Kitlist

0.6.6

Generated by Doxygen 1.5.6

Mon Nov 16 18:30:30 2009

Contents

Chapter 1

Kitlist Documentation

1.1 Introduction

The kitlist program has been developed to be run on multiple platforms. It is known to run on the following platforms:

- Debian 5.0 (Lenny)
- Maemo (Nokia Tablet)
- Windows XP

It can be compiled to use either a PostgreSQL database or XML documents as the data store (Debian Linux only). When compiled to use PostgreSQL, the program can be used from the command line without a GUI.

Where the program is executed without any arguments, the GUI is shown. If there are any arguments, the command line version is executed unless the '-g' option is specified to force running the GUI.

1.2 Additional Documentation

See the README in the root of the source code for information on building the application.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 anonymous_namespace{kitlistgui.cpp} Namespace Reference

Variables

- const string GLADE_APP_FILE = "kitlist.glade"

  Resource file name.

- const guint SB_ITEM_COUNT = 1000

  Status bar message constant for displaying item counts.

- const guint SB_SAVE = SB_ITEM_COUNT + 1

  Status bar message constant for save notifications.

- const guint SB_MSG = SB_SAVE + 1

  Status bara message constant for general messages.

- const char item_target_custom [ ] = "kitlistclipboard"

  Key used for custom clipboard.

- const char item_target_text [ ] = "text/plain"

  Mime type for clipboard content.

- const char XML_ELEMENT_ID [ ] = "id"

  Tag name for the ID element in the clipbard XML document.

- const Glib::ustring DEFAULT_FILENAME_EXTENSION = ".kit"

  Default filename extension.

- const Glib::ustring DEFAULT_FILENAME = "kitlist" + DEFAULT_FILENAME_EXTENSION

  The default filename.

- const Glib::ustring GCONF_KEY = "/apps/kitlist"

  The application's root key in the GConf hierarchy.

- const Glib::ustring GCONF_KEY_CURRENT_FILENAME = GCONF_KEY + "/current_-filename"

  GConf entry for the current filename.

- const gint DEFAULT_MAX_RECENT_FILES = 4

  The maximum number of recent files to maintain.

- const Glib::ustring GCONF_KEY_RECENT_FILES = GCONF_KEY + "/recent_files"

  GConf entry for recent files.

- const Glib::ustring GCONF_KEY_MAX_RECENT_FILES = GCONF_KEY + "/max_recent_-files"

  GConf entry for max recent files.

6.1.1 Variable Documentation

6.1.1.1 const Glib::ustring anonymous_namespace{kitlistgui.cpp}::DEFAULT_FILENAME = "kitlist" + DEFAULT_FILENAME_EXTENSION

The default filename.

Definition at line 87 of file kitlistgui.cpp.

Referenced by KitListGui::init().

6.1.1.2 const Glib::ustring anonymous_namespace{kitlistgui.cpp}::DEFAULT_FILENAME_-EXTENSION = ".kit"

Default filename extension.

Definition at line 84 of file kitlistgui.cpp.

Referenced by KitListGui::choose_filename(), and KitListGui::on_menu_file_open().

6.1.1.3 const gint anonymous_namespace{kitlistgui.cpp}::DEFAULT_MAX_RECENT_FILES = 4

The maximum number of recent files to maintain.

Definition at line 100 of file kitlistgui.cpp.

Referenced by KitListGui::get_max_recent_files().

6.1.1.4 const Glib::ustring anonymous_namespace{kitlistgui.cpp}::GCONF_KEY = "/apps/kitlist"

The application's root key in the GConf hierarchy.

Definition at line 93 of file kitlistgui.cpp.

Referenced by KitListGui::init().

6.1.1.5 const Glib::ustring anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_CURRENT_FILENAME = GCONF_KEY + "/current_filename"

GConf entry for the current filename.

Definition at line 97 of file kitlistgui.cpp.

Referenced by KitListGui::init(), KitListGui::on_menu_file_new(), KitListGui::on_menu_recent_file(), KitListGui::on_menu_save(), KitListGui::on_menu_save_as(), and KitListGui::open_file().

6.1.1.6 const Glib::ustring anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_MAX_RECENT_FILES = GCONF_KEY + "/max_recent_files"

GConf entry for max recent files.

Definition at line 106 of file kitlistgui.cpp.

Referenced by KitListGui::get_max_recent_files().

6.1.1.7 const Glib::ustring anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_RECENT_FILES = GCONF_KEY + "/recent_files"

GConf entry for recent files.

Definition at line 103 of file kitlistgui.cpp.

Referenced by KitListGui::update_recent_files(), and KitListGui::update_recent_files_menu().

6.1.1.8 const string anonymous_namespace{kitlistgui.cpp}::GLADE_APP_FILE = "kitlist.glade"

Resource file name.

Definition at line 66 of file kitlistgui.cpp.

Referenced by KitListGui::init().

6.1.1.9 const char anonymous_namespace{kitlistgui.cpp}::item_target_custom[ ] = "kitlistclipboard"

Key used for custom clipboard.

Definition at line 76 of file kitlistgui.cpp.

Referenced by KitListGui::copy_selected_items_to_clipboard(), KitListGui::on_clipboard_get(), KitListGui::on_clipboard_received(), and KitListGui::paste_status_received().

6.1.1.10 const char anonymous_namespace{kitlistgui.cpp}::item_target_text[ ] = "text/plain"

Mime type for clipboard content.

Definition at line 78 of file kitlistgui.cpp.

Referenced by KitListGui::copy_selected_items_to_clipboard(), KitListGui::on_clipboard_get(), KitListGui::on_clipboard_received(), KitListGui::on_menu_paste(), and KitListGui::paste_status_received().

6.1.1.11 const guint anonymous_namespace{kitlistgui.cpp}::SB_ITEM_COUNT = 1000

Status bar message constant for displaying item counts.

Definition at line 69 of file kitlistgui.cpp.

Referenced by KitListGui::update_item_count().

6.1.1.12 const guint anonymous_namespace{kitlistgui.cpp}::SB_MSG = SB_SAVE + 1

Status bara message constant for general messages.

Definition at line 73 of file kitlistgui.cpp.

Referenced by KitListGui::on_menu_cut(), KitListGui::on_menu_delete_category(), and KitListGui::on_-menu_rename_category().

6.1.1.13 const guint anonymous_namespace{kitlistgui.cpp}::SB_SAVE = SB_ITEM_COUNT + 1

Status bar message constant for save notifications.

Definition at line 71 of file kitlistgui.cpp.

Referenced by KitListGui::on_menu_file_new(), KitListGui::on_menu_file_open(), KitListGui::on_-menu_save(), KitListGui::on_menu_save_as(), and KitListGui::safe_open_file().

6.1.1.14 const char anonymous_namespace{kitlistgui.cpp}::XML_ELEMENT_ID[ ] = "id"

Tag name for the ID element in the clipbard XML document.

Definition at line 81 of file kitlistgui.cpp.

Referenced by KitListGui::copy_selected_items_to_clipboard(), and KitListGui::paste_from_xml().

Chapter 7

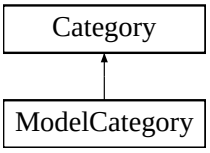
Class Documentation

7.1 Category Class Reference

Represents a Category.

`#include <category.hpp>`

Inheritance diagram for Category::



Public Member Functions

- ~Category ()
- void set_id (long id)
- long get_id ()
- void set_name (const std::string name)
- std::string get_name ()
- virtual void add_item (Item *item)

  Associates the passed item with this Category.

- virtual void remove_item (Item *item)

  Removes the association of the passed item from this Category.

- virtual size_t item_count ()

  Returns the number of items associated with this category.

- virtual bool has_items ()

  Returns true if there are any items associated with this category.

- void foreach_item (const SlotForeachItem &slot)

  Executes a callback function for each associated item.

- void execute (ItemFunctor &functor)

  Executes the passed ItemFunctor.

Protected Attributes

- long m_id

  Unique id.

- std::string m_name

  The category name.

- ItemContainer m_items

  List of associated items.

Friends

- class CategoryCompareName
- class CategoryCompareId
- class KitModel

7.1.1 Detailed Description

Represents a Category.

Many categories may have one or more items.

Definition at line 37 of file category.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Category::~Category () [inline]

Definition at line 43 of file category.hpp.

7.1.3 Member Function Documentation

7.1.3.1 void Category::set_id (long *id*) [inline]

Definition at line 44 of file category.hpp.

References m_id.

Referenced by Service::create_category(), and KitParser::process_category().

7.1.3.2 long Category::get_id () [inline]

Definition at line 45 of file category.hpp.

References m_id.

Referenced by KitModel::add_category(), XmlDao::add_category_to_dom(), KitListGui::close_add_category_window(), XmlDao::get_model(), KitListGui::on_menu_create_category(), KitListGui::on_menu_rename_category(), KitParser::process_category_item(), and KitListGui::refresh_category_list().

7.1.3.3 void Category::set_name (const std::string *name*) [inline]

Definition at line 46 of file category.hpp.

References m_name.

Referenced by KitListGui::close_add_category_window(), KitParser::on_end_element(), KitListGui::on_menu_create_category(), and KitListGui::on_menu_rename_category().

7.1.3.4 std::string Category::get_name () [inline]

Definition at line 47 of file category.hpp.

References m_name.

Referenced by XmlDao::add_category_to_dom(), KitListGui::on_menu_rename_category(), and KitListGui::refresh_category_list().

7.1.3.5 void Category::add_item (Item * *item*) [virtual]

Associates the passed item with this Category.

Reimplemented in ModelCategory.

Definition at line 29 of file category.cpp.

References m_items.

Referenced by ModelCategory::add_item().

7.1.3.6 void Category::remove_item (Item * *item*) [virtual]

Removes the association of the passed item from this Category.

The passed item is not deleted.

Parameters:

item the item to un-associate.

Reimplemented in ModelCategory.

Definition at line 40 of file category.cpp.

References m_items.

Referenced by ModelCategory::remove_item().

7.1.3.7 virtual size_t Category::item_count () [inline, virtual]

Returns the number of items associated with this category.

Definition at line 51 of file category.hpp.

References m_items.

Referenced by KitList::list_items().

7.1.3.8 virtual bool Category::has_items () [inline, virtual]

Returns true if there are any items associated with this category.

Definition at line 53 of file category.hpp.

References m_items.

Referenced by KitList::list_items().

7.1.3.9 void Category::foreach_item (const SlotForeachItem & *slot*)

Executes a callback function for each associated item.

The callback function will be passed a reference to the current item being iterated.

Parameters:

slot the callback function.

Definition at line 55 of file category.cpp.

References m_items.

Referenced by XmlDao::add_category_to_dom(), and KitList::list_items().

7.1.3.10 void Category::execute (ItemFunctor & *functor*)

Executes the passed ItemFunctor.

The ItemFunctor's override operator() method is called, passing a reference to the item being iterated over. If the called method returns true, the iteration stops and no more calls will be made to the functor.

Definition at line 71 of file category.cpp.

References m_items.

Referenced by KitList::tick_items().

7.1.4 Friends And Related Function Documentation

7.1.4.1 friend class CategoryCompareName [friend]

Definition at line 56 of file category.hpp.

7.1.4.2 friend class CategoryCompareId [friend]

Definition at line 57 of file category.hpp.

7.1.4.3 friend class KitModel [friend]

Reimplemented in ModelCategory.

Definition at line 58 of file category.hpp.

7.1.5 Member Data Documentation

7.1.5.1 long Category::m_id [protected]

Unique id.

Definition at line 39 of file category.hpp.

Referenced by get_id(), CategoryCompareId::operator()(), and set_id().

7.1.5.2 std::string Category::m_name [protected]

The category name.

Definition at line 40 of file category.hpp.

Referenced by get_name(), CategoryCompareName::operator()(), and set_name().

7.1.5.3 ItemContainer Category::m_items [protected]

List of associated items.

Definition at line 41 of file category.hpp.

Referenced by add_item(), KitModel::copy_items(), execute(), foreach_item(), ModelCategory::get_items(), ModelCategory::get_model_items(), has_items(), item_count(), ModelCategory::purge(), and remove_item().

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

- category.hpp
- category.cpp

7.2 CategoryCompareId Class Reference

Comparator used for comparing Categories by id.

`#include <category.hpp>`

Public Member Functions

- CategoryCompareId ()
- int operator() (Category *c1, Category *c2)

7.2.1 Detailed Description

Comparator used for comparing Categories by id.

See also:

Category

Definition at line 77 of file category.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 CategoryCompareId::CategoryCompareId () [inline]

Definition at line 79 of file category.hpp.

7.2.3 Member Function Documentation

7.2.3.1 int CategoryCompareId::operator() (Category * *c1*, Category * *c2*) [inline]

Definition at line 80 of file category.hpp.

References Category::m_id.

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

- category.hpp

7.3 CategoryCompareName Class Reference

Comparator used for sorting Categories by name.

`#include <category.hpp>`

Public Member Functions

- CategoryCompareName ()
- int operator() (Category *c1, Category *c2)

7.3.1 Detailed Description

Comparator used for sorting Categories by name.

See also:

Category

Definition at line 66 of file category.hpp.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 CategoryCompareName::CategoryCompareName () [inline]

Definition at line 68 of file category.hpp.

7.3.3 Member Function Documentation

7.3.3.1 int CategoryCompareName::operator() (Category * *c1*, Category * *c2*) [inline]

Definition at line 69 of file category.hpp.

References Category::m_name.

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

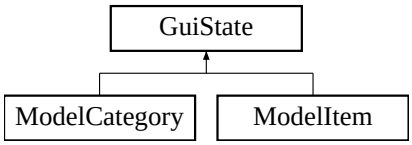
- category.hpp

7.4 GuiState Class Reference

Class encapsulating state of an object in the data model.

`#include <kitmodel.hpp>`

Inheritance diagram for GuiState::



Public Member Functions

- GuiState ()
- bool is_dirty ()
- void set_dirty (bool dirty=true)
- bool is_deleted ()
- void set_deleted (bool deleted)
- void set_new_flag (bool flag)
- bool is_new ()
- virtual void reset ()

  resets the state of each flag to it's default.

Protected Attributes

- bool m_dirty
- bool m_deleted
- bool m_new

7.4.1 Detailed Description

Class encapsulating state of an object in the data model.

Provides additional flags to identityf whether the object is dirty, has been deleted or is new.

Definition at line 38 of file kitmodel.hpp.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 GuiState::GuiState () `[inline]`

Definition at line 44 of file kitmodel.hpp.

7.4.3 Member Function Documentation

7.4.3.1 bool GuiState::is_dirty () [inline]

Definition at line 45 of file kitmodel.hpp.

References m_dirty.

7.4.3.2 void GuiState::set_dirty (bool *dirty* = true) [inline]

Definition at line 46 of file kitmodel.hpp.

References m_dirty.

Referenced by KitModel::copy_items(), Service::create_category(), Service::create_item(), Service::delete_category(), Service::delete_item(), KitListGui::on_menu_cut(), ModelItem::set_checked(), and Service::update_item().

7.4.3.3 bool GuiState::is_deleted () [inline]

Definition at line 47 of file kitmodel.hpp.

References m_deleted.

Referenced by XmlDao::add_category_to_dom(), XmlDao::add_item_to_dom(), KitModel::get_categories(), ModelCategory::get_items(), ModelCategory::get_model_items(), KitModel::purge(), KitListGui::refresh_category_list(), and KitModel::reset().

7.4.3.4 void GuiState::set_deleted (bool *deleted*) [inline]

Definition at line 48 of file kitmodel.hpp.

References m_deleted.

Referenced by Service::delete_category(), and Service::delete_item().

7.4.3.5 void GuiState::set_new_flag (bool *flag*) [inline]

Definition at line 49 of file kitmodel.hpp.

References m_new.

Referenced by Service::create_category(), and Service::create_item().

7.4.3.6 bool GuiState::is_new () [inline]

Definition at line 50 of file kitmodel.hpp.

References m_new.

7.4.3.7 void GuiState::reset () [virtual]

resets the state of each flag to it's default.

The dirty, deleted and new flags are set to false.

Reimplemented in ModelCategory.

Definition at line 36 of file kitmodel.cpp.

References m_deleted, m_dirty, and m_new.

Referenced by KitModel::reset(), and ModelCategory::reset().

7.4.4 Member Data Documentation

7.4.4.1 bool GuiState::m_dirty [protected]

Definition at line 40 of file kitmodel.hpp.

Referenced by is_dirty(), reset(), and set_dirty().

7.4.4.2 bool GuiState::m_deleted [protected]

Definition at line 41 of file kitmodel.hpp.

Referenced by is_deleted(), reset(), and set_deleted().

7.4.4.3 bool GuiState::m_new [protected]

Definition at line 42 of file kitmodel.hpp.

Referenced by is_new(), reset(), and set_new_flag().

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

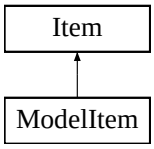
- kitmodel.hpp
- kitmodel.cpp

7.5 Item Class Reference

Represents an Item.

`#include <item.hpp>`

Inheritance diagram for Item::



Public Member Functions

- Item ()
- Item (const Item &i)

  Creates a copy of this item based on the passed item.

- void set_id (long id)
- long get_id ()
- void set_description (const std::string description)
- std::string get_description ()
- virtual void set_checked (bool checked)
- bool get_checked ()

Private Attributes

- long m_id

  Unique ID.

- std::string m_desc

  The item's description.

- bool m_checked

  Whether checked/ticked or not.

Friends

- class ItemCompareName
- class ItemCompareId
- std::ostream & operator<< (std::ostream &os, const Item &i)

7.5.1 Detailed Description

Represents an Item.

Definition at line 37 of file item.hpp.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Item::Item () [inline]

Definition at line 42 of file item.hpp.

7.5.2.2 Item::Item (const Item & *i*) [inline]

Creates a copy of this item based on the passed item.

Definition at line 44 of file item.hpp.

7.5.3 Member Function Documentation

7.5.3.1 void Item::set_id (long *id*) [inline]

Definition at line 45 of file item.hpp.

References m_id.

Referenced by Service::create_item(), and KitParser::process_item().

7.5.3.2 long Item::get_id () [inline]

Definition at line 46 of file item.hpp.

References m_id.

Referenced by XmlDao::add_category_item_to_dom(), KitModel::add_item(), ModelCategory::add_item(), XmlDao::add_item_to_dom(), XmlDao::get_model(), KitList::list_item(), TickItem::operator()(), ModelItemCompareId::operator()(), and ModelCategory::remove_item().

7.5.3.3 void Item::set_description (const std::string *description*) [inline]

Definition at line 47 of file item.hpp.

References m_desc.

Referenced by KitListGui::close_add_item_window(), KitParser::on_end_element(), KitListGui::on_menu_add(), and Service::update_item().

7.5.3.4 std::string Item::get_description () [inline]

Definition at line 48 of file item.hpp.

