xapian-core Reference Manual

0.9.10

Generated by Doxygen 1.4.6

Tue Mar 6 00:13:42 2007

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 library lives in the Xapian namespace.

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 ExpandDecider

  Base class for expand 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 Error

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

- class ErrorHandler

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

- class ExpandDeciderFilterTerms

  One useful expand decision functor, which provides a way of filtering out a fixed list of terms from the expand set.

- class ExpandDeciderAnd

  An expand decision functor which can be used to join two functors with an AND operation.

- class TermPosWrapper

  A wrapper class for a termpos which returns the termpos if dereferenced with $*$.

- class PositionIterator

  An iterator pointing to items in a list of positions.

- class DocIDWrapper

  A wrapper class for a docid which returns the docid if dereferenced with $*$.

- 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.

- class QueryParser

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

- class Stem

  Class representing a stemming algorithm.

- class TermNameWrapper

  A wrapper class for a termname which returns the termname if dereferenced with ∗.

- class TermIterator

  An iterator pointing to items in a list of terms.

- class ValueIterator

  An iterator pointing to values associated with a document.

Functions

- bool **operator==** (const MSetIterator &a, const MSetIterator &b)
- bool **operator!=** (const MSetIterator &a, const MSetIterator &b)
- bool **operator==** (const ESetIterator &a, const ESetIterator &b)
- bool **operator!=** (const ESetIterator &a, const ESetIterator &b)
- 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 **serialise_qint_** (const Xapian::Query::Internal *qint, Xapian::termpos &curpos)
- bool **operator==** (const TermIterator &a, const TermIterator &b)
- bool **operator!=** (const TermIterator &a, const TermIterator &b)
- bool **operator==** (const ValueIterator &a, const ValueIterator &b)
- bool **operator!=** (const ValueIterator &a, const ValueIterator &b)

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.

6.1.1 Detailed Description

The Xapian library lives in the Xapian namespace.

6.1.2 Function Documentation

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

Test inequality of two PostingIterators.

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

Test inequality of two PositionIterators.

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

Test equality of two PostingIterators.

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

Test equality of two PositionIterators.

6.1.3 Variable Documentation

6.1.3.1 const int Xapian::DB_CREATE = 2

Create a new database; fail if db exists.

6.1.3.2 const int Xapian::DB_CREATE_OR_OPEN = 1

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

6.1.3.3 const int Xapian::DB_CREATE_OR_OVERWRITE = 3

Overwrite existing db; create if none exists.

6.1.3.4 const int Xapian::DB_OPEN = 4

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

Chapter 7

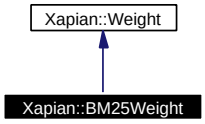
xapian-core Class Documentation

7.1 Xapian::BM25Weight Class Reference

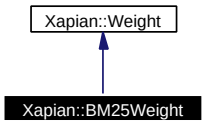
BM25 weighting scheme.

`#include <enquire.h>`

Inheritance diagram for Xapian::BM25Weight:



Collaboration 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

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

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 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.

7.1.3.3 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.

7.1.3.4 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.

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

return false if the weight object doesn't need doclength

Reimplemented from Xapian::Weight.

7.1.3.7 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.

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

Serialise object parameters into a string.

Implements Xapian::Weight.

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

Create object given string serialisation returned by serialise().

Implements Xapian::Weight.

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

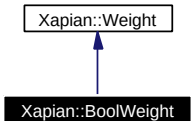
- include/xapian/enquire.h

7.2 Xapian::BoolWeight Class Reference

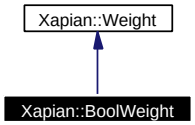
Boolean weighting scheme (everything gets 0).

`#include <enquire.h>`

Inheritance diagram for Xapian::BoolWeight:



Collaboration 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 &) const
  Create object given string serialisation returned by serialise().
- Xapian::weight get_sumpart (Xapian::termcount, Xapian::doclength) 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) 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 [inline, 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 Xapian::weight Xapian::BoolWeight::get_maxextra () const [inline, virtual]

Gets the maximum value that get_sumextra() may return.

This is used in optimising searches.

Implements Xapian::Weight.

7.2.2.3 Xapian::weight Xapian::BoolWeight::get_maxpart () const [inline, 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.

7.2.2.4 Xapian::weight Xapian::BoolWeight::get_sumextra (Xapian::doclength) const [inline, 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.

7.2.2.5 Xapian::weight Xapian::BoolWeight::get_sumpart (Xapian::termcount, Xapian::doclength) const [inline, 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 bool Xapian::BoolWeight::get_sumpart_needs_doclength () const [inline, virtual]

return false if the weight object doesn't need doclength

Reimplemented from Xapian::Weight.

7.2.2.7 std::string Xapian::BoolWeight::name () const [inline, virtual]

Name of the weighting scheme.

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

Implements Xapian::Weight.

7.2.2.8 std::string Xapian::BoolWeight::serialise () const [inline, virtual]

Serialise object parameters into a string.

Implements Xapian::Weight.

7.2.2.9 BoolWeight* Xapian::BoolWeight::unserialise (const std::string &) const [inline, virtual]

Create object given string serialisation returned by serialise().

Implements Xapian::Weight.

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

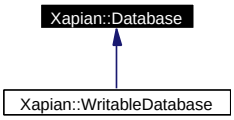
- include/xapian/enquire.h

7.3 Xapian::Database Class Reference

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

`#include <database.h>`

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.

- **Database** (Internal *internal)
- 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

  Introspection method.

- 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().

- 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.

Public Attributes

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

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).

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*)

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 TermIterator Xapian::Database::allterms_begin () const

An iterator which runs across all terms in the database.

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

Corresponding end iterator to allterms_begin().

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

Get the average length of the documents in the database.

7.3.3.5 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 ocurrences of the term in each document it indexes: ie, the sum of the within document frequencies of the term.

Parameters:

tname The term whose collection frequency is being requested.

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

Introspection method.

Returns:

A string describing this object.

Reimplemented in Xapian::WritableDatabase.

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

Get the number of documents in the database.

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

Get the length of a document.

7.3.3.9 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.

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

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

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

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

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

Does this database have any positional information?

7.3.3.13 void Xapian::Database::keep_alive ()

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

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

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

7.3.3.15 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.

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

Corresponding end iterator to positionlist_begin().

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

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

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

Corresponding end iterator to postlist_begin().

7.3.3.19 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.

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

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

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

Corresponding end iterator to termlist_begin().

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

- include/xapian/database.h

7.4 Xapian::DocIDWrapper Class Reference

A wrapper class for a docid which returns the docid if dereferenced with ∗.

`#include <postingiterator.h>`

Public Member Functions

- **DocIDWrapper** (docid did_)
- docid **operator** ∗ () const

7.4.1 Detailed Description

A wrapper class for a docid which returns the docid if dereferenced with ∗.

We need this to implement input_iterator semantics.

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

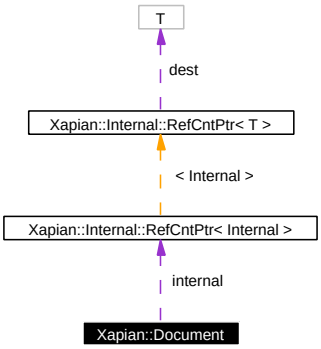
- include/xapian/postingiterator.h

7.5 Xapian::Document Class Reference

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

`#include <document.h>`

Collaboration diagram for Xapian::Document:



Public Member Functions

- **Document** (Internal *internal_)
- 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 value) 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.

- XAPIAN_DEPRECATED (void add_term_nopos(const std::string &term, Xapian::termcount wdfinc=1))

  Old name for add_term().

- 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

  Count the terms in this document.

- 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().

- std::string get_description () const

  Introspection method.

Public Attributes

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

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::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.

7.5.3.2 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.

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

Add a new value.

It will replace any existing value with the same number.

7.5.3.4 void Xapian::Document::clear_terms ()

Remove all terms (and postings) from the document.

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

Introspection method.

Returns:

A string representing this Document.

7.5.3.8 std::string Xapian::Document::get_value (Xapian::valueno *value*) const

Get value by number.

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

Parameters:

value The number of the value.

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

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

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::remove_value (Xapian::valueno *valueno*)

Remove any value with the given number.

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

Set data stored in the document.

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

Iterator for the terms in this document.

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

Count the terms in this document.

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

Equivalent end iterator for termlist_begin().

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

Iterator for the values in this document.

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

Count the values in this document.

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

Equivalent end iterator for values_begin().

7.5.3.20 Xapian::Document::XAPIAN_DEPRECATED (void *add_term_nopos*(const std::string &term, Xapian::termcount wdfinc=1))

Old name for add_term().

Deprecated

This method is deprecated and present only for backward compatibility. Use add_term() instead.

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

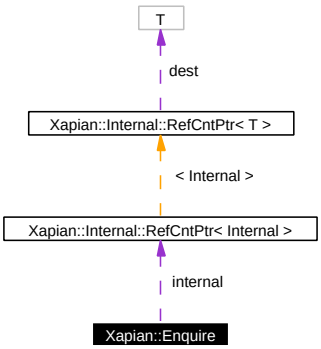
- include/xapian/document.h

7.6 Xapian::Enquire Class Reference

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

`#include <enquire.h>`

Collaboration diagram for Xapian::Enquire:



Public Types

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

Public Member Functions

- Enquire (const Database &databases, 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 ()

  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.

- XAPIAN_DEPRECATED (void set_sort_forward(bool sort_forward))

  For compatibility with Xapian 0.8.5 and earlier.

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

  Set the percentage and/or weight cutoffs.

- XAPIAN_DEPRECATED (void set_sorting(Xapian::valueno sort_key, int sort_bands, bool sort_by_relevance=false))

  For compatibility with Xapian 0.8.5 and earlier.

- void set_sort_by_relevance ()

  Set the sorting to be by relevance only.

- void set_sort_by_value (Xapian::valueno sort_key, bool ascending=true)

  Set the sorting to be by value only.

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

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

- void set_sort_by_relevance_then_value (Xapian::valueno sort_key, bool ascending=true)

  Set the sorting to be by relevance then value.

- void set_bias (Xapian::weight bias_weight, time_t bias_halflife)

  Set the bias functor parameters.

- 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, 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
  Introspection method.

Public Attributes

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

Static Public Attributes

- static const int **include_query_terms** = 1
- static const int **use_exact_termfreq** = 2

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 Database & *databases*, ErrorHandler * *errorhandler_* = 0)

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.

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.

7.6.2.2 Xapian::Enquire::~Enquire ()

Close the Xapian::Enquire object.

7.6.3 Member Function Documentation

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

Introspection method.

Returns:

A string representing the enquire object.

7.6.3.2 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.

7.6.3.3 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.

7.6.3.4 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.

7.6.3.5 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.

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

End iterator corresponding to get_matching_terms_begin().

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

End iterator corresponding to get_matching_terms_begin().

7.6.3.8 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.

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 that 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

Returns:

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

Exceptions:

Xapian::InvalidArgumentError See class documentation.

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

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.

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

Register a MatchDecider.

Parameters:

name The name to register this matchdecider as.

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

7.6.3.11 void Xapian::Enquire::set_bias (Xapian::weight *bias_weight*, time_t *bias_halflife*)

Set the bias functor parameters.

NB this is a temporary API for this feature.

Parameters:

bias_weight Maximum weight bias functor can add (and which is given to document with a time now or in the future).

bias_halflife the match bias decays exponentially as you go back in time. This sets the half-life of this decay in seconds (default 0 => no bias).

7.6.3.12 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::valueno(-1) which means no collapsing).

7.6.3.13 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).

7.6.3.14 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".

7.6.3.15 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.

7.6.3.16 void Xapian::Enquire::set_sort_by_relevance ()

Set the sorting to be by relevance only.

This is the default.

7.6.3.17 void Xapian::Enquire::set_sort_by_relevance_then_value (Xapian::valueno *sort_key*, bool *ascending* = true)

Set the sorting to be by relevance then value.

