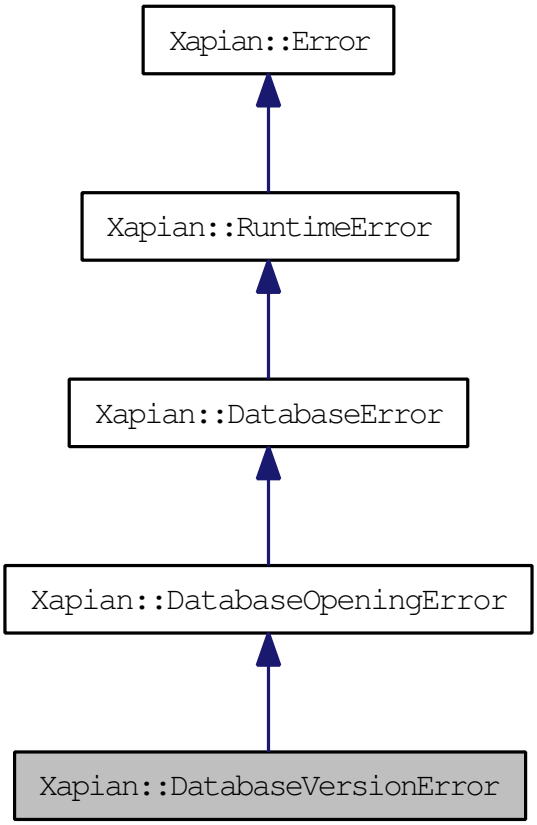
The documentation for this class was generated from the following file:

- xapian/error.h

7.12 Xapian::DatabaseVersionError Class Reference

DatabaseVersionError indicates that a database is in an unsupported format.

Inheritance diagram for Xapian::DatabaseVersionError:



Public Member Functions

- DatabaseVersionError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- DatabaseVersionError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.12.1 Detailed Description

DatabaseVersionError indicates that a database is in an unsupported format.

From time to time, new versions of Xapian will require the database format to be changed, to allow new information to be stored or new optimisations to be performed. Backwards compatibility will sometimes be maintained, so that new versions of Xapian can open old databases, but in some cases Xapian will be unable to open a database because it is in too old (or new) a format. This can be resolved either be upgrading or downgrading the version of Xapian in use, or by rebuilding the database from scratch with the current version of Xapian.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Xapian::DatabaseVersionError::DatabaseVersionError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

*msg_* Message giving details of the error, intended for human consumption.

*context_* Optional context information for this error.

*errno_* Optional errno value associated with this error.

7.12.2.2 Xapian::DatabaseVersionError::DatabaseVersionError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

*msg_* Message giving details of the error, intended for human consumption.

*errno_* Optional errno value associated with this error.

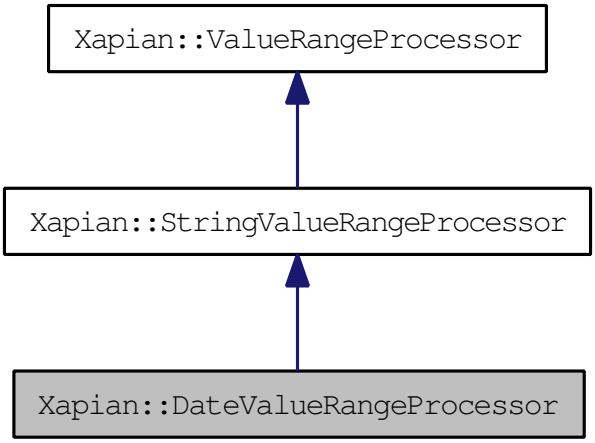
The documentation for this class was generated from the following file:

- xapian/error.h

7.13 Xapian::DateValueRangeProcessor Class Reference

Handle a date range.

Inheritance diagram for Xapian::DateValueRangeProcessor:



Public Member Functions

- DateValueRangeProcessor (Xapian::valueno slot_, bool prefer_mdy_=false, int epoch_year_=1970)

  Constructor.

- DateValueRangeProcessor (Xapian::valueno slot_, const std::string &str_, bool prefix_=true, bool prefer_mdy_=false, int epoch_year_=1970)

  Constructor.

- Xapian::valueno operator() (std::string &begin, std::string &end)

  Check for a valid date range.

7.13.1 Detailed Description

Handle a date range.

Begin and end must be dates in a recognised format.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Xapian::DateValueRangeProcessor::DateValueRangeProcessor (Xapian::valueno *slot_*, bool *prefer_mdy_* = `false`, int *epoch_year_* = `1970`) `[inline]`

Constructor.

Parameters:

*slot_* The value number to return from operator().

*prefer_mdy_* Should ambiguous dates be interpreted as month/day/year rather than day/month/year? (default: false)

*epoch_year_* Year to use as the epoch for dates with 2 digit years (default: 1970, so 1/1/69 is 2069 while 1/1/70 is 1970).

7.13.2.2 Xapian::DateValueRangeProcessor::DateValueRangeProcessor (Xapian::valueno *slot_*, const std::string & *str_*, bool *prefix_* = `true`, bool *prefer_mdy_* = `false`, int *epoch_year_* = `1970`) `[inline]`

Constructor.

Parameters:

*slot_* The value number to return from operator().

*str_* A string to look for to recognise values as belonging to this date range.

*prefix_* Whether to look for the string at the start or end of the values. If true, the string is a prefix; if false, the string is a suffix (default: true).

*prefer_mdy_* Should ambiguous dates be interpreted as month/day/year rather than day/month/year? (default: false)

*epoch_year_* Year to use as the epoch for dates with 2 digit years (default: 1970, so 1/1/69 is 2069 while 1/1/70 is 1970).

The string supplied in str_ is used by *operator()* to decide whether the pair of strings supplied to it constitute a valid range. If prefix_ is true, the first value in a range must begin with str_ (and the second value may optionally begin with str_); if prefix_ is false, the second value in a range must end with str_ (and the first value may optionally end with str_).

If str_ is empty, the setting of prefix_ is irrelevant, and no special strings are required at the start or end of the strings defining the range.

The remainder of both strings defining the endpoints must be valid dates.

For example, if str_ is "created:" and prefix_ is true, and the range processor has been added to the queryparser, the queryparser will accept "created:1/1/2000..31/12/2001".

7.13.3 Member Function Documentation

7.13.3.1 Xapian::valueno Xapian::DateValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) [virtual]

Check for a valid date range.

Parameters:

↔ ***begin*** The start of the range as specified in the query string by the user. This parameter is a non-const reference so the ValueRangeProcessor can modify it to return the value to start the range with.

↔ ***end*** The end of the range. This is also a non-const reference so it can be modified.

Returns:

If BEGIN..END is a sensible date range, this method modifies them into the format YYYYMMDD and returns the value of slot_ passed at construction time. Otherwise it returns Xapian::BAD_VALUENO.

Reimplemented from Xapian::StringValueRangeProcessor.

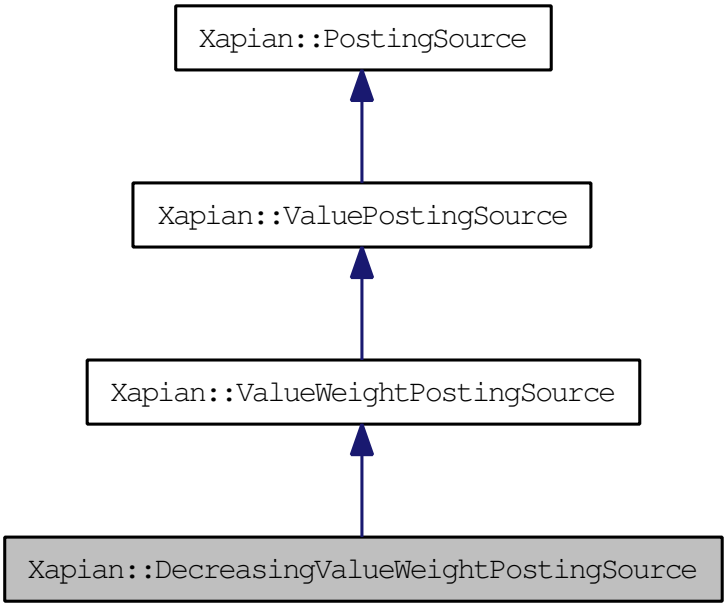
The documentation for this class was generated from the following file:

- xapian/queryparser.h

7.14 Xapian::DecreasingValueWeightPostingSource Class Reference

Read weights from a value which is known to decrease as docid increases.

Inheritance diagram for Xapian::DecreasingValueWeightPostingSource:



Public Member Functions

- Xapian::weight get_weight () const

  Return the weight contribution for the current document.

- DecreasingValueWeightPostingSource * clone () const

  Clone the posting source.

- std::string name () const

  Name of the posting source class.

- std::string serialise () const

  Serialise object parameters into a string.

- DecreasingValueWeightPostingSource * unserialise (const std::string &s) const

  Create object given string serialisation returned by serialise().

- void init (const Xapian::Database &db_)

Set this PostingSource to the start of the list of postings.

- void next (Xapian::weight min_wt)

  Advance the current position to the next matching document.

- void skip_to (Xapian::docid min_docid, Xapian::weight min_wt)

  Advance to the specified docid.

- bool check (Xapian::docid min_docid, Xapian::weight min_wt)

  Check if the specified docid occurs.

- std::string get_description () const

  Return a string describing this object.

Protected Member Functions

- void skip_if_in_range (Xapian::weight min_wt)

  Skip the iterator forward if in the decreasing range, and weight is low.

Protected Attributes

- bool items_at_end

  Flag, set to true if there are docs after the end of the range.

7.14.1 Detailed Description

Read weights from a value which is known to decrease as docid increases.

This posting source can be used, like ValueWeightPostingSource, to add a weight contribution to a query based on the values stored in a slot. The values in the slot must be serialised as by *sortable_serialise()*.

However, this posting source is additionally given a range of document IDs, within which the weight is known to be decreasing. ie, for all documents with ids A and B within this range (including the endpoints), where A is less than B, the weight of A is less than or equal to the weight of B. This can allow the posting source to skip to the end of the range quickly if insufficient weight is left in the posting source for a particular source.

By default, the range is assumed to cover all document IDs.

The ordering property can be arranged at index time, or by sorting an indexed database to produce a new, sorted, database.

7.14.2 Member Function Documentation

7.14.2.1 bool Xapian::DecreasingValueWeightPostingSource::check (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after *did*.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn't.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

did The document id to check.

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::ValuePostingSource.

7.14.2.2 DecreasingValueWeightPostingSource* Xapian::DecreasingValueWeightPostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.3 std::string Xapian::DecreasingValueWeightPostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.4 Xapian::weight Xapian::DecreasingValueWeightPostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.5 void Xapian::DecreasingValueWeightPostingSource::init (const Xapian::Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.6 std::string Xapian::DecreasingValueWeightPostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.7 void Xapian::DecreasingValueWeightPostingSource::next (Xapian::weight *min_wt*) [virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::ValuePostingSource.

7.14.2.8 std::string Xapian::DecreasingValueWeightPostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.9 void Xapian::DecreasingValueWeightPostingSource::skip_to (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Advance to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at_end() if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

did The document id to advance to.

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::ValuePostingSource.

7.14.2.10 DecreasingValueWeightPostingSource* Xapian::DecreasingValueWeightPostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::ValueWeightPostingSource.

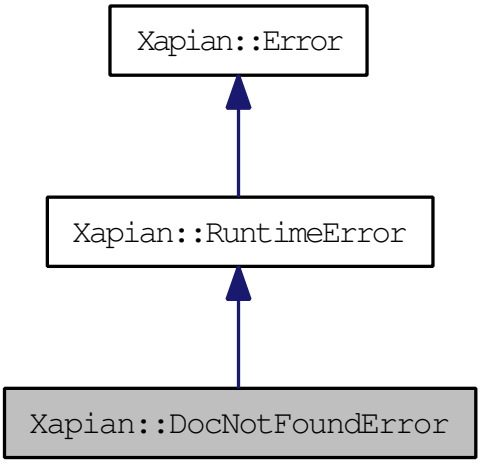
The documentation for this class was generated from the following file:

- xapian/postingsource.h

7.15 Xapian::DocNotFoundError Class Reference

Indicates an attempt to access a document not present in the database.

Inheritance diagram for Xapian::DocNotFoundError:



Public Member Functions

- DocNotFoundError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- DocNotFoundError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.15.1 Detailed Description

Indicates an attempt to access a document not present in the database.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 Xapian::DocNotFoundError::DocNotFoundError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.15.2.2 Xapian::DocNotFoundError::DocNotFoundError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

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

- xapian/error.h

7.16 Xapian::Document Class Reference

A handle representing a document in a Xapian database.

Public Member Functions

- Document (const Document &other)

  Copying is allowed.

- void operator= (const Document &other)

  Assignment is allowed.

- Document ()

  Make a new empty Document.

- ~Document ()

  Destructor.

- std::string get_value (Xapian::valueno slot) const

  Get value by number.

- void add_value (Xapian::valueno slot, const std::string &value)

  Add a new value.

- void remove_value (Xapian::valueno slot)

  Remove any value with the given number.

- void clear_values ()

  Remove all values associated with the document.

- std::string get_data () const

  Get data stored in the document.

- void set_data (const std::string &data)

  Set data stored in the document.

- void add_posting (const std::string &tname, Xapian::termpos tpos, Xapian::termcount wdfinc=1)

  Add an occurrence of a term at a particular position.

- void add_term (const std::string &tname, Xapian::termcount wdfinc=1)

  Add a term to the document, without positional information.

- void add_boolean_term (const std::string &term)

  Add a boolean filter term to the document.

- void remove_posting (const std::string &tname, Xapian::termpos tpos, Xapian::termcount wdfdec=1)

  Remove a posting of a term from the document.

- void remove_term (const std::string &tname)

  Remove a term and all postings associated with it.

- void clear_terms ()

  Remove all terms (and postings) from the document.

- Xapian::termcount termlist_count () const

  The length of the termlist - i.e.

- TermIterator termlist_begin () const

  Iterator for the terms in this document.

- TermIterator termlist_end () const

  Equivalent end iterator for termlist_begin().

- Xapian::termcount values_count () const

  Count the values in this document.

- ValueIterator values_begin () const

  Iterator for the values in this document.

- ValueIteratorEnd_ values_end () const

  Equivalent end iterator for values_begin().

- docid get_docid () const

  Get the document id which is associated with this document (if any).

- std::string serialise () const

  Serialise document into a string.

- std::string get_description () const

  Return a string describing this object.

Static Public Member Functions

- static Document unserialise (const std::string &s)

  Unserialise a document from a string produced by serialise().

7.16.1 Detailed Description

A handle representing a document in a Xapian database.

The Document class fetches information from the database lazily. Usually this behaviour isn't visible to users (except for the speed benefits), but if the document in the database is modified or deleted, then preexisting Document objects may return the old or new versions of data (or throw Xapian::DocNotFoundError in the case of deletion).

Since Database objects work on a snapshot of the database's state, the situation above can only happen with a WritableDatabase object, or if you call Database::reopen() on a Database object.

We recommend you avoid designs where this behaviour is an issue, but if you need a way to make a non-lazy version of a Document object, you can do this like so:

doc = Xapian::Document::unserialise(doc.serialise());

7.16.2 Constructor & Destructor Documentation

7.16.2.1 Xapian::Document::Document (const Document & *other*)

Copying is allowed.

The internals are reference counted, so copying is cheap.

Parameters:

other The object to copy.

7.16.3 Member Function Documentation

7.16.3.1 void Xapian::Document::add_boolean_term (const std::string & *term*) `[inline]`

Add a boolean filter term to the document.

This method adds *term* to the document with wdf of 0 - this is generally what you want for a term used for boolean filtering as the wdf of such terms is ignored, and it doesn't make sense for them to contribute to the document's length.

If the specified term already indexes this document, this method has no effect.

It is exactly the same as add_term(term, 0).

This method was added in Xapian 1.0.18.

Parameters:

term The term to add.

7.16.3.2 void Xapian::Document::add_posting (const std::string & *tname*, Xapian::termpos *tpos*, Xapian::termcount *wdfinc* = 1)

Add an occurrence of a term at a particular position.

Multiple occurrences of the term at the same position are represented only once in the positional information, but do increase the wdf.

If the term is not already in the document, it will be added to it.

Parameters:

tname The name of the term.

tpos The position of the term.

wdfinc The increment that will be applied to the wdf for this term.

7.16.3.3 void Xapian::Document::add_term (const std::string & *tname*, Xapian::termcount *wdfinc* = 1)

Add a term to the document, without positional information.

Any existing positional information for the term will be left unmodified.

Parameters:

tname The name of the term.

wdfinc The increment that will be applied to the wdf for this term (default: 1).

7.16.3.4 void Xapian::Document::add_value (Xapian::valueno *slot*, const std::string & *value*)

Add a new value.

The new value will replace any existing value with the same number (or if the new value is empty, it will remove any existing value with the same number).

Parameters:

slot The value slot to add the value in.

value The value to set.

7.16.3.5 std::string Xapian::Document::get_data () const

Get data stored in the document.

This is potentially a relatively expensive operation, and shouldn't normally be used during the match (e.g. in a PostingSource or match decider functor. Put data for use by match deciders in a value instead.

7.16.3.6 docid Xapian::Document::get_docid () const

Get the document id which is associated with this document (if any).

NB If multiple databases are being searched together, then this will be the document id in the individual database, not the merged database!

Returns:

If this document came from a database, return the document id in that database. Otherwise, return 0 (in Xapian 1.0.22/1.2.4 or later; prior to this the returned value was uninitialised).

7.16.3.7 std::string Xapian::Document::get_value (Xapian::valueno *slot*) const

Get value by number.

Returns an empty string if no value with the given number is present in the document.

Parameters:

slot The number of the value.

7.16.3.8 void Xapian::Document::operator= (const Document & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

Parameters:

other The object to copy.

7.16.3.9 void Xapian::Document::remove_posting (const std::string & *tname*, Xapian::termpos *tpos*, Xapian::termcount *wdfdec* = 1)

Remove a posting of a term from the document.

Note that the term will still index the document even if all occurrences are removed. To remove a term from a document completely, use remove_term().

Parameters:

tname The name of the term.

tpos The position of the term.

wdfdec The decrement that will be applied to the wdf when removing this posting. The wdf will not go below the value of 0.

Exceptions:

Xapian::InvalidArgumentError will be thrown if the term is not at the position specified in the position list for this term in this document.

Xapian::InvalidArgumentError will be thrown if the term is not in the document

7.16.3.10 void Xapian::Document::remove_term (const std::string & *tname*)

Remove a term and all postings associated with it.

Parameters:

tname The name of the term.

Exceptions:

Xapian::InvalidArgumentError will be thrown if the term is not in the document

7.16.3.11 std::string Xapian::Document::serialise () const

Serialise document into a string.

The document representation may change between Xapian releases: even between minor versions. However, it is guaranteed not to change if the remote database protocol has not changed between releases.

7.16.3.12 void Xapian::Document::set_data (const std::string & *data*)

Set data stored in the document.

Xapian treats the data as an opaque blob. It may try to compress it, but other than that it will just store it and return it when requested.

Parameters:

data The data to store.

7.16.3.13 Xapian::termcount Xapian::Document::termlist_count () const

The length of the termlist - i.e.

the number of different terms which index this document.

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

- xapian/document.h

7.17 Xapian::Enquire Class Reference

This class provides an interface to the information retrieval system for the purpose of searching.

Public Member Functions

- Enquire (const Enquire &other)

  Copying is allowed (and is cheap).

- void operator= (const Enquire &other)

  Assignment is allowed (and is cheap).

- Enquire (const Database &database, ErrorHandler *errorhandler_=0)

  Create a Xapian::Enquire object.

- ~Enquire ()

  Close the Xapian::Enquire object.

- void set_query (const Xapian::Query &query, Xapian::termcount qlen=0)

  Set the query to run.

- const Xapian::Query & get_query () const

  Get the query which has been set.

- void add_matchspy (MatchSpy *spy)

  Add a matchspy.

- void clear_matchspies ()

  Remove all the matchspies.

- void set_weighting_scheme (const Weight &weight_)

  Set the weighting scheme to use for queries.

- void set_collapse_key (Xapian::valueno collapse_key, Xapian::doccount collapse_max=1)

  Set the collapse key to use for queries.

- void set_docid_order (docid_order order)

  Set the direction in which documents are ordered by document id in the returned MSet.

- void set_cutoff (Xapian::percent percent_cutoff, Xapian::weight weight_cutoff=0)

  Set the percentage and/or weight cutoffs.

- void set_sort_by_relevance ()

Set the sorting to be by relevance only.

- void set_sort_by_value (Xapian::valueno sort_key, bool reverse)

  Set the sorting to be by value only.

- void set_sort_by_key (Xapian::KeyMaker *sorter, bool reverse)

  Set the sorting to be by key generated from values only.

- void set_sort_by_value_then_relevance (Xapian::valueno sort_key, bool reverse)

  Set the sorting to be by value, then by relevance for documents with the same value.

- void set_sort_by_key_then_relevance (Xapian::KeyMaker *sorter, bool reverse)

  Set the sorting to be by keys generated from values, then by relevance for documents with identical keys.

- void set_sort_by_relevance_then_value (Xapian::valueno sort_key, bool reverse)

  Set the sorting to be by relevance then value.

- void set_sort_by_relevance_then_key (Xapian::KeyMaker *sorter, bool reverse)

  Set the sorting to be by relevance, then by keys generated from values.

- ESet get_eset (Xapian::termcount maxitems, const RSet &omrset, int flags=0, double k=1.0, const Xapian::ExpandDecider *edecider=0) const

  Get the expand set for the given rset.

- ESet get_eset (Xapian::termcount maxitems, const RSet &omrset, const Xapian::ExpandDecider *edecider) const

  Get the expand set for the given rset.

- ESet get_eset (Xapian::termcount maxitems, const RSet &omrset, int flags, double k, const Xapian::ExpandDecider *edecider, Xapian::weight min_wt) const

  Get the expand set for the given rset.

- TermIterator get_matching_terms_begin (Xapian::docid did) const

  Get terms which match a given document, by document id.

- TermIterator get_matching_terms_end (Xapian::docid) const

  End iterator corresponding to get_matching_terms_begin().

- TermIterator get_matching_terms_begin (const MSetIterator &it) const

  Get terms which match a given document, by match set item.

- TermIterator get_matching_terms_end (const MSetIterator &) const

End iterator corresponding to get_matching_terms_begin().

- std::string get_description () const

  Return a string describing this object.

- MSet get_mset (Xapian::doccount first, Xapian::doccount maxitems, Xapian::doccount checkatleast=0, const RSet *omrset=0, const MatchDecider *mdecider=0) const

  Get (a portion of) the match set for the current query.

- MSet get_mset (Xapian::doccount first, Xapian::doccount maxitems, Xapian::doccount checkatleast, const RSet *omrset, const MatchDecider *mdecider, const MatchDecider *matchspy) const

  Get (a portion of) the match set for the current query.

- MSet get_mset (Xapian::doccount first, Xapian::doccount maxitems, const RSet *omrset, const MatchDecider *mdecider=0) const

  Get (a portion of) the match set for the current query.

7.17.1 Detailed Description

This class provides an interface to the information retrieval system for the purpose of searching.

Databases are usually opened lazily, so exceptions may not be thrown where you would expect them to be. You should catch Xapian::Error exceptions when calling any method in Xapian::Enquire.

Exceptions:

Xapian::InvalidArgumentError will be thrown if an invalid argument is supplied, for example, an unknown database type.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Xapian::Enquire::Enquire (const Database & *database*, ErrorHandler * *errorhandler_* = 0) [explicit]

Create a Xapian::Enquire object.

This specification cannot be changed once the Xapian::Enquire is opened: you must create a new Xapian::Enquire object to access a different database, or set of databases.

The database supplied must have been initialised (ie, must not be the result of calling the Database::Database() constructor). If you need to handle a situation where you have no index gracefully, a database created with InMemory::open() can be passed here, which represents a completely empty database.

Parameters:

database Specification of the database or databases to use.

errorhandler_ A pointer to the error handler to use. Ownership of the object pointed to is not assumed by the Xapian::Enquire object - the user should delete the Xapian::ErrorHandler object after the Xapian::Enquire object is deleted. To use no error handler, this parameter should be 0.

Exceptions:

Xapian::InvalidArgumentError will be thrown if an empty Database object is supplied.

7.17.3 Member Function Documentation

7.17.3.1 void Xapian::Enquire::add_matchspy (MatchSpy * *spy*)

Add a matchspy.

This matchspy will be called with some of the documents which match the query, during the match process. Exactly which of the matching documents are passed to it depends on exactly when certain optimisations occur during the match process, but it can be controlled to some extent by setting the *checkatleast* parameter to *get_mset()*.

In particular, if there are enough matching documents, at least the number specified by *checkatleast* will be passed to the matchspy. This means that you can force the matchspy to be shown all matching documents by setting *checkatleast* to the number of documents in the database.

Parameters:

spy The MatchSpy subclass to add. The caller must ensure that this remains valid while the Enquire object remains active, or until *clear_matchspies()* is called.

7.17.3.2 ESet Xapian::Enquire::get_eset (Xapian::termcount *maxitems*, const RSet & *omrset*, int *flags*, double *k*, const Xapian::ExpandDecider * *edecider*, Xapian::weight *min_wt*) const

Get the expand set for the given rset.

Parameters:

maxitems the maximum number of items to return.

omrset the relevance set to use when performing the expand operation.

flags zero or more of these values |-ed together:

- Xapian::Enquire::INCLUDE_QUERY_TERMS query terms may be returned from expand
- Xapian::Enquire::USE_EXACT_TERMFREQ for multi dbs, calculate the exact termfreq; otherwise an approximation is used which can greatly improve efficiency, but still returns good results.

k the parameter k in the query expansion algorithm (default is 1.0)

edecider a decision functor to use to decide whether a given term should be put in the ESet

min_wt the minimum weight for included terms

Returns:

An ESet object containing the results of the expand.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

7.17.3.3 ESet Xapian::Enquire::get_eset (Xapian::termcount *maxitems*, const RSet & *omrset*, const Xapian::ExpandDecider * *edecider*) const `[inline]`

Get the expand set for the given rset.

Parameters:

maxitems the maximum number of items to return.

omrset the relevance set to use when performing the expand operation.

edecider a decision functor to use to decide whether a given term should be put in the ESet

Returns:

An ESet object containing the results of the expand.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

7.17.3.4 ESet Xapian::Enquire::get_eset (Xapian::termcount *maxitems*, const RSet & *omrset*, int *flags* = `0`, double *k* = `1.0`, const Xapian::ExpandDecider * *edecider* = `0`) const

Get the expand set for the given rset.

Parameters:

maxitems the maximum number of items to return.

omrset the relevance set to use when performing the expand operation.

flags zero or more of these values |-ed together:

- Xapian::Enquire::INCLUDE_QUERY_TERMS query terms may be returned from expand

- Xapian::Enquire::USE_EXACT_TERMFREQ for multi dbs, calculate the exact termfreq; otherwise an approximation is used which can greatly improve efficiency, but still returns good results.

k the parameter k in the query expansion algorithm (default is 1.0)

edecider a decision functor to use to decide whether a given term should be put in the ESet

Returns:

An ESet object containing the results of the expand.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

7.17.3.5 TermIterator Xapian::Enquire::get_matching_terms_begin (const MSetIterator & *it*) const

Get terms which match a given document, by match set item.

This method returns the terms in the current query which match the given document.

If the underlying database has suitable support, using this call (rather than passing a Xapian::docid) will enable the system to ensure that the correct data is returned, and that the document has not been deleted or changed since the query was performed.

Parameters:

it The iterator for which to retrieve the matching terms.

Returns:

An iterator returning the terms which match the document. The terms will be returned (as far as this makes any sense) in the same order as the terms in the query. Terms will not occur more than once, even if they do in the query.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

Xapian::DocNotFoundError The document specified could not be found in the database.

7.17.3.6 TermIterator Xapian::Enquire::get_matching_terms_begin (Xapian::docid *did*) const

Get terms which match a given document, by document id.

This method returns the terms in the current query which match the given document.