References m_desc.

Referenced by XmlDao::add_item_to_dom(), KitList::list_item(), and TickItem::operator()().

7.5.3.5 virtual void Item::set_checked (bool *checked*) [inline, virtual]

Reimplemented in ModelItem.

Definition at line 49 of file item.hpp.

References m_checked.

Referenced by KitListGui::close_add_item_window(), KitListGui::on_menu_add(), TickItem::operator()(), and ModelItem::set_checked().

7.5.3.6 bool Item::get_checked () [inline]

Definition at line 50 of file item.hpp.

References m_checked.

Referenced by XmlDao::add_item_to_dom(), and TickItem::operator()().

7.5.4 Friends And Related Function Documentation

7.5.4.1 friend class ItemCompareName [friend]

Definition at line 51 of file item.hpp.

7.5.4.2 friend class ItemCompareId [friend]

Definition at line 52 of file item.hpp.

7.5.4.3 std::ostream& operator<< (std::ostream & *os*, const Item & *i*) [friend]

Definition at line 53 of file item.hpp.

7.5.5 Member Data Documentation

7.5.5.1 long Item::m_id [private]

Unique ID.

Definition at line 38 of file item.hpp.

Referenced by get_id(), ItemCompareId::operator()(), and set_id().

7.5.5.2 std::string Item::m_desc [private]

The item's description.

Definition at line 39 of file item.hpp.

Referenced by get_description(), ItemCompareName::operator()(), and set_description().

7.5.5.3 bool Item::m_checked [private]

Whether checked/ticked or not.

Definition at line 40 of file item.hpp.

Referenced by get_checked(), and set_checked().

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

- item.hpp

7.6 ItemCompareId Class Reference

Comparator used for comparing Items by id.

`#include <item.hpp>`

Public Member Functions

- ItemCompareId ()
- int operator() (Item *i1, Item *i2)

7.6.1 Detailed Description

Comparator used for comparing Items by id.

See also:

Item

Definition at line 72 of file item.hpp.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 ItemCompareId::ItemCompareId () [inline]

Definition at line 74 of file item.hpp.

7.6.3 Member Function Documentation

7.6.3.1 int ItemCompareId::operator() (Item * *i1*, Item * *i2*) [inline]

Definition at line 75 of file item.hpp.

References Item::m_id.

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

- item.hpp

7.7 ItemCompareName Class Reference

Comparator used for sorting Items by name.

`#include <item.hpp>`

Public Member Functions

- ItemCompareName ()
- int operator() (Item *i1, Item *i2)

7.7.1 Detailed Description

Comparator used for sorting Items by name.

See also:

Item

Definition at line 61 of file item.hpp.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ItemCompareName::ItemCompareName () [inline]

Definition at line 63 of file item.hpp.

7.7.3 Member Function Documentation

7.7.3.1 int ItemCompareName::operator() (Item * *i1*, Item * *i2*) [inline]

Definition at line 64 of file item.hpp.

References Item::m_desc.

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

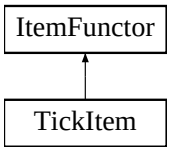
- item.hpp

7.8 ItemFunctor Class Reference

Functor for processing items.

`#include <item.hpp>`

Inheritance diagram for ItemFunctor::



Public Member Functions

- virtual bool operator() (Item &item)=0

7.8.1 Detailed Description

Functor for processing items.

Intended to have the operator() overriden by an actual implementation.

Definition at line 85 of file item.hpp.

7.8.2 Member Function Documentation

7.8.2.1 virtual bool ItemFunctor::operator() (Item & *item*) `[pure virtual]`

Implemented in TickItem.

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

- item.hpp

7.9 KitList Class Reference

Main application class.

Public Member Functions

- KitList (const string dbname, const string user, const string port, bool verbose=false)
  Constructor specifying Postgresql database connection parameters.
- ~KitList ()
- KitListDao * get_dao ()
- void add_item (const string name)
- void add_item (const string name, long cat_id)
- void append_items_to_category (long from_cat_id, long to_cat_id, item_choice choice)
  Copies items from one category to another.
- void associate_item_with_category (long id, long cat_id)
  Associates an existing item with an existing category.
- void list_items_start (bool empty_list)
  Called before writing a list of items to STDOUT.
- void list_item (Item &item)
  Outputs details of the passed item to STDOUT.
- void list_items_end (bool empty_list, size_t count)
  Called after writing a list of items to STDOUT.
- void list_items (Category &c)
- bool on_list_item (Item &item)
- void list_items (ItemContainer &items)
- void list_items (long cat_id, item_choice choice)
- void execute (ItemContainer &items, ItemFunctor &functor)
- void tick_items (Category &c)
- void tick_items (ItemContainer &items)
- void tick_items (long cat_id, item_choice choice)
  Checks/ticks all items belonging to a category.
- void list_categories ()
  Lists details of all categories to STDOUT.
- void new_category (const string name)
- void delete_category (long id)
- void delete_item (long id)
- void remove_item_from_category (long id, long cat_id)
- void set_item_flag (long id)
- void unset_item_flag (long id)
- void set_category_flag (long id)
- void unset_category_flag (long id)
- void set_all_flags ()
- void unset_all_flags ()

Protected Attributes

- KitListDao * m_dao

  Reference to an implementation DAO class.

7.9.1 Detailed Description

Main application class.

Primarily performs application startup and command line parsing. Allows the application to be run either from the command line or as an interactive GUI application.

Definition at line 83 of file main.cpp.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 KitList::KitList (const string *dbname*, const string *user*, const string *port*, bool *verbose* = `false`)

Constructor specifying Postgresql database connection parameters.

Note that the PostgreSQL libraries will use default values for these parameters if they are not specified, including examining environment variables. See the PostgreSQL documentation for full details.

Parameters:

dbname The name of the database.

user The database user name to connect with.

port The database port to connect to.

verbose Optional parameter indicating whether to include verbose output to STDOUT. Defaults to false.

Definition at line 167 of file main.cpp.

References m_dao.

7.9.2.2 KitList::~KitList ()

Definition at line 176 of file main.cpp.

References m_dao.

7.9.3 Member Function Documentation

7.9.3.1 KitListDao* KitList::get_dao () `[inline]`

Definition at line 89 of file main.cpp.

References m_dao.

Referenced by main().

7.9.3.2 void KitList::add_item (const string *name*)

Referenced by main().

7.9.3.3 void KitList::add_item (const string *name*, long *cat_id*)

Creates a new item with the specified name and optionally associates it with a category.

Parameters:

name The name of the item to create.

cat_id The ID of the existing category to associate the item with. If the ID is less than zero, the item will not be associated with a category. Otherwise, the category must already exist.

Definition at line 189 of file main.cpp.

References KitListDao::add_item(), and m_dao.

7.9.3.4 void KitList::append_items_to_category (long *from_cat_id*, long *to_cat_id*, item_choice *choice*)

Copies items from one category to another.

Optionally, only checked or unchecked items are copied.

Parameters:

from_cat_id The ID of the source category.

to_cat_id The ID of the target category.

choice One of ALL_ITEMS, CHECKED_ITEMS or UNCHECKED_ITEMS.

Definition at line 208 of file main.cpp.

References KitListDao::append_items_to_category(), and m_dao.

Referenced by main().

7.9.3.5 void KitList::associate_item_with_category (long *id*, long *cat_id*)

Associates an existing item with an existing category.

Parameters:

id The Item ID

cat_id The Category ID

Definition at line 219 of file main.cpp.

References KitListDao::associate_item_with_category(), and m_dao.

Referenced by main().

7.9.3.6 void KitList::list_items_start (bool *empty_list*)

Called before writing a list of items to STDOUT.

Fundamentally outputs headings, in either HTML or text format, depending on the value of html_flag. The html_flag is set by the command line option, '–html'.

Parameters:

empty_list Set to true if the list is empty. If so, table headings will not be output.

Definition at line 243 of file main.cpp.

References html_flag.

Referenced by list_items().

7.9.3.7 void KitList::list_item (Item & *item*)

Outputs details of the passed item to STDOUT.

Details are wrapped in HTML format if the html_flag has been set. The html_flag is set by the command line option, '–html'.

Definition at line 297 of file main.cpp.

References Item::get_description(), Item::get_id(), and html_flag.

Referenced by list_items(), and on_list_item().

7.9.3.8 void KitList::list_items_end (bool *empty_list*, size_t *count*)

Called after writing a list of items to STDOUT.

Fundamentally outputs a footer, in either HTML or text format, depending on the value of html_flag. The html_flag is set by the command line option, '–html'.

Parameters:

empty_list Set to true if the list is empty. If so, a table footer will not be output.

Definition at line 274 of file main.cpp.

References html_flag.

Referenced by list_items().

7.9.3.9 void KitList::list_items (Category & *c*)

Lists details of all items in the passed Category to STDOUT.

Definition at line 313 of file main.cpp.

References Category::foreach_item(), Category::has_items(), Category::item_count(), list_items_end(), list_items_start(), and on_list_item().

Referenced by list_items(), and main().

7.9.3.10 bool KitList::on_list_item (Item & *item*)

A callback function to output details of the passed Item to STDOUT.

Definition at line 228 of file main.cpp.

References list_item().

Referenced by list_items().

7.9.3.11 void KitList::list_items (ItemContainer & *items*)

Lists details of all items in the passed ItemContainer to STDOUT.

Definition at line 323 of file main.cpp.

References list_item(), list_items_end(), and list_items_start().

7.9.3.12 void KitList::list_items (long *cat_id*, item_choice *choice* = ALL_ITEMS)

Lists details of items to STDOUT.

Parameters:

cat_id The ID Of the Category to list. If the value is less than zero, all items are listed.

choice Optional. One of ALL_ITEMS (default), CHECKED_ITEMS or UNCHECKED_ITEMS.

Definition at line 341 of file main.cpp.

References KitListDao::get_all_items(), KitListDao::get_category(), list_items(), and m_dao.

7.9.3.13 void KitList::execute (ItemContainer & *items*, ItemFunctor & *functor*)

Executes the passed ItemFunctor for each item in the ItemContainer.

See also:

ItemFunctor.

Definition at line 361 of file main.cpp.

Referenced by tick_items().

7.9.3.14 void KitList::tick_items (Category & *c*)

Checks/ticks all items in the passed Category.

Definition at line 383 of file main.cpp.

References Category::execute(), m_dao, and KitListDao::update_item_checked_state().

Referenced by main(), and tick_items().

7.9.3.15 void KitList::tick_items (ItemContainer & *items*)

Checks/ticks all items in the passed ItemContainer.

Definition at line 372 of file main.cpp.

References execute(), m_dao, and KitListDao::update_item_checked_state().

7.9.3.16 void KitList::tick_items (long *cat_id*, item_choice *choice* = ALL_ITEMS)

Checks/ticks all items belonging to a category.

Acts on all items if the cat_id is less than zero. Optionally operates on checked or unchecked items, depending on the value of choice.

Parameters:

choice One of ALL_ITEMS, CHECKED_ITEMS or UNCHECKED_ITEMS.

Definition at line 399 of file main.cpp.

References KitListDao::get_all_items(), KitListDao::get_category(), m_dao, and tick_items().

7.9.3.17 void KitList::list_categories ()

Lists details of all categories to STDOUT.

Renders with HTML if the html_flag has been set using the '–html' command line option.

Definition at line 452 of file main.cpp.

References KitListDao::get_categories(), html_flag, and m_dao.

Referenced by main().

7.9.3.18 void KitList::new_category (const string *name*)

Creates a new category with the passed name.

Definition at line 501 of file main.cpp.

References m_dao, and KitListDao::new_category().

Referenced by main().

7.9.3.19 void KitList::delete_category (long *id*)

Deletes the Category with the specified id.

Parameters:

id The Category id.

Definition at line 440 of file main.cpp.

References KitListDao::delete_category(), and m_dao.

Referenced by main().

7.9.3.20 void KitList::delete_item (long *id*)

Deletes the item with the passed id.

Definition at line 418 of file main.cpp.

References KitListDao::delete_item(), and m_dao.

Referenced by main().

7.9.3.21 void KitList::remove_item_from_category (long *id*, long *cat_id*)

Un-associates the specified item from the specified category.

Parameters:

id The Item id.

cat_id The Category id.

Definition at line 430 of file main.cpp.

References m_dao, and KitListDao::remove_item_from_category().

Referenced by main().

7.9.3.22 void KitList::set_item_flag (long *id*)

Checks/ticks the specified item.

Definition at line 510 of file main.cpp.

References m_dao, and KitListDao::set_item_flag().

Referenced by main().

7.9.3.23 void KitList::unset_item_flag (long *id*)

Unchecks/unticks the specified item.

Definition at line 519 of file main.cpp.

References m_dao, and KitListDao::unset_item_flag().

Referenced by main().

7.9.3.24 void KitList::set_category_flag (long *id*)

Checks/ticks all items in the specified Category.

Definition at line 528 of file main.cpp.

References m_dao, and KitListDao::set_category_flag().

Referenced by main().

7.9.3.25 void KitList::unset_category_flag (long *id*)

Unchecks/unticks all items in the specified Category.

Definition at line 537 of file main.cpp.

References m_dao, and KitListDao::unset_category_flag().

Referenced by main().

7.9.3.26 void KitList::set_all_flags ()

Checks/ticks all items.

Definition at line 546 of file main.cpp.

References m_dao, and KitListDao::set_all_flags().

Referenced by main().

7.9.3.27 void KitList::unset_all_flags ()

Unchecks/unticks all items.

Definition at line 555 of file main.cpp.

References m_dao, and KitListDao::unset_all_flags().

Referenced by main().

7.9.4 Member Data Documentation

7.9.4.1 KitListDao* KitList::m_dao [protected]

Reference to an implementation DAO class.

Definition at line 85 of file main.cpp.

Referenced by add_item(), append_items_to_category(), associate_item_with_category(), delete_category(), delete_item(), get_dao(), KitList(), list_categories(), list_items(), new_category(), remove_item_from_category(), set_all_flags(), set_category_flag(), set_item_flag(), tick_items(), unset_all_flags(), unset_category_flag(), unset_item_flag(), and ~KitList().

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

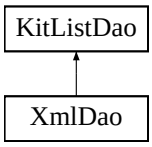
- main.cpp

7.10 KitListDao Class Reference

Defines the methods that an implementation of this class must implement.

`#include <kitlistdao.hpp>`

Inheritance diagram for KitListDao::



Public Member Functions

- KitListDao (int verbose=0)

  Constructor which will use default database connection parameters.

- virtual ~KitListDao ()
- virtual KitModel * get_model ()=0

  Loads the data model.

- virtual void save_model (KitModel *model)=0

  Saves the current data model.

- void set_verbose (int verbose_flag)
- int is_verbose ()
- virtual Category * get_category (long cat_id, item_choice choice=ALL_ITEMS)=0

  Loads a category.

- virtual ItemContainer * get_all_items (item_choice choice=ALL_ITEMS)=0

  Returns a list of all items.

- virtual long add_item (const std::string name)=0
- virtual long add_item (const std::string name, long cat_id)=0
- virtual void append_items_to_category (long to_cat_id, long from_cat_id=-1, item_choice choice=ALL_ITEMS)=0

  Copies items from one category to another.

- virtual void associate_item_with_category (long id, long cat_id)=0

  Associates an existing item with an existing category.

- virtual CategoryContainer get_categories ()=0
- virtual long new_category (const std::string name)=0

  Creates a new category.

- virtual void delete_item (long id)=0
- virtual void update_item_checked_state (ItemContainer &items)=0

  Persists the state of the 'checked' flag of each item.

- virtual void remove_item_from_category (long id, long cat_id)=0
- virtual long get_next_item_id ()=0
- virtual long get_next_category_id ()=0
- virtual void delete_category (long id)=0
- virtual void set_item_flag (long id)=0
- virtual void unset_item_flag (long id)=0
- virtual void set_category_flag (long id)=0
- virtual void unset_category_flag (long id)=0
- virtual void set_all_flags ()=0
- virtual void unset_all_flags ()=0
- virtual bool require_filename ()

  Indicates whether the implementation of the data model requires a filename.

Protected Attributes

- int m_verbose_flag

7.10.1 Detailed Description

Defines the methods that an implementation of this class must implement.

Definition at line 46 of file kitlistdao.hpp.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 KitListDao::KitListDao (int *verbose* = 0) [inline]

Constructor which will use default database connection parameters.

Parameters:

verbose Optional parameter indicating whether to include verbose output to STDOUT. Defaults to false.

Definition at line 59 of file kitlistdao.hpp.

7.10.2.2 virtual KitListDao::~KitListDao () [inline, virtual]

Definition at line 61 of file kitlistdao.hpp.

7.10.3 Member Function Documentation

7.10.3.1 virtual KitModel* KitListDao::get_model () [pure virtual]

Loads the data model.

The data model holds a rich graph of objects, representing the entire list of categories and items.

See also:

KitModel

Implemented in XmlDao.

Referenced by Service::create_default_model(), and Service::load_model().

7.10.3.2 virtual void KitListDao::save_model (KitModel * *model*) [pure virtual]

Saves the current data model.

Note: The data model will only be saved if it is dirty. Further, only items that are individually flagged as dirty will be saved.

See also:

GuiState

Implemented in XmlDao.

Referenced by Service::save().

7.10.3.3 void KitListDao::set_verbose (int *verbose_flag*) [inline]

Indicates whether verbose output should be written to STDOUT.

Definition at line 85 of file kitlistdao.hpp.

References m_verbose_flag.

7.10.3.4 int KitListDao::is_verbose () [inline]

Indicates whether verbose output should be written to STDOUT.

Definition at line 92 of file kitlistdao.hpp.

References m_verbose_flag.

7.10.3.5 virtual Category* KitListDao::get_category (long *cat_id*, item_choice *choice* = ALL_ITEMS) [pure virtual]

Loads a category.

Parameters:

choice Which items to load for the category. One of ALL_ITEMS, CHECKED_ITEMS or UNCHECKED_ITEMS.

Implemented in XmlDao.

Referenced by KitList::list_items(), and KitList::tick_items().

7.10.3.6 virtual ItemContainer* KitListDao::get_all_items (item_choice *choice* = ALL_ITEMS) [pure virtual]

Returns a list of all items.

Parameters:

choice Which items to load. One of ALL_ITEMS, CHECKED_ITEMS or UNCHECKED_ITEMS.

Implemented in XmlDao.

Referenced by KitList::list_items(), and KitList::tick_items().

7.10.3.7 virtual long KitListDao::add_item (const std::string *name*) [pure virtual]

Creates a new item.

Parameters:

name The name of the new item.

Implemented in XmlDao.

Referenced by KitList::add_item().

7.10.3.8 virtual long KitListDao::add_item (const std::string *name*, long *cat_id*) [pure virtual]

Creates a new item and associates it with a category.

Parameters:

name The name of the new item.

Implemented in XmlDao.

