xapian-core Reference Manual

1.0.16

Generated by Doxygen 1.5.2

Thu Sep 10 07:50:56 2009

Contents

Chapter 1

xapian-core Namespace Index

1.1 xapian-core Namespace List

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

Chapter 2

xapian-core Hierarchical Index

2.1 xapian-core Class Hierarchy

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

Chapter 3

xapian-core Class Index

3.1 xapian-core Class List

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

Chapter 4

xapian-core File Index

4.1 xapian-core File List

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

Chapter 5

xapian-core Page Index

5.1 xapian-core Related Pages

Here is a list of all related documentation pages:

Chapter 6

xapian-core Namespace Documentation

6.1 Xapian Namespace Reference

The Xapian namespace contains public interfaces for the Xapian library.

Classes

- 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 document in the database - holds data, values, terms, and postings.

- 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 Weight

  Abstract base class for weighting schemes.

- class BoolWeight

  Boolean weighting scheme (everything gets 0).

- class BM25Weight

  BM25 weighting scheme.

- class TradWeight

  Traditional probabilistic weighting scheme.

- 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 PositionIterator

  An iterator pointing to items in a list of positions.

- class PostingIterator

  An iterator pointing to items in a list of postings.

- 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 Sorter

  Virtual base class for sorter functor.

- class MultiValueSorter

  Sorter subclass which sorts by a several values.

- 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

  An iterator pointing to values associated with a document.

Namespaces

- namespace Auto

  Database factory functions which determine the database type automatically.

- namespace Flint

  Database factory functions for the flint backend.

- namespace InMemory

  Database factory functions for the inmemory backend.

- namespace Quartz

  Database factory functions for the quartz backend.

- namespace Remote

  Database factory functions for the remote backend.

- namespace Unicode

  Functions associated with handling Unicode characters.

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 microseconds.

- 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.

- const char * xapian_version_string ()

  For compatibility with Xapian 0.9.5 and earlier.

- int major_version ()

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

- int xapian_major_version ()

  For compatibility with Xapian 0.9.5 and earlier.

- int minor_version ()

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

- int xapian_minor_version ()

  For compatibility with Xapian 0.9.5 and earlier.

- int revision ()

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

- int xapian_revision ()

  For compatibility with Xapian 0.9.5 and earlier.

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 unsigned Xapian::termpos

A term position within a document or query.

6.1.2.9 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".

6.1.2.10 typedef unsigned Xapian::timeout

A timeout value in microseconds.

There are 1 million microseconds in a second, so for example, to set a timeout of 5 seconds use 5000000.

6.1.2.11 typedef unsigned Xapian::valueno

The number for a value slot in a document.

Any value slot number except Xapian::BAD_VALUENO is valid.

6.1.2.12 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".

6.1.2.13 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 bool Xapian::operator!= (const ValueIterator & *a*, const ValueIterator & *b*) [inline]

Inequality test for ValueIterator objects.

6.1.3.4 bool Xapian::operator!= (const TermIterator & *a*, const TermIterator & *b*) [inline]

Inequality test for TermIterator objects.

6.1.3.5 bool Xapian::operator!= (const PostingIterator & *a*, const PostingIterator & *b*) [inline]

Test inequality of two PostingIterators.

6.1.3.6 bool Xapian::operator!= (const PositionIterator & *a*, const PositionIterator & *b*) [inline]

Test inequality of two PositionIterators.

6.1.3.7 bool Xapian::operator!= (const ESetIterator & *a*, const ESetIterator & *b*) [inline]

Inequality test for ESetIterator objects.

6.1.3.8 bool Xapian::operator!= (const MSetIterator & *a*, const MSetIterator & *b*) [inline]

Inequality test for MSetIterator objects.

6.1.3.9 bool Xapian::operator== (const ValueIterator & *a*, const ValueIterator & *b*) [inline]

Equality test for ValueIterator objects.

6.1.3.10 bool Xapian::operator== (const TermIterator & *a*, const TermIterator & *b*) [inline]

Equality test for TermIterator objects.

6.1.3.11 bool Xapian::operator== (const PostingIterator & *a*, const PostingIterator & *b*) [inline]

Test equality of two PostingIterators.

6.1.3.12 bool Xapian::operator== (const PositionIterator & *a*, const PositionIterator & *b*) [inline]

Test equality of two PositionIterators.

6.1.3.13 bool Xapian::operator== (const ESetIterator & *a*, const ESetIterator & *b*) [inline]

Equality test for ESetIterator objects.

6.1.3.14 bool Xapian::operator== (const MSetIterator & *a*, const MSetIterator & *b*) [inline]

Equality test for MSetIterator objects.

6.1.3.15 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.

6.1.3.16 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.

6.1.3.17 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.

6.1.3.18 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.

6.1.3.19 int Xapian::xapian_major_version ()

For compatibility with Xapian 0.9.5 and earlier.

Deprecated

This function is now deprecated, use Xapian::major_version() instead.

6.1.3.20 int Xapian::xapian_minor_version ()

For compatibility with Xapian 0.9.5 and earlier.

Deprecated

This function is now deprecated, use Xapian::minor_version() instead.

6.1.3.21 int Xapian::xapian_revision ()

For compatibility with Xapian 0.9.5 and earlier.

Deprecated

This function is now deprecated, use Xapian::revision() instead.

6.1.3.22 const char* Xapian::xapian_version_string ()

For compatibility with Xapian 0.9.5 and earlier.

Deprecated

This function is now deprecated, use Xapian::version_string() instead.

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.

6.2.1 Detailed Description

Database factory functions which determine the database type automatically.

6.2.2 Function Documentation

6.2.2.1 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::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.

6.3.1 Detailed Description

Database factory functions for the flint backend.

6.3.2 Function Documentation

6.3.2.1 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.

6.3.2.2 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.

6.4 Xapian::InMemory Namespace Reference

Database factory functions for the inmemory backend.

Functions

- WritableDatabase open ()

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

6.4.1 Detailed Description

Database factory functions for the inmemory backend.

6.4.2 Function Documentation

6.4.2.1 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.

6.5 Xapian::Quartz Namespace Reference

Database factory functions for the quartz backend.

Functions

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

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

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

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

6.5.1 Detailed Description

Database factory functions for the quartz backend.

6.5.2 Function Documentation

6.5.2.1 Database Xapian::Quartz::open (const std::string & *dir*)

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

The Quartz backend is deprecated - use the Flint backend instead.

Parameters:

dir pathname of the directory containing the database.

6.5.2.2 Xapian::Quartz::WritableDatabaseopen (const std::string & *dir*, int *action*, int *block_size* = `8192`)

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

The Quartz backend is deprecated - use the Flint backend instead.

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.

6.6 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.

6.6.1 Detailed Description

Database factory functions for the remote backend.

6.6.2 Function Documentation

6.6.2.1 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).

6.6.2.2 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).

6.6.2.3 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).

6.6.2.4 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).

6.7 Xapian::Unicode Namespace Reference

Functions associated with handling Unicode characters.

Enumerations

- enum category {

  UNASSIGNED, **UPPERCASE_LETTER**, **LOWERCASE_LETTER**, **TITLECASE_LETTER**,

  MODIFIER_LETTER, **OTHER_LETTER**, **NON_SPACING_MARK**, **ENCLOSING_MARK**,

  COMBINING_SPACING_MARK, **DECIMAL_DIGIT_NUMBER**, **LETTER_NUMBER**, **OTHER_NUMBER**,

  SPACE_SEPARATOR, **LINE_SEPARATOR**, **PARAGRAPH_SEPARATOR**, **CONTROL**,

  FORMAT, **PRIVATE_USE**, **SURROGATE**, **CONNECTOR_PUNCTUATION**,

  DASH_PUNCTUATION, **OPEN_PUNCTUATION**, **CLOSE_PUNCTUATION**, **INITIAL_QUOTE_PUNCTUATION**,

  FINAL_QUOTE_PUNCTUATION, **OTHER_PUNCTUATION**, **MATH_SYMBOL**, **CURRENCY_SYMBOL**,

  MODIFIER_SYMBOL, **OTHER_SYMBOL** }

  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.

6.7.1 Detailed Description

Functions associated with handling Unicode characters.

6.7.2 Enumeration Type Documentation

6.7.2.1 enum Xapian::Unicode::category

Each Unicode character is in exactly one of these categories.

6.7.3 Function Documentation

6.7.3.1 void Xapian::Unicode::append_utf8 (std::string & *s*, unsigned *ch*) [inline]

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

6.7.3.2 category Xapian::Unicode::get_category (unsigned *ch*) [inline]

Return the category which a given Unicode character falls into.

6.7.3.3 bool Xapian::Unicode::is_currency (unsigned *ch*) [inline]

Test if a given Unicode character is a currency symbol.

6.7.3.4 bool Xapian::Unicode::is_whitespace (unsigned *ch*) [inline]

Test if a given Unicode character is a whitespace character.

6.7.3.5 bool Xapian::Unicode::is_wordchar (unsigned *ch*) [inline]

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

6.7.3.6 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().

The character *ch* (which must be > 128) is written to the buffer *buf* and the length of the resultant UTF-8 character is returned.

NB buf must have space for (at least) 4 bytes.

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

Convert a single Unicode character to UTF-8.

The character *ch* is written to the buffer *buf* and the length of the resultant UTF-8 character is returned.

NB buf must have space for (at least) 4 bytes.

6.7.3.8 std::string Xapian::Unicode::tolower (const std::string & *term*) [inline]

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

6.7.3.9 unsigned Xapian::Unicode::tolower (unsigned *ch*) [inline]

Convert a Unicode character to lowercase.

6.7.3.10 std::string Xapian::Unicode::toupper (const std::string & *term*) [inline]

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

6.7.3.11 unsigned Xapian::Unicode::toupper (unsigned *ch*) [inline]

Convert a Unicode character to uppercase.

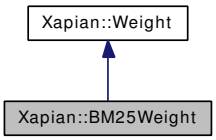
Chapter 7

xapian-core Class Documentation

7.1 Xapian::BM25Weight Class Reference

BM25 weighting scheme.

Inheritance diagram for Xapian::BM25Weight:



Public Member Functions

- BM25Weight (double k1_, double k2_, double k3_, double b_, double min_normlen_)

  Construct a BM25 weight.

- BM25Weight * clone () const

  Return a new weight object of this type.

- std::string name () const

  Name of the weighting scheme.

- std::string serialise () const

  Serialise object parameters into a string.

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

Create object given string serialisation returned by serialise().

- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::doclength len) const

  Get a weight which is part of the sum over terms being performed.

- Xapian::weight get_maxpart () const

  Gets the maximum value that get_sumpart() may return.

- Xapian::weight get_sumextra (Xapian::doclength len) const

  Get an extra weight for a document to add to the sum calculated over the query terms.

- Xapian::weight get_maxextra () const

  Gets the maximum value that get_sumextra() may return.

- bool get_sumpart_needs_doclength () const

  return false if the weight object doesn't need doclength

7.1.1 Detailed Description

BM25 weighting scheme.

BM25 weighting options : The BM25 formula is

$$\frac{k_2.n_q}{1+L_d} + \sum_t \frac{(k_3+1)q_t}{k_3+q_t} \cdot \frac{(k_1+1)f_{t,d}}{k_1((1-b)+bL_d)+f_{t,d}} . w_t$$

where

- w_t is the termweight of term t
- $f_{t,d}$ is the within document frequency of term t in document d
- q_t is the within query frequency of term t
- L_d is the normalised length of document d
- n_q is the size of the query
- k_1, k_2, k_3 and b are user specified parameters

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Xapian::BM25Weight::BM25Weight (double *k1_*, double *k2_*, double *k3_*, double *b_*, double *min_normlen_*) [inline]

Construct a BM25 weight.