It is possible for the document to have been removed from the database between the time it is returned in an MSet, and the time that this call is made. If possible, you should specify an MSetIterator instead of a Xapian::docid, since this will enable database backends with suitable support to prevent this occurring.

Note that a query does not need to have been run in order to make this call.

Parameters:

did The document id for which to retrieve the matching terms.

Returns:

An iterator returning the terms which match the document. The terms will be returned (as far as this makes any sense) in the same order as the terms in the query. Terms will not occur more than once, even if they do in the query.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

Xapian::DocNotFoundError The document specified could not be found in the database.

7.17.3.7 MSet Xapian::Enquire::get_mset (Xapian::doccount *first*, Xapian::doccount *maxitems*, const RSet * *omrset*, const MatchDecider * *mdecider* = 0) const [inline]

Get (a portion of) the match set for the current query.

Parameters:

first the first item in the result set to return. A value of zero corresponds to the first item returned being that with the highest score. A value of 10 corresponds to the first 10 items being ignored, and the returned items starting at the eleventh.

maxitems the maximum number of items to return. If you want all matches, then you can pass the result of calling get_doccount() on the Database object (though if you are doing this so you can filter results, you are likely to get much better performance by using Xapian's match-time filtering features instead). You can pass 0 for maxitems which will give you an empty MSet with valid statistics (such as get_matches_estimated()) calculated without looking at any postings, which is very quick, but means the estimates may be more approximate and the bounds may be much looser.

checkatleast the minimum number of items to check. Because the matcher optimises, it won't consider every document which might match, so the total number of matches is estimated. Setting checkatleast forces it to consider at least this many matches and so allows for reliable paging links.

omrset the relevance set to use when performing the query.

mdecider a decision functor to use to decide whether a given document should be put in the MSet.

matchspy a decision functor to use to decide whether a given document should be put in the MSet. The matchspy is applied to every document which is a potential candidate for the MSet, so if there are checkatleast or more such documents, the matchspy will see at least checkatleast. The mdecider is assumed to be a relatively expensive test so may be applied in a lazier fashion.

Deprecated

this parameter is deprecated - use the newer MatchSpy class and add_matchspy() method instead.

Returns:

A Xapian::MSet object containing the results of the query.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

7.17.3.8 MSet Xapian::Enquire::get_mset (Xapian::doccount *first*, Xapian::doccount *maxitems*, Xapian::doccount *checkatleast*, const RSet * *omrset*, const MatchDecider * *mdecider*, const MatchDecider * *matchspy*) const

Get (a portion of) the match set for the current query.

Parameters:

first the first item in the result set to return. A value of zero corresponds to the first item returned being that with the highest score. A value of 10 corresponds to the first 10 items being ignored, and the returned items starting at the eleventh.

maxitems the maximum number of items to return. If you want all matches, then you can pass the result of calling get_doccount() on the Database object (though if you are doing this so you can filter results, you are likely to get much better performance by using Xapian's match-time filtering features instead). You can pass 0 for maxitems which will give you an empty MSet with valid statistics (such as get_matches_estimated()) calculated without looking at any postings, which is very quick, but means the estimates may be more approximate and the bounds may be much looser.

checkatleast the minimum number of items to check. Because the matcher optimises, it won't consider every document which might match, so the total number of matches is estimated. Setting checkatleast forces it to consider at least this many matches and so allows for reliable paging links.

omrset the relevance set to use when performing the query.

mdecider a decision functor to use to decide whether a given document should be put in the MSet.

matchspy a decision functor to use to decide whether a given document should be put in the MSet. The matchspy is applied to every document which is a potential candidate for the MSet, so if there are checkatleast or more such documents, the matchspy will see at least checkatleast. The mdecider is assumed to be a relatively expensive test so may be applied in a lazier fashion.

Deprecated

this parameter is deprecated - use the newer MatchSpy class and add_matchspy() method instead.

Returns:

A Xapian::MSet object containing the results of the query.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

7.17.3.9 MSet Xapian::Enquire::get_mset (Xapian::doccount *first*, Xapian::doccount *maxitems*, Xapian::doccount *checkatleast* = 0, const RSet * *omrset* = 0, const MatchDecider * *mdecider* = 0) const

Get (a portion of) the match set for the current query.

Parameters:

first the first item in the result set to return. A value of zero corresponds to the first item returned being that with the highest score. A value of 10 corresponds to the first 10 items being ignored, and the returned items starting at the eleventh.

maxitems the maximum number of items to return. If you want all matches, then you can pass the result of calling get_doccount() on the Database object (though if you are doing this so you can filter results, you are likely to get much better performance by using Xapian's match-time filtering features instead). You can pass 0 for maxitems which will give you an empty MSet with valid statistics (such as get_matches_estimated()) calculated without looking at any postings, which is very quick, but means the estimates may be more approximate and the bounds may be much looser.

checkatleast the minimum number of items to check. Because the matcher optimises, it won't consider every document which might match, so the total number of matches is estimated. Setting checkatleast forces it to consider at least this many matches and so allows for reliable paging links.

omrset the relevance set to use when performing the query.

mdecider a decision functor to use to decide whether a given document should be put in the MSet.

matchspy a decision functor to use to decide whether a given document should be put in the MSet. The matchspy is applied to every document which is a

potential candidate for the MSet, so if there are checkatleast or more such documents, the matchspy will see at least checkatleast. The mdecider is assumed to be a relatively expensive test so may be applied in a lazier fashion.

Deprecated

this parameter is deprecated - use the newer MatchSpy class and add_matchspy() method instead.

Returns:

A Xapian::MSet object containing the results of the query.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

7.17.3.10 const Xapian::Query& Xapian::Enquire::get_query () const

Get the query which has been set.

This is only valid after set_query() has been called.

Exceptions:

Xapian::InvalidArgumentError will be thrown if query has not yet been set.

7.17.3.11 void Xapian::Enquire::set_collapse_key (Xapian::valueno *collapse_key*, Xapian::doccount *collapse_max* = 1)

Set the collapse key to use for queries.

Parameters:

collapse_key value number to collapse on - at most one MSet entry with each particular value will be returned (default is Xapian::BAD_VALUENO which means no collapsing).

collapse_max Max number of items with the same key to leave after collapsing (default 1).

The MSet returned by get_mset() will have only the "best" (at most) *collapse_max* entries with each particular value of *collapse_key* ("best" being highest ranked - i.e. highest weight or highest sorting key).

An example use might be to create a value for each document containing an MD5 hash of the document contents. Then duplicate documents from different sources can be eliminated at search time by collapsing with *collapse_max* = 1 (it's better to eliminate duplicates at index time, but this may not be always be possible - for example the search may be over more than one Xapian database).

Another use is to group matches in a particular category (e.g. you might collapse a mailing list search on the Subject: so that there's only one result per discussion thread). In this case you can use get_collapse_count() to give the user some idea how many other results there are. And if you index the Subject: as a boolean term as well as putting it in a value, you can offer a link to a non-collapsed search restricted to that thread using a boolean filter.

7.17.3.12 void Xapian::Enquire::set_cutoff (Xapian::percent *percent_cutoff*, Xapian::weight *weight_cutoff* = 0)

Set the percentage and/or weight cutoffs.

Parameters:

percent_cutoff Minimum percentage score for returned documents. If a document has a lower percentage score than this, it will not appear in the MSet. If your intention is to return only matches which contain all the terms in the query, then it's more efficient to use Xapian::Query::OP_AND instead of Xapian::Query::OP_OR in the query than to use set_cutoff(100). (default 0 => no percentage cut-off).

weight_cutoff Minimum weight for a document to be returned. If a document has a lower score that this, it will not appear in the MSet. It is usually only possible to choose an appropriate weight for cutoff based on the results of a previous run of the same query; this is thus mainly useful for alerting operations. The other potential use is with a user specified weighting scheme. (default 0 => no weight cut-off).

7.17.3.13 void Xapian::Enquire::set_docid_order (docid_order *order*)

Set the direction in which documents are ordered by document id in the returned MSet.

This order only has an effect on documents which would otherwise have equal rank. For a weighted probabilistic match with no sort value, this means documents with equal weight. For a boolean match, with no sort value, this means all documents. And if a sort value is used, this means documents with equal sort value (and also equal weight if ordering on relevance after the sort).

Parameters:

order This can be:

- Xapian::Enquire::ASCENDING docids sort in ascending order (default)
- Xapian::Enquire::DESCENDING docids sort in descending order
- Xapian::Enquire::DONT_CARE docids sort in whatever order is most efficient for the backend

Note: If you add documents in strict date order, then a boolean search - i.e. set_weighting_scheme(Xapian::BoolWeight()) - with set_docid_order(Xapian::Enquire::DESCENDING) is an efficient way to perform "sort by

date, newest first", and with set_docid_order(Xapian::Enquire::ASCENDING) a very efficient way to perform "sort by date, oldest first".

7.17.3.14 void Xapian::Enquire::set_query (const Xapian::Query & *query*, Xapian::termcount *qlen* = 0)

Set the query to run.

Parameters:

query the new query to run.

qlen the query length to use in weight calculations - by default the sum of the wqf of all terms is used.

7.17.3.15 void Xapian::Enquire::set_sort_by_key (Xapian::KeyMaker * *sorter*, bool *reverse*)

Set the sorting to be by key generated from values only.

Parameters:

sorter The functor to use for generating keys.

reverse If true, reverses the sort order.

7.17.3.16 void Xapian::Enquire::set_sort_by_key_then_relevance (Xapian::KeyMaker * *sorter*, bool *reverse*)

Set the sorting to be by keys generated from values, then by relevance for documents with identical keys.

Parameters:

sorter The functor to use for generating keys.

reverse If true, reverses the sort order.

7.17.3.17 void Xapian::Enquire::set_sort_by_relevance ()

Set the sorting to be by relevance only.

This is the default.

7.17.3.18 void Xapian::Enquire::set_sort_by_relevance_then_key (Xapian::KeyMaker * *sorter*, bool *reverse*)

Set the sorting to be by relevance, then by keys generated from values.

Note that with the default BM25 weighting scheme parameters, non-identical documents will rarely have the same weight, so this setting will give very similar results to set_sort_by_relevance(). It becomes more useful with particular BM25 parameter settings (e.g. BM25Weight(1,0,1,0,0)) or custom weighting schemes.

Parameters:

sorter The functor to use for generating keys.

reverse If true, reverses the sort order.

7.17.3.19 void Xapian::Enquire::set_sort_by_relevance_then_value (Xapian::valueno *sort_key*, bool *reverse*)

Set the sorting to be by relevance then value.

Note that sorting by values uses a string comparison, so to use this to sort by a numeric value you'll need to store the numeric values in a manner which sorts appropriately. For example, you could use Xapian::sortable_serialise() (which works for floating point numbers as well as integers), or store numbers padded with leading zeros or spaces, or with the number of digits prepended.

Note that with the default BM25 weighting scheme parameters, non-identical documents will rarely have the same weight, so this setting will give very similar results to set_sort_by_relevance(). It becomes more useful with particular BM25 parameter settings (e.g. BM25Weight(1,0,1,0,0)) or custom weighting schemes.

Parameters:

sort_key value number to sort on.

reverse If true, reverses the sort order.

7.17.3.20 void Xapian::Enquire::set_sort_by_value (Xapian::valueno *sort_key*, bool *reverse*)

Set the sorting to be by value only.

Note that sorting by values uses a string comparison, so to use this to sort by a numeric value you'll need to store the numeric values in a manner which sorts appropriately. For example, you could use Xapian::sortable_serialise() (which works for floating point numbers as well as integers), or store numbers padded with leading zeros or spaces, or with the number of digits prepended.

Parameters:

sort_key value number to sort on.

reverse If true, reverses the sort order.

7.17.3.21 void Xapian::Enquire::set_sort_by_value_then_relevance (Xapian::valueno *sort_key*, bool *reverse*)

Set the sorting to be by value, then by relevance for documents with the same value.

Note that sorting by values uses a string comparison, so to use this to sort by a numeric value you'll need to store the numeric values in a manner which sorts appropriately. For example, you could use Xapian::sortable_serialise() (which works for floating point numbers as well as integers), or store numbers padded with leading zeros or spaces, or with the number of digits prepended.

Parameters:

sort_key value number to sort on.

reverse If true, reverses the sort order.

7.17.3.22 void Xapian::Enquire::set_weighting_scheme (const Weight & *weight_*)

Set the weighting scheme to use for queries.

Parameters:

weight_ the new weighting scheme. If no weighting scheme is specified, the default is BM25 with the default parameters.

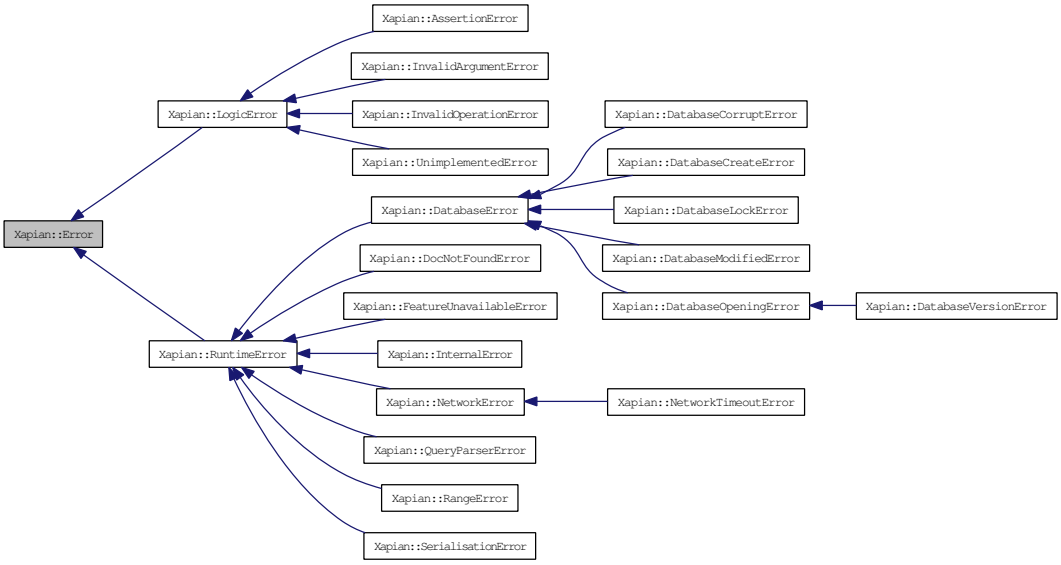
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.18 Xapian::Error Class Reference

All exceptions thrown by Xapian are subclasses of Xapian::Error.

Inheritance diagram for Xapian::Error:



Public Member Functions

- const char * get_type () const

  The type of this error (e.g. "DocNotFoundError".).

- const std::string & get_msg () const

  Message giving details of the error, intended for human consumption.

- const std::string & get_context () const

  Optional context information.

- const char * get_error_string () const

  Returns any system error string associated with this exception.

- std::string get_description () const

  Return a string describing this object.

7.18.1 Detailed Description

All exceptions thrown by Xapian are subclasses of Xapian::Error.

This class can not be instantiated directly - instead a subclass should be used.

7.18.2 Member Function Documentation

7.18.2.1 const std::string& Xapian::Error::get_context () const [inline]

Optional context information.

This context is intended for use by Xapian::ErrorHandler (for example so it can know which remote server is unreliable and report the problem and remove that server from those being searched). But it's typically a plain-text string, and so also fit for human consumption.

7.18.2.2 const char* Xapian::Error::get_error_string () const

Returns any system error string associated with this exception.

The system error string may come from errno, h_errno (on UNIX), or GetLastError() (on MS Windows). If there is no associated system error string, NULL is returned.

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

- xapian/error.h

7.19 Xapian::ErrorHandler Class Reference

Decide if a Xapian::Error exception should be ignored.

Public Member Functions

- ErrorHandler ()

  Default constructor.

- virtual ~ErrorHandler ()

  We require a virtual destructor because we have virtual methods.

- void operator() (Xapian::Error &error)

  Handle a Xapian::Error object.

7.19.1 Detailed Description

Decide if a Xapian::Error exception should be ignored.

You can create your own subclass of this class and pass in an instance of it when you construct a Xapian::Enquire object. Xapian::Error exceptions which happen during the match process are passed to this object and it can decide whether they should propagate or whether Enquire should attempt to continue.

The motivation is to allow searching over remote databases to handle a remote server which has died (both to allow results to be returned, and also so that such errors can be logged and dead servers temporarily removed from use).

7.19.2 Member Function Documentation

7.19.2.1 void Xapian::ErrorHandler::operator() (Xapian::Error & *error*)

Handle a Xapian::Error object.

This method is called when a Xapian::Error object is thrown and caught inside Enquire. If this is the first ErrorHandler that the Error has been passed to, then the handle_error() virtual method is called, which allows the API user to decide how to handle the error.

Parameters:

error The Xapian::Error object under consideration.

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

- xapian/errorhandler.h

7.20 Xapian::ESet Class Reference

Class representing an ordered set of expand terms (an ESet).

Public Member Functions

- ESet ()

  Construct an empty ESet.

- ~ESet ()

  Destructor.

- ESet (const ESet &other)

  Copying is allowed (and is cheap).

- void operator= (const ESet &other)

  Assignment is allowed (and is cheap).

- Xapian::termcount get_ebound () const

  A lower bound on the number of terms which are in the full set of results of the expand.

- Xapian::termcount size () const

  The number of terms in this E-Set.

- Xapian::termcount max_size () const

  Required to allow use as an STL container.

- bool empty () const

  Test if this E-Set is empty.

- void swap (ESet &other)

  Swap the E-Set we point to with another.

- ESetIterator begin () const

  Iterator for the terms in this E-Set.

- ESetIterator end () const

  End iterator corresponding to begin().

- ESetIterator back () const

  Iterator pointing to the last element of this E-Set.

- ESetIterator operator[ ] (Xapian::termcount i) const

  This returns the term at position i in this E-Set.

- std::string get_description () const

Return a string describing this object.

7.20.1 Detailed Description

Class representing an ordered set of expand terms (an ESet).

This set represents the results of an expand operation, which is performed by Xapian::Enquire::get_eset().

7.20.2 Member Function Documentation

7.20.2.1 Xapian::termcount Xapian::ESet::get_ebound () const

A lower bound on the number of terms which are in the full set of results of the expand.

This will be greater than or equal to size()

7.20.2.2 Xapian::termcount Xapian::ESet::max_size () const [inline]

Required to allow use as an STL container.

7.20.2.3 ESetIterator Xapian::ESet::operator[ ] (Xapian::termcount *i*) const

This returns the term at position i in this E-Set.

Parameters:

i The index into the ESet.

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

- xapian/enquire.h

7.21 Xapian::ESetIterator Class Reference

Iterate through terms in the ESet.

Public Types

- typedef std::bidirectional_iterator_tag iterator_category
  Allow use as an STL iterator.
- typedef std::string value_type
  Allow use as an STL iterator.
- typedef Xapian::termcount_diff difference_type
  Allow use as an STL iterator.
- typedef std::string * pointer
  Allow use as an STL iterator.
- typedef std::string & reference
  Allow use as an STL iterator.

Public Member Functions

- ESetIterator ()
  Create an uninitialised iterator; this cannot be used, but is convenient syntactically.
- ESetIterator (const ESetIterator &other)
  Copying is allowed (and is cheap).
- void operator= (const ESetIterator &other)
  Assignment is allowed (and is cheap).
- ESetIterator & operator++ ()
  Advance the iterator.
- ESetIterator operator++ (int)
  Advance the iterator (postfix variant).
- ESetIterator & operator– ()
  Decrement the iterator.
- ESetIterator operator– (int)
  Decrement the iterator (postfix variant).
- const std::string & operator* () const

Get the term for the current position.

- Xapian::weight get_weight () const

  Get the weight of the term at the current position.

- std::string get_description () const

  Return a string describing this object.

Friends

- bool operator== (const ESetIterator &a, const ESetIterator &b)

  Equality test for ESetIterator objects.

- bool operator!= (const ESetIterator &a, const ESetIterator &b)

  Inequality test for ESetIterator objects.

7.21.1 Detailed Description

Iterate through terms in the ESet.

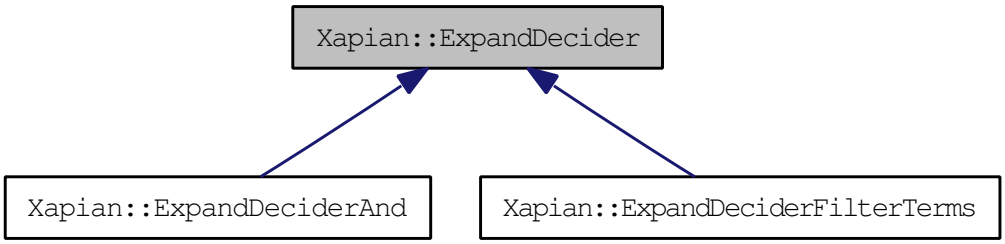
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.22 Xapian::ExpandDecider Class Reference

Virtual base class for expand decider functor.

Inheritance diagram for Xapian::ExpandDecider:



Public Member Functions

- virtual bool operator() (const std::string &term) const =0

  Do we want this term in the ESet?

- virtual ~ExpandDecider ()

  Virtual destructor, because we have virtual methods.

7.22.1 Detailed Description

Virtual base class for expand decider functor.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 virtual Xapian::ExpandDecider::~ExpandDecider () [virtual]

Virtual destructor, because we have virtual methods.

7.22.3 Member Function Documentation

7.22.3.1 virtual bool Xapian::ExpandDecider::operator() (const std::string & *term*) const [pure virtual]

Do we want this term in the ESet?

Parameters:

term The term to test.

Implemented in Xapian::ExpandDeciderAnd, and Xapian::ExpandDeciderFilterTerms.

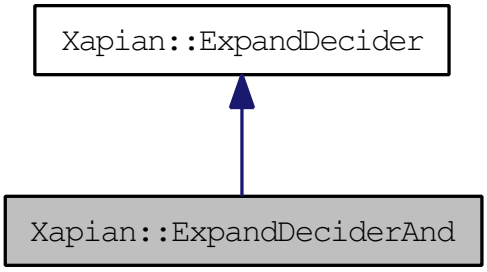
The documentation for this class was generated from the following file:

- xapian/expanddecider.h

7.23 Xapian::ExpandDeciderAnd Class Reference

ExpandDecider subclass which rejects terms using two ExpandDeciders.

Inheritance diagram for Xapian::ExpandDeciderAnd:



Public Member Functions

- ExpandDeciderAnd (const ExpandDecider &first_, const ExpandDecider &second_)

  Terms will be checked with first*, and if accepted, then checked with* second*.*

- ExpandDeciderAnd (const ExpandDecider *first_, const ExpandDecider *second_)

  Compatibility method.

- virtual bool operator() (const std::string &term) const

  Do we want this term in the ESet?

7.23.1 Detailed Description

ExpandDecider subclass which rejects terms using two ExpandDeciders.

Terms are only accepted if they are accepted by both of the specified ExpandDecider objects.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const ExpandDecider & *first_*, const ExpandDecider & *second_*) [inline]

Terms will be checked with *first*, and if accepted, then checked with *second*.

Parameters:

*first_* First ExpandDecider object to test with.

*second_* ExpandDecider object to test with if first_ accepts.

7.23.2.2 Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const ExpandDecider * *first_*, const ExpandDecider * *second_*) [inline]

Compatibility method.

Parameters:

*first_* First ExpandDecider object to test with.

*second_* ExpandDecider object to test with if first_ accepts.

7.23.3 Member Function Documentation

7.23.3.1 virtual bool Xapian::ExpandDeciderAnd::operator() (const std::string & *term*) const [virtual]

Do we want this term in the ESet?

Parameters:

term The term to test.

Implements Xapian::ExpandDecider.

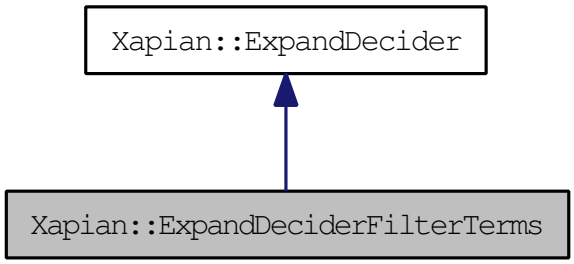
The documentation for this class was generated from the following file:

- xapian/expanddecider.h

7.24 Xapian::ExpandDeciderFilterTerms Class Reference

ExpandDecider subclass which rejects terms in a specified list.

Inheritance diagram for Xapian::ExpandDeciderFilterTerms:



Public Member Functions

- template<class Iterator >
  ExpandDeciderFilterTerms (Iterator reject_begin, Iterator reject_end)

  The two iterators specify a list of terms to be rejected.

- virtual bool operator() (const std::string &term) const

  Do we want this term in the ESet?

7.24.1 Detailed Description

ExpandDecider subclass which rejects terms in a specified list.

ExpandDeciderFilterTerms provides an easy way to filter out terms from a fixed list when generating an ESet.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 template<class Iterator > Xapian::ExpandDeciderFilterTerms::ExpandDeciderFilterTerms (Iterator *reject_begin*, Iterator *reject_end*) [inline]

The two iterators specify a list of terms to be rejected.

Parameters:

reject_begin Begin iterator for the list of terms to reject. It can be any input_-iterator type which returns std::string or char * (e.g. TermIterator or char **).

reject_end End iterator for the list of terms to reject.

7.24.3 Member Function Documentation

7.24.3.1 virtual bool Xapian::ExpandDeciderFilterTerms::operator() (const std::string & *term*) const [virtual]

Do we want this term in the ESet?

Parameters:

term The term to test.

Implements Xapian::ExpandDecider.

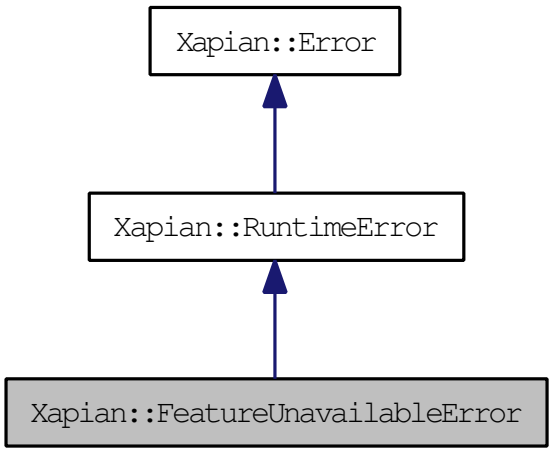
The documentation for this class was generated from the following file:

- xapian/expanddecider.h

7.25 Xapian::FeatureUnavailableError Class Reference

Indicates an attempt to use a feature which is unavailable.

Inheritance diagram for Xapian::FeatureUnavailableError:



Public Member Functions

- FeatureUnavailableError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- FeatureUnavailableError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.25.1 Detailed Description

Indicates an attempt to use a feature which is unavailable.

Typically a feature is unavailable because it wasn't compiled in, or because it requires other software or facilities which aren't available.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 Xapian::FeatureUnavailableError::FeatureUnavailableError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.25.2.2 Xapian::FeatureUnavailableError::FeatureUnavailableError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

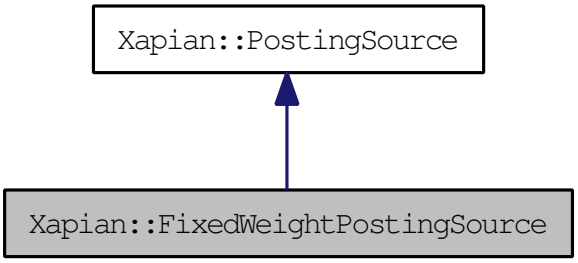
The documentation for this class was generated from the following file:

- xapian/error.h

7.26 Xapian::FixedWeightPostingSource Class Reference

A posting source which returns a fixed weight for all documents.

Inheritance diagram for Xapian::FixedWeightPostingSource:



Public Member Functions

- FixedWeightPostingSource (Xapian::weight wt)

  Construct a FixedWeightPostingSource.

- Xapian::doccount get_termfreq_min () const

  A lower bound on the number of documents this object can return.

- Xapian::doccount get_termfreq_est () const

  An estimate of the number of documents this object can return.

- Xapian::doccount get_termfreq_max () const

  An upper bound on the number of documents this object can return.

- Xapian::weight get_weight () const

  Return the weight contribution for the current document.