7.10.3.9 virtual void KitListDao::append_items_to_category (long *to_cat_id*, long *from_cat_id* = -1, item_choice *choice* = ALL_ITEMS) [pure virtual]

Copies items from one category to another.

Optionally, only checked or unchecked items are copied.

Parameters:

from_cat_id The ID of the source category.

to_cat_id The ID of the target category.

choice One of ALL_ITEMS, CHECKED_ITEMS or UNCHECKED_ITEMS.

Implemented in XmlDao.

Referenced by KitList::append_items_to_category().

7.10.3.10 virtual void KitListDao::associate_item_with_category (long *id*, long *cat_id*) [pure virtual]

Associates an existing item with an existing category.

Parameters:

id The Item ID

cat_id The Category ID

Implemented in XmlDao.

Referenced by KitList::associate_item_with_category().

7.10.3.11 virtual CategoryContainer KitListDao::get_categories () [pure virtual]

Returns a list of all categories.

Implemented in XmlDao.

Referenced by KitList::list_categories().

7.10.3.12 virtual long KitListDao::new_category (const std::string *name*) [pure virtual]

Creates a new category.

Parameters:

name the name of the new category.

Implemented in XmlDao.

Referenced by KitList::new_category().

7.10.3.13 virtual void KitListDao::delete_item (long *id*) [pure virtual]

Deletes an item by it's ID.

Parameters:

id the ID of the item to delete.

Implemented in XmlDao.

Referenced by KitList::delete_item().

7.10.3.14 virtual void KitListDao::update_item_checked_state (ItemContainer & *items*) [pure virtual]

Persists the state of the 'checked' flag of each item.

Implemented in XmlDao.

Referenced by KitList::tick_items().

7.10.3.15 virtual void KitListDao::remove_item_from_category (long *id*, long *cat_id*) [pure virtual]

Un-associates the specified item from the specified category.

Parameters:

id The Item id.

cat_id The Category id.

Implemented in XmlDao.

Referenced by KitList::remove_item_from_category().

7.10.3.16 virtual long KitListDao::get_next_item_id () [pure virtual]

Returns the next unused unique id for items.

Implemented in XmlDao.

Referenced by Service::get_next_item_id().

7.10.3.17 virtual long KitListDao::get_next_category_id () [pure virtual]

Returns the next unused unique id for categories.

Implemented in XmlDao.

Referenced by Service::get_next_category_id().

7.10.3.18 virtual void KitListDao::delete_category (long *id*) [pure virtual]

Deletes a category.

Implemented in XmlDao.

Referenced by KitList::delete_category().

7.10.3.19 virtual void KitListDao::set_item_flag (long *id*) [pure virtual]

Sets the checked flag for an item.

Parameters:

id The id of the item to change.

Implemented in XmlDao.

Referenced by KitList::set_item_flag().

7.10.3.20 virtual void KitListDao::unset_item_flag (long *id*) [pure virtual]

Clears the checked flag for an item.

Implemented in XmlDao.

Referenced by KitList::unset_item_flag().

7.10.3.21 virtual void KitListDao::set_category_flag (long *id*) [pure virtual]

Checks/ticks all items in the specified Category.

Implemented in XmlDao.

Referenced by KitList::set_category_flag().

7.10.3.22 virtual void KitListDao::unset_category_flag (long *id*) [pure virtual]

Unchecks/unticks all items in the specified Category.

Implemented in XmlDao.

Referenced by KitList::unset_category_flag().

7.10.3.23 virtual void KitListDao::set_all_flags () [pure virtual]

Checks/ticks all items.

Implemented in XmlDao.

Referenced by KitList::set_all_flags().

7.10.3.24 virtual void KitListDao::unset_all_flags () [pure virtual]

Unchecks/unticks all items.

Implemented in XmlDao.

Referenced by KitList::unset_all_flags().

7.10.3.25 virtual bool KitListDao::require_filename () [inline, virtual]

Indicates whether the implementation of the data model requires a filename.

Some persistence models may require a filename to be chosen, others (e.g. database) may be defined through another mechanism. If a filename has not been set, then fire up save-as instead.

Returns:

Always returns false.

Reimplemented in XmlDao.

Definition at line 225 of file kitlistdao.hpp.

Referenced by Service::require_filename().

7.10.4 Member Data Documentation

7.10.4.1 int KitListDao::m_verbose_flag [protected]

Optional parameter indicating whether to include verbose output to STDOUT. Defaults to false.

Definition at line 50 of file kitlistdao.hpp.

Referenced by is_verbose(), and set_verbose().

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

- kitlistdao.hpp

7.11 KitListGui Class Reference

Encapsulates the methods for the application's GUI front end.

`#include <kitlistgui.hpp>`

Public Member Functions

- KitListGui (int argc, char **argv, Service &service)
- ~KitListGui ()
- virtual void raise ()

  Make this application topmost.

- virtual void safe_open_file (const Glib::ustring &filename)

  Opens an existing XML document, checking for unsaved changes.

- void run ()

  Starts the GUI application running.

Protected Member Functions

- virtual void init ()
- virtual gint get_max_recent_files ()
- virtual ModelItemContainer * get_selected_items ()

  Returns a list of items selected in the item list.

- virtual void add_items (const ModelItemContainer &items)

  Associates the passed list of items with the currently selected category.

- virtual void close_add_item_window ()
- virtual void cancel_add_item_window ()
- virtual void close_add_category_window ()

  Called when the add/rename category dialog is closed using the 'OK' button.

- virtual void cancel_add_category_window ()

  Called when the add/rename category dialog is closed using the 'Cancel' button.

- virtual long get_selected_category ()

  Returns the ID of the currently selected Category.

- virtual void init_add_item_window ()
- virtual void delete_selected_items ()

  Deletes the currently selected items.

- virtual ModelItemContainer * copy_selected_items_to_clipboard ()
- virtual bool confirm_lose_changes (const Glib::ustring &message)

  Shows a confirmation message.

- virtual void update_recent_files_menu ()

Updates the recent files sub menu.

- virtual void update_recent_files (const Glib::ustring &filename)
- virtual bool on_delete_event (GdkEventAny *event)

  Called when the application is closed by the user.

- virtual void on_menu_quit ()

  Called when the user chooses to quit the application.

- virtual void on_menu_file_new ()

  Creates a new empty model.

- virtual void on_menu_file_open ()

  Allows the user to open an existing XML document.

- virtual void open_file (const Glib::ustring &filename)

  Opens an existing XML document.

- virtual void on_menu_save ()

  Saves the current application state.

- virtual void on_menu_save_as ()

  Saves the current application state in a new document.

- virtual void on_menu_recent_file (const Glib::ustring &filename)

  displays the most recent files menu

- virtual void on_menu_add ()
- virtual void on_menu_delete ()

  Called when the user chooses to delete items.

- virtual void on_menu_cut ()
- virtual void on_menu_copy ()

  Called when the use chooses the 'copy' menu option.

- virtual void on_menu_paste ()

  Called when the user chooses the 'paste' menu option.

- virtual void on_menu_show_all ()

  Causes all items to be displayed.

- virtual void on_menu_show_checked ()

  Causes only checked items to be displayed.

- virtual void on_menu_show_unchecked ()

  Causes only unchecked items to be displayed.

- virtual void on_menu_select_all ()

  Called when the user chooses the 'select all' menu option.

- virtual void on_menu_check_selected ()

  Marks all selected items as checked.

- virtual void on_menu_uncheck_selected ()

  Marks all selected items as unchecked.

- virtual void on_menu_create_category ()

  Called when the user chooses to create a new category.

- virtual void on_menu_delete_category ()

  Called when the user chooses the delete category menu option.

- virtual void on_menu_rename_category ()

  Called when the user chooses to rename a category.

- virtual void on_menu_help_about ()
- virtual void on_menu_help_contents ()
- virtual void on_clipboard_get (Gtk::SelectionData &selection_date, guint)
- virtual void on_clipboard_clear ()

  This method gets called after a second 'copy' operation.

- virtual void on_clipboard_received (const Gtk::SelectionData &selection_data)
- virtual void on_category_change ()
- virtual void on_cell_edit (const Glib::ustring s)

  Callback method called when a cell has been edited in the item list.

- virtual bool choose_filename (Glib::ustring &filename)

  Displays a file chooser dialog for a user to choose a filename.

- virtual void update_paste_status ()

  Enables or disables the paste menu option.

- virtual void paste_status_received (const Glib::StringArrayHandle &targets_array)
- virtual void paste_from_xml (const Glib::ustring &document)
- virtual void refresh_item_list ()
- virtual void refresh_category_list (long cat_id=-2)

  Refreshes the category combo box list.

- virtual void selected_row_callback (const Gtk::TreeModel::iterator &iter)
- virtual void set_selected (bool checked)
- virtual void toggle_selected ()
- void on_row_changed (const Gtk::TreeModel::Path path, const Gtk::TreeModel::iterator iter)

  Callback method called when an item has been modified in the item list.

- virtual void update_item_count (size_t n)

Protected Attributes

- Glib::ustring m_filename

  The filename currently associated with the loaded model.

- Glib::ustring m_clipboard_items

  Holder for items pasted to the clipboard.

- bool m_ignore_list_events

  Temporarily ignore events on the item list.

- Gtk::Main m_kit

  The main application.

- Gtk::Window * m_window

  The main application window.

- Gtk::Window * m_window_add_item

  The 'Add Item' dialog.

- Gtk::Window * m_window_add_category

  The 'Add Category' dialog.

- Gtk::Entry * m_entry_add_item

  The text entry field of the 'Add Item' dialog.

- Gtk::Entry * m_entry_add_category

  The text entry field of the 'Add Category' dialog.

- Gtk::ImageMenuItem * m_file_save_menu_item

  The file save menu item.

- Gtk::ToolButton * m_file_save_tool_button

  The file save toolbar button.

- Gtk::MenuItem * m_recent_files_menu_item

  The recent files menu item.

- Gtk::ImageMenuItem * m_paste_menu_item

  The menu paste button.

- Gtk::CheckButton * m_checkbutton_add_item

  The check button field of the 'Add Item' dialog.

- Gtk::ComboBox * m_category_combo

  The combo box holding a list of categories.

- Glib::RefPtr< Gtk::ListStore > m_ref_category_list_store

  The model backing the category combo box.

- ModelCategoryColumns m_category_cols
  The definition of the category combo box columns.
- Gtk::TreeView * m_item_tree_view
  The item list view definition.
- ModelItemColumns m_item_cols
  The definition of the item list's columns.
- Service & m_service
  The business/service object.
- Glib::RefPtr< Gtk::ListStore > m_ref_item_tree_model
  The model backing the item list.
- Gtk::Statusbar * m_status_bar
  The application status bar.
- enum gui_state m_state
  Indicates whether a category is being created or renamed.
- long m_current_cat_id
  temporary reference to a category id, usually being renamed

7.11.1 Detailed Description

Encapsulates the methods for the application's GUI front end.

This is the GTK+ implementation.

Definition at line 97 of file kitlistgui.hpp.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 KitListGui::KitListGui (int *argc*, char ** *argv*, Service & *service*)

Constructor to create the GUI application.

Parameters:

argc command line argument count passed to Gtk::Main
argv command line arguments passed to Gtk::Main
service a reference to the Service object, providing all business/service methods.

Definition at line 277 of file kitlistgui.cpp.

References init().

7.11.2.2 KitListGui::~KitListGui ()

Definition at line 303 of file kitlistgui.cpp.

7.11.3 Member Function Documentation

7.11.3.1 void KitListGui::init () [protected, virtual]

Initialises all the GUI components prior to the GUI application being run.

Definition at line 1859 of file kitlistgui.cpp.

References cancel_add_category_window(), cancel_add_item_window(), close_add_category_window(), close_add_item_window(), anonymous_namespace{kitlistgui.cpp}::DEFAULT_FILENAME, file_exists(), anonymous_namespace{kitlistgui.cpp}::GCONF_KEY, anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_CURRENT_FILENAME, get_glade_ref_ptr(), Service::get_items(), anonymous_namespace{kitlistgui.cpp}::GLADE_APP_FILE, KITLIST_SERVICE_IFACE, KITLIST_SERVICE_NAME, KITLIST_SERVICE_OBJECT, m_category_cols, m_category_combo, m_checkbutton_add_item, ModelItemColumns::m_col_checked, ModelItemColumns::m_col_num, ModelItemColumns::m_col_text, ModelCategoryColumns::m_col_text, m_entry_add_category, m_entry_add_item, m_file_save_menu_item, m_file_save_tool_button, m_filename, m_ignore_list_events, m_item_cols, m_item_tree_view, m_paste_menu_item, m_recent_files_menu_item, m_ref_category_list_store, m_ref_item_tree_model, m_service, m_status_bar, m_window, m_window_add_category, m_window_add_item, on_category_change(), on_delete_event(), on_menu_add(), on_menu_check_selected(), on_menu_copy(), on_menu_create_category(), on_menu_cut(), on_menu_delete(), on_menu_delete_category(), on_menu_file_new(), on_menu_file_open(), on_menu_help_about(), on_menu_help_contents(), on_menu_paste(), on_menu_quit(), on_menu_rename_category(), on_menu_save(), on_menu_save_as(), on_menu_select_all(), on_menu_show_all(), on_menu_show_checked(), on_menu_show_unchecked(), on_menu_uncheck_selected(), on_row_changed(), Service::open_as_xml(), refresh_category_list(), refresh_item_list(), Service::require_filename(), Service::set_model_dirty(), toggle_selected(), update_item_count(), and update_recent_files_menu().

Referenced by KitListGui().

7.11.3.2 gint KitListGui::get_max_recent_files () [protected, virtual]

Definition at line 336 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::DEFAULT_MAX_RECENT_FILES, and anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_MAX_RECENT_FILES.

Referenced by update_recent_files().

7.11.3.3 ModelItemContainer * KitListGui::get_selected_items () [protected, virtual]

Returns a list of items selected in the item list.

Definition at line 822 of file kitlistgui.cpp.

References Service::find_item(), ModelItemColumns::m_col_num, m_item_cols, m_item_tree_view, m_ref_item_tree_model, and m_service.

Referenced by copy_selected_items_to_clipboard(), set_selected(), and toggle_selected().

7.11.3.4 void KitListGui::add_items (const ModelItemContainer & *items*) [protected, virtual]

Associates the passed list of items with the currently selected category.

Where items already exist in the Category, then are not duplicated, just silently ignored.

Definition at line 1335 of file kitlistgui.cpp.

References Service::copy_items(), get_selected_category(), m_service, and refresh_item_list().

Referenced by paste_from_xml().

7.11.3.5 void KitListGui::close_add_item_window () [protected, virtual]

Called when the user closes the 'Add Item' dialog using the 'OK' button.

Definition at line 1814 of file kitlistgui.cpp.

References Service::create_item(), get_selected_category(), m_checkbutton_add_item, m_entry_add_item, m_service, m_window_add_item, refresh_item_list(), Item::set_checked(), and Item::set_description().

Referenced by init().

7.11.3.6 void KitListGui::cancel_add_item_window () [protected, virtual]

Called when the user closes the 'Add Item' dialog using the 'Cancel' button.

Definition at line 1834 of file kitlistgui.cpp.

References m_window_add_item.

Referenced by init().

7.11.3.7 void KitListGui::close_add_category_window () [protected, virtual]

Called when the add/rename category dialog is closed using the 'OK' button.

The same dialog is used for both creating a new category and renaming an existing one. m_state is used to indicate which operation is relevant (create or rename) and m_current_cat_id is used to indicate the category being renamed for the rename operation.

Definition at line 1089 of file kitlistgui.cpp.

References ADD_CATEGORY, Service::create_category(), Service::find_category(), Category::get_id(), m_current_cat_id, m_entry_add_category, m_service, m_state, m_window_add_category, refresh_category_list(), refresh_item_list(), Service::set_model_dirty(), and Category::set_name().

Referenced by init().

7.11.3.8 void KitListGui::cancel_add_category_window () [protected, virtual]

Called when the add/rename category dialog is closed using the 'Cancel' button.

Definition at line 1114 of file kitlistgui.cpp.

References m_window_add_category.

Referenced by init().

7.11.3.9 long KitListGui::get_selected_category () [protected, virtual]

Returns the ID of the currently selected Category.

Returns -1 if no Category is currently selected, or the 'all items' option is selected.

Definition at line 1797 of file kitlistgui.cpp.

References m_category_cols, m_category_combo, and ModelCategoryColumns::m_col_num.

Referenced by add_items(), close_add_item_window(), on_menu_add(), on_menu_cut(), on_menu_-delete_category(), on_menu_rename_category(), refresh_category_list(), and refresh_item_list().

7.11.3.10 void KitListGui::init_add_item_window () [protected, virtual]

Initialises the 'Add Item' dialog prior to it being displayed.

Definition at line 1784 of file kitlistgui.cpp.

References m_entry_add_item.

Referenced by on_menu_add().

7.11.3.11 void KitListGui::delete_selected_items () [protected, virtual]

Deletes the currently selected items.

The items are flagged as deleted. When save is called, they will no longer be persisted, i.e. they will be permanently deleted.

Definition at line 868 of file kitlistgui.cpp.

References Service::delete_item(), ModelItemColumns::m_col_num, m_item_cols, m_item_tree_view, m_ref_item_tree_model, m_service, refresh_item_list(), and selected_row_callback().

Referenced by on_menu_delete().

7.11.3.12 ModelItemContainer * KitListGui::copy_selected_items_to_clipboard () [protected, virtual]

Copies the currently selected items to the clipboard.

Definition at line 923 of file kitlistgui.cpp.

References get_selected_items(), anonymous_namespace{kitlistgui.cpp}::item_target_custom, anonymous_namespace{kitlistgui.cpp}::item_target_text, m_clipboard_items, on_clipboard_clear(), on_-clipboard_get(), update_paste_status(), and anonymous_namespace{kitlistgui.cpp}::XML_ELEMENT_-ID.

Referenced by on_menu_copy(), and on_menu_cut().

7.11.3.13 bool KitListGui::confirm_lose_changes (const Glib::ustring & *message*) [protected, virtual]

Shows a confirmation message.

Definition at line 393 of file kitlistgui.cpp.

References m_filename, m_service, m_window, Service::save(), Service::save_as_xml(), and Service::set_model_dirty().

Referenced by on_delete_event(), on_menu_file_new(), on_menu_file_open(), on_menu_quit(), on_-menu_recent_file(), and safe_open_file().

7.11.3.14 void KitListGui::update_recent_files_menu () [protected, virtual]

Updates the recent files sub menu.

Definition at line 431 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_RECENT_FILES, m_recent_files_menu_item, and on_menu_recent_file().

Referenced by init(), and update_recent_files().

7.11.3.15 void KitListGui::update_recent_files (const Glib::ustring & *filename*) [protected, virtual]

Updates the list of most recently used files.

Parameters:

filename The filename to append to the list it it does not already exist.