Parameters:

k1 governs the importance of within document frequency. Must be >= 0. 0 means ignore wdf. Default is 1.

k2 compensation factor for the high wdf values in large documents. Must be >= 0. 0 means no compensation. Default is 0.

k3 governs the importance of within query frequency. Must be >= 0. 0 means ignore wqf. Default is 1.

b Relative importance of within document frequency and document length. Must be >= 0 and <= 1. Default is 0.5.

min_normlen specifies a cutoff on the minimum value that can be used for a normalised document length - smaller values will be forced up to this cutoff. This prevents very small documents getting a huge bonus weight. Default is 0.5.

7.1.3 Member Function Documentation

7.1.3.1 BM25Weight* Xapian::BM25Weight::clone () const [virtual]

Return a new weight object of this type.

A subclass called FooWeight taking parameters param1 and param2 should implement this as:

virtual FooWeight * clone() const { return new FooWeight(param1, param2); }

Implements Xapian::Weight.

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

Name of the weighting scheme.

If the subclass is called FooWeight, this should return "Foo".

Implements Xapian::Weight.

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

Serialise object parameters into a string.

Implements Xapian::Weight.

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

Create object given string serialisation returned by serialise().

Implements Xapian::Weight.

7.1.3.5 Xapian::weight Xapian::BM25Weight::get_sumpart (Xapian::termcount *wdf*, Xapian::doclength *len*) const [virtual]

Get a weight which is part of the sum over terms being performed.

This returns a weight for a given term and document. These weights are summed to give a total weight for the document.

Parameters:

wdf the within document frequency of the term.

len the (unnormalised) document length.

Implements Xapian::Weight.

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

Gets the maximum value that get_sumpart() may return.

This is used in optimising searches, by having the postlist tree decay appropriately when parts of it can have limited, or no, further effect.

Implements Xapian::Weight.

7.1.3.7 Xapian::weight Xapian::BM25Weight::get_sumextra (Xapian::doclength *len*) const [virtual]

Get an extra weight for a document to add to the sum calculated over the query terms.

This returns a weight for a given document, and is used by some weighting schemes to account for influence such as document length.

Parameters:

len the (unnormalised) document length.

Implements Xapian::Weight.

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

Gets the maximum value that get_sumextra() may return.

This is used in optimising searches.

Implements Xapian::Weight.

7.1.3.9 bool Xapian::BM25Weight::get_sumpart_needs_doclength () const [virtual]

return false if the weight object doesn't need doclength

Reimplemented from Xapian::Weight.

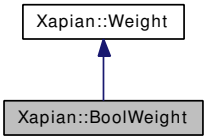
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.2 Xapian::BoolWeight Class Reference

Boolean weighting scheme (everything gets 0).

Inheritance diagram for Xapian::BoolWeight:



Public Member Functions

- BoolWeight * clone () const

  Return a new weight object of this type.

- std::string name () const

  Name of the weighting scheme.

- std::string serialise () const

  Serialise object parameters into a string.

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

  Create object given string serialisation returned by serialise().

- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::doclength len) const

  Get a weight which is part of the sum over terms being performed.

- Xapian::weight get_maxpart () const

  Gets the maximum value that get_sumpart() may return.

- Xapian::weight get_sumextra (Xapian::doclength len) const

  Get an extra weight for a document to add to the sum calculated over the query terms.

- Xapian::weight get_maxextra () const

  Gets the maximum value that get_sumextra() may return.

- bool get_sumpart_needs_doclength () const

  return false if the weight object doesn't need doclength

7.2.1 Detailed Description

Boolean weighting scheme (everything gets 0).

7.2.2 Member Function Documentation

7.2.2.1 BoolWeight* Xapian::BoolWeight::clone () const [virtual]

Return a new weight object of this type.

A subclass called FooWeight taking parameters param1 and param2 should implement this as:

virtual FooWeight * clone() const { return new FooWeight(param1, param2); }

Implements Xapian::Weight.

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

Name of the weighting scheme.

If the subclass is called FooWeight, this should return "Foo".

Implements Xapian::Weight.

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

Serialise object parameters into a string.

Implements Xapian::Weight.

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

Create object given string serialisation returned by serialise().

Implements Xapian::Weight.

7.2.2.5 Xapian::weight Xapian::BoolWeight::get_sumpart (Xapian::termcount *wdf*, Xapian::doclength *len*) const [virtual]

Get a weight which is part of the sum over terms being performed.

This returns a weight for a given term and document. These weights are summed to give a total weight for the document.

Parameters:

wdf the within document frequency of the term.

len the (unnormalised) document length.

Implements Xapian::Weight.

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

Gets the maximum value that get_sumpart() may return.

This is used in optimising searches, by having the postlist tree decay appropriately when parts of it can have limited, or no, further effect.

Implements Xapian::Weight.

7.2.2.7 Xapian::weight Xapian::BoolWeight::get_sumextra (Xapian::doclength *len*) const [virtual]

Get an extra weight for a document to add to the sum calculated over the query terms.

This returns a weight for a given document, and is used by some weighting schemes to account for influence such as document length.

Parameters:

len the (unnormalised) document length.

Implements Xapian::Weight.

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

Gets the maximum value that get_sumextra() may return.

This is used in optimising searches.

Implements Xapian::Weight.

7.2.2.9 bool Xapian::BoolWeight::get_sumpart_needs_doclength () const [virtual]

return false if the weight object doesn't need doclength

Reimplemented from Xapian::Weight.

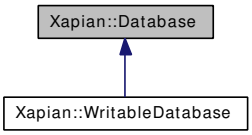
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.3 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 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::doclength 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="") const

  An iterator which returns all terms which have synonyms.

- Xapian::TermIterator synonym_keys_end (const 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="") const

  An iterator which returns all user-specified metadata keys.

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

  Corresponding end iterator to metadata_keys_begin().

7.3.1 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).

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Xapian::Database::Database ()

Create a Database with no databases in.

7.3.2.2 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.

7.3.2.3 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.

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

Copying is allowed.

The internals are reference counted, so copying is cheap.

7.3.3 Member Function Documentation

7.3.3.1 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.

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

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

7.3.3.3 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.

7.3.3.4 virtual std::string Xapian::Database::get_description () const [virtual]

Return a string describing this object.

Reimplemented in Xapian::WritableDatabase.

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

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

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.

7.3.3.6 PostingIterator Xapian::Database::postlist_end (const std::string &) const [inline]

Corresponding end iterator to postlist_begin().

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

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

7.3.3.8 TermIterator Xapian::Database::termlist_end (Xapian::docid) const [inline]

Corresponding end iterator to termlist_begin().

7.3.3.9 bool Xapian::Database::has_positions () const

Does this database have any positional information?

7.3.3.10 PositionIterator Xapian::Database::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.

7.3.3.11 PositionIterator Xapian::Database::positionlist_end (Xapian::docid, const std::string &) const [inline]

Corresponding end iterator to positionlist_begin().

7.3.3.12 TermIterator Xapian::Database::allterms_begin () const

An iterator which runs across all terms in the database.

7.3.3.13 TermIterator Xapian::Database::allterms_end () const [inline]

Corresponding end iterator to allterms_begin().

7.3.3.14 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 network databases.

Parameters:

prefix The prefix to restrict the returned terms to.

7.3.3.15 TermIterator Xapian::Database::allterms_end (const std::string &) const [inline]

Corresponding end iterator to allterms_begin(prefix).

7.3.3.16 Xapian::doccount Xapian::Database::get_doccount () const

Get the number of documents in the database.

7.3.3.17 Xapian::docid Xapian::Database::get_lastdocid () const

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

7.3.3.18 Xapian::doclength Xapian::Database::get_avlength () const

Get the average length of the documents in the database.

7.3.3.19 Xapian::doccount Xapian::Database::get_termfreq (const std::string & *tname*) const

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

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

Check if a given term exists in the database.

Return 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.

7.3.3.21 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.

7.3.3.22 Xapian::doclength Xapian::Database::get_doclength (Xapian::docid *did*) const

Get the length of a document.

7.3.3.23 void Xapian::Database::keep_alive ()

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

7.3.3.24 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 for which to retrieve the data.

Returns:

A Xapian::Document object containing the document data

Exceptions:

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

7.3.3.25 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).

7.3.3.26 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.

7.3.3.27 Xapian::TermIterator Xapian::Database::spellings_end () const [inline]

Corresponding end iterator to spellings_begin().

7.3.3.28 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.

7.3.3.29 Xapian::TermIterator Xapian::Database::synonyms_end (const std::string &) const [inline]

Corresponding end iterator to synonyms_begin(term).

7.3.3.30 Xapian::TermIterator Xapian::Database::synonym_keys_begin (const std::string & *prefix* = "") const

An iterator which returns all terms which have synonyms.

Parameters:

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

7.3.3.31 Xapian::TermIterator Xapian::Database::synonym_keys_end (const std::string & = "") const [inline]

Corresponding end iterator to synonym_keys_begin(prefix).

7.3.3.32 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.

7.3.3.33 Xapian::TermIterator Xapian::Database::metadata_keys_begin (const std::string & *prefix* = "") 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).

7.3.3.34 Xapian::TermIterator Xapian::Database::metadata_keys_end (const std::string & = "") const [inline]

Corresponding end iterator to metadata_keys_begin().

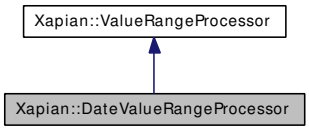
The documentation for this class was generated from the following file:

- xapian/database.h

7.4 Xapian::DateValueRangeProcessor Class Reference

Handle a date range.

Inheritance diagram for Xapian::DateValueRangeProcessor:



Public Member Functions

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

  Constructor.

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

  Check for a valid range of this type.

7.4.1 Detailed Description

Handle a date range.

Begin and end must be dates in a recognised format.

7.4.2 Constructor & Destructor Documentation

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

Constructor.

Parameters:

valno_ 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).

7.4.3 Member Function Documentation

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

Check for a valid range of this type.

If BEGIN..END is a sensible date range, this method returns the value number of range filter on. Otherwise it returns Xapian::BAD_VALUENO.

Implements Xapian::ValueRangeProcessor.

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

- xapian/queryparser.h

7.5 Xapian::Document Class Reference

A document in the database - holds data, values, terms, and postings.

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 valueno) const

  Get value by number.

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

  Add a new value.

- void remove_value (Xapian::valueno valueno)

  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 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.

- ValueIterator 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 get_description () const

  Return a string describing this object.

7.5.1 Detailed Description

A document in the database - holds data, values, terms, and postings.

7.5.2 Constructor & Destructor Documentation

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

Copying is allowed.

The internals are reference counted, so copying is cheap.

7.5.2.2 Xapian::Document::Document ()

Make a new empty Document.

7.5.2.3 Xapian::Document::~Document ()

Destructor.

7.5.3 Member Function Documentation

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

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

7.5.3.2 std::string Xapian::Document::get_value (Xapian::valueno *valueno*) const

Get value by number.

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

Parameters:

valueno The number of the value.

7.5.3.3 void Xapian::Document::add_value (Xapian::valueno *valueno*, 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).

7.5.3.4 void Xapian::Document::remove_value (Xapian::valueno *valueno*)

Remove any value with the given number.

7.5.3.5 void Xapian::Document::clear_values ()

Remove all values associated with the document.

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

Get data stored in the document.

This is a potentially expensive operation, and shouldn't normally be used in a match decider functor. Put data for use by match deciders in a value instead.

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

Set data stored in the document.

7.5.3.8 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.

7.5.3.9 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.