- void next (Xapian::weight min_wt)

  Advance the current position to the next matching document.

- void skip_to (Xapian::docid min_docid, Xapian::weight min_wt)

  Advance to the specified docid.

- bool check (Xapian::docid min_docid, Xapian::weight min_wt)

  Check if the specified docid occurs.

- bool at_end () const

  Return true if the current position is past the last entry in this list.

- Xapian::docid get_docid () const

  Return the current docid.

- FixedWeightPostingSource * clone () const

  Clone the posting source.

- std::string name () const

  Name of the posting source class.

- std::string serialise () const

  Serialise object parameters into a string.

- FixedWeightPostingSource * unserialise (const std::string &s) const

  Create object given string serialisation returned by serialise().

- void init (const Database &db_)

  Set this PostingSource to the start of the list of postings.

- std::string get_description () const

  Return a string describing this object.

7.26.1 Detailed Description

A posting source which returns a fixed weight for all documents.

This returns entries for all documents in the given database, with a fixed weight (specified by a parameter to the constructor).

7.26.2 Constructor & Destructor Documentation

7.26.2.1 Xapian::FixedWeightPostingSource::FixedWeightPostingSource (Xapian::weight *wt*)

Construct a FixedWeightPostingSource.

Parameters:

wt The fixed weight to return.

7.26.3 Member Function Documentation

7.26.3.1 bool Xapian::FixedWeightPostingSource::at_end () const [virtual]

Return true if the current position is past the last entry in this list.

At least one of *next()*, *skip_to()* or *check()* will be called before this method is first called.

Implements Xapian::PostingSource.

7.26.3.2 bool Xapian::FixedWeightPostingSource::check (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after *did*.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn't.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

did The document id to check.

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::PostingSource.

7.26.3.3 FixedWeightPostingSource* Xapian::FixedWeightPostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the

matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented from Xapian::PostingSource.

7.26.3.4 std::string Xapian::FixedWeightPostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.

7.26.3.5 Xapian::docid Xapian::FixedWeightPostingSource::get_docid () const [virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document". See *get_weight()* for details.

Note: in the case of a multi-database search, the returned docid should be in the single subdatabase relevant to this posting source. See the *init()* method for details.

Implements Xapian::PostingSource.

7.26.3.6 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_est () const [virtual]

An estimate of the number of documents this object can return.

It must always be true that:

get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

7.26.3.7 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_max () const [virtual]

An upper bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

7.26.3.8 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_min () const [virtual]

A lower bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

7.26.3.9 Xapian::weight Xapian::FixedWeightPostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::PostingSource.

7.26.3.10 void Xapian::FixedWeightPostingSource::init (const Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Implements Xapian::PostingSource.

7.26.3.11 std::string Xapian::FixedWeightPostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::PostingSource.

7.26.3.12 void Xapian::FixedWeightPostingSource::next (Xapian::weight *min_wt*) [virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Implements Xapian::PostingSource.

7.26.3.13 std::string Xapian::FixedWeightPostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::PostingSource.

7.26.3.14 void Xapian::FixedWeightPostingSource::skip_to (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Advance to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at_end() if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

did The document id to advance to.

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::PostingSource.

7.26.3.15 FixedWeightPostingSource* Xapian::FixedWeightPostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::PostingSource.

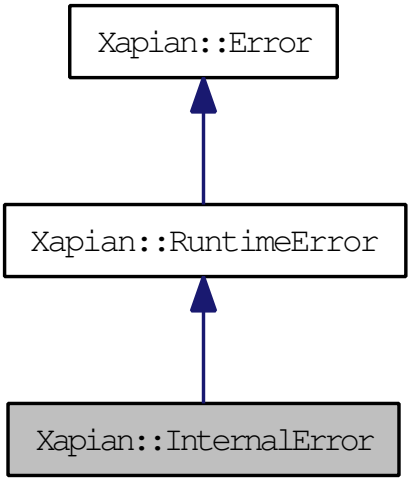
The documentation for this class was generated from the following file:

- xapian/postingsource.h

7.27 Xapian::InternalError Class Reference

InternalError indicates a runtime problem of some sort.

Inheritance diagram for Xapian::InternalError:



Public Member Functions

- InternalError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- InternalError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.27.1 Detailed Description

InternalError indicates a runtime problem of some sort.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 Xapian::InternalError::InternalError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.27.2.2 Xapian::InternalError::InternalError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

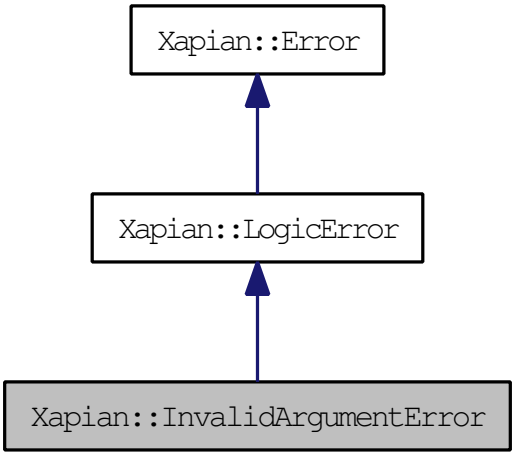
The documentation for this class was generated from the following file:

- xapian/error.h

7.28 Xapian::InvalidArgumentError Class Reference

InvalidArgumentError indicates an invalid parameter value was passed to the API.

Inheritance diagram for Xapian::InvalidArgumentError:



Public Member Functions

- InvalidArgumentError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- InvalidArgumentError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.28.1 Detailed Description

InvalidArgumentError indicates an invalid parameter value was passed to the API.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 Xapian::InvalidArgumentError::InvalidArgumentError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.28.2.2 Xapian::InvalidArgumentError::InvalidArgumentError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

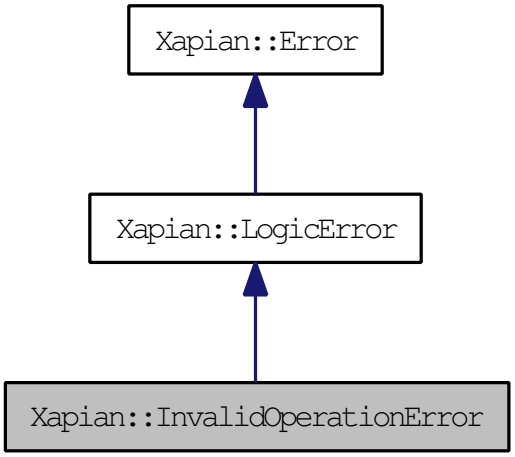
The documentation for this class was generated from the following file:

- xapian/error.h

7.29 Xapian::InvalidOperationError Class Reference

InvalidOperationError indicates the API was used in an invalid way.

Inheritance diagram for Xapian::InvalidOperationError:



Public Member Functions

- InvalidOperationError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- InvalidOperationError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.29.1 Detailed Description

InvalidOperationError indicates the API was used in an invalid way.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 Xapian::InvalidOperationError::InvalidOperationError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.29.2.2 Xapian::InvalidOperationError::InvalidOperationError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

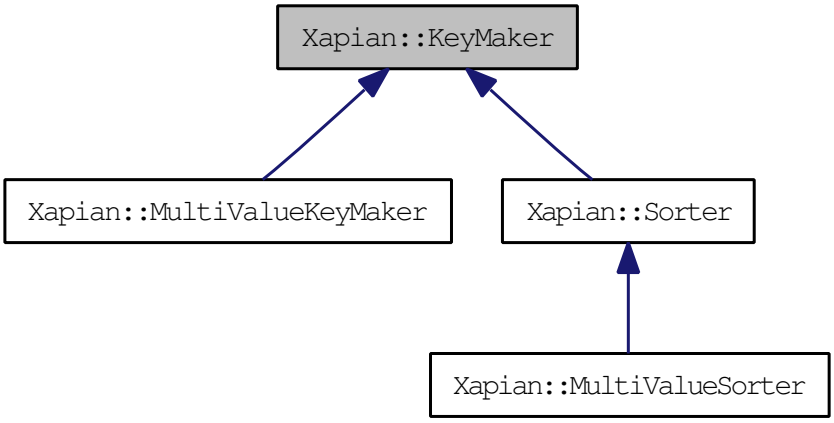
The documentation for this class was generated from the following file:

- xapian/error.h

7.30 Xapian::KeyMaker Class Reference

Virtual base class for key making functors.

Inheritance diagram for Xapian::KeyMaker:



Public Member Functions

- virtual std::string operator() (const Xapian::Document &doc) const =0

  Build a key string for a Document.

- virtual ~KeyMaker ()

  Virtual destructor, because we have virtual methods.

7.30.1 Detailed Description

Virtual base class for key making functors.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 virtual Xapian::KeyMaker::~KeyMaker () [virtual]

Virtual destructor, because we have virtual methods.

7.30.3 Member Function Documentation

7.30.3.1 virtual std::string Xapian::KeyMaker::operator() (const Xapian::Document & *doc*) const [pure virtual]

Build a key string for a Document.

These keys can be used for sorting or collapsing matching documents.

Parameters:

doc Document object to build a key for.

Implemented in Xapian::MultiValueKeyMaker, and Xapian::MultiValueSorter.

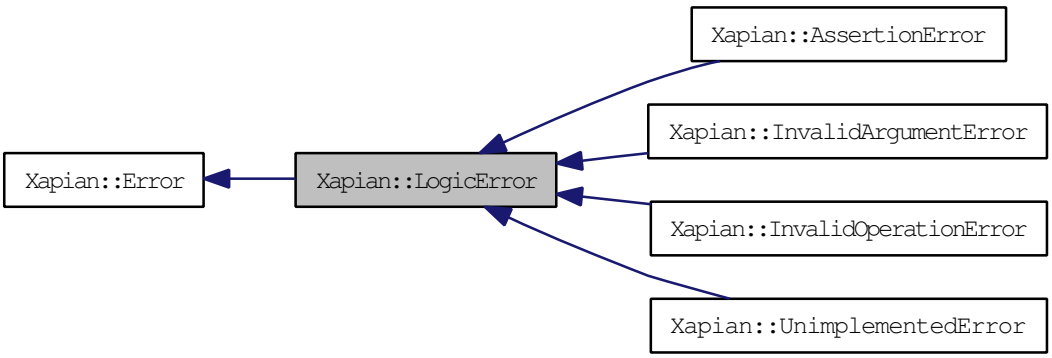
The documentation for this class was generated from the following file:

- xapian/keymaker.h

7.31 Xapian::LogicError Class Reference

The base class for exceptions indicating errors in the program logic.

Inheritance diagram for Xapian::LogicError:



7.31.1 Detailed Description

The base class for exceptions indicating errors in the program logic.

A subclass of LogicError will be thrown if Xapian detects a violation of a class invariant or a logical precondition or postcondition, etc.

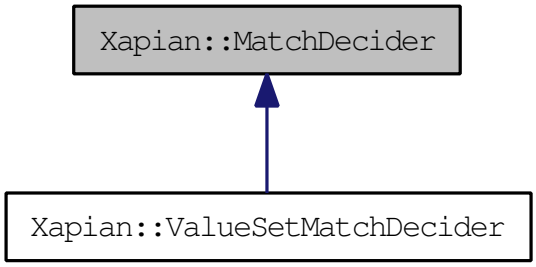
The documentation for this class was generated from the following file:

- xapian/error.h

7.32 Xapian::MatchDecider Class Reference

Base class for matcher decision functor.

Inheritance diagram for Xapian::MatchDecider:



Public Member Functions

- virtual bool operator() (const Xapian::Document &doc) const =0

  Decide whether we want this document to be in the MSet.

- virtual ~MatchDecider ()

  Destructor.

7.32.1 Detailed Description

Base class for matcher decision functor.

7.32.2 Member Function Documentation

7.32.2.1 virtual bool Xapian::MatchDecider::operator() (const Xapian::Document & *doc*) const [pure virtual]

Decide whether we want this document to be in the MSet.

Parameters:

doc The document to test.

Returns:

true if the document is acceptable, or false if the document should be excluded from the MSet.

Implemented in Xapian::ValueSetMatchDecider.

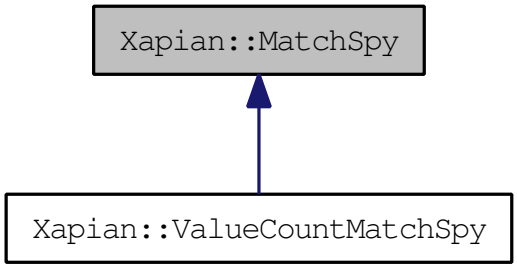
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.33 Xapian::MatchSpy Class Reference

Abstract base class for match spies.

Inheritance diagram for Xapian::MatchSpy:



Public Member Functions

- virtual ~MatchSpy ()

  Virtual destructor, because we have virtual methods.

- virtual void operator() (const Xapian::Document &doc, Xapian::weight wt)=0

  Register a document with the match spy.

- virtual MatchSpy * clone () const

  Clone the match spy.

- virtual std::string name () const

  Return the name of this match spy.

- virtual std::string serialise () const

  Return this object's parameters serialised as a single string.

- virtual MatchSpy * unserialise (const std::string &s, const Registry &context) const

  Unserialise parameters.

- virtual std::string serialise_results () const

  Serialise the results of this match spy.

- virtual void merge_results (const std::string &s)

  Unserialise some results, and merge them into this matchspy.

- virtual std::string get_description () const

  Return a string describing this object.

Protected Member Functions

- MatchSpy ()

  Default constructor, needed by subclass constructors.

7.33.1 Detailed Description

Abstract base class for match spies.

The subclasses will generally accumulate information seen during the match, to calculate aggregate functions, or other profiles of the matching documents.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 virtual Xapian::MatchSpy::~MatchSpy () [virtual]

Virtual destructor, because we have virtual methods.

7.33.3 Member Function Documentation

7.33.3.1 virtual MatchSpy* Xapian::MatchSpy::clone () const [virtual]

Clone the match spy.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be passed information about the results seen by the parent.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented in Xapian::ValueCountMatchSpy.

7.33.3.2 virtual std::string Xapian::MatchSpy::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer, to avoid forcing those deriving their own MatchSpy subclasses from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented in Xapian::ValueCountMatchSpy.

7.33.3.3 virtual void Xapian::MatchSpy::merge_results (const std::string & *s*) [virtual]

Unserialise some results, and merge them into this matchspy.

The order in which results are merged should not be significant, since this order is not specified (and will vary depending on the speed of the search in each sub-database).

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters:

s A string containing the serialised results.

Reimplemented in Xapian::ValueCountMatchSpy.

7.33.3.4 virtual std::string Xapian::MatchSpy::name () const [virtual]

Return the name of this match spy.

This name is used by the remote backend. It is passed with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called MyApp::FooMatchSpy, return "MyApp::FooMatchSpy" from this method.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented in Xapian::ValueCountMatchSpy.

7.33.3.5 virtual void Xapian::MatchSpy::operator() (const Xapian::Document & *doc*, Xapian::weight *wt*) [pure virtual]

Register a document with the match spy.

This is called by the matcher once with each document seen by the matcher during the match process. Note that the matcher will often not see all the documents which match the query, due to optimisations which allow low-weighted documents to be skipped, and allow the match process to be terminated early.

Parameters:

doc The document seen by the match spy.

wt The weight of the document.

Implemented in Xapian::ValueCountMatchSpy.

7.33.3.6 virtual std::string Xapian::MatchSpy::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented in Xapian::ValueCountMatchSpy.

7.33.3.7 virtual std::string Xapian::MatchSpy::serialise_results () const [virtual]

Serialise the results of this match spy.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented in Xapian::ValueCountMatchSpy.

7.33.3.8 virtual MatchSpy* Xapian::MatchSpy::unserialise (const std::string & *s*, const Registry & *context*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Parameters:

s A string containing the serialised results.

context Registry object to use for unserialisation to permit MatchSpy subclasses with sub-MatchSpy objects to be implemented.

Reimplemented in Xapian::ValueCountMatchSpy.

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

- xapian/matchspy.h

7.34 Xapian::MSet Class Reference

A match set (MSet).

Public Types

- typedef MSetIterator value_type
  Allow use as an STL container.
- typedef MSetIterator iterator
  Allow use as an STL container.
- typedef MSetIterator const_iterator
  Allow use as an STL container.
- typedef MSetIterator & reference
  Allow use as an STL container.
- typedef MSetIterator & const_reference
  Allow use as an STL container.
- typedef MSetIterator * pointer
  Allow use as an STL container.
- typedef Xapian::doccount_diff difference_type
  Allow use as an STL container.
- typedef Xapian::doccount size_type
  Allow use as an STL container.

Public Member Functions

- MSet ()
  Create an empty Xapian::MSet.
- ~MSet ()
  Destroy a Xapian::MSet.
- MSet (const MSet &other)
  Copying is allowed (and is cheap).
- void operator= (const MSet &other)
  Assignment is allowed (and is cheap).
- void fetch (const MSetIterator &begin, const MSetIterator &end) const

Fetch the document info for a set of items in the MSet.

- void fetch (const MSetIterator &item) const

  Fetch the single item specified.

- void fetch () const

  Fetch all the items in the MSet.

- Xapian::percent convert_to_percent (Xapian::weight wt) const

  This converts the weight supplied to a percentage score.

- Xapian::percent convert_to_percent (const MSetIterator &it) const

  Return the percentage score for a particular item.

- Xapian::doccount get_termfreq (const std::string &tname) const

  Return the term frequency of the given query term.

- Xapian::weight get_termweight (const std::string &tname) const

  Return the term weight of the given query term.

- Xapian::doccount get_firstitem () const

  The index of the first item in the result which was put into the MSet.

- Xapian::doccount get_matches_lower_bound () const

  A lower bound on the number of documents in the database which match the query.

- Xapian::doccount get_matches_estimated () const

  An estimate for the number of documents in the database which match the query.

- Xapian::doccount get_matches_upper_bound () const

  An upper bound on the number of documents in the database which match the query.

- Xapian::doccount get_uncollapsed_matches_lower_bound () const

  A lower bound on the number of documents in the database which would match the query if collapsing wasn't used.

- Xapian::doccount get_uncollapsed_matches_estimated () const

  A estimate of the number of documents in the database which would match the query if collapsing wasn't used.

- Xapian::doccount get_uncollapsed_matches_upper_bound () const

  A upper bound on the number of documents in the database which would match the query if collapsing wasn't used.

- Xapian::weight get_max_possible () const

  The maximum possible weight in the MSet.

- Xapian::weight get_max_attained () const

  The greatest weight which is attained by any document in the database.

- Xapian::doccount size () const

  The number of items in this MSet.

- Xapian::doccount max_size () const

  Required to allow use as an STL container.

- bool empty () const

  Test if this MSet is empty.

- void swap (MSet &other)

  Swap the MSet we point to with another.

- MSetIterator begin () const

  Iterator for the items in this MSet.

- MSetIterator end () const

  End iterator corresponding to begin().

- MSetIterator back () const

  Iterator pointing to the last element of this MSet.

- MSetIterator operator[ ] (Xapian::doccount i) const

  This returns the document at position i in this MSet object.

- std::string get_description () const

  Return a string describing this object.

7.34.1 Detailed Description

A match set (MSet).

This class represents (a portion of) the results of a query.

7.34.2 Member Function Documentation

7.34.2.1 Xapian::percent Xapian::MSet::convert_to_percent (Xapian::weight *wt*) const

This converts the weight supplied to a percentage score.

The return value will be in the range 0 to 100, and will be 0 if and only if the item did not match the query at all.

Parameters:

wt The weight to convert.

7.34.2.2 void Xapian::MSet::fetch (const MSetIterator & *begin*, const MSetIterator & *end*) const

Fetch the document info for a set of items in the MSet.

This method causes the documents in the range specified by the iterators to be fetched from the database, and cached in the Xapian::MSet object. This has little effect when performing a search across a local database, but will greatly speed up subsequent access to the document contents when the documents are stored in a remote database.

The iterators must be over this Xapian::MSet - undefined behaviour will result otherwise.

Parameters:

begin MSetIterator for first item to fetch.

end MSetIterator for item after last item to fetch.

7.34.2.3 Xapian::doccount Xapian::MSet::get_firstitem () const

The index of the first item in the result which was put into the MSet.

This corresponds to the parameter "first" specified in Xapian::Enquire::get_mset(). A value of 0 corresponds to the highest result being the first item in the MSet.

7.34.2.4 Xapian::doccount Xapian::MSet::get_matches_estimated () const

An estimate for the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This value is returned because there is sometimes a request to display such information. However, our experience is that presenting this value to users causes them to worry about the large number of results, rather than how useful those at the top of the result set are, and is thus undesirable.

7.34.2.5 Xapian::doccount Xapian::MSet::get_matches_lower_bound () const

A lower bound on the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This number is usually considerably less than the actual number of documents which match the query.

7.34.2.6 Xapian::doccount Xapian::MSet::get_matches_upper_bound () const

An upper bound on the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This number is usually considerably greater than the actual number of documents which match the query.

7.34.2.7 Xapian::weight Xapian::MSet::get_max_attained () const

The greatest weight which is attained by any document in the database.

If firstitem == 0 and the primary ordering is by relevance, this is the weight of the first entry in the MSet.

If no documents are found by the query, this will be 0.

Note that calculation of max_attained requires calculation of at least one result item - therefore, if no items were requested when the query was performed (by specifying maxitems = 0 in Xapian::Enquire::get_mset()), this value will be 0.

7.34.2.8 Xapian::weight Xapian::MSet::get_max_possible () const

The maximum possible weight in the MSet.

This weight is likely not to be attained in the set of results, but represents an upper bound on the weight which a document could attain for the given query.

7.34.2.9 Xapian::doccount Xapian::MSet::get_termfreq (const std::string & *tname*) const

Return the term frequency of the given query term.

Parameters:

tname The term to look for.

This is sometimes more efficient than asking the database directly for the term frequency - in particular, if the term was in the query, its frequency will usually be cached in the MSet.

7.34.2.10 Xapian::weight Xapian::MSet::get_termweight (const std::string & *tname*) const

Return the term weight of the given query term.

Parameters:

tname The term to look for.

Exceptions:

Xapian::InvalidArgumentError is thrown if the term was not in the query.

7.34.2.11 Xapian::doccount Xapian::MSet::max_size () const [inline]

Required to allow use as an STL container.

7.34.2.12 MSetIterator Xapian::MSet::operator[ ] (Xapian::doccount *i*) const

This returns the document at position i in this MSet object.

Note that this is not the same as the document at rank i in the query, unless the "first" parameter to Xapian::Enquire::get_mset was 0. Rather, it is the document at rank i + first.

In other words, the offset is into the documents represented by this object, not into the set of documents matching the query.

Parameters:

i The index into the MSet.

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

- xapian/enquire.h

7.35 Xapian::MSetIterator Class Reference

An iterator pointing to items in an MSet.

Public Types

- typedef std::bidirectional_iterator_tag iterator_category
  Allow use as an STL iterator.
- typedef Xapian::docid value_type
  Allow use as an STL iterator.
- typedef Xapian::doccount_diff difference_type
  Allow use as an STL iterator.
- typedef Xapian::docid * pointer
  Allow use as an STL iterator.
- typedef Xapian::docid & reference
  Allow use as an STL iterator.

Public Member Functions

- MSetIterator ()
  Create an uninitialised iterator; this cannot be used, but is convenient syntactically.
- MSetIterator (const MSetIterator &other)
  Copying is allowed (and is cheap).
- void operator= (const MSetIterator &other)
  Assignment is allowed (and is cheap).
- MSetIterator & operator++ ()
  Advance the iterator.
- MSetIterator operator++ (int)
  Advance the iterator (postfix variant).
- MSetIterator & operator– ()
  Decrement the iterator.
- MSetIterator operator– (int)
  Decrement the iterator (postfix variant).
- Xapian::docid operator* () const

Get the document ID for the current position.

- Xapian::Document get_document () const
  Get a Xapian::Document object for the current position.
- Xapian::doccount get_rank () const
  Get the rank of the document at the current position.
- Xapian::weight get_weight () const
  Get the weight of the document at the current position.
- std::string get_collapse_key () const
  Get the collapse key for this document.
- Xapian::doccount get_collapse_count () const
  Get an estimate of the number of documents that have been collapsed into this one.
- Xapian::percent get_percent () const
  This returns the weight of the document as a percentage score.
- std::string get_description () const
  Return a string describing this object.

Friends

- bool operator== (const MSetIterator &a, const MSetIterator &b)
  Equality test for MSetIterator objects.
- bool operator!= (const MSetIterator &a, const MSetIterator &b)
  Inequality test for MSetIterator objects.

7.35.1 Detailed Description

An iterator pointing to items in an MSet.

This is used for access to individual results of a match.

7.35.2 Member Function Documentation

7.35.2.1 Xapian::doccount Xapian::MSetIterator::get_collapse_count () const

Get an estimate of the number of documents that have been collapsed into this one.

The estimate will always be less than or equal to the actual number of other documents satisfying the match criteria with the same collapse key as this document.

This method may return 0 even though there are other documents with the same collapse key which satisfying the match criteria. However if this method returns non-zero, there definitely are other such documents. So this method may be used to inform the user that there are "at least N other matches in this group", or to control whether to offer a "show other documents in this group" feature (but note that it may not offer it in every case where it would show other documents).

7.35.2.2 Xapian::Document Xapian::MSetIterator::get_document () const

Get a Xapian::Document object for the current position.

This method returns a Xapian::Document object which provides the information about the document pointed to by the MSetIterator.

If the underlying database has suitable support, using this call (rather than asking the database for a document based on its document ID) will enable the system to ensure that the correct data is returned, and that the document has not been deleted or changed since the query was performed.

Returns:

A Xapian::Document object containing the document data.

Exceptions:

Xapian::DocNotFoundError The document specified could not be found in the database.

7.35.2.3 Xapian::percent Xapian::MSetIterator::get_percent () const

This returns the weight of the document as a percentage score.

The return value will be an integer in the range 0 to 100: 0 meaning that the item did not match the query at all.

The intention is that the highest weighted document will get 100 if it matches all the weight-contributing terms in the query. However, currently it may get a lower percentage score if you use a MatchDecider and the sorting is primarily by value. In this case, the percentage for a particular document may vary depending on the first, max_size, and checkatleast parameters passed to Enquire::get_mset() (this bug is hard to fix without having to apply the MatchDecider to potentially many more documents, which is potentially costly).

7.35.2.4 Xapian::doccount Xapian::MSetIterator::get_rank () const [inline]

Get the rank of the document at the current position.

The rank is the position that this document is at in the ordered list of results of the query. The result is 0-based - i.e. the top-ranked document has a rank of 0.

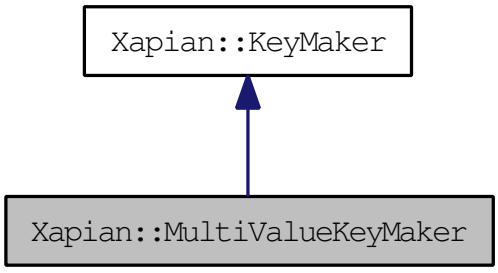
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.36 Xapian::MultiValueKeyMaker Class Reference

KeyMaker subclass which combines several values.

Inheritance diagram for Xapian::MultiValueKeyMaker:



Public Member Functions

- virtual std::string operator() (const Xapian::Document &doc) const
  Build a key string for a Document.

7.36.1 Detailed Description

KeyMaker subclass which combines several values.

When the result is used for sorting, results are ordered by the first value. In the event of a tie, the second is used. If this is the same for both, the third is used, and so on. If *reverse* is true for a value, then the sort order for that value is reversed.

When used for collapsing, the documents will only be considered equal if all the values specified match. If none of the specified values are set then the generated key will be empty, so such documents won't be collapsed (which is consistent with the behaviour in the "collapse on a value" case). If you'd prefer that documents with none of the keys set are collapsed together, then you can set *reverse* for at least one of the values. Other than this, it isn't useful to set *reverse* for collapsing.

7.36.2 Member Function Documentation

7.36.2.1 virtual std::string Xapian::MultiValueKeyMaker::operator() (const Xapian::Document & *doc*) const `[virtual]`

Build a key string for a Document.

These keys can be used for sorting or collapsing matching documents.

Parameters:

doc Document object to build a key for.