Definition at line 468 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_RECENT_FILES, get_max_recent_files(), and update_recent_files_menu().

Referenced by on_menu_recent_file(), on_menu_save(), on_menu_save_as(), and open_file().

7.11.3.16 bool KitListGui::on_delete_event (GdkEventAny * *event*) [protected, virtual]

Called when the application is closed by the user.

Definition at line 497 of file kitlistgui.cpp.

References confirm_lose_changes(), Service::is_model_dirty(), m_service, and Service::set_model_dirty().

Referenced by init().

7.11.3.17 void KitListGui::on_menu_quit () [protected, virtual]

Called when the user chooses to quit the application.

Definition at line 511 of file kitlistgui.cpp.

References confirm_lose_changes(), Service::is_model_dirty(), m_service, m_window, and Service::set_model_dirty().

Referenced by init().

7.11.3.18 void KitListGui::on_menu_file_new () [protected, virtual]

Creates a new empty model.

Definition at line 528 of file kitlistgui.cpp.

References confirm_lose_changes(), Service::create_default_model(), anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_CURRENT_FILENAME, Service::is_model_dirty(),

m_filename, m_service, m_status_bar, refresh_category_list(), refresh_item_list(), Service::require_-filename(), and anonymous_namespace{kitlistgui.cpp}::SB_SAVE.

Referenced by init().

7.11.3.19 void KitListGui::on_menu_file_open () [protected, virtual]

Allows the user to open an existing XML document.

If there are unsaved changes, the user is first prompted to discard them or cancel the operation.

Definition at line 575 of file kitlistgui.cpp.

References confirm_lose_changes(), anonymous_namespace{kitlistgui.cpp}::DEFAULT_FILENAME_-EXTENSION, Service::is_model_dirty(), m_service, m_status_bar, m_window, open_file(), and anonymous_namespace{kitlistgui.cpp}::SB_SAVE.

Referenced by init().

7.11.3.20 void KitListGui::open_file (const Glib::ustring & *filename*) [protected, virtual]

Opens an existing XML document.

Definition at line 633 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_CURRENT_FILENAME, m_-filename, m_service, Service::open_as_xml(), refresh_category_list(), refresh_item_list(), and update_-recent_files().

Referenced by on_menu_file_open(), and safe_open_file().

7.11.3.21 void KitListGui::on_menu_save () [protected, virtual]

Saves the current application state.

Definition at line 678 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_CURRENT_FILENAME, Service::is_model_dirty(), m_filename, m_service, m_status_bar, on_menu_save_as(), Service::require_-filename(), Service::save(), Service::save_as_xml(), anonymous_namespace{kitlistgui.cpp}::SB_SAVE, and update_recent_files().

Referenced by init().

7.11.3.22 void KitListGui::on_menu_save_as () [protected, virtual]

Saves the current application state in a new document.

Definition at line 731 of file kitlistgui.cpp.

References choose_filename(), anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_CURRENT_-FILENAME, m_filename, m_service, m_status_bar, Service::save_as_xml(), anonymous_-namespace{kitlistgui.cpp}::SB_SAVE, and update_recent_files().

Referenced by init(), and on_menu_save().

7.11.3.23 void KitListGui::on_menu_recent_file (const Glib::ustring & *filename*) [protected, virtual]

displays the most recent files menu

Definition at line 789 of file kitlistgui.cpp.

References confirm_lose_changes(), anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_-CURRENT_FILENAME, Service::is_model_dirty(), m_filename, m_service, Service::open_as_xml(), refresh_category_list(), refresh_item_list(), and update_recent_files().

Referenced by update_recent_files_menu().

7.11.3.24 void KitListGui::on_menu_add () [protected, virtual]

Called when the users chooses to create a new Item.

Definition at line 1730 of file kitlistgui.cpp.

References Service::create_item(), get_selected_category(), init_add_item_window(), m_entry_add_item, m_service, m_window, m_window_add_item, refresh_item_list(), Item::set_checked(), and Item::set_-description().

Referenced by init().

7.11.3.25 void KitListGui::on_menu_delete () [protected, virtual]

Called when the user chooses to delete items.

The user is prompted to confirm deletion, prior to the items being flagged as deleted in the model. Once the model is saved, the will be permanently deleted from the persistence store.

Definition at line 900 of file kitlistgui.cpp.

References delete_selected_items(), and m_window.

Referenced by init().

7.11.3.26 void KitListGui::on_menu_cut () [protected, virtual]

Copies the currently selected items to the clipboard as a 'cut' operation.

Definition at line 956 of file kitlistgui.cpp.

References copy_selected_items_to_clipboard(), Service::find_category(), get_selected_category(), m_service, m_status_bar, refresh_item_list(), ModelCategory::remove_items(), anonymous_-namespace{kitlistgui.cpp}::SB_MSG, GuiState::set_dirty(), and Service::set_model_dirty().

Referenced by init().

7.11.3.27 void KitListGui::on_menu_copy () [protected, virtual]

Called when the use chooses the 'copy' menu option.

Definition at line 985 of file kitlistgui.cpp.

References copy_selected_items_to_clipboard().

Referenced by init().

7.11.3.28 void KitListGui::on_menu_paste () [protected, virtual]

Called when the user chooses the 'paste' menu option.

Definition at line 994 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::item_target_text, m_clipboard_items, on_clipboard_-received(), paste_from_xml(), and update_paste_status().

Referenced by init().

7.11.3.29 virtual void KitListGui::on_menu_show_all () [inline, protected, virtual]

Causes all items to be displayed.

Definition at line 215 of file kitlistgui.hpp.

References m_service, refresh_item_list(), and Service::show_all().

Referenced by init().

7.11.3.30 virtual void KitListGui::on_menu_show_checked () [inline, protected, virtual]

Causes only checked items to be displayed.

Definition at line 217 of file kitlistgui.hpp.

References m_service, refresh_item_list(), and Service::show_checked_only().

Referenced by init().

7.11.3.31 virtual void KitListGui::on_menu_show_unchecked () [inline, protected, virtual]

Causes only unchecked items to be displayed.

Definition at line 219 of file kitlistgui.hpp.

References m_service, refresh_item_list(), and Service::show_unchecked_only().

Referenced by init().

7.11.3.32 void KitListGui::on_menu_select_all () [protected, virtual]

Called when the user chooses the 'select all' menu option.

Definition at line 1008 of file kitlistgui.cpp.

References m_item_tree_view.

Referenced by init().

7.11.3.33 virtual void KitListGui::on_menu_check_selected () [inline, protected, virtual]

Marks all selected items as checked.

Definition at line 222 of file kitlistgui.hpp.

References set_selected().

Referenced by init().

7.11.3.34 virtual void KitListGui::on_menu_uncheck_selected () [inline, protected, virtual]

Marks all selected items as unchecked.

Definition at line 224 of file kitlistgui.hpp.

References set_selected().

Referenced by init().

7.11.3.35 void KitListGui::on_menu_create_category () [protected, virtual]

Called when the user chooses to create a new category.

Definition at line 1019 of file kitlistgui.cpp.

References ADD_CATEGORY, Service::create_category(), Category::get_id(), m_current_cat_id, m_entry_add_category, m_service, m_state, m_window, m_window_add_category, refresh_category_list(), refresh_item_list(), Service::set_model_dirty(), and Category::set_name().

Referenced by init().

7.11.3.36 void KitListGui::on_menu_delete_category () [protected, virtual]

Called when the user chooses the delete category menu option.

Displays a confirmation box before deleting the currently selected category.

Definition at line 1125 of file kitlistgui.cpp.

References Service::delete_category(), get_selected_category(), m_service, m_status_bar, m_window, refresh_category_list(), refresh_item_list(), and anonymous_namespace{kitlistgui.cpp}::SB_MSG.

Referenced by init().

7.11.3.37 void KitListGui::on_menu_rename_category () [protected, virtual]

Called when the user chooses to rename a category.

Definition at line 1162 of file kitlistgui.cpp.

References Service::find_category(), Category::get_id(), Category::get_name(), get_selected_category(), m_current_cat_id, m_entry_add_category, m_service, m_state, m_status_bar, m_window, m_window_add_category, refresh_category_list(), refresh_item_list(), RENAME_CATEGORY, anonymous_namespace{kitlistgui.cpp}::SB_MSG, Service::set_model_dirty(), and Category::set_name().

Referenced by init().

7.11.3.38 void KitListGui::on_menu_help_about () [protected, virtual]

Shows the help about dialog

Definition at line 1256 of file kitlistgui.cpp.

Referenced by init().

7.11.3.39 void KitListGui::on_menu_help_contents () [protected, virtual]

Displays the help documentation

Definition at line 1242 of file kitlistgui.cpp.

Referenced by init().

7.11.3.40 void KitListGui::on_clipboard_get (Gtk::SelectionData & *selection_data*, guint) [protected, virtual]

Called when the current clipboard contents are required.

Definition at line 1293 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::item_target_custom, anonymous_namespace{kitlistgui.cpp}::item_target_text, and m_clipboard_items.

Referenced by copy_selected_items_to_clipboard().

7.11.3.41 void KitListGui::on_clipboard_clear () [protected, virtual]

This method gets called after a second 'copy' operation.

i.e. if we have previously called Clipboard::set() and then called it a second time. However, I think this is intended to only clean up data objects that may have been allocated by a previous call to on_clipboard_get() where we might have created an expensive object based on a list of IDs or something. This gives us the opportunity to release the big object.

However, in the way we're using the clipboard, we're holding the expensive object (in this case an XML document). We should probably be holding the list of selected ID's and only creating the XML document when the clipboard contents are requested.

As things are, we don't want to clear the XML document... m_clipboard_items.clear();

Definition at line 1324 of file kitlistgui.cpp.

Referenced by copy_selected_items_to_clipboard().

7.11.3.42 void KitListGui::on_clipboard_received (const Gtk::SelectionData & *selection_data*) [protected, virtual]

Callback notification method for capturing clipboard contents.

Definition at line 1383 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::item_target_custom, anonymous_namespace{kitlistgui.cpp}::item_target_text, and m_clipboard_items.

Referenced by on_menu_paste().

7.11.3.43 void KitListGui::on_category_change () [protected, virtual]

Called after the user has changed the currently selected Category.

Definition at line 1842 of file kitlistgui.cpp.

References m_ignore_list_events, and refresh_item_list().

Referenced by init().

7.11.3.44 void KitListGui::on_cell_edit (const Glib::ustring *s*) [protected, virtual]

Callback method called when a cell has been edited in the item list.

Flags the model as dirty.

Definition at line 1433 of file kitlistgui.cpp.

References m_ignore_list_events, m_service, and Service::set_model_dirty().

7.11.3.45 bool KitListGui::choose_filename (Glib::ustring & *filename*) [protected, virtual]

Displays a file chooser dialog for a user to choose a filename.

If the file already exists, the user is asked to confirm whether to overwrite.

Parameters:

filename A reference to a string to populate with the chosen filename.

Returns:

true if the user selects the OK, false otherwise.

Definition at line 1450 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::DEFAULT_FILENAME_EXTENSION, file_exists(), and m_window.

Referenced by on_menu_save_as().

7.11.3.46 void KitListGui::update_paste_status () [protected, virtual]

Enables or disables the paste menu option.

Definition at line 1549 of file kitlistgui.cpp.

References paste_status_received().

Referenced by copy_selected_items_to_clipboard(), and on_menu_paste().

7.11.3.47 void KitListGui::paste_status_received (const Glib::StringArrayHandle & *targets_array*) [protected, virtual]

Callback method for enabling or disabling the paste menu option based on the clipboard contents.

See also:

KitListGui::update_paste_status()

Definition at line 1561 of file kitlistgui.cpp.

References anonymous_namespace{kitlistgui.cpp}::item_target_custom, anonymous_namespace{kitlistgui.cpp}::item_target_text, and m_paste_menu_item.

Referenced by update_paste_status().

7.11.3.48 void KitListGui::paste_from_xml (const Glib::ustring & *document*) [protected, virtual]

Performs a paste of items from the clipboard.

Definition at line 1347 of file kitlistgui.cpp.

References add_items(), Service::find_item(), m_service, and anonymous_namespace{kitlistgui.cpp}::XML_ELEMENT_ID.

Referenced by on_menu_paste().

7.11.3.49 void KitListGui::refresh_item_list () [protected, virtual]

Refreshes the item list.

Definition at line 1636 of file kitlistgui.cpp.

References Service::filter(), Service::get_items(), get_selected_category(), ModelItemColumns::m_col_checked, ModelItemColumns::m_col_num, ModelItemColumns::m_col_text, m_ignore_list_events, m_item_cols, m_ref_item_tree_model, m_service, and update_item_count().

Referenced by add_items(), close_add_category_window(), close_add_item_window(), delete_selected_items(), init(), on_category_change(), on_menu_add(), on_menu_create_category(), on_menu_cut(), on_menu_delete_category(), on_menu_file_new(), on_menu_recent_file(), on_menu_rename_category(), on_menu_show_all(), on_menu_show_checked(), on_menu_show_unchecked(), open_file(), set_selected(), and toggle_selected().

7.11.3.50 void KitListGui::refresh_category_list (long *cat_id* = -2) [protected, virtual]

Refreshes the category combo box list.

Parameters:

cat_id the id of the category to select in the combo box. If set to -2 the currently selected category is used, otherwise the specified category ID is used. If the category ID does not exist, or if -1 is specified, then no category is selected.

Definition at line 1675 of file kitlistgui.cpp.

References Service::get_categories(), Category::get_id(), Category::get_name(), get_selected_category(), GuiState::is_deleted(), m_category_cols, m_category_combo, ModelCategoryColumns::m_col_num, ModelCategoryColumns::m_col_text, m_ignore_list_events, m_ref_category_list_store, and m_service.

Referenced by close_add_category_window(), init(), on_menu_create_category(), on_menu_delete_category(), on_menu_file_new(), on_menu_recent_file(), on_menu_rename_category(), and open_file().

7.11.3.51 void KitListGui::selected_row_callback (const Gtk::TreeModel::iterator & *iter*) [protected, virtual]

Called to delete an Item referenced by a row iterator.

Definition at line 1606 of file kitlistgui.cpp.

References Service::delete_item(), ModelItemColumns::m_col_num, m_item_cols, and m_service.

Referenced by delete_selected_items().

7.11.3.52 void KitListGui::set_selected (bool *checked*) [protected, virtual]

Checks or unchecks the currently selected items.

Definition at line 1271 of file kitlistgui.cpp.

References get_selected_items(), m_service, refresh_item_list(), and Service::select_items().

Referenced by on_menu_check_selected(), and on_menu_uncheck_selected().

7.11.3.53 void KitListGui::toggle_selected () [protected, virtual]

Toggles the checked state of the currently selected items.

Definition at line 1282 of file kitlistgui.cpp.

References get_selected_items(), m_service, refresh_item_list(), and Service::toggle_selected_items().

Referenced by init().

7.11.3.54 void KitListGui::on_row_changed (const Gtk::TreeModel::Path *path*, const Gtk::TreeModel::iterator *iter*) [protected]

Callback method called when an item has been modified in the item list.

Sets the model as dirty and copies the row details to the appropriate Item in the model. The item's state is also set to dirty.

Definition at line 1590 of file kitlistgui.cpp.

References ModelItemColumns::m_col_checked, ModelItemColumns::m_col_num, ModelItemColumns::m_col_text, m_ignore_list_events, m_item_cols, m_service, Service::set_model_dirty(), and Service::update_item().

Referenced by init().

7.11.3.55 void KitListGui::update_item_count (size_t *n*) [protected, virtual]

Writes the count of currently displayed items to the status bar.

Definition at line 1397 of file kitlistgui.cpp.

References m_status_bar, and anonymous_namespace{kitlistgui.cpp}::SB_ITEM_COUNT.

Referenced by init(), and refresh_item_list().

7.11.3.56 void KitListGui::raise () [virtual]

Make this application topmost.

Definition at line 1850 of file kitlistgui.cpp.

References m_window.

7.11.3.57 void KitListGui::safe_open_file (const Glib::ustring & *filename*) [virtual]

Opens an existing XML document, checking for unsaved changes.

If there are unsaved changes, the user is first prompted to discard them or cancel the operation.

Definition at line 664 of file kitlistgui.cpp.

References confirm_lose_changes(), Service::is_model_dirty(), m_service, m_status_bar, open_file(), and anonymous_namespace{kitlistgui.cpp}::SB_SAVE.

7.11.3.58 void KitListGui::run ()

Starts the GUI application running.

Definition at line 314 of file kitlistgui.cpp.

References Service::is_model_dirty(), m_filename, m_kit, m_service, m_window, and Service::save_as_-xml().

7.11.4 Member Data Documentation

7.11.4.1 Glib::ustring KitListGui::m_filename [protected]

The filename currently associated with the loaded model.

Should be an empty string if not related to a file.

Definition at line 108 of file kitlistgui.hpp.

Referenced by confirm_lose_changes(), init(), on_menu_file_new(), on_menu_recent_file(), on_menu_-save(), on_menu_save_as(), open_file(), and run().

7.11.4.2 Glib::ustring KitListGui::m_clipboard_items [protected]

Holder for items pasted to the clipboard.

Definition at line 114 of file kitlistgui.hpp.

Referenced by copy_selected_items_to_clipboard(), on_clipboard_get(), on_clipboard_received(), and on_menu_paste().

7.11.4.3 bool KitListGui::m_ignore_list_events [protected]

Temporarily ignore events on the item list.

Definition at line 116 of file kitlistgui.hpp.

Referenced by init(), on_category_change(), on_cell_edit(), on_row_changed(), refresh_category_list(), and refresh_item_list().

7.11.4.4 Gtk::Main KitListGui::m_kit [protected]

The main application.

Definition at line 118 of file kitlistgui.hpp.

Referenced by run().

7.11.4.5 Gtk::Window* KitListGui::m_window [protected]

The main application window.

Definition at line 130 of file kitlistgui.hpp.

Referenced by choose_filename(), confirm_lose_changes(), init(), on_menu_add(), on_menu_create_category(), on_menu_delete(), on_menu_delete_category(), on_menu_file_open(), on_menu_quit(), on_menu_rename_category(), raise(), and run().

7.11.4.6 Gtk::Window* KitListGui::m_window_add_item [protected]

The 'Add Item' dialog.

Definition at line 133 of file kitlistgui.hpp.

Referenced by cancel_add_item_window(), close_add_item_window(), init(), and on_menu_add().

7.11.4.7 Gtk::Window* KitListGui::m_window_add_category [protected]

The 'Add Category' dialog.

Definition at line 135 of file kitlistgui.hpp.

Referenced by cancel_add_category_window(), close_add_category_window(), init(), on_menu_create_category(), and on_menu_rename_category().

7.11.4.8 Gtk::Entry* KitListGui::m_entry_add_item [protected]

The text entry field of the 'Add Item' dialog.

Definition at line 137 of file kitlistgui.hpp.