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 reorder on. Sorting is with a string compare. If ascending is true (the default) higher is better; if ascending is false, lower is better.

ascending If true, documents values which sort higher by string compare are better. If false, the sort order is reversed. (default true)

7.6.3.18 void Xapian::Enquire::set_sort_by_value (Xapian::valueno *sort_key*, bool *ascending* = `true`)

Set the sorting to be by value only.

Parameters:

sort_key value number to reorder on. Sorting is with a string compare. If ascending is true (the default) higher is better; if ascending is false, lower is better.

ascending If true, documents values which sort higher by string compare are better. If false, the sort order is reversed. (default true)

7.6.3.19 void Xapian::Enquire::set_sort_by_value_then_relevance (Xapian::valueno *sort_key*, bool *ascending* = `true`)

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

Parameters:

sort_key value number to reorder on. Sorting is with a string compare. If ascending is true (the default) higher is better; if ascending is false, lower is better.

ascending If true, documents values which sort higher by string compare are better. If false, the sort order is reversed. (default true)

7.6.3.20 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.

7.6.3.21 Xapian::Enquire::XAPIAN_DEPRECATED (void *set_sorting*(Xapian::valueno sort_key, int sort_bands, bool sort_by_relevance=false))

For compatibility with Xapian 0.8.5 and earlier.

Deprecated

This method is now deprecated, use set_sort_by_relevance(), set_sort_by_value(), or set_sort_by_value_then_relevance() instead.

set_sorting(KEY, 1) -> set_sort_by_value(KEY)

set_sorting(KEY, 1, false) -> set_sort_by_value(KEY)

set_sorting(KEY, 1, true) -> set_sort_by_value_then_relevance(KEY)

set_sorting(ANYTHING, 0) -> set_sort_by_relevance()

set_sorting(Xapian::valueno(-1), ANYTHING) -> set_sort_by_relevance()

7.6.3.22 Xapian::Enquire::XAPIAN_DEPRECATED (void *set_sort_forward*(bool sort_forward))

For compatibility with Xapian 0.8.5 and earlier.

Deprecated

This method is now deprecated, use set_docid_order() instead - set_sort_forward(true) -> set_docid_order(ASCENDING) and set_sort_forward(false) -> set_docid_order(DESCENDING).

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

- include/xapian/enquire.h

7.7 Xapian::Error Class Reference

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

`#include <error.h>`

Public Member Functions

- std::string get_type () const
  The type of this error (e.g. "DocNotFoundError").
- const std::string & get_msg () const
  Message giving details of the error, intended for human consumption.
- const std::string & get_context () const
  Optional context information.
- int get_errno () const
  Optional value of 'errno' associated with this error.

Friends

- class **ErrorHandler**

7.7.1 Detailed Description

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

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

7.7.2 Member Function Documentation

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

Optional context information.

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

7.7.2.2 int Xapian::Error::get_errno () const [inline]

Optional value of 'errno' associated with this error.

If no 'errno' value is associated, returns 0.

7.7.2.3 const std::string& Xapian::Error::get_msg () const [inline]

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

7.7.2.4 std::string Xapian::Error::get_type () const [inline]

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

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

- include/xapian/error.h

7.8 Xapian::ErrorHandler Class Reference

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

`#include <errorhandler.h>`

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.

7.8.1 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).

7.8.2 Constructor & Destructor Documentation

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

Default constructor.

7.8.2.2 virtual Xapian::ErrorHandler::~ErrorHandler () [inline, virtual]

We require a virtual destructor because we have virtual methods.

7.8.3 Member Function Documentation

7.8.3.1 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:

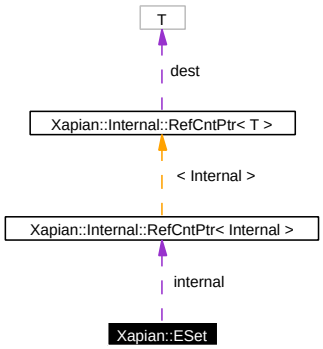
- include/xapian/errorhandler.h

7.9 Xapian::ESet Class Reference

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

`#include <enquire.h>`

Collaboration diagram for Xapian::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

  Introspection method.

Public Attributes

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

7.9.1 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().

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Xapian::ESet::ESet ()

Construct an empty ESet.

7.9.2.2 Xapian::ESet::~ESet ()

Destructor.

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

Copying is allowed (and is cheap).

7.9.3 Member Function Documentation

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

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

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

Iterator for the terms in this E-Set.

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

Test if this E-Set is empty.

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

End iterator corresponding to begin().

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

Introspection method.

Returns:
A string representing this ESet.

7.9.3.6 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()

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

Required to allow use as an STL container.

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

Assignment is allowed (and is cheap).

7.9.3.9]

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

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

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

The number of terms in this E-Set.

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

Swap the E-Set we point to with another.

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

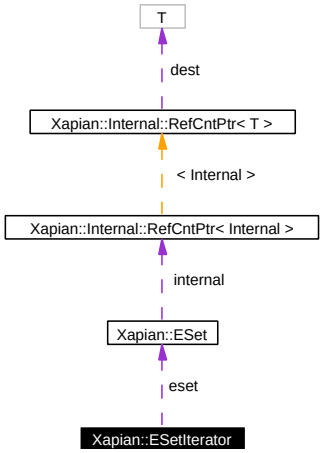
- include/xapian/enquire.h

7.10 Xapian::ESetIterator Class Reference

Iterate through terms in the ESet.

`#include <enquire.h>`

Collaboration diagram for Xapian::ESetIterator:



Public Types

- typedef std::bidirectional_iterator_tag iterator_category
  Allow use as an STL iterator.

- typedef std::string **value_type**
- typedef Xapian::termcount_diff **difference_type**
- typedef std::string * **pointer**
- typedef std::string & **reference**

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

  Returns a string describing this object.

Friends

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

7.10.1 Detailed Description

Iterate through terms in the ESet.

7.10.2 Member Typedef Documentation

7.10.2.1 typedef std::bidirectional_iterator_tag Xapian::ESetIterator::iterator_category

Allow use as an STL iterator.

7.10.3 Constructor & Destructor Documentation

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

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

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

Copying is allowed (and is cheap).

7.10.4 Member Function Documentation

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

Returns a string describing this object.

Introspection method.

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

Get the weight of the term at the current position.

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

Get the term for the current position.

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

Advance the iterator (postfix variant).

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

Advance the iterator.

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

Decrement the iterator (postfix variant).

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

Decrement the iterator.

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

Assignment is allowed (and is cheap).

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

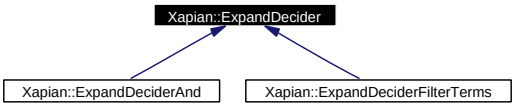
- include/xapian/enquire.h

7.11 Xapian::ExpandDecider Class Reference

Base class for expand decision functor.

`#include <enquire.h>`

Inheritance diagram for Xapian::ExpandDecider:



Public Member Functions

- virtual int operator() (const std::string &tname) const =0

  Decide whether we want this term to be in the expand set.

- virtual ~ExpandDecider ()

  Destructor.

7.11.1 Detailed Description

Base class for expand decision functor.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 virtual Xapian::ExpandDecider::~ExpandDecider () `[inline, virtual]`

Destructor.

7.11.3 Member Function Documentation

7.11.3.1 virtual int Xapian::ExpandDecider::operator() (const std::string & *tname*) const `[pure virtual]`

Decide whether we want this term to be in the expand set.

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

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

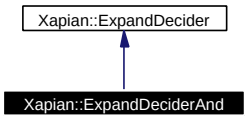
- include/xapian/enquire.h

7.12 Xapian::ExpandDeciderAnd Class Reference

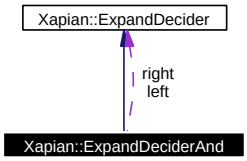
An expand decision functor which can be used to join two functors with an AND operation.

`#include <expanddecider.h>`

Inheritance diagram for Xapian::ExpandDeciderAnd:



Collaboration diagram for Xapian::ExpandDeciderAnd:



Public Member Functions

- ExpandDeciderAnd (const ExpandDecider *left_, const ExpandDecider *right_-
)

  Constructor, which takes as arguments the two decision functors to AND together.

- virtual int operator() (const std::string &tname) const

  Decide whether we want this term to be in the expand set.

7.12.1 Detailed Description

An expand decision functor which can be used to join two functors with an AND operation.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const ExpandDecider * *left_*, const ExpandDecider * *right_*)

Constructor, which takes as arguments the two decision functors to AND together.

ExpandDeciderAnd will not delete its sub-functors.

7.12.3 Member Function Documentation

7.12.3.1 virtual int Xapian::ExpandDeciderAnd::operator() (const std::string & *tname*) const [virtual]

Decide whether we want this term to be in the expand set.

Implements Xapian::ExpandDecider.

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

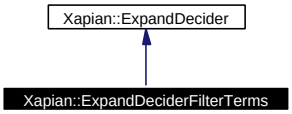
- include/xapian/expanddecider.h

7.13 Xapian::ExpandDeciderFilterTerms Class Reference

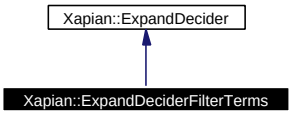
One useful expand decision functor, which provides a way of filtering out a fixed list of terms from the expand set.

`#include <expanddecider.h>`

Inheritance diagram for Xapian::ExpandDeciderFilterTerms:



Collaboration diagram for Xapian::ExpandDeciderFilterTerms:



Public Member Functions

- ExpandDeciderFilterTerms (Xapian::TermIterator terms, Xapian::TermIterator termsend)

  Constructor, which takes a list of terms which will be filtered out.

- virtual int operator() (const std::string &tname) const

  Decide whether we want this term to be in the expand set.

7.13.1 Detailed Description

One useful expand decision functor, which provides a way of filtering out a fixed list of terms from the expand set.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Xapian::ExpandDeciderFilterTerms::ExpandDeciderFilterTerms (Xapian::TermIterator *terms*, Xapian::TermIterator *termsend*)

Constructor, which takes a list of terms which will be filtered out.

7.13.3 Member Function Documentation

7.13.3.1 virtual int Xapian::ExpandDeciderFilterTerms::operator() (const std::string & *tname*) const [virtual]

Decide whether we want this term to be in the expand set.

Implements Xapian::ExpandDecider.

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

- include/xapian/expanddecider.h

7.14 Xapian::MatchDecider Class Reference

Base class for matcher decision functor.

`#include <enquire.h>`

Public Member Functions

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

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

- virtual ~MatchDecider ()

  Destructor.

7.14.1 Detailed Description

Base class for matcher decision functor.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 virtual Xapian::MatchDecider::~MatchDecider () [inline, virtual]

Destructor.

7.14.3 Member Function Documentation

7.14.3.1 virtual int Xapian::MatchDecider::operator() (const Xapian::Document & *doc*) const [pure virtual]

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

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

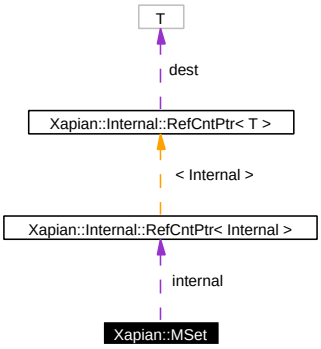
- include/xapian/enquire.h

7.15 Xapian::MSet Class Reference

A match set (MSet).

`#include <enquire.h>`

Collaboration diagram for Xapian::MSet:



Public Types

- typedef MSetIterator value_type
  Allow use as an STL container.

- typedef MSetIterator **iterator**
- typedef MSetIterator **const_iterator**
- typedef MSetIterator & **reference**
- typedef MSetIterator & **const_reference**
- typedef MSetIterator * **pointer**
- typedef Xapian::doccount_diff **difference_type**
- typedef Xapian::doccount **size_type**

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 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

  Returns a string representing the mset.

Public Attributes

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

7.15.1 Detailed Description

A match set (MSet).

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

7.15.2 Member Typedef Documentation

7.15.2.1 typedef MSetIterator Xapian::MSet::value_type

Allow use as an STL container.