Implements Xapian::KeyMaker.

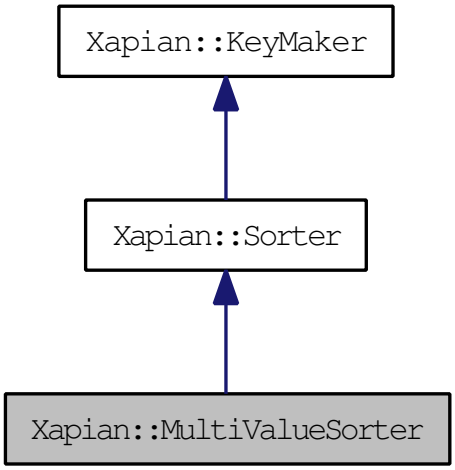
The documentation for this class was generated from the following file:

- xapian/keymaker.h

7.37 Xapian::MultiValueSorter Class Reference

Sorter subclass which sorts by a several values.

Inheritance diagram for Xapian::MultiValueSorter:



Public Member Functions

- virtual std::string operator() (const Xapian::Document &doc) const

  Build a key string for a Document.

7.37.1 Detailed Description

Sorter subclass which sorts by a several values.

Results are ordered by the first value. In the event of a tie, the second is used. If this is the same for both, the third is used, and so on.

Deprecated

> This class is deprecated - you should migrate to using MultiValueKeyMaker instead. Note that MultiValueSorter::add() becomes MultiValueKeyMaker::add_value(), but the sense of the direction flag is reversed (to be consistent with Enquire::set_sort_by_value()), so:

MultiValueSorter sorter; // Primary ordering is forwards on value 4. sorter.add(4); // Secondary ordering is reverse on value 5. sorter.add(5, false);

becomes:

MultiValueKeyMaker sorter; // Primary ordering is forwards on value 4. sorter.add_value(4); // Secondary ordering is reverse on value 5. sorter.add_value(5, true);

7.37.2 Member Function Documentation

7.37.2.1 virtual std::string Xapian::MultiValueSorter::operator() (const Xapian::Document & *doc*) const [virtual]

Build a key string for a Document.

These keys can be used for sorting or collapsing matching documents.

Parameters:

doc Document object to build a key for.

Implements Xapian::KeyMaker.

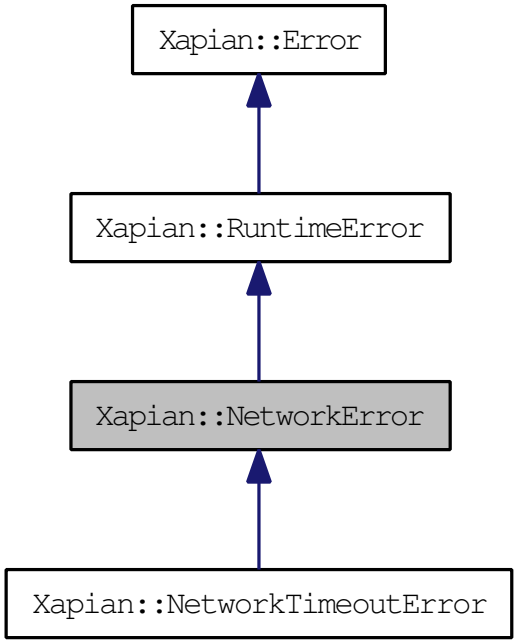
The documentation for this class was generated from the following file:

- xapian/keymaker.h

7.38 Xapian::NetworkError Class Reference

Indicates a problem communicating with a remote database.

Inheritance diagram for Xapian::NetworkError:



Public Member Functions

- NetworkError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- NetworkError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.38.1 Detailed Description

Indicates a problem communicating with a remote database.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 Xapian::NetworkError::NetworkError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.38.2.2 Xapian::NetworkError::NetworkError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

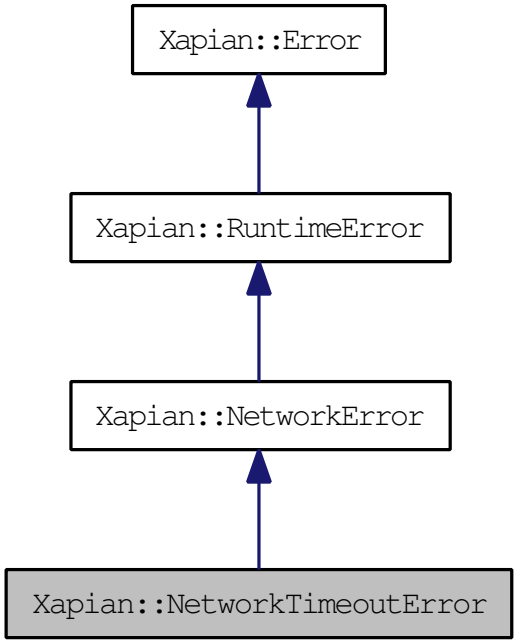
The documentation for this class was generated from the following file:

- xapian/error.h

7.39 Xapian::NetworkTimeoutError Class Reference

Indicates a timeout expired while communicating with a remote database.

Inheritance diagram for Xapian::NetworkTimeoutError:



Public Member Functions

- NetworkTimeoutError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- NetworkTimeoutError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.39.1 Detailed Description

Indicates a timeout expired while communicating with a remote database.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 Xapian::NetworkTimeoutError::NetworkTimeoutError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.39.2.2 Xapian::NetworkTimeoutError::NetworkTimeoutError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

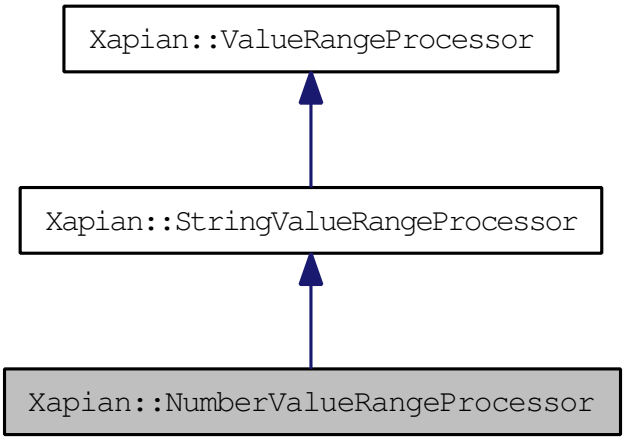
The documentation for this class was generated from the following file:

- xapian/error.h

7.40 Xapian::NumberValueRangeProcessor Class Reference

Handle a number range.

Inheritance diagram for Xapian::NumberValueRangeProcessor:



Public Member Functions

- NumberValueRangeProcessor (Xapian::valueno slot_)

  Constructor.

- NumberValueRangeProcessor (Xapian::valueno slot_, const std::string &str_-, bool prefix_=true)

  Constructor.

- Xapian::valueno operator() (std::string &begin, std::string &end)

  Check for a valid numeric range.

7.40.1 Detailed Description

Handle a number range.

This class must be used on values which have been encoded using Xapian::sortable_-serialise() which turns numbers into strings which will sort in the same order as the numbers (the same values can be used to implement a numeric sort).

7.40.2 Constructor & Destructor Documentation

7.40.2.1 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno *slot_*) `[inline]`

Constructor.

Parameters:

*slot_* The value number to return from operator().

7.40.2.2 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno *slot_*, const std::string & *str_*, bool *prefix_* = `true`) `[inline]`

Constructor.

Parameters:

*slot_* The value number to return from operator().

*str_* A string to look for to recognise values as belonging to this numeric range.

*prefix_* Whether to look for the string at the start or end of the values. If true, the string is a prefix; if false, the string is a suffix (default: true).

The string supplied in str_ is used by *operator()* to decide whether the pair of strings supplied to it constitute a valid range. If prefix_ is true, the first value in a range must begin with str_ (and the second value may optionally begin with str_); if prefix_ is false, the second value in a range must end with str_ (and the first value may optionally end with str_).

If str_ is empty, the setting of prefix_ is irrelevant, and no special strings are required at the start or end of the strings defining the range.

The remainder of both strings defining the endpoints must be valid floating point numbers. (FIXME: define format recognised).

For example, if str_ is "$" and prefix_ is true, and the range processor has been added to the queryparser, the queryparser will accept "$10..50" or "$10..$50", but not "10..50" or "10..$50" as valid ranges. If str_ is "kg" and prefix_ is false, the queryparser will accept "10..50kg" or "10kg..50kg", but not "10..50" or "10kg..50" as valid ranges.

7.40.3 Member Function Documentation

7.40.3.1 Xapian::valueno Xapian::NumberValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) `[virtual]`

Check for a valid numeric range.

Parameters:

↔ ***begin*** The start of the range as specified in the query string by the user. This parameter is a non-const reference so the ValueRangeProcessor can modify it to return the value to start the range with.

↔ ***end*** The end of the range. This is also a non-const reference so it can be modified.

Returns:

If BEGIN..END is a valid numeric range with the specified prefix/suffix (if one was specified), this method modifies them by removing the prefix/suffix, converting to a number, and encoding with Xapian::sortable_serialise(), and returns the value of slot_ passed at construction time. Otherwise it returns Xapian::BAD_VALUENO.

Reimplemented from Xapian::StringValueRangeProcessor.

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

- xapian/queryparser.h

7.41 Xapian::PositionIterator Class Reference

An iterator pointing to items in a list of positions.

Public Member Functions

- PositionIterator ()

  Default constructor - for declaring an uninitialised iterator.

- ~PositionIterator ()

  Destructor.

- PositionIterator (const PositionIterator &o)

  Copying is allowed.

- void operator= (const PositionIterator &o)

  Assignment is allowed.

- void skip_to (Xapian::termpos pos)

  Advance the iterator to the specified termpos.

- std::string get_description () const

  Return a string describing this object.

Friends

- bool operator== (const PositionIterator &a, const PositionIterator &b)

  Test equality of two PositionIterators.

7.41.1 Detailed Description

An iterator pointing to items in a list of positions.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 Xapian::PositionIterator::PositionIterator (const PositionIterator & *o*)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

7.41.3 Member Function Documentation

7.41.3.1 void Xapian::PositionIterator::operator= (const PositionIterator & *o*)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

7.41.3.2 void Xapian::PositionIterator::skip_to (Xapian::termpos *pos*)

Advance the iterator to the specified termpos.

If the specified termpos isn't in the list, position ourselves on the first termpos after it (or at_end() if no greater term positions are present).

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

- xapian/positioniterator.h

7.42 Xapian::PostingIterator Class Reference

An iterator pointing to items in a list of postings.

Public Types

- typedef std::input_iterator_tag iterator_category
  Allow use as an STL iterator.
- typedef Xapian::docid value_type
  Allow use as an STL iterator.
- typedef Xapian::doccount_diff difference_type
  Allow use as an STL iterator.
- typedef Xapian::docid * pointer
  Allow use as an STL iterator.
- typedef Xapian::docid & reference
  Allow use as an STL iterator.

Public Member Functions

- PostingIterator ()
  Default constructor - for declaring an uninitialised iterator.
- ~PostingIterator ()
  Destructor.
- PostingIterator (const PostingIterator &other)
  Copying is allowed.
- void operator= (const PostingIterator &other)
  Assignment is allowed.
- void skip_to (Xapian::docid did)
  Advance the iterator to the specified docid.
- Xapian::docid operator* () const
  Get the document id at the current position in the postlist.
- Xapian::termcount get_doclength () const
  Get the length of the document at the current position in the postlist.
- Xapian::termcount get_wdf () const

Get the within document frequency of the document at the current position in the postlist.

- PositionIterator positionlist_begin () const

  Return PositionIterator pointing to start of positionlist for current document.

- PositionIterator positionlist_end () const

  Return PositionIterator pointing to end of positionlist for current document.

- std::string get_description () const

  Return a string describing this object.

Friends

- bool operator== (const PostingIterator &a, const PostingIterator &b)

  Test equality of two PostingIterators.

7.42.1 Detailed Description

An iterator pointing to items in a list of postings.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 Xapian::PostingIterator::PostingIterator (const PostingIterator & *other*)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

7.42.3 Member Function Documentation

7.42.3.1 Xapian::termcount Xapian::PostingIterator::get_doclength () const

Get the length of the document at the current position in the postlist.

This information may be stored in the postlist, in which case this lookup should be extremely fast (indeed, not require further disk access). If the information is not present in the postlist, it will be retrieved from the database, at a greater performance cost.

7.42.3.2 void Xapian::PostingIterator::operator= (const PostingIterator & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

7.42.3.3 void Xapian::PostingIterator::skip_to (Xapian::docid *did*)

Advance the iterator to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at_end() if no greater docids are present).

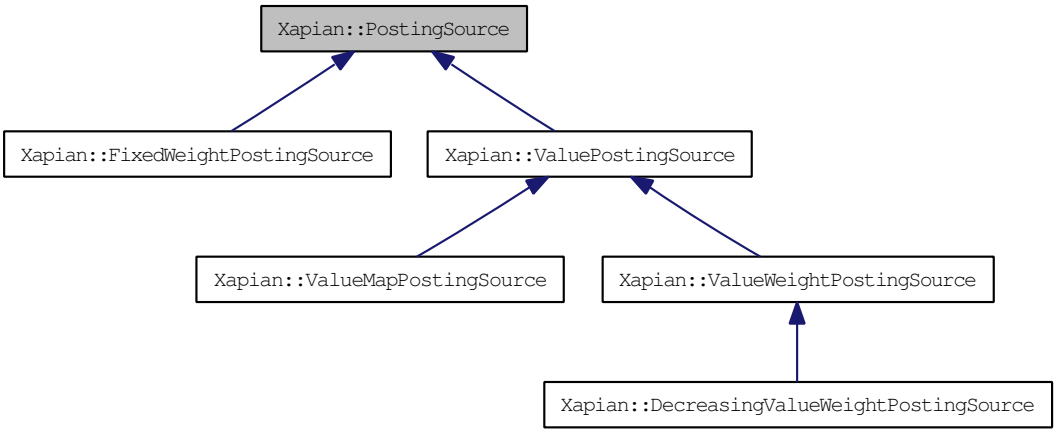
The documentation for this class was generated from the following file:

- xapian/postingiterator.h

7.43 Xapian::PostingSource Class Reference

Base class which provides an "external" source of postings.

Inheritance diagram for Xapian::PostingSource:



Public Member Functions

- virtual Xapian::doccount get_termfreq_min () const =0

  A lower bound on the number of documents this object can return.

- virtual Xapian::doccount get_termfreq_est () const =0

  An estimate of the number of documents this object can return.

- virtual Xapian::doccount get_termfreq_max () const =0

  An upper bound on the number of documents this object can return.

- Xapian::weight get_maxweight () const

  Return the currently set upper bound on what get_weight() can return.

- virtual Xapian::weight get_weight () const

  Return the weight contribution for the current document.

- virtual Xapian::docid get_docid () const =0

  Return the current docid.

- virtual void next (Xapian::weight min_wt)=0

  Advance the current position to the next matching document.

- virtual void skip_to (Xapian::docid did, Xapian::weight min_wt)

  Advance to the specified docid.

- virtual bool check (Xapian::docid did, Xapian::weight min_wt)

  Check if the specified docid occurs.

- virtual bool at_end () const =0

  Return true if the current position is past the last entry in this list.

- virtual PostingSource * clone () const

  Clone the posting source.

- virtual std::string name () const

  Name of the posting source class.

- virtual std::string serialise () const

  Serialise object parameters into a string.

- virtual PostingSource * unserialise (const std::string &s) const

  Create object given string serialisation returned by serialise().

- virtual void init (const Database &db)=0

  Set this PostingSource to the start of the list of postings.

- virtual std::string get_description () const

  Return a string describing this object.

Protected Member Functions

- PostingSource ()

  Allow subclasses to be instantiated.

- void set_maxweight (Xapian::weight max_weight)

  Set an upper bound on what get_weight() can return from now on.

7.43.1 Detailed Description

Base class which provides an "external" source of postings.

7.43.2 Member Function Documentation

7.43.2.1 virtual bool Xapian::PostingSource::at_end () const [pure virtual]

Return true if the current position is past the last entry in this list.

At least one of *next()*, *skip_to()* or *check()* will be called before this method is first called.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.2 virtual bool Xapian::PostingSource::check (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after *did*.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn't.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

did The document id to check.

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented in Xapian::ValuePostingSource, Xapian::DecreasingValueWeightPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.3 virtual PostingSource* Xapian::PostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.4 virtual std::string Xapian::PostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.5 virtual Xapian::docid Xapian::PostingSource::get_docid () const [pure virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document". See *get_weight()* for details.

Note: in the case of a multi-database search, the returned docid should be in the single subdatabase relevant to this posting source. See the *init()* method for details.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.6 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_est () const [pure virtual]

An estimate of the number of documents this object can return.

It must always be true that:

get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()

Xapian will always call init() on a PostingSource before calling this for the first time.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.7 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_max () const [pure virtual]

An upper bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.8 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_min () const [pure virtual]

A lower bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.9 virtual Xapian::weight Xapian::PostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.10 virtual void Xapian::PostingSource::init (const Database & *db*) [pure virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Implemented in Xapian::ValuePostingSource, Xapian::ValueWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.11 virtual std::string Xapian::PostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.12 virtual void Xapian::PostingSource::next (Xapian::weight *min_wt*) [pure virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Implemented in Xapian::ValuePostingSource, Xapian::DecreasingValueWeightPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.13 virtual std::string Xapian::PostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.14 void Xapian::PostingSource::set_maxweight (Xapian::weight *max_weight*) [protected]

Set an upper bound on what get_weight() can return from now on.

This upper bound is used by the matcher to perform various optimisations, so if you can return a good bound, then matches will generally run faster.

This method should be called after calling init(), and may be called during iteration if the upper bound drops.

It is valid for the posting source to have returned a higher value from get_weight() earlier in the iteration, but the posting source must not return a higher value from get_weight() than the currently set upper bound, and the upper bound must not be increased (until init() has been called).

If you don't call this method, the upper bound will default to 0, for convenience when implementing "weight-less" PostingSource subclasses.

Parameters:

max_weight The upper bound to set.

7.43.2.15 virtual void Xapian::PostingSource::skip_to (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Advance to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at_end() if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

did The document id to advance to.

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented in Xapian::ValuePostingSource, Xapian::DecreasingValueWeightPostingSource, and Xapian::FixedWeightPostingSource.

7.43.2.16 virtual PostingSource* Xapian::PostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

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

- xapian/postingsource.h

7.44 Xapian::Query Class Reference

Class representing a query.

Public Types

- enum op {
  OP_AND, OP_OR, OP_AND_NOT, OP_XOR,
  OP_AND_MAYBE, OP_FILTER, OP_NEAR, OP_PHRASE,
  OP_VALUE_RANGE, OP_SCALE_WEIGHT, OP_ELITE_SET, OP_-VALUE_GE,
  OP_VALUE_LE, OP_SYNONYM }

  Enum of possible query operations.

Public Member Functions

- Query (const Query &copyme)

  Copy constructor.

- Query & operator= (const Query &copyme)

  Assignment.

- Query ()

  Default constructor: makes an empty query which matches no documents.

- ~Query ()

  Destructor.

- Query (const std::string &tname_, Xapian::termcount wqf_=1, Xapian::termpos pos_=0)

  A query consisting of a single term.

- Query (Query::op op_, const Query &left, const Query &right)

  A query consisting of two subqueries, opp-ed together.

- Query (Query::op op_, const std::string &left, const std::string &right)

  A query consisting of two termnames opp-ed together.

- template<class Iterator >
  Query (Query::op op_, Iterator qbegin, Iterator qend, Xapian::termcount parameter=0)

  Combine a number of Xapian::Query-s with the specified operator.

- Query (Query::op op_, Xapian::Query q, double parameter)

Apply the specified operator to a single Xapian::Query object, with a double parameter.

- Query (Query::op op_, Xapian::valueno slot, const std::string &begin, const std::string &end)

  Construct a value range query on a document value.

- Query (Query::op op_, Xapian::valueno slot, const std::string &value)

  Construct a value comparison query on a document value.

- Query (Xapian::PostingSource *external_source)

  Construct an external source query.

- Xapian::termcount get_length () const

  Get the length of the query, used by some ranking formulae.

- TermIterator get_terms_begin () const

  Return a Xapian::TermIterator returning all the terms in the query, in order of termpos.

- TermIterator get_terms_end () const

  Return a Xapian::TermIterator to the end of the list of terms in the query.

- bool empty () const

  Test if the query is empty (i.e.

- std::string serialise () const

  Serialise query into a string.

- std::string get_description () const

  Return a string describing this object.

Static Public Member Functions

- static Query unserialise (const std::string &s)

  Unserialise a query from a string produced by serialise().

- static Query unserialise (const std::string &s, const Registry &registry)

  Unserialise a query from a string produced by serialise().

Static Public Attributes

- static const Xapian::Query MatchAll

  A query which matches all documents in the database.

- static const Xapian::Query MatchNothing

  A query which matches no documents.

7.44.1 Detailed Description

Class representing a query.

Queries are represented as a tree of objects.

7.44.2 Member Enumeration Documentation

7.44.2.1 enum Xapian::Query::op

Enum of possible query operations.

Enumerator:

OP_AND Return iff both subqueries are satisfied.

OP_OR Return if either subquery is satisfied.

OP_AND_NOT Return if left but not right satisfied.

OP_XOR Return if one query satisfied, but not both.

OP_AND_MAYBE Return iff left satisfied, but use weights from both.

OP_FILTER As AND, but use only weights from left subquery.

OP_NEAR Find occurrences of a list of terms with all the terms occurring within a specified window of positions.

Each occurrence of a term must be at a different position, but the order they appear in is irrelevant.

The window parameter should be specified for this operation, but will default to the number of terms in the list.

OP_PHRASE Find occurrences of a list of terms with all the terms occurring within a specified window of positions, and all the terms appearing in the order specified.

Each occurrence of a term must be at a different position.

The window parameter should be specified for this operation, but will default to the number of terms in the list.

OP_VALUE_RANGE Filter by a range test on a document value.

OP_SCALE_WEIGHT Scale the weight of a subquery by the specified factor.

A factor of 0 means this subquery will contribute no weight to the query - it will act as a purely boolean subquery.

If the factor is negative, Xapian::InvalidArgumentError will be thrown.

OP_ELITE_SET Select an elite set from the subqueries, and perform a query with these combined as an OR query.

OP_VALUE_GE Filter by a greater-than-or-equal test on a document value.

OP_VALUE_LE Filter by a less-than-or-equal test on a document value.

OP_SYNONYM Treat a set of queries as synonyms.

This returns all results which match at least one of the queries, but weighting as if all the sub-queries are instances of the same term: so multiple matching terms for a document increase the wdf value used, and the term frequency is based on the number of documents which would match an OR of all the subqueries.

The term frequency used will usually be an approximation, because calculating the precise combined term frequency would be overly expensive.

Identical to OP_OR, except for the weightings returned.

7.44.3 Constructor & Destructor Documentation

7.44.3.1 Xapian::Query::Query (const Query & *copyme*)

Copy constructor.

7.44.3.2 Xapian::Query::Query ()

Default constructor: makes an empty query which matches no documents.

Also useful for defining a Query object to be assigned to later.

An exception will be thrown if an attempt is made to use an undefined query when building up a composite query.

7.44.3.3 Xapian::Query::~Query ()

Destructor.

7.44.3.4 Xapian::Query::Query (const std::string & *tname_*, Xapian::termcount *wqf_* = 1, Xapian::termpos *pos_* = 0)

A query consisting of a single term.

7.44.3.5 Xapian::Query::Query (Query::op *op_*, const Query & *left*, const Query & *right*)

A query consisting of two subqueries, opp-ed together.

7.44.3.6 Xapian::Query::Query (Query::op *op_*, const std::string & *left*, const std::string & *right*)

A query consisting of two termnames opp-ed together.

7.44.3.7 template<class Iterator > Xapian::Query::Query (Query::op *op_*, Iterator *qbegin*, Iterator *qend*, Xapian::termcount *parameter* = 0) [inline]

Combine a number of Xapian::Query-s with the specified operator.

The Xapian::Query objects are specified with begin and end iterators.

AND, OR, SYNONYM, NEAR and PHRASE can take any number of subqueries. Other operators take exactly two subqueries.

The iterators may be to Xapian::Query objects, pointers to Xapian::Query objects, or termnames (std::string-s).

For NEAR and PHRASE, a window size can be specified in parameter.

For ELITE_SET, the elite set size can be specified in parameter.

7.44.3.8 Xapian::Query::Query (Query::op *op_*, Xapian::valueno *slot*, const std::string & *begin*, const std::string & *end*)

Construct a value range query on a document value.

A value range query matches those documents which have a value stored in the slot given by *slot* which is in the range specified by *begin* and *end* (in lexicographical order), including the endpoints.

Parameters:

- ***op_*** The operator to use for the query. Currently, must be OP_VALUE_RANGE.
- ***slot*** The slot number to get the value from.
- ***begin*** The start of the range.
- ***end*** The end of the range.

7.44.3.9 Xapian::Query::Query (Query::op *op_*, Xapian::valueno *slot*, const std::string & *value*)

Construct a value comparison query on a document value.

This query matches those documents which have a value stored in the slot given by *slot* which compares, as specified by the operator, to *value*.

Parameters:

- ***op_*** The operator to use for the query. Currently, must be OP_VALUE_GE or OP_VALUE_LE.
- ***slot*** The slot number to get the value from.
- ***value*** The value to compare.

7.44.3.10 Xapian::Query::Query (Xapian::PostingSource * *external_source*) [explicit]

Construct an external source query.

An attempt to clone the posting source will be made immediately, so if the posting source supports clone(), the source supplied may be safely deallocated after this call. If the source does not support clone(), the caller must ensure that the posting source remains valid until the Query is deallocated.

Parameters:

external_source The source to use in the query.

7.44.4 Member Function Documentation

7.44.4.1 bool Xapian::Query::empty () const

Test if the query is empty (i.e.

was constructed using the default ctor or with an empty iterator ctor).

7.44.4.2 Xapian::termcount Xapian::Query::get_length () const

Get the length of the query, used by some ranking formulae.

This value is calculated automatically - if you want to override it you can pass a different value to Enquire::set_query().

7.44.4.3 TermIterator Xapian::Query::get_terms_begin () const

Return a Xapian::TermIterator returning all the terms in the query, in order of termpos.

If multiple terms have the same term position, their order is unspecified. Duplicates (same term and termpos) will be removed.

7.44.4.4 Query& Xapian::Query::operator= (const Query & *copyme*)

Assignment.

7.44.4.5 std::string Xapian::Query::serialise () const

Serialise query into a string.

The query representation may change between Xapian releases: even between minor versions. However, it is guaranteed not to change unless the remote database protocol has also changed between releases.

7.44.4.6 static Query Xapian::Query::unserialise (const std::string & *s*, const Registry & *registry*) [static]

Unserialise a query from a string produced by serialise().

The supplied registry will be used to attempt to unserialise any external PostingSource leaf nodes. This method will fail if the query contains any external PostingSource leaf nodes which are not registered in the registry.

Parameters:

s The string representing the serialised query.

registry Xapian::Registry to use.

7.44.4.7 static Query Xapian::Query::unserialise (const std::string & *s*) [static]

Unserialise a query from a string produced by serialise().

This method will fail if the query contains any external PostingSource leaf nodes.

Parameters:

s The string representing the serialised query.

7.44.5 Member Data Documentation

7.44.5.1 const Xapian::Query Xapian::Query::MatchAll [static]

A query which matches all documents in the database.

7.44.5.2 const Xapian::Query Xapian::Query::MatchNothing [static]

A query which matches no documents.

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

- xapian/query.h

7.45 Xapian::QueryParser Class Reference

Build a Xapian::Query object from a user query string.

Public Types

- enum feature_flag {

  FLAG_BOOLEAN = 1, FLAG_PHRASE = 2, FLAG_LOVEHATE = 4, FLAG_BOOLEAN_ANY_CASE = 8,

  FLAG_WILDCARD = 16, FLAG_PURE_NOT = 32, FLAG_PARTIAL = 64, FLAG_SPELLING_CORRECTION = 128,

  FLAG_SYNONYM = 256, FLAG_AUTO_SYNONYMS = 512, FLAG_-AUTO_MULTIWORD_SYNONYMS = 1024 | FLAG_AUTO_SYNONYMS, FLAG_DEFAULT = FLAG_PHRASE|FLAG_BOOLEAN|FLAG_LOVEHATE }

  Enum of feature flags.