Referenced by close_add_item_window(), init(), init_add_item_window(), and on_menu_add().

7.11.4.9 Gtk::Entry* KitListGui::m_entry_add_category [protected]

The text entry field of the 'Add Category' dialog.

Definition at line 139 of file kitlistgui.hpp.

Referenced by close_add_category_window(), init(), on_menu_create_category(), and on_menu_rename_category().

7.11.4.10 Gtk::ImageMenuItem* KitListGui::m_file_save_menu_item [protected]

The file save menu item.

Definition at line 141 of file kitlistgui.hpp.

Referenced by init().

7.11.4.11 Gtk::ToolButton* KitListGui::m_file_save_tool_button [protected]

The file save toolbar button.

Definition at line 143 of file kitlistgui.hpp.

Referenced by init().

7.11.4.12 Gtk::MenuItem* KitListGui::m_recent_files_menu_item [protected]

The recent files menu item.

Definition at line 145 of file kitlistgui.hpp.

Referenced by init(), and update_recent_files_menu().

7.11.4.13 Gtk::ImageMenuItem* KitListGui::m_paste_menu_item [protected]

The menu paste button.

Definition at line 147 of file kitlistgui.hpp.

Referenced by init(), and paste_status_received().

7.11.4.14 Gtk::CheckButton* KitListGui::m_checkbutton_add_item [protected]

The check button field of the 'Add Item' dialog.

Definition at line 153 of file kitlistgui.hpp.

Referenced by close_add_item_window(), and init().

7.11.4.15 Gtk::ComboBox* KitListGui::m_category_combo [protected]

The combo box holding a list of categories.

Definition at line 155 of file kitlistgui.hpp.

Referenced by get_selected_category(), init(), and refresh_category_list().

7.11.4.16 Glib::RefPtr<Gtk::ListStore> KitListGui::m_ref_category_list_store [protected]

The model backing the category combo box.

Definition at line 157 of file kitlistgui.hpp.

Referenced by init(), and refresh_category_list().

7.11.4.17 ModelCategoryColumns KitListGui::m_category_cols [protected]

The definition of the category combo box columns.

Definition at line 159 of file kitlistgui.hpp.

Referenced by get_selected_category(), init(), and refresh_category_list().

7.11.4.18 Gtk::TreeView* KitListGui::m_item_tree_view [protected]

The item list view definition.

Definition at line 161 of file kitlistgui.hpp.

Referenced by delete_selected_items(), get_selected_items(), init(), and on_menu_select_all().

7.11.4.19 ModelItemColumns KitListGui::m_item_cols [protected]

The definition of the item list's columns.

Definition at line 163 of file kitlistgui.hpp.

Referenced by delete_selected_items(), get_selected_items(), init(), on_row_changed(), refresh_item_list(), and selected_row_callback().

7.11.4.20 Service& KitListGui::m_service [protected]

The business/service object.

Definition at line 165 of file kitlistgui.hpp.

Referenced by add_items(), close_add_category_window(), close_add_item_window(), confirm_lose_changes(), delete_selected_items(), get_selected_items(), init(), on_cell_edit(), on_delete_event(), on_menu_add(), on_menu_create_category(), on_menu_cut(), on_menu_delete_category(), on_menu_file_new(), on_menu_file_open(), on_menu_quit(), on_menu_recent_file(), on_menu_rename_category(), on_menu_save(), on_menu_save_as(), on_menu_show_all(), on_menu_show_checked(), on_menu_show_unchecked(), on_row_changed(), open_file(), paste_from_xml(), refresh_category_list(), refresh_item_list(), run(), safe_open_file(), selected_row_callback(), set_selected(), and toggle_selected().

7.11.4.21 Glib::RefPtr<Gtk::ListStore> KitListGui::m_ref_item_tree_model [protected]

The model backing the item list.

Definition at line 167 of file kitlistgui.hpp.

Referenced by delete_selected_items(), get_selected_items(), init(), and refresh_item_list().

7.11.4.22 Gtk::Statusbar* KitListGui::m_status_bar [protected]

The application status bar.

Definition at line 169 of file kitlistgui.hpp.

Referenced by init(), on_menu_cut(), on_menu_delete_category(), on_menu_file_new(), on_menu_file_open(), on_menu_rename_category(), on_menu_save(), on_menu_save_as(), safe_open_file(), and update_item_count().

7.11.4.23 enum gui_state KitListGui::m_state [protected]

Indicates whether a category is being created or renamed.

Only used whilst the 'Add/Rename dialog is being displayed', or when it has been closed to determine the correct action.

Definition at line 177 of file kitlistgui.hpp.

Referenced by close_add_category_window(), on_menu_create_category(), and on_menu_rename_category().

7.11.4.24 long KitListGui::m_current_cat_id [protected]

temporary reference to a category id, usually being renamed

Definition at line 178 of file kitlistgui.hpp.

Referenced by close_add_category_window(), on_menu_create_category(), and on_menu_rename_category().

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

- kitlistgui.hpp
- kitlistgui.cpp

7.12 KitModel Class Reference

Holds a rich graph of objects representing the application's data model.

`#include <kitmodel.hpp>`

Public Member Functions

- KitModel ()
  Creates an empty data model.
- ~KitModel ()
  Destructor.
- void foreach_item_iter (const SlotForeachModelItemIter &slot)
  Calls the provided slot for each item in the map.
- void foreach_item (const SlotForeachModelItem &slot)
  Calls the provided slot for each item in the map.
- void foreach_category_iter (const SlotForeachCategoryIter &slot)
  Calls the provided slot for each category in the map.
- void foreach_category (const SlotForeachCategory &slot)
  Calls the provided slot for each category in the map.
- virtual ModelCategory * find_category (long id)
  Finds a Category by it's unique ID.
- virtual ModelItem * find_item (long id)
  Finds an item by it's unique ID.
- virtual CategoryContainer * get_categories ()
  Returns a list of all categories.
- virtual ItemContainer * get_all_items ()
  Returns a list of all items.
- virtual void add_category (ModelCategory *category)
  Add a category to the model.
- virtual void add_item (ModelItem *item)
  Adds an item to the model.
- virtual void add_item (ModelItem *item, long cat_id)
  Adds an item to the model and associates it with the specified category.
- virtual void copy_items (const ModelItemContainer &items, long cat_id=-1)
  Copies items to the specified category.

- virtual bool filter (bool checked)

  Applies the current filter.

- virtual void set_dirty (bool dirty=true)
- virtual bool is_dirty ()
- virtual void show_all ()

  Removes filter. All items are shown.

- virtual void show_checked_only ()

  Sets the filter to show only checked items.

- virtual void show_unchecked_only ()

  Sets the filter to show only unchecked items.

- virtual void reset ()

  Resets all contained objects to their default states.

- virtual void purge ()

  Purges deleted categories and items from the model.

Protected Attributes

- bool m_dirty

  Indicates whether the model needs saving. I.e it's state has changed.

- CategoryMap * m_category_map

  Map allowing categories to be located by ID.

- ItemMap * m_item_map

  Map allowing items to be located by ID.

- enum item_filter_types m_item_filter

  Indicates whether and how items are currently filtered.

7.12.1 Detailed Description

Holds a rich graph of objects representing the application's data model.

Definition at line 134 of file kitmodel.hpp.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 KitModel::KitModel ()

Creates an empty data model.

Definition at line 166 of file kitmodel.cpp.

References m_category_map, and m_item_map.

7.12.2.2 KitModel::~KitModel ()

Destructor.

Deletes all items and categories belonging to the model.

Definition at line 177 of file kitmodel.cpp.

References m_category_map, and m_item_map.

7.12.3 Member Function Documentation

7.12.3.1 void KitModel::foreach_item_iter (const SlotForeachModelItemIter & *slot*)

Calls the provided slot for each item in the map.

Definition at line 197 of file kitmodel.cpp.

References m_item_map.

7.12.3.2 void KitModel::foreach_item (const SlotForeachModelItem & *slot*)

Calls the provided slot for each item in the map.

Definition at line 209 of file kitmodel.cpp.

References m_item_map.

Referenced by XmlDao::save_model().

7.12.3.3 void KitModel::foreach_category_iter (const SlotForeachCategoryIter & *slot*)

Calls the provided slot for each category in the map.

Definition at line 221 of file kitmodel.cpp.

References m_category_map.

7.12.3.4 void KitModel::foreach_category (const SlotForeachCategory & *slot*)

Calls the provided slot for each category in the map.

Definition at line 233 of file kitmodel.cpp.

References m_category_map.

Referenced by XmlDao::save_model().

7.12.3.5 ModelCategory * KitModel::find_category (long *id*) [virtual]

Finds a Category by it's unique ID.

Definition at line 244 of file kitmodel.cpp.

References m_category_map.

Referenced by add_item(), copy_items(), Service::delete_category(), Service::find_category(), and Service::get_items().

7.12.3.6 ModelItem * KitModel::find_item (long *id*) [virtual]

Finds an item by it's unique ID.

Definition at line 257 of file kitmodel.cpp.

References m_item_map.

Referenced by Service::delete_item(), Service::find_item(), KitParser::process_category_item(), and Service::update_item().

7.12.3.7 CategoryContainer * KitModel::get_categories () [virtual]

Returns a list of all categories.

Definition at line 279 of file kitmodel.cpp.

References GuiState::is_deleted(), and m_category_map.

Referenced by Service::get_categories(), and XmlDao::get_model().

7.12.3.8 ItemContainer * KitModel::get_all_items () [virtual]

Returns a list of all items.

Excluded items are excluded from the list. The caller is responsible for deleting the returned item list.

Definition at line 306 of file kitmodel.cpp.

References m_item_map.

Referenced by Service::get_items(), and XmlDao::get_model().

7.12.3.9 void KitModel::add_category (ModelCategory * *category*) [virtual]

Add a category to the model.

The category is included in the model's map.

Definition at line 322 of file kitmodel.cpp.

References Category::get_id(), and m_category_map.

Referenced by Service::create_category(), and KitParser::process_category().

7.12.3.10 void KitModel::add_item (ModelItem * *item*) [virtual]

Adds an item to the model.

The item is included in the model's map.

Definition at line 332 of file kitmodel.cpp.

References Item::get_id(), and m_item_map.

Referenced by Service::create_item(), and KitParser::process_item().

7.12.3.11 void KitModel::add_item (ModelItem * *item*, long *cat_id*) [virtual]

Adds an item to the model and associates it with the specified category.

The category must have already been added to the model's map.

Definition at line 343 of file kitmodel.cpp.

References ModelCategory::add_item(), find_category(), Item::get_id(), and m_item_map.

7.12.3.12 void KitModel::copy_items (const ModelItemContainer & *items*, long *cat_id* = -1) [virtual]

Copies items to the specified category.

Only copies those items that do not already exist in the target category.

Definition at line 360 of file kitmodel.cpp.

References ModelCategory::add_item(), find_category(), Category::m_items, and GuiState::set_dirty().

Referenced by Service::copy_items().

7.12.3.13 bool KitModel::filter (bool *checked*) [virtual]

Applies the current filter.

Parameters:

checked The checked/ticked state of the item being filtered.

Returns:

true if the item should be included.

Definition at line 407 of file kitmodel.cpp.

References ALL, CHECKED, m_item_filter, and UNCHECKED.

Referenced by Service::filter().

7.12.3.14 virtual void KitModel::set_dirty (bool *dirty* = true) [inline, virtual]

Definition at line 174 of file kitmodel.hpp.

References m_dirty.

Referenced by Service::copy_items(), Service::create_category(), Service::create_item(), Service::delete_category(), Service::delete_item(), Service::select_items(), Service::set_model_dirty(), and Service::toggle_selected_items().

7.12.3.15 virtual bool KitModel::is_dirty () [inline, virtual]

Definition at line 175 of file kitmodel.hpp.

References m_dirty.

Referenced by Service::is_model_dirty().

7.12.3.16 virtual void KitModel::show_all () [inline, virtual]

Removes filter. All items are shown.

Definition at line 177 of file kitmodel.hpp.

References ALL, and m_item_filter.

Referenced by Service::show_all().

7.12.3.17 virtual void KitModel::show_checked_only () [inline, virtual]

Sets the filter to show only checked items.

Definition at line 179 of file kitmodel.hpp.

References CHECKED, and m_item_filter.

Referenced by Service::show_checked_only().

7.12.3.18 virtual void KitModel::show_unchecked_only () [inline, virtual]

Sets the filter to show only unchecked items.

Definition at line 181 of file kitmodel.hpp.

References m_item_filter, and UNCHECKED.

Referenced by Service::show_unchecked_only().

7.12.3.19 void KitModel::reset () [virtual]

Resets all contained objects to their default states.

This method needs to be called after load or save operations to ensure all the dirty, deleted and new flags of each object are reset.

After a save operation, KitModel::purge() should be called before this method.

Definition at line 427 of file kitmodel.cpp.

References GuiState::is_deleted(), m_category_map, m_dirty, m_item_map, GuiState::reset(), and ModelCategory::reset().

Referenced by Service::create_default_model(), XmlDao::get_model(), and XmlDao::save_model().

7.12.3.20 void KitModel::purge () [virtual]

Purges deleted categories and items from the model.

Typically, this method is called after a save operation, but before calling KitModel::reset()

Definition at line 447 of file kitmodel.cpp.

References GuiState::is_deleted(), m_category_map, m_item_map, and ModelCategory::purge().

Referenced by XmlDao::save_model().

7.12.4 Member Data Documentation

7.12.4.1 bool KitModel::m_dirty [protected]

Indicates whether the model needs saving. I.e it's state has changed.

Definition at line 137 of file kitmodel.hpp.

Referenced by is_dirty(), reset(), and set_dirty().

7.12.4.2 CategoryMap* KitModel::m_category_map [protected]

Map allowing categories to be located by ID.

The map's key is the category ID, with the second field of the pair containing a pointer to the ModelCategory instance.

Definition at line 144 of file kitmodel.hpp.

Referenced by add_category(), find_category(), foreach_category(), foreach_category_iter(), get_categories(), KitModel(), purge(), reset(), and ~KitModel().

7.12.4.3 ItemMap* KitModel::m_item_map [protected]

Map allowing items to be located by ID.

The map's key is the item ID, with the second field of the pair containing a pointer to the ModelItem instance.

Definition at line 151 of file kitmodel.hpp.

Referenced by add_item(), find_item(), foreach_item(), foreach_item_iter(), get_all_items(), KitModel(), purge(), reset(), and ~KitModel().

7.12.4.4 enum item_filter_types KitModel::m_item_filter [protected]

Indicates whether and how items are currently filtered.

One of ALL, CHECKED or UNCHECKED.

Definition at line 157 of file kitmodel.hpp.

Referenced by filter(), show_all(), show_checked_only(), and show_unchecked_only().

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

- kitmodel.hpp
- kitmodel.cpp

7.13 KitParser Class Reference

SaxParser implementation for reading the KitModel from an XML document.

`#include <kitparser.hpp>`

Public Member Functions

- KitParser (KitModel &model)

  Constructor taking the rich data model to save the XML document within.

- virtual ~KitParser ()

Protected Member Functions

- virtual void on_start_document ()

  Does nothing. Called at the start of a document.

- virtual void on_end_document ()

  Does nothing. Called at the end of a document.

- virtual void on_start_element (const Glib::ustring &name, const AttributeList &attributes)

  Called for each element.

- virtual void on_end_element (const Glib::ustring &name)

  Called at the end of each element.

- virtual void on_characters (const Glib::ustring &text)

  Called for each piece of CDATA belonging to an element.

- virtual void on_comment (const Glib::ustring &text)

  Does nothing. Called for each comment.

- virtual void on_warning (const Glib::ustring &text)

  Outputs any warnings to STDOUT.

- virtual void on_error (const Glib::ustring &text)

  Outputs any errors to STDOUT.

- virtual void on_fatal_error (const Glib::ustring &text)

  Outputs any fatal errors to STDOUT.

Protected Attributes

- KitModel & m_model

Private Member Functions

- void process_item (const AttributeList &attributes)
  <The most recently processed CDATA
- void process_category (const AttributeList &attributes)
  Reads a category's attributes from a 'category' element.
- void process_category_item (const AttributeList &attributes)
  Reads details of an item association with a category from a 'category-item' element.

Private Attributes

- ModelCategory * m_category
  The most recently processed category element.
- ModelItem * m_item
  The most recently processed item element.
- Glib::ustring m_cdata

7.13.1 Detailed Description

SaxParser implementation for reading the KitModel from an XML document.

Definition at line 35 of file kitparser.hpp.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 KitParser::KitParser (KitModel & *model*) [inline]

Constructor taking the rich data model to save the XML document within.

Definition at line 45 of file kitparser.hpp.

7.13.2.2 virtual KitParser::~KitParser () [inline, virtual]

Definition at line 46 of file kitparser.hpp.

7.13.3 Member Function Documentation

7.13.3.1 void KitParser::process_item (const AttributeList & *attributes*) [private]

<The most recently processed CDATA

Reads an item's attributes from an 'item' element.

Parameters:

attributes The list of attributes for the element.

Definition at line 45 of file kitparser.cpp.

References KitModel::add_item(), m_item, m_model, ModelItem::set_checked(), and Item::set_id().

Referenced by on_start_element().

7.13.3.2 void KitParser::process_category (const AttributeList & *attributes*) [private]

Reads a category's attributes from a 'category' element.

Parameters:

attributes The list of attributes for the element.

Definition at line 68 of file kitparser.cpp.

References KitModel::add_category(), m_category, m_model, and Category::set_id().

Referenced by on_start_element().

7.13.3.3 void KitParser::process_category_item (const AttributeList & *attributes*) [private]

Reads details of an item association with a category from a 'category-item' element.

Parameters:

attributes The list of attributes for the element.

Definition at line 87 of file kitparser.cpp.

References ModelCategory::add_item(), KitModel::find_item(), Category::get_id(), m_category, and m_-model.

Referenced by on_start_element().

7.13.3.4 void KitParser::on_start_document () [protected, virtual]

Does nothing. Called at the start of a document.

Definition at line 29 of file kitparser.cpp.

7.13.3.5 void KitParser::on_end_document () [protected, virtual]

Does nothing. Called at the end of a document.

Definition at line 35 of file kitparser.cpp.

7.13.3.6 void KitParser::on_start_element (const Glib::ustring & *name*, const AttributeList & *attributes*) [protected, virtual]

Called for each element.

Determines which element is being processed and calls appropriate method to process the corresponding attributes.

Definition at line 115 of file kitparser.cpp.

References m_cdata, process_category(), process_category_item(), and process_item().

7.13.3.7 void KitParser::on_end_element (const Glib::ustring & *name*) [protected, virtual]

Called at the end of each element.

If any CDATA existed in the element body, sets the text belonging to the appropriate element object with the previously captured CDATA.

Definition at line 141 of file kitparser.cpp.

References m_category, m_cdata, m_item, Item::set_description(), and Category::set_name().