7.15.3 Constructor & Destructor Documentation

7.15.3.1 Xapian::MSet::MSet ()

Create an empty Xapian::MSet.

7.15.3.2 Xapian::MSet::~MSet ()

Destroy a Xapian::MSet.

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

Copying is allowed (and is cheap).

7.15.4 Member Function Documentation

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

Iterator pointing to the last element of this MSet.

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

Iterator for the terms in this MSet.

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

Return the percentage score for a particular item.

7.15.4.4 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.

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

Test if this MSet is empty.

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

End iterator corresponding to begin().

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

Fetch all the items in the MSet.

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

Fetch the single item specified.

7.15.4.9 void Xapian::MSet::fetch (const MSetIterator & *begin*, const MSetIterator & *end*) const

Fetch the 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.

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

Returns a string representing the mset.

Introspection method.

7.15.4.11 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.

7.15.4.12 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.

7.15.4.13 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.

7.15.4.14 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.

7.15.4.15 Xapian::weight Xapian::MSet::get_max_attained () const

The greatest weight which is attained by any document in the database.

If firstitem == 0, this is the weight of the first entry in items.

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.

7.15.4.16 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.

7.15.4.17 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.

7.15.4.18 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.

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

Required to allow use as an STL container.

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

Assignment is allowed (and is cheap).

7.15.4.21]

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.

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

The number of items in this MSet.

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

Swap the MSet we point to with another.

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

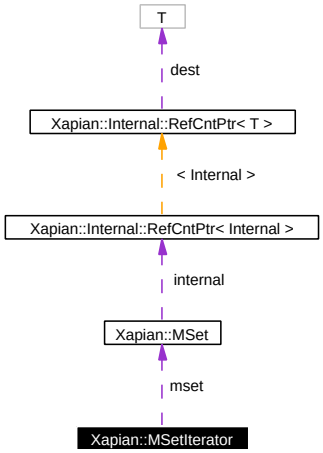
- include/xapian/enquire.h

7.16 Xapian::MSetIterator Class Reference

An iterator pointing to items in an MSet.

`#include <enquire.h>`

Collaboration diagram for Xapian::MSetIterator:



Public Types

- typedef std::bidirectional_iterator_tag iterator_category
  Allow use as an STL iterator.

- typedef Xapian::docid **value_type**
- typedef Xapian::doccount_diff **difference_type**
- typedef Xapian::docid * **pointer**
- typedef Xapian::docid & **reference**

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

  Returns a string describing this object.

Friends

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

7.16.1 Detailed Description

An iterator pointing to items in an MSet.

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

7.16.2 Member Typedef Documentation

7.16.2.1 typedef std::bidirectional_iterator_tag Xapian::MSetIterator::iterator_category

Allow use as an STL iterator.

7.16.3 Constructor & Destructor Documentation

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

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

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

Copying is allowed (and is cheap).

7.16.4 Member Function Documentation

7.16.4.1 Xapian::doccount Xapian::MSetIterator::get_collapse_count () const

Get an estimate of the number of documents that have been collapsed into this one.

The estimate will always be less than or equal to the actual number of other documents satisfying the match criteria with the same collapse key as this document.

This method may return 0 even though there are other documents with the same collapse key which satisfying the match criteria. However if this method returns non-zero, there definitely are other such documents. So this method may be used to inform the user that there are "at least N other matches in this group", or to control whether to offer a "show other documents in this group" feature (but note that it may not offer it in every case where it would show other documents).

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

Get the collapse key for this document.

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

Returns a string describing this object.

Introspection method.

7.16.4.4 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.

7.16.4.5 Xapian::percent Xapian::MSetIterator::get_percent () const

This returns the weight of the document as a percentage score.

The return value will be in the range 0 to 100: 0 meaning that the item did not match the query at all.

7.16.4.6 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 document judged "most relevant" will have rank of 0.

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

Get the weight of the document at the current position.

7.16.4.8 Xapian::docid Xapian::MSetIterator::operator * () const

Get the document ID for the current position.

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

Advance the iterator (postfix variant).

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

Advance the iterator.

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

Decrement the iterator (postfix variant).

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

Decrement the iterator.

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

Assignment is allowed (and is cheap).

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

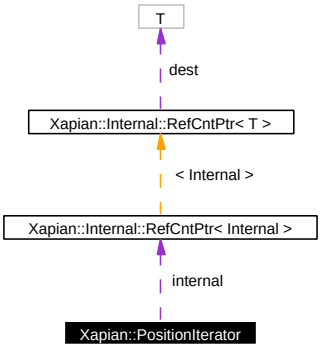
- include/xapian/enquire.h

7.17 Xapian::PositionIterator Class Reference

An iterator pointing to items in a list of positions.

`#include <positioniterator.h>`

Collaboration diagram for Xapian::PositionIterator:



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

  Returns a string describing this object.

Public Attributes

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

Friends

- class **PostingIterator**
- class **TermIterator**
- class **Database**
- bool operator== (const PositionIterator &a, const PositionIterator &b)

  Test equality of two PositionIterators.

7.17.1 Detailed Description

An iterator pointing to items in a list of positions.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Xapian::PositionIterator::PositionIterator ()

Default constructor - for declaring an uninitialised iterator.

7.17.2.2 Xapian::PositionIterator::~PositionIterator ()

Destructor.

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

Copying is allowed.

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

7.17.3 Member Function Documentation

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

Returns a string describing this object.

Introspection method.

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

Assignment is allowed.

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

7.17.4 Friends And Related Function Documentation

7.17.4.1 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:

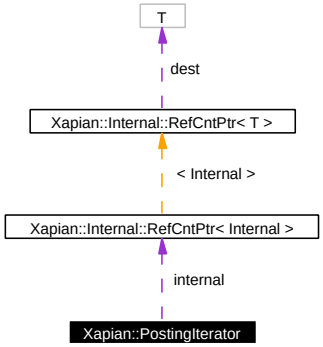
- include/xapian/positioniterator.h

7.18 Xapian::PostingIterator Class Reference

An iterator pointing to items in a list of postings.

`#include <postingiterator.h>`

Collaboration diagram for Xapian::PostingIterator:



Public Types

- typedef std::input_iterator_tag iterator_category
  Allow use as an STL iterator.

- typedef Xapian::docid **value_type**
- typedef Xapian::doccount_diff **difference_type**
- typedef Xapian::docid * **pointer**
- typedef Xapian::docid & **reference**

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

  Returns a string describing this object.

Public Attributes

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

Friends

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

  Test equality of two PostingIterators.

7.18.1 Detailed Description

An iterator pointing to items in a list of postings.

7.18.2 Member Typedef Documentation

7.18.2.1 typedef std::input_iterator_tag Xapian::PostingIterator::iterator_category

Allow use as an STL iterator.

7.18.3 Constructor & Destructor Documentation

7.18.3.1 Xapian::PostingIterator::PostingIterator ()

Default constructor - for declaring an uninitialised iterator.

7.18.3.2 Xapian::PostingIterator::~PostingIterator ()

Destructor.

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

Copying is allowed.

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

7.18.4 Member Function Documentation

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

Returns a string describing this object.

Introspection method.

7.18.4.2 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.

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

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

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

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

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

Assignment is allowed.

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

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

Return PositionIterator pointing to start of positionlist for current document.

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

Return PositionIterator pointing to end of positionlist for current document.

7.18.4.8 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.

7.18.5 Friends And Related Function Documentation

7.18.5.1 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:

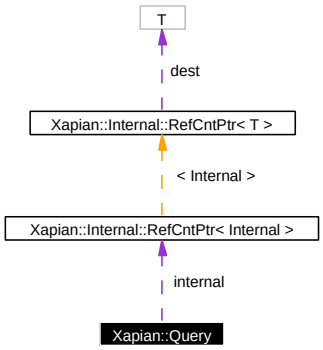
- include/xapian/postingiterator.h

7.19 Xapian::Query Class Reference

Class representing a query.

`#include <query.h>`

Collaboration diagram for Xapian::Query:



Public Types

- enum op {
  OP_AND, OP_OR, OP_AND_NOT, OP_XOR,
  OP_AND_MAYBE, OP_FILTER, OP_NEAR, OP_PHRASE,
  OP_ELITE_SET = 10 }
  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.

- 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.

- XAPIAN_DEPRECATED (bool is_empty() const)
- std::string get_description () const

  Returns a string representing the query.

- template<class Iterator> **Query** (Query::op op_, Iterator qbegin, Iterator qend, termcount parameter)

Public Attributes

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

Classes

- class Internal

  Internal class, implementing most of Xapian::Query.

7.19.1 Detailed Description

Class representing a query.

Queries are represented as a tree of objects.

7.19.2 Member Enumeration Documentation

7.19.2.1 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_ELITE_SET Select an elite set from the subqueries, and perform a query with these combined as an OR query.

7.19.3 Constructor & Destructor Documentation

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

Copy constructor.

7.19.3.2 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.

7.19.3.3 Xapian::Query::~Query ()

Destructor.

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

A query consisting of a single term.

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

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

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

A query consisting of two termnames opp-ed together.

7.19.3.7 template<class Iterator> Xapian::Query::Query (Query::op *op_*, Iterator *qbegin*, Iterator *qend*, Xapian::termcount *parameter* = 0)

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.

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

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

7.19.4 Member Function Documentation

7.19.4.1 bool Xapian::Query::empty () const

Test if the query is empty (i.e.

was constructed using the default ctor or with an empty iterator ctor).

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

Returns a string representing the query.

Introspection method.

7.19.4.3 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().

7.19.4.4 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.

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

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

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

Assignment.

7.19.4.7 Xapian::Query::XAPIAN_DEPRECATED (bool is_empty() *const*)

Deprecated

Deprecated alias for empty()

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

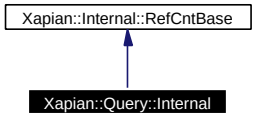
- include/xapian/query.h

7.20 Xapian::Query::Internal Class Reference

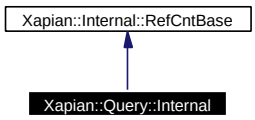
Internal class, implementing most of Xapian::Query.

`#include <query.h>`

Inheritance diagram for Xapian::Query::Internal:



Collaboration diagram for Xapian::Query::Internal:



Public Types

- typedef std::vector< Internal * > subquery_list

  The container type for storing pointers to subqueries.

- typedef int op_t

  Type storing the operation.

Public Member Functions

- Internal (const Query::Internal &copyme)

  Copy constructor.

- void operator= (const Query::Internal &copyme)

  Assignment.

- Internal (const std::string &tname_, Xapian::termcount wqf_=1, Xapian::termpos term_pos_=0)

  A query consisting of a single term.

- Internal (op_t op_, Xapian::termcount parameter)

  Create internals given only the operator and a parameter.

- ~Internal ()

  Destructor.

- void add_subquery (const Query::Internal &subq)

  Add a subquery.

- Query::Internal * end_construction ()

  Finish off the construction.

- std::string serialise () const

  Return a string in an easily parsed form which contains all the information in a query.

- std::string get_description () const

  Returns a string representing the query.

- Xapian::termcount get_length () const

  Get the length of the query, used by some ranking formulae.

- TermIterator get_terms () const

  Return an iterator over all the terms in the query, in order of termpos.

Static Public Member Functions

- static Xapian::Query::Internal * **unserialise** (const std::string &s)

Static Public Attributes

- static const int **OP_LEAF** = -1

Friends

- class **::MultiMatch**
- class **::LocalSubMatch**
- struct **::SortPosName**
- std::string **serialise_qint_** (const Xapian::Query::Internal *qint, Xapian::termpos &curpos)

7.20.1 Detailed Description

Internal class, implementing most of Xapian::Query.

7.20.2 Member Typedef Documentation

7.20.2.1 typedef int Xapian::Query::Internal::op_t

Type storing the operation.

7.20.2.2 typedef std::vector<Internal *> Xapian::Query::Internal::subquery_list

The container type for storing pointers to subqueries.

7.20.3 Constructor & Destructor Documentation

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

Copy constructor.