7.5.3.10 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

7.5.3.11 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

7.5.3.12 void Xapian::Document::clear_terms ()

Remove all terms (and postings) from the document.

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

The length of the termlist - i.e.

the number of different terms which index this document.

7.5.3.14 TermIterator Xapian::Document::termlist_begin () const

Iterator for the terms in this document.

7.5.3.15 TermIterator Xapian::Document::termlist_end () const [inline]

Equivalent end iterator for termlist_begin().

7.5.3.16 Xapian::termcount Xapian::Document::values_count () const

Count the values in this document.

7.5.3.17 ValueIterator Xapian::Document::values_begin () const

Iterator for the values in this document.

7.5.3.18 ValueIterator Xapian::Document::values_end () const

Equivalent end iterator for values_begin().

7.5.3.19 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.

7.5.3.20 std::string Xapian::Document::get_description () const

Return a string describing this object.

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

- xapian/document.h

7.6 Xapian::Enquire Class Reference

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

Public Types

- enum **docid_order** { **ASCENDING** = 1, **DESCENDING** = 0, **DONT_CARE** = 2 }

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 set_weighting_scheme (const Weight &weight_)

  Set the weighting scheme to use for queries.

- void set_collapse_key (Xapian::valueno collapse_key)

  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=true)

  Set the sorting to be by value only.

- void set_sort_by_key (Xapian::Sorter *sorter, bool reverse=true)

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

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

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

- void set_sort_by_key_then_relevance (Xapian::Sorter *sorter, bool reverse=true)

  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=true)

  Set the sorting to be by relevance then value.

- void set_sort_by_relevance_then_key (Xapian::Sorter *sorter, bool reverse=true)

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

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

- 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().

- void register_match_decider (const std::string &name, const MatchDecider *mdecider=NULL)

  Register a MatchDecider.

- std::string get_description () const

  Return a string describing this object.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

Static Public Attributes

- static const int **INCLUDE_QUERY_TERMS** = 1
- static const int **USE_EXACT_TERMFREQ** = 2
- static const int include_query_terms = 1

  Deprecated in Xapian 1.0.0, use INCLUDE_QUERY_TERMS instead.

- static const int use_exact_termfreq = 2

  Deprecated in Xapian 1.0.0, use USE_EXACT_TERMFREQ instead.

7.6.1 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.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 Xapian::Enquire::Enquire (const Enquire & *other*)

Copying is allowed (and is cheap).

7.6.2.2 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 initialised Database object is supplied.

7.6.2.3 Xapian::Enquire::~Enquire ()

Close the Xapian::Enquire object.

7.6.3 Member Function Documentation

7.6.3.1 void Xapian::Enquire::operator= (const Enquire & *other*)

Assignment is allowed (and is cheap).

7.6.3.2 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.

7.6.3.3 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.

7.6.3.4 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.

7.6.3.5 void Xapian::Enquire::set_collapse_key (Xapian::valueno *collapse_key*)

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.

The entry returned will be the best entry with that particular value (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 (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.

(default is Xapian::BAD_VALUENO which means no collapsing).

7.6.3.6 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 a very efficient way to perform "sort by date, newest first".

7.6.3.7 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).

7.6.3.8 void Xapian::Enquire::set_sort_by_relevance ()

Set the sorting to be by relevance only.

This is the default.

7.6.3.9 void Xapian::Enquire::set_sort_by_value (Xapian::valueno *sort_key*, bool *reverse* = `true`)

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. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

7.6.3.10 void Xapian::Enquire::set_sort_by_key (Xapian::Sorter * *sorter*, bool *reverse* = `true`)

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. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

7.6.3.11 void Xapian::Enquire::set_sort_by_value_then_relevance (Xapian::valueno *sort_key*, bool *reverse* = `true`)

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. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

7.6.3.12 void Xapian::Enquire::set_sort_by_key_then_relevance (Xapian::Sorter * *sorter*, bool *reverse* = `true`)

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. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

7.6.3.13 void Xapian::Enquire::set_sort_by_relevance_then_value (Xapian::valueno *sort_key*, bool *reverse* = `true`)

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. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

7.6.3.14 void Xapian::Enquire::set_sort_by_relevance_then_key (Xapian::Sorter * *sorter*, bool *reverse* = `true`)

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. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

7.6.3.15 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.

Returns:

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

Exceptions:

Xapian::InvalidArgumentError See class documentation.

7.6.3.16 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.

7.6.3.17 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.

7.6.3.18 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.

7.6.3.19 TermIterator Xapian::Enquire::get_matching_terms_end (Xapian::docid) const [inline]

End iterator corresponding to get_matching_terms_begin().

7.6.3.20 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.

7.6.3.21 TermIterator Xapian::Enquire::get_matching_terms_end (const MSetIterator &) const [inline]

End iterator corresponding to get_matching_terms_begin().

7.6.3.22 void Xapian::Enquire::register_match_decider (const std::string & *name*, const MatchDecider * *mdecider* = NULL)

Register a MatchDecider.

This is used to associate a name with a matchdecider.

Deprecated

This method is deprecated. It was added long ago with the intention that it would allow the remote backend to support use of MatchDecider objects, but there's a better approach.

Parameters:

name The name to register this matchdecider as.

mdecider The matchdecider. If omitted, then remove any matchdecider registered with this name.

7.6.3.23 std::string Xapian::Enquire::get_description () const

Return a string describing this object.

7.6.4 Member Data Documentation

7.6.4.1 const int Xapian::Enquire::include_query_terms = 1 [static]

Deprecated in Xapian 1.0.0, use INCLUDE_QUERY_TERMS instead.

7.6.4.2 const int Xapian::Enquire::use_exact_termfreq = 2 [static]

Deprecated in Xapian 1.0.0, use USE_EXACT_TERMFREQ instead.

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

- xapian/enquire.h

7.7 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.

7.7.1 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).

7.7.2 Constructor & Destructor Documentation

7.7.2.1 Xapian::ErrorHandler::ErrorHandler () [inline]

Default constructor.

7.7.2.2 virtual Xapian::ErrorHandler::~ErrorHandler () [virtual]

We require a virtual destructor because we have virtual methods.

7.7.3 Member Function Documentation

7.7.3.1 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

7.8 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.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

7.8.1 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().

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Xapian::ESet::ESet ()

Construct an empty ESet.

7.8.2.2 Xapian::ESet::~ESet ()

Destructor.

7.8.2.3 Xapian::ESet::ESet (const ESet & *other*)

Copying is allowed (and is cheap).

7.8.3 Member Function Documentation

7.8.3.1 void Xapian::ESet::operator= (const ESet & *other*)

Assignment is allowed (and is cheap).

7.8.3.2 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()

7.8.3.3 Xapian::termcount Xapian::ESet::size () const

The number of terms in this E-Set.

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

Required to allow use as an STL container.

7.8.3.5 bool Xapian::ESet::empty () const

Test if this E-Set is empty.

7.8.3.6 void Xapian::ESet::swap (ESet & *other*)

Swap the E-Set we point to with another.

7.8.3.7 ESetIterator Xapian::ESet::begin () const

Iterator for the terms in this E-Set.

7.8.3.8 ESetIterator Xapian::ESet::end () const

End iterator corresponding to begin().

7.8.3.9 ESetIterator Xapian::ESet::back () const

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

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

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

7.8.3.11 std::string Xapian::ESet::get_description () const

Return a string describing this object.

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

- xapian/enquire.h

7.9 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.

7.9.1 Detailed Description

Iterate through terms in the ESet.

7.9.2 Member Typedef Documentation

7.9.2.1 typedef std::bidirectional_iterator_tag Xapian::ESetIterator::iterator_category

Allow use as an STL iterator.

7.9.2.2 typedef std::string Xapian::ESetIterator::value_type

Allow use as an STL iterator.

7.9.2.3 typedef Xapian::termcount_diff Xapian::ESetIterator::difference_type

Allow use as an STL iterator.

7.9.2.4 typedef std::string* Xapian::ESetIterator::pointer

Allow use as an STL iterator.

7.9.2.5 typedef std::string& Xapian::ESetIterator::reference

Allow use as an STL iterator.

7.9.3 Constructor & Destructor Documentation

7.9.3.1 Xapian::ESetIterator::ESetIterator () [inline]

Create an uninitialised iterator; this cannot be used, but is convenient syntactically.

7.9.3.2 Xapian::ESetIterator::ESetIterator (const ESetIterator & *other*) [inline]

Copying is allowed (and is cheap).

7.9.4 Member Function Documentation

7.9.4.1 void Xapian::ESetIterator::operator= (const ESetIterator & *other*) [inline]

Assignment is allowed (and is cheap).

7.9.4.2 ESetIterator& Xapian::ESetIterator::operator++ () [inline]

Advance the iterator.

7.9.4.3 ESetIterator Xapian::ESetIterator::operator++ (int) [inline]

Advance the iterator (postfix variant).

7.9.4.4 ESetIterator& Xapian::ESetIterator::operator– () [inline]

Decrement the iterator.

7.9.4.5 ESetIterator Xapian::ESetIterator::operator– (int) [inline]

Decrement the iterator (postfix variant).

7.9.4.6 const std::string& Xapian::ESetIterator::operator * () const

Get the term for the current position.

7.9.4.7 Xapian::weight Xapian::ESetIterator::get_weight () const

Get the weight of the term at the current position.

7.9.4.8 std::string Xapian::ESetIterator::get_description () const

Return a string describing this object.

7.9.5 Friends And Related Function Documentation

7.9.5.1 bool operator== (const ESetIterator & *a*, const ESetIterator & *b*) [friend]

Equality test for ESetIterator objects.

7.9.5.2 bool operator!= (const ESetIterator & *a*, const ESetIterator & *b*) [friend]

Inequality test for ESetIterator objects.

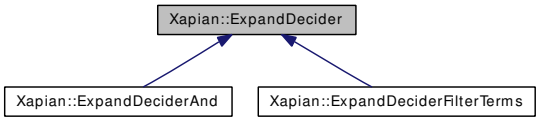
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.10 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.

7.10.1 Detailed Description

Virtual base class for expand decider functor.

7.10.2 Constructor & Destructor Documentation

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

Virtual destructor, because we have virtual methods.

7.10.3 Member Function Documentation

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

Do we want this term in the ESet?

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

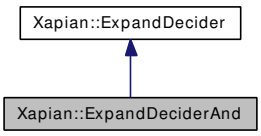
The documentation for this class was generated from the following file:

- xapian/expanddecider.h

7.11 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?

7.11.1 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.

7.11.2 Constructor & Destructor Documentation

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

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

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

Compatibility method.

7.11.3 Member Function Documentation

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

Do we want this term in the ESet?

Implements Xapian::ExpandDecider.

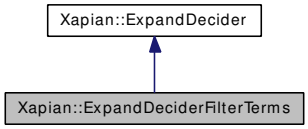
The documentation for this class was generated from the following file:

- xapian/expanddecider.h

7.12 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?

7.12.1 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.

7.12.2 Constructor & Destructor Documentation

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

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

reject_begin and *reject_end* can be any input_iterator type which returns std::string or char * (e.g. TermIterator or char **).

7.12.3 Member Function Documentation

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

Do we want this term in the ESet?

Implements Xapian::ExpandDecider.

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

- xapian/expanddecider.h

7.13 Xapian::MatchDecider Class Reference

Base class for matcher decision functor.

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.

7.13.1 Detailed Description

Base class for matcher decision functor.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 virtual Xapian::MatchDecider::~MatchDecider () [virtual]

Destructor.

7.13.3 Member Function Documentation

7.13.3.1 virtual bool Xapian::MatchDecider::operator() (const Xapian::Document & *doc*) const [pure virtual]

Decide whether we want this document to be in the MSet.