Public Member Functions

- QueryParser (const QueryParser &o)

  Copy constructor.

- QueryParser & operator= (const QueryParser &o)

  Assignment.

- QueryParser ()

  Default constructor.

- ~QueryParser ()

  Destructor.

- void set_stemmer (const Xapian::Stem &stemmer)

  Set the stemmer.

- void set_stemming_strategy (stem_strategy strategy)

  Set the stemming strategy.

- void set_stopper (const Stopper *stop=NULL)

  Set the stopper.

- void set_default_op (Query::op default_op)

  Set the default operator.

- Query::op get_default_op () const

Get the current default operator.

- void set_database (const Database &db)

  Specify the database being searched.

- void set_max_wildcard_expansion (Xapian::termcount limit)

  Specify the maximum expansion of a wildcard term.

- Query parse_query (const std::string &query_string, unsigned flags=FLAG_-DEFAULT, const std::string &default_prefix=std::string())

  Parse a query.

- void add_prefix (const std::string &field, const std::string &prefix)

  Add a probabilistic term prefix.

- void add_boolean_prefix (const std::string &field, const std::string &prefix, bool exclusive)

  Add a boolean term prefix allowing the user to restrict a search with a boolean filter specified in the free text query.

- TermIterator stoplist_begin () const

  Iterate over terms omitted from the query as stopwords.

- TermIterator unstem_begin (const std::string &term) const

  Iterate over unstemmed forms of the given (stemmed) term used in the query.

- void add_valuerangeprocessor (Xapian::ValueRangeProcessor *vrproc)

  Register a ValueRangeProcessor.

- std::string get_corrected_query_string () const

  Get the spelling-corrected query string.

- std::string get_description () const

  Return a string describing this object.

7.45.1 Detailed Description

Build a Xapian::Query object from a user query string.

7.45.2 Member Enumeration Documentation

7.45.2.1 enum Xapian::QueryParser::feature_flag

Enum of feature flags.

Enumerator:

FLAG_BOOLEAN Support AND, OR, etc and bracketed subexpressions.

FLAG_PHRASE Support quoted phrases.

FLAG_LOVEHATE Support + and -.

FLAG_BOOLEAN_ANY_CASE Support AND, OR, etc even if they aren't in ALLCAPS.

FLAG_WILDCARD Support right truncation (e.g.

Xap∗).

Currently you can't use wildcards with boolean filter prefixes, or in a phrase (either an explicitly quoted one, or one implicitly generated by hyphens or other punctuation).

NB: You need to tell the QueryParser object which database to expand wildcards from by calling set_database.

FLAG_PURE_NOT Allow queries such as 'NOT apples'.

These require the use of a list of all documents in the database which is potentially expensive, so this feature isn't enabled by default.

FLAG_PARTIAL Enable partial matching.

Partial matching causes the parser to treat the query as a "partially entered" search. This will automatically treat the final word as a wildcarded match, unless it is followed by whitespace, to produce more stable results from interactive searches.

Currently FLAG_PARTIAL doesn't do anything if the final word in the query has a boolean filter prefix, or if it is in a phrase (either an explicitly quoted one, or one implicitly generated by hyphens or other punctuation). It also doesn't do anything if if the final word is part of a value range.

NB: You need to tell the QueryParser object which database to expand wildcards from by calling set_database.

FLAG_SPELLING_CORRECTION Enable spelling correction.

For each word in the query which doesn't exist as a term in the database, Database::get_spelling_suggestion() will be called and if a suggestion is returned, a corrected version of the query string will be built up which can be read using QueryParser::get_corrected_query_string(). The query returned is based on the uncorrected query string however - if you want a parsed query based on the corrected query string, you must call QueryParser::parse_query() again.

NB: You must also call set_database() for this to work.

FLAG_SYNONYM Enable synonym operator '∼'.

NB: You must also call set_database() for this to work.

FLAG_AUTO_SYNONYMS Enable automatic use of synonyms for single terms.

NB: You must also call set_database() for this to work.

FLAG_AUTO_MULTIWORD_SYNONYMS Enable automatic use of synonyms for single terms and groups of terms.

NB: You must also call set_database() for this to work.

FLAG_DEFAULT The default flags.

Used if you don't explicitly pass any to *parse_query()*. The default flags are FLAG_PHRASE|FLAG_BOOLEAN|FLAG_LOVEHATE.

Added in Xapian 1.0.11.

7.45.3 Member Function Documentation

7.45.3.1 void Xapian::QueryParser::add_boolean_prefix (const std::string & *field*, const std::string & *prefix*, bool *exclusive*)

Add a boolean term prefix allowing the user to restrict a search with a boolean filter specified in the free text query.

For example:

```
qp.add_boolean_prefix("site", "H");
```

This allows the user to restrict a search with site:xapian.org which will be converted to Hxapian.org combined with any probabilistic query with `Xapian::Query::OP_FILTER`.

If multiple boolean filters are specified in a query for the same prefix, they will be combined with the `Xapian::Query::OP_OR` operator. Then, if there are boolean filters for different prefixes, they will be combined with the `Xapian::Query::OP_AND` operator.

Multiple fields can be mapped to the same prefix (so for example you can make site: and domain: aliases for each other). Instances of fields with different aliases but the same prefix will still be combined with the OR operator.

For example, if "site" and "domain" map to "H", but author maps to "A", a search for "site:foo domain:bar author:Fred" will map to "(Hfoo OR Hbar) AND Afred".

As of 1.0.4, you can call this method multiple times with the same value of field to allow a single field to be mapped to multiple prefixes. Multiple terms being generated for such a field, and combined with `Xapian::Query::OP_OR`.

Calling this method with an empty string for *field* will cause a `Xapian::InvalidArgumentError`.

If you call `add_prefix()` and `add_boolean_prefix()` for the same value of *field*, a `Xapian::InvalidOperationError` exception will be thrown.

In 1.0.3 and earlier, subsequent calls to this method with the same value of *field* had no effect.

Parameters:

field The user visible field name

prefix The term prefix to map this to

exclusive If true, each document can have at most one term with this prefix, so multiple filters with this prefix should be combined with OP_OR. If false,

each document can have multiple terms with this prefix, so multiple filters should be combined with OP_AND, like happens with filters with different prefixes. [default: true]

7.45.3.2 void Xapian::QueryParser::add_prefix (const std::string & *field*, const std::string & *prefix*)

Add a probabilistic term prefix.

For example:

```
qp.add_prefix("author", "A");
```

This allows the user to search for author:Orwell which will be converted to a search for the term "Aorwell".

Multiple fields can be mapped to the same prefix. For example, you can make title: and subject: aliases for each other.

As of 1.0.4, you can call this method multiple times with the same value of field to allow a single field to be mapped to multiple prefixes. Multiple terms being generated for such a field, and combined with `Xapian::Query::OP_OR`.

If any prefixes are specified for the empty field name (i.e. you call this method with an empty string as the first parameter) these prefixes will be used for terms without a field specifier. If you do this and also specify the `default_prefix` parameter to `parse_query()`, then the `default_prefix` parameter will override.

If the prefix parameter is empty, then "field:word" will produce the term "word" (and this can be one of several prefixes for a particular field, or for terms without a field specifier).

If you call `add_prefix()` and `add_boolean_prefix()` for the same value of *field*, a `Xapian::InvalidOperationError` exception will be thrown.

In 1.0.3 and earlier, subsequent calls to this method with the same value of *field* had no effect.

Parameters:

field The user visible field name

prefix The term prefix to map this to

7.45.3.3 std::string Xapian::QueryParser::get_corrected_query_string () const

Get the spelling-corrected query string.

This will only be set if FLAG_SPELLING_CORRECTION is specified when QueryParser::parse_query() was last called.

If there were no corrections, an empty string is returned.

7.45.3.4 Query::op Xapian::QueryParser::get_default_op () const

Get the current default operator.

7.45.3.5 Query Xapian::QueryParser::parse_query (const std::string & *query_string*, unsigned *flags* = `FLAG_DEFAULT`, const std::string & *default_prefix* = `std::string()`)

Parse a query.

Parameters:

query_string A free-text query as entered by a user

flags Zero or more Query::feature_flag specifying what features the QueryParser should support. Combine multiple values with bitwise-or (|) (default FLAG_DEFAULT).

default_prefix The default term prefix to use (default none). For example, you can pass "A" when parsing an "Author" field.

Exceptions:

If the query string can't be parsed, then Xapian::QueryParserError is thrown. You can get an English error message to report to the user by catching it and calling get_msg() on the caught exception. The current possible values (in case you want to translate them) are:

- Unknown range operation
- parse error
- Syntax: <expression> AND <expression>
- Syntax: <expression> AND NOT <expression>
- Syntax: <expression> NOT <expression>
- Syntax: <expression> OR <expression>
- Syntax: <expression> XOR <expression>

7.45.3.6 void Xapian::QueryParser::set_database (const Database & *db*)

Specify the database being searched.

Parameters:

db The database to use for wildcard expansion (FLAG_WILDCARD and FLAG_PARTIAL), spelling correction (FLAG_SPELLING_CORRECTION), and synonyms (FLAG_SYNONYM, FLAG_AUTO_SYNONYMS, and FLAG_AUTO_MULTIWORD_SYNONYMS).

7.45.3.7 void Xapian::QueryParser::set_default_op (Query::op *default_op*)

Set the default operator.

Parameters:

default_op The operator to use to combine non-filter query items when no explicit operator is used.

The most useful values for this are OP_OR (the default) and OP_AND. OP_NEAR and OP_PHRASE can also be useful.

So for example, 'weather forecast' is parsed as if it were 'weather OR forecast' by default.

7.45.3.8 void Xapian::QueryParser::set_max_wildcard_expansion (Xapian::termcount *limit*)

Specify the maximum expansion of a wildcard term.

Note: you must also set FLAG_WILDCARD for wildcard expansion to happen.

Parameters:

limit The maximum number of terms each wildcard in the query can expand to, or 0 for no limit (which is the default).

7.45.3.9 void Xapian::QueryParser::set_stemmer (const Xapian::Stem & *stemmer*)

Set the stemmer.

This sets the stemming algorithm which will be used by the query parser. Note that the stemming algorithm will only be used according to the stemming strategy set by set_stemming_strategy(), which defaults to STEM_NONE. Therefore, to use a stemming algorithm, you will also need to call set_stemming_strategy() with a value other than STEM_NONE.

Parameters:

stemmer The Xapian::Stem object to set.

7.45.3.10 void Xapian::QueryParser::set_stemming_strategy (stem_strategy *strategy*)

Set the stemming strategy.

This controls how the query parser will apply the stemming algorithm. Note that the stemming algorithm is only applied to words in probabilistic fields - boolean filter terms are never stemmed.

Parameters:

strategy The strategy to use - possible values are:

- STEM_NONE: Don't perform any stemming (the default).
- STEM_SOME: Search for stemmed forms of terms except for those which start with a capital letter, or are followed by certain characters (currently: (/@<>=*[{"), or are used with operators which need positional information. Stemmed terms are prefixed with 'Z'.
- STEM_ALL: Search for stemmed forms of all words (note: no 'Z' prefix is added).

7.45.3.11 void Xapian::QueryParser::set_stopper (const Stopper * *stop* = `NULL`)

Set the stopper.

Parameters:

stop The Stopper object to set (default NULL, which means no stopwords).

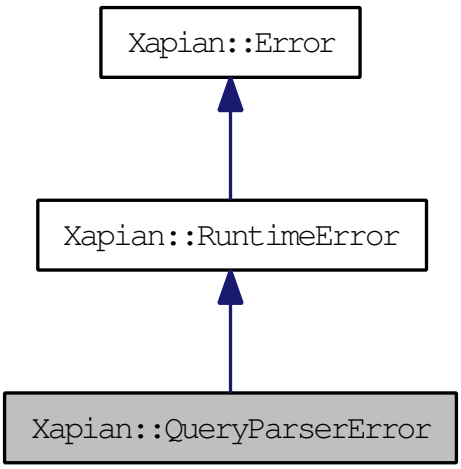
The documentation for this class was generated from the following file:

- xapian/queryparser.h

7.46 Xapian::QueryParserError Class Reference

Indicates a query string can't be parsed.

Inheritance diagram for Xapian::QueryParserError:



Public Member Functions

- QueryParserError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- QueryParserError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.46.1 Detailed Description

Indicates a query string can't be parsed.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 Xapian::QueryParserError::QueryParserError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.46.2.2 Xapian::QueryParserError::QueryParserError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

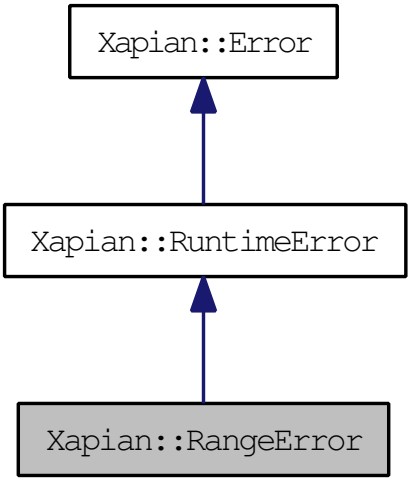
The documentation for this class was generated from the following file:

- xapian/error.h

7.47 Xapian::RangeError Class Reference

RangeError indicates an attempt to access outside the bounds of a container.

Inheritance diagram for Xapian::RangeError:



Public Member Functions

- RangeError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- RangeError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.47.1 Detailed Description

RangeError indicates an attempt to access outside the bounds of a container.

7.47.2 Constructor & Destructor Documentation

7.47.2.1 Xapian::RangeError::RangeError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.47.2.2 Xapian::RangeError::RangeError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

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

- xapian/error.h

7.48 Xapian::Registry Class Reference

Registry for user subclasses.

Public Member Functions

- Registry (const Registry &other)

  Copy constructor.

- Registry & operator= (const Registry &other)

  Assignment operator.

- Registry ()

  Default constructor.

- void register_weighting_scheme (const Xapian::Weight &wt)

  Register a weighting scheme.

- const Xapian::Weight * get_weighting_scheme (const std::string &name) const

  Get the weighting scheme given a name.

- void register_posting_source (const Xapian::PostingSource &source)

  Register a user-defined posting source class.

- const Xapian::PostingSource * get_posting_source (const std::string &name) const

  Get a posting source given a name.

- void register_match_spy (const Xapian::MatchSpy &spy)

  Register a user-defined match spy class.

- const Xapian::MatchSpy * get_match_spy (const std::string &name) const

  Get a match spy given a name.

7.48.1 Detailed Description

Registry for user subclasses.

This class provides a way for the remote server to look up user subclasses when unserialising.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 Xapian::Registry::Registry (const Registry & *other*)

Copy constructor.

The internals are reference counted, so copying is cheap.

Parameters:

other The object to copy.

7.48.2.2 Xapian::Registry::Registry ()

Default constructor.

The registry will contain all standard subclasses of user-subclassable classes.

7.48.3 Member Function Documentation

7.48.3.1 const Xapian::MatchSpy* Xapian::Registry::get_match_spy (const std::string & *name*) const

Get a match spy given a name.

Parameters:

name The name of the match spy to find.

Returns:

An object with the requested name, or NULL if the match spy could not be found. The returned object is owned by the registry and so must not be deleted by the caller.

7.48.3.2 const Xapian::PostingSource* Xapian::Registry::get_posting_source (const std::string & *name*) const

Get a posting source given a name.

Parameters:

name The name of the posting source to find.

Returns:

An object with the requested name, or NULL if the posting source could not be found. The returned object is owned by the registry and so must not be deleted by the caller.

7.48.3.3 const Xapian::Weight* Xapian::Registry::get_weighting_scheme (const std::string & *name*) const

Get the weighting scheme given a name.

Parameters:

name The name of the weighting scheme to find.

Returns:

An object with the requested name, or NULL if the weighting scheme could not be found. The returned object is owned by the registry and so must not be deleted by the caller.

7.48.3.4 Registry& Xapian::Registry::operator= (const Registry & *other*)

Assignment operator.

The internals are reference counted, so assignment is cheap.

Parameters:

other The object to copy.

7.48.3.5 void Xapian::Registry::register_match_spy (const Xapian::MatchSpy & *spy*)

Register a user-defined match spy class.

Parameters:

spy The match spy to register.

7.48.3.6 void Xapian::Registry::register_posting_source (const Xapian::PostingSource & *source*)

Register a user-defined posting source class.

Parameters:

source The posting source to register.

7.48.3.7 void Xapian::Registry::register_weighting_scheme (const Xapian::Weight & *wt*)

Register a weighting scheme.

Parameters:

wt The weighting scheme to register.

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

- xapian/registry.h

7.49 Xapian::RSet Class Reference

A relevance set (R-Set).

Public Member Functions

- RSet (const RSet &rset)

  Copy constructor.

- void operator= (const RSet &rset)

  Assignment operator.

- RSet ()

  Default constructor.

- ~RSet ()

  Destructor.

- Xapian::doccount size () const

  The number of documents in this R-Set.

- bool empty () const

  Test if this R-Set is empty.

- void add_document (Xapian::docid did)

  Add a document to the relevance set.

- void add_document (const Xapian::MSetIterator &i)

  Add a document to the relevance set.

- void remove_document (Xapian::docid did)

  Remove a document from the relevance set.

- void remove_document (const Xapian::MSetIterator &i)

  Remove a document from the relevance set.

- bool contains (Xapian::docid did) const

  Test if a given document in the relevance set.

- bool contains (const Xapian::MSetIterator &i) const

  Test if a given document in the relevance set.

- std::string get_description () const

  Return a string describing this object.

7.49.1 Detailed Description

A relevance set (R-Set).

This is the set of documents which are marked as relevant, for use in modifying the term weights, and in performing query expansion.

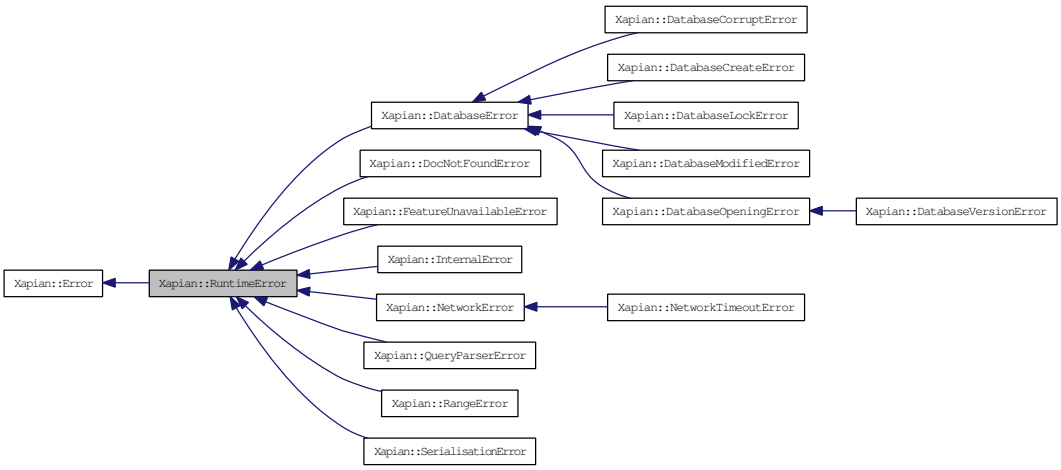
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.50 Xapian::RuntimeError Class Reference

The base class for exceptions indicating errors only detectable at runtime.

Inheritance diagram for Xapian::RuntimeError:



7.50.1 Detailed Description

The base class for exceptions indicating errors only detectable at runtime.

A subclass of RuntimeError will be thrown if Xapian detects an error which is exception derived from RuntimeError is thrown when an error is caused by problems with the data or environment rather than a programming mistake.

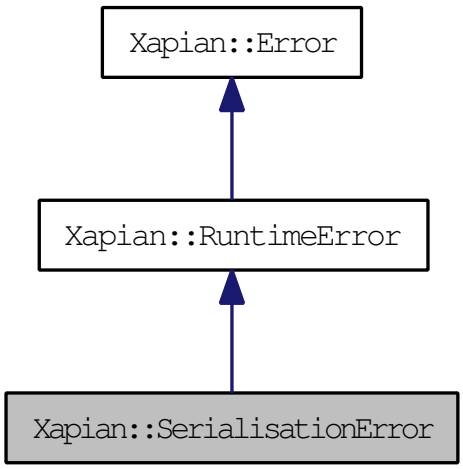
The documentation for this class was generated from the following file:

- xapian/error.h

7.51 Xapian::SerialisationError Class Reference

Indicates an error in the std::string serialisation of an object.

Inheritance diagram for Xapian::SerialisationError:



Public Member Functions

- SerialisationError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- SerialisationError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.51.1 Detailed Description

Indicates an error in the std::string serialisation of an object.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 Xapian::SerialisationError::SerialisationError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.51.2.2 Xapian::SerialisationError::SerialisationError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

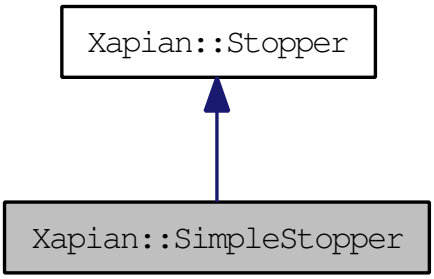
The documentation for this class was generated from the following file:

- xapian/error.h

7.52 Xapian::SimpleStopper Class Reference

Simple implementation of Stopper class - this will suit most users.

Inheritance diagram for Xapian::SimpleStopper:



Public Member Functions

- SimpleStopper ()

  Default constructor.

- template<class Iterator >
  SimpleStopper (Iterator begin, Iterator end)

  Initialise from a pair of iterators.

- void add (const std::string &word)

  Add a single stop word.

- virtual bool operator() (const std::string &term) const

  Is term a stop-word?

- virtual std::string get_description () const

  Return a string describing this object.

7.52.1 Detailed Description

Simple implementation of Stopper class - this will suit most users.

7.52.2 Member Function Documentation

7.52.2.1 virtual bool Xapian::SimpleStopper::operator() (const std::string & *term*) const `[inline, virtual]`

Is term a stop-word?

Parameters:

term The term to test.

Implements Xapian::Stopper.

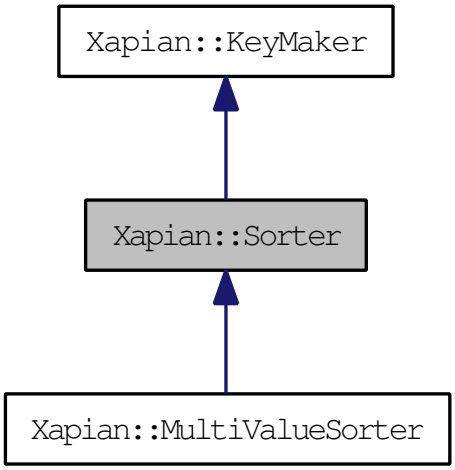
The documentation for this class was generated from the following file:

- xapian/queryparser.h

7.53 Xapian::Sorter Class Reference

Virtual base class for sorter functor.

Inheritance diagram for Xapian::Sorter:



7.53.1 Detailed Description

Virtual base class for sorter functor.

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

- xapian/keymaker.h

7.54 Xapian::Stem Class Reference

Class representing a stemming algorithm.

Public Member Functions

- Stem (const Stem &o)

  Copy constructor.

- void operator= (const Stem &o)

  Assignment.

- Stem ()

  Construct a Xapian::Stem object which doesn't change terms.

- Stem (const std::string &language)

  Construct a Xapian::Stem object for a particular language.

- Stem (StemImplementation *p)

  Construct a Xapian::Stem object with a user-provided stemming algorithm.

- ~Stem ()

  Destructor.

- std::string operator() (const std::string &word) const

  Stem a word.

- std::string get_description () const

  Return a string describing this object.

Static Public Member Functions

- static std::string get_available_languages ()

  Return a list of available languages.

7.54.1 Detailed Description

Class representing a stemming algorithm.

7.54.2 Constructor & Destructor Documentation

7.54.2.1 Xapian::Stem::Stem ()

Construct a Xapian::Stem object which doesn't change terms.

Equivalent to Stem("none").

7.54.2.2 Xapian::Stem::Stem (const std::string & *language*) [explicit]

Construct a Xapian::Stem object for a particular language.

Parameters:

language Either the English name for the language or the two letter ISO639 code.

The following language names are understood (aliases follow the name):

- none - don't stem terms
- danish (da)
- dutch (nl)
- english (en) - Martin Porter's 2002 revision of his stemmer
- english_lovins (lovins) - Lovin's stemmer
- english_porter (porter) - Porter's stemmer as described in his 1980 paper
- finnish (fi)
- french (fr)
- german (de)
- german2 - Normalises umlauts and ß
- hungarian (hu)
- italian (it)
- kraaij_pohlmann - A different Dutch stemmer
- norwegian (nb, nn, no)
- portuguese (pt)
- romanian (ro)
- russian (ru)
- spanish (es)
- swedish (sv)

- turkish (tr)

Exceptions:

Xapian::InvalidArgumentError is thrown if language isn't recognised.

7.54.2.3 Xapian::Stem::Stem (StemImplementation * *p*) [explicit]

Construct a Xapian::Stem object with a user-provided stemming algorithm.

You can subclass Xapian::StemImplementation to implement your own stemming algorithm (or to wrap a third-party algorithm) and then wrap your implementation in a Xapian::Stem object to pass to the Xapian API.

Parameters:

p The user-subclassed StemImplementation object. This is reference counted, and so will be automatically deleted by the Xapian::Stem wrapper when no longer required.

7.54.3 Member Function Documentation

7.54.3.1 static std::string Xapian::Stem::get_available_languages () [static]

Return a list of available languages.

Each stemmer is only included once in the list (not once for each alias). The name included is the English name of the language.

The list is returned as a string, with language names separated by spaces. This is a static method, so a Xapian::Stem object is not required for this operation.

7.54.3.2 std::string Xapian::Stem::operator() (const std::string & *word*) const

Stem a word.

Parameters:

word a word to stem.

Returns:

the stem

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

- xapian/stem.h

7.55 Xapian::StemImplementation Struct Reference

Class representing a stemming algorithm implementation.

Public Member Functions

- virtual ~StemImplementation ()

  Virtual destructor.

- virtual std::string operator() (const std::string &word)=0

  Stem the specified word.

- virtual std::string get_description () const =0

  Return a string describing this object.

7.55.1 Detailed Description

Class representing a stemming algorithm implementation.

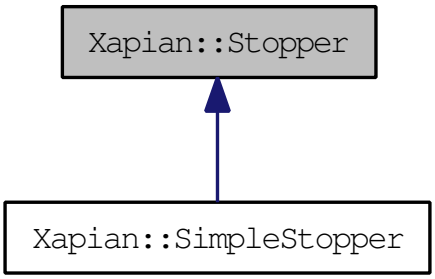
The documentation for this struct was generated from the following file:

- xapian/stem.h

7.56 Xapian::Stopper Class Reference

Base class for stop-word decision functor.

Inheritance diagram for Xapian::Stopper:



Public Member Functions

- virtual bool operator() (const std::string &term) const =0

  Is term a stop-word?

- virtual ~Stopper ()

  Class has virtual methods, so provide a virtual destructor.

- virtual std::string get_description () const

  Return a string describing this object.

7.56.1 Detailed Description

Base class for stop-word decision functor.

7.56.2 Member Function Documentation

7.56.2.1 virtual bool Xapian::Stopper::operator() (const std::string & *term*) const `[pure virtual]`

Is term a stop-word?

Parameters:

term The term to test.

Implemented in Xapian::SimpleStopper.