7.20.3.2 Xapian::Query::Internal::Internal (const std::string & *tname_*, Xapian::termcount *wqf_* = 1, Xapian::termpos *term_pos_* = 0)

A query consisting of a single term.

7.20.3.3 Xapian::Query::Internal::Internal (op_t *op_*, Xapian::termcount *parameter*)

Create internals given only the operator and a parameter.

7.20.3.4 Xapian::Query::Internal::~Internal ()

Destructor.

7.20.4 Member Function Documentation

7.20.4.1 void Xapian::Query::Internal::add_subquery (const Query::Internal & *subq*)

Add a subquery.

7.20.4.2 Query::Internal* Xapian::Query::Internal::end_construction ()

Finish off the construction.

7.20.4.3 std::string Xapian::Query::Internal::get_description () const

Returns a string representing the query.

Introspection method.

7.20.4.4 Xapian::termcount Xapian::Query::Internal::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().

7.20.4.5 TermIterator Xapian::Query::Internal::get_terms () const

Return an iterator over 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.

7.20.4.6 void Xapian::Query::Internal::operator= (const Query::Internal & *copyme*)

Assignment.

7.20.4.7 std::string Xapian::Query::Internal::serialise () const

Return a string in an easily parsed form which contains all the information in a query.

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

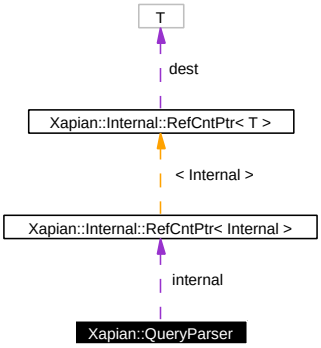
- include/xapian/query.h

7.21 Xapian::QueryParser Class Reference

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

`#include <queryparser.h>`

Collaboration diagram for Xapian::QueryParser:



Public Types

- enum feature_flag {

  FLAG_BOOLEAN = 1, FLAG_PHRASE = 2, FLAG_LOVEHATE = 4, FLAG_BOOLEAN_ANY_CASE = 8,

  FLAG_WILDCARD = 16 }

  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.

- XAPIAN_DEPRECATED (void set_stemming_options(const std::string &lang, bool stem_all=false, const Stopper *stop=NULL))
- void set_default_op (Query::op default_op)

  Set the default boolean operator.

- Query::op get_default_op () const

  Get the default boolean 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)

  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
- std::string get_description () const

  Return a string describing this object.

Public Attributes

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

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 bracketted 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 using set_database.

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 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.

E.g. qp.add_boolean_prefix("site", "H");

Allows the user to restrict a search with site:xapian.org which will be converted to Hxapian.org combined with any probabilistic query with OP_FILTER.

Multiple fields can be mapped to the same prefix (so you can e.g. make site: and domain: aliases for each other).

Parameters:

field The user visible field name

prefix The term prefix to map this to

7.21.4.2 void Xapian::QueryParser::add_prefix (const std::string & *field*, const std::string & *prefix*)

Add a probabilistic term prefix.

E.g. qp.add_prefix("author", "A");

Allows the user to search for author:orwell which will search for the term "Aorwel" (assuming English stemming is in use). Multiple fields can be mapped to the same prefix (so you can e.g. make title: and subject: aliases for each other).

Parameters:

field The user visible field name

prefix The term prefix to map this to

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

Get the default boolean operator.

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

Return a string describing this object.

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

Assignment.

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

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 (|).

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

Specify the database being searched.

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

Set the default boolean operator.

7.21.4.9 void Xapian::QueryParser::set_stemmer (const Xapian::Stem & *stemmer*)

Set the stemmer.

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

Set the stemming strategy.

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

Set the stopper.

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

Iterate over terms omitted from the query as stopwords.

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

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

7.21.4.14 Xapian::QueryParser::XAPIAN_DEPRECATED (void *set_stemming_options*(const std::string &lang, bool stem_all=false, const Stopper *stop=NULL))

Deprecated

Deprecated method for backward compatibility.

Use set_stemmer, set_stemming_strategy and/or set_stopper instead.

set_stemming_options("") -> set_stemming_strategy(Xapian::QueryParser::STEM_NONE)

set_stemming_options("none") -> set_stemming_strategy(Xapian::QueryParser::STEM_NONE)

set_stemming_options(LANG) -> set_stemmer(Xapian::Stem(LANG); set_stemming_strategy(Xapian::QueryParser::STEM_SOME)

set_stemming_options(LANG, false) -> set_stemmer(Xapian::Stem(LANG); set_stemming_strategy(Xapian::QueryParser::STEM_SOME)

set_stemming_options(LANG, true) -> set_stemmer(Xapian::Stem(LANG); set_stemming_strategy(Xapian::QueryParser::STEM_ALL)

If a third parameter is passed, set_stopper(PARAM3) and treat the first two parameters as above.

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

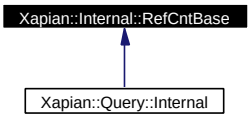
- include/xapian/queryparser.h

7.22 Xapian::Internal::RefCntBase Class Reference

Reference counted internal classes should inherit from RefCntBase.

`#include <base.h>`

Inheritance diagram for Xapian::Internal::RefCntBase:



Public Types

- typedef unsigned int **ref_count_t**

Public Member Functions

- RefCntBase ()

  The constructor, which initialises the ref_count to 0.

Public Attributes

- ref_count_t ref_count

  The actual reference count.

Protected Member Functions

- RefCntBase (const RefCntBase &)

  The copy constructor.

7.22.1 Detailed Description

Reference counted internal classes should inherit from RefCntBase.

This gives the object a reference count used by RefCntPtr.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 Xapian::Internal::RefCntBase::RefCntBase (const RefCntBase &) [inline, protected]

The copy constructor.

This is protected since it'll only be used by derived classes, which should only rarely need copying (this is, after all, a refcount implementation). Sometimes it's needed, though, since we need to zero ref_count in the copy.

7.22.2.2 Xapian::Internal::RefCntBase::RefCntBase () [inline]

The constructor, which initialises the ref_count to 0.

7.22.3 Member Data Documentation

7.22.3.1 ref_count_t Xapian::Internal::RefCntBase::ref_count [mutable]

The actual reference count.

It's mutable so we can have reference counting work with const pointers.

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

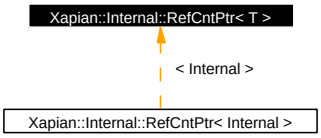
- include/xapian/base.h

7.23 Xapian::Internal::RefCntPtr< T > Class Template Reference

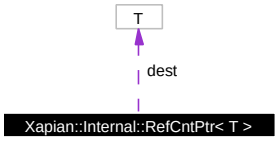
A reference-counted pointer.

`#include <base.h>`

Inheritance diagram for Xapian::Internal::RefCntPtr< T >:



Collaboration diagram for Xapian::Internal::RefCntPtr< T >:



Public Member Functions

- T * **operator** -> () const
- T & **operator** * () const
- T * **get** () const
- RefCntPtr (T *dest_)

  Make a RefCntPtr for an object which may already have reference counted pointers.

- **RefCntPtr** (const RefCntPtr &other)
- void **operator=** (const RefCntPtr &other)
- void **operator=** (T *dest_)
- template<class U> **RefCntPtr** (const RefCntPtr< U > &other)

7.23.1 Detailed Description

template<class T> class Xapian::Internal::RefCntPtr< T >

A reference-counted pointer.

Can be used with any class derived from RefCntBase, as long as it is allocated on the heap by new (not new[]!).

7.23.2 Constructor & Destructor Documentation

7.23.2.1 template<class T> Xapian::Internal::RefCntPtr< T >::RefCntPtr (T * *dest_*) [inline]

Make a RefCntPtr for an object which may already have reference counted pointers.

You usually pass in a newly created object, or an object may pass in "this" to get a RefCntPtr to itself to pass to other classes. (e.g. a database might pass a newly created postlist a reference counted pointer to itself.)

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

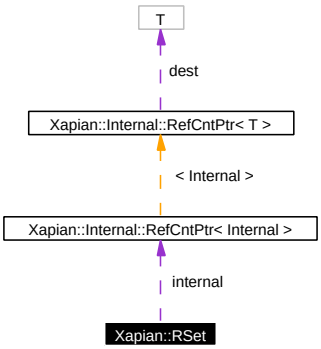
- include/xapian/base.h

7.24 Xapian::RSet Class Reference

A relevance set (R-Set).

`#include <enquire.h>`

Collaboration diagram for Xapian::RSet:



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)

  Test if a given document in the relevance set.

- std::string get_description () const

  Introspection method.

Public Attributes

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

7.24.1 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.

7.24.2 Constructor & Destructor Documentation

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

Copy constructor.

7.24.2.2 Xapian::RSet::RSet ()

Default constructor.

7.24.2.3 Xapian::RSet::~RSet ()

Destructor.

7.24.3 Member Function Documentation

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

Add a document to the relevance set.

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

Add a document to the relevance set.

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

Test if a given document in the relevance set.

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

Test if a given document in the relevance set.

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

Test if this R-Set is empty.

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

Introspection method.

Returns:
A string representing this RSet.

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

Assignment operator.

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

Remove a document from the relevance set.

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

Remove a document from the relevance set.

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

The number of documents in this R-Set.

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

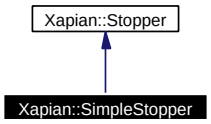
- include/xapian/enquire.h

7.25 Xapian::SimpleStopper Class Reference

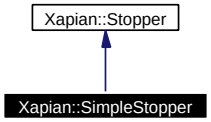
Simple implementation of Stopper class - this will suit most users.

`#include <queryparser.h>`

Inheritance diagram for Xapian::SimpleStopper:



Collaboration 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.

7.25.1 Detailed Description

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

7.25.2 Constructor & Destructor Documentation

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

Default constructor.

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

Initialise from a pair of iterators.

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

Destructor.

7.25.3 Member Function Documentation

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

Add a single stop word.

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

Return a string describing this object.

Reimplemented from Xapian::Stopper.

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

Is term a stop-word?

Implements Xapian::Stopper.

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

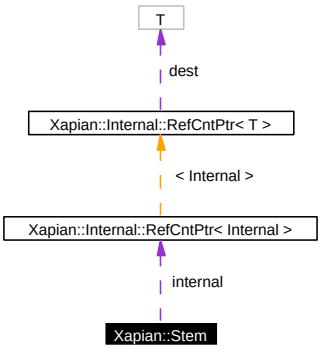
- include/xapian/queryparser.h

7.26 Xapian::Stem Class Reference

Class representing a stemming algorithm.

`#include <stem.h>`

Collaboration diagram for Xapian::Stem:



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 stem_word (const std::string &word) const
  For compatibility with Xapian 0.8.5 and earlier.
- 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.

Public Attributes

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

7.26.1 Detailed Description

Class representing a stemming algorithm.

7.26.2 Constructor & Destructor Documentation

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

Copy constructor.

7.26.2.2 Xapian::Stem::Stem ()

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

Equivalent to Stem("none").

7.26.2.3 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.

7.26.2.4 Xapian::Stem::~Stem ()

Destructor.

7.26.3 Member Function Documentation

7.26.3.1 static std::string Xapian::Stem::get_available_languages () [static]

Return a list of available languages.

Each stemmer is only included once in the list (not once for each alias). The name included is the English name of the language.

The list is returned as a string, with language names separated by spaces. This is a static method, so a Xapian::Stem object is not required for this operation.

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

Return a string describing this object.

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

Stem a word.

Parameters:
word a word to stem.

Returns:
the stem

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

Assignment.

7.26.3.5 std::string Xapian::Stem::stem_word (const std::string & *word*) const [inline]

For compatibility with Xapian 0.8.5 and earlier.

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

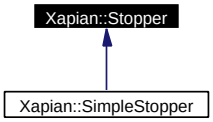
- include/xapian/stem.h

7.27 Xapian::Stopper Class Reference

Base class for stop-word decision functor.

`#include <queryparser.h>`

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.