Return true if the document is acceptable, or false if the document should be excluded from the MSet.

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

- xapian/enquire.h

7.14 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** (MSet::Internal *internal_)
- 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::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 terms 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.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

7.14.1 Detailed Description

A match set (MSet).

This class represents (a portion of) the results of a query.

7.14.2 Member Typedef Documentation

7.14.2.1 typedef MSetIterator Xapian::MSet::value_type

Allow use as an STL container.

7.14.2.2 typedef MSetIterator Xapian::MSet::iterator

Allow use as an STL container.

7.14.2.3 typedef MSetIterator Xapian::MSet::const_iterator

Allow use as an STL container.

7.14.2.4 typedef MSetIterator& Xapian::MSet::reference

Allow use as an STL container.

7.14.2.5 typedef MSetIterator& Xapian::MSet::const_reference

Allow use as an STL container.

7.14.2.6 typedef MSetIterator* Xapian::MSet::pointer

Allow use as an STL container.

7.14.2.7 typedef Xapian::doccount_diff Xapian::MSet::difference_type

Allow use as an STL container.

7.14.2.8 typedef Xapian::doccount Xapian::MSet::size_type

Allow use as an STL container.

7.14.3 Constructor & Destructor Documentation

7.14.3.1 Xapian::MSet::MSet ()

Create an empty Xapian::MSet.

7.14.3.2 Xapian::MSet::~MSet ()

Destroy a Xapian::MSet.

7.14.3.3 Xapian::MSet::MSet (const MSet & *other*)

Copying is allowed (and is cheap).

7.14.4 Member Function Documentation

7.14.4.1 void Xapian::MSet::operator= (const MSet & *other*)

Assignment is allowed (and is cheap).

7.14.4.2 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.

7.14.4.3 void Xapian::MSet::fetch (const MSetIterator & *item*) const

Fetch the single item specified.

7.14.4.4 void Xapian::MSet::fetch () const

Fetch all the items in the MSet.

7.14.4.5 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.

7.14.4.6 Xapian::percent Xapian::MSet::convert_to_percent (const MSetIterator & *it*) const

Return the percentage score for a particular item.

7.14.4.7 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.

Exceptions:

Xapian::InvalidArgumentError is thrown if the term was not in the query.

7.14.4.8 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.

7.14.4.9 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.

7.14.4.10 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.

7.14.4.11 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.

7.14.4.12 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.

7.14.4.13 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.

7.14.4.14 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.

7.14.4.15 Xapian::doccount Xapian::MSet::size () const

The number of items in this MSet.

7.14.4.16 Xapian::doccount Xapian::MSet::max_size () const [inline]

Required to allow use as an STL container.

7.14.4.17 bool Xapian::MSet::empty () const

Test if this MSet is empty.

7.14.4.18 void Xapian::MSet::swap (MSet & *other*)

Swap the MSet we point to with another.

7.14.4.19 MSetIterator Xapian::MSet::begin () const

Iterator for the terms in this MSet.

7.14.4.20 MSetIterator Xapian::MSet::end () const

End iterator corresponding to begin().

7.14.4.21 MSetIterator Xapian::MSet::back () const

Iterator pointing to the last element of this MSet.

7.14.4.22 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.

7.14.4.23 std::string Xapian::MSet::get_description () const

Return a string describing this object.

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

- xapian/enquire.h

7.15 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.

7.15.1 Detailed Description

An iterator pointing to items in an MSet.

This is used for access to individual results of a match.

7.15.2 Member Typedef Documentation

7.15.2.1 typedef std::bidirectional_iterator_tag Xapian::MSetIterator::iterator_category

Allow use as an STL iterator.

7.15.2.2 typedef Xapian::docid Xapian::MSetIterator::value_type

Allow use as an STL iterator.

7.15.2.3 typedef Xapian::doccount_diff Xapian::MSetIterator::difference_type

Allow use as an STL iterator.

7.15.2.4 typedef Xapian::docid* Xapian::MSetIterator::pointer

Allow use as an STL iterator.

7.15.2.5 typedef Xapian::docid& Xapian::MSetIterator::reference

Allow use as an STL iterator.

7.15.3 Constructor & Destructor Documentation

7.15.3.1 Xapian::MSetIterator::MSetIterator () [inline]

Create an uninitialised iterator; this cannot be used, but is convenient syntactically.

7.15.3.2 Xapian::MSetIterator::MSetIterator (const MSetIterator & *other*) [inline]

Copying is allowed (and is cheap).

7.15.4 Member Function Documentation

7.15.4.1 void Xapian::MSetIterator::operator= (const MSetIterator & *other*) [inline]

Assignment is allowed (and is cheap).

7.15.4.2 MSetIterator& Xapian::MSetIterator::operator++ () [inline]

Advance the iterator.

7.15.4.3 MSetIterator Xapian::MSetIterator::operator++ (int) [inline]

Advance the iterator (postfix variant).

7.15.4.4 MSetIterator& Xapian::MSetIterator::operator– () [inline]

Decrement the iterator.

7.15.4.5 MSetIterator Xapian::MSetIterator::operator– (int) [inline]

Decrement the iterator (postfix variant).

7.15.4.6 Xapian::docid Xapian::MSetIterator::operator ∗ () const

Get the document ID for the current position.

7.15.4.7 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.

7.15.4.8 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.

7.15.4.9 Xapian::weight Xapian::MSetIterator::get_weight () const

Get the weight of the document at the current position.

7.15.4.10 std::string Xapian::MSetIterator::get_collapse_key () const

Get the collapse key for this document.

7.15.4.11 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).

7.15.4.12 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).

7.15.4.13 std::string Xapian::MSetIterator::get_description () const

Return a string describing this object.

7.15.5 Friends And Related Function Documentation

7.15.5.1 bool operator== (const MSetIterator & *a*, const MSetIterator & *b*) [friend]

Equality test for MSetIterator objects.

7.15.5.2 bool operator!= (const MSetIterator & *a*, const MSetIterator & *b*) [friend]

Inequality test for MSetIterator objects.

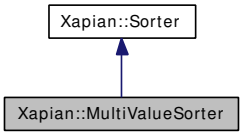
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.16 Xapian::MultiValueSorter Class Reference

Sorter subclass which sorts by a several values.

Inheritance diagram for Xapian::MultiValueSorter:



Public Member Functions

- template<class Iterator>
  MultiValueSorter (Iterator begin, Iterator end)
- virtual std::string operator() (const Xapian::Document &doc) const

  This method takes a Document object and builds a sort key from it.
- void **add** (Xapian::valueno valno, bool forward=true)

7.16.1 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.

7.16.2 Member Function Documentation

7.16.2.1 virtual std::string Xapian::MultiValueSorter::operator() (const Xapian::Document & *doc*) const [virtual]

This method takes a Document object and builds a sort key from it.

Documents are then ordered by a string compare on the sort keys.

Implements Xapian::Sorter.

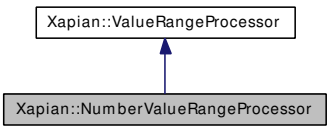
The documentation for this class was generated from the following file:

- xapian/sorter.h

7.17 Xapian::NumberValueRangeProcessor Class Reference

Handle a number range.

Inheritance diagram for Xapian::NumberValueRangeProcessor:



Public Member Functions

- NumberValueRangeProcessor (Xapian::valueno valno_)

  Constructor.

- NumberValueRangeProcessor (Xapian::valueno valno_, const std::string &str_, bool prefix_=true)

  Constructor.

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

  Check for a valid range of this type.

7.17.1 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).

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno *valno_*) [inline]

Constructor.

Parameters:

valno_ The value number to return from operator().

7.17.2.2 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno *valno_*, const std::string & *str_*, bool *prefix_* = `true`) `[inline]`

Constructor.

Parameters:

valno_ 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.

7.17.3 Member Function Documentation

7.17.3.1 Xapian::valueno Xapian::NumberValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) `[virtual]`

Check for a valid range of this type.

If BEGIN..END is a valid numeric value range, and has the appropriate prefix or suffix (if specified) required for this NumberValueRangeProcessor, this method returns the value number of range filter on, and sets begin and end to the appropriate serialised values needed to delimit the range. Otherwise it returns Xapian::BAD_VALUENO.

Implements Xapian::ValueRangeProcessor.

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

- xapian/queryparser.h

7.18 Xapian::PositionIterator Class Reference

An iterator pointing to items in a list of positions.

Public Types

- typedef std::input_iterator_tag **iterator_category**
- typedef Xapian::termpos **value_type**
- typedef Xapian::termpos_diff **difference_type**
- typedef Xapian::termpos * **pointer**
- typedef Xapian::termpos & **reference**

Public Member Functions

- **PositionIterator** (Internal *internal_)
- 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.

- Xapian::termpos **operator** * () const
- PositionIterator & **operator++** ()
- TermPosWrapper **operator++** (int)
- void **skip_to** (Xapian::termpos pos)
- 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.

7.18.1 Detailed Description

An iterator pointing to items in a list of positions.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Xapian::PositionIterator::PositionIterator ()

Default constructor - for declaring an uninitialised iterator.

7.18.2.2 Xapian::PositionIterator::~PositionIterator ()

Destructor.

7.18.2.3 Xapian::PositionIterator::PositionIterator (const PositionIterator & *o*)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

7.18.3 Member Function Documentation

7.18.3.1 void Xapian::PositionIterator::operator= (const PositionIterator & *o*)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

7.18.3.2 std::string Xapian::PositionIterator::get_description () const

Return a string describing this object.

7.18.4 Friends And Related Function Documentation

7.18.4.1 bool operator== (const PositionIterator & *a*, const PositionIterator & *b*) [friend]

Test equality of two PositionIterators.

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

- xapian/positioniterator.h

7.19 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.
- PostingIterator & **operator++** ()
- DocIDWrapper **operator++** (int)
- void skip_to (Xapian::docid did)
  Skip the iterator to document did, or the first document after did if did isn't in the list of documents being iterated.
- Xapian::docid operator * () const
  Get the document id at the current position in the postlist.
- Xapian::doclength 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.

7.19.1 Detailed Description

An iterator pointing to items in a list of postings.

7.19.2 Member Typedef Documentation

7.19.2.1 typedef std::input_iterator_tag Xapian::PostingIterator::iterator_category

Allow use as an STL iterator.

7.19.2.2 typedef Xapian::docid Xapian::PostingIterator::value_type

Allow use as an STL iterator.

7.19.2.3 typedef Xapian::doccount_diff Xapian::PostingIterator::difference_type

Allow use as an STL iterator.

7.19.2.4 typedef Xapian::docid* Xapian::PostingIterator::pointer

Allow use as an STL iterator.

7.19.2.5 typedef Xapian::docid& Xapian::PostingIterator::reference

Allow use as an STL iterator.

7.19.3 Constructor & Destructor Documentation

7.19.3.1 Xapian::PostingIterator::PostingIterator ()

Default constructor - for declaring an uninitialised iterator.

7.19.3.2 Xapian::PostingIterator::~PostingIterator ()

Destructor.

7.19.3.3 Xapian::PostingIterator::PostingIterator (const PostingIterator & *other*)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

7.19.4 Member Function Documentation

7.19.4.1 void Xapian::PostingIterator::operator= (const PostingIterator & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

7.19.4.2 void Xapian::PostingIterator::skip_to (Xapian::docid *did*)

Skip the iterator to document did, or the first document after did if did isn't in the list of documents being iterated.

7.19.4.3 Xapian::docid Xapian::PostingIterator::operator * () const

Get the document id at the current position in the postlist.