7.13.3.8 void KitParser::on_characters (const Glib::ustring & *text*) [protected, virtual]

Called for each piece of CDATA belonging to an element.

This may be called a number of times, each time passing a bit more of the parsed CDATA. The processing of the CDATA is split up by any embedded entities.

The CDATA is held in a private member variable. The CDATA variable is cleared at the start of each new element and appended to at each call to this method. The captured text is then set on the appropriate object during KitParser::on_end_element().

Definition at line 164 of file kitparser.cpp.

References m_cdata.

7.13.3.9 void KitParser::on_comment (const Glib::ustring & *text*) [protected, virtual]

Does nothing. Called for each comment.

Definition at line 171 of file kitparser.cpp.

7.13.3.10 void KitParser::on_warning (const Glib::ustring & *text*) [protected, virtual]

Outputs any warnings to STDOUT.

Definition at line 177 of file kitparser.cpp.

7.13.3.11 void KitParser::on_error (const Glib::ustring & *text*) [protected, virtual]

Outputs any errors to STDOUT.

Definition at line 183 of file kitparser.cpp.

7.13.3.12 void KitParser::on_fatal_error (const Glib::ustring & *text*) [protected, virtual]

Outputs any fatal errors to STDOUT.

Definition at line 189 of file kitparser.cpp.

7.13.4 Member Data Documentation

7.13.4.1 ModelCategory* KitParser::m_category [private]

The most recently processed category element.

Definition at line 37 of file kitparser.hpp.

Referenced by on_end_element(), process_category(), and process_category_item().

7.13.4.2 ModelItem* KitParser::m_item [private]

The most recently processed item element.

Definition at line 38 of file kitparser.hpp.

Referenced by on_end_element(), and process_item().

7.13.4.3 Glib::ustring KitParser::m_cdata [private]

Definition at line 39 of file kitparser.hpp.

Referenced by on_characters(), on_end_element(), and on_start_element().

7.13.4.4 KitModel& KitParser::m_model [protected]

Definition at line 48 of file kitparser.hpp.

Referenced by process_category(), process_category_item(), and process_item().

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

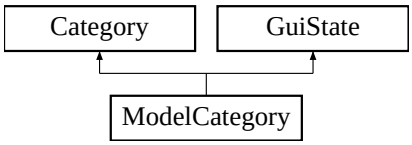
- kitparser.hpp
- kitparser.cpp

7.14 ModelCategory Class Reference

Represents a Category combined with GuiState attributes.

`#include <kitmodel.hpp>`

Inheritance diagram for ModelCategory::



Public Member Functions

- ModelCategory ()
- ~ModelCategory ()
- virtual ModelItemContainer * get_model_items ()
- virtual ItemContainer * get_items ()

  Returns the list of items belonging to the Category.

- virtual ItemMap * get_removed_children ()

  Returns a list of items that have been removed from the Category.

- virtual ItemMap * get_added_children ()

  Returns a list of items that have been added to the Category.

- virtual void add_item (Item *)

  Adds the passed item to this ModelCategory.

- virtual void remove_item (Item *item)

  Removes the passed item from this ModelCategory.

- virtual void remove_items (ModelItemContainer *items)

  Removes all the passed items from this ModelCategory.

- virtual void reset ()

  Resets the Category state.

- virtual void purge ()

Protected Attributes

- ItemMap * m_removed_children

  List of items removed from the Category.

- ItemMap * m_added_children

  List of items added to the Category.

Friends

- class KitModel

7.14.1 Detailed Description

Represents a Category combined with GuiState attributes.

Definition at line 94 of file kitmodel.hpp.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 ModelCategory::ModelCategory ()

Definition at line 43 of file kitmodel.cpp.

References m_added_children, and m_removed_children.

7.14.2.2 ModelCategory::~ModelCategory ()

Definition at line 49 of file kitmodel.cpp.

References m_added_children, and m_removed_children.

7.14.3 Member Function Documentation

7.14.3.1 ModelItemContainer * ModelCategory::get_model_items () [virtual]

Returns the list of items belonging to the Category.

Items flagged as deleted are excluded.

Definition at line 134 of file kitmodel.cpp.

References GuiState::is_deleted(), and Category::m_items.

7.14.3.2 ItemContainer * ModelCategory::get_items () [virtual]

Returns the list of items belonging to the Category.

Items flagged as deleted are excluded.

Definition at line 151 of file kitmodel.cpp.

References GuiState::is_deleted(), and Category::m_items.

Referenced by Service::get_items().

7.14.3.3 virtual ItemMap* ModelCategory::get_removed_children () [inline, virtual]

Returns a list of items that have been removed from the Category.

Definition at line 106 of file kitmodel.hpp.

References m_removed_children.

7.14.3.4 virtual ItemMap* ModelCategory::get_added_children () [inline, virtual]

Returns a list of items that have been added to the Category.

Definition at line 108 of file kitmodel.hpp.

References m_added_children.

7.14.3.5 void ModelCategory::add_item (Item * *item*) [virtual]

Adds the passed item to this ModelCategory.

Also updates the lists of removed and added items.

Reimplemented from Category.

Definition at line 88 of file kitmodel.cpp.

References Category::add_item(), Item::get_id(), m_added_children, and m_removed_children.

Referenced by KitModel::add_item(), KitModel::copy_items(), and KitParser::process_category_item().

7.14.3.6 void ModelCategory::remove_item (Item * *item*) [virtual]

Removes the passed item from this ModelCategory.

Also updates the lists of removed and added items.

Reimplemented from Category.

Definition at line 106 of file kitmodel.cpp.

References Item::get_id(), m_added_children, m_removed_children, and Category::remove_item().

Referenced by remove_items().

7.14.3.7 void ModelCategory::remove_items (ModelItemContainer * *items*) [virtual]

Removes all the passed items from this ModelCategory.

Also updates the lists of removed and added items.

Definition at line 122 of file kitmodel.cpp.

References remove_item().

Referenced by KitListGui::on_menu_cut().

7.14.3.8 void ModelCategory::reset () [virtual]

Resets the Category state.

Removes any items flagged as deleted. Sets the GuiState to it's defaults and clears the lists of added and removed items.

Reimplemented from GuiState.

Definition at line 61 of file kitmodel.cpp.

References m_added_children, m_removed_children, and GuiState::reset().

Referenced by KitModel::reset().

7.14.3.9 void ModelCategory::purge () [virtual]

Definition at line 68 of file kitmodel.cpp.

References Category::m_items.

Referenced by KitModel::purge().

7.14.4 Friends And Related Function Documentation

7.14.4.1 friend class KitModel [friend]

Reimplemented from Category.

Definition at line 114 of file kitmodel.hpp.

7.14.5 Member Data Documentation

7.14.5.1 ItemMap* ModelCategory::m_removed_children [protected]

List of items removed from the Category.

Definition at line 97 of file kitmodel.hpp.

Referenced by add_item(), get_removed_children(), ModelCategory(), remove_item(), reset(), and ~ModelCategory().

7.14.5.2 ItemMap* ModelCategory::m_added_children [protected]

List of items added to the Category.

Definition at line 99 of file kitmodel.hpp.

Referenced by add_item(), get_added_children(), ModelCategory(), remove_item(), reset(), and ~ModelCategory().

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

- kitmodel.hpp
- kitmodel.cpp

7.15 ModelCategoryColumns Class Reference

A definition for displaying a ModelCategory in a combo box.

`#include <kitlistgui.hpp>`

Public Member Functions

- ModelCategoryColumns ()

Public Attributes

- Gtk::TreeModelColumn< Glib::ustring > m_col_text
- Gtk::TreeModelColumn< int > m_col_num

7.15.1 Detailed Description

A definition for displaying a ModelCategory in a combo box.

Definition at line 56 of file kitlistgui.hpp.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 ModelCategoryColumns::ModelCategoryColumns () [inline]

Definition at line 59 of file kitlistgui.hpp.

References m_col_num, and m_col_text.

7.15.3 Member Data Documentation

7.15.3.1 Gtk::TreeModelColumn<Glib::ustring> ModelCategoryColumns::m_col_text

Definition at line 64 of file kitlistgui.hpp.

Referenced by KitListGui::init(), ModelCategoryColumns(), and KitListGui::refresh_category_list().

7.15.3.2 Gtk::TreeModelColumn<int> ModelCategoryColumns::m_col_num

Definition at line 65 of file kitlistgui.hpp.

Referenced by KitListGui::get_selected_category(), ModelCategoryColumns(), and KitListGui::refresh_category_list().

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

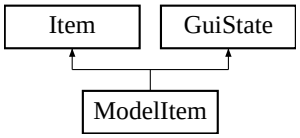
- kitlistgui.hpp

7.16 ModelItem Class Reference

Represents an Item combined with GuiState attributes.

`#include <kitmodel.hpp>`

Inheritance diagram for ModelItem::



Public Member Functions

- ModelItem ()
- virtual void set_checked (bool checked)

Friends

- class ModelItemCompareId

7.16.1 Detailed Description

Represents an Item combined with GuiState attributes.

Definition at line 60 of file kitmodel.hpp.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 ModelItem::ModelItem () `[inline]`

Definition at line 62 of file kitmodel.hpp.

7.16.3 Member Function Documentation

7.16.3.1 virtual void ModelItem::set_checked (bool *checked*) `[inline, virtual]`

Reimplemented from Item.

Definition at line 64 of file kitmodel.hpp.

References Item::set_checked(), and GuiState::set_dirty().

Referenced by KitParser::process_item(), and Service::update_item().

7.16.4 Friends And Related Function Documentation

7.16.4.1 friend class ModelItemCompareId [friend]

Definition at line 63 of file kitmodel.hpp.

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

- kitmodel.hpp

7.17 ModelItemColumns Class Reference

A definition for displaying an item in a multi-column list.

`#include <kitlistgui.hpp>`

Public Member Functions

- ModelItemColumns ()

Public Attributes

- Gtk::TreeModelColumn< Glib::ustring > m_col_text
- Gtk::TreeModelColumn< bool > m_col_checked
- Gtk::TreeModelColumn< int > m_col_num

7.17.1 Detailed Description

A definition for displaying an item in a multi-column list.

Definition at line 75 of file kitlistgui.hpp.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 ModelItemColumns::ModelItemColumns () [inline]

Definition at line 78 of file kitlistgui.hpp.

References m_col_checked, m_col_num, and m_col_text.

7.17.3 Member Data Documentation

7.17.3.1 Gtk::TreeModelColumn<Glib::ustring> ModelItemColumns::m_col_text

Definition at line 84 of file kitlistgui.hpp.

Referenced by KitListGui::init(), ModelItemColumns(), KitListGui::on_row_changed(), and KitListGui::refresh_item_list().

7.17.3.2 Gtk::TreeModelColumn<bool> ModelItemColumns::m_col_checked

Definition at line 85 of file kitlistgui.hpp.

Referenced by KitListGui::init(), ModelItemColumns(), KitListGui::on_row_changed(), and KitListGui::refresh_item_list().

7.17.3.3 Gtk::TreeModelColumn<int> ModelItemColumns::m_col_num

Definition at line 86 of file kitlistgui.hpp.

Referenced by KitListGui::delete_selected_items(), KitListGui::get_selected_items(), KitListGui::init(), ModelItemColumns(), KitListGui::on_row_changed(), KitListGui::refresh_item_list(), and KitListGui::selected_row_callback().

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

- kitlistgui.hpp

7.18 ModelItemCompareId Class Reference

Comparator for comparing items by their unique ID.

`#include <kitmodel.hpp>`

Public Member Functions

- ModelItemCompareId ()
- int operator() (ModelItem *i1, ModelItem *i2)

7.18.1 Detailed Description

Comparator for comparing items by their unique ID.

See also:

Item

Definition at line 72 of file kitmodel.hpp.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 ModelItemCompareId::ModelItemCompareId () [inline]

Definition at line 74 of file kitmodel.hpp.

7.18.3 Member Function Documentation

7.18.3.1 int ModelItemCompareId::operator() (ModelItem * *i1*, ModelItem * *i2*) [inline]

Definition at line 75 of file kitmodel.hpp.

References Item::get_id().

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

- kitmodel.hpp

7.19 Service Class Reference

Business/service layer implementation.

`#include <service.hpp>`

Public Member Functions

- Service (KitListDao &dao)

  Loads the data model from the persistence store.

- ~Service ()
- ModelItem * find_item (long id)
- ModelCategory * find_category (long cat_id)
- void copy_items (const ModelItemContainer &items, long cat_id)

  Copies items to the specified category.

- Item * create_item (long cat_id)

  Creates a new ModelItem and associates it with the specified category.

- bool delete_item (long id)

  Flags an item as deleted.

- bool delete_category (long cat_id)

  Flags a category as deleted.

- Category * create_category ()

  Creates a new category.

- bool is_model_dirty ()
- void set_model_dirty (bool flag=true)
- virtual bool filter (bool checked)

  Applies the current filter.

- void create_default_model ()

  Creates a default model.

- void open_as_xml (const Glib::ustring &filename)

  Loads a new data model from the named XML document.

- void save ()
- void save_as_xml (const Glib::ustring &filename)

  Saves the model's state to an XML document.

- bool update_item (long id, const std::string description, bool checked)

  Updates the attributes of an item.

- ItemContainer * get_items (long cat_id=-1)

  Returns a list of items.

- CategoryContainer * get_categories ()

  Returns a list of all categories.

- virtual void show_all ()

  Removes filter. All items are shown.

- virtual void show_checked_only ()

  Sets the filter to show only checked items.

- virtual void show_unchecked_only ()

  Sets the filter to show only unchecked items.

- virtual void select_items (ModelItemContainer *items, bool checked=true)

  Checks or unchecks all the passed items.

- virtual void toggle_selected_items (ModelItemContainer *items)

  Toggles the checked state of all the passed items.

- virtual bool require_filename ()

Protected Member Functions

- KitModel * load_model ()

  Loads the data model from the persistence store.

- long get_next_item_id ()
- long get_next_category_id ()

Protected Attributes

- KitListDao & m_dao

  Reference to the perisitence data access object.

- KitModel * m_model

  The application's data model.

7.19.1 Detailed Description

Business/service layer implementation.

Implements the service layer of the application, such that a different front-end can re-use the provided business methods.

Definition at line 38 of file service.hpp.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Service::Service (KitListDao & *dao*)

Loads the data model from the persistence store.

Definition at line 33 of file service.cpp.

References load_model(), and m_model.

7.19.2.2 Service::~Service ()

Definition at line 38 of file service.cpp.

References m_model.

7.19.3 Member Function Documentation

7.19.3.1 KitModel * Service::load_model () [protected]

Loads the data model from the persistence store.

Definition at line 47 of file service.cpp.

References KitListDao::get_model(), and m_dao.

Referenced by Service().

7.19.3.2 long Service::get_next_item_id () [protected]

Returns an unused unique id for an Item.

Definition at line 324 of file service.cpp.

References KitListDao::get_next_item_id(), and m_dao.

Referenced by create_item().

7.19.3.3 long Service::get_next_category_id () [protected]

Returns an unused unique id for a Category.

Definition at line 332 of file service.cpp.

References KitListDao::get_next_category_id(), and m_dao.

Referenced by create_category().

7.19.3.4 ModelItem * Service::find_item (long *id*)

Returns the item with the specified unique ID.

Definition at line 114 of file service.cpp.

References KitModel::find_item(), and m_model.

Referenced by KitListGui::get_selected_items(), and KitListGui::paste_from_xml().

7.19.3.5 ModelCategory * Service::find_category (long *cat_id*)

Returns the category with the specified unique ID.

Definition at line 122 of file service.cpp.

References KitModel::find_category(), and m_model.

Referenced by KitListGui::close_add_category_window(), KitListGui::on_menu_cut(), and KitListGui::on_menu_rename_category().

7.19.3.6 void Service::copy_items (const ModelItemContainer & *items*, long *cat_id*)

Copies items to the specified category.

Only copies those items that do not already exist in the target category.

Definition at line 133 of file service.cpp.

References KitModel::copy_items(), m_model, and KitModel::set_dirty().

Referenced by KitListGui::add_items().

7.19.3.7 Item * Service::create_item (long *cat_id*)

Creates a new ModelItem and associates it with the specified category.

A new item is created and assigned a new unique Id. The item is added to the data model and associated with a category if the cat_id is greater than or equal to zero. Otherwise the item is added to the model without being associated with a category.

Definition at line 148 of file service.cpp.

References KitModel::add_item(), get_next_item_id(), m_model, GuiState::set_dirty(), KitModel::set_dirty(), Item::set_id(), and GuiState::set_new_flag().

Referenced by KitListGui::close_add_item_window(), and KitListGui::on_menu_add().

7.19.3.8 bool Service::delete_item (long *id*)

Flags an item as deleted.

Finds the item with the specified ID and flags it as deleted.

Returns:

false if the item does not exist.

Definition at line 169 of file service.cpp.

References KitModel::find_item(), m_model, GuiState::set_deleted(), GuiState::set_dirty(), and KitModel::set_dirty().

Referenced by KitListGui::delete_selected_items(), and KitListGui::selected_row_callback().

7.19.3.9 bool Service::delete_category (long *cat_id*)

Flags a category as deleted.

Finds the category with the specified ID and flags it as deleted.

Returns:

false if the category does not exist.

Definition at line 189 of file service.cpp.

References KitModel::find_category(), m_model, GuiState::set_deleted(), GuiState::set_dirty(), and KitModel::set_dirty().

Referenced by KitListGui::on_menu_delete_category().

7.19.3.10 Category * Service::create_category ()

Creates a new category.

A new ModelCategory is created, assigned a new unique ID and added to the model.

Definition at line 209 of file service.cpp.

References KitModel::add_category(), get_next_category_id(), m_model, GuiState::set_dirty(), KitModel::set_dirty(), Category::set_id(), and GuiState::set_new_flag().

Referenced by KitListGui::close_add_category_window(), and KitListGui::on_menu_create_category().

7.19.3.11 bool Service::is_model_dirty () [inline]

Definition at line 55 of file service.hpp.

References KitModel::is_dirty(), and m_model.

Referenced by KitListGui::on_delete_event(), KitListGui::on_menu_file_new(), KitListGui::on_menu_file_open(), KitListGui::on_menu_quit(), KitListGui::on_menu_recent_file(), KitListGui::on_menu_save(), KitListGui::run(), and KitListGui::safe_open_file().

7.19.3.12 void Service::set_model_dirty (bool *flag* = true)

Flags the model as being dirty.

Definition at line 253 of file service.cpp.

References m_model, and KitModel::set_dirty().

Referenced by KitListGui::close_add_category_window(), KitListGui::confirm_lose_changes(), KitListGui::init(), KitListGui::on_cell_edit(), KitListGui::on_delete_event(), KitListGui::on_menu_create_category(), KitListGui::on_menu_cut(), KitListGui::on_menu_quit(), KitListGui::on_menu_rename_category(), and KitListGui::on_row_changed().