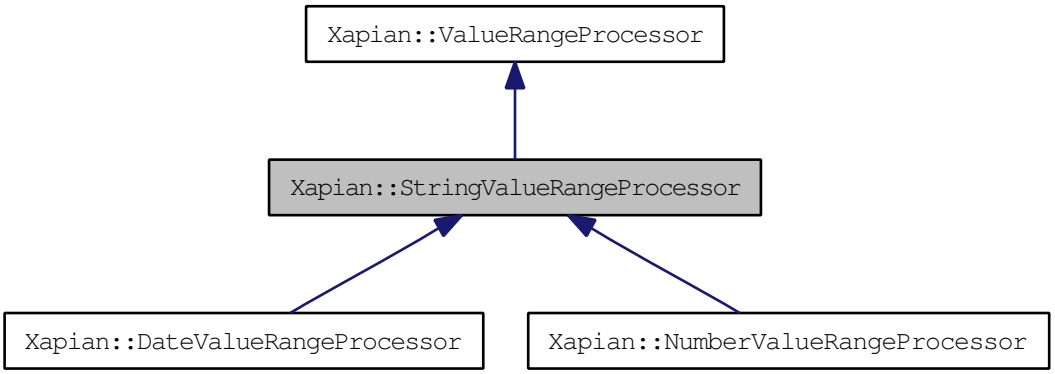
The documentation for this class was generated from the following file:

- xapian/queryparser.h

7.57 Xapian::StringValueRangeProcessor Class Reference

Handle a string range.

Inheritance diagram for Xapian::StringValueRangeProcessor:



Public Member Functions

- StringValueRangeProcessor (Xapian::valueno slot_)

  Constructor.

- StringValueRangeProcessor (Xapian::valueno slot_, const std::string &str_, bool prefix_=true)

  Constructor.

- Xapian::valueno operator() (std::string &begin, std::string &end)

  Check for a valid string range.

7.57.1 Detailed Description

Handle a string range.

The end points can be any strings.

7.57.2 Constructor & Destructor Documentation

7.57.2.1 Xapian::StringValueRangeProcessor::StringValueRangeProcessor (Xapian::valueno *slot_*) [inline]

Constructor.

Parameters:

slot_ The value number to return from operator().

7.57.2.2 Xapian::StringValueRangeProcessor::StringValueRangeProcessor (Xapian::valueno *slot_*, const std::string & *str_*, bool *prefix_* = `true`) `[inline]`

Constructor.

Parameters:

slot_ The value number to return from operator().

str_ A string to look for to recognise values as belonging to this range.

prefix_ Flag specifying whether to check for str_ as a prefix or a suffix.

7.57.3 Member Function Documentation

7.57.3.1 Xapian::valueno Xapian::StringValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) `[virtual]`

Check for a valid string range.

Parameters:

↔ ***begin*** The start of the range as specified in the query string by the user. This parameter is a non-const reference so the ValueRangeProcessor can modify it to return the value to start the range with.

↔ ***end*** The end of the range. This is also a non-const reference so it can be modified.

Returns:

A StringValueRangeProcessor always accepts a range it is offered, and returns the value of slot_ passed at construction time. It doesn't modify *begin* or *end*.

Implements Xapian::ValueRangeProcessor.

Reimplemented in Xapian::DateValueRangeProcessor, and Xapian::NumberValueRangeProcessor.

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

- xapian/queryparser.h

7.58 Xapian::TermGenerator Class Reference

Parses a piece of text and generate terms.

Public Types

- enum flags { FLAG_SPELLING = 128 }

  Flags to OR together and pass to TermGenerator::set_flags().

Public Member Functions

- TermGenerator (const TermGenerator &o)

  Copy constructor.

- TermGenerator & operator= (const TermGenerator &o)

  Assignment.

- TermGenerator ()

  Default constructor.

- ~TermGenerator ()

  Destructor.

- void set_stemmer (const Xapian::Stem &stemmer)

  Set the Xapian::Stem object to be used for generating stemmed terms.

- void set_stopper (const Xapian::Stopper *stop=NULL)

  Set the Xapian::Stopper object to be used for identifying stopwords.

- void set_document (const Xapian::Document &doc)

  Set the current document.

- const Xapian::Document & get_document () const

  Get the current document.

- void set_database (const Xapian::WritableDatabase &db)

  Set the database to index spelling data to.

- flags set_flags (flags toggle, flags mask=flags(0))

  Set flags.

- void index_text (const Xapian::Utf8Iterator &itor, Xapian::termcount wdf_inc=1, const std::string &prefix=std::string())

  Index some text.

- void index_text (const std::string &text, Xapian::termcount wdf_inc=1, const std::string &prefix=std::string())

  Index some text in a std::string.

- void index_text_without_positions (const Xapian::Utf8Iterator &itor, Xapian::termcount wdf_inc=1, const std::string &prefix=std::string())

  Index some text without positional information.

- void index_text_without_positions (const std::string &text, Xapian::termcount wdf_inc=1, const std::string &prefix=std::string())

  Index some text in a std::string without positional information.

- void increase_termpos (Xapian::termcount delta=100)

  Increase the term position used by index_text.

- Xapian::termcount get_termpos () const

  Get the current term position.

- void set_termpos (Xapian::termcount termpos)

  Set the current term position.

- std::string get_description () const

  Return a string describing this object.

7.58.1 Detailed Description

Parses a piece of text and generate terms.

This module takes a piece of text and parses it to produce words which are then used to generate suitable terms for indexing. The terms generated are suitable for use with Query objects produced by the QueryParser class.

7.58.2 Member Enumeration Documentation

7.58.2.1 enum Xapian::TermGenerator::flags

Flags to OR together and pass to TermGenerator::set_flags().

Enumerator:

FLAG_SPELLING Index data required for spelling correction.

7.58.3 Member Function Documentation

7.58.3.1 void Xapian::TermGenerator::increase_termpos (Xapian::termcount *delta* = `100`)

Increase the term position used by index_text.

This can be used between indexing text from different fields or other places to prevent phrase searches from spanning between them (e.g. between the title and body text, or between two chapters in a book).

Parameters:

delta Amount to increase the term position by (default: 100).

7.58.3.2 void Xapian::TermGenerator::index_text (const std::string & *text*, Xapian::termcount *wdf_inc* = `1`, const std::string & *prefix* = `std::string()`) `[inline]`

Index some text in a std::string.

Parameters:

text The text to index.

wdf_inc The wdf increment (default 1).

prefix The term prefix to use (default is no prefix).

7.58.3.3 void Xapian::TermGenerator::index_text (const Xapian::Utf8Iterator & *itor*, Xapian::termcount *wdf_inc* = `1`, const std::string & *prefix* = `std::string()`)

Index some text.

Parameters:

itor Utf8Iterator pointing to the text to index.

wdf_inc The wdf increment (default 1).

prefix The term prefix to use (default is no prefix).

7.58.3.4 void Xapian::TermGenerator::index_text_without_positions (const std::string & *text*, Xapian::termcount *wdf_inc* = `1`, const std::string & *prefix* = `std::string()`) `[inline]`

Index some text in a std::string without positional information.

Just like index_text, but no positional information is generated. This means that the database will be significantly smaller, but that phrase searching and NEAR won't be supported.

Parameters:

text The text to index.

wdf_inc The wdf increment (default 1).

prefix The term prefix to use (default is no prefix).

7.58.3.5 void Xapian::TermGenerator::index_text_without_positions (const Xapian::Utf8Iterator & *itor*, Xapian::termcount *wdf_inc* = `1`, const std::string & *prefix* = `std::string()`)

Index some text without positional information.

Just like index_text, but no positional information is generated. This means that the database will be significantly smaller, but that phrase searching and NEAR won't be supported.

Parameters:

itor Utf8Iterator pointing to the text to index.

wdf_inc The wdf increment (default 1).

prefix The term prefix to use (default is no prefix).

7.58.3.6 flags Xapian::TermGenerator::set_flags (flags *toggle*, flags *mask* = flags `(0)`)

Set flags.

The new value of flags is: (flags & mask) ^ toggle

To just set the flags, pass the new flags in toggle and the default value for mask.

Parameters:

toggle Flags to XOR.

mask Flags to AND with first.

Returns:

The old flags setting.

7.58.3.7 void Xapian::TermGenerator::set_stopper (const Xapian::Stopper * *stop* = `NULL`)

Set the Xapian::Stopper object to be used for identifying stopwords.

Stemmed forms of stopwords aren't indexed, but unstemmed forms still are so that searches for phrases including stop words still work.

Parameters:

stop The Stopper object to set (default NULL, which means no stopwords).

7.58.3.8 void Xapian::TermGenerator::set_termpos (Xapian::termcount *termpos*)

Set the current term position.

Parameters:

termpos The new term position to set.

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

- xapian/termgenerator.h

7.59 Xapian::TermIterator Class Reference

An iterator pointing to items in a list of terms.

Public Types

- typedef std::input_iterator_tag iterator_category
  Allow use as an STL iterator.
- typedef std::string value_type
  Allow use as an STL iterator.
- typedef Xapian::termcount_diff difference_type
  Allow use as an STL iterator.
- typedef std::string * pointer
  Allow use as an STL iterator.
- typedef std::string & reference
  Allow use as an STL iterator.

Public Member Functions

- TermIterator ()
  Default constructor - for declaring an uninitialised iterator.
- ~TermIterator ()
  Destructor.
- TermIterator (const TermIterator &other)
  Copying is allowed.
- void operator= (const TermIterator &other)
  Assignment is allowed.
- std::string operator* () const
  Return the current term.
- void skip_to (const std::string &tname)
  Advance the iterator to the specified term.
- Xapian::termcount get_wdf () const
  Return the wdf of the current term (if meaningful).
- Xapian::doccount get_termfreq () const

Return the term frequency of the current term (if meaningful).

- Xapian::termcount positionlist_count () const

  Return length of positionlist for current term.

- PositionIterator positionlist_begin () const

  Return PositionIterator pointing to start of positionlist for current term.

- PositionIterator positionlist_end () const

  Return PositionIterator pointing to end of positionlist for current term.

- std::string get_description () const

  Return a string describing this object.

7.59.1 Detailed Description

An iterator pointing to items in a list of terms.

7.59.2 Constructor & Destructor Documentation

7.59.2.1 Xapian::TermIterator::TermIterator (const TermIterator & *other*)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

7.59.3 Member Function Documentation

7.59.3.1 Xapian::doccount Xapian::TermIterator::get_termfreq () const

Return the term frequency of the current term (if meaningful).

The term frequency is the number of documents which a term indexes.

7.59.3.2 Xapian::termcount Xapian::TermIterator::get_wdf () const

Return the wdf of the current term (if meaningful).

The wdf (within document frequency) is the number of occurrences of a term in a particular document.

7.59.3.3 void Xapian::TermIterator::operator= (const TermIterator & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

7.59.3.4 void Xapian::TermIterator::skip_to (const std::string & *tname*)

Advance the iterator to the specified term.

If the specified term isn't in the list, position ourselves on the first term after it (or at_end() if no greater terms are present).

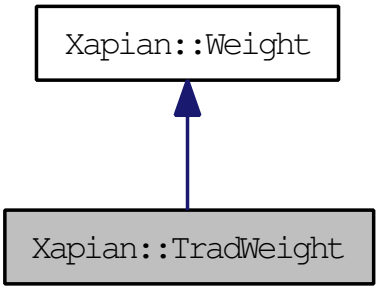
The documentation for this class was generated from the following file:

- xapian/termiterator.h

7.60 Xapian::TradWeight Class Reference

Xapian::Weight subclass implementing the traditional probabilistic formula.

Inheritance diagram for Xapian::TradWeight:



Public Member Functions

- TradWeight (double k=1.0)

  Construct a TradWeight.

- std::string name () const

  Return the name of this weighting scheme.

- std::string serialise () const

  Return this object's parameters serialised as a single string.

- TradWeight * unserialise (const std::string &s) const

  Unserialise parameters.

- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const

  Calculate the weight contribution for this object's term to a document.

- Xapian::weight get_maxpart () const

  Return an upper bound on what get_sumpart() can return for any document.

- Xapian::weight get_sumextra (Xapian::termcount doclen) const

  Calculate the term-independent weight component for a document.

- Xapian::weight get_maxextra () const

  Return an upper bound on what get_sumextra() can return for any document.

7.60.1 Detailed Description

Xapian::Weight subclass implementing the traditional probabilistic formula.

This class implements the "traditional" Probabilistic Weighting scheme, as described by the early papers on Probabilistic Retrieval. BM25 generally gives better results.

TradWeight(k) is equivalent to BM25Weight(k, 0, 0, 1, 0), except that the latter returns weights (k+1) times larger.

7.60.2 Constructor & Destructor Documentation

7.60.2.1 Xapian::TradWeight::TradWeight (double *k* = `1.0`) `[inline, explicit]`

Construct a TradWeight.

Parameters:

k A non-negative parameter controlling how influential within-document-frequency (wdf) and document length are. k=0 means that wdf and document length don't affect the weights. The larger k1 is, the more they do. (default 1)

7.60.3 Member Function Documentation

7.60.3.1 Xapian::weight Xapian::TradWeight::get_maxextra () const `[virtual]`

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.60.3.2 Xapian::weight Xapian::TradWeight::get_maxpart () const `[virtual]`

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.60.3.3 Xapian::weight Xapian::TradWeight::get_sumextra (Xapian::termcount *doclen*) const `[virtual]`

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters:

doclen The document's length (unnormalised).

Implements Xapian::Weight.

7.60.3.4 Xapian::weight Xapian::TradWeight::get_sumpart (Xapian::termcount *wdf*, Xapian::termcount *doclen*) const [virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters:

wdf The within document frequency of the term in the document.

doclen The document's length (unnormalised).

Implements Xapian::Weight.

7.60.3.5 std::string Xapian::TradWeight::name () const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend, you can use the default implementation which simply returns an empty string.

Reimplemented from Xapian::Weight.

7.60.3.6 std::string Xapian::TradWeight::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::Weight.

7.60.3.7 TradWeight* Xapian::TradWeight::unserialise (const std::string & *s*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Parameters:

s A string containing the serialised parameters.

Reimplemented from Xapian::Weight.

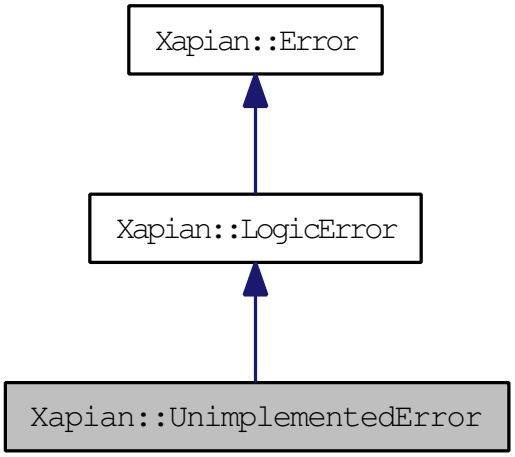
The documentation for this class was generated from the following file:

- xapian/weight.h

7.61 Xapian::UnimplementedError Class Reference

UnimplementedError indicates an attempt to use an unimplemented feature.

Inheritance diagram for Xapian::UnimplementedError:



Public Member Functions

- UnimplementedError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- UnimplementedError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.61.1 Detailed Description

UnimplementedError indicates an attempt to use an unimplemented feature.

7.61.2 Constructor & Destructor Documentation

7.61.2.1 Xapian::UnimplementedError::UnimplementedError (const std::string & *msg_*, const std::string & *context_* = `std::string()`, int *errno_* = `0`) `[inline, explicit]`

General purpose constructor.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

context_ Optional context information for this error.

errno_ Optional errno value associated with this error.

7.61.2.2 Xapian::UnimplementedError::UnimplementedError (const std::string & *msg_*, int *errno_*) `[inline]`

Construct from message and errno value.

Parameters:

msg_ Message giving details of the error, intended for human consumption.

errno_ Optional errno value associated with this error.

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

- xapian/error.h

7.62 Xapian::Utf8Iterator Class Reference

An iterator which returns Unicode character values from a UTF-8 encoded string.

Public Types

- typedef std::input_iterator_tag iterator_category
  We implement the semantics of an STL input_iterator.
- typedef unsigned value_type
  We implement the semantics of an STL input_iterator.
- typedef size_t difference_type
  We implement the semantics of an STL input_iterator.
- typedef const unsigned * pointer
  We implement the semantics of an STL input_iterator.
- typedef const unsigned & reference
  We implement the semantics of an STL input_iterator.

Public Member Functions

- const char * raw () const
  *Return the raw const char * pointer for the current position.*
- size_t left () const
  Return the number of bytes left in the iterator's buffer.
- void assign (const char *p_, size_t len)
  Assign a new string to the iterator.
- void assign (const std::string &s)
  Assign a new string to the iterator.
- Utf8Iterator (const char *p_)
  Create an iterator given a pointer to a null terminated string.
- Utf8Iterator (const char *p_, size_t len)
  Create an iterator given a pointer and a length.
- Utf8Iterator (const std::string &s)
  Create an iterator given a string.
- Utf8Iterator ()

Create an iterator which is at the end of its iteration.

- unsigned operator∗ () const

  Get the current Unicode character value pointed to by the iterator.

- Utf8Iterator operator++ (int)

  Move forward to the next Unicode character.

- Utf8Iterator & operator++ ()

  Move forward to the next Unicode character.

- bool operator== (const Utf8Iterator &other) const

  Test two Utf8Iterators for equality.

- bool operator!= (const Utf8Iterator &other) const

  Test two Utf8Iterators for inequality.

7.62.1 Detailed Description

An iterator which returns Unicode character values from a UTF-8 encoded string.

7.62.2 Constructor & Destructor Documentation

7.62.2.1 Xapian::Utf8Iterator::Utf8Iterator (const char ∗ *p_*) `[explicit]`

Create an iterator given a pointer to a null terminated string.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

p_ A pointer to the start of the null terminated string to read.

7.62.2.2 Xapian::Utf8Iterator::Utf8Iterator (const char ∗ *p_*, size_t *len*) `[inline]`

Create an iterator given a pointer and a length.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

p_ A pointer to the start of the string to read.

len The length of the string to read.

7.62.2.3 Xapian::Utf8Iterator::Utf8Iterator (const std::string & *s*) [inline]

Create an iterator given a string.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

s The string to read. Must not be modified while the iteration is in progress.

7.62.2.4 Xapian::Utf8Iterator::Utf8Iterator () [inline]

Create an iterator which is at the end of its iteration.

This can be compared to another iterator to check if the other iterator has reached its end.

7.62.3 Member Function Documentation

7.62.3.1 void Xapian::Utf8Iterator::assign (const std::string & *s*) [inline]

Assign a new string to the iterator.

The iterator will forget the string it was iterating through, and return characters from the start of the new string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

s The string to read. Must not be modified while the iteration is in progress.

References assign().

Referenced by assign().

7.62.3.2 void Xapian::Utf8Iterator::assign (const char * *p_*, size_t *len*) [inline]

Assign a new string to the iterator.

The iterator will forget the string it was iterating through, and return characters from the start of the new string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

*p_* A pointer to the start of the string to read.

len The length of the string to read.

7.62.3.3 size_t Xapian::Utf8Iterator::left () const [inline]

Return the number of bytes left in the iterator's buffer.

7.62.3.4 bool Xapian::Utf8Iterator::operator!= (const Utf8Iterator & *other*) const [inline]

Test two Utf8Iterators for inequality.

Parameters:

other The Utf8Iterator to compare this one with.

Returns:

true iff the iterators do not point to the same position.

7.62.3.5 unsigned Xapian::Utf8Iterator::operator∗ () const

Get the current Unicode character value pointed to by the iterator.

Returns unsigned(-1) if the iterator has reached the end of its buffer.

7.62.3.6 Utf8Iterator& Xapian::Utf8Iterator::operator++ () [inline]

Move forward to the next Unicode character.

Returns:

A reference to this object.

7.62.3.7 Utf8Iterator Xapian::Utf8Iterator::operator++ (int) [inline]

Move forward to the next Unicode character.

Returns:

An iterator pointing to the position before the move.

7.62.3.8 bool Xapian::Utf8Iterator::operator== (const Utf8Iterator & *other*) const [inline]

Test two Utf8Iterators for equality.

Parameters:

other The Utf8Iterator to compare this one with.

Returns:

true iff the iterators point to the same position.

7.62.3.9 const char* Xapian::Utf8Iterator::raw () const [inline]

Return the raw const char * pointer for the current position.

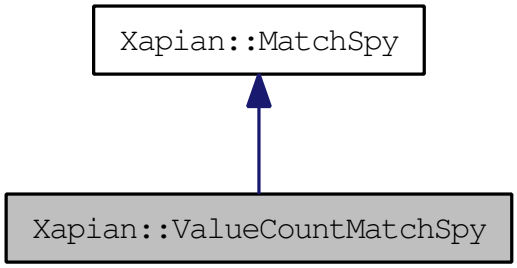
The documentation for this class was generated from the following file:

- xapian/unicode.h

7.63 Xapian::ValueCountMatchSpy Class Reference

Class for counting the frequencies of values in the matching documents.

Inheritance diagram for Xapian::ValueCountMatchSpy:



Public Member Functions

- ValueCountMatchSpy ()

  Construct an empty ValueCountMatchSpy.

- ValueCountMatchSpy (Xapian::valueno slot_)

  Construct a MatchSpy which counts the values in a particular slot.

- size_t get_total () const

  Return the total number of documents tallied.

- TermIterator values_begin () const

  Get an iterator over the values seen in the slot.

- TermIterator values_end () const

  End iterator corresponding to values_begin().

- TermIterator top_values_begin (size_t maxvalues) const

  Get an iterator over the most frequent values seen in the slot.

- TermIterator top_values_end (size_t) const

  End iterator corresponding to top_values_begin().

- void operator() (const Xapian::Document &doc, Xapian::weight wt)

  Implementation of virtual operator().

- virtual MatchSpy * clone () const

  Clone the match spy.

- virtual std::string name () const

Return the name of this match spy.

- virtual std::string serialise () const

  Return this object's parameters serialised as a single string.

- virtual MatchSpy * unserialise (const std::string &s, const Registry &context) const

  Unserialise parameters.

- virtual std::string serialise_results () const

  Serialise the results of this match spy.

- virtual void merge_results (const std::string &s)

  Unserialise some results, and merge them into this matchspy.

- virtual std::string get_description () const

  Return a string describing this object.

7.63.1 Detailed Description

Class for counting the frequencies of values in the matching documents.

7.63.2 Member Function Documentation

7.63.2.1 virtual MatchSpy* Xapian::ValueCountMatchSpy::clone () const [virtual]

Clone the match spy.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be passed information about the results seen by the parent.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented from Xapian::MatchSpy.

7.63.2.2 virtual std::string Xapian::ValueCountMatchSpy::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer, to avoid forcing those deriving their own MatchSpy subclasses from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::MatchSpy.

7.63.2.3 size_t Xapian::ValueCountMatchSpy::get_total () const [inline]

Return the total number of documents tallied.

7.63.2.4 virtual void Xapian::ValueCountMatchSpy::merge_results (const std::string & *s*) [virtual]

Unserialise some results, and merge them into this matchspy.

The order in which results are merged should not be significant, since this order is not specified (and will vary depending on the speed of the search in each sub-database).

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters:

s A string containing the serialised results.

Reimplemented from Xapian::MatchSpy.

7.63.2.5 virtual std::string Xapian::ValueCountMatchSpy::name () const [virtual]

Return the name of this match spy.

This name is used by the remote backend. It is passed with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called MyApp::FooMatchSpy, return "MyApp::FooMatchSpy" from this method.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::MatchSpy.

7.63.2.6 void Xapian::ValueCountMatchSpy::operator() (const Xapian::Document & *doc*, Xapian::weight *wt*) [virtual]

Implementation of virtual operator().

This implementation tallies values for a matching document.

Parameters:

doc The document to tally values for.

wt The weight of the document (ignored by this class).

Implements Xapian::MatchSpy.

7.63.2.7 virtual std::string Xapian::ValueCountMatchSpy::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::MatchSpy.

7.63.2.8 virtual std::string Xapian::ValueCountMatchSpy::serialise_results () const [virtual]

Serialise the results of this match spy.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::MatchSpy.

7.63.2.9 TermIterator Xapian::ValueCountMatchSpy::top_values_begin (size_t *maxvalues*) const

Get an iterator over the most frequent values seen in the slot.

Items will be returned in descending order of frequency. Values with the same frequency will be returned in ascending alphabetical order.

During the iteration, the frequency of the current value can be obtained with the get_termfreq() method on the iterator.

Parameters:

maxvalues The maximum number of values to return.

7.63.2.10 virtual MatchSpy* Xapian::ValueCountMatchSpy::unserialise (const std::string & *s*, const Registry & *context*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Parameters:

s A string containing the serialised results.

context Registry object to use for unserialisation to permit MatchSpy subclasses with sub-MatchSpy objects to be implemented.

Reimplemented from Xapian::MatchSpy.

7.63.2.11 TermIterator Xapian::ValueCountMatchSpy::values_begin () const

Get an iterator over the values seen in the slot.

Items will be returned in ascending alphabetical order.

During the iteration, the frequency of the current value can be obtained with the get_-termfreq() method on the iterator.

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

- xapian/matchspy.h

7.64 Xapian::ValueIterator Class Reference

Class for iterating over document values.

Public Member Functions

- ValueIterator (const ValueIterator &o)
  Copy constructor.
- ValueIterator & operator= (const ValueIterator &o)
  Assignment.
- ValueIterator ()
  Default constructor.
- ~ValueIterator ()
  Destructor.
- std::string operator* () const
  Return the value at the current position.
- ValueIterator & operator++ ()
  Advance the iterator to the next position.
- DerefWrapper_< std::string > operator++ (int)
  Advance the iterator to the next position (postfix version).
- Xapian::docid get_docid () const
  Return the docid at the current position.
- Xapian::valueno get_valueno () const
  Return the value slot number for the current position.
- void skip_to (Xapian::docid docid_or_slot)
  Advance the iterator to document id or value slot docid_or_slot.
- bool check (Xapian::docid docid)
  Check if the specified docid occurs.
- std::string get_description () const
  Return a string describing this object.

7.64.1 Detailed Description

Class for iterating over document values.

7.64.2 Constructor & Destructor Documentation

7.64.2.1 Xapian::ValueIterator::ValueIterator ()

Default constructor.

Creates an uninitialised iterator, which can't be used before being assigned to, but is sometimes syntactically convenient.

7.64.3 Member Function Documentation

7.64.3.1 bool Xapian::ValueIterator::check (Xapian::docid *docid*)

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database.

This method acts like skip_to() if that can be done at little extra cost, in which case it then returns true. This is how brass and chert databases behave because they store values in streams which allow for an efficient implementation of skip_to().

Otherwise it simply checks if a particular docid is present. If it is, it returns true. If it isn't, it returns false, and leaves the position unspecified (and hence the result of calling methods which depends on the current position, such as get_docid(), are also unspecified). In this state, next() will advance to the first matching position after document *did*, and skip_to() will act as it would if the position was the first matching position after document *did*.

Currently the inmemory, flint, and remote backends behave in the latter way because they don't support streamed values and so skip_to() must check each document it skips over which is significantly slower.

Parameters:

docid The document id to check.

7.64.3.2 Xapian::docid Xapian::ValueIterator::get_docid () const

Return the docid at the current position.

If we're iterating over values of a document, this method will throw Xapian::InvalidOperationError.

7.64.3.3 Xapian::valueno Xapian::ValueIterator::get_valueno () const

Return the value slot number for the current position.

If the iterator is over all values in a slot, this returns that slot's number. If the iterator is over the values in a particular document, it returns the number of each slot in turn.

7.64.3.4 void Xapian::ValueIterator::skip_to (Xapian::docid *docid_or_slot*)

Advance the iterator to document id or value slot *docid_or_slot*.

If this iterator is over values in a document, then this method advances the iterator to value slot *docid_or_slot*, or the first slot after it if there is no value in slot *slot*.

If this iterator is over values in a particular slot, then this method advances the iterator to document id *docid_or_slot*, or the first document id after it if there is no value in the slot we're iterating over for document *docid_or_slot*.

Note: The "two-faced" nature of this method is due to how C++ overloading works. Xapian::docid and Xapian::valueno are both typedefs for the same unsigned integer type, so overloading can't distinguish them.

Parameters:

docid_or_slot The docid/slot to advance to.