7.27.1 Detailed Description

Base class for stop-word decision functor.

7.27.2 Constructor & Destructor Documentation

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

Class has virtual methods, so provide a virtual destructor.

7.27.3 Member Function Documentation

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

Return a string describing this object.

Reimplemented in Xapian::SimpleStopper.

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

Is term a stop-word?

Implemented in Xapian::SimpleStopper.

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

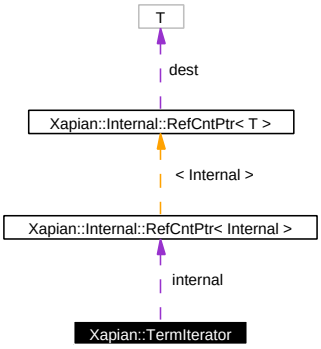
- include/xapian/queryparser.h

7.28 Xapian::TermIterator Class Reference

An iterator pointing to items in a list of terms.

`#include <termiterator.h>`

Collaboration diagram for Xapian::TermIterator:



Public Types

- typedef std::input_iterator_tag iterator_category
  Allow use as an STL iterator.

- typedef std::string **value_type**
- typedef Xapian::termcount_diff **difference_type**
- typedef std::string * **pointer**
- typedef std::string & **reference**

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

  Returns a string describing this object.

Public Attributes

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

7.28.1 Detailed Description

An iterator pointing to items in a list of terms.

7.28.2 Member Typedef Documentation

7.28.2.1 typedef std::input_iterator_tag Xapian::TermIterator::iterator_category

Allow use as an STL iterator.

7.28.3 Constructor & Destructor Documentation

7.28.3.1 Xapian::TermIterator::TermIterator ()

Default constructor - for declaring an uninitialised iterator.

7.28.3.2 Xapian::TermIterator::~TermIterator ()

Destructor.

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

Copying is allowed.

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

7.28.4 Member Function Documentation

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

Returns a string describing this object.

Introspection method.

7.28.4.2 Xapian::doccount Xapian::TermIterator::get_termfreq () const

Return the term frequency of the current term (if meaningful).

7.28.4.3 Xapian::termcount Xapian::TermIterator::get_wdf () const

Return the wdf of the current term (if meaningful).

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

Return the current term.

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

Assignment is allowed.

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

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

Return PositionIterator pointing to start of positionlist for current term.

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

Return length of positionlist for current term.

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

Return PositionIterator pointing to end of positionlist for current term.

7.28.4.9 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.

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

- include/xapian/termiterator.h

7.29 Xapian::TermNameWrapper Class Reference

A wrapper class for a termname which returns the termname if dereferenced with ∗.

`#include <termiterator.h>`

Public Member Functions

- **TermNameWrapper** (const std::string &tname_)
- const std::string & **operator** ∗ () const

7.29.1 Detailed Description

A wrapper class for a termname which returns the termname if dereferenced with ∗.

We need this to implement input_iterator semantics.

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

- include/xapian/termiterator.h

7.30 Xapian::TermPosWrapper Class Reference

A wrapper class for a termpos which returns the termpos if dereferenced with *.

`#include <positioniterator.h>`

Public Member Functions

- **TermPosWrapper** (termpos pos_)
- termpos **operator** * () const

7.30.1 Detailed Description

A wrapper class for a termpos which returns the termpos if dereferenced with *.

We need this to implement input_iterator semantics.

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

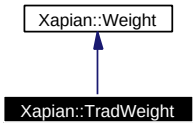
- include/xapian/positioniterator.h

7.31 Xapian::TradWeight Class Reference

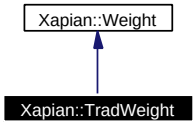
Traditional probabilistic weighting scheme.

`#include <enquire.h>`

Inheritance diagram for Xapian::TradWeight:



Collaboration 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

7.31.1 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.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 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.

7.31.3 Member Function Documentation

7.31.3.1 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.

7.31.3.2 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.

7.31.3.3 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.

7.31.3.4 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.

7.31.3.5 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.

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

return false if the weight object doesn't need doclength

Reimplemented from Xapian::Weight.

7.31.3.7 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.

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

Serialise object parameters into a string.

Implements Xapian::Weight.

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

Create object given string serialisation returned by serialise().

Implements Xapian::Weight.

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

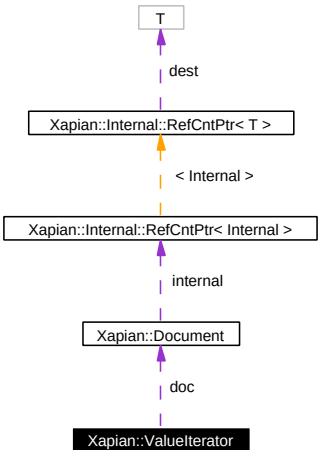
- include/xapian/enquire.h

7.32 Xapian::ValueIterator Class Reference

An iterator pointing to values associated with a document.

`#include <valueiterator.h>`

Collaboration diagram for Xapian::ValueIterator:



Public Types

- typedef std::input_iterator_tag iterator_category
  Allow use as an STL iterator.

- typedef std::string **value_type**
- typedef Xapian::valueno_diff **difference_type**
- typedef std::string * **pointer**
- typedef std::string & **reference**

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

  Returns a string describing this object.

Friends

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

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.

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

Returns a string describing this object.

Introspection method.

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

Get the number of the value at the current position.

7.32.4.3 const std::string& Xapian::ValueIterator::operator ∗ () const

Get the value for the current position.

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

Advance the iterator (postfix variant).

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

Advance the iterator.

7.32.4.6 const std::string∗ Xapian::ValueIterator::operator → () const

Get the value for the current position.

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

Assignment is allowed (and is cheap).

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

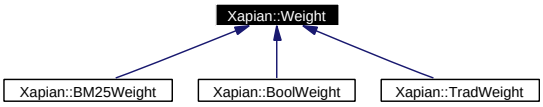
- include/xapian/valueiterator.h

7.33 Xapian::Weight Class Reference

Abstract base class for weighting schemes.

`#include <enquire.h>`

Inheritance diagram for Xapian::Weight:



Public Member Functions

- Weight * create (const Internal *internal_, Xapian::doclength querysize_-, Xapian::termcount wqf_, 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**

Friends

- class **Enquire**
- class **::RemoteServer**

7.33.1 Detailed Description

Abstract base class for weighting schemes.

7.33.2 Member Function Documentation

7.33.2.1 Weight* Xapian::Weight::create (const Internal * *internal_*, Xapian::doclength *querysize_*, Xapian::termcount *wqf_*, std::string *tname_*) const [inline]

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.

7.33.2.2 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.

7.33.2.3 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.

7.33.2.4 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.

7.33.2.5 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.

7.33.2.6 virtual bool Xapian::Weight::get_sumpart_needs_doclength () const [inline, virtual]

return false if the weight object doesn't need doclength

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

7.33.2.7 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.

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

Serialise object parameters into a string.

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

7.33.2.9 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.

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

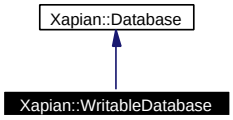
- include/xapian/enquire.h

7.34 Xapian::WritableDatabase Class Reference

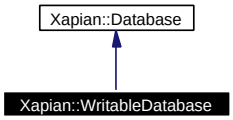
This class provides read/write access to a database.

`#include <database.h>`

Inheritance diagram for Xapian::WritableDatabase:



Collaboration 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** (Database::Internal *internal)
- 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.

- std::string get_description () const

  Introspection method.

7.34.1 Detailed Description

This class provides read/write access to a database.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 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.

7.34.2.2 Xapian::WritableDatabase::WritableDatabase ()

Create an empty WritableDatabase.

7.34.2.3 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

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

Copying is allowed.

The internals are reference counted, so copying is cheap.

7.34.3 Member Function Documentation

7.34.3.1 Xapian::docid Xapian::WritableDatabase::add_document (const Xapian::Document & *document*)

Add a new document to the database.

This method adds the specified document to the database, returning a newly allocated document ID. Automatically allocated document IDs come from a per-database monotonically increasing counter, so IDs from deleted documents won't be reused.

If you want to specify the document ID to be used, you should call replace_document() instead.

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

7.34.3.2 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.

7.34.3.3 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.

7.34.3.4 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.

7.34.3.5 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.

The intended use is to allow UIDs from another system to easily be mapped to terms in Xapian, although this method probably has other uses.

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.

7.34.3.6 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.

7.34.3.7 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.

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::DatabaseLockError will be thrown if a lock couldn't be acquired on the database.

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

Introspection method.

Returns:
A string describing this object.

Reimplemented from Xapian::Database.

7.34.3.9 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.

7.34.3.10 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.

The intended use is to allow UIDs from another system to easily be mapped to terms in Xapian, although this method probably has other uses.

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.

7.34.3.11 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*.

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.

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

- include/xapian/database.h

Chapter 8

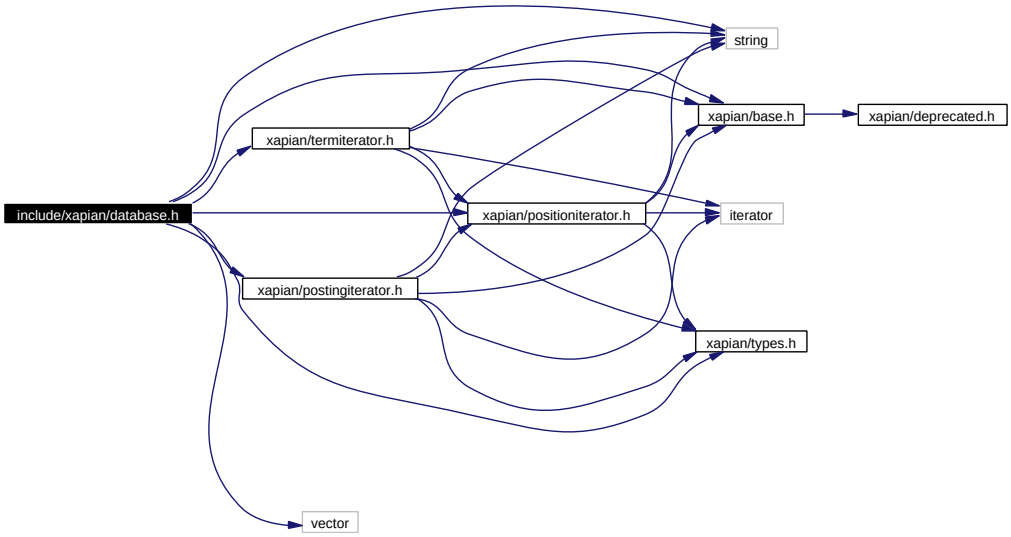
xapian-core File Documentation

8.1 include/xapian/database.h File Reference

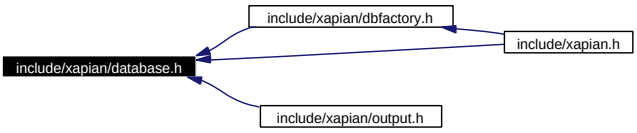
API for working with Xapian databases.

```
#include <string>
#include <vector>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
#include <xapian/postingiterator.h>
#include <xapian/termiterator.h>
```

Include dependency graph for database.h:



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



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.

8.1.1 Detailed Description

API for working with Xapian databases.

8.2 include/xapian/dbfactory.h File Reference

Factory functions for constructing Database and WritableDatabase objects.