7.19.3.13 virtual bool Service::filter (bool *checked*) [inline, virtual]

Applies the current filter.

Parameters:

checked The checked/ticked state of the item being filtered.

Returns:

true if the item should be included.

Definition at line 63 of file service.hpp.

References KitModel::filter(), and m_model.

Referenced by KitListGui::refresh_item_list().

7.19.3.14 void Service::create_default_model ()

Creates a default model.

By default, a new empty XmlDao data model is created.

Definition at line 72 of file service.cpp.

References KitListDao::get_model(), m_dao, m_model, and KitModel::reset().

Referenced by KitListGui::on_menu_file_new().

7.19.3.15 void Service::open_as_xml (const Glib::ustring & *filename*)

Loads a new data model from the named XML document.

Creates a new data model based on the passed filename. The existing data model is destroyed after successfully loading the new one.

Definition at line 59 of file service.cpp.

References m_dao, and m_model.

Referenced by KitListGui::init(), KitListGui::on_menu_recent_file(), and KitListGui::open_file().

7.19.3.16 void Service::save ()

Saves the model's state to the persistence store.

Definition at line 87 of file service.cpp.

References m_dao, m_model, and KitListDao::save_model().

Referenced by KitListGui::confirm_lose_changes(), and KitListGui::on_menu_save().

7.19.3.17 void Service::save_as_xml (const Glib::ustring & *filename*)

Saves the model's state to an XML document.

This is primarily intended to support switching from one dao implementation to the XmlDao implemenation.

Parameters:

filename The full pathname and filename to save the file to.

Definition at line 100 of file service.cpp.

References m_dao, m_model, and XmlDao::save_model().

Referenced by KitListGui::confirm_lose_changes(), KitListGui::on_menu_save(), KitListGui::on_menu_save_as(), and KitListGui::run().

7.19.3.18 bool Service::update_item (long *id*, const std::string *description*, bool *checked*)

Updates the attributes of an item.

Locates the item using the supplied unique ID, then assigns the passed values.

Parameters:

id The unique ID of the item to update.

description The item's descriptive text.

checked The checked/ticked state of the item.

Returns:

Returns true if the item exists, false otherwise.

Definition at line 271 of file service.cpp.

References KitModel::find_item(), m_model, ModelItem::set_checked(), Item::set_description(), and GuiState::set_dirty().

Referenced by KitListGui::on_row_changed().

7.19.3.19 ItemContainer * Service::get_items (long *cat_id* = -1)

Returns a list of items.

If cat_id is less than zero, returns all items, otherwise only those items associated with the specified category ID.

Parameters:

cat_id the unique ID of a category or '-1' to indicate all items are to be retrieved.

Definition at line 294 of file service.cpp.

References KitModel::find_category(), KitModel::get_all_items(), ModelCategory::get_items(), and m_model.

Referenced by KitListGui::init(), and KitListGui::refresh_item_list().

7.19.3.20 CategoryContainer * Service::get_categories ()

Returns a list of all categories.

Categories flagged as deleted are excluded.

Definition at line 314 of file service.cpp.

References KitModel::get_categories(), and m_model.

Referenced by KitListGui::refresh_category_list().

7.19.3.21 virtual void Service::show_all () [inline, virtual]

Removes filter. All items are shown.

Definition at line 72 of file service.hpp.

References m_model, and KitModel::show_all().

Referenced by KitListGui::on_menu_show_all().

7.19.3.22 virtual void Service::show_checked_only () [inline, virtual]

Sets the filter to show only checked items.

Definition at line 74 of file service.hpp.

References m_model, and KitModel::show_checked_only().

Referenced by KitListGui::on_menu_show_checked().

7.19.3.23 virtual void Service::show_unchecked_only () [inline, virtual]

Sets the filter to show only unchecked items.

Definition at line 76 of file service.hpp.

References m_model, and KitModel::show_unchecked_only().

Referenced by KitListGui::on_menu_show_unchecked().

7.19.3.24 void Service::select_items (ModelItemContainer * *items*, bool *checked* = true) [virtual]

Checks or unchecks all the passed items.

Parameters:

items The items to change. the state to change the to.

Definition at line 227 of file service.cpp.

References m_model, and KitModel::set_dirty().

Referenced by KitListGui::set_selected().

7.19.3.25 void Service::toggle_selected_items (ModelItemContainer * *items*) [virtual]

Toggles the checked state of all the passed items.

Parameters:

items The items to change.

Definition at line 241 of file service.cpp.

References m_model, and KitModel::set_dirty().

Referenced by KitListGui::toggle_selected().

7.19.3.26 virtual bool Service::require_filename () [inline, virtual]

Definition at line 79 of file service.hpp.

References m_dao, and KitListDao::require_filename().

Referenced by KitListGui::init(), KitListGui::on_menu_file_new(), and KitListGui::on_menu_save().

7.19.4 Member Data Documentation

7.19.4.1 KitListDao& Service::m_dao [protected]

Reference to the perisitence data access object.

Definition at line 40 of file service.hpp.

Referenced by create_default_model(), get_next_category_id(), get_next_item_id(), load_model(), open_as_xml(), require_filename(), save(), and save_as_xml().

7.19.4.2 KitModel* Service::m_model [protected]

The application's data model.

Definition at line 41 of file service.hpp.

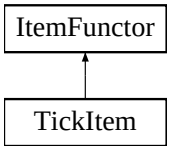
Referenced by copy_items(), create_category(), create_default_model(), create_item(), delete_category(), delete_item(), filter(), find_category(), find_item(), get_categories(), get_items(), is_model_dirty(), open_as_xml(), save(), save_as_xml(), select_items(), Service(), set_model_dirty(), show_all(), show_checked_only(), show_unchecked_only(), toggle_selected_items(), update_item(), and ~Service().

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

- service.hpp
- service.cpp

7.20 TickItem Class Reference

Inheritance diagram for TickItem::



Public Member Functions

- TickItem (ItemContainer &changed)
- bool operator() (Item &item)

Private Attributes

- ItemContainer & m_changed

7.20.1 Detailed Description

Definition at line 119 of file main.cpp.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 TickItem::TickItem (ItemContainer & *changed*) [inline]

Definition at line 122 of file main.cpp.

7.20.3 Member Function Documentation

7.20.3.1 bool TickItem::operator() (Item & *item*) [inline, virtual]

Implements ItemFunctor.

Definition at line 123 of file main.cpp.

References Item::get_checked(), Item::get_description(), Item::get_id(), m_changed, and Item::set_checked().

7.20.4 Member Data Documentation

7.20.4.1 ItemContainer& TickItem::m_changed [private]

Definition at line 120 of file main.cpp.

Referenced by operator()().

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

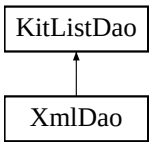
- main.cpp

7.21 XmlDao Class Reference

Implementation of a KitListDao using XML as the persistence store.

`#include <xmldao.hpp>`

Inheritance diagram for XmlDao::



Public Member Functions

- XmlDao (int verbose=0)
- KitModel * get_model ()

  Loads the data model from the previously set filename.

- KitModel * get_model (Glib::ustring filename)
- void save_model (KitModel *model)

  Saves the model as an XML document.

- void save_model (KitModel *model, Glib::ustring filename)

  Saves the model as an XML document.

- Category * get_category (long cat_id, item_choice choice)

  Loads a category.

- ItemContainer * get_all_items (item_choice choice)

  Returns a list of all items.

- long add_item (const std::string name)
- long add_item (const std::string name, long cat_id)
- void append_items_to_category (long to_cat_id, long from_cat_id, item_choice choice)

  Copies items from one category to another.

- void associate_item_with_category (long id, long cat_id)

  Associates an existing item with an existing category.

- CategoryContainer get_categories ()
- long new_category (const std::string name)

  Creates a new category.

- void delete_item (long id)
- void update_item_checked_state (ItemContainer &items)

  Persists the state of the 'checked' flag of each item.

- void remove_item_from_category (long id, long cat_id)

- long get_next_item_id ()

  Returns the next unused unique id for items.

- long get_next_category_id ()
- void delete_category (long id)
- void set_item_flag (long id)
- void unset_item_flag (long id)
- void set_category_flag (long id)
- void unset_category_flag (long id)
- void set_all_flags ()
- void unset_all_flags ()
- void set_filename (Glib::ustring filename)
- virtual bool require_filename ()

  Indicates that this implementation requires a filename.

Protected Attributes

- Glib::ustring m_filename

  The filename to load or save the XML document from.

- xmlpp::Element * m_items_node

  Temporary reference to the items' node.

- xmlpp::Element * m_categories_node

  Temporary reference to the categories' node.

- xmlpp::Element * m_cat_items_node

  Temporary reference to the categories' items' node.

- long m_max_item_id

  The last used ID for items.

- long m_max_category_id

  The last used ID for categories.

Private Member Functions

- bool add_item_to_dom (ModelItem &item)

  Adds the passed item to the current items' node.

- bool add_category_item_to_dom (Item &item)

  Adds the passed item to the current category's node.

- bool add_category_to_dom (ModelCategory &item)

  Adds the passed item to the current categories' node.

7.21.1 Detailed Description

Implementation of a KitListDao using XML as the persistence store.

Definition at line 42 of file xmldao.hpp.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 XmlDao::XmlDao (int *verbose* = 0) [inline]

Definition at line 67 of file xmldao.hpp.

7.21.3 Member Function Documentation

7.21.3.1 bool XmlDao::add_item_to_dom (ModelItem & *item*) [private]

Adds the passed item to the current items' node.

See also:

XmlDao::save_model(KitModel&)

Definition at line 87 of file xmldao.cpp.

References Item::get_checked(), Item::get_description(), Item::get_id(), GuiState::is_deleted(), and m_items_node.

Referenced by save_model().

7.21.3.2 bool XmlDao::add_category_item_to_dom (Item & *item*) [private]

Adds the passed item to the current category's node.

See also:

XmlDao::save_model(KitModel&)

Definition at line 106 of file xmldao.cpp.

References Item::get_id(), and m_cat_items_node.

Referenced by add_category_to_dom().

7.21.3.3 bool XmlDao::add_category_to_dom (ModelCategory & *category*) [private]

Adds the passed item to the current categories' node.

See also:

XmlDao::save_model(KitModel&)

Definition at line 123 of file xmldao.cpp.

References add_category_item_to_dom(), Category::foreach_item(), Category::get_id(), Category::get_name(), GuiState::is_deleted(), m_cat_items_node, and m_categories_node.

Referenced by save_model().

7.21.3.4 KitModel * XmlDao::get_model () [virtual]

Loads the data model from the previously set filename.

The data model holds a rich graph of objects, representing the entire list of categories and items.

The filename must be set before calling this method by calling XmlDao::set_filename(), otherwise an empty model is returned.

Return values:

Returns a newly created data model. The caller is responsible for destroying the model.

See also:

KitModel

Implements KitListDao.

Definition at line 46 of file xmldao.cpp.

References KitModel::get_all_items(), KitModel::get_categories(), Category::get_id(), Item::get_id(), m_filename, m_max_category_id, m_max_item_id, and KitModel::reset().

Referenced by get_model().

7.21.3.5 KitModel* XmlDao::get_model (Glib::ustring *filename*) [inline]

Definition at line 76 of file xmldao.hpp.

References get_model(), and set_filename().

7.21.3.6 void XmlDao::save_model (KitModel * *model*) [virtual]

Saves the model as an XML document.

The filename must be set before calling this method by calling XmlDao::set_filename();

Parameters:

model The model to save.

Implements KitListDao.

Definition at line 163 of file xmldao.cpp.

References add_category_to_dom(), add_item_to_dom(), KitModel::foreach_category(), KitModel::foreach_item(), m_categories_node, m_filename, m_items_node, KitModel::purge(), and KitModel::reset().

Referenced by Service::save_as_xml(), and save_model().

7.21.3.7 void XmlDao::save_model (KitModel * *model*, Glib::ustring *filename*) [inline]

Saves the model as an XML document.

Parameters:

model The model to save.

filename The name of the file to save to.

Definition at line 86 of file xmldao.hpp.

References save_model(), and set_filename().

7.21.3.8 Category* XmlDao::get_category (long *cat_id*, item_choice *choice*) [inline, virtual]

Loads a category.

Parameters:

choice Which items to load for the category. One of ALL_ITEMS, CHECKED_ITEMS or UNCHECKED_ITEMS.

Implements KitListDao.

Definition at line 88 of file xmldao.hpp.

References NYI.

7.21.3.9 ItemContainer* XmlDao::get_all_items (item_choice *choice*) [inline, virtual]

Returns a list of all items.

Parameters:

choice Which items to load. One of ALL_ITEMS, CHECKED_ITEMS or UNCHECKED_ITEMS.

Implements KitListDao.

Definition at line 90 of file xmldao.hpp.

References NYI.

7.21.3.10 long XmlDao::add_item (const std::string *name*) [inline, virtual]

Creates a new item.

Parameters:

name The name of the new item.

Implements KitListDao.

Definition at line 92 of file xmldao.hpp.

References NYI.

7.21.3.11 long XmlDao::add_item (const std::string *name*, long *cat_id*) [inline, virtual]

Creates a new item and associates it with a category.

Parameters:

name The name of the new item.

Implements KitListDao.

Definition at line 94 of file xmldao.hpp.

References NYI.

7.21.3.12 void XmlDao::append_items_to_category (long *to_cat_id*, long *from_cat_id*, item_choice *choice*) [inline, virtual]

Copies items from one category to another.

Optionally, only checked or unchecked items are copied.

Parameters:

from_cat_id The ID of the source category.

to_cat_id The ID of the target category.

choice One of ALL_ITEMS, CHECKED_ITEMS or UNCHECKED_ITEMS.

Implements KitListDao.

Definition at line 96 of file xmldao.hpp.

References NYI.

7.21.3.13 void XmlDao::associate_item_with_category (long *id*, long *cat_id*) [inline, virtual]

Associates an existing item with an existing category.

Parameters:

id The Item ID

cat_id The Category ID

Implements KitListDao.

Definition at line 98 of file xmldao.hpp.

References NYI.

7.21.3.14 CategoryContainer XmlDao::get_categories () [inline, virtual]

Returns a list of all categories.

Implements KitListDao.

Definition at line 100 of file xmldao.hpp.

References NYI.

7.21.3.15 long XmlDao::new_category (const std::string *name*) [inline, virtual]

Creates a new category.

Parameters:

name the name of the new category.

Implements KitListDao.

Definition at line 102 of file xmldao.hpp.

References NYI.

7.21.3.16 void XmlDao::delete_item (long *id*) [inline, virtual]

Deletes an item by it's ID.

Parameters:

id the ID of the item to delete.

Implements KitListDao.

Definition at line 104 of file xmldao.hpp.

References NYI.

7.21.3.17 void XmlDao::update_item_checked_state (ItemContainer & *items*) [inline, virtual]

Persists the state of the 'checked' flag of each item.

Implements KitListDao.

Definition at line 106 of file xmldao.hpp.

References NYI.

7.21.3.18 void XmlDao::remove_item_from_category (long *id*, long *cat_id*) [inline, virtual]

Un-associates the specified item from the specified category.

Parameters:

id The Item id.
cat_id The Category id.

Implements KitListDao.

Definition at line 108 of file xmldao.hpp.

References NYI.

7.21.3.19 long XmlDao::get_next_item_id () [virtual]

Returns the next unused unique id for items.

Implements KitListDao.

Definition at line 142 of file xmldao.cpp.

References m_max_item_id.

7.21.3.20 long XmlDao::get_next_category_id () [virtual]

Returns the next unused unique id for categories.

Implements KitListDao.

Definition at line 150 of file xmldao.cpp.

References m_max_category_id.

7.21.3.21 void XmlDao::delete_category (long *id*) [inline, virtual]

Deletes a category.

Implements KitListDao.

Definition at line 114 of file xmldao.hpp.

References NYI.

7.21.3.22 void XmlDao::set_item_flag (long *id*) [inline, virtual]

Sets the checked flag for an item.

Parameters:

id The id of the item to change.

Implements KitListDao.

Definition at line 116 of file xmldao.hpp.

References NYI.

7.21.3.23 void XmlDao::unset_item_flag (long *id*) [inline, virtual]

Clears the checked flag for an item.

Implements KitListDao.

Definition at line 118 of file xmldao.hpp.

References NYI.

7.21.3.24 void XmlDao::set_category_flag (long *id*) [inline, virtual]

Checks/ticks all items in the specified Category.

Implements KitListDao.

Definition at line 120 of file xmldao.hpp.

References NYI.

7.21.3.25 void XmlDao::unset_category_flag (long *id*) [inline, virtual]

Unchecks/unticks all items in the specified Category.

Implements KitListDao.

Definition at line 122 of file xmldao.hpp.

References NYI.

7.21.3.26 void XmlDao::set_all_flags () [inline, virtual]

Checks/ticks all items.

Implements KitListDao.

Definition at line 124 of file xmldao.hpp.

References NYI.

7.21.3.27 void XmlDao::unset_all_flags () [inline, virtual]

Unchecks/unticks all items.

Implements KitListDao.

Definition at line 126 of file xmldao.hpp.

References NYI.

7.21.3.28 void XmlDao::set_filename (Glib::ustring *filename*) [inline]

Definition at line 128 of file xmldao.hpp.

References m_filename.

Referenced by get_model(), and save_model().

7.21.3.29 virtual bool XmlDao::require_filename () [inline, virtual]

Indicates that this implementation requires a filename.

This persistence model requires a filename to be chosen before the data can be persisted.

Returns:

Always returns true.

Reimplemented from KitListDao.

Definition at line 138 of file xmldao.hpp.

7.21.4 Member Data Documentation

7.21.4.1 Glib::ustring XmlDao::m_filename [protected]

The filename to load or save the XML document from.

Definition at line 54 of file xmldao.hpp.

Referenced by get_model(), save_model(), and set_filename().

7.21.4.2 xmlpp::Element* XmlDao::m_items_node [protected]

Temporary reference to the items' node.

Definition at line 56 of file xmldao.hpp.

Referenced by add_item_to_dom(), and save_model().

7.21.4.3 xmlpp::Element* XmlDao::m_categories_node [protected]

Temporary reference to the categories' node.

Definition at line 58 of file xmldao.hpp.

Referenced by add_category_to_dom(), and save_model().

7.21.4.4 xmlpp::Element* XmlDao::m_cat_items_node [protected]

Temporary reference to the categories' items' node.

Definition at line 60 of file xmldao.hpp.

Referenced by add_category_item_to_dom(), and add_category_to_dom().

7.21.4.5 long XmlDao::m_max_item_id [protected]

The last used ID for items.

Definition at line 62 of file xmldao.hpp.

Referenced by get_model(), and get_next_item_id().

7.21.4.6 long XmlDao::m_max_category_id [protected]

The last used ID for categories.

Definition at line 64 of file xmldao.hpp.