7.19.4.4 Xapian::doclength 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.

7.19.4.5 Xapian::termcount Xapian::PostingIterator::get_wdf () const

Get the within document frequency of the document at the current position in the postlist.

7.19.4.6 PositionIterator Xapian::PostingIterator::positionlist_begin () const

Return PositionIterator pointing to start of positionlist for current document.

7.19.4.7 PositionIterator Xapian::PostingIterator::positionlist_end () const [inline]

Return PositionIterator pointing to end of positionlist for current document.

7.19.4.8 std::string Xapian::PostingIterator::get_description () const

Return a string describing this object.

7.19.5 Friends And Related Function Documentation

7.19.5.1 bool operator== (const PostingIterator & *a*, const PostingIterator & *b*) [friend]

Test equality of two PostingIterators.

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

- xapian/postingiterator.h

7.20 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 }
  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)

Apply the specified operator to a single Xapian::Query object.

- 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 valno, const std::string &begin, const std::string &end)

  Construct a value range query on a document value.

- Query (Query::op op_, Xapian::valueno valno, const std::string &value)

  Construct a value comparison query on a document value.

- 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 get_description () const

  Return a string describing this object.

Static Public Attributes

- static Xapian::Query MatchAll

  A query which matches all documents in the database.

- static Xapian::Query MatchNothing

  A query which matches no documents.

7.20.1 Detailed Description

Class representing a query.

Queries are represented as a tree of objects.

7.20.2 Member Enumeration Documentation

7.20.2.1 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.

7.20.3 Constructor & Destructor Documentation

7.20.3.1 Xapian::Query::Query (const Query & *copyme*)

Copy constructor.

7.20.3.2 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.

7.20.3.3 Xapian::Query::~Query ()

Destructor.

7.20.3.4 Xapian::Query::Query (const std::string & *tname_*, Xapian::termcount *wqf_* = 1, Xapian::termpos *pos_* = 0)

A query consisting of a single term.

7.20.3.5 Xapian::Query::Query (Query::op *op_*, const Query & *left*, const Query & *right*)

A query consisting of two subqueries, opp-ed together.

7.20.3.6 Xapian::Query::Query (Query::op *op_*, const std::string & *left*, const std::string & *right*)

A query consisting of two termnames opp-ed together.

7.20.3.7 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, 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.

7.20.3.8 Xapian::Query::Query (Query::op *op_*, Xapian::Query *q*)

Apply the specified operator to a single Xapian::Query object.

Deprecated

This method is deprecated because it isn't useful, since none of the current query operators can be usefully applied to a single subquery without a parameter value.

7.20.3.9 Xapian::Query::Query (Query::op *op_*, Xapian::Query *q*, double *parameter*)

Apply the specified operator to a single Xapian::Query object, with a double parameter.

7.20.3.10 Xapian::Query::Query (Query::op *op_*, Xapian::valueno *valno*, 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 *valno* 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.

valno The slot number to get the value from.

begin The start of the range.

end The end of the range.

7.20.3.11 Xapian::Query::Query (Query::op *op_*, Xapian::valueno *valno*, 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 *valno* 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.

valno The slot number to get the value from.

value The value to compare.

7.20.4 Member Function Documentation

7.20.4.1 Query& Xapian::Query::operator= (const Query & *copyme*)

Assignment.

7.20.4.2 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().

7.20.4.3 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.

7.20.4.4 TermIterator Xapian::Query::get_terms_end () const [inline]

Return a Xapian::TermIterator to the end of the list of terms in the query.

7.20.4.5 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).

7.20.4.6 std::string Xapian::Query::get_description () const

Return a string describing this object.

7.20.5 Member Data Documentation

7.20.5.1 Xapian::Query Xapian::Query::MatchAll [static]

A query which matches all documents in the database.

7.20.5.2 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

7.21 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.

- enum **stem_strategy** { **STEM_NONE**, **STEM_SOME**, **STEM_ALL** }

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.

- Query parse_query (const std::string &query_string, unsigned flags=FLAG_PHRASE|FLAG_BOOLEAN|FLAG_LOVEHATE, const std::string &default_prefix="")

  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)

  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 **stoplist_end** () const
- TermIterator unstem_begin (const std::string &term) const

  Iterate over unstemmed forms of the given (stemmed) term used in the query.

- TermIterator **unstem_end** (const std::string &) const
- 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.

7.21.1 Detailed Description

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

7.21.2 Member Enumeration Documentation

7.21.2.1 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∗).

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.

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()*.

Added in Xapian 1.0.11.

7.21.3 Constructor & Destructor Documentation

7.21.3.1 Xapian::QueryParser::QueryParser (const QueryParser & *o*)

Copy constructor.

7.21.3.2 Xapian::QueryParser::QueryParser ()

Default constructor.

7.21.3.3 Xapian::QueryParser::~QueryParser ()

Destructor.

7.21.4 Member Function Documentation

7.21.4.1 QueryParser& Xapian::QueryParser::operator= (const QueryParser & *o*)

Assignment.

7.21.4.2 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.

7.21.4.3 void Xapian::QueryParser::set_stemming_strategy (stem_strategy *strategy*)

Set the stemming strategy.

This controls how the query parser will apply the stemming algorithm. The default value is STEM_NONE. The possible values are:

- STEM_NONE: Don't perform any stemming.
- 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).

Note that the stemming algorithm is only applied to words in probabilistic fields - boolean filter terms are never stemmed.

7.21.4.4 void Xapian::QueryParser::set_stopper (const Stopper * *stop* = NULL)

Set the stopper.

7.21.4.5 void Xapian::QueryParser::set_default_op (Query::op *default_op*)

Set the default operator.

This operator is used 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.

7.21.4.6 Query::op Xapian::QueryParser::get_default_op () const

Get the current default operator.

7.21.4.7 void Xapian::QueryParser::set_database (const Database & *db*)

Specify the database being searched.

7.21.4.8 Query Xapian::QueryParser::parse_query (const std::string & *query_string*, unsigned *flags* = FLAG_PHRASE|FLAG_BOOLEAN|FLAG_LOVEHATE, const std::string & *default_prefix* = "")

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.

7.21.4.9 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

7.21.4.10 void Xapian::QueryParser::add_boolean_prefix (const std::string & *field*, const std::string & *prefix*)

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

7.21.4.11 TermIterator Xapian::QueryParser::stoplist_begin () const

Iterate over terms omitted from the query as stopwords.

7.21.4.12 TermIterator Xapian::QueryParser::unstem_begin (const std::string & *term*) const

Iterate over unstemmed forms of the given (stemmed) term used in the query.

7.21.4.13 void Xapian::QueryParser::add_valuerangeprocessor (Xapian::ValueRangeProcessor * *vrproc*)

Register a ValueRangeProcessor.

7.21.4.14 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.

7.21.4.15 std::string Xapian::QueryParser::get_description () const

Return a string describing this object.

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

- xapian/queryparser.h

7.22 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.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

7.22.1 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.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 Xapian::RSet::RSet (const RSet & *rset*)

Copy constructor.

7.22.2.2 Xapian::RSet::RSet ()

Default constructor.

7.22.2.3 Xapian::RSet::~RSet ()

Destructor.

7.22.3 Member Function Documentation

7.22.3.1 void Xapian::RSet::operator= (const RSet & *rset*)

Assignment operator.

7.22.3.2 Xapian::doccount Xapian::RSet::size () const

The number of documents in this R-Set.

7.22.3.3 bool Xapian::RSet::empty () const

Test if this R-Set is empty.

7.22.3.4 void Xapian::RSet::add_document (Xapian::docid *did*)

Add a document to the relevance set.

7.22.3.5 void Xapian::RSet::add_document (const Xapian::MSetIterator & *i*) [inline]

Add a document to the relevance set.

7.22.3.6 void Xapian::RSet::remove_document (Xapian::docid *did*)

Remove a document from the relevance set.

7.22.3.7 void Xapian::RSet::remove_document (const Xapian::MSetIterator & *i*) [inline]

Remove a document from the relevance set.

7.22.3.8 bool Xapian::RSet::contains (Xapian::docid *did*) const

Test if a given document in the relevance set.

7.22.3.9 bool Xapian::RSet::contains (const Xapian::MSetIterator & *i*) const [inline]

Test if a given document in the relevance set.

7.22.3.10 std::string Xapian::RSet::get_description () const

Return a string describing this object.

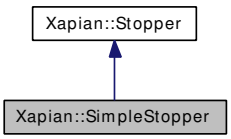
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.23 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 ~SimpleStopper ()

  Destructor.

- virtual std::string get_description () const

  Return a string describing this object.

7.23.1 Detailed Description

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

7.23.2 Constructor & Destructor Documentation

7.23.2.1 Xapian::SimpleStopper::SimpleStopper () [inline]

Default constructor.

7.23.2.2 template<class Iterator> Xapian::SimpleStopper::SimpleStopper (Iterator *begin*, Iterator *end*) [inline]

Initialise from a pair of iterators.

7.23.2.3 virtual Xapian::SimpleStopper::~SimpleStopper () [inline, virtual]

Destructor.

7.23.3 Member Function Documentation

7.23.3.1 void Xapian::SimpleStopper::add (const std::string & *word*) [inline]

Add a single stop word.

7.23.3.2 virtual bool Xapian::SimpleStopper::operator() (const std::string & *term*) const [inline, virtual]

Is term a stop-word?

Implements Xapian::Stopper.

7.23.3.3 virtual std::string Xapian::SimpleStopper::get_description () const [virtual]

Return a string describing this object.

Reimplemented from Xapian::Stopper.

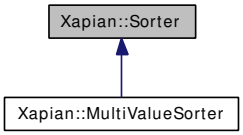
The documentation for this class was generated from the following file:

- xapian/queryparser.h

7.24 Xapian::Sorter Class Reference

Virtual base class for sorter functor.

Inheritance diagram for Xapian::Sorter:



Public Member Functions

- virtual std::string operator() (const Xapian::Document &doc) const=0

  This method takes a Document object and builds a sort key from it.

- virtual ~Sorter ()

  Virtual destructor, because we have virtual methods.

7.24.1 Detailed Description

Virtual base class for sorter functor.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 virtual Xapian::Sorter::~Sorter () [virtual]

Virtual destructor, because we have virtual methods.

7.24.3 Member Function Documentation

7.24.3.1 virtual std::string Xapian::Sorter::operator() (const Xapian::Document & *doc*) const [pure virtual]

This method takes a Document object and builds a sort key from it.

Documents are then ordered by a string compare on the sort keys.

Implemented in Xapian::MultiValueSorter.

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

- xapian/sorter.h

7.25 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 ()

  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.

7.25.1 Detailed Description

Class representing a stemming algorithm.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 Xapian::Stem::Stem (const Stem & *o*)

Copy constructor.

7.25.2.2 Xapian::Stem::Stem ()

Construct a Xapian::Stem object which doesn't change terms.

Equivalent to Stem("none").

7.25.2.3 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)
- italian (it)
- norwegian (no)
- portuguese (pt)
- russian (ru)
- spanish (es)
- swedish (sv)

Exceptions:

Xapian::InvalidArgumentError is thrown if language isn't recognised.

7.25.2.4 Xapian::Stem::~Stem ()

Destructor.

7.25.3 Member Function Documentation

7.25.3.1 void Xapian::Stem::operator= (const Stem & *o*)

Assignment.

7.25.3.2 std::string Xapian::Stem::operator() (const std::string & *word*) const

Stem a word.

Parameters:

word a word to stem.

Returns:

the stem

7.25.3.3 std::string Xapian::Stem::get_description () const

Return a string describing this object.

7.25.3.4 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.