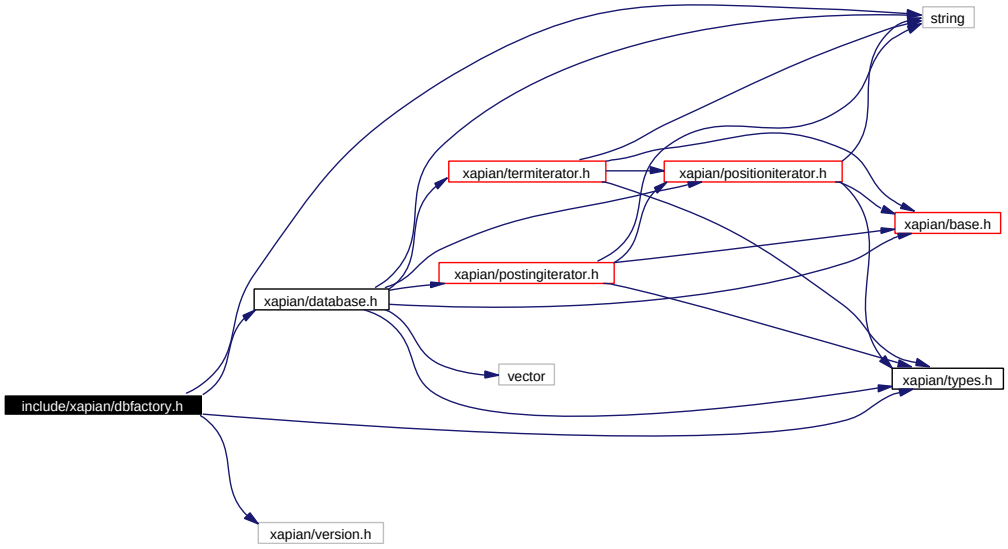
`#include <string>`

`#include <xapian/types.h>`

`#include <xapian/database.h>`

`#include <xapian/version.h>`

Include dependency graph for dbfactory.h:



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

Namespaces

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

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

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

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

- WritableDatabase **Xapian::Remote::open_writable** (const std::string &host, unsigned int port, Xapian::timeout timeout, Xapian::timeout connect_timeout)

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

Functions

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

  Construct a Database object for a stub database file.

- Xapian::Auto::XAPIAN_DEPRECATED (Database open(const std::string &path))

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

- Database Xapian::Auto::open (const std::string &path)

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

- Xapian::Auto::XAPIAN_DEPRECATED (WritableDatabase open(const std::string &path, int action))

  Construct a WritableDatabase object for update access to a database.

- WritableDatabase **Xapian::Auto::open** (const std::string &path, int action)
- WritableDatabase Xapian::InMemory::open ()

  Construct a Database object for update access to an InMemory database.

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

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

- WritableDatabase Xapian::Quartz::open (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 &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=10000)

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

8.2.1 Detailed Description

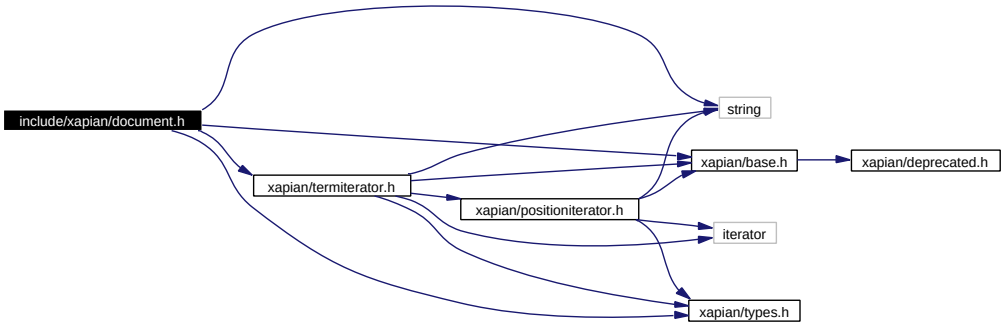
Factory functions for constructing Database and WritableDatabase objects.

8.3 include/xapian/document.h File Reference

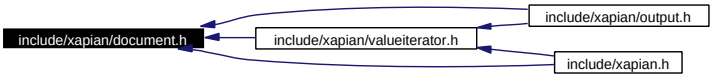
API for working with documents.

```
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
```

Include dependency graph for document.h:



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



Namespaces

- namespace Xapian

Classes

- class Xapian::Document

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

8.3.1 Detailed Description

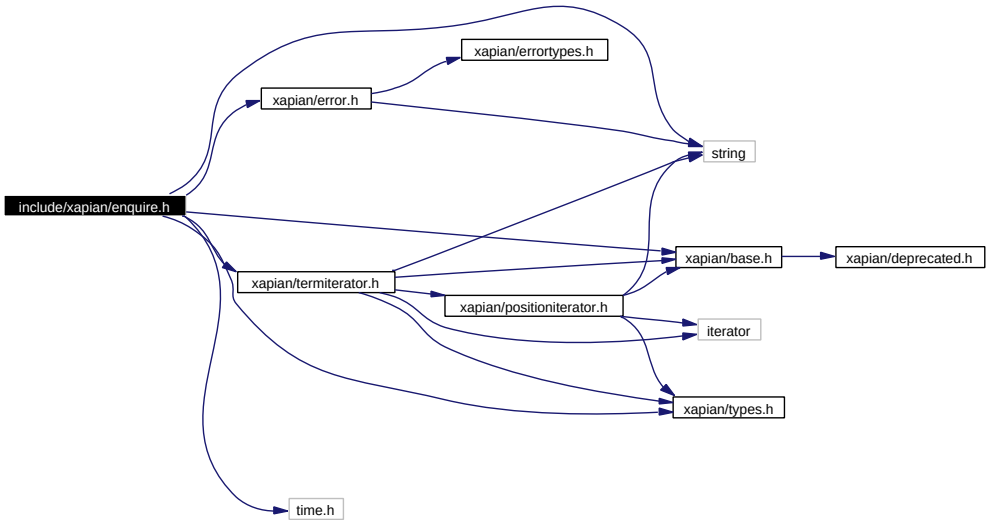
API for working with documents.

8.4 include/xapian/enquire.h File Reference

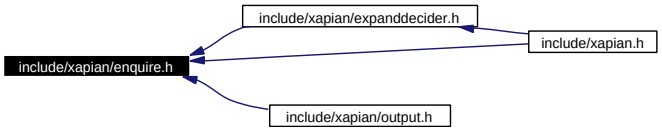
API for running queries.

```
#include <string>
#include <time.h>
#include <xapian/base.h>
#include <xapian/error.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
```

Include dependency graph for enquire.h:



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



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::ExpandDecider

  Base class for expand 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)
- bool **Xapian::operator!=** (const MSetIterator &a, const MSetIterator &b)
- bool **Xapian::operator==** (const ESetIterator &a, const ESetIterator &b)
- bool **Xapian::operator!=** (const ESetIterator &a, const ESetIterator &b)

8.4.1 Detailed Description

API for running queries.

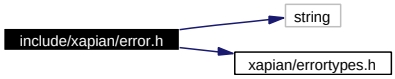
8.5 include/xapian/error.h File Reference

Hierarchy of classes which Xapian can throw as exceptions.

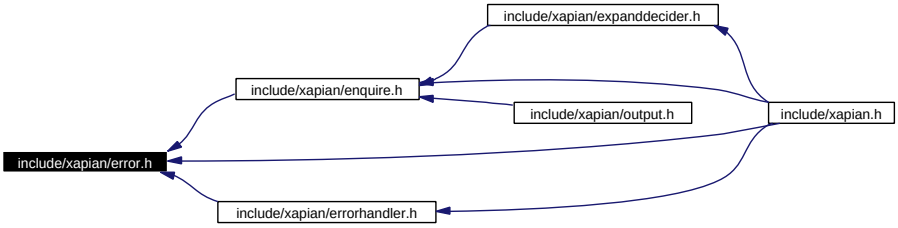
`#include <string>`

`#include <xapian/errortypes.h>`

Include dependency graph for error.h:



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



Namespaces

- namespace Xapian

Classes

- class Xapian::Error

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

Defines

- #define **XAPIAN_EXCEPTION_EXPORT**
- #define XAPIAN_DEFINE_ERROR_BASECLASS(CLASS, PARENT)

  For internal use only.

  Macro to define an abstract Xapian::Error subclass.

- #define XAPIAN_DEFINE_ERROR_CLASS(CLASS, PARENT)

  For internal use only.

  Macro to define a concrete Xapian::Error subclass.

8.5.1 Detailed Description

Hierarchy of classes which Xapian can throw as exceptions.

8.5.2 Define Documentation

8.5.2.1 #define XAPIAN_DEFINE_ERROR_BASECLASS(CLASS, PARENT)

Value:

```
class XAPIAN_EXCEPTION_EXPORT CLASS : public PARENT { \
  protected: \ \
    CLASS(const std::string &msg_, const std::string &context_, \
          const char * type_, int errno_) \
        : PARENT(msg_, context_, type_, errno_) {} \
}
```

For internal use only.

Macro to define an abstract Xapian::Error subclass.

8.5.2.2 #define XAPIAN_DEFINE_ERROR_CLASS(CLASS, PARENT)

Value:

```
class XAPIAN_EXCEPTION_EXPORT CLASS : public PARENT { \
  public: \ \
    CLASS(const std::string &msg_, const std::string &context_ = "", \
          int errno_ = 0) \
        : PARENT(msg_, context_, #CLASS, errno_) {} \ \
    CLASS(const std::string &msg_, int errno_) \
        : PARENT(msg_, "", #CLASS, errno_) {} \
  protected: \ \
    CLASS(const std::string &msg_, const std::string &context_, \
          const char * type_, int errno_) \
        : PARENT(msg_, context_, type_, errno_) {} \
}
```

For internal use only.

Macro to define a concrete Xapian::Error subclass.

8.6 include/xapian/errorhandler.h File Reference

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

`#include <xapian/error.h>`

Include dependency graph for errorhandler.h:

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

Classes

- class Xapian::ErrorHandler

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

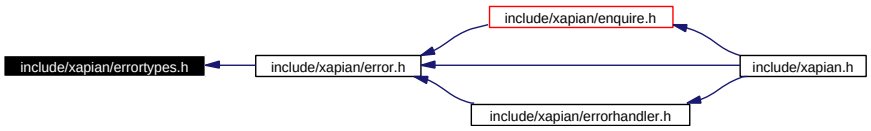
8.6.1 Detailed Description

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

8.7 include/xapian/errortypes.h File Reference

Exception subclasses.

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



Functions

- XAPIAN_DEFINE_ERROR_BASECLASS (LogicError, Error)

  Base class for errors due to programming errors.

- XAPIAN_DEFINE_ERROR_BASECLASS (RuntimeError, Error)

  Base class for errors due to run time problems.

- XAPIAN_DEFINE_ERROR_CLASS (AssertionError, LogicError)

  Thrown if an internal consistency check fails.

- XAPIAN_DEFINE_ERROR_CLASS (UnimplementedError, LogicError)

  Thrown when an attempt to use an unimplemented feature is made.

- XAPIAN_DEFINE_ERROR_CLASS (InvalidArgumentError, LogicError)

  Thrown when an invalid argument is supplied to the API.

- XAPIAN_DEFINE_ERROR_CLASS (InvalidOperationError, LogicError)

  Thrown when API calls are made in an invalid way.

- XAPIAN_DEFINE_ERROR_CLASS (DocNotFoundError, RuntimeError)

  Thrown when an attempt is made to access a document which is not in the database.

- XAPIAN_DEFINE_ERROR_CLASS (RangeError, RuntimeError)

  thrown when an element is out of range.

- XAPIAN_DEFINE_ERROR_CLASS (InternalError, RuntimeError)

  thrown when really weird stuff happens.

- XAPIAN_DEFINE_ERROR_CLASS (DatabaseError, RuntimeError)

  thrown for miscellaneous database errors.

- XAPIAN_DEFINE_ERROR_CLASS (FeatureUnavailableError, RuntimeError)

  Thrown if a feature is unavailable - usually due to not being compiled in.

- XAPIAN_DEFINE_ERROR_CLASS (NetworkError, RuntimeError)

  thrown when there is a communications problem with a remote database.

- XAPIAN_DEFINE_ERROR_CLASS (NetworkTimeoutError, NetworkError)

  Thrown when a network timeout is exceeded.

- XAPIAN_DEFINE_ERROR_CLASS (DatabaseCorruptError, DatabaseError)

  thrown if the database is corrupt.

- XAPIAN_DEFINE_ERROR_CLASS (DatabaseCreateError, DatabaseError)

  Thrown when creating a database fails.