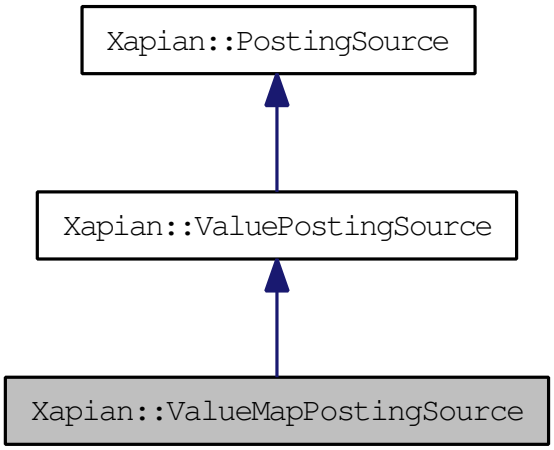
The documentation for this class was generated from the following file:

- xapian/valueiterator.h

7.65 Xapian::ValueMapPostingSource Class Reference

A posting source which looks up weights in a map using values as the key.

Inheritance diagram for Xapian::ValueMapPostingSource:



Public Member Functions

- ValueMapPostingSource (Xapian::valueno slot_)

  Construct a ValueWeightPostingSource.

- void add_mapping (const std::string &key, double wt)

  Add a mapping.

- void clear_mappings ()

  Clear all mappings.

- void set_default_weight (double wt)

  Set a default weight for document values not in the map.

- Xapian::weight get_weight () const

  Return the weight contribution for the current document.

- ValueMapPostingSource * clone () const

  Clone the posting source.

- std::string name () const

  Name of the posting source class.

- std::string serialise () const

  Serialise object parameters into a string.

- ValueMapPostingSource * unserialise (const std::string &s) const

  Create object given string serialisation returned by serialise().

- void init (const Database &db_)

  Set this PostingSource to the start of the list of postings.

- std::string get_description () const

  Return a string describing this object.

7.65.1 Detailed Description

A posting source which looks up weights in a map using values as the key.

This allows will return entries for all documents in the given database which have a value in the slot specified. The values will be mapped to the corresponding weight in the weight map. If there is no mapping for a particular value, the default weight will be returned (which itself defaults to 0.0).

7.65.2 Constructor & Destructor Documentation

7.65.2.1 Xapian::ValueMapPostingSource::ValueMapPostingSource (Xapian::valueno *slot_*)

Construct a ValueWeightPostingSource.

Parameters:

slot_ The value slot to read values from.

7.65.3 Member Function Documentation

7.65.3.1 void Xapian::ValueMapPostingSource::add_mapping (const std::string & *key*, double *wt*)

Add a mapping.

Parameters:

key The key looked up from the value slot.

wt The weight to give this key.

7.65.3.2 void Xapian::ValueMapPostingSource::clear_mappings ()

Clear all mappings.

7.65.3.3 ValueMapPostingSource* Xapian::ValueMapPostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented from Xapian::PostingSource.

7.65.3.4 std::string Xapian::ValueMapPostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.

7.65.3.5 Xapian::weight Xapian::ValueMapPostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::PostingSource.

7.65.3.6 void Xapian::ValueMapPostingSource::init (const Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Reimplemented from Xapian::ValuePostingSource.

7.65.3.7 std::string Xapian::ValueMapPostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::PostingSource.

7.65.3.8 std::string Xapian::ValueMapPostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::PostingSource.

7.65.3.9 void Xapian::ValueMapPostingSource::set_default_weight (double *wt*)

Set a default weight for document values not in the map.

Parameters:

wt The weight to set as the default.

7.65.3.10 ValueMapPostingSource* Xapian::ValueMapPostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::PostingSource.

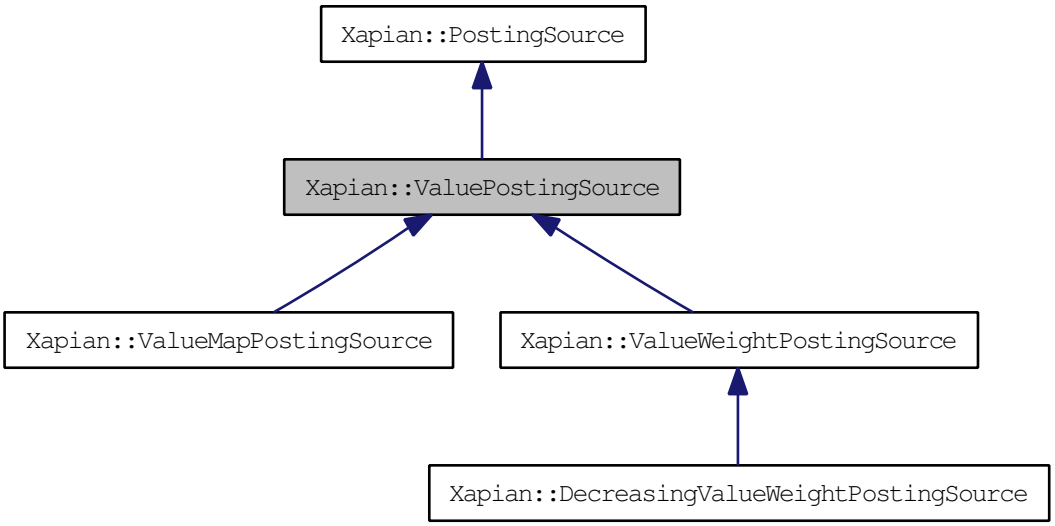
The documentation for this class was generated from the following file:

- xapian/postingsource.h

7.66 Xapian::ValuePostingSource Class Reference

A posting source which generates weights from a value slot.

Inheritance diagram for Xapian::ValuePostingSource:



Public Member Functions

- ValuePostingSource (Xapian::valueno slot_)

  Construct a ValuePostingSource.

- Xapian::doccount get_termfreq_min () const

  A lower bound on the number of documents this object can return.

- Xapian::doccount get_termfreq_est () const

  An estimate of the number of documents this object can return.

- Xapian::doccount get_termfreq_max () const

  An upper bound on the number of documents this object can return.

- void next (Xapian::weight min_wt)

  Advance the current position to the next matching document.

- void skip_to (Xapian::docid min_docid, Xapian::weight min_wt)

  Advance to the specified docid.

- bool check (Xapian::docid min_docid, Xapian::weight min_wt)

  Check if the specified docid occurs.

- bool at_end () const

  Return true if the current position is past the last entry in this list.

- Xapian::docid get_docid () const

  Return the current docid.

- void init (const Database &db_)

  Set this PostingSource to the start of the list of postings.

Protected Attributes

- Xapian::Database db

  The database we're reading values from.

- Xapian::valueno slot

  The slot we're reading values from.

- Xapian::ValueIterator value_it

  Value stream iterator.

- bool started

  Flag indicating if we've started (true if we have).

- Xapian::doccount termfreq_min

  A lower bound on the term frequency.

- Xapian::doccount termfreq_est

  An estimate of the term frequency.

- Xapian::doccount termfreq_max

  An upper bound on the term frequency.

7.66.1 Detailed Description

A posting source which generates weights from a value slot.

This is a base class for classes which generate weights using values stored in the specified slot. For example, ValueWeightPostingSource uses sortable_unserialise to convert values directly to weights.

The upper bound on the weight returned is set to DBL_MAX. Subclasses should call set_maxweight() in their init() methods after calling ValuePostingSource::init() if they know a tighter bound on the weight.

7.66.2 Constructor & Destructor Documentation

7.66.2.1 Xapian::ValuePostingSource::ValuePostingSource (Xapian::valueno *slot_*)

Construct a ValuePostingSource.

Parameters:

*slot_* The value slot to read values from.

7.66.3 Member Function Documentation

7.66.3.1 bool Xapian::ValuePostingSource::at_end () const [virtual]

Return true if the current position is past the last entry in this list.

At least one of *next()*, *skip_to()* or *check()* will be called before this method is first called.

Implements Xapian::PostingSource.

7.66.3.2 bool Xapian::ValuePostingSource::check (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after *did*.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn't.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

did The document id to check.

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.66.3.3 Xapian::docid Xapian::ValuePostingSource::get_docid () const [virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document". See *get_weight()* for details.

Note: in the case of a multi-database search, the returned docid should be in the single subdatabase relevant to this posting source. See the *init()* method for details.

Implements Xapian::PostingSource.

7.66.3.4 Xapian::doccount Xapian::ValuePostingSource::get_termfreq_est () const [virtual]

An estimate of the number of documents this object can return.

It must always be true that:

get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

7.66.3.5 Xapian::doccount Xapian::ValuePostingSource::get_termfreq_max () const [virtual]

An upper bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

7.66.3.6 Xapian::doccount Xapian::ValuePostingSource::get_termfreq_min () const [virtual]

A lower bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

7.66.3.7 void Xapian::ValuePostingSource::init (const Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Implements Xapian::PostingSource.

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, and Xapian::ValueMapPostingSource.

7.66.3.8 void Xapian::ValuePostingSource::next (Xapian::weight *min_wt*) [virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Implements Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.66.3.9 void Xapian::ValuePostingSource::skip_to (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Advance to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at_end() if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

did The document id to advance to.

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.66.4 Member Data Documentation

7.66.4.1 Xapian::doccount Xapian::ValuePostingSource::termfreq_est [protected]

An estimate of the term frequency.

Subclasses should set this if they are overriding the next(), skip_to() or check() methods.

7.66.4.2 Xapian::doccount Xapian::ValuePostingSource::termfreq_max [protected]

An upper bound on the term frequency.

Subclasses should set this if they are overriding the next(), skip_to() or check() methods.

7.66.4.3 Xapian::doccount Xapian::ValuePostingSource::termfreq_min [protected]

A lower bound on the term frequency.

Subclasses should set this if they are overriding the next(), skip_to() or check() methods to return fewer documents.

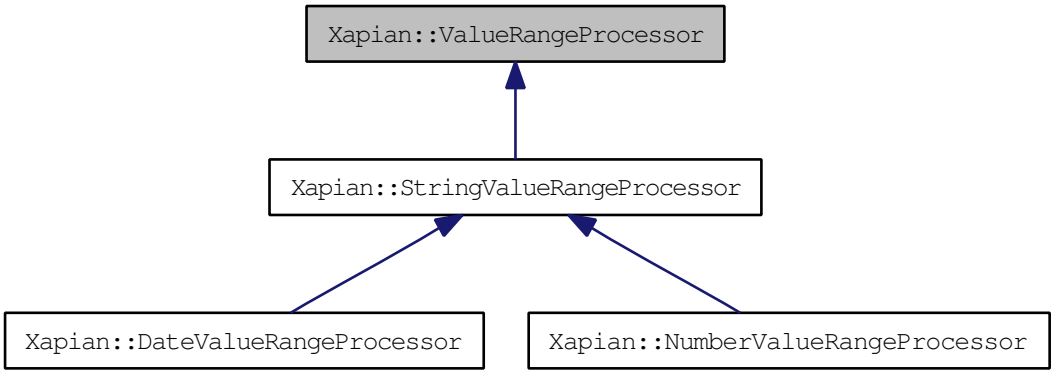
The documentation for this class was generated from the following file:

- xapian/postingsource.h

7.67 Xapian::ValueRangeProcessor Struct Reference

Base class for value range processors.

Inheritance diagram for Xapian::ValueRangeProcessor:



Public Member Functions

- virtual ~ValueRangeProcessor ()

  Destructor.

- virtual Xapian::valueno operator() (std::string &begin, std::string &end)=0

  Check for a valid range of this type.

7.67.1 Detailed Description

Base class for value range processors.

7.67.2 Member Function Documentation

7.67.2.1 virtual Xapian::valueno Xapian::ValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) [pure virtual]

Check for a valid range of this type.

Parameters:

↔ ***begin*** The start of the range as specified in the query string by the user. This parameter is a non-const reference so the ValueRangeProcessor can modify it to return the value to start the range with.

↔ ***end*** The end of the range. This is also a non-const reference so it can be modified.

Returns:

If this ValueRangeProcessor recognises the range BEGIN..END it returns the value slot number to range filter on. Otherwise it returns Xapian::BAD_VALUENO.

Implemented in Xapian::StringValueRangeProcessor, Xapian::DateValueRangeProcessor, and Xapian::NumberValueRangeProcessor.

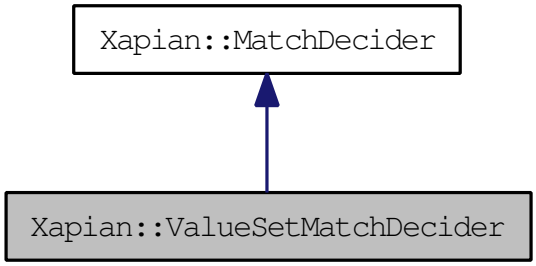
The documentation for this struct was generated from the following file:

- xapian/queryparser.h

7.68 Xapian::ValueSetMatchDecider Class Reference

MatchDecider filtering results based on whether document values are in a user-defined set.

Inheritance diagram for Xapian::ValueSetMatchDecider:



Public Member Functions

- ValueSetMatchDecider (Xapian::valueno slot, bool inclusive_)

  Construct a ValueSetMatchDecider.

- void add_value (const std::string &value)

  Add a value to the test set.

- void remove_value (const std::string &value)

  Remove a value from the test set.

- bool operator() (const Xapian::Document &doc) const

  Decide whether we want a particular document to be in the MSet.

7.68.1 Detailed Description

MatchDecider filtering results based on whether document values are in a user-defined set.

7.68.2 Constructor & Destructor Documentation

7.68.2.1 Xapian::ValueSetMatchDecider::ValueSetMatchDecider (Xapian::valueno *slot*, bool *inclusive_*) [inline]

Construct a ValueSetMatchDecider.

Parameters:

slot The value slot number to look in.

inclusive_ If true, match decider accepts documents which have a value in the specified slot which is a member of the test set; if false, match decider accepts documents which do not have a value in the specified slot.

7.68.3 Member Function Documentation

7.68.3.1 void Xapian::ValueSetMatchDecider::add_value (const std::string & *value*) [inline]

Add a value to the test set.

Parameters:

value The value to add to the test set.

7.68.3.2 bool Xapian::ValueSetMatchDecider::operator() (const Xapian::Document & *doc*) const [virtual]

Decide whether we want a particular document to be in the MSet.

Parameters:

doc The document to test.

Returns:

true if the document is acceptable, or false if the document should be excluded from the MSet.

Implements Xapian::MatchDecider.

7.68.3.3 void Xapian::ValueSetMatchDecider::remove_value (const std::string & *value*) [inline]

Remove a value from the test set.

Parameters:

value The value to remove from the test set.

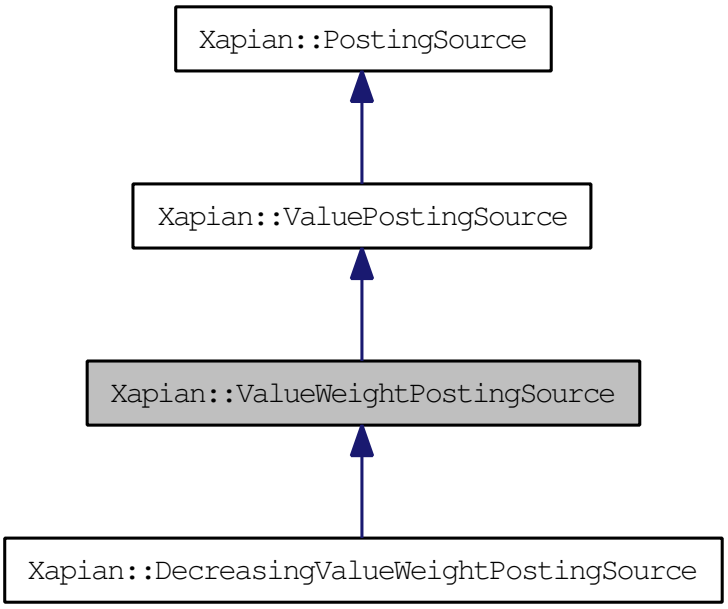
The documentation for this class was generated from the following file:

- xapian/valuesetmatchdecider.h

7.69 Xapian::ValueWeightPostingSource Class Reference

A posting source which reads weights from a value slot.

Inheritance diagram for Xapian::ValueWeightPostingSource:



Public Member Functions

- ValueWeightPostingSource (Xapian::valueno slot_)

  Construct a ValueWeightPostingSource.

- Xapian::weight get_weight () const

  Return the weight contribution for the current document.

- ValueWeightPostingSource * clone () const

  Clone the posting source.

- std::string name () const

  Name of the posting source class.

- std::string serialise () const

  Serialise object parameters into a string.

- ValueWeightPostingSource * unserialise (const std::string &s) const

  Create object given string serialisation returned by serialise().

- void init (const Database &db_)

  Set this PostingSource to the start of the list of postings.

- std::string get_description () const

  Return a string describing this object.

7.69.1 Detailed Description

A posting source which reads weights from a value slot.

This returns entries for all documents in the given database which have a non empty values in the specified slot. It returns a weight calculated by applying sortable_unserialise to the value stored in the slot (so the values stored should probably have been calculated by applying sortable_serialise to a floating point number at index time).

The upper bound on the weight returned is set using the upper bound on the values in the specified slot, or DBL_MAX if value bounds aren't supported by the current backend.

For efficiency, this posting source doesn't check that the stored values are valid in any way, so it will never raise an exception due to invalid stored values. In particular, it doesn't ensure that the unserialised values are positive, which is a requirement for weights. The behaviour if the slot contains values which unserialise to negative values is undefined.

7.69.2 Constructor & Destructor Documentation

7.69.2.1 Xapian::ValueWeightPostingSource::ValueWeightPostingSource (Xapian::valueno *slot_*)

Construct a ValueWeightPostingSource.

Parameters:

slot_ The value slot to read values from.

7.69.3 Member Function Documentation

7.69.3.1 ValueWeightPostingSource* Xapian::ValueWeightPostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the

matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.69.3.2 std::string Xapian::ValueWeightPostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.69.3.3 Xapian::weight Xapian::ValueWeightPostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.69.3.4 void Xapian::ValueWeightPostingSource::init (const Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Reimplemented from Xapian::ValuePostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.69.3.5 std::string Xapian::ValueWeightPostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.69.3.6 std::string Xapian::ValueWeightPostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.69.3.7 ValueWeightPostingSource* Xapian::ValueWeightPostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

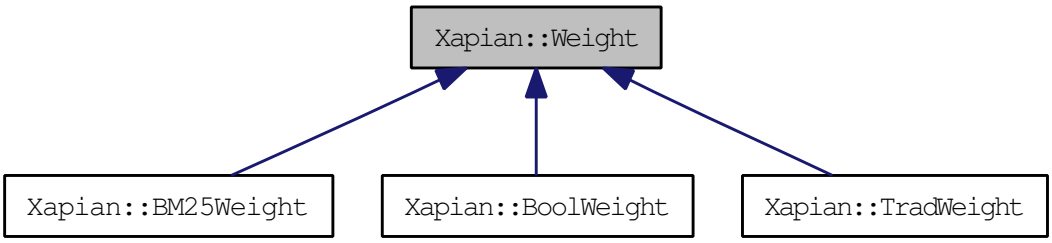
The documentation for this class was generated from the following file:

- xapian/postingsource.h

7.70 Xapian::Weight Class Reference

Abstract base class for weighting schemes.

Inheritance diagram for Xapian::Weight:



Public Member Functions

- virtual ~Weight ()

  Virtual destructor, because we have virtual methods.

- virtual Weight * clone () const =0

  Clone this object.

- virtual std::string name () const

  Return the name of this weighting scheme.

- virtual std::string serialise () const

  Return this object's parameters serialised as a single string.

- virtual Weight * unserialise (const std::string &s) const

  Unserialise parameters.

- virtual Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const =0

  Calculate the weight contribution for this object's term to a document.

- virtual Xapian::weight get_maxpart () const =0

  Return an upper bound on what get_sumpart() can return for any document.

- virtual Xapian::weight get_sumextra (Xapian::termcount doclen) const =0

  Calculate the term-independent weight component for a document.

- virtual Xapian::weight get_maxextra () const =0

  Return an upper bound on what get_sumextra() can return for any document.

Protected Types

- enum stat_flags

  Stats which the weighting scheme can use (see need_stat()).

Protected Member Functions

- void need_stat (stat_flags flag)

  Tell Xapian that your subclass will want a particular statistic.

- virtual void init (double factor)=0

  Allow the subclass to perform any initialisation it needs to.

- Weight (const Weight &)

  Don't allow copying.

- Weight ()

  Default constructor, needed by subclass constructors.

- Xapian::doccount get_collection_size () const

  The number of documents in the collection.

- Xapian::doccount get_rset_size () const

  The number of documents marked as relevant.

- Xapian::doclength get_average_length () const

  The average length of a document in the collection.

- Xapian::doccount get_termfreq () const

  The number of documents which this term indexes.

- Xapian::doccount get_reltermfreq () const

  The number of relevant documents which this term indexes.

- Xapian::termcount get_query_length () const

  The length of the query.

- Xapian::termcount get_wqf () const

  The within-query-frequency of this term.

- Xapian::termcount get_doclength_upper_bound () const

  An lower bound on the maximum length of any document in the database.

- Xapian::termcount get_doclength_lower_bound () const

  An upper bound on the maximum length of any document in the database.

- Xapian::termcount get_wdf_upper_bound () const

  An upper bound on the wdf of this term.

7.70.1 Detailed Description

Abstract base class for weighting schemes.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 virtual Xapian::Weight::~Weight () [virtual]

Virtual destructor, because we have virtual methods.

7.70.2.2 Xapian::Weight::Weight (const Weight &) [protected]

Don't allow copying.

This would ideally be private, but that causes a compilation error with GCC 4.1 (which appears to be a bug).

7.70.3 Member Function Documentation

7.70.3.1 virtual Weight* Xapian::Weight::clone () const [pure virtual]

Clone this object.

This method allocates and returns a copy of the object it is called on.

If your subclass is called FooWeight and has parameters a and b, then you would implement FooWeight::clone() like so:

FooWeight * FooWeight::clone() const { return new FooWeight(a, b); }

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

7.70.3.2 Xapian::termcount Xapian::Weight::get_doclength_lower_bound () const [inline, protected]

An upper bound on the maximum length of any document in the database.

This should only be used by get_maxpart() and get_maxextra().

7.70.3.3 Xapian::termcount Xapian::Weight::get_doclength_upper_bound () const [inline, protected]

An lower bound on the maximum length of any document in the database.

This should only be used by get_maxpart() and get_maxextra().

7.70.3.4 virtual Xapian::weight Xapian::Weight::get_maxextra () const [pure virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.70.3.5 virtual Xapian::weight Xapian::Weight::get_maxpart () const [pure virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.70.3.6 virtual Xapian::weight Xapian::Weight::get_sumextra (Xapian::termcount *doclen*) const [pure virtual]

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters:

doclen The document's length (unnormalised).

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.70.3.7 virtual Xapian::weight Xapian::Weight::get_sumpart (Xapian::termcount *wdf*, Xapian::termcount *doclen*) const [pure virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters:

wdf The within document frequency of the term in the document.

doclen The document's length (unnormalised).

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.70.3.8 Xapian::termcount Xapian::Weight::get_wdf_upper_bound () const [inline, protected]

An upper bound on the wdf of this term.

This should only be used by get_maxpart() and get_maxextra().

7.70.3.9 virtual void Xapian::Weight::init (double *factor*) [protected, pure virtual]

Allow the subclass to perform any initialisation it needs to.

Parameters:

factor Any scaling factor (e.g. from OP_SCALE_WEIGHT).

7.70.3.10 virtual std::string Xapian::Weight::name () const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend, you can use the default implementation which simply returns an empty string.

Reimplemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.70.3.11 void Xapian::Weight::need_stat (stat_flags *flag*) [inline, protected]

Tell Xapian that your subclass will want a particular statistic.

Some of the statistics can be costly to fetch or calculate, so Xapian needs to know which are actually going to be used. You should call need_stat() from your constructor for each such statistic.

Parameters:

flag The stat_flags value for a required statistic.

7.70.3.12 virtual std::string Xapian::Weight::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.70.3.13 virtual Weight* Xapian::Weight::unserialise (const std::string & *s*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Parameters:

s A string containing the serialised parameters.

Reimplemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

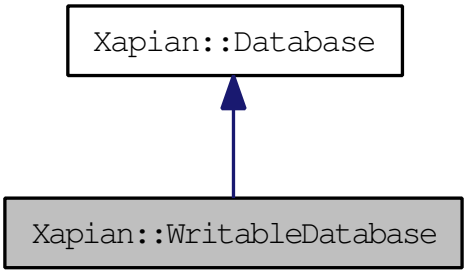
The documentation for this class was generated from the following file:

- xapian/weight.h

7.71 Xapian::WritableDatabase Class Reference

This class provides read/write access to a database.

Inheritance diagram for Xapian::WritableDatabase:



Public Member Functions

- virtual ~WritableDatabase ()

  Destroy this handle on the database.

- WritableDatabase ()

  Create an empty WritableDatabase.

- WritableDatabase (const std::string &path, int action)

  Open a database for update, automatically determining the database backend to use.

- WritableDatabase (const WritableDatabase &other)

  Copying is allowed.

- void operator= (const WritableDatabase &other)

  Assignment is allowed.

- void commit ()

  Commit any pending modifications made to the database.

- void flush ()

  Pre-1.1.0 name for commit().

- void begin_transaction (bool flushed=true)

  Begin a transaction.

- void commit_transaction ()

  Complete the transaction currently in progress.

- void cancel_transaction ()

Abort the transaction currently in progress, discarding the pending modifications made to the database.

- Xapian::docid add_document (const Xapian::Document &document)

  Add a new document to the database.

- void delete_document (Xapian::docid did)

  Delete a document from the database.

- void delete_document (const std::string &unique_term)

  Delete any documents indexed by a term from the database.

- void replace_document (Xapian::docid did, const Xapian::Document &document)

  Replace a given document in the database.

- Xapian::docid replace_document (const std::string &unique_term, const Xapian::Document &document)

  Replace any documents matching a term.

- void add_spelling (const std::string &word, Xapian::termcount freqinc=1) const

  Add a word to the spelling dictionary.

- void remove_spelling (const std::string &word, Xapian::termcount freqdec=1) const

  Remove a word from the spelling dictionary.

- void add_synonym (const std::string &term, const std::string &synonym) const

  Add a synonym for a term.

- void remove_synonym (const std::string &term, const std::string &synonym) const

  Remove a synonym for a term.

- void clear_synonyms (const std::string &term) const

  Remove all synonyms for a term.

- void set_metadata (const std::string &key, const std::string &value)

  Set the user-specified metadata associated with a given key.

- std::string get_description () const

  Return a string describing this object.

7.71.1 Detailed Description

This class provides read/write access to a database.

7.71.2 Constructor & Destructor Documentation

7.71.2.1 virtual Xapian::WritableDatabase::~WritableDatabase () [virtual]

Destroy this handle on the database.

If there are no copies of this object remaining, the database will be closed. If there are any transactions in progress these will be aborted as if cancel_transaction had been called.

7.71.2.2 Xapian::WritableDatabase::WritableDatabase (const std::string & *path*, int *action*)

Open a database for update, automatically determining the database backend to use.

If the database is to be created, Xapian will try to create the directory indicated by path if it doesn't already exist (but only the leaf directory, not recursively).

Parameters:

path directory that the database is stored in.

action one of:

- Xapian::DB_CREATE_OR_OPEN open for read/write; create if no db exists
- Xapian::DB_CREATE create new database; fail if db exists
- Xapian::DB_CREATE_OR_OVERWRITE overwrite existing db; create if none exists
- Xapian::DB_OPEN open for read/write; fail if no db exists

Exceptions:

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

Xapian::DatabaseLockError will be thrown if a lock couldn't be acquired on the database.

7.71.2.3 Xapian::WritableDatabase::WritableDatabase (const WritableDatabase & *other*)

Copying is allowed.

The internals are reference counted, so copying is cheap.

Parameters:

other The object to copy.

7.71.3 Member Function Documentation

7.71.3.1 Xapian::docid Xapian::WritableDatabase::add_document (const Xapian::Document & *document*)

Add a new document to the database.

This method adds the specified document to the database, returning a newly allocated document ID. Automatically allocated document IDs come from a per-database monotonically increasing counter, so IDs from deleted documents won't be reused.

If you want to specify the document ID to be used, you should call replace_document() instead.

Note that changes to the database won't be immediately committed to disk; see commit() for more details.