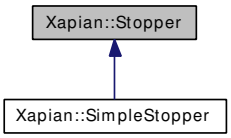
The documentation for this class was generated from the following file:

- xapian/stem.h

7.26 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.

7.26.1 Detailed Description

Base class for stop-word decision functor.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 virtual Xapian::Stopper::~Stopper () [inline, virtual]

Class has virtual methods, so provide a virtual destructor.

7.26.3 Member Function Documentation

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

Is term a stop-word?

Implemented in Xapian::SimpleStopper.

7.26.3.2 virtual std::string Xapian::Stopper::get_description () const [virtual]

Return a string describing this object.

Reimplemented in Xapian::SimpleStopper.

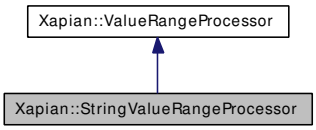
The documentation for this class was generated from the following file:

- xapian/queryparser.h

7.27 Xapian::StringValueRangeProcessor Class Reference

Handle a string range.

Inheritance diagram for Xapian::StringValueRangeProcessor:



Public Member Functions

- StringValueRangeProcessor (Xapian::valueno valno_)
  Constructor.
- Xapian::valueno operator() (std::string &, std::string &)
  Any strings are valid as begin and end.

7.27.1 Detailed Description

Handle a string range.

The end points can be any strings.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 Xapian::StringValueRangeProcessor::StringValueRangeProcessor (Xapian::valueno *valno_*) [inline]

Constructor.

Parameters:

valno_ The value number to return from operator().

7.27.3 Member Function Documentation

7.27.3.1 Xapian::valueno Xapian::StringValueRangeProcessor::operator() (std::string &, std::string &) [inline, virtual]

Any strings are valid as begin and end.

Implements Xapian::ValueRangeProcessor.

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

- xapian/queryparser.h

7.28 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 weight=1, const std::string &prefix="")

  Index some text.

- void index_text (const std::string &text, Xapian::termcount weight=1, const std::string &prefix="")

  Index some text in a std::string.

- void index_text_without_positions (const Xapian::Utf8Iterator &itor, Xapian::termcount weight=1, const std::string &prefix="")

  Index some text without positional information.

- void index_text_without_positions (const std::string &text, Xapian::termcount weight=1, const std::string &prefix="")

  Index some text in a std::string without positional information.

- void increase_termpos (Xapian::termcount delta=100)

  Increase the termpos used by index_text by delta.

- 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.

7.28.1 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.

7.28.2 Member Enumeration Documentation

7.28.2.1 enum Xapian::TermGenerator::flags

Flags to OR together and pass to TermGenerator::set_flags().

Enumerator:

FLAG_SPELLING Index data required for spelling correction.

7.28.3 Constructor & Destructor Documentation

7.28.3.1 Xapian::TermGenerator::TermGenerator (const TermGenerator & *o*)

Copy constructor.

7.28.3.2 Xapian::TermGenerator::TermGenerator ()

Default constructor.

7.28.3.3 Xapian::TermGenerator::~TermGenerator ()

Destructor.

7.28.4 Member Function Documentation

7.28.4.1 TermGenerator& Xapian::TermGenerator::operator= (const TermGenerator & *o*)

Assignment.

7.28.4.2 void Xapian::TermGenerator::set_stemmer (const Xapian::Stem & *stemmer*)

Set the Xapian::Stem object to be used for generating stemmed terms.

7.28.4.3 void Xapian::TermGenerator::set_stopper (const Xapian::Stopper * *stop* = NULL)

Set the Xapian::Stopper object to be used for identifying stopwords.

7.28.4.4 void Xapian::TermGenerator::set_document (const Xapian::Document & *doc*)

Set the current document.

7.28.4.5 const Xapian::Document& Xapian::TermGenerator::get_document () const

Get the current document.

7.28.4.6 void Xapian::TermGenerator::set_database (const Xapian::WritableDatabase & *db*)

Set the database to index spelling data to.

7.28.4.7 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.

7.28.4.8 void Xapian::TermGenerator::index_text (const Xapian::Utf8Iterator & *itor*, Xapian::termcount *weight* = 1, const std::string & *prefix* = "")

Index some text.

Parameters:

weight The wdf increment (default 1).

prefix The term prefix to use (default is no prefix).

7.28.4.9 void Xapian::TermGenerator::index_text (const std::string & *text*, Xapian::termcount *weight* = 1, const std::string & *prefix* = "") [inline]

Index some text in a std::string.

Parameters:

weight The wdf increment (default 1).

prefix The term prefix to use (default is no prefix).

7.28.4.10 void Xapian::TermGenerator::index_text_without_positions (const Xapian::Utf8Iterator & *itor*, Xapian::termcount *weight* = 1, const std::string & *prefix* = "")

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.

7.28.4.11 void Xapian::TermGenerator::index_text_without_positions (const std::string & *text*, Xapian::termcount *weight* = 1, const std::string & *prefix* = "") [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.

7.28.4.12 void Xapian::TermGenerator::increase_termpos (Xapian::termcount *delta* = 100)

Increase the termpos used by index_text by *delta*.

This can be used to prevent phrase searches from spanning two unconnected blocks of text (e.g. the title and body text).

7.28.4.13 Xapian::termcount Xapian::TermGenerator::get_termpos () const

Get the current term position.

7.28.4.14 void Xapian::TermGenerator::set_termpos (Xapian::termcount *termpos*)

Set the current term position.

7.28.4.15 std::string Xapian::TermGenerator::get_description () const

Return a string describing this object.

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

- xapian/termgenerator.h

7.29 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** (Internal *internal_)
- 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.
- TermIterator & **operator++** ()
- TermNameWrapper **operator++** (int)
- void skip_to (const std::string &tname)
  Skip the iterator to term tname, or the first term after tname if tname isn't in the list of terms being iterated.
- 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.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

7.29.1 Detailed Description

An iterator pointing to items in a list of terms.

7.29.2 Member Typedef Documentation

7.29.2.1 typedef std::input_iterator_tag Xapian::TermIterator::iterator_category

Allow use as an STL iterator.

7.29.2.2 typedef std::string Xapian::TermIterator::value_type

Allow use as an STL iterator.

7.29.2.3 typedef Xapian::termcount_diff Xapian::TermIterator::difference_type

Allow use as an STL iterator.

7.29.2.4 typedef std::string* Xapian::TermIterator::pointer

Allow use as an STL iterator.

7.29.2.5 typedef std::string& Xapian::TermIterator::reference

Allow use as an STL iterator.

7.29.3 Constructor & Destructor Documentation

7.29.3.1 Xapian::TermIterator::TermIterator ()

Default constructor - for declaring an uninitialised iterator.

7.29.3.2 Xapian::TermIterator::~TermIterator ()

Destructor.

7.29.3.3 Xapian::TermIterator::TermIterator (const TermIterator & *other*)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

7.29.4 Member Function Documentation

7.29.4.1 void Xapian::TermIterator::operator= (const TermIterator & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

7.29.4.2 std::string Xapian::TermIterator::operator * () const

Return the current term.

7.29.4.3 void Xapian::TermIterator::skip_to (const std::string & *tname*)

Skip the iterator to term tname, or the first term after tname if tname isn't in the list of terms being iterated.

7.29.4.4 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 occurences of a term in a particular document.

7.29.4.5 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.

7.29.4.6 Xapian::termcount Xapian::TermIterator::positionlist_count () const

Return length of positionlist for current term.

7.29.4.7 PositionIterator Xapian::TermIterator::positionlist_begin () const

Return PositionIterator pointing to start of positionlist for current term.

7.29.4.8 PositionIterator Xapian::TermIterator::positionlist_end () const [inline]

Return PositionIterator pointing to end of positionlist for current term.

7.29.4.9 std::string Xapian::TermIterator::get_description () const

Return a string describing this object.

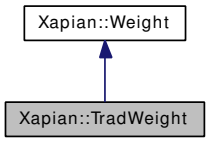
The documentation for this class was generated from the following file:

- xapian/termiterator.h

7.30 Xapian::TradWeight Class Reference

Traditional probabilistic weighting scheme.

Inheritance diagram for Xapian::TradWeight:



Public Member Functions

- TradWeight (double k)

  Construct a TradWeight.

- TradWeight * clone () const

  Return a new weight object of this type.

- std::string name () const

  Name of the weighting scheme.

- std::string serialise () const

  Serialise object parameters into a string.

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

  Create object given string serialisation returned by serialise().

- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::doclength len) const

  Get a weight which is part of the sum over terms being performed.

- Xapian::weight get_maxpart () const

  Gets the maximum value that get_sumpart() may return.

- Xapian::weight get_sumextra (Xapian::doclength len) const

  Get an extra weight for a document to add to the sum calculated over the query terms.

- Xapian::weight get_maxextra () const

  Gets the maximum value that get_sumextra() may return.

- bool get_sumpart_needs_doclength () const

  return false if the weight object doesn't need doclength

7.30.1 Detailed Description

Traditional probabilistic weighting scheme.

This class implements the Traditional Probabilistic Weighting scheme, as described by the early papers on Probabilistic Retrieval. BM25 generally gives better results.

The Traditional weighting scheme formula is

$$\sum_t \frac{f_{t,d}}{k.L_d + f_{t,d}}.w_t$$

where

- w_t is the termweight of term t
- $f_{t,d}$ is the within document frequency of term t in document d
- L_d is the normalised length of document d
- k is a user specifiable parameter

TradWeight(k) is equivalent to BM25Weight(k, 0, 0, 1, 0), except that the latter returns weights (k+1) times larger.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 Xapian::TradWeight::TradWeight (double *k*) [inline, explicit]

Construct a TradWeight.

Parameters:

k parameter governing the importance of within document frequency and document length - any non-negative number (0 meaning to ignore wdf and doc length when calculating weights). Default is 1.

7.30.3 Member Function Documentation

7.30.3.1 TradWeight* Xapian::TradWeight::clone () const [virtual]

Return a new weight object of this type.

A subclass called FooWeight taking parameters param1 and param2 should implement this as:

virtual FooWeight * clone() const { return new FooWeight(param1, param2); }

Implements Xapian::Weight.

7.30.3.2 std::string Xapian::TradWeight::name () const [virtual]

Name of the weighting scheme.

If the subclass is called FooWeight, this should return "Foo".

Implements Xapian::Weight.

7.30.3.3 std::string Xapian::TradWeight::serialise () const [virtual]

Serialise object parameters into a string.

Implements Xapian::Weight.

7.30.3.4 TradWeight* Xapian::TradWeight::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Implements Xapian::Weight.

7.30.3.5 Xapian::weight Xapian::TradWeight::get_sumpart (Xapian::termcount *wdf*, Xapian::doclength *len*) const [virtual]

Get a weight which is part of the sum over terms being performed.

This returns a weight for a given term and document. These weights are summed to give a total weight for the document.

Parameters:

wdf the within document frequency of the term.

len the (unnormalised) document length.

Implements Xapian::Weight.

7.30.3.6 Xapian::weight Xapian::TradWeight::get_maxpart () const [virtual]

Gets the maximum value that get_sumpart() may return.

This is used in optimising searches, by having the postlist tree decay appropriately when parts of it can have limited, or no, further effect.

Implements Xapian::Weight.

7.30.3.7 Xapian::weight Xapian::TradWeight::get_sumextra (Xapian::doclength *len*) const [virtual]

Get an extra weight for a document to add to the sum calculated over the query terms.

This returns a weight for a given document, and is used by some weighting schemes to account for influence such as document length.

Parameters:

len the (unnormalised) document length.

Implements Xapian::Weight.

7.30.3.8 Xapian::weight Xapian::TradWeight::get_maxextra () const [virtual]

Gets the maximum value that get_sumextra() may return.