- XAPIAN_DEFINE_ERROR_CLASS (DatabaseOpeningError, DatabaseError)

  Thrown when opening a database fails.

- XAPIAN_DEFINE_ERROR_CLASS (DatabaseLockError, DatabaseError)

  Thrown when gaining a lock on a database fails.

- XAPIAN_DEFINE_ERROR_CLASS (DatabaseModifiedError, DatabaseError)

  Thrown when a database has been modified whilst being read.

8.7.1 Detailed Description

Exception subclasses.

8.7.2 Function Documentation

8.7.2.1 XAPIAN_DEFINE_ERROR_BASECLASS (RuntimeError, Error)

Base class for errors due to run time problems.

An exception derived from RuntimeError is thrown when an error is caused by problems with the data or environment rather than a programming mistake.

8.7.2.2 XAPIAN_DEFINE_ERROR_BASECLASS (LogicError, Error)

Base class for errors due to programming errors.

An exception derived from LogicError is thrown when a misuse of the API is detected.

8.7.2.3 XAPIAN_DEFINE_ERROR_CLASS (DatabaseModifiedError, DatabaseError)

Thrown when a database has been modified whilst being read.

8.7.2.4 XAPIAN_DEFINE_ERROR_CLASS (DatabaseLockError, DatabaseError)

Thrown when gaining a lock on a database fails.

8.7.2.5 XAPIAN_DEFINE_ERROR_CLASS (DatabaseOpeningError, DatabaseError)

Thrown when opening a database fails.

8.7.2.6 XAPIAN_DEFINE_ERROR_CLASS (DatabaseCreateError, DatabaseError)

Thrown when creating a database fails.

8.7.2.7 XAPIAN_DEFINE_ERROR_CLASS (DatabaseCorruptError, DatabaseError)

thrown if the database is corrupt.

8.7.2.8 XAPIAN_DEFINE_ERROR_CLASS (NetworkTimeoutError, NetworkError)

Thrown when a network timeout is exceeded.

8.7.2.9 XAPIAN_DEFINE_ERROR_CLASS (NetworkError, RuntimeError)

thrown when there is a communications problem with a remote database.

8.7.2.10 XAPIAN_DEFINE_ERROR_CLASS (FeatureUnavailableError, RuntimeError)

Thrown if a feature is unavailable - usually due to not being compiled in.

8.7.2.11 XAPIAN_DEFINE_ERROR_CLASS (DatabaseError, RuntimeError)

thrown for miscellaneous database errors.

8.7.2.12 XAPIAN_DEFINE_ERROR_CLASS (InternalError, RuntimeError)

thrown when really weird stuff happens.

If this is thrown something has gone badly wrong.

8.7.2.13 XAPIAN_DEFINE_ERROR_CLASS (RangeError, RuntimeError)

thrown when an element is out of range.

8.7.2.14 XAPIAN_DEFINE_ERROR_CLASS (DocNotFoundError, RuntimeError)

Thrown when an attempt is made to access a document which is not in the database.

This could occur either due to a programming error, or because the database has changed since running the query.

8.7.2.15 XAPIAN_DEFINE_ERROR_CLASS (InvalidOperationError, LogicError)

Thrown when API calls are made in an invalid way.

8.7.2.16 XAPIAN_DEFINE_ERROR_CLASS (InvalidArgumentError, LogicError)

Thrown when an invalid argument is supplied to the API.

8.7.2.17 XAPIAN_DEFINE_ERROR_CLASS (UnimplementedError, LogicError)

Thrown when an attempt to use an unimplemented feature is made.

8.7.2.18 XAPIAN_DEFINE_ERROR_CLASS (AssertionError, LogicError)

Thrown if an internal consistency check fails.

This represents a bug in Xapian.

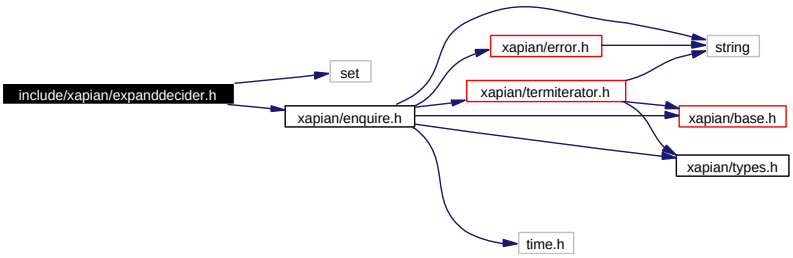
8.8 include/xapian/expanddecider.h File Reference

Classes for filtering which terms returned by expand.

`#include <set>`

`#include <xapian/enquire.h>`

Include dependency graph for expanddecider.h:



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

Namespaces

- namespace Xapian

Classes

- class Xapian::ExpandDeciderFilterTerms

  One useful expand decision functor, which provides a way of filtering out a fixed list of terms from the expand set.

- class Xapian::ExpandDeciderAnd

  An expand decision functor which can be used to join two functors with an AND operation.

8.8.1 Detailed Description

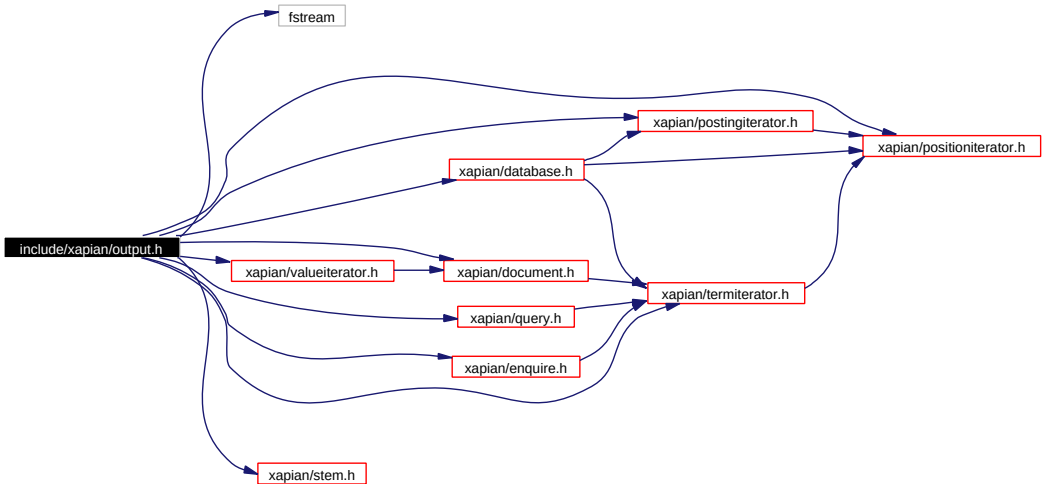
Classes for filtering which terms returned by expand.

8.9 include/xapian/output.h File Reference

Functions for output of strings describing Xapian objects.

```
#include <fstream>
#include <xapian/database.h>
#include <xapian/document.h>
#include <xapian/query.h>
#include <xapian/enquire.h>
#include <xapian/stem.h>
#include <xapian/postingiterator.h>
#include <xapian/positioniterator.h>
#include <xapian/termiterator.h>
#include <xapian/valueiterator.h>
```

Include dependency graph for output.h:



Defines

- #define XAPIAN_OUTPUT_FUNCTION(CLASS)

  For internal use only.

  Helper macro for defining stream output of Xapian class.

8.9.1 Detailed Description

Functions for output of strings describing Xapian objects.

8.9.2 Define Documentation

8.9.2.1 #define XAPIAN_OUTPUT_FUNCTION(CLASS)

Value:

```
inline std::ostream & \
operator<<(std::ostream & os, const CLASS & object) { \
    return os << object.get_description(); \
}
```

For internal use only.

Helper macro for defining stream output of Xapian class.

8.10 include/xapian/positioniterator.h File Reference

Classes for iterating through position lists.

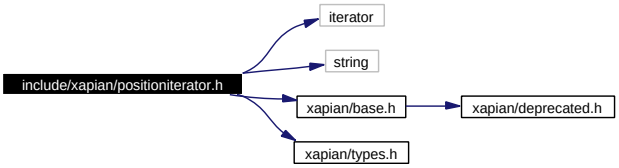
`#include <iterator>`

`#include <string>`

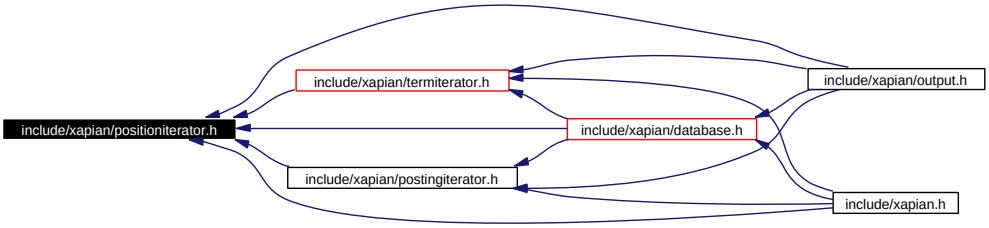
`#include <xapian/base.h>`

`#include <xapian/types.h>`

Include dependency graph for positioniterator.h:



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



Namespaces

- namespace Xapian

Classes

- class Xapian::TermPosWrapper

  A wrapper class for a termpos which returns the termpos if dereferenced with ∗.

- 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.

8.10.1 Detailed Description

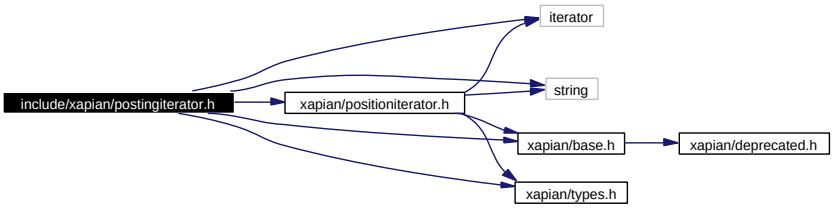
Classes for iterating through position lists.

8.11 include/xapian/postingiterator.h File Reference

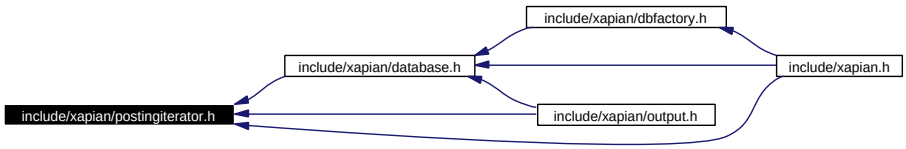
Classes for iterating through posting lists.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
```

Include dependency graph for postingiterator.h:



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



Namespaces

- namespace Xapian

Classes

- class Xapian::DocIDWrapper

  A wrapper class for a docid which returns the docid if dereferenced with ∗.

- 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.

8.11.1 Detailed Description

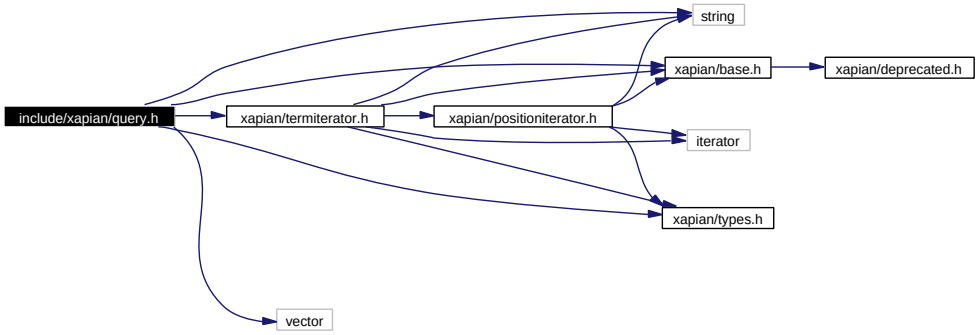
Classes for iterating through posting lists.

8.12 include/xapian/query.h File Reference

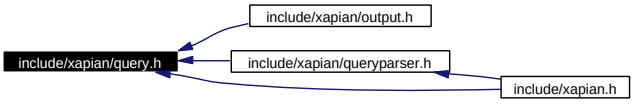
Classes for representing a query.