Referenced by get_model(), and get_next_category_id().

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

- xmldao.hpp
- xmldao.cpp

Chapter 8

File Documentation

8.1 category.cpp File Reference

```
#include <algorithm>
#include "category.hpp"
```

8.2 category.hpp File Reference

```
#include <iostream>
#include <vector>
#include "item.hpp"
```

Classes

- class Category

  Represents a Category.

- class CategoryCompareName

  Comparator used for sorting Categories by name.

- class CategoryCompareId

  Comparator used for comparing Categories by id.

Defines

- #define CATEGORY_H 1

Typedefs

- typedef std::vector< Category * > CategoryContainer
- typedef CategoryContainer::iterator CategoryIter

8.2.1 Define Documentation

8.2.1.1 #define CATEGORY_H 1

Definition at line 24 of file category.hpp.

8.2.2 Typedef Documentation

8.2.2.1 typedef std::vector<Category*> CategoryContainer

Definition at line 84 of file category.hpp.

8.2.2.2 typedef CategoryContainer::iterator CategoryIter

Definition at line 85 of file category.hpp.

8.3 item.hpp File Reference

```
#include <iostream>
#include <list>
#include <string>
#include <vector>
#include <sigc++/signal.h>
```

Classes

- class Item

  Represents an Item.

- class ItemCompareName

  Comparator used for sorting Items by name.

- class ItemCompareId

  Comparator used for comparing Items by id.

- class ItemFunctor

  Functor for processing items.

Defines

- #define ITEM_H 1

Typedefs

- typedef std::vector< Item * > ItemContainer
- typedef ItemContainer::iterator ItemIter
- typedef std::list< Item > ItemList
- typedef ItemList::iterator ItemListIter
- typedef sigc::slot< bool, Item & > SlotForeachItem

8.3.1 Define Documentation

8.3.1.1 #define ITEM_H 1

Definition at line 24 of file item.hpp.

8.3.2 Typedef Documentation

8.3.2.1 typedef std::vector<Item*> ItemContainer

Definition at line 91 of file item.hpp.

8.3.2.2 typedef ItemContainer::iterator ItemIter

Definition at line 92 of file item.hpp.

8.3.2.3 typedef std::list<Item> ItemList

Definition at line 94 of file item.hpp.

8.3.2.4 typedef ItemList::iterator ItemListIter

Definition at line 95 of file item.hpp.

8.3.2.5 typedef sigc::slot<bool, Item&> SlotForeachItem

Definition at line 97 of file item.hpp.

8.4 kitlistdao.hpp File Reference

```
#include "category.hpp"
#include "item.hpp"
#include "kitmodel.hpp"
#include <string>
```

Classes

- class KitListDao

  Defines the methods that an implementation of this class must implement.

Defines

- #define KIT_LIST_DAO_H 1

Enumerations

- enum item_choice { ALL_ITEMS, CHECKED_ITEMS, UNCHECKED_ITEMS }

8.4.1 Define Documentation

8.4.1.1 #define KIT_LIST_DAO_H 1

Definition at line 24 of file kitlistdao.hpp.

8.4.2 Enumeration Type Documentation

8.4.2.1 enum item_choice

Options for selecting all items, only checked items, or only unchecked items.

Enumerator:

ALL_ITEMS

CHECKED_ITEMS

UNCHECKED_ITEMS

Definition at line 36 of file kitlistdao.hpp.

8.5 kitlistgui.cpp File Reference

```
#include "kitlistgui.hpp"
#include <cassert>
#include <glibmm/i18n.h>
#include <glibmm/refptr.h>
#include <glibmm/ustring.h>
#include <gtkmm/aboutdialog.h>
#include <gtkmm/cellrenderertoggle.h>
#include <gtkmm/clipboard.h>
#include <gtkmm/filechooserdialog.h>
#include <gtkmm/menuitem.h>
#include <gtkmm/messagedialog.h>
#include <gtkmm/stock.h>
#include <gtkmm/targetentry.h>
#include <gtkmm/window.h>
#include <libglademm/xml.h>
#include <libxml++/libxml++.h>
#include <string>
#include <sstream>
#include <sys/stat.h>
#include <config.h>
```

Namespaces

- namespace anonymous_namespace{kitlistgui.cpp}

Defines

- #define KITLIST_SERVICE_NAME "com.nokia."PACKAGE_NAME
  Maemo service name.
- #define KITLIST_SERVICE_OBJECT "/com/nokia/"PACKAGE_NAME
  Maemo service path.
- #define KITLIST_SERVICE_IFACE "com.nokia."PACKAGE_NAME
  Maemo service interface name.

Typedefs

- typedef Gtk::TreeModel::Children type_children

Functions

- const bool file_exists (const Glib::ustring &filename)

  Returns true if the passed file exists.

- const string load_resource_glade_file (const Glib::ustring &filename)

  Locate the Glade resource file from a list of potential locations.

- Glib::RefPtr< Gnome::Glade::Xml > get_glade_ref_ptr (const string &filename, const Glib::ustring &root=Glib::ustring(), const Glib::ustring &domain=Glib::ustring())

  Returns a reference to the Glade resource file.

Variables

- const string anonymous_namespace{kitlistgui.cpp}::GLADE_APP_FILE = "kitlist.glade"

  Resource file name.

- const guint anonymous_namespace{kitlistgui.cpp}::SB_ITEM_COUNT = 1000

  Status bar message constant for displaying item counts.

- const guint anonymous_namespace{kitlistgui.cpp}::SB_SAVE = SB_ITEM_COUNT + 1

  Status bar message constant for save notifications.

- const guint anonymous_namespace{kitlistgui.cpp}::SB_MSG = SB_SAVE + 1

  Status bara message constant for general messages.

- const char anonymous_namespace{kitlistgui.cpp}::item_target_custom [ ] = "kitlistclipboard"

  Key used for custom clipboard.

- const char anonymous_namespace{kitlistgui.cpp}::item_target_text [ ] = "text/plain"

  Mime type for clipboard content.

- const char anonymous_namespace{kitlistgui.cpp}::XML_ELEMENT_ID [ ] = "id"

  Tag name for the ID element in the clipbard XML document.

- const Glib::ustring anonymous_namespace{kitlistgui.cpp}::DEFAULT_FILENAME_EXTENSION = ".kit"

  Default filename extension.

- const Glib::ustring anonymous_namespace{kitlistgui.cpp}::DEFAULT_FILENAME = "kitlist" + DEFAULT_FILENAME_EXTENSION

  The default filename.

- const Glib::ustring anonymous_namespace{kitlistgui.cpp}::GCONF_KEY = "/apps/kitlist"

  The application's root key in the GConf hierarchy.

- const Glib::ustring anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_CURRENT_FILENAME = GCONF_KEY + "/current_filename"

  GConf entry for the current filename.

- const gint anonymous_namespace{kitlistgui.cpp}::DEFAULT_MAX_RECENT_FILES = 4

  The maximum number of recent files to maintain.

- const Glib::ustring anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_RECENT_FILES = GCONF_KEY + "/recent_files"

  GConf entry for recent files.

- const Glib::ustring anonymous_namespace{kitlistgui.cpp}::GCONF_KEY_MAX_RECENT_-FILES = GCONF_KEY + "/max_recent_files"

  GConf entry for max recent files.

8.5.1 Define Documentation

8.5.1.1 #define KITLIST_SERVICE_IFACE "com.nokia."PACKAGE_NAME

Maemo service interface name.

Definition at line 54 of file kitlistgui.cpp.

Referenced by KitListGui::init().

8.5.1.2 #define KITLIST_SERVICE_NAME "com.nokia."PACKAGE_NAME

Maemo service name.

Definition at line 50 of file kitlistgui.cpp.

Referenced by KitListGui::init().

8.5.1.3 #define KITLIST_SERVICE_OBJECT "/com/nokia/"PACKAGE_NAME

Maemo service path.

Definition at line 52 of file kitlistgui.cpp.

Referenced by KitListGui::init().

8.5.2 Typedef Documentation

8.5.2.1 typedef Gtk::TreeModel::Children type_children

Definition at line 129 of file kitlistgui.cpp.

8.5.3 Function Documentation

8.5.3.1 const bool file_exists (const Glib::ustring & *filename*)

Returns true if the passed file exists.

Note: We do not test for the file type, just it's existence regardless of whether it's a directory etc.

Parameters:

filename The file to test for existence.

Returns:

true if the file exists.

Definition at line 140 of file kitlistgui.cpp.

Referenced by KitListGui::choose_filename(), KitListGui::init(), and load_resource_glade_file().

8.5.3.2 Glib::RefPtr<Gnome::Glade::Xml> get_glade_ref_ptr (const string & *filename*, const Glib::ustring & *root* = `Glib::ustring()`, const Glib::ustring & *domain* = `Glib::ustring()`)

Returns a reference to the Glade resource file.

Attempts to locate the external Glade resource file from a number of common locations.

Definition at line 169 of file kitlistgui.cpp.

References load_resource_glade_file().

Referenced by KitListGui::init().

8.5.3.3 const string load_resource_glade_file (const Glib::ustring & *filename*)

Locate the Glade resource file from a list of potential locations.

Definition at line 150 of file kitlistgui.cpp.

References file_exists().

Referenced by get_glade_ref_ptr().

8.6 kitlistgui.hpp File Reference

```
#include "service.hpp"
#include <iostream>
#include <gtkmm/button.h>
#include <gtkmm/checkbutton.h>
#include <gtkmm/combobox.h>
#include <gtkmm/entry.h>
#include <gtkmm/imagemenuitem.h>
#include <gtkmm/main.h>
#include <gtkmm/statusbar.h>
#include <gtkmm/toolbutton.h>
#include <gtkmm/treemodelcolumn.h>
#include <gtkmm/liststore.h>
```

Classes

- class ModelCategoryColumns

  A definition for displaying a ModelCategory in a combo box.

- class ModelItemColumns

  A definition for displaying an item in a multi-column list.

- class KitListGui

  Encapsulates the methods for the application's GUI front end.

Defines

- #define KIT_LIST_GUI_H 1

Enumerations

- enum gui_state { ADD_CATEGORY, RENAME_CATEGORY }

Variables

- const int CHECKED_COL_POSITION = 0

  The position in the list of the tick box column.

8.6.1 Define Documentation

8.6.1.1 #define KIT_LIST_GUI_H 1

Definition at line 24 of file kitlistgui.hpp.

8.6.2 Enumeration Type Documentation

8.6.2.1 enum gui_state

Enumerator:

ADD_CATEGORY

RENAME_CATEGORY

Definition at line 51 of file kitlistgui.hpp.

8.6.3 Variable Documentation

8.6.3.1 const int CHECKED_COL_POSITION = 0

The position in the list of the tick box column.

Definition at line 70 of file kitlistgui.hpp.

8.7 kitlistpgsqldao.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include "kitlistpgsqldao.hpp"
```

8.8 kitlistpgsqldao.hpp File Reference

```
#include <config.h>
```

Defines

- #define KIT_LIST_PGSQL_DAO_H 1

8.8.1 Define Documentation

8.8.1.1 #define KIT_LIST_PGSQL_DAO_H 1

Definition at line 24 of file kitlistpgsqldao.hpp.

8.9 kitmodel.cpp File Reference

```
#include "kitmodel.hpp"
#include <algorithm>
#include <cassert>
#include <glib.h>
#include <iostream>
#include <iterator>
```

8.10 kitmodel.hpp File Reference

```
#include "item.hpp"
#include "category.hpp"
#include <map>
#include <sigc++/signal.h>
```

Classes

- class GuiState

  Class encapsulating state of an object in the data model.

- class ModelItem

  Represents an Item combined with GuiState attributes.

- class ModelItemCompareId

  Comparator for comparing items by their unique ID.

- class ModelCategory

  Represents a Category combined with GuiState attributes.

- class KitModel

  Holds a rich graph of objects representing the application's data model.

Defines

- #define KIT_MODEL_H 1

Typedefs

- typedef std::vector< ModelItem * > ModelItemContainer
- typedef ModelItemContainer::iterator ModelItemIter
- typedef std::map< long, ModelItem * > ItemMap
- typedef ItemMap::iterator ItemMapIter
- typedef sigc::slot< bool, const ItemMap::iterator & > SlotForeachModelItemIter
- typedef sigc::slot< bool, ModelItem & > SlotForeachModelItem
- typedef std::vector< ModelCategory * > ModelCategoryContainer
- typedef ModelCategoryContainer::iterator ModelCategoryIter
- typedef std::map< long, ModelCategory * > CategoryMap
- typedef CategoryMap::iterator CategoryMapIter
- typedef sigc::slot< bool, const CategoryMap::iterator & > SlotForeachCategoryIter
- typedef sigc::slot< bool, ModelCategory & > SlotForeachCategory

Enumerations

- enum item_filter_types { ALL, CHECKED, UNCHECKED }

8.10.1 Define Documentation

8.10.1.1 #define KIT_MODEL_H 1

Definition at line 24 of file kitmodel.hpp.

8.10.2 Typedef Documentation

8.10.2.1 typedef std::map<long, ModelCategory*> CategoryMap

Definition at line 121 of file kitmodel.hpp.

8.10.2.2 typedef CategoryMap::iterator CategoryMapIter

Definition at line 122 of file kitmodel.hpp.

8.10.2.3 typedef std::map<long, ModelItem*> ItemMap

Definition at line 84 of file kitmodel.hpp.

8.10.2.4 typedef ItemMap::iterator ItemMapIter

Definition at line 85 of file kitmodel.hpp.

8.10.2.5 typedef std::vector<ModelCategory*> ModelCategoryContainer

Definition at line 118 of file kitmodel.hpp.

8.10.2.6 typedef ModelCategoryContainer::iterator ModelCategoryIter

Definition at line 119 of file kitmodel.hpp.

8.10.2.7 typedef std::vector<ModelItem*> ModelItemContainer

Definition at line 81 of file kitmodel.hpp.

8.10.2.8 typedef ModelItemContainer::iterator ModelItemIter

Definition at line 82 of file kitmodel.hpp.

8.10.2.9 typedef sigc::slot<bool, ModelCategory&> SlotForeachCategory

Definition at line 125 of file kitmodel.hpp.

8.10.2.10 typedef sigc::slot<bool, const CategoryMap::iterator&> SlotForeachCategoryIter

Definition at line 124 of file kitmodel.hpp.

8.10.2.11 typedef sigc::slot<bool, ModelItem&> SlotForeachModelItem

Definition at line 88 of file kitmodel.hpp.

8.10.2.12 typedef sigc::slot<bool, const ItemMap::iterator&> SlotForeachModelItemIter

Definition at line 87 of file kitmodel.hpp.

8.10.3 Enumeration Type Documentation

8.10.3.1 enum item_filter_types

Enumerator:

ALL

CHECKED

UNCHECKED

Definition at line 127 of file kitmodel.hpp.

8.11 kitparser.cpp File Reference

```
#include "kitparser.hpp"
#include <algorithm>
#include <iostream>
```

8.12 kitparser.hpp File Reference

```
#include <libxml++/libxml++.h>
#include "kitmodel.hpp"
```

Classes

- class KitParser

  SaxParser implementation for reading the KitModel from an XML document.

Defines

- #define KIT_PARSER_H 1

8.12.1 Define Documentation

8.12.1.1 #define KIT_PARSER_H 1

Definition at line 24 of file kitparser.hpp.

8.13 main.cpp File Reference

```
#include <config.h>
#include <locale.h>
#include <iostream>
#include <getopt.h>
#include <libintl.h>
#include "kitlistgui.hpp"
#include "kitlistpgsqldao.hpp"
#include "xmldao.hpp"
```

Classes

- class KitList

  Main application class.

- class TickItem

Functions

- int main (int argc, char **argv)

Variables

- static int verbose_flag = 0

  Flag set by '–verbose'.

- static int html_flag = 0

  Flag set by '–html'.

8.13.1 Function Documentation

8.13.1.1 int main (int *argc*, char ** *argv*)

Parses the command line and executes the main application.

Definition at line 564 of file main.cpp.

References KitList::add_item(), ALL_ITEMS, KitList::append_items_to_category(), KitList::associate_item_with_category(), CHECKED_ITEMS, KitList::delete_category(), KitList::delete_item(), KitList::get_dao(), html_flag, KitList::list_categories(), KitList::list_items(), KitList::new_category(), KitList::remove_item_from_category(), KitList::set_all_flags(), KitList::set_category_flag(), KitList::set_item_flag(), KitList::tick_items(), UNCHECKED_ITEMS, KitList::unset_all_flags(), KitList::unset_category_flag(), KitList::unset_item_flag(), and verbose_flag.

8.13.2 Variable Documentation

8.13.2.1 int html_flag = 0 [static]

Flag set by '–html'.

Definition at line 74 of file main.cpp.

Referenced by KitList::list_categories(), KitList::list_item(), KitList::list_items_end(), KitList::list_items_start(), and main().

8.13.2.2 int verbose_flag = 0 [static]

Flag set by '–verbose'.

Definition at line 71 of file main.cpp.

Referenced by main().

8.14 service.cpp File Reference

```
#include "service.hpp"
#include <cassert>
#include <iostream>
```

8.15 service.hpp File Reference

```
#include "kitlistdao.hpp"
#include "kitmodel.hpp"
#include "xmldao.hpp"
#include <glibmm/ustring.h>
```

Classes

- class Service

  Business/service layer implementation.

Defines

- #define SERVICE_H 1

8.15.1 Define Documentation

8.15.1.1 #define SERVICE_H 1

Definition at line 24 of file service.hpp.

8.16 xmldao.cpp File Reference

```
#include <algorithm>
#include <iostream>
#include <sstream>
#include "xmldao.hpp"
#include "kitparser.hpp"
```

8.17 xmldao.hpp File Reference

```
#include <cassert>
#include <libxml++/libxml++.h>
#include "kitlistdao.hpp"
#include "kitmodel.hpp"
```

Classes

- class XmlDao

  Implementation of a KitListDao using XML as the persistence store.

Defines

- #define XMLDAO_H 1
- #define NYI assert(false == "Method not implemented")

8.17.1 Define Documentation

8.17.1.1 #define NYI assert(false == "Method not implemented")

Definition at line 35 of file xmldao.hpp.

Referenced by XmlDao::add_item(), XmlDao::append_items_to_category(), XmlDao::associate_item_with_category(), XmlDao::delete_category(), XmlDao::delete_item(), XmlDao::get_all_items(), XmlDao::get_categories(), XmlDao::get_category(), XmlDao::new_category(), XmlDao::remove_item_from_category(), XmlDao::set_all_flags(), XmlDao::set_category_flag(), XmlDao::set_item_flag(), XmlDao::unset_all_flags(), XmlDao::unset_category_flag(), XmlDao::unset_item_flag(), and XmlDao::update_item_checked_state().

8.17.1.2 #define XMLDAO_H 1

Definition at line 28 of file xmldao.hpp.

Index