This is used in optimising searches.

Implements Xapian::Weight.

7.30.3.9 bool Xapian::TradWeight::get_sumpart_needs_doclength () const [virtual]

return false if the weight object doesn't need doclength

Reimplemented from Xapian::Weight.

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

- xapian/enquire.h

7.31 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.

7.31.1 Detailed Description

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

7.31.2 Member Typedef Documentation

7.31.2.1 typedef std::input_iterator_tag Xapian::Utf8Iterator::iterator_category

We implement the semantics of an STL input_iterator.

7.31.2.2 typedef unsigned Xapian::Utf8Iterator::value_type

We implement the semantics of an STL input_iterator.

7.31.2.3 typedef size_t Xapian::Utf8Iterator::difference_type

We implement the semantics of an STL input_iterator.

7.31.2.4 typedef const unsigned∗ Xapian::Utf8Iterator::pointer

We implement the semantics of an STL input_iterator.

7.31.2.5 typedef const unsigned& Xapian::Utf8Iterator::reference

We implement the semantics of an STL input_iterator.

7.31.3 Constructor & Destructor Documentation

7.31.3.1 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.

7.31.3.2 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.

7.31.3.3 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.

7.31.3.4 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.

7.31.4 Member Function Documentation

7.31.4.1 const char* Xapian::Utf8Iterator::raw () const [inline]

Return the raw const char * pointer for the current position.

7.31.4.2 size_t Xapian::Utf8Iterator::left () const [inline]

Return the number of bytes left in the iterator's buffer.

7.31.4.3 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.

7.31.4.4 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.

7.31.4.5 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.

7.31.4.6 Utf8Iterator Xapian::Utf8Iterator::operator++ (int) [inline]

Move forward to the next Unicode character.

Returns:

An iterator pointing to the position before the move.

7.31.4.7 Utf8Iterator& Xapian::Utf8Iterator::operator++ () [inline]

Move forward to the next Unicode character.

Returns:

A reference to this object.

7.31.4.8 bool Xapian::Utf8Iterator::operator== (const Utf8Iterator & *other*) const [inline]

Test two Utf8Iterators for equality.

Returns:

true iff the iterators point to the same position.

7.31.4.9 bool Xapian::Utf8Iterator::operator!= (const Utf8Iterator & *other*) const [inline]

Test two Utf8Iterators for inequality.

Returns:

true iff the iterators do not point to the same position.

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

- xapian/unicode.h

7.32 Xapian::ValueIterator Class Reference

An iterator pointing to values associated with a document.

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::valueno_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

- ValueIterator ()
  Create an uninitialised iterator; this cannot be used, but is convenient syntactically.
- ValueIterator (const ValueIterator &other)
  Copying is allowed (and is cheap).
- void operator= (const ValueIterator &other)
  Assignment is allowed (and is cheap).
- ValueIterator & operator++ ()
  Advance the iterator.
- ValueIterator operator++ (int)
  Advance the iterator (postfix variant).
- const std::string & operator * () const
  Get the value for the current position.
- const std::string * operator → () const
  Get the value for the current position.
- Xapian::valueno get_valueno () const

Get the number of the value at the current position.

- std::string get_description () const

  Return a string describing this object.

Friends

- 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.

7.32.1 Detailed Description

An iterator pointing to values associated with a document.

7.32.2 Member Typedef Documentation

7.32.2.1 typedef std::input_iterator_tag Xapian::ValueIterator::iterator_category

Allow use as an STL iterator.

7.32.2.2 typedef std::string Xapian::ValueIterator::value_type

Allow use as an STL iterator.

7.32.2.3 typedef Xapian::valueno_diff Xapian::ValueIterator::difference_type

Allow use as an STL iterator.

7.32.2.4 typedef std::string* Xapian::ValueIterator::pointer

Allow use as an STL iterator.

7.32.2.5 typedef std::string& Xapian::ValueIterator::reference

Allow use as an STL iterator.

7.32.3 Constructor & Destructor Documentation

7.32.3.1 Xapian::ValueIterator::ValueIterator () [inline]

Create an uninitialised iterator; this cannot be used, but is convenient syntactically.

7.32.3.2 Xapian::ValueIterator::ValueIterator (const ValueIterator & *other*) [inline]

Copying is allowed (and is cheap).

7.32.4 Member Function Documentation

7.32.4.1 void Xapian::ValueIterator::operator= (const ValueIterator & *other*) [inline]

Assignment is allowed (and is cheap).

7.32.4.2 ValueIterator& Xapian::ValueIterator::operator++ () [inline]

Advance the iterator.

7.32.4.3 ValueIterator Xapian::ValueIterator::operator++ (int) [inline]

Advance the iterator (postfix variant).

7.32.4.4 const std::string& Xapian::ValueIterator::operator * () const

Get the value for the current position.

7.32.4.5 const std::string* Xapian::ValueIterator::operator → () const

Get the value for the current position.

7.32.4.6 Xapian::valueno Xapian::ValueIterator::get_valueno () const

Get the number of the value at the current position.

7.32.4.7 std::string Xapian::ValueIterator::get_description () const

Return a string describing this object.

7.32.5 Friends And Related Function Documentation

7.32.5.1 bool operator== (const ValueIterator & *a*, const ValueIterator & *b*) [friend]

Equality test for ValueIterator objects.

7.32.5.2 bool operator!= (const ValueIterator & *a*, const ValueIterator & *b*) [friend]

Inequality test for ValueIterator objects.

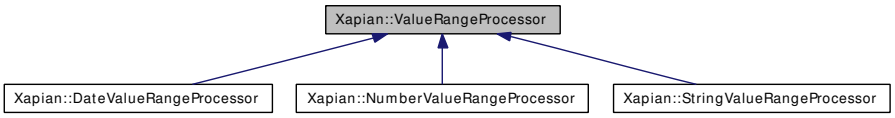
The documentation for this class was generated from the following file:

- xapian/valueiterator.h

7.33 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.

7.33.1 Detailed Description

Base class for value range processors.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 virtual Xapian::ValueRangeProcessor::~ValueRangeProcessor () [virtual]

Destructor.

7.33.3 Member Function Documentation

7.33.3.1 virtual Xapian::valueno Xapian::ValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) [pure virtual]

Check for a valid range of this type.

If this ValueRangeProcessor recognises BEGIN..END it returns the value number of range filter on. Otherwise it returns Xapian::BAD_VALUENO.

Implemented in Xapian::StringValueRangeProcessor, Xapian::DateValueRangeProcessor, and Xapian::NumberValueRangeProcessor.

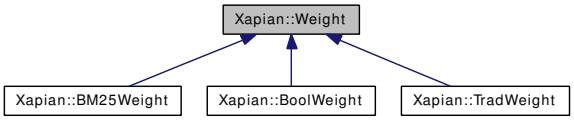
The documentation for this struct was generated from the following file:

- xapian/queryparser.h

7.34 Xapian::Weight Class Reference

Abstract base class for weighting schemes.

Inheritance diagram for Xapian::Weight:



Public Member Functions

- Weight * create (const Internal *internal_, Xapian::doclength querysize_-, Xapian::termcount wqf_, const std::string &tname_) const

  Create a new weight object of the same type as this and initialise it with the specified statistics.

- virtual std::string name () const=0

  Name of the weighting scheme.

- virtual std::string serialise () const=0

  Serialise object parameters into a string.

- virtual Weight * unserialise (const std::string &s) const=0

  Create object given string serialisation returned by serialise().

- virtual Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::doclength len) const=0

  Get a weight which is part of the sum over terms being performed.

- virtual Xapian::weight get_maxpart () const=0

  Gets the maximum value that get_sumpart() may return.

- virtual Xapian::weight get_sumextra (Xapian::doclength len) const=0

  Get an extra weight for a document to add to the sum calculated over the query terms.

- virtual Xapian::weight get_maxextra () const=0

  Gets the maximum value that get_sumextra() may return.

- virtual bool get_sumpart_needs_doclength () const

  return false if the weight object doesn't need doclength

Protected Member Functions

- **Weight** (const Weight &)

Protected Attributes

- const Internal * **internal**
- Xapian::doclength **querysize**
- Xapian::termcount **wqf**
- std::string **tname**

7.34.1 Detailed Description

Abstract base class for weighting schemes.

7.34.2 Member Function Documentation

7.34.2.1 Weight* Xapian::Weight::create (const Internal * *internal_*, Xapian::doclength *querysize_*, Xapian::termcount *wqf_*, const std::string & *tname_*) const

Create a new weight object of the same type as this and initialise it with the specified statistics.

You shouldn't call this method yourself - it's called by Enquire.

Parameters:

internal_ Object to ask for collection statistics.

querysize_ Query size.

wqf_ Within query frequency of term this object is associated with.

tname_ Term which this object is associated with.

7.34.2.2 virtual std::string Xapian::Weight::name () const [pure virtual]

Name of the weighting scheme.

If the subclass is called FooWeight, this should return "Foo".

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.34.2.3 virtual std::string Xapian::Weight::serialise () const [pure virtual]

Serialise object parameters into a string.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.34.2.4 virtual Weight* Xapian::Weight::unserialise (const std::string & *s*) const [pure virtual]

Create object given string serialisation returned by serialise().

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.34.2.5 virtual Xapian::weight Xapian::Weight::get_sumpart (Xapian::termcount *wdf*, Xapian::doclength *len*) const [pure virtual]

Get a weight which is part of the sum over terms being performed.

This returns a weight for a given term and document. These weights are summed to give a total weight for the document.

Parameters:

wdf the within document frequency of the term.

len the (unnormalised) document length.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.34.2.6 virtual Xapian::weight Xapian::Weight::get_maxpart () const [pure virtual]

Gets the maximum value that get_sumpart() may return.

This is used in optimising searches, by having the postlist tree decay appropriately when parts of it can have limited, or no, further effect.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.34.2.7 virtual Xapian::weight Xapian::Weight::get_sumextra (Xapian::doclength *len*) const [pure virtual]

Get an extra weight for a document to add to the sum calculated over the query terms.

This returns a weight for a given document, and is used by some weighting schemes to account for influence such as document length.

Parameters:

len the (unnormalised) document length.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.34.2.8 virtual Xapian::weight Xapian::Weight::get_maxextra () const [pure virtual]

Gets the maximum value that get_sumextra() may return.

This is used in optimising searches.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

7.34.2.9 virtual bool Xapian::Weight::get_sumpart_needs_doclength () const [virtual]

return false if the weight object doesn't need doclength

Reimplemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

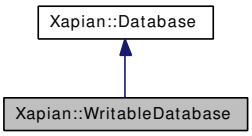
The documentation for this class was generated from the following file:

- xapian/enquire.h

7.35 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 flush ()

  Flush to disk any modifications made to the database.

- 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 potential 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.

7.35.1 Detailed Description

This class provides read/write access to a database.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 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.

7.35.2.2 Xapian::WritableDatabase::WritableDatabase ()

Create an empty WritableDatabase.

7.35.2.3 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.

7.35.2.4 Xapian::WritableDatabase::WritableDatabase (const WritableDatabase & *other*)

Copying is allowed.

The internals are reference counted, so copying is cheap.

7.35.3 Member Function Documentation

7.35.3.1 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.

7.35.3.2 void Xapian::WritableDatabase::flush ()

Flush to disk any 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.

Flush may be called at any time to ensure that the modifications which have been made are written to disk: if the flush succeeds, all the preceding modifications will have been written to disk.

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 flush within a transaction.

Beware of calling flush too frequently: this will have a severe performance cost.

Note that flush 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.

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.

7.35.3.3 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 flush before and after so that the modifications stand and fall without affecting modifications before or after.