```
#include <string>
#include <vector>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
```

Include dependency graph for query.h:



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



Namespaces

- namespace Xapian

Classes

- class Xapian::Query

  Class representing a query.

- class Xapian::Query::Internal

Internal class, implementing most of Xapian::Query.

Functions

- std::string **Xapian::serialise_qint_** (const Xapian::Query::Internal *qint, Xapian::termpos &curpos)

8.12.1 Detailed Description

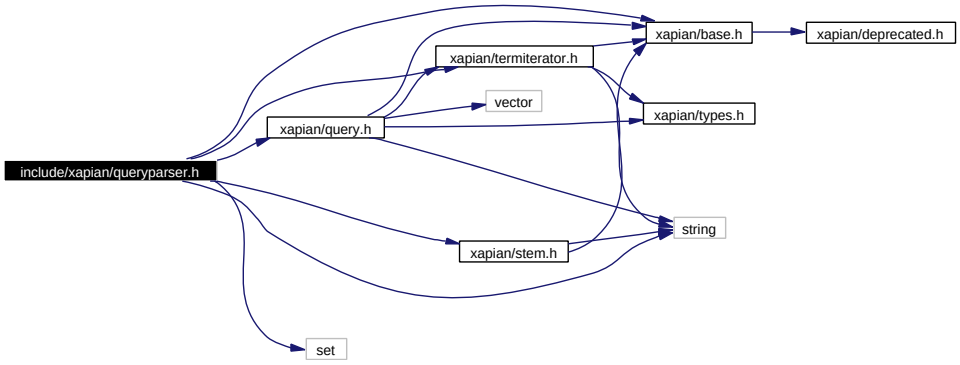
Classes for representing a query.

8.13 include/xapian/queryparser.h File Reference

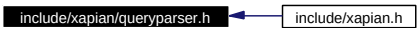
parsing a user query string to build a Xapian::Query object

```
#include <xapian/base.h>
#include <xapian/query.h>
#include <xapian/stem.h>
#include <xapian/termiterator.h>
#include <set>
#include <string>
```

Include dependency graph for queryparser.h:



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



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.

- class Xapian::QueryParser

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

8.13.1 Detailed Description

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

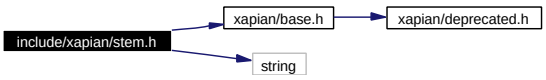
8.14 include/xapian/stem.h File Reference

stemming algorithms

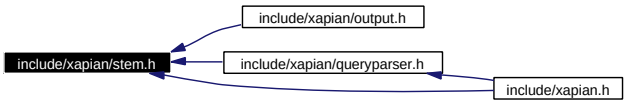
`#include <xapian/base.h>`

`#include <string>`

Include dependency graph for stem.h:



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



Namespaces

- namespace Xapian

Classes

- class Xapian::Stem

  Class representing a stemming algorithm.

8.14.1 Detailed Description

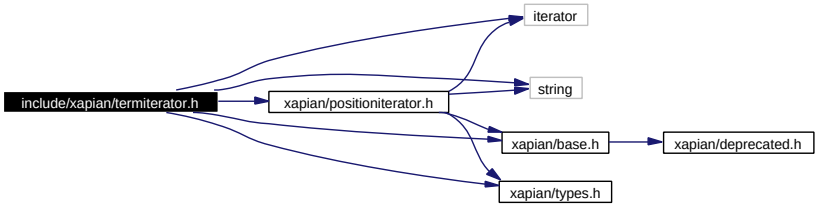
stemming algorithms

8.15 include/xapian/termiterator.h File Reference

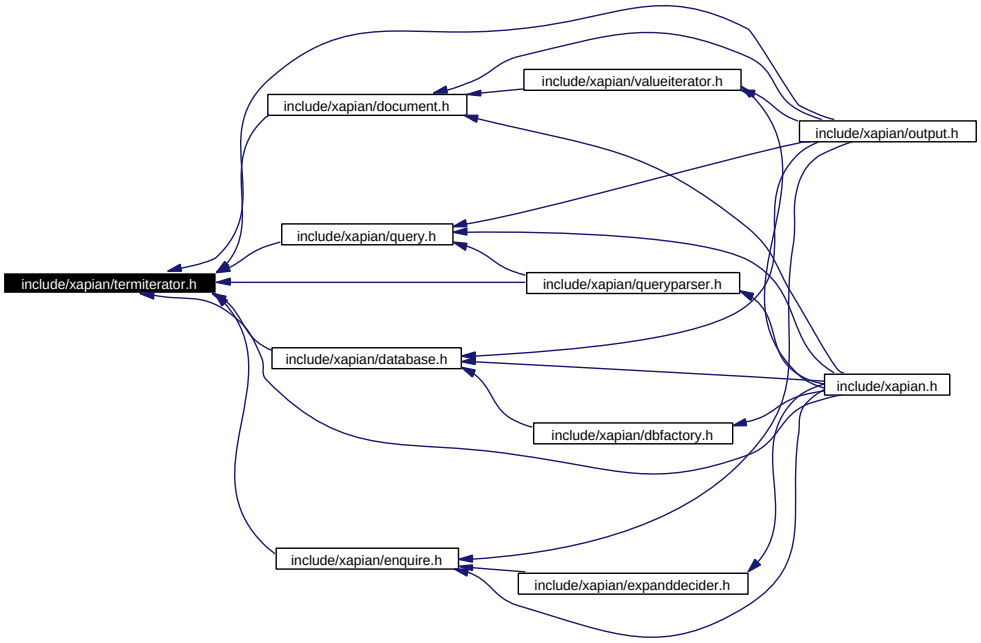
Classes for iterating through term lists.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
```

Include dependency graph for termiterator.h:



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



Namespaces

- namespace Xapian

Classes

- class Xapian::TermNameWrapper

  A wrapper class for a termname which returns the termname if dereferenced with ∗.

- class Xapian::TermIterator

  An iterator pointing to items in a list of terms.

Functions

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

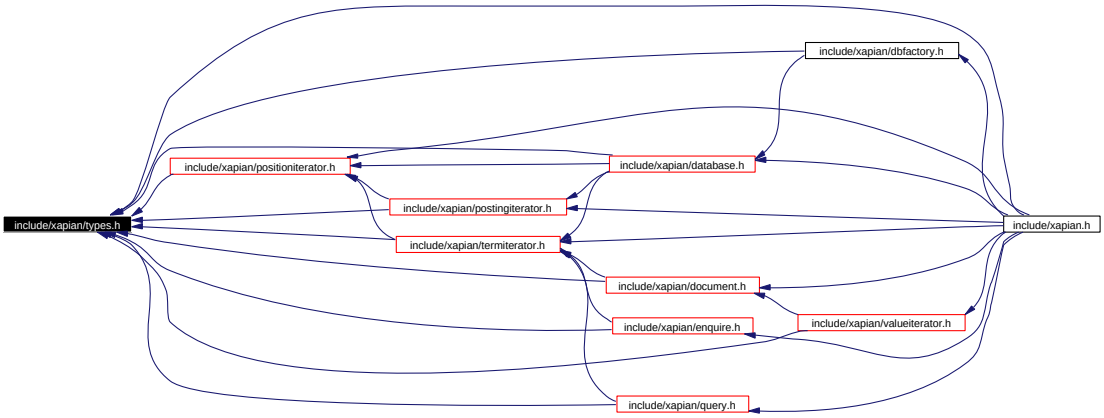
8.15.1 Detailed Description

Classes for iterating through term lists.

8.16 include/xapian/types.h File Reference

Common types used.

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



Typedefs

- typedef unsigned int docid

  A unique id for a document.

- typedef docid doccount

  Type for counts of documents.

- typedef int doccount_diff

  Type for signed difference between counts of documents.

- typedef unsigned int termcount

  Type for counts of terms (eg, wdf, wqf).

- typedef int termcount_diff

  Type for signed difference between counts of terms.

- typedef unsigned int termpos

  Type for term positions within documents.

- typedef int termpos_diff

  Type for signed difference between term positions.

- typedef double doclength

  Type for (normalised) lengths of documents.

- typedef unsigned int valueno

  Type for referring to the number of a value in document.

- typedef int valueno_diff

  Type for signed difference between two valueno-s.

- typedef double weight

  A calculated weight, for a term or document.

- typedef int percent

  A percentage weight, for a term or document.

- typedef unsigned int timeout

  Type for specifying a timeout.

8.16.1 Detailed Description

Common types used.

8.16.2 Typedef Documentation

8.16.2.1 typedef docid doccount

Type for counts of documents.

8.16.2.2 typedef int doccount_diff

Type for signed difference between counts of documents.

8.16.2.3 typedef unsigned int docid

A unique id for a document.

Document ids start at 1. A zero docid isn't valid, and may be used to indicate "no document".

8.16.2.4 typedef double doclength

Type for (normalised) lengths of documents.

8.16.2.5 typedef int percent

A percentage weight, for a term or document.

8.16.2.6 typedef unsigned int termcount

Type for counts of terms (eg, wdf, wqf).

8.16.2.7 typedef int termcount_diff

Type for signed difference between counts of terms.

8.16.2.8 typedef unsigned int termpos

Type for term positions within documents.

These start at 1. A value of 0 means that the positional information is not available for that term.

8.16.2.9 typedef int termpos_diff

Type for signed difference between term positions.

8.16.2.10 typedef unsigned int timeout

Type for specifying a timeout.

This refers to a time in microseconds: ie. a timeout value of 1000000 corresponds to a timeout of 1 second.

8.16.2.11 typedef unsigned int valueno

Type for referring to the number of a value in document.

8.16.2.12 typedef int valueno_diff

Type for signed difference between two valueno-s.

8.16.2.13 typedef double weight

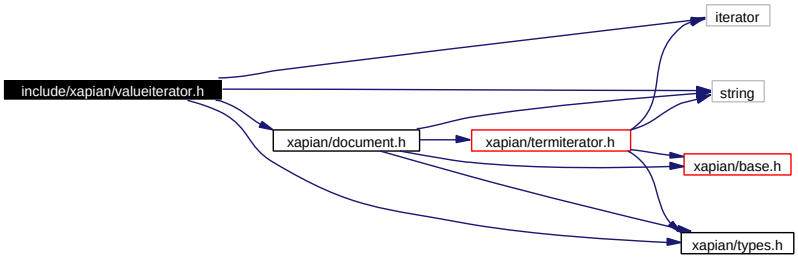
A calculated weight, for a term or document.

8.17 include/xapian/valueiterator.h File Reference

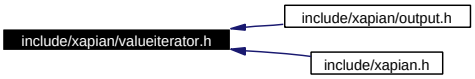
classes for iterating through values

```
#include <iterator>
#include <string>
#include <xapian/types.h>
#include <xapian/document.h>
```

Include dependency graph for valueiterator.h:



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



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

8.17.1 Detailed Description

classes for iterating through values

Chapter 9

xapian-core Page Documentation

9.1 Deprecated List

Member Xapian::Document::XAPIAN_DEPRECATED(void add_term_nopos(const std::string &term, Xapian::term
This method is deprecated and present only for backward compatibility. Use add_term() instead.

Member Xapian::Enquire::XAPIAN_DEPRECATED(void set_sorting(Xapian::valueno sort_key, int sort_bands, boo
This method is now deprecated, use set_sort_by_relevance(), set_sort_by_-value(), or set_sort_by_value_then_relevance() instead.

Member Xapian::Enquire::XAPIAN_DEPRECATED(void set_sort_forward(bool sort_forward))
This method is now deprecated, use set_docid_order() instead - set_sort_-forward(true) -> set_docid_order(ASCENDING) and set_sort_forward(false) -> set_docid_order(DESCENDING).

Member Xapian::Query::XAPIAN_DEPRECATED(bool is_empty() const)
Deprecated alias for empty()

Member Xapian::QueryParser::XAPIAN_DEPRECATED(void set_stemming_options(const std::string &lang, bool
Deprecated method for backward compatibility.

Index