As with all database modification operations, the effect is atomic: the document will either be fully added, or the document fails to be added and an exception is thrown (possibly at a later time when commit() is called or the database is closed).

Parameters:

document The new document to be added.

Returns:

The document ID of the newly added document.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

7.71.3.2 void Xapian::WritableDatabase::add_spelling (const std::string & *word*, Xapian::termcount *freqinc* = 1) const

Add a word to the spelling dictionary.

If the word is already present, its frequency is increased.

Parameters:

word The word to add.

freqinc How much to increase its frequency by (default 1).

7.71.3.3 void Xapian::WritableDatabase::add_synonym (const std::string & *term*, const std::string & *synonym*) const

Add a synonym for a term.

Parameters:

term The term to add a synonym for.

synonym The synonym to add. If this is already a synonym for *term*, then no action is taken.

7.71.3.4 void Xapian::WritableDatabase::begin_transaction (bool *flushed* = `true`)

Begin a transaction.

In Xapian a transaction is a group of modifications to the database which are linked such that either all will be applied simultaneously or none will be applied at all. Even in the case of a power failure, this characteristic should be preserved (as long as the filesystem isn't corrupted, etc).

A transaction is started with begin_transaction() and can either be committed by calling commit_transaction() or aborted by calling cancel_transaction().

By default, a transaction implicitly calls commit() before and after so that the modifications stand and fall without affecting modifications before or after.

The downside of these implicit calls to commit() is that small transactions can harm indexing performance in the same way that explicitly calling commit() frequently can.

If you're applying atomic groups of changes and only wish to ensure that each group is either applied or not applied, then you can prevent the automatic commit() before and after the transaction by starting the transaction with begin_transaction(false). However, if cancel_transaction is called (or if commit_transaction isn't called before the WritableDatabase object is destroyed) then any changes which were pending before the transaction began will also be discarded.

Transactions aren't currently supported by the InMemory backend.

Parameters:

flushed Is this a flushed transaction? By default transactions are "flushed", which means that committing a transaction will ensure those changes are permanently written to the database. By contrast, unflushed transactions only ensure that changes within the transaction are either all applied or all aren't.

Exceptions:

Xapian::UnimplementedError will be thrown if transactions are not available for this database type.

Xapian::InvalidOperationError will be thrown if this is called at an invalid time, such as when a transaction is already in progress.

7.71.3.5 void Xapian::WritableDatabase::cancel_transaction ()

Abort the transaction currently in progress, discarding the pending modifications made to the database.

If an error occurs in this method, an exception will be thrown, but the transaction will be cancelled anyway.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while modifying the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

Xapian::InvalidOperationError will be thrown if a transaction is not currently in progress.

Xapian::UnimplementedError will be thrown if transactions are not available for this database type.

7.71.3.6 void Xapian::WritableDatabase::clear_synonyms (const std::string & *term*) const

Remove all synonyms for a term.

Parameters:

term The term to remove all synonyms for. If the term has no synonyms, no action is taken.

7.71.3.7 void Xapian::WritableDatabase::commit ()

Commit any pending modifications made to the database.

For efficiency reasons, when performing multiple updates to a database it is best (indeed, almost essential) to make as many modifications as memory will permit in a single pass through the database. To ensure this, Xapian batches up modifications.

This method may be called at any time to commit any pending modifications to the database.

If any of the modifications fail, an exception will be thrown and the database will be left in a state in which each separate addition, replacement or deletion operation has either been fully performed or not performed at all: it is then up to the application to work out which operations need to be repeated.

It's not valid to call commit() within a transaction.

Beware of calling commit() too frequently: this will make indexing take much longer.

Note that commit() need not be called explicitly: it will be called automatically when the database is closed, or when a sufficient number of modifications have been made.

By default, this is every 10000 documents added, deleted, or modified. This value is rather conservative, and if you have a machine with plenty of memory, you can improve indexing throughput dramatically by setting XAPIAN_FLUSH_THRESHOLD in the environment to a larger value.

This method was new in Xapian 1.1.0 - in earlier versions it was called flush().

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while modifying the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

7.71.3.8 void Xapian::WritableDatabase::commit_transaction ()

Complete the transaction currently in progress.

If this method completes successfully and this is a flushed transaction, all the database modifications made during the transaction will have been committed to the database.

If an error occurs, an exception will be thrown, and none of the modifications made to the database during the transaction will have been applied to the database.

In all cases the transaction will no longer be in progress.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while modifying the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

Xapian::InvalidOperationError will be thrown if a transaction is not currently in progress.

Xapian::UnimplementedError will be thrown if transactions are not available for this database type.

7.71.3.9 void Xapian::WritableDatabase::delete_document (const std::string & *unique_term*)

Delete any documents indexed by a term from the database.

This method removes any documents indexed by the specified term from the database.

A major use is for convenience when UIDs from another system are mapped to terms in Xapian, although this method has other uses (for example, you could add a "deletion date" term to documents at index time and use this method to delete all documents due for deletion on a particular date).

Parameters:

unique_term The term to remove references to.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

7.71.3.10 void Xapian::WritableDatabase::delete_document (Xapian::docid *did*)

Delete a document from the database.

This method removes the document with the specified document ID from the database.

Note that changes to the database won't be immediately committed to disk; see commit() for more details.

As with all database modification operations, the effect is atomic: the document will either be fully removed, or the document fails to be removed and an exception is thrown (possibly at a later time when commit() is called or the database is closed).

Parameters:

did The document ID of the document to be removed.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

7.71.3.11 void Xapian::WritableDatabase::flush () `[inline]`

Pre-1.1.0 name for commit().

Use commit() instead in new code. This alias may be deprecated in the future.

7.71.3.12 void Xapian::WritableDatabase::operator= (const WritableDatabase & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

Note that only an WritableDatabase may be assigned to an WritableDatabase: an attempt to assign a Database is caught at compile-time.

Parameters:

other The object to copy.

7.71.3.13 void Xapian::WritableDatabase::remove_spelling (const std::string & *word*, Xapian::termcount *freqdec* = 1) const

Remove a word from the spelling dictionary.

The word's frequency is decreased, and if would become zero or less then the word is removed completely.

Parameters:

word The word to remove.

freqdec How much to decrease its frequency by (default 1).

7.71.3.14 void Xapian::WritableDatabase::remove_synonym (const std::string & *term*, const std::string & *synonym*) const

Remove a synonym for a term.

Parameters:

term The term to remove a synonym for.

synonym The synonym to remove. If this isn't currently a synonym for *term*, then no action is taken.

7.71.3.15 Xapian::docid Xapian::WritableDatabase::replace_document (const std::string & *unique_term*, const Xapian::Document & *document*)

Replace any documents matching a term.

This method replaces any documents indexed by the specified term with the specified document. If any documents are indexed by the term, the lowest document ID will be used for the document, otherwise a new document ID will be generated as for add_-document.

One common use is to allow UIDs from another system to easily be mapped to terms in Xapian. Note that this method doesn't automatically add unique_term as a term, so you'll need to call document.add_term(unique_term) first when using replace_-document() in this way.

Note that changes to the database won't be immediately committed to disk; see commit() for more details.

As with all database modification operations, the effect is atomic: the document(s) will either be fully replaced, or the document(s) fail to be replaced and an exception is thrown (possibly at a later time when commit() is called or the database is closed).

Parameters:

unique_term The "unique" term.

document The new document.

Returns:

The document ID that document was given.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

7.71.3.16 void Xapian::WritableDatabase::replace_document (Xapian::docid *did*, const Xapian::Document & *document*)

Replace a given document in the database.

This method replaces the document with the specified document ID. If document ID *did* isn't currently used, the document will be added with document ID *did*.

The monotonic counter used for automatically allocating document IDs is increased so that the next automatically allocated document ID will be did + 1. Be aware that if you use this method to specify a high document ID for a new document, and also use WritableDatabase::add_document(), Xapian may get to a state where this counter wraps around and will be unable to automatically allocate document IDs!

Note that changes to the database won't be immediately committed to disk; see commit() for more details.

As with all database modification operations, the effect is atomic: the document will either be fully replaced, or the document fails to be replaced and an exception is thrown (possibly at a later time when commit() is called or the database is closed).

Parameters:

did The document ID of the document to be replaced.

document The new document.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

7.71.3.17 void Xapian::WritableDatabase::set_metadata (const std::string & *key*, const std::string & *value*)

Set the user-specified metadata associated with a given key.

This method sets the metadata value associated with a given key. If there is already a metadata value stored in the database with the same key, the old value is replaced. If you want to delete an existing item of metadata, just set its value to the empty string.

User-specified metadata allows you to store arbitrary information in the form of (key,tag) pairs.

There's no hard limit on the number of metadata items, or the size of the metadata values. Metadata keys have a limited length, which depends on the backend. We recommend limiting them to 200 bytes. Empty keys are not valid, and specifying one will cause an exception.

Metadata modifications are committed to disk in the same way as modifications to the documents in the database are: i.e., modifications are atomic, and won't be committed to disk immediately (see commit() for more details). This allows metadata to be used to link databases with versioned external resources by storing the appropriate version number in a metadata item.

You can also use the metadata to store arbitrary extra information associated with terms, documents, or postings by encoding the termname and/or document id into the metadata key.

Parameters:

key The key of the metadata item to set.

value The value of the metadata item to set.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

Xapian::InvalidArgumentError will be thrown if the key supplied is empty.

Xapian::UnimplementedError will be thrown if the database backend in use doesn't support user-specified metadata.

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

- xapian/database.h

Chapter 8

File Documentation

8.1 xapian/error.h File Reference

Hierarchy of classes which Xapian can throw as exceptions.

Classes

- class Xapian::Error

  All exceptions thrown by Xapian are subclasses of Xapian::Error.

- class Xapian::LogicError

  The base class for exceptions indicating errors in the program logic.

- class Xapian::RuntimeError

  The base class for exceptions indicating errors only detectable at runtime.

- class Xapian::AssertionError

  AssertionError is thrown if a logical assertion inside Xapian fails.

- class Xapian::InvalidArgumentError

  InvalidArgumentError indicates an invalid parameter value was passed to the API.

- class Xapian::InvalidOperationError

  InvalidOperationError indicates the API was used in an invalid way.

- class Xapian::UnimplementedError

  UnimplementedError indicates an attempt to use an unimplemented feature.

- class Xapian::DatabaseError

  DatabaseError indicates some sort of database related error.

- class Xapian::DatabaseCorruptError

  DatabaseCorruptError indicates database corruption was detected.

- class Xapian::DatabaseCreateError

  DatabaseCreateError indicates a failure to create a database.

- class Xapian::DatabaseLockError

  DatabaseLockError indicates failure to lock a database.

- class Xapian::DatabaseModifiedError

  DatabaseModifiedError indicates a database was modified.

- class Xapian::DatabaseOpeningError

  DatabaseOpeningError indicates failure to open a database.

- class Xapian::DatabaseVersionError

  DatabaseVersionError indicates that a database is in an unsupported format.

- class Xapian::DocNotFoundError

  Indicates an attempt to access a document not present in the database.

- class Xapian::FeatureUnavailableError

  Indicates an attempt to use a feature which is unavailable.

- class Xapian::InternalError

  InternalError indicates a runtime problem of some sort.

- class Xapian::NetworkError

  Indicates a problem communicating with a remote database.

- class Xapian::NetworkTimeoutError

  Indicates a timeout expired while communicating with a remote database.

- class Xapian::QueryParserError

  Indicates a query string can't be parsed.

- class Xapian::SerialisationError

  Indicates an error in the std::string serialisation of an object.

- class Xapian::RangeError

  RangeError indicates an attempt to access outside the bounds of a container.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.1.1 Detailed Description

Hierarchy of classes which Xapian can throw as exceptions.

8.2 xapian/version.h File Reference

Define preprocesor symbols for the library version.

Defines

- #define XAPIAN_ENABLE_VISIBILITY

  The library was compiled with GCC's -fvisibility=hidden option.

- #define XAPIAN_VERSION "1.2.8"

  The version of Xapian as a C string literal.

- #define XAPIAN_MAJOR_VERSION 1

  The major component of the Xapian version.

- #define XAPIAN_MINOR_VERSION 2

  The minor component of the Xapian version.

- #define XAPIAN_REVISION 8

  The revision component of the Xapian version.

- #define XAPIAN_HAS_BRASS_BACKEND 1

  XAPIAN_HAS_BRASS_BACKEND Defined if the brass backend is enabled.

- #define XAPIAN_HAS_CHERT_BACKEND 1

  XAPIAN_HAS_CHERT_BACKEND Defined if the chert backend is enabled.

- #define XAPIAN_HAS_FLINT_BACKEND 1

  XAPIAN_HAS_FLINT_BACKEND Defined if the flint backend is enabled.

- #define XAPIAN_HAS_INMEMORY_BACKEND 1

  XAPIAN_HAS_INMEMORY_BACKEND Defined if the inmemory backend is enabled.

- #define XAPIAN_HAS_REMOTE_BACKEND 1

  XAPIAN_HAS_REMOTE_BACKEND Defined if the remote backend is enabled.

8.2.1 Detailed Description

Define preprocesor symbols for the library version.

8.2.2 Define Documentation

8.2.2.1 #define XAPIAN_MAJOR_VERSION 1

The major component of the Xapian version.

E.g. for Xapian 1.0.14 this would be: 1

8.2.2.2 #define XAPIAN_MINOR_VERSION 2

The minor component of the Xapian version.

E.g. for Xapian 1.0.14 this would be: 0

8.2.2.3 #define XAPIAN_REVISION 8

The revision component of the Xapian version.

E.g. for Xapian 1.0.14 this would be: 14

8.3 xapian.h File Reference

Public interfaces for the Xapian library.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- const char * Xapian::version_string ()

  Report the version string of the library which the program is linked with.

- int Xapian::major_version ()

  Report the major version of the library which the program is linked with.

- int Xapian::minor_version ()

  Report the minor version of the library which the program is linked with.

- int Xapian::revision ()

  Report the revision of the library which the program is linked with.

8.3.1 Detailed Description

Public interfaces for the Xapian library.

8.4 xapian/compactor.h File Reference

Compact a database, or merge and compact several.

Classes

- class Xapian::Compactor

  Compact a database, or merge and compact several.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.4.1 Detailed Description

Compact a database, or merge and compact several.

8.5 xapian/database.h File Reference

API for working with Xapian databases.

Classes

- class Xapian::Database

  This class is used to access a database, or a group of databases.

- class Xapian::WritableDatabase

  This class provides read/write access to a database.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Variables

- const int Xapian::DB_CREATE_OR_OPEN = 1

  Open for read/write; create if no db exists.

- const int Xapian::DB_CREATE = 2

  Create a new database; fail if db exists.

- const int Xapian::DB_CREATE_OR_OVERWRITE = 3

  Overwrite existing db; create if none exists.

- const int Xapian::DB_OPEN = 4

  Open for read/write; fail if no db exists.

8.5.1 Detailed Description

API for working with Xapian databases.

8.6 xapian/dbfactory.h File Reference

Factory functions for constructing Database and WritableDatabase objects.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

- namespace Xapian::Auto

  Database factory functions which determine the database type automatically.

- namespace Xapian::InMemory

  Database factory functions for the inmemory backend.

- namespace Xapian::Brass

  Database factory functions for the brass backend.

- namespace Xapian::Chert

  Database factory functions for the chert backend.

- namespace Xapian::Flint

  Database factory functions for the flint backend.

- namespace Xapian::Remote

  Database factory functions for the remote backend.

Functions

- Database Xapian::Auto::open_stub (const std::string &file)

  Construct a Database object for a stub database file.

- WritableDatabase Xapian::Auto::open_stub (const std::string &file, int action)

  Construct a WritableDatabase object for a stub database file.

- WritableDatabase Xapian::InMemory::open ()

  Construct a WritableDatabase object for a new, empty InMemory database.

- Database Xapian::Brass::open (const std::string &dir)

  Construct a Database object for read-only access to a Brass database.

- WritableDatabase Xapian::Brass::open (const std::string &dir, int action, int block_size=8192)

  Construct a Database object for update access to a Brass database.

- Database Xapian::Chert::open (const std::string &dir)

  Construct a Database object for read-only access to a Chert database.

- WritableDatabase Xapian::Chert::open (const std::string &dir, int action, int block_size=8192)

  Construct a Database object for update access to a Chert database.

- Database Xapian::Flint::open (const std::string &dir)

  Construct a Database object for read-only access to a Flint database.

- WritableDatabase Xapian::Flint::open (const std::string &dir, int action, int block_size=8192)

  Construct a Database object for update access to a Flint database.

- Database Xapian::Remote::open (const std::string &host, unsigned int port, Xapian::timeout timeout=10000, Xapian::timeout connect_timeout=10000)

  Construct a Database object for read-only access to a remote database accessed via a TCP connection.

- WritableDatabase Xapian::Remote::open_writable (const std::string &host, unsigned int port, Xapian::timeout timeout=0, Xapian::timeout connect_-timeout=10000)

  Construct a WritableDatabase object for update access to a remote database accessed via a TCP connection.

- Database Xapian::Remote::open (const std::string &program, const std::string &args, Xapian::timeout timeout=10000)

  Construct a Database object for read-only access to a remote database accessed via a program.

- WritableDatabase Xapian::Remote::open_writable (const std::string &program, const std::string &args, Xapian::timeout timeout=0)

  Construct a WritableDatabase object for update access to a remote database accessed via a program.

8.6.1 Detailed Description

Factory functions for constructing Database and WritableDatabase objects.

8.7 xapian/document.h File Reference

API for working with documents.

Classes

- class Xapian::Document

  A handle representing a document in a Xapian database.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.7.1 Detailed Description

API for working with documents.

8.8 xapian/enquire.h File Reference

API for running queries.

Classes

- class Xapian::MSet

  A match set (MSet).

- class Xapian::MSetIterator

  An iterator pointing to items in an MSet.

- class Xapian::ESet

  Class representing an ordered set of expand terms (an ESet).

- class Xapian::ESetIterator

  Iterate through terms in the ESet.

- class Xapian::RSet

  A relevance set (R-Set).

- class Xapian::MatchDecider

  Base class for matcher decision functor.

- class Xapian::Enquire

  This class provides an interface to the information retrieval system for the purpose of searching.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- bool Xapian::operator== (const MSetIterator &a, const MSetIterator &b)

  Equality test for MSetIterator objects.

- bool Xapian::operator!= (const MSetIterator &a, const MSetIterator &b)

  Inequality test for MSetIterator objects.

- bool Xapian::operator== (const ESetIterator &a, const ESetIterator &b)

  Equality test for ESetIterator objects.

- bool Xapian::operator!= (const ESetIterator &a, const ESetIterator &b)

  Inequality test for ESetIterator objects.

8.8.1 Detailed Description

API for running queries.

8.9 xapian/errorhandler.h File Reference

Decide if a Xapian::Error exception should be ignored.

Classes

- class Xapian::ErrorHandler

  Decide if a Xapian::Error exception should be ignored.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.9.1 Detailed Description

Decide if a Xapian::Error exception should be ignored.

8.10 xapian/expanddecider.h File Reference

Allow rejection of terms during ESet generation.

Classes

- class Xapian::ExpandDecider

  Virtual base class for expand decider functor.

- class Xapian::ExpandDeciderAnd

  ExpandDecider subclass which rejects terms using two ExpandDeciders.

- class Xapian::ExpandDeciderFilterTerms

  ExpandDecider subclass which rejects terms in a specified list.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.10.1 Detailed Description

Allow rejection of terms during ESet generation.

8.11 xapian/keymaker.h File Reference

Build key strings for MSet ordering or collapsing.

Classes

- class Xapian::KeyMaker

  Virtual base class for key making functors.

- class Xapian::MultiValueKeyMaker

  KeyMaker subclass which combines several values.

- class Xapian::Sorter

  Virtual base class for sorter functor.

- class Xapian::MultiValueSorter

  Sorter subclass which sorts by a several values.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.11.1 Detailed Description

Build key strings for MSet ordering or collapsing.

8.12 xapian/matchspy.h File Reference

MatchSpy implementation.

Classes

- class Xapian::MatchSpy

  Abstract base class for match spies.

- class Xapian::ValueCountMatchSpy

  Class for counting the frequencies of values in the matching documents.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.12.1 Detailed Description

MatchSpy implementation.

8.13 xapian/positioniterator.h File Reference

Classes for iterating through position lists.

Classes

- class Xapian::PositionIterator

  An iterator pointing to items in a list of positions.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- bool Xapian::operator== (const PositionIterator &a, const PositionIterator &b)

  Test equality of two PositionIterators.

- bool Xapian::operator!= (const PositionIterator &a, const PositionIterator &b)

  Test inequality of two PositionIterators.

8.13.1 Detailed Description

Classes for iterating through position lists.

8.14 xapian/postingiterator.h File Reference

Classes for iterating through posting lists.

Classes

- class Xapian::PostingIterator

  An iterator pointing to items in a list of postings.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- bool Xapian::operator== (const PostingIterator &a, const PostingIterator &b)

  Test equality of two PostingIterators.

- bool Xapian::operator!= (const PostingIterator &a, const PostingIterator &b)

  Test inequality of two PostingIterators.

8.14.1 Detailed Description

Classes for iterating through posting lists.

8.15 xapian/postingsource.h File Reference

External sources of posting information.

Classes

- class Xapian::PostingSource

  Base class which provides an "external" source of postings.

- class Xapian::ValuePostingSource

  A posting source which generates weights from a value slot.

- class Xapian::ValueWeightPostingSource

  A posting source which reads weights from a value slot.

- class Xapian::DecreasingValueWeightPostingSource

  Read weights from a value which is known to decrease as docid increases.

- class Xapian::ValueMapPostingSource

  A posting source which looks up weights in a map using values as the key.

- class Xapian::FixedWeightPostingSource

  A posting source which returns a fixed weight for all documents.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.15.1 Detailed Description

External sources of posting information.

8.16 xapian/query.h File Reference

Classes for representing a query.

Classes

- class Xapian::Query

  Class representing a query.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.16.1 Detailed Description

Classes for representing a query.

8.17 xapian/queryparser.h File Reference

parsing a user query string to build a Xapian::Query object

Classes

- class Xapian::Stopper

  Base class for stop-word decision functor.

- class Xapian::SimpleStopper

  Simple implementation of Stopper class - this will suit most users.

- struct Xapian::ValueRangeProcessor

  Base class for value range processors.

- class Xapian::StringValueRangeProcessor

  Handle a string range.

- class Xapian::DateValueRangeProcessor

  Handle a date range.

- class Xapian::NumberValueRangeProcessor

  Handle a number range.

- class Xapian::QueryParser

  Build a Xapian::Query object from a user query string.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- std::string Xapian::sortable_serialise (double value)

  Convert a floating point number to a string, preserving sort order.

- double Xapian::sortable_unserialise (const std::string &value)

  Convert a string encoded using sortable_serialise *back to a floating point number.*

8.17.1 Detailed Description

parsing a user query string to build a Xapian::Query object

8.18 xapian/registry.h File Reference

Class for looking up user subclasses during unserialisation.

Classes

- class Xapian::Registry

  Registry for user subclasses.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.18.1 Detailed Description

Class for looking up user subclasses during unserialisation.

8.19 xapian/stem.h File Reference

stemming algorithms

Classes

- struct Xapian::StemImplementation

  Class representing a stemming algorithm implementation.

- class Xapian::Stem

  Class representing a stemming algorithm.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.19.1 Detailed Description

stemming algorithms

8.20 xapian/termgenerator.h File Reference

parse free text and generate terms

Classes

- class Xapian::TermGenerator

  Parses a piece of text and generate terms.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.20.1 Detailed Description

parse free text and generate terms

8.21 xapian/termiterator.h File Reference

Classes for iterating through term lists.

Classes

- class Xapian::TermIterator

  An iterator pointing to items in a list of terms.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- bool Xapian::operator== (const TermIterator &a, const TermIterator &b)

  Equality test for TermIterator objects.

- bool Xapian::operator!= (const TermIterator &a, const TermIterator &b)

  Inequality test for TermIterator objects.

8.21.1 Detailed Description

Classes for iterating through term lists.

8.22 xapian/types.h File Reference

typedefs for Xapian

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Typedefs

- typedef unsigned Xapian::doccount

  A count of documents.

- typedef int Xapian::doccount_diff

  A signed difference between two counts of documents.

- typedef unsigned Xapian::docid

  A unique identifier for a document.

- typedef double Xapian::doclength

  A normalised document length.

- typedef int Xapian::percent

  The percentage score for a document in an MSet.

- typedef unsigned Xapian::termcount

  A counts of terms.

- typedef int Xapian::termcount_diff

  A signed difference between two counts of terms.

- typedef unsigned Xapian::termpos

  A term position within a document or query.

- typedef int Xapian::termpos_diff

  A signed difference between two term positions.

- typedef unsigned Xapian::timeout

  A timeout value in milliseconds.

- typedef unsigned Xapian::valueno

  The number for a value slot in a document.

- typedef int Xapian::valueno_diff

  A signed difference between two value slot numbers.

- typedef double Xapian::weight

  The weight of a document or term.

Variables

- const valueno Xapian::BAD_VALUENO = static_cast<valueno>(-1)

  Reserved value to indicate "no valueno".

8.22.1 Detailed Description

typedefs for Xapian

8.23 xapian/unicode.h File Reference

Unicode and UTF-8 related classes and functions.

Classes

- class Xapian::Utf8Iterator

  An iterator which returns Unicode character values from a UTF-8 encoded string.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

- namespace Xapian::Unicode

  Functions associated with handling Unicode characters.

Enumerations

- enum Xapian::Unicode::category

  Each Unicode character is in exactly one of these categories.

Functions

- unsigned Xapian::Unicode::nonascii_to_utf8 (unsigned ch, char *buf)

  Convert a single non-ASCII Unicode character to UTF-8.

- unsigned Xapian::Unicode::to_utf8 (unsigned ch, char *buf)

  Convert a single Unicode character to UTF-8.

- void Xapian::Unicode::append_utf8 (std::string &s, unsigned ch)

  Append the UTF-8 representation of a single Unicode character to a std::string.

- category Xapian::Unicode::get_category (unsigned ch)

  Return the category which a given Unicode character falls into.

- bool Xapian::Unicode::is_wordchar (unsigned ch)

  Test if a given Unicode character is "word character".

- bool Xapian::Unicode::is_whitespace (unsigned ch)

  Test if a given Unicode character is a whitespace character.

- bool Xapian::Unicode::is_currency (unsigned ch)

  Test if a given Unicode character is a currency symbol.

- unsigned Xapian::Unicode::tolower (unsigned ch)

  Convert a Unicode character to lowercase.

- unsigned Xapian::Unicode::toupper (unsigned ch)

  Convert a Unicode character to uppercase.

- std::string Xapian::Unicode::tolower (const std::string &term)

  Convert a UTF-8 std::string to lowercase.

- std::string Xapian::Unicode::toupper (const std::string &term)

  Convert a UTF-8 std::string to uppercase.

8.23.1 Detailed Description

Unicode and UTF-8 related classes and functions.

8.24 xapian/valueiterator.h File Reference

Class for iterating over document values.

Classes

- class Xapian::ValueIterator

  Class for iterating over document values.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- bool Xapian::operator== (const ValueIterator &a, const ValueIterator &b)

  Equality test for ValueIterator objects.

- bool Xapian::operator!= (const ValueIterator &a, const ValueIterator &b)

  Inequality test for ValueIterator objects.

8.24.1 Detailed Description

Class for iterating over document values.

8.25 xapian/valuesetmatchdecider.h File Reference

MatchDecider subclass for filtering results by value.

Classes

- class Xapian::ValueSetMatchDecider

  MatchDecider filtering results based on whether document values are in a user-defined set.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.25.1 Detailed Description

MatchDecider subclass for filtering results by value.

8.26 xapian/weight.h File Reference

Weighting scheme API.

Classes

- class Xapian::Weight

  Abstract base class for weighting schemes.

- class Xapian::BoolWeight

  Class implementing a "boolean" weighting scheme.

- class Xapian::BM25Weight

  Xapian::Weight subclass implementing the BM25 probabilistic formula.

- class Xapian::TradWeight

  Xapian::Weight subclass implementing the traditional probabilistic formula.

Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

8.26.1 Detailed Description

Weighting scheme API.

Index