The downside of this flushing is that small transactions cause modifications to be frequently flushed which can harm indexing performance in the same way that explicitly calling flush 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 flush 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.

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.

7.35.3.4 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.

7.35.3.5 void Xapian::WritableDatabase::cancel_transaction ()

Abort the transaction currently in progress, discarding the potential 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.

7.35.3.6 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 flush() 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 flush 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.

7.35.3.7 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 flush() 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 flush 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.

7.35.3.8 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.

7.35.3.9 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 flush() 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 flush 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.

7.35.3.10 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.

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).

Note that changes to the database won't be immediately committed to disk; see flush() 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 flush 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.

7.35.3.11 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).

7.35.3.12 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).

7.35.3.13 void Xapian::WritableDatabase::add_synonym (const std::string & *term*, const std::string & *synonym*) const

Add a synonym for a term.

If *synonym* is already a synonym for *term*, then no action is taken.

7.35.3.14 void Xapian::WritableDatabase::remove_synonym (const std::string & *term*, const std::string & *synonym*) const

Remove a synonym for a term.

If *synonym* isn't a synonym for *term*, then no action is taken.

7.35.3.15 void Xapian::WritableDatabase::clear_synonyms (const std::string & *term*) const

Remove all synonyms for a term.

If *term* has no synonyms, no action is taken.

7.35.3.16 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 flush() 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.

7.35.3.17 std::string Xapian::WritableDatabase::get_description () const [virtual]

Return a string describing this object.

Reimplemented from Xapian::Database.

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

- xapian/database.h

Chapter 8

xapian-core File Documentation

8.1 xapian/version.h File Reference

Define preprocesor symbols for the library version.

Defines

- #define XAPIAN_VERSION "1.0.16"

  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 0

  The minor component of the Xapian version.

- #define XAPIAN_REVISION 16

  The revision component of the Xapian version.

- #define XAPIAN_HAS_FLINT_BACKEND 1

  XAPIAN_HAS_FLINT_BACKEND Defined if the flint backend is enabled.

- #define XAPIAN_HAS_QUARTZ_BACKEND 1

  XAPIAN_HAS_QUARTZ_BACKEND Defined if the quartz 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.

8.1.1 Detailed Description

Define preprocesor symbols for the library version.

8.1.2 Define Documentation

8.1.2.1 #define XAPIAN_HAS_FLINT_BACKEND 1

XAPIAN_HAS_FLINT_BACKEND Defined if the flint backend is enabled.

8.1.2.2 #define XAPIAN_HAS_INMEMORY_BACKEND 1

XAPIAN_HAS_INMEMORY_BACKEND Defined if the inmemory backend is enabled.

8.1.2.3 #define XAPIAN_HAS_QUARTZ_BACKEND 1

XAPIAN_HAS_QUARTZ_BACKEND Defined if the quartz backend is enabled.

8.1.2.4 #define XAPIAN_HAS_REMOTE_BACKEND 1

XAPIAN_HAS_REMOTE_BACKEND Defined if the remote backend is enabled.

8.1.2.5 #define XAPIAN_MAJOR_VERSION 1

The major component of the Xapian version.

E.g. for Xapian 1.0.14 this would be: 1

8.1.2.6 #define XAPIAN_MINOR_VERSION 0

The minor component of the Xapian version.

E.g. for Xapian 1.0.14 this would be: 0

8.1.2.7 #define XAPIAN_REVISION 16

The revision component of the Xapian version.

E.g. for Xapian 1.0.14 this would be: 14

8.1.2.8 #define XAPIAN_VERSION "1.0.16"

The version of Xapian as a C string literal.

8.2 xapian.h File Reference

Public interfaces for the Xapian library.

Namespaces

- namespace Xapian

Functions

- const char * Xapian::version_string ()
  Report the version string of the library which the program is linked with.
- const char * Xapian::xapian_version_string ()
  For compatibility with Xapian 0.9.5 and earlier.
- int Xapian::major_version ()
  Report the major version of the library which the program is linked with.
- int Xapian::xapian_major_version ()
  For compatibility with Xapian 0.9.5 and earlier.
- int Xapian::minor_version ()
  Report the minor version of the library which the program is linked with.
- int Xapian::xapian_minor_version ()
  For compatibility with Xapian 0.9.5 and earlier.
- int Xapian::revision ()
  Report the revision of the library which the program is linked with.
- int Xapian::xapian_revision ()
  For compatibility with Xapian 0.9.5 and earlier.

8.2.1 Detailed Description

Public interfaces for the Xapian library.

8.3 xapian/database.h File Reference

API for working with Xapian databases.

Namespaces

- namespace Xapian

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.

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.

8.3.1 Detailed Description

API for working with Xapian databases.

8.4 xapian/dbfactory.h File Reference

Factory functions for constructing Database and WritableDatabase objects.

Namespaces

- namespace Xapian
- namespace Xapian::Auto
- namespace Xapian::InMemory
- namespace Xapian::Quartz
- namespace Xapian::Flint
- namespace Xapian::Remote

Functions

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

  Construct a Database object for a stub database file.

- WritableDatabase Xapian::InMemory::open ()

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

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

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

- Xapian::Quartz::WritableDatabaseopen (const std::string &dir, int action, int block_size=8192)

  Construct a Database object for update access to a Quartz 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.

8.4.1 Detailed Description

Factory functions for constructing Database and WritableDatabase objects.

8.5 xapian/document.h File Reference

API for working with documents.

Namespaces

- namespace Xapian

Classes

- class Xapian::Document

  A document in the database - holds data, values, terms, and postings.

8.5.1 Detailed Description

API for working with documents.

8.6 xapian/enquire.h File Reference

API for running queries.

Namespaces

- namespace Xapian

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.

- class Xapian::Weight

  Abstract base class for weighting schemes.

- class Xapian::BoolWeight

  Boolean weighting scheme (everything gets 0).

- class Xapian::BM25Weight

  BM25 weighting scheme.

- class Xapian::TradWeight

  Traditional probabilistic weighting scheme.

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.

8.6.1 Detailed Description

API for running queries.

8.7 xapian/errorhandler.h File Reference

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

Namespaces

- namespace Xapian

Classes

- class Xapian::ErrorHandler
  Decide if a Xapian::Error exception should be ignored.

8.7.1 Detailed Description

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

8.8 xapian/expanddecider.h File Reference

Allow rejection of terms during ESet generation.

Namespaces

- namespace Xapian

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.

8.8.1 Detailed Description

Allow rejection of terms during ESet generation.

8.9 xapian/positioniterator.h File Reference

Classes for iterating through position lists.

Namespaces

- namespace Xapian

Classes

- class Xapian::PositionIterator

  An iterator pointing to items in a list of positions.

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.

8.9.1 Detailed Description

Classes for iterating through position lists.

8.10 xapian/postingiterator.h File Reference

Classes for iterating through posting lists.

Namespaces

- namespace Xapian

Classes

- class Xapian::PostingIterator
  An iterator pointing to items in a list of postings.

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.

8.10.1 Detailed Description

Classes for iterating through posting lists.

8.11 xapian/query.h File Reference

Classes for representing a query.

Namespaces

- namespace Xapian

Classes

- class Xapian::Query
  Class representing a query.

8.11.1 Detailed Description

Classes for representing a query.

8.12 xapian/queryparser.h File Reference

parsing a user query string to build a Xapian::Query object

Namespaces

- namespace Xapian

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.

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.*

8.12.1 Detailed Description

parsing a user query string to build a Xapian::Query object

8.13 xapian/sorter.h File Reference

Build sort keys for MSet ordering.

Namespaces

- namespace Xapian

Classes

- class Xapian::Sorter

  Virtual base class for sorter functor.

- class Xapian::MultiValueSorter

  Sorter subclass which sorts by a several values.

8.13.1 Detailed Description

Build sort keys for MSet ordering.

8.14 xapian/stem.h File Reference

stemming algorithms

Namespaces

- namespace Xapian

Classes

- class Xapian::Stem

  Class representing a stemming algorithm.

8.14.1 Detailed Description

stemming algorithms

8.15 xapian/termgenerator.h File Reference

parse free text and generate terms

Namespaces

- namespace Xapian

Classes

- class Xapian::TermGenerator

  Parses a piece of text and generate terms.

8.15.1 Detailed Description

parse free text and generate terms

8.16 xapian/termiterator.h File Reference

Classes for iterating through term lists.

Namespaces

- namespace Xapian

Classes

- class Xapian::TermIterator

  An iterator pointing to items in a list of terms.

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.

8.16.1 Detailed Description

Classes for iterating through term lists.

8.17 xapian/types.h File Reference

typedefs for Xapian

Namespaces

- namespace Xapian

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 microseconds.

- 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".

8.17.1 Detailed Description

typedefs for Xapian

8.18 xapian/unicode.h File Reference

Unicode and UTF-8 related classes and functions.

Namespaces

- namespace Xapian
- namespace Xapian::Unicode
- namespace **Xapian::Unicode::Internal**

Classes

- class Xapian::Utf8Iterator

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

Enumerations

- enum Xapian::Unicode::category {

  UNASSIGNED, **UPPERCASE_LETTER**, **LOWERCASE_LETTER**, **TITLECASE_LETTER**,

  MODIFIER_LETTER, **OTHER_LETTER**, **NON_SPACING_MARK**, **ENCLOSING_MARK**,

  COMBINING_SPACING_MARK, **DECIMAL_DIGIT_NUMBER**, **LETTER_NUMBER**, **OTHER_NUMBER**,

  SPACE_SEPARATOR, **LINE_SEPARATOR**, **PARAGRAPH_SEPARATOR**, **CONTROL**,

  FORMAT, **PRIVATE_USE**, **SURROGATE**, **CONNECTOR_PUNCTUATION**,

  DASH_PUNCTUATION, **OPEN_PUNCTUATION**, **CLOSE_PUNCTUATION**, **INITIAL_QUOTE_PUNCTUATION**,

  FINAL_QUOTE_PUNCTUATION, **OTHER_PUNCTUATION**, **MATH_SYMBOL**, **CURRENCY_SYMBOL**,

  MODIFIER_SYMBOL, **OTHER_SYMBOL** }

  Each Unicode character is in exactly one of these categories.

Functions

- int **Xapian::Unicode::Internal::get_character_info** (unsigned ch)
- int **Xapian::Unicode::Internal::get_case_type** (int info)
- category **Xapian::Unicode::Internal::get_category** (int info)
- int **Xapian::Unicode::Internal::get_delta** (int info)

- 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.

8.18.1 Detailed Description

Unicode and UTF-8 related classes and functions.

8.19 xapian/valueiterator.h File Reference

classes for iterating through values

Namespaces

- namespace Xapian

Classes

- class Xapian::ValueIterator
  An iterator pointing to values associated with a document.

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.

8.19.1 Detailed Description

classes for iterating through values

Chapter 9

xapian-core Page Documentation

9.1 Deprecated List

Member Xapian::Enquire::register_match_decider(const std::string &name, const MatchDecider *mdecider=NULL This method is deprecated. It was added long ago with the intention that it would allow the remote backend to support use of MatchDecider objects, but there's a better approach.

Member Xapian::Query::Query(Query::op op_, Xapian::Query q) This method is deprecated because it isn't useful, since none of the current query operators can be usefully applied to a single subquery without a parameter value.

Member Xapian::xapian_major_version() This function is now deprecated, use Xapian::major_version() instead.

Member Xapian::xapian_minor_version() This function is now deprecated, use Xapian::minor_version() instead.

Member Xapian::xapian_revision() This function is now deprecated, use Xapian::revision() instead.

Member Xapian::xapian_version_string() This function is now deprecated, use Xapian::version_string() instead.